Nubeva Controller Documentation

Release 1.0

Product

Apr 28, 2021

NUBEVA CONTROLLER INSTALLATION

1 Nubeva Controller Overview 1

2 Prerequisites 3

2.1 AWS 3

2.2 VMWare 4

3 Installing a Nubeva Controller 5

3.1 AWS 5

3.2 VMWare 14

3.3 Controller Container Images 14

4 Support 17

CHAPTER

ONE

NUBEVA CONTROLLER OVERVIEW

Nubeva Controller runs in your VPC, as a self contained Kubernetes cluster which does not require Internet access. As the diagram below illustrates, All traffic, including `Sensors` and `Decryptors` communications with the `Nubeva Controller` (shown by the solid black lines) remain within your VPC. The early access version is available on AWS. Azure, GCP and VMWare configurations are coming soon.

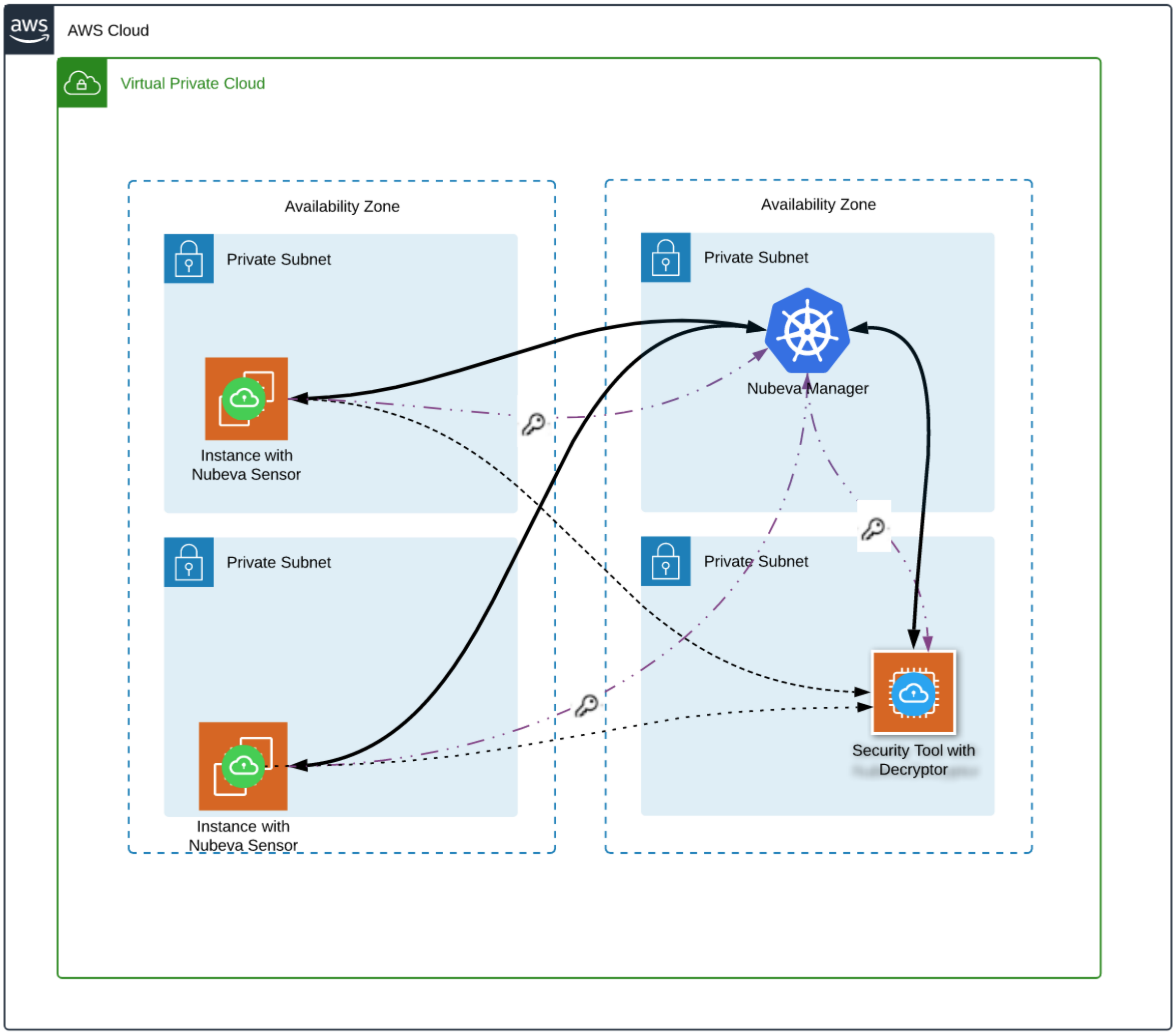


Figure 1: Architecture of a TLS Decryption Solution within a VPC

Discovered keys are stored securely in a key depot within Nubeva Controller's cluster. The purple dashed lines represent the paths of session keys from `sensors` to the key depot, and from the key depot to `decryptors`.

Note: Decryptors handle the synchronization of keys with packet flows, assuring that all the traffic received is matched with keys, and is fully decrypted.

`Sensors` and `Decryptors` register with the `Nubeva Controller` when they launch and obtain configuration updates. The Nubeva Controller maintains a local image repository for software updates. The root URL of the repository is configurable, allowing you to select other repositories within your cloud.

CHAPTER

TWO

PREREQUISITES

2.1 AWS

You need an AWS account with programmatic access to the following services:

- VPC read and write
- S3 read and write
- Route53 read and write
- EC2 Instances read and write
- Elastic Load Balancing read

The installation package uses a `Terraform` script which runs `Kops` to set up and configure the Nubeva Controller cluster. The following packages are required in order to install and run a Nubeva Controller:

2.1.1 AWS CLI

The aws cli is required by portions of the terraform script to query the AWS API for resource codes. Please see the .

2.1.2 Docker

Please see the .

2.1.3 Terraform

For information about Terraform please visit the . Please follow the to choose the package for your system.

NOTE: Minimum supported terraform version of is **0.12.0**

2.1.4 eksctl

Please refer to for installation instructions.

Note: EKS is supported in the following regions: us-west-2, us-east-1, us-east-2, ca-central-1, eu-west-1, eu-west-2, eu-west-3, eu-north-1, eu-central-1, ap-northeast-1, ap-northeast-2, ap-southeast-1, ap-southeast-2, ap-south-1, ap-east-1, me-south-1, sa-east-1

2.1.5 jq

Please refer to for installation instructions.

2.1.6 kubectl

`kubectl` is the Kubernetes command line tool. It allows you to run commands against Kubernetes clusters. You can use `kubectl` to deploy applications, inspect and manage cluster resources, and view logs. To install `kubectl` please follow the .

2.2 VMWare

VMware installations require VSphere version 6.7

- Linux environment with Internet access.
- Internal DNS name-server.
- Kubernetes cluster version 1.16 with all nodes configured to use the internal DNS name-server. The cluster should have four nodes, each with at least 4 free cores and 16GB of memory.
- All VMs MUST support UUID
- Kubernetes cluster must have installed in order to allow Kubernetes to create persistent volumes on ESXi provisioned storage.
- Kubernetes cluster must have installed.
- Outbound Internet access for K8S cluster nodes (air-gap installation is supported with additional configuration)

CHAPTER

THREE

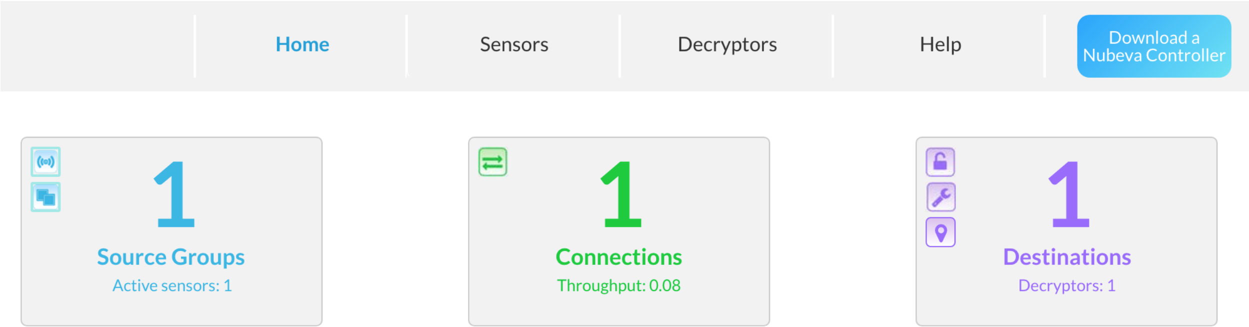
INSTALLING A NUBEVA CONTROLLER

3.1 AWS

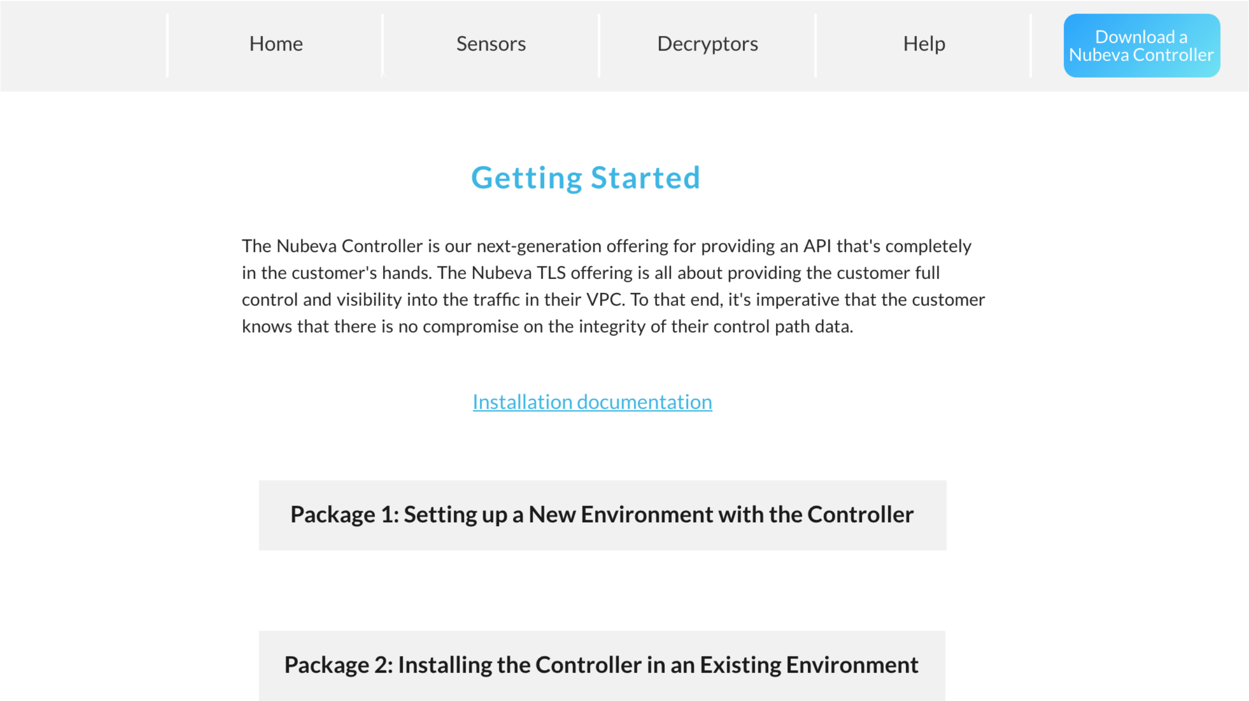
3.1.1 Download the Nubeva Controller Package

To download a Nubeva Controller installation package you need to have an account on the .

Note: Nubeva Controller package downloads are currently accessible only to designated users. Authorized users will see a download button on the main menu of the console.



Clicking the button navigates to a page which allows you to download the package you need. Packages are created when the page is loaded. This can take a few seconds. When the packages are ready you will see the interface depicted below:



Getting Started

The Nubeva Controller is our next-generation offering for providing an API that's completely in the customer's hands. The Nubeva TLS offering is all about providing the customer full control and visibility into the traffic in their VPC. To that end, it's imperative that the customer knows that there is no compromise on the integrity of their control path data.

Installation documentation

Package 1: Setting up a New Environment with the Controller
Package 2: Installing the Controller in an Existing Environment

3.1.2 Set up a Nubeva Controller in a New AWS Environment

This package contains a Terraform script that launches a fully configured environment in AWS. The package sets up a Kubernetes cluster running the Nubeva Controller, a bastion host, a source instance where you can launch a sensor, and a destination instance on which you can launch a decryptor.

Please make sure you have your AWS profile configured with the account you want to use. Then in the package you just downloaded, go to the `nukates/terraform_creation` directory. Create a copy of the `default.tfvars.template` file and name it `terraform.tfvars`. Modify the variables inside `terraform.tfvars` based on the following guidelines:

- **cluster_name**: The name you want to use for your cluster. e.g. `nubevaeks`
- **aws_region**: The AWS region that you want to launch in. e.g. `us-east-1`
- **ssh_public_key**: The *public* SSH key configured for use with your AWS account e.g. `~/.ssh/id_rsa.pub`
- **ssh_private_key**: The *private* SSH key configured for use with your AWS account. e.g. `~/.ssh/id_rsa`
- **nuconfig_file_location**: `../../tmp` (you can use the default value)

Run the following command in the nukates/terraform_creation directory:

```
terraform init && terraform apply -var-file=terraform.tfvars
```

You will be prompted to enter "yes" to confirm the changes. The script will then take ~25 minutes to complete. Once complete, the IP addresses and SSH strings to use to connect to the machines will be output. If this information is lost, simply go back to the folder nukates/terraform_creation and run the command:

```
terraform output
```

You now have a bastion passthrough machine, a machine running a sensor and another machine running the decryptor. Skip ahead to the Testing section to confirm that the `Nubeva Controller` is working.

If at any time you want to delete your environment, run the following command in the nukates/terraform_creation directory:

```
terraform destroy -var-file=terraform.tfvars
```

You will be prompted to enter “yes” to confirm the changes. Once confirmed, please wait until the script has finished deleting before closing the terminal.

3.1.3 Adding a Nubeva Controller to an Existing Environment

Note: In order to add a Nubeva Controller to an existing environment please make sure that you have the following resources and services available:

- At least 4 free cores on each instance.
- A total of 12 cores available across all nodes.
- A storage class provisioned named “standard”.

The package to add to an existing enviroment holds the yaml files for the Nubeva Controller Operator. All you will need to do is apply the operator files to your cluster and you will have the Nubeva Controller up and running.

Start by executing the following commands inside the package you downloaded:

```
cd nukates/operator/k8s-config

# rbac
kubectl apply -f service_account.yaml
kubectl apply -f role.yaml
kubectl apply -f role_binding.yaml
kubectl apply -f cluster_role_binding.yaml

# nubeva custom resource definitions
kubectl apply -f nukates.nuos.io_nukates_crd.yaml
kubectl apply -f nukates.nuos.io_nuconfigs_crd.yaml

# deploy the operator
kubectl apply -f operator.yaml

# additional custom resource definitions
kubectl apply -f NuComplete.yaml

# deploys all our microservices
kubectl apply -f nukates.yaml

cd ../../../tmp

# apply the Nubeva config
kubectl apply -f nuconfig.yaml
```

Then make sure to do the following to your environment so that the Nubeva Controller works properly and you have access to it:

- Add .nuos.io to your DNS resolver. On AWS add for .nuos.io. to your VPC
- Create a record in your personal DNS resolver for `k.nuos.io` resolving to the Load Balancer of the ambassador Kubernetes service:

```
kubectl get svc ambassador -o jsonpath="{.status.loadBalancer.ingress[0].hostname}"
```

- On AWS: create a record in the private hosted zone for k.nuos.io pointing to the ELB of the proxy service.

To test the Nubeva Controller to make sure it is working, you will need access to your environment and have two instances running to act as a source and destination.

3.1.4 Testing the Nubeva Controller

You now have a bastion pass-through machine, a machine running a sensor and another machine running the decryptor.

To confirm everything is working, we will begin by accessing the Nubeva dashboard.

Please add an entry from 0.0.0.0 to k.nuos.io to your host database. Open up your hosts file (*sudo vim /etc/hosts*) and add k.nuos.io as a mapping for 0.0.0.0.

Copy the bastion_passthrough command and open up a new terminal window. Set the SSH_KEY environment variable and then enter the bastion_passthrough command. Open up a browser and go to https://k.nuos.io:8080/grafana.

Welcome to the Nubeva Controller login page! Please proceed by logging in with username "admin" and password "admin". Add a new user by going to Server Admin > Users > New user. Email is optional. Logout and log back in with that user.

If you are following from package 2, let's learn how to launch the sensor and decryptor. Upon logging in, you will land on the New page. Here, you will find instructions on how to launch the sensor and decryptor on multiple environments. Please follow the instructions for the environment you are running on and launch a sensor and decryptor on your source and destination instances, respectively, before continuing.

Under Packet Mirroring, click on the Configure page. If the sensor on your source instance is running properly, you should see Active sources showing 1. Please create a source group for your running sensor, a destination group for your running decryptor, and create a connection between your source and destination group. Make sure to use the IP address of the instance running your decryptor for the destination group. It can be found in the terraform output.

At this point, you just configured packet mirroring between a source and destination instance in your environment. Let's see if we can get some decrypted traffic!

Go ahead and open up two more terminal windows. In each, please set the SSH_KEY environment variable and then use the two remaining SSH commands to connect to your source and destination instances.

On the destination instance, run the following command:

```
sudo tcpdump -Ani nurx0 tcp port 80
```

This will show us any encrypted traffic received as decrypted traffic. The nurx0 port is made available specially because we are running the decryptor.

On the source instance, run the following command:

```
curl -k https://<BASTION_PRIVATE_IP>
```

Note: If you are following from package 2, use a private IP address that is not your source or destination machine.

Tip: If you see decrypted traffic then the `Nubeva Controller` running correctly, `sensors` can discover TLS session keys and `decyptors` receive mirrored packets and decrypt them successfully.

3.1.5 Controller Services and Pods

The following resources should be running in a cluster:

```
API
Pods: api (3), putlog-api (3)

Binary Repository
Pods: bin-repo (1)

Configuration Management
Pods: nuconfig (1)

Configuration Metrics
Pods: config-metrics (3)

Grafana
Pods: grafana (1)

Healthchecks
Pods: kuberhealthy (these are periodic short lived so the number of pods will vary)

Logging
Pods: fluentd (4), rsyslog-server-set (1)

Message Queue System
Pods: create-numq-topics (1), kafka (3), numq-controller (1), zoo (3)

MongoDB
Pods: mongodb (3)

Nubeva Operator
Pods: kates (1)

Nubeva Registry
Pods: docker-registry (1)

Postgres
Pods: acid-minimal-cluster (2), postgres-operator (1)

Prometheus
Pods: alertmanager (3), kube-state-metrics (1), node-exporter (4), prometheus-adapter␣
↪(1), prometheus-k8s (2), prometheus-operator (1)

Routing/LoadBalancing
Pods: ambassador (1)

TSDB
Pods: etcd (3), m3db-operator (1), persistent-cluster-rep (1)
```

Detailed resource information can be retrieved using `kubectl`:

```
#Deployments
kubectl get deployments

NAME                   READY   UP-TO-DATE   AVAILABLE   AGE
ambassador             1/1     1            1           3h
api                    3/3     3            3           176m
```

(continues on next page)

(continued from previous page)

```
bin-repo              1/1     1            1           176m
config-metrics        3/3     3            3           176m
docker-registry       1/1     1            1           176m
grafana               1/1     1            1           177m
kates                 1/1     1            1           3h
kube-state-metrics    1/1     1            1           177m
kuberhealthy          2/2     2            2           177m
nuconfig-pod          1/1     1            1           3h
numq-controller       1/1     1            1           179m
postgres-operator     1/1     1            1           179m
prometheus-adapter    1/1     1            1           177m
prometheus-operator   1/1     1            1           177m
putlog-api            3/3     3            3           179m

#Services
kubectl get services

NAME                               TYPE           CLUSTER-IP       EXTERNAL-IP        ␣
↪                                                                PORT(S)              ␣
↪                                             AGE
acid-minimal-cluster               ClusterIP      172.20.234.197   <none>             ␣
↪                                                                5432/TCP             ␣
↪                                             3h4m
acid-minimal-cluster-config        ClusterIP      None             <none>             ␣
↪                                                                <none>               ␣
↪                                             3h3m
acid-minimal-cluster-repl          ClusterIP      172.20.177.84    <none>             ␣
↪                                                                5432/TCP             ␣
↪                                             3h4m
alertmanager-main                  ClusterIP      172.20.45.31     <none>             ␣
↪                                                                9093/TCP             ␣
↪                                             3h2m
alertmanager-operated              ClusterIP      None             <none>             ␣
↪                                                                9093/TCP,9094/TCP,   ␣
↪9094/UDP                                        3h2m
ambassador                         LoadBalancer   172.20.33.19     internal-
↪a08171ba5657311eaa54f0e3ee5d4f42-165627743.us-east-1.elb.amazonaws.com   80:30316/
↪TCP,443:31489/TCP                                        3h5m
ambassador-admin                   NodePort       172.20.136.21    <none>             ␣
↪                                                                8877:32603/TCP       ␣
↪                                             3h5m
api                                ClusterIP      172.20.171.35    <none>             ␣
↪                                                                5000/TCP             ␣
↪                                             3h4m
bin-repo                           ClusterIP      172.20.174.226   <none>             ␣
↪                                                                80/TCP               ␣
↪                                             3h2m
broker                             ClusterIP      None             <none>             ␣
↪                                                                9092/TCP             ␣
↪                                             3h4m
config-metrics                     ClusterIP      172.20.92.157    <none>             ␣
↪                                                                8000/TCP             ␣
↪                                             3h2m
docker-registry                    ClusterIP      172.20.209.109   <none>             ␣
↪                                                                80/TCP               ␣
↪                                             3h2m
etcd                               ClusterIP      None             <none>             ␣
↪                                                                2379/TCP,2380/TCP    ␣
↪                                             3h2m
```

(continues on next page)

(continued from previous page)

```
etcd-cluster                       ClusterIP     172.20.175.91    <none>             ␣
↪                                                                2379/UDP             ␣
↪                                             3h2m
grafana                            ClusterIP     172.20.192.166   <none>             ␣
↪                                                                3000/TCP             ␣
↪                                             3h2m
kates-metrics                      ClusterIP     172.20.185.8     <none>             ␣
↪                                                                8383/TCP,8686/TCP    ␣
↪                                             3h5m
kube-state-metrics                 ClusterIP     None             <none>             ␣
↪                                                                8443/TCP,9443/TCP    ␣
↪                                             3h2m
kuberhealthy                       ClusterIP     172.20.162.0     <none>             ␣
↪                                                                80/TCP               ␣
↪                                             3h2m
kubernetes                         ClusterIP     172.20.0.1       <none>             ␣
↪                                                                443/TCP              ␣
↪                                             5h50m
m3coordinator-persistent-cluster   ClusterIP     172.20.154.227   <none>             ␣
↪                                                                7201/TCP,7203/TCP    ␣
↪                                             3h
m3dbnode-persistent-cluster        ClusterIP     None             <none>             ␣
↪                                                                9000/TCP,9001/TCP,
↪9002/TCP,9003/TCP,9004/TCP,7201/TCP,7203/TCP   3h
mongodb                            ClusterIP     None             <none>             ␣
↪                                                                27017/TCP            ␣
↪                                             3h5m
node-exporter                      ClusterIP     None             <none>             ␣
↪                                                                9100/TCP             ␣
↪                                             3h2m
numq                               ClusterIP     172.20.23.198    <none>             ␣
↪                                                                9092/TCP             ␣
↪                                             3h4m
prometheus-adapter                 ClusterIP     172.20.162.75    <none>             ␣
↪                                                                443/TCP              ␣
↪                                             3h2m
prometheus-k8s                     ClusterIP     172.20.91.220    <none>             ␣
↪                                                                9090/TCP             ␣
↪                                             3h2m
prometheus-operated                ClusterIP     None             <none>             ␣
↪                                                                9090/TCP             ␣
↪                                             3h2m
prometheus-operator                ClusterIP     None             <none>             ␣
↪                                                                8080/TCP             ␣
↪                                             3h2m
putlog-api                         ClusterIP     172.20.5.212     <none>             ␣
↪                                                                5000/TCP             ␣
↪                                             3h4m
rsyslog-server                     ClusterIP     172.20.6.20      <none>             ␣
↪                                                                514/UDP,514/TCP,6000/
↪TCP                                          3h5m
rsyslog-server-headless            ClusterIP     None             <none>             ␣
↪                                                                514/UDP,514/TCP,6000/
↪TCP                                          3h5m
zoo                                ClusterIP     None             <none>             ␣
↪                                                                2888/TCP,3888/TCP    ␣
↪                                             3h5m
```

(continues on next page)

(continued from previous page)

```
zookeeper                          ClusterIP      172.20.26.239     <none>            ␣
↪                                                                 2181/TCP            ␣
↪                                             3h5m

#Pods
kubectl get pods

NAME                                                READY   STATUS      RESTARTS   AGE
acid-minimal-cluster-0                              1/1     Running     0          ␣
↪3h7m
acid-minimal-cluster-1                              1/1     Running     0          ␣
↪3h6m
alertmanager-main-0                                 2/2     Running     0          ␣
↪3h5m
alertmanager-main-1                                 2/2     Running     0          ␣
↪3h5m
alertmanager-main-2                                 2/2     Running     0          ␣
↪3h5m
ambassador-6bf6cb6c48-lm9x6                         1/1     Running     0          ␣
↪3h8m
api-67b4d8867d-h2wqq                                2/2     Running     0          ␣
↪3h4m
api-67b4d8867d-vlzhh                                2/2     Running     0          ␣
↪3h4m
api-67b4d8867d-vw84p                                2/2     Running     0          ␣
↪3h4m
bin-repo-54bf9dd944-pp28z                           1/1     Running     2          ␣
↪3h5m
config-metrics-64cd9644b8-gw7t4                     1/1     Running     0          ␣
↪3h5m
config-metrics-64cd9644b8-hbkjj                     1/1     Running     0          ␣
↪3h5m
config-metrics-64cd9644b8-hgv68                     1/1     Running     0          ␣
↪3h5m
create-numq-topics                                  0/1     Completed   3          ␣
↪3h7m
docker-registry-59ffdddb5b-v5fml                    1/1     Running     0          ␣
↪3h5m
etcd-0                                              1/1     Running     0          ␣
↪3h5m
etcd-1                                              1/1     Running     0          ␣
↪3h4m
etcd-2                                              1/1     Running     0          ␣
↪3h4m
fluentd-6q4k9                                       1/1     Running     0          ␣
↪3h8m
fluentd-9tzcd                                       1/1     Running     0          ␣
↪3h8m
fluentd-9zvcx                                       1/1     Running     0          ␣
↪3h8m
fluentd-htxgd                                       1/1     Running     0          ␣
↪3h8m
grafana-599f4b964c-q5cvv                            1/1     Running     0          ␣
↪3h5m
kafka-0                                             1/1     Running     1          ␣
↪3h7m
kafka-1                                             1/1     Running     1          ␣
↪3h7m
```

(continues on next page)

(continued from previous page)

```
kafka-2                                             1/1     Running     0         ␣
↪3h7m
kates-6765ccf96b-8nz7w                              1/1     Running     0         ␣
↪3h8m
kube-state-metrics-56b48cbdd5-x84dl                 3/3     Running     0         ␣
↪3h5m
kuberhealthy-74b6b599ff-ktdbw                       1/1     Running     0         ␣
↪3h5m
kuberhealthy-74b6b599ff-ktdbw-deployment            0/1     Completed   0          72s
kuberhealthy-74b6b599ff-ktdbw-dns-status-internal   0/1     Completed   0          12s
kuberhealthy-74b6b599ff-ktdbw-pod-status            0/1     Completed   0          12s
kuberhealthy-74b6b599ff-rfrj8                       1/1     Running     0         ␣
↪3h5m
m3db-operator-0                                     1/1     Running     0         ␣
↪3h5m
mongodb-0                                           3/3     Running     0         ␣
↪3h8m
mongodb-1                                           3/3     Running     0         ␣
↪3h7m
mongodb-2                                           3/3     Running     0         ␣
↪3h7m
node-exporter-kwbqv                                 2/2     Running     0         ␣
↪3h5m
node-exporter-tktkt                                 2/2     Running     0         ␣
↪3h5m
node-exporter-v9ctw                                 2/2     Running     0         ␣
↪3h5m
node-exporter-z7t6x                                 2/2     Running     0         ␣
↪3h5m
nuconfig-pod-6677cd57bd-fwprh                       1/1     Running     0         ␣
↪3h8m
numq-controller-86b6d76b87-9mpkg                    1/1     Running     0         ␣
↪3h7m
persistent-cluster-rep0-0                           1/1     Running     0         ␣
↪3h3m
postgres-operator-787bcd5ffc-t45ms                  1/1     Running     0         ␣
↪3h7m
prometheus-adapter-6f6558456f-vrqnq                 1/1     Running     0         ␣
↪3h5m
prometheus-k8s-0                                    3/3     Running     1         ␣
↪3h4m
prometheus-k8s-1                                    3/3     Running     1         ␣
↪3h4m
prometheus-operator-5785d84664-52j7n                1/1     Running     0         ␣
↪3h5m
putlog-api-589cf997c7-5dn2g                         1/1     Running     0         ␣
↪3h7m
putlog-api-589cf997c7-8ss5x                         1/1     Running     0         ␣
↪3h7m
putlog-api-589cf997c7-pmpc5                         1/1     Running     0         ␣
↪3h7m
rsyslog-server-set-0                                1/1     Running     0         ␣
↪3h8m
zoo-0                                               1/1     Running     0         ␣
↪3h8m
zoo-1                                               1/1     Running     0         ␣
↪3h8m
```

(continues on next page)

(continued from previous page)

```
zoo-2                                                        1/1     Running     0          ␣
↪3h8m
```

3.2 VMWare

1. Download the second package for an existing environment from (see *Download the Nubeva Controller Package* above).
2. Copy the package archive into a host inside your VMware environment.
3. Unzip the package, go to the `/operator/k8s-config/` directory.
4. Locate `nukates.yaml` file and edit the cloud on line 16 cloud: `vmware`.
5. (OPTIONAL) If you do NOT have VMware NSX-T associated with your cluster and proxy resource is not available, update line 65 to be: `hostNetwork:  true`.
6. Run `deploy.sh`
7. Run

```
kubectl apply -f ../../../tmp/nuconfig.yaml
```

8.1 (OPTION A) If you have specified hostNetwork: true in step 5 above, watch the deployment and once the ambassador pod is in READY state, run the following command to extract the IP address of the Controller internal load balancer:

```
kkubectl get pods -l service=ambassador --output=jsonpath={.items[0].status.podIP}
```

If this command does not return anything, use

```
kubectl describe pods -l service=ambassador
```

and search the output for the node IP address the pod is attached to. That will be your DNS entry for k.nuos.io

8.2 (OPTION B) If you have NSX-T support, then wait run the following command and copy External IP field of the ambassador service:

9. Update your DNS server to resolve `k.nuos.io` hostname to the IP address noted in the previous step. Confirm that your nodes can resolve the address by running

Wait until the deployment is complete, it usually takes around 7 minutes. You will stop seeing new pods being created inside the default namespace. Once the deployment is complete, your Controller is ready to be used.

3.3 Controller Container Images

Below is the list of container images run by the controller:

```
cvallance/mongo-k8s-sidecar:latest
fluent/fluentd-kubernetes-daemonset:v1-debian-syslog
mongo:latest
nubeva/bin-repo
nubeva/muxer
nubeva/nuagent
```

(continues on next page)

(continued from previous page)

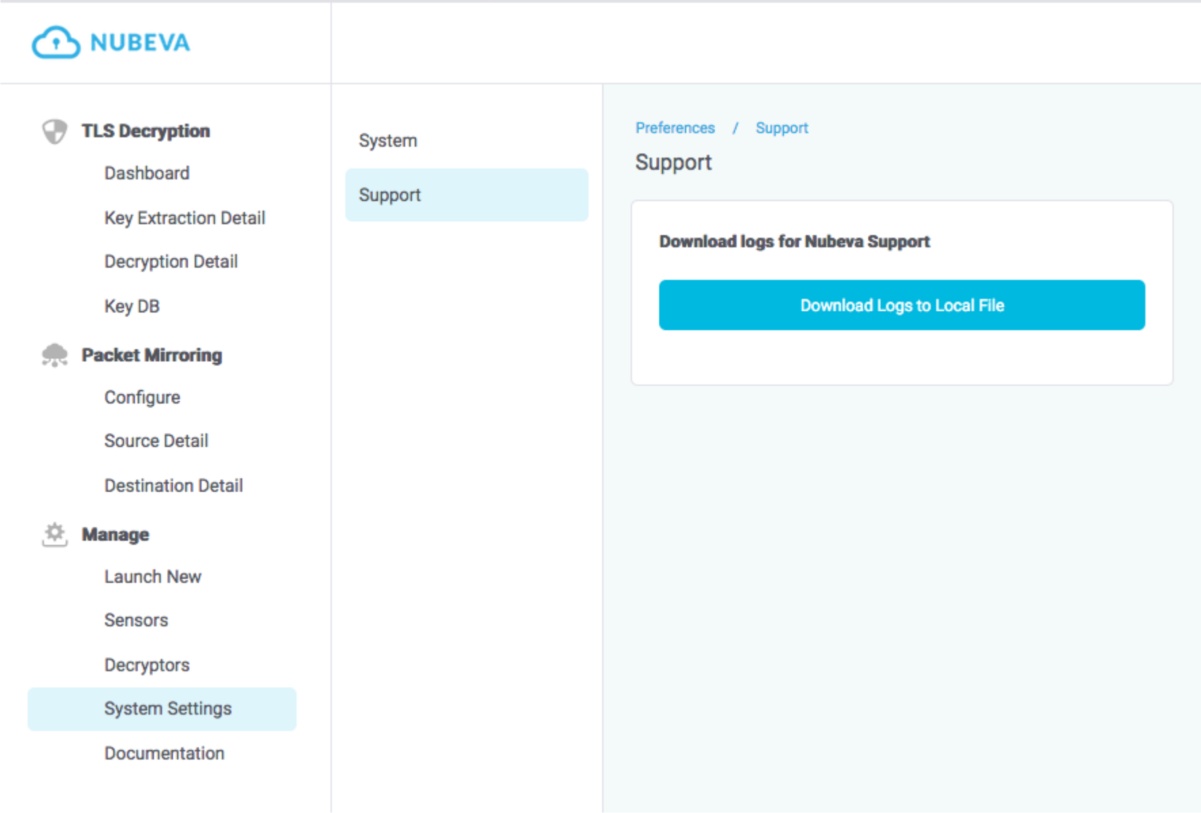
```
nubeva/nuana
nubeva/nukates:5b4814c86779
nubeva/nukatesapig
nubeva/nukatescfg
nubeva/nukatescfg-metrics
nubeva/nukatesregistry
nubeva/numq
nubeva/numq-controller
nubeva/numq-topics
nubeva/nurx
nubeva/postgres-operator
nubeva/putlogapig
nubeva/syslog
quay.io/comcast/deployment-check:1.0.3
quay.io/comcast/dns-status-check:1.0.0
quay.io/comcast/http-check:1.0.0
quay.io/comcast/kuberhealthy:2.0.0
quay.io/comcast/pod-restarts-check:2.0.0
quay.io/comcast/pod-status-check:1.1.0
quay.io/coreos/configmap-reload:v0.0.1
quay.io/coreos/etcd:v3.3.3
quay.io/coreos/k8s-prometheus-adapter-amd64:v0.4.1
quay.io/coreos/kube-rbac-proxy:v0.4.1
quay.io/coreos/kube-rbac-proxy:v0.4.1
quay.io/coreos/kube-rbac-proxy:v0.4.1
quay.io/coreos/kube-state-metrics:v1.7.2
quay.io/coreos/prometheus-config-reloader:v0.33.0
quay.io/coreos/prometheus-operator:v0.33.0
quay.io/datawire/ambassador:1.2.2
quay.io/m3db/m3db-operator:v0.2.0
quay.io/m3db/m3dbnode:v0.14.2
quay.io/prometheus/alertmanager:v0.20.0
quay.io/prometheus/node-exporter:v0.18.1
quay.io/prometheus/prometheus:v2.11.0
registry.opensource.zalan.do/acid/spilo-11:1.6-p1
solsson/kafka-initutils:latest
solsson/kafka-initutils:latest
solsson/kafka-initutils:latest
solsson/kafka:2.3.0
solsson/kafka:2.3.0
ssheehy/mongodb-exporter:latest
```

CHAPTER

FOUR

SUPPORT

You can submit support tickets from the `Help` page of your account. Please download logs from your cluster and attach them to your ticket. Logs can be downloaded from the `Support` screen depicted below:



In case you cannot access the cluster from your browser please try to download the logs using the following steps:

```
# 1. POST to '/api/logs/prepare' will start compressing the logs
curl -X POST https://k.nuos.io:8080/api/logs/prepare

# 2. GET to '/api/logs/prepare' will return a json with the status.
# This will be repeatedly hit until the logs have been compressed.
# If the logs have not been compressed within 5 minutes, it will timeout.
curl -X GET https://k.nuos.io:8080/api/logs/prepare

# 3. GET to '/api/logs/fetch' will serve the logs as a zipped file
curl -X GET https://k.nuos.io:8080/api/logs/fetch
```

If this fails, it could be that the cluster is not accessible through its DNS name. Please download the logs using the following steps:

```
#If the cluster is not accessible by its DNS name, you can try to download
#the logs using port forwarding.

####################
# Terminal shell 1 #
####################

#Enter the following commands to setup port forwarding

APIPOD=$(kubectl get po | grep '^api' | head -n1 | cut -d " " -f1)

kubectl port-forward $APIPOD 5000:5000 --request-timeout=0

####################
# Terminal shell 2 #
####################

# Prepare the logs by starting to compress. Compression can take max of 5 minutes
# depending on how big your logs are

curl -X POST http://localhost:5000/api/logs/prepare

#This will return the following output (the timestamps will vary based on your␣
↪cluster)
{
        "CompressionLastCompleted": "Mon, 01 Jan 1 00:00:00 GMT",
        "CompressionStart": "Tue, 25 Feb 2020 18:08:23 GMT",
        "IsCompressing": true
}

# Please wait until the logs have been fully compressed. You can check the status by␣
↪running
# the following GET command.

curl -X GET http://localhost:5000/api/logs/prepare

# You can tell by the "CompressionLastCompleted" value being a
# data that is after the "CompressionStart" value

{
        "CompressionLastCompleted": "Tue, 25 Feb 2020 18:09:32 GMT",
        "CompressionStart": "Tue, 25 Feb 2020 18:08:23 GMT",
        "IsCompressing": false
}

# To get the logs enter the following command (you can choose any output file name).

curl -X GET http://localhost:5000/api/logs/fetch > ~/Desktop/logs.zip
```