Federator.ai Release 4.2
Installation Guide

Content

Overview 2

Federator.ai 2

Requirements and Recommended Resource Configuration 3

Platform 3

Federator.ai Resource Requirements 3

Federator.ai Version 3

Recommended Prometheus 3

Persistent Volumes 3

Kafka 3

Federator.ai Installation and Configuration 4

Pre-installation Check List 4

Installation 6

Configuration 9

Manage Federator.ai License Keycode 11

Apply Keycode 11

Delete Keycode 11

Activate Keycode 12

Appendix 14

Federator.ai Endpoints in OpenShift 14

Federator.ai Endpoints in Kubernetes 14

Overview

Federator.ai

ProphetStor Federator.ai is an AI-based solution that helps enterprise manage, optimize, auto-scale resources for any applications on OpenShift/Kubernetes. Using advanced machine learning algorithms to predict application workload, Federator.ai scales the right amount of resources at the right time for optimized application performance.

Federator.ai for OpenShift/Kubernetes delivers the following key features:

AI-based workload prediction: Federator.ai applies multiple analytics tools, such as machine learning and signal processing, to predict containerized application and node resource usage as the basis for pod resource recommendations. Federator.ai supports both physical and virtual CPUs and memories.

Application-aware recommendation execution: The application resource demand determines the number and size of pods. Federator.ai utilizes resource usage prediction based on workload patterns to recommend the right pod sizes.

Policy-driven planning of CPU and memory: Federator.ai provides cluster-wide CPU and memory allocation for different types of applications according to the policy specified by users.

Enterprise-ready: Federator.ai is designed to work with any OpenShift/Kubernetes environment. Federator.ai provides application lifecycle management based on the Operator Framework and works seamlessly with standard OpenShift/Kubernetes.

Easy installation: Installing Federator.ai is easy as it works as an Operator on OpenShift/Kubernetes.

Continuous recommendations for optimal resource planning: Federator.ai continuously generates recommendations and learns with more accurate prediction when more metrics data is collected.

Requirements and Recommended Resource Configuration

Platform

- OpenShift : 3.11/4.3/4.4/4.5
- Kubernetes : 1.11 ~ 1.17.x

Federator.ai Resource Requirements

- Total Resource Requirments
  - 4 CPU cores
  - 4 GB Memory
  - StorageClass: 450GB (require ReadWriteMany access mode)

- Resource requirements for AI Engine
  - There must be at least one worker node with at least 2 CPU cores and 1 GB memory available
  - The 2 CPU cores and 1 GB memory are included in the total 4 CPU cores and 4 GB memory requirements

Federator.ai Version

- Version: Release 4.2
- 30 days trial license

Recommended Prometheus

- OpenShift
  - Default installed Prometheus
- Kubernetes
  - Prometheus operator helm chart version: 8.5.11
  - Prometheus operator version: 0.34.0
  - Prometheus server version: 2.13.1

Persistent Volumes

- The StorageClass that provides the persistent volumes must support RWX (read-write many) access mode.
- It is recommended to use persistent volumes instead of using ephemeral storage to store the data in the production environment.

Kafka

- For Federator.ai's application-aware Kafka consumer resource/performance optimaztion feature, the following version of Kafka is supported :

  Kafka operator version : Strimzi/kafka:0.17.0-kafka-2.4.0

Federator.ai Installation and Configuration

Pre-installation Check List

OpenShift/Kubernetes:

#	Check list Item	Requirement	Details
1	What is the Kubernetes version?	1.11~1.17.x	Use the command below to get Kubernetes version: **$ kubectl version** ... Server Version: version.Info{Major:"**1**", Minor:"17", GitVersion:"**v1.17.2**", GitCommit:"59603c6e503c87169aea6106f57b9f242f64df89", GitTreeState:"clean", BuildDate:"2020-01-18T23:22:30Z", GoVersion:"go1.13.5", Compiler:"gc", Platform:"linux/amd64"}
2	What is the Prometheus version? (for Kubernetes)	Recommended version - Prometheus operator helm chart version: 8.5.11 - Prometheus operator version: 0.34.0 - Prometheus server version: 2.13.1	Use the command below to get Prometheus version: **~# helm ls -A |grep -i prometheus** prometheus-adapter monitoring 1 2020-03-13 15:35:05.28963154 +0800 CST deployed prometheus-adapter-2.1.3 v0.6.0 prometheus-operator monitoring 1 2020-03-13 14:34:16.132479221 +0800 CST deployed prometheus-operator-8.12.1 0.37.0 **~# kubectl get deployment -A -o custom-columns=IMAGE:.spec.template.spec.containers[0].image |grep -i prometheus** directxman12/k8s-prometheus-adapter-amd64:v0.6.0 quay.io/coreos/prometheus-operator:v0.37.0
3	Does installation on Kubernetes cluster require private image repository?	If private image repository is required, the following information is needed during installation - Private image repository URL - Credential of the private image repository	Input the URL and credential when Federator.ai installation script asks for the information.
4	StorageClass and Persistent Volumes requirement	StorageClass supports ReadWriteMany access mode. Available storage size is larger than 450GB.	Minimum storage size for Federator.ai Release 4.2 is 450GB, including database, data, and logs.
5	OpenShift/Kubernetes cluster CPU/memory requirement	Minimum CPU/mem/storage: - CPU: 4 Cores - Memory: 4 GB - Storage Class Capacity: 450GB At least one worker node with - CPU: 2 Cores - Memory: 1GB	To be able to run AI Engine pod, there must be at least one worker node that has more than 2 CPU cores and 1 GB memory available. 2 CPU Cores and 1GB for AI Engine are included in the total 4 CPU Cores and 4GB memory requirements.
6	Is OpenShift/Kubernetes cluster allowed for NodePort configuration?	Federator.ai creates two NodePorts for GUI and REST API by default - REST API - https://<server>:31011 - GUI - https://<server>:31012	If NodePort is not allowed, answer 'N' when installation script prompts for creating NodePorts. Users need to expose Federator.ai GUI and REST API service manually.

7	Will there be a resource quota imposed for the namespace where Federator.ai is installed?	CPU/mem request quota should be more than minimum resource requirement - CPU: 4 Cores - Memory: 4 GB	The CPU/memory required for Federator.ai depends on the number of clusters and applications being monitored/managed. Suggestion for initial namespace quota is - CPU 8 cores - Memory 12G The quota could be adjusted if number of managed clusters/applications increases. Use the command to get namespace resource quota $ **kubectl get resourcequota --all-namespaces**
8	Does deployment must have resource request/limit specified?	By default, Federator.ai deployments do not specify resource requests/limits. It can be done by setting up an environment variable before installation starts.	To turn on resource request/limit settings for all Federator.ai deployments, manually export environment variable before running 'federatorai-launcher.sh' $ **export ENABLE_RESOURCE_REQUIREMENT=y** $ ./federatorai-launcher.sh

Installation

1. Log into OpenShift/Kubernetes cluster

2. Install the Federator.ai for OpenShift/Kubernetes by using following command

```
$ curl https://raw.githubusercontent.com/containers-ai/federatorai-
operator/master/deploy/federatorai-launcher.sh | bash
```

```
~# curl https://raw.githubusercontent.com/containers-ai/federatorai-
operator/master/deploy/federatorai-launcher.sh|bash
  00  3756  100  3756    0     0   5801      0 --:--:-- --:--:-- --:--:--  5814
Please input Federator.ai version tag (e.g., v4.2.755):  v4.2.759

Downloading scripts ...
Done
Do you want to use private repository URL? [default: n]:
Do you want to launch Federator.ai installation script? [default: y]:

Executing install.sh ...
Checking environment version...
...Passed
Enter the namespace you want to install Federator.ai [default: federatorai]:

----------------------------------------
tag_number = v4.2.759
install_namespace = federatorai
----------------------------------------
Is the above information correct? [default: y]:
Downloading file 00-namespace.yaml ...
Done
Downloading file 01-serviceaccount.yaml ...
Done
Downloading file 02-alamedaservice.crd.yaml ...
Done
Downloading file 03-federatorai-operator.deployment.yaml ...
Done
Downloading file 04-clusterrole.yaml ...
Done
Downloading file 05-clusterrolebinding.yaml ...
Done
Downloading file 06-role.yaml ...
Done
Downloading file 07-rolebinding.yaml ...
Done

Applying Federator.ai operator yaml files...
Applying 00-namespace.yaml...
namespace/federatorai created
Applying 01-serviceaccount.yaml...
serviceaccount/federatorai-operator created
Applying 02-alamedaservice.crd.yaml...
customresourcedefinition.apiextensions.k8s.io/alamedaservices.federatorai.containers.ai
created
Applying 03-federatorai-operator.deployment.yaml...
deployment.apps/federatorai-operator created
Applying 04-clusterrole.yaml...
clusterrole.rbac.authorization.k8s.io/federatorai-operator created
clusterrole.rbac.authorization.k8s.io/alameda-gc created
Applying 05-clusterrolebinding.yaml...
clusterrolebinding.rbac.authorization.k8s.io/federatorai-operator created
Applying 06-role.yaml...
role.rbac.authorization.k8s.io/federatorai-operator created
Applying 07-rolebinding.yaml...
rolebinding.rbac.authorization.k8s.io/federatorai-operator created
```

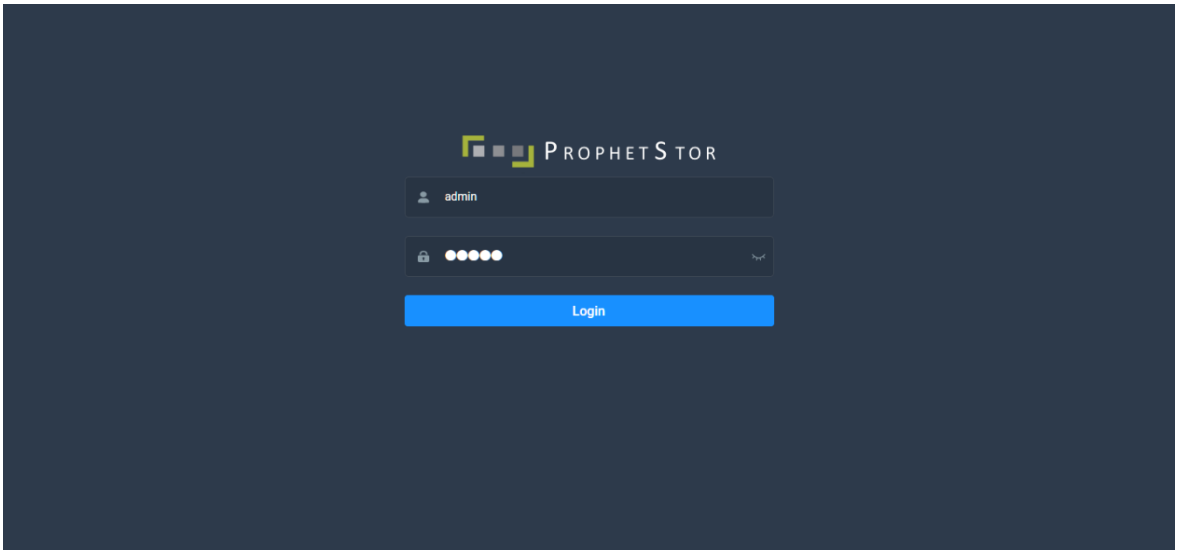
```
Checking pods...

All federatorai pods are ready.

Install Federator.ai operator v4.2.759 successfully

Downloading alameda CR sample files ...
Done
========================================
Do you want to enable execution? [default: y]: :
Enter the Prometheus service address
[default: https://prometheus-k8s-0.prometheus-operated.openshift-monitoring:9091]:
Which storage type you would like to use? ephemeral or persistent?
[default: ephemeral]:

----------------------------------------
install_namespace = federatorai
enable_execution = true
prometheus_address = https://prometheus-k8s-0.prometheus-operated.openshift-monitoring:9091
storage_type = ephemeral
----------------------------------------
Is the above information correct [default: y]:
Processing...
Waiting for datahub(v4.2.759) pod to be ready ...

datahub pod is running.


Checking pods...
Waiting pod admission-controller-69866f646-8ft9s in namespace federatorai to be ready.
phase: [Running]
Waiting for pods in namespace federatorai to be ready...
Waiting pod admission-controller-69866f646-8ft9s in namespace federatorai to be ready.
phase: [Running]
Waiting for pods in namespace federatorai to be ready...

All federatorai pods are ready.

========================================
You can now access GUI through https://federatorai-dashboard-frontend-
federatorai.apps.172.31.5.206.nip.io
Default login credential is admin/admin

#Federator.ai GUI access link and login credential

Also, you can start to apply alamedascaler CR for the namespace you would like to monitor.
Review administration guide for further details.
========================================

========================================
You can now access Federatorai REST API through https://federatorai-rest-
federatorai.apps.172.31.5.206.nip.io
Default login credential is admin/admin
The REST API online document can be found in https://federatorai-rest-
federatorai.apps.172.31.5.206.nip.io/apis/v1/swagger/index.html
========================================

Install Alameda v4.2.759 successfully

Downloaded YAML files are located under /tmp/install-op
```

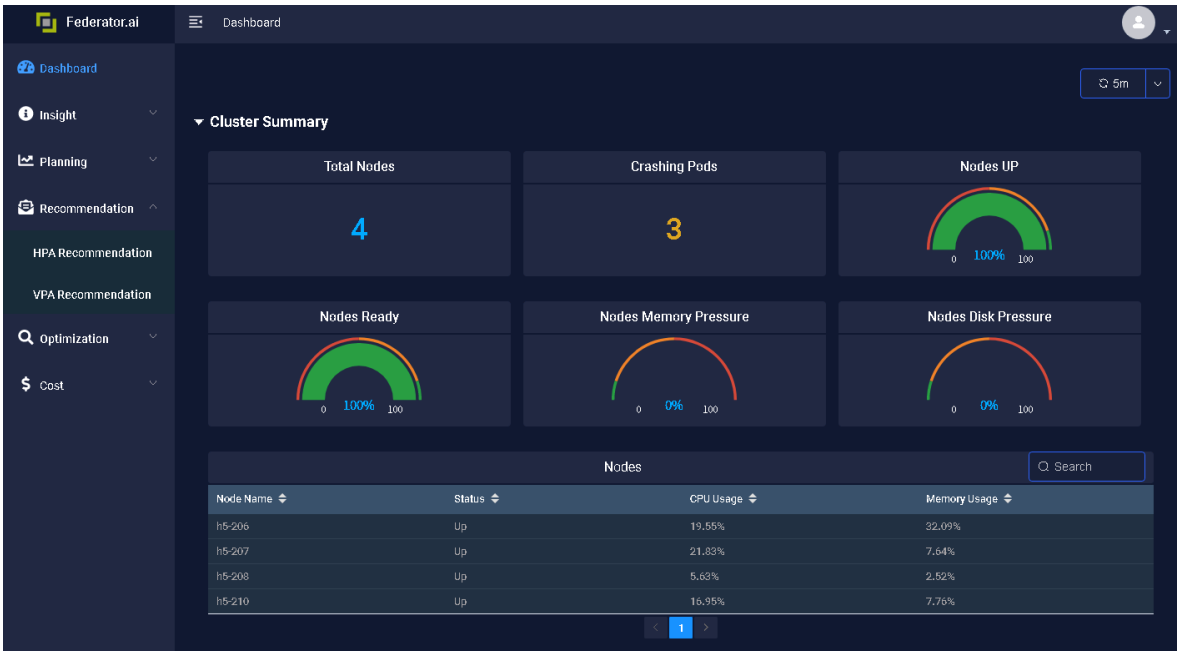
3. Verify Federator.ai pods are running properly

```
~# kubectl get pod -n federatorai
NAME                                        READY   STATUS    RESTARTS   AGE
alameda-ai-8465c78dbf-shtxx                 1/1     Running   0          13m
```

```
alameda-ai-dispatcher-5c77f7bc9f-dk8pc                1/1     Running   0          13m
alameda-analyzer-85b57c6d56-mnkdv                     1/1     Running   0          13m
alameda-datahub-5bc7846f5b-hxsxs                      1/1     Running   0          13m
alameda-executor-57f7ddfc56-29ncp                     1/1     Running   0          13m
alameda-influxdb-0                                    1/1     Running   0          13m
alameda-notifier-79f7c8d9c-8hkxb                      1/1     Running   0          13m
alameda-operator-5cc9fd7488-jzzrs                     1/1     Running   0          13m
alameda-rabbitmq-6c6744db65-bb7f4                     1/1     Running   0          13m
alameda-recommender-8588fbb6b4-dqx7n                  1/1     Running   0          13m
fedemeter-api-684c475dfd-gx9w9                        1/1     Running   0          13m
fedemeter-influxdb-0                                  1/1     Running   0          13m
federatorai-agent-7bdb666b56-mzp79                    1/1     Running   0          13m
federatorai-agent-app-75758b6c8-2td5w                 1/1     Running   0          13m
federatorai-dashboard-backend-7b44d4bccb-gz95p        1/1     Running   0          13m
federatorai-dashboard-frontend-bbf786447-k7p4k        1/1     Running   0          13m
federatorai-operator-c5b74799d-hnnwv                  1/1     Running   0          15m
federatorai-rest-78dfbb768f-bhjd4                     1/1     Running   0          13m
```

4. Log on Federator.ai GUI
   URL and login credential could be found in the output of Step 2.





Configuration

To monitor and manage application pods, Federator.ai defines an “AlamedaScaler” CRD for users to specify applications to be managed. By creating an “AlamedaScaler” CR and configuring with the application label and namespace, Federator.ai will discover and start managing the pods of the application in the namespace.

In this example, we create an “AlamedaScaler” CR for Federator.ai to manage an NGINX application, “nginx-1”, running in the namespace, “nginx-1”.

Currently Federator.ai supports the application which is a “**Deployment**”, “**DeploymentConfig**” or “**Statefulset**”.

Configuration

1. Get the label of the NGINX application

   Use ‘kubectl’ command line command to get the label. Please ensure you get the labels of the "controller" (deployment, deploymentConfig, statefulSet), not the labels of the "pod".

```
$ kubectl get deploy -n nginx-1 --show-labels
```

```
~# kubectl get deploy -n nginx-1 --show-labels
NAME      READY   UP-TO-DATE   AVAILABLE   AGE     LABELS
nginx-1   1/1     1            1           5h20m   app=nginx-1
```

2. Create “AlamedaScaler” CR and configure the application label and namespace

   2.1. Edit the sample “alamedascaler.yaml” with the application label and namespace. “alamedascaler.yaml” should be downloaded by “federatorai-launcher.sh” and saved in “/tmp/install-op/” directory. If you can’t find the sample “alamedascaler.yaml” in “/tmp/install-op/”, run “federatorai-launcher.sh” to download again.

```
$ vi /tmp/install-op/alamedascaler.yaml
```

“alamedascaler.yaml”:

```
apiVersion: autoscaling.containers.ai/v1alpha1
kind: AlamedaScaler
metadata:
  name: nginx-1              ## any name whatever you want, just for identification
  namespace: nginx-1         ## the same namespace with NGINX
spec:
  policy: stable             ## ‘stable’ or ‘compact’
  enableExecution: false     ## ‘true’/’false’, enable/disable autoscaling execution
  scalingTool:
    type: hpa                ## ‘hpa’/’vpa’
  selector:
    matchLabels:
      app: nginx-1           ## label of NGINX Deployment
```

2.2. Apply alamedascaler.yaml to create the “AlamedaScaler” CR

```
$ kubectl apply -f /tmp/install-op/alamedascaler.yaml
```

Read the “AlamedaScaler” CR by ‘kubectl get’ command to confirm Federator.ai is successfully managing the “nginx-1” application.

```
$ kubectl get alamedascaler -n nginx-1-o yaml
```

```
~# kubectl get alamedascaler -n nginx-1-o yaml
apiVersion: v1
items:
- apiVersion: autoscaling.containers.ai/v1alpha1
  kind: AlamedaScaler
  metadata:
    annotations:
  status:
    alamedaController:
      deployments:
        nginx-1/nginx-1:
          effective: true
          message: ""
          name: nginx-1
          namespace: nginx-1
          pods:
            nginx-1/nginx-1-8447b54549-9bj72:
              containers:
              - name: nginx-prepared
                resources: {}
              name: nginx-1-8447b54549-9bj72
              namespace: nginx-1
              uid: 15b01303-fa2f-440a-ad9d-8e54975e5855
          specReplicas: 1
          uid: 07624477-947f-46b9-a2ab-e7298bb07cca kind: List
metadata:
  resourceVersion: ""
  selfLink: ""
```

3. Log on Federator.ai GUI to get the detailed information of the application

Manage Federator.ai License Keycode

Federator.ai uses a keycode to control the license. A 30-day trial keycode is installed by default. It requires replacing with a valid keycode from ProphetStor to continue using Federator.ai after the 30-day trial.
The keycode operations are done by editing the “AlamedaService” CR which is created during Federator.ai installation.

Apply Keycode

1. Get “AlamedaService” CR name

```
~# kubectl get alamedaservice --all-namespaces
NAMESPACE       NAME                        EXECUTION     VERSION      PROMETHEUS
AGE
federatorai     my-alamedaservice     false         v4.2.759     https://prometheus-
k8s.openshift-monitoring:9091   45d
```

2. Edit the “AlamedaService” CR

```
~# kubectl edit alamedaservice my-alamedaservice -n <namespace>
```

3. Go to “keycode:” section, replace the value of “codeNumber” with the new keycode and then save the change

```
apiVersion: federatorai.containers.ai/v1alpha1
kind: AlamedaService
metadata:
  name: my-alamedaservice
......
spec:
  ......
  keycode:
    codeNumber: K4AMOC4TSDXXXXXXXXXXXXXXXXXXX
```

Delete Keycode

1. Get “AlamedaService” CR name

```
~# kubectl get alamedaservice --all-namespaces
NAMESPACE       NAME                        EXECUTION     VERSION      PROMETHEUS
AGE
federatorai     my-alamedaservice     false         v4.2.759     https://prometheus-
k8s.openshift-monitoring:9091   45d
```

2. Edit the “AlamedaService” CR

```
~# kubectl edit alamedaservice my-alamedaservice
```

3. Go to “keycode:” section, delete the keycode from “codeNumber” and then save the change

```
apiVersion: federatorai.containers.ai/v1alpha1
kind: AlamedaService
metadata:
  name: my-alamedaservice
......
spec:
  ......
  keycode:
    codeNumber:
```

Activate Keycode

1. Get “AlamedaService” CR name

```
~# kubectl get alamedaservice --all-namespaces
NAMESPACE       NAME                        EXECUTION    VERSION      PROMETHEUS
AGE
federatorai    my-alamedaservice    false        v4.2.759    https://prometheus-
k8s.openshift-monitoring:9091   45d
```

2. Edit the “AlamedaService” CR

```
~# kubectl edit alamedaservice my-alamedaservice
```

3. Go to “status.keycodeStatus:” section, copy the value of “registrationData” and email to register@prophetstor.com

```
apiVersion: federatorai.containers.ai/v1alpha1
kind: AlamedaService
metadata:
  name: my-alamedaservice
......
status:
  ......
  keycodeStatus:
    codeNumber: K4AMOC4TSDXXXXXXXXXXXXXXXXXXQ
    lastErrorMessage: ""
    registrationData: H4sICAavJl8C/2ZlZGFpLXJlZ2RhdGEudGd6AO3ad1DTZxjA8R9IoMoe
KkMhZSiKSAYgG0IwjBBkb4kRAgZJwwa3oAiKUPYoSB0E0YooKquADCGmgkiQCigqVoYgURCFiEJtr3
wegjWPwWLNYAt0UfY4hgsAYYrCGCxiBfQExUNC0SjUZoYTQmPYj2r+ciWazo/1vy76W+yNM/B2c3R3
uiEIKEsagUrZnnh3s6lyZ/YfrFk5cVpi86XqYU4E4XqnoFLDYPsOp1xeTw5iR365holofk8dRD7VQP
2prLF+uEPGkmGsIS7AxNoeT1cR2W6u7GekO3Lp/TEBGxrKoUXEP5TmlvF3RNqd6N2UoyPbrr+8Z8Zi
e9613bfzzvvHs+/zz3vXvfu/f5/6UxdKPokbGMQDo1kh6yOYgWjXx2mE8M9fX/mNgtBtg/+5O/Jg6P
...
JJhu2gMPKh7XEl16h4Ojfv5pHrafOCXvxB0zbTXkyjklVgoLsdVXGd1HDARd6sVWbcrBqLP3M2bDP9
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

4. Once ProphetStor received the activation request email and validated the “registrationData”, it returns the activation code, “signatureData”, via an email. Copy the “signatureData” from the email, fill in the “keycode. signatureData” field and save the change.

```
apiVersion: federatorai.containers.ai/v1alpha1
kind: AlamedaService
metadata:
  name: my-alamedaservice
......
spec:
  ......
  keycode:
    codeNumber: K4AMOC4TSDXXXXXXXXXXXXXXXXXXQ
    signatureData: F5nmus478ertgnldd430gvsef90gNYAt0UfY4hgsAYYrCGCxiBfQExUNC0S
KkMhZSiKSAYgG pi86XqYU4E4a3oAiKUPYoSB0E0YooKquADCGmgkiQCigYrCGCxergHwernREBo4E
wegjWPwWLNYAt0UfY4hgsAYYrCGCx6UxdKPokG0SjUZoYTQmPYj2r+ciWazo/1vy76W+yNM/B2c3R3
OYgWjXx2mE8M9fh3s6lyZ/YfrFk5cVpixdKPokbGMQDo1khRTNBOp1xeTw5iR365holofk8dRD7VQP
2prLF+uEPGkmGsIS7AxNoeT1cR2W6u7GekO3Lp/TEBGxrKoUXEP5TmlvF3RNqd6N2UoyPbrr+8Z8Zi
e9613bfzzvvHs+/2ZlZGFpLXJlZAO3ad1DTZxjA8R9IoxdKPokbGMQDo1kh6yOYg8M9fX/RwtBVerh
...
JBoerBTR445h4536g456UJdfsheryhryu6JwerJwerYjJKER5zQ6kZrFFhkr6sVWbcrBqLPregUh9
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

Appendix

Federator.ai Endpoints in OpenShift

In OpenShift, a service is exposed by creating a route. Use “oc get routes -n <namespace>” command to get the routes of Federator.ai services.

- Federator.ai GUI
  - E.g., https://federatorai-dashboard-frontend-federatorai.apps.<IP>.nip.io
  - Default credentials: admin/admin

- REST API URL
  - E.g., https://federatorai-rest-federatorai.apps.<IP>.nip.io

- REST API Documentation
  - E.g., https://federatorai-rest-federatorai.apps.<IP>.nip.io/apis/v1/swagger/index.html

Federator.ai Endpoints in Kubernetes

- Federator.ai GUI
  - https://<cluster_ip_address>:31012
  - Default credentials: admin/admin

- REST API URL
  - https://<cluster_ip_address>:31011

- REST API Documentation
  - https://<cluster_ip_address>: 31011/apis/v1/swagger/index.html