ProphetStor

Federator.ai for Datadog
Installation and Configuration Guide

Content

Overview

Federator.ai

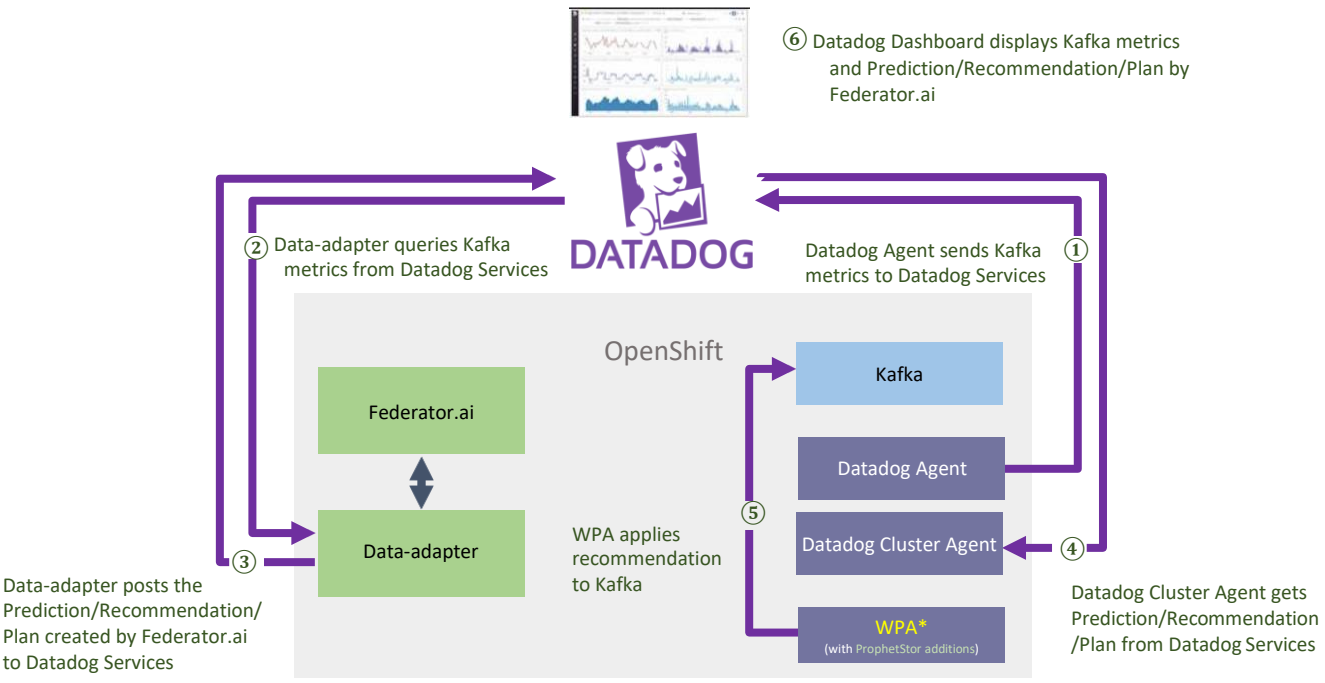
ProphetStor Federator.ai is an AI-based solution that helps enterprise manage, optimize, auto-scale resources for any applications on Kubernetes. Using advanced machine learning algorithms to predict application workload, Federator.ai scales the right amount of resources at the right time for optimized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

- AI-based workload prediction for Kafka or any applications
- Resource recommendation based on workload prediction, application, Kubernetes and other related metrics
- Automatic scaling of application containers through Datadog Watermark Pod Autoscaler (WPA)

Datadog Integration Workflows

Using Kafka as an example, the following diagram shows how Kafka metrics are used by Federator.ai to predict Kafka workload and to automatically scale Kafka consumer for better performance. Specifically,

- Datadog Agent sends Kafka metrics to Datadog Services
- Data-adapter queries Kafka metrics from Datadog Services and fowards to Federator.ai AI engine
- Data-adapter posts the Prediction/Recommendation/Plan created by Federator.ai to Datadog Services
- Datadog Cluster Agent gets Prediction/Recommendation/Plan from Datadog Services
- WPA applies plans to Kafka
- Datadog Dashboard displays Kafka metrics and Prediction/Recommendation/Plan by Federator.ai



Requirements and Recommended Configuration

Platform

- OpenShift : 3.11/4.3/4.4
- Kubernetes : 1.11 ~ 1.17

Cluster Configuration

- Master/Worker nodes
  - 8 vCPU
  - 32 GB Memory
  - 100GB disk

Federator.ai

- Version: v4.2 build 790
- 30 days license

Datadog

- Datadog Agent version : v7
- Datadog Cluster Agent version : 1.5.2
- Datadog Watermark Pod Autoscaler version : v0.1.0

Persistent Volume

- It is recommended to use persistent volume with RWM (read-write many) support instead of using ephemeral storage to store the data in the production environment.

Recommended Prometheus

- Prometheus operator helm chart : 8.5.11
- Prometheus operator version : 0.34.0
- Prometheus server version : 2.13.1

Supported Applications (Kafka)

- Kafka operator version : Strimzi/kafka:0.17.0-kafka-2.4.0
- kube-state-metrics : v1.5.0 (for OpenShift 3.11, Kubernetes 1.11 ~ 1.12)
  v1.9.5 (for OpenShift 4.3/4.4, Kubernetes 1.13 ~ 1.17)

Federator.ai Installation and Configuration

Requirements

- Install User Role: Cluster Admin

Installation

1. **Log in OpenShift/Kubernetes cluster**

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

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

```
~# curl https://raw.githubusercontent.com/containers-ai/federatorai-
operator/datadog/deploy/install.sh |bash
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left Speed
100 27760  100 27760    0     0  56194      0 --:--:-- --:--:-- --:----- 56194
Checking environment version...
...Passed
Please input Federator.ai Operator tag: datadog
Enter the namespace you want to install Federator.ai [default: federatorai]:

----------------------------------------
tag_number = datadog
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.containe
rs.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 datadog successfully
Do you want to patch the prometheus rule to meet the Federator.ai requirement?
Choose 'n' will abort the installation process. [default: y]: :

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(datadog) pod to be ready ...

datahub pod is running.


Checking pods...
Waiting pod alameda-ai-86876fd58-qb2zs in namespace federatorai to be ready.
phase: [Running]
Waiting for pods in namespace federatorai to be ready...
Waiting pod alameda-ai-86876fd58-qb2zs in namespace federatorai to be ready.
phase: [Running]
Waiting for pods in namespace federatorai to be ready...
Waiting pod alameda-ai-86876fd58-qb2zs in namespace federatorai to be ready.
phase: [Running]
Waiting for pods in namespace federatorai to be ready...
Waiting pod alameda-ai-86876fd58-qb2zs in namespace federatorai to be ready.
phase: [Running]
Waiting for pods in namespace federatorai to be ready...
Waiting pod alameda-ai-86876fd58-qb2zs in namespace federatorai to be ready.
phase: [Running]
Waiting for pods in namespace federatorai to be ready...

All federatorai pods are ready.

========================================
```

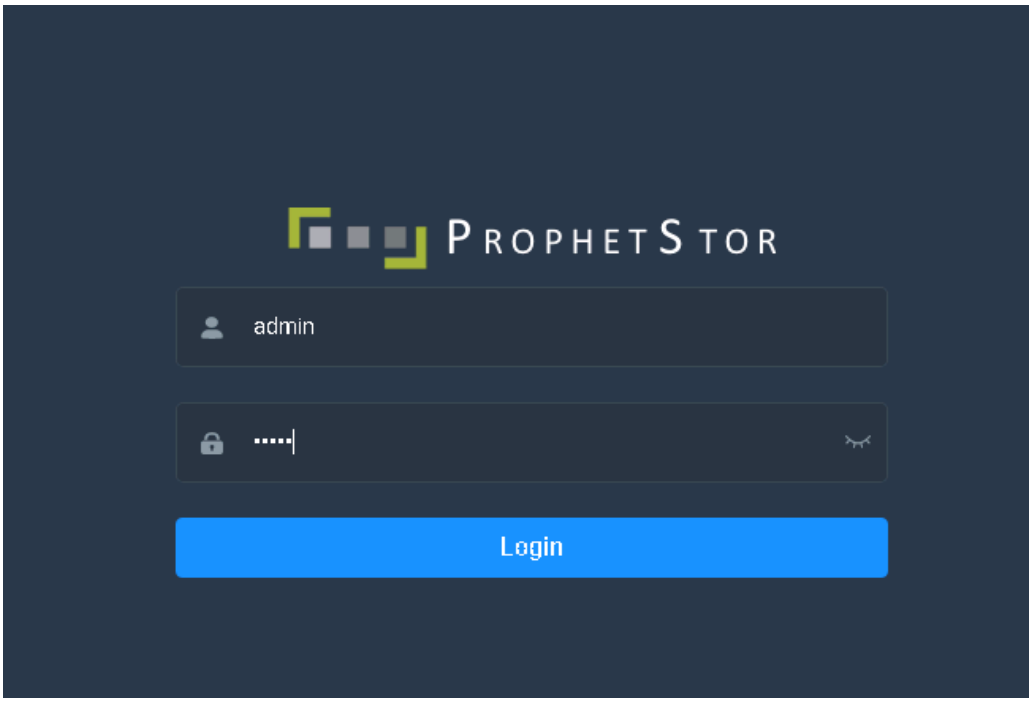
```
You can now access GUI through https://federatorai-dashboard-frontend-
federatorai.apps.jc-ocp4.172-31-17-84.nip.io
Default login credential is admin/admin

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.jc-ocp4.172-31-17-84.nip.io
Default login credential is admin/admin
The REST API online document can be found in https://federatorai-rest-
federatorai.apps.jc-ocp4.172-31-17-84.nip.io/apis/v1/swagger/index.html
========================================

Install Alameda datadog successfully

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

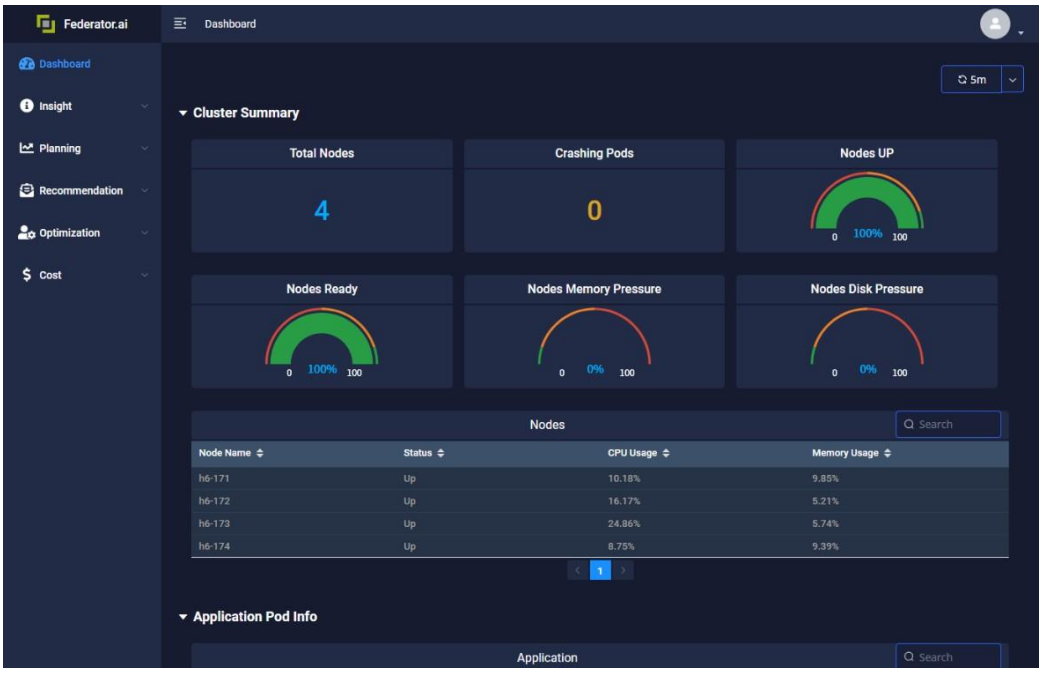
3. Verify Federator.ai pods are running properly

```
~# kubectl get pod -n federatorai
NAME                                          READY   STATUS    RESTARTS   AGE
alameda-ai-86876fd58-qb2zs                    1/1     Running   0          8m17s
alameda-ai-dispatcher-777969bb68-9xrsw        1/1     Running   0          8m17s
alameda-analyzer-78685d4f74-bjzsr             1/1     Running   0          8m17s
alameda-datahub-57dd6fcf6f-fr8sq              1/1     Running   0          8m17s
alameda-executor-574876f847-5k4js             1/1     Running   0          8m17s
alameda-grafana-766556f78d-8bpjv              1/1     Running   0          8m17s
alameda-influxdb-7c866f7687-ddddz             1/1     Running   0          8m17s
alameda-notifier-85d867468c-2klbg             1/1     Running   0          8m17s
alameda-operator-f99db8755-cgrrs              1/1     Running   2          8m17s
alameda-rabbitmq-658fb6c645-gmcrm             1/1     Running   0          8m17s
alameda-recommender-8547f66f46-4kv28          1/1     Running   0          8m17s
fedemeter-api-75c64b99bd-8xhkt                1/1     Running   0          8m17s
fedemeter-influxdb-0                          1/1     Running   0          8m17s
federatorai-agent-69b8f6565b-nlwt7            1/1     Running   0          8m17s
federatorai-dashboard-backend-79d494ff55-b762m  1/1   Running   0          8m17s
federatorai-dashboard-frontend-7fd5457957-c5mfn 1/1   Running   0          8m17s
federatorai-data-adapter-5cb559f64f-zhdl2     1/1     Running   4          8m17s
federatorai-operator-78984d7b9b-7gkt4         1/1     Running   0          9m22s
federatorai-rest-7c56c9bcd8-cws2f             1/1     Running   0          8m17s
```

4. Login to Federator.ai GUI

URL and login credential could be found in the output of Step 2.



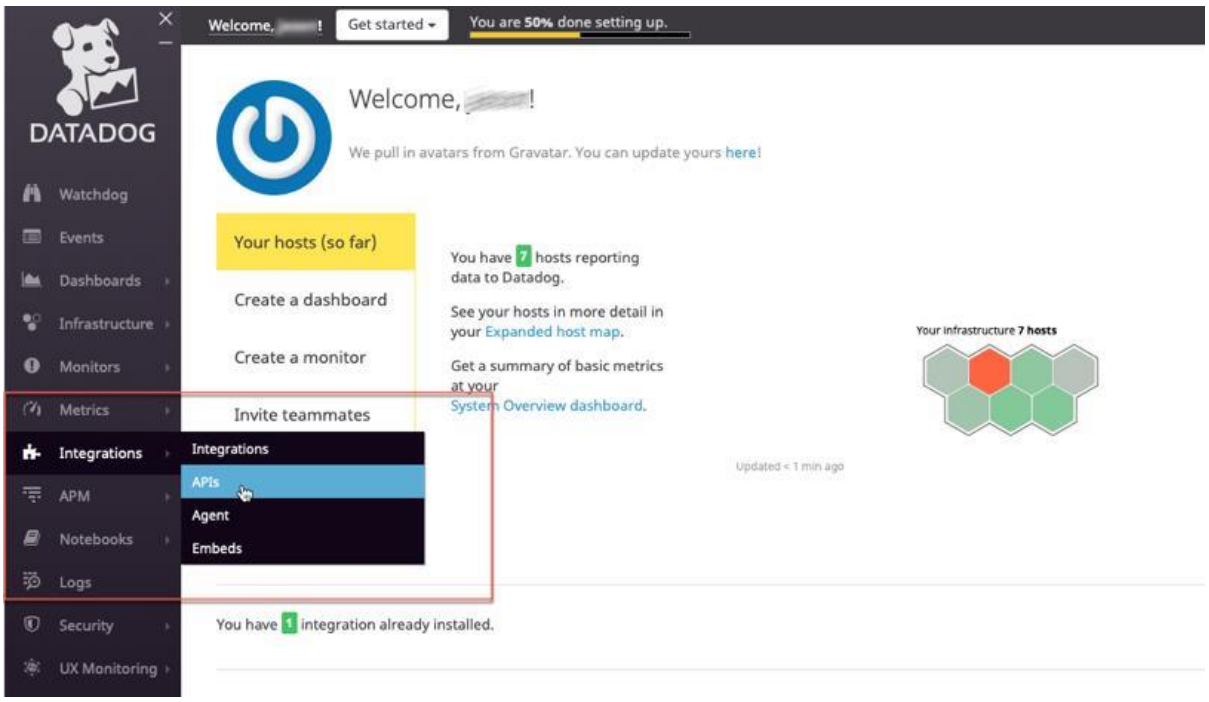


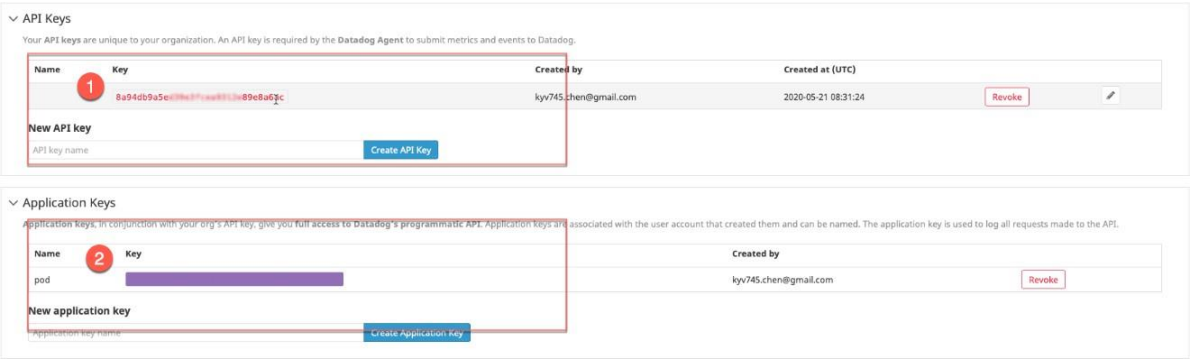
Configuration

1. A Datadog account is required for connecting and using Datadog Cloud Service. If you don't have an account, visit Datadog website and sign up for a free trial account.
   - https://www.datadoghq.com/

2. Log in Datadog Cloud Service with your account and get an API key and Application key for using Datadog API

- https://docs.datadoghq.com/account_management/api-app-keys/





Copy the API Key and Application Key for Federator.ai Data-Adapter configuration

3. Configure Federator.ai Data-Adapter

- Download Data-Adapter configuration script

```
$ curl https://raw.githubusercontent.com/containers-ai/federatorai-operator/datadog/deploy/federatorai-setup-for-datadog.sh -O
```

- Change execution permission

```
$ chomd +x federatorai-setup-for-datadog.sh
```

- Prepare .kubeconfig (*sh -c "export KUBECONFIG=.kubeconfig; oc login <K8s_LOGIN_URL>"*) or use existing one, for example,

```
$ sh -c "export KUBECONFIG=.kubeconfig; oc login
https://api.ocp4.example.com:6443"
```

- Run the configuration script and follow the steps to input configuration parameters

```
$ ./federatorai-setup-for-datadog.sh -k .kubeconfig
```

```
~# ./federatorai-setup-for-datadog.sh -k .kubeconfig
You are connecting to cluster: https://api.jc-ocp4.172-31-17-84.nip.io:6443

Getting Datadog info...
Input a Datadog API Key []:7c8475872d97cbc155b893**********
Input a Datadog Application Key []:a4c3f7620db747800d3dcb7c325fcb**********

Getting the Kafka info... No.1
Input Kafka consumer deployment name []: consumer
Input Kafka consumer deeployment namespace []: myproject
Input Kafka consumer minimum replica number []: 1
Input Kafka consumer maximum replica number []: 30
Input Kafka consumer group name []: group0001
Input Kafka consumer group namespace []: myproject
Input Kafka consumer topic name []: topic0001
Input Kafka consumer topic namespace []: myproject

Do you want to input another set? [default: n]:
Warning: kubectl apply should be used on resource created by either kubectl
create --save-config or kubectl apply
secret/federatorai-data-adapter-secret configured
Warning: kubectl apply should be used on resource created by either kubectl
create --save-config or kubectl apply
configmap/federatorai-data-adapter-config configured

Setup Federator.ai for datadog successfully
```

- Verify configuration result

```
~# ls -l config-result/
total 12
-rw-r--r-- 1 root root 5210 Jun  2 16:59 adapter-configmap.yaml
-rw-r--r-- 1 root root  723 Jun  2 16:59 adapter-secret.yaml
```

adapter-configmap.yaml

```
~# cat config-result/adapter-configmap.yaml
...
...
      [[inputs.datadog_application_aware]]
        urls = ["$DATADOG_QUERY_URL"]
        api_key = "$DATADOG_API_KEY"
        application_key = "$DATADOG_APPLICATION_KEY"
        # If we keep CLUSTER_NAME value empty, the agent will get k8s cluster
name automatically.
        cluster_name = "$CLUSTER_NAME"

        [inputs.datadog_application_aware.watched_kafka_consumer]
          application = "consumer"
          namespace = "myproject"
          min_replicas = 1
          max_replicas = 30

      [[inputs.alameda_datahub_query]]
        url = "$DATAHUB_URL"
        port = "$DATAHUB_PORT"
        ##The recommendation query range, unit: minutes
        recommendation_interval = 5
```

```
        # If we keep CLUSTER_NAME value empty, the agent will get k8s cluster
name automatically.
        cluster_name = "$CLUSTER_NAME"

        [[inputs.alameda_datahub_query.watched_source]]
          name = "group0001"
          namespace = "myproject"
          measurement = "kafka_consumer_group"
          scope = "recommendation"

        [[inputs.alameda_datahub_query.watched_source]]
          name = "group0001"
          namespace = "myproject"
          measurement = "kafka_consumer_group_current_offset"
          scope = "prediction"

        [[inputs.alameda_datahub_query.watched_source]]
          name = "topic0001"
          namespace = "myproject"
          measurement = "kafka_topic_partition_current_offset"
          scope = "prediction"
```

adapter-secret.yaml

```
~# ls -l config-result/adapter-secret.yaml
~# cat config_result/adapter-secret.yaml
apiVersion: v1
data:
  datadog_api_key:
T0dFNU5HUmlPV0UxWldRek9XVXpabU5sWVRrek1USmx***************=
  datadog_application_key:
TlRJNU1EUTNNakExT1RRMU1qVXdNRFkyTXpBNFltUmxaVGhoTURkbFl6azJ***************=
kind: Secret
metadata:
  creationTimestamp: "2020-06-02T08:09:54Z"
  name: federatorai-data-adapter-secret
  namespace: federatorai
  ownerReferences:
  - apiVersion: federatorai.containers.ai/v1alpha1
    blockOwnerDeletion: true
    controller: true
    kind: AlamedaService
    name: my-alamedaservice
    uid: f92a5775-e50d-463b-a5c8-c38720b1bce3
  resourceVersion: "49616"
  selfLink: /api/v1/namespaces/federatorai/secrets/federatorai-data-adapter-
secret
  uid: eca3c382-c431-4800-848f-6f4271ed7729
```

Datadog Agent/Cluster Agent/WPA Installation and Configuration

Preparation

Reported by Datadog, if your platform is OpenShift, there is a known issue that the Datadog is unable to detect OpenShift *kube-state-metrics* service.

- https://www.datadoghq.com/blog/openshift-monitoring-with-datadog/

Please see the following references below for more details.

- https://github.com/kubernetes/kube-state-metrics/issues/1099
- https://github.com/kubernetes/kube-state-metrics/issues/754
- https://github.com/kubernetes/kube-state-metrics/issues/730

To address this issue, please install *kube-state-metrics* version v1.5.0.

1. **Download "kube-state-metrics"**

```
~# git clone https://github.com/kubernetes/kube-state-metrics.git
~# cd kube-state-metrics/
~# git checkout v1.5.0
```

2. **Edit "ClusterRole" name and "ClusterRoleBinding" name in kubernetes folder to prevent cluster role name and cluster role binding name collision**

   Edit these files:
   - *kube-state-metrics-cluster-role-binding.yaml*
   - *kube-state-metrics-cluster-role.yaml*

```
~# vi kube-state-metrics/kubernetes/kube-state-metrics-cluster-role-binding.yaml
apiVersion: rbac.authorization.k8s.io/v1
# kubernetes versions before 1.8.0 should use rbac.authorization.k8s.io/v1beta1
kind: ClusterRoleBinding
metadata:
  name: datadog-kube-state-metrics
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: kube-state-metrics
subjects:
- kind: ServiceAccount
  name: kube-state-metrics
  namespace: kube-system

~# vi kube-state-metrics/kubernetes/kube-state-metrics-cluster-role.yaml
apiVersion: rbac.authorization.k8s.io/v1
# kubernetes versions before 1.8.0 should use rbac.authorization.k8s.io/v1beta1
kind: ClusterRole
metadata:
  name: datadog-kube-state-metrics
rules:
- apiGroups: [""]
  resources:
....
```

3. **Apply "kube-state-metrics" YAML files**

```
~# cd kube-state-metrics/
~# kubectl apply -d kubernetes/
```

Installation

1. Deploy Datadog Agent

- Download Datadog Agent package

```
~# wget datadog_agent.tar.gz
~# tar zxf datadog_agent.tar.gz
```

- Edit datadog-secret.yaml, fill in base64 encoded API Key and Application Key

```
~# vi datadog-secret.yaml
apiVersion: v1
kind: Secret
type: Opaque
metadata:
  name: datadog-secret
data:
  api-key: O2M4N*****cyZDk*****MTU1Y*****E4MzI*****Yyc=
  app-key: ZTRjM*****IwZGI*****MDBkM*****djMzI*****MDhhM*****UzAA==
```

- Edit datadog-agent.yaml, change DD_TAGS “cluster.local” to your cluster name

```
~# vi datadog-agent.yaml
...
...
      valueFrom:
              fieldRef:
                fieldPath: status.hostIP
          - name: DD_APM_ENABLED
            value: "true"
          - name: DD_TAGS
            value: "kube_cluster:8e107522-4a51-48e4-bb34-0928f93ce927"
          - name: DD_CLUSTER_AGENT_AUTH_TOKEN
            valueFrom:
              secretKeyRef:
                name: datadog-auth-token
                key: token
          - name: DD_CLUSTER_AGENT_ENABLED
```

- Edit kafka-comsumer-conf.yaml, fill in Kafka connection address. Kafka connection address can be obtained by ‘kubectl get svc -n <namespace>’ command.

```
~# kubectl get svc -n myproject
my-cluster-kafka-bootstrap    ClusterIP   172.30.184.191   <none>
9091/TCP,9092/TCP,9093/TCP,9404/TCP   167m
my-cluster-kafka-brokers      ClusterIP   None             <none>
9091/TCP,9092/TCP,9093/TCP            167m
my-cluster-kafka-exporter     ClusterIP   172.30.48.30     <none>        9404/TCP
166m
```

```
my-cluster-zookeeper-client   ClusterIP   172.30.126.227   <none>
9404/TCP,2181/TCP                         168m
my-cluster-zookeeper-nodes    ClusterIP   None             <none>
2181/TCP,2888/TCP,3888/TCP                168m
```

```
~# vi datadog-consumer-conf.yaml
instances:
        ## @param kafka_connect_str - list of strings - required
        ## Kafka endpoints and port to connect to.
        ##
        ## In a production environment, it's often useful to specify multiple
        ## Kafka nodes for a single check instance. This way you
        ## only generate a single check process, but if one host goes down,
        ## KafkaClient tries contacting the next host.
        ## Details: https://github.com/DataDog/dd-agent/issues/2943
        #
      - kafka_connect_str:
        # - <kafka-broker-service-name>.<kafka-namespace>:9092
         - my-cluster-kafka-brokers.myproject:9092
```

- Deploy Datadog Agent and confirm the status

```
~# kubectl apply -f deploy/
~# kubectl get pods -n default
```

```
~# kubectl apply -f deploy/

clusterrole.rbac.authorization.k8s.io/datadog-agent created
clusterrolebinding.rbac.authorization.k8s.io/datadog-agent created
podsecuritypolicy.policy/datadog-agent created
securitycontextconstraints.security.openshift.io/datadog-agent created
daemonset.apps/datadog-agent created
secret/datadog-auth-token created
configmap/datadog-conf created
secret/datadog-secret created
configmap/kafka-conf created
configmap/kafka-consumer-conf created
configmap/kafka-metrics created
serviceaccount/datadog-agent created

~# kubectl get pod -n default
NAME                                          READY   STATUS    RESTARTS   AGE
datadog-agent-pzw97                           1/1     Running   0          36s
datadog-agent-r6ztr                           1/1     Running   0          36s
datadog-agent-vp57s                           1/1     Running   0          36s
```

2. Deploy Datadog Cluster Agent

- Download Datadog Cluster Agent package

```
~# wget datadog-cluster-agent.tar.gz
~# tar zxf datadog_cluster_agent.tar.gz
```

- Deploy Datadog Cluster Agent and confirm the status

```
~# kubectl apply -f datadog_cluster_agent/deploy/

service/datadog-custom-metrics-server created
deployment.apps/datadog-cluster-agent created
service/datadog-cluster-agent created
clusterrole.rbac.authorization.k8s.io/dca created
clusterrolebinding.rbac.authorization.k8s.io/dca created
serviceaccount/dca created
clusterrolebinding.rbac.authorization.k8s.io/system:auth-delegator created
rolebinding.rbac.authorization.k8s.io/dca created
apiservice.apiregistration.k8s.io/v1beta1.external.metrics.k8s.io created
clusterrole.rbac.authorization.k8s.io/external-metrics-reader-cluster-agent
created
clusterrolebinding.rbac.authorization.k8s.io/external-metrics-reader-cluster-agent
created
securitycontextconstraints.security.openshift.io/datadog-cluster-agent created

~# kubectl get pod -n default
NAME                                          READY   STATUS    RESTARTS   AGE
datadog-agent-pzw97                           1/1     Running   0
6m36s
datadog-agent-r6ztr                           1/1     Running   0
6m36s
datadog-agent-vp57s                           1/1     Running   0
6m36s
datadog-cluster-agent-ccb767b69-5vsqg         1/1     Running   0          21s
```

3. Deploy Datadog Watermark Pod Autoscaler Controller (OKD3.11)

- Download Datadog WPA package

```
~# wget datadog-wpa.tar.gz
~# tar zxf datadog-wpa.tar.gz
```

- Install Watermark Pod Autoscaler controller. Note, WPA package is a Helm Chart package. If you don't have *helm* installed, use the following command to install.

```
~# curl -L https://raw.githubusercontent.com/helm/helm/master/scripts/get-helm-3 |
bash
```

Set up environment variables and then use 'helm' command to install WPA.

```
$ DD_NAMESPACE="default"
$ DD_NAMEWPA="wpacontroller"
$ helm install $DD_NAMEWPA -n $DD_NAMESPACE ./chart/watermarkpodautoscaler
```

```
~# pwd
/root/datadog_wpa/watermarkpodautoscaler_for_k8s_1.11
~# DD_NAMESPACE="default"
~# DD_NAMEWPA="wpacontroller"
~# helm install $DD_NAMEWPA -n $DD_NAMESPACE ./chart/watermarkpodautoscaler
~# kubectl get pods -n default
NAME                                          READY   STATUS    RESTARTS   AGE
datadog-agent-pzw97                           1/1     Running   0          24m
datadog-agent-r6ztr                           1/1     Running   0          24m
```

```
datadog-agent-vp57s                                    1/1     Running   0          24m
datadog-cluster-agent-ccb767b69-5vsqg                  1/1     Running   0          18m
prometheus-adapter-67c9c85966-lbgfr                    1/1     Running   0          20h
wpacontroller-watermarkpodautoscaler-68484f8dd4-48775 1/1     Running   0          15h
```

Note: If you got the following errors, try to run additional commands to solve the issues.

2:15 PM
[root@JeOKD311-12140 watermarkpodautoscaler_for_k8s_1.11]# helm install $DD_NAMEWPA -n $DD_NAMESPACE ./chart/watermarkpodautoscaler
Error: rendered manifests contain a resource that already exists. Unable to continue with install: existing resource conflict: namespace: , name: external-metrics-reader, existing_kind: rbac.authorization.k8s.io/v1, Kind=ClusterRoleBinding, new_kind: rbac.authorization.k8s.io/v1, Kind=ClusterRoleBinding

```
$ kubectl get clusterrolebinding | grep external-metrics-reader
$ lubectl delete clusterrolebinding external-metrics-reader
```

- Edit wpa.yaml, fill in Kafka consumer information

```
~# vi wpa.yaml
apiVersion: datadoghq.com/v1alpha1
kind: WatermarkPodAutoscaler
metadata:
  name: <wpa_name>      → e.g "consumer"
  namespace: <kafka-consumer_deployment_namespace>   → e.g."myproject"
spec:
  # Add fields here
  # algorithm must be average
  algorithm: average
  maxReplicas: 15
  minReplicas: 1
  tolerance: 0.01
  downscaleForbiddenWindowSeconds: 300
  upscaleForbiddenWindowSeconds: 15
  scaleUpLimitFactor: 50
  scaleDownLimitFactor: 20
  scaleTargetRef:
    kind: Deployment
    apiVersion: apps/v1
    name: <kafka-consumer_deployment_name>   → e.g "consumer"
  readinessDelay: 10
  metrics:
  # Resource or External type supported
  # Example usage of External type
  - type: External
    external:
      # do not edit highWatermakr, and lowWatermark
      # highWatermark and lowWatermark must be 1
      highWatermark: "1"
      lowWatermark: "1"
      metricName: federatorai.recommendation
      metricSelector:
        matchLabels:
          resource: replicas
```

```
          kube_cluster: <cluster_name>     → "K8s cluster uuid: 9a6aaa20-5d2b-
11ea-86b6-00505698xxxx"
          kube_deployment: <kafka-consumer_deployment_name>   → e.g "consumer"
          kube_namespace: <kafka-consumer_deployment_namespace>   → e.g
"myproject"

~# kubectl apply -f  wpa.yaml
```

- Deploy WPA and confirm the status

```
~# kubectl apply -f wpa.yaml
```

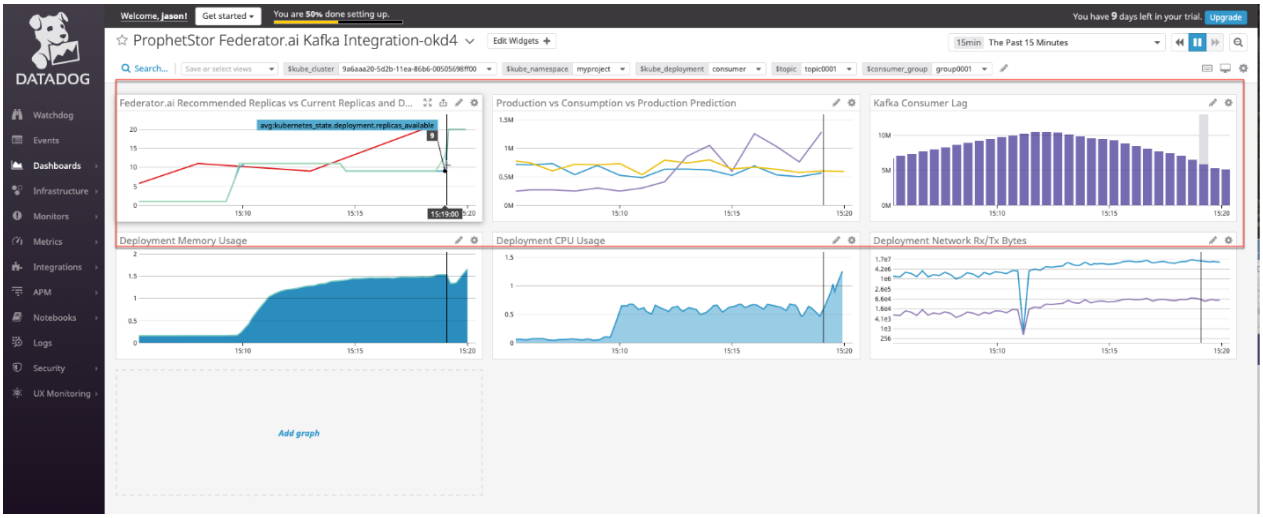
- Do a dry-run and check Datadog dashboard to confirm the status

```
~# ./run.sh -i 1 -t 10 -m false -n 1200

You are connecting to cluster: https://jeokd311-12140:8443
Checking OpenShift version...
openshift_version=3
...Passed
Checking environment...
Current compute nodes customized labels...
jeokd311-12140 test=ai
jeokd311-12141 test=producer
jeokd311-12142 test=consumer
jeokd311-12143 test=consumer

Labeling compute nodes...
Setting jeokd311-12143 to test=consumer
Setting jeokd311-12140 to test=ai
Setting jeokd311-12141 to test=producer
Setting jeokd311-12142 to test=consumer
Done
pod "alameda-recommender-6bcd8b8fc6-xw4rb" deleted

Checking pods...
```

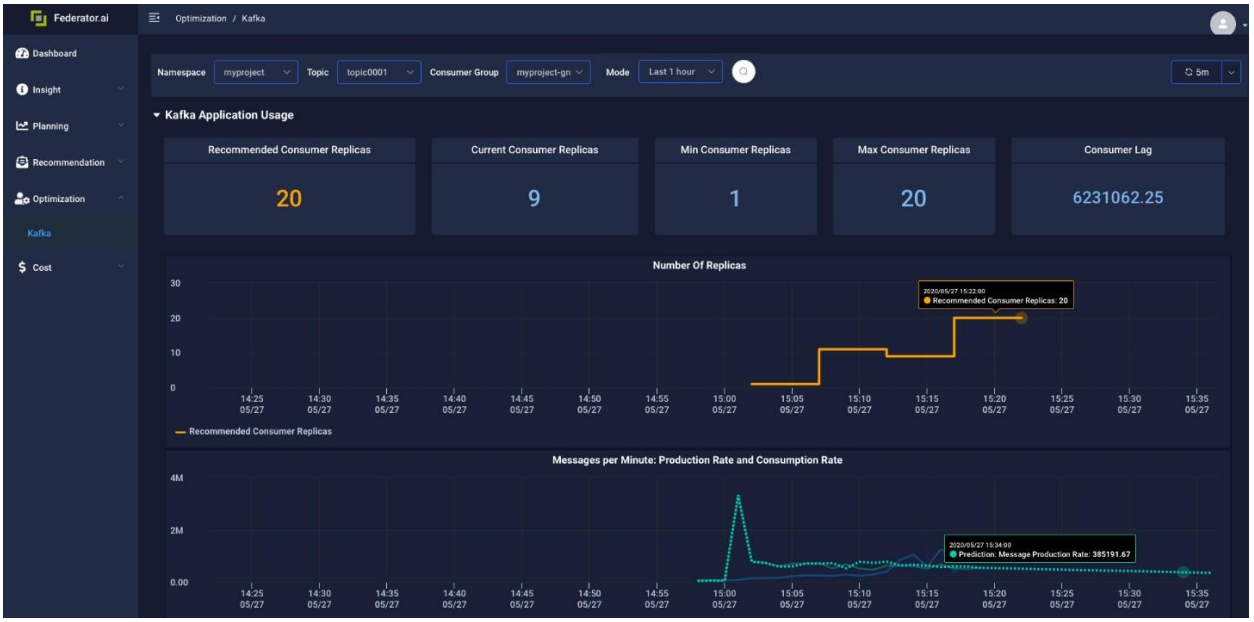
Check WPA controller

```
~# kubectl deployment consumer -n myproject --watch
NAME       DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
consumer   20        20        20           20          28m
consumer   16        20        20           20          39m
consumer   16        20        20           20          39m
consumer   16        16        16           16          39m
consumer   13        16        16           16          39m
consumer   13        16        16           16          39m
consumer   13        13        13           13          39m
consumer   11        13        13           13          50m
consumer   11        13        13           13          50m
consumer   11        11        11           11          50m
consumer   10        11        11           11          50m
consumer   10        11        11           11          50m
consumer   10        10        10           10          50m
```

Check Datadog dashboard



Check Federator.ai GUI



Appendix

OKD 3.11 Installation

1. OpenShift Cluster Specification

- Dell R720 or Later model.
- 1 x Master Node:
  - CPU : 8 vCore
  - Memory: 16 GB
  - Disk(OS): 100GB (SSD is better)
- 2 x Worker Node: (For ai , Producer pod)
  - CPU : 8 vCore
  - Memory: 16 GB
  - Disk(OS): 100GB (SSD is better)
- 1 x Worker Node: (For Consumer pods)
  - CPU : 16 vCore
  - Memory: 32 GB (64GB is better)
  - Disk(OS): 100GB (SSD is better)

2. Cluster Node OS installation & Configuration

- Install CentOS7.6 with minimal installation
- Configure static IP
- Install prerequiestis packages

```
$ yum install -y wget
$ wget http://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm
$ rpm -ivh epel-release-latest-7.noarch.rpm
$ yum -y --enablerepo=epel install pyOpenSSL
$ curl -o ansible.rpm https://releases.ansible.com/ansible/rpm/release/epel-7-x86_64/ansible-2.6.5-1.el7.ans.noarch.rpm
$ yum -y --enablerepo=epel install ansible.rpm
```

3. Install OKD 3.11

- Download script : https://gitlab.prophetservice.com/cheesiong.lee/openshift-deploy
- Upload script including “deploy.sh” & “inventory.ini.template” to CentOS 7 node (as master node)

```
$ ./deploy.sh [-u ssh_username] [-m master_ip] → all-in-one environment
$ ./deploy.sh [-u ssh_username] [-m master_ip,...] [-c compute_ip,...] → New deployment cluster environment
$ ./deploy.sh add [-c new_compute_ip, ...]  → Add new compute node to existing cluster
```

Troubleshooting

1. WPA Report error during autoscaling

- Error message in WPA Controller

```
~# kubectl get pod -n default
NAME                                                READY   STATUS    RESTARTS   AGE
datadog-agent-2m6kk                                 1/1     Running   2          2d
datadog-agent-8kd54                                 1/1     Running   0          2d
datadog-agent-94rl6                                 1/1     Running   0          2d
datadog-agent-mq4mv                                 1/1     Running   0          2d
datadog-cluster-agent-74f44fdd4d-82tjp              1/1     Running   0          1d
docker-registry-1-vw59s                             1/1     Running   4          324d
prometheus-adapter-799b7dfc4f-rs7zj                 1/1     Running   1          6d
registry-console-2-jxfdl                            1/1     Running   2          6d
router-1-sw78l                                      1/1     Running   4          324d
wpacontroller-watermarkpodautoscaler-7ffbb97f9d-hcbsg 1/1   Running   0          1d

~# kubectl logs wpacontroller-watermarkpodautoscaler-7ffbb97f9d-hcbsg -n default
```

{"level":"info","ts":1589533961.5993037,"logger":"wpa_controller","msg":"Successful rescale","Request.Namespace":"myproject","Request.Name":"consumer1-topic0001-group-0001","currentReplicas":40,"desiredReplicas":40,"rescaleReason":""}

{"level":"error","ts":1589533961.600972,"logger":"wpa_controller","msg":"Error during reconcileWPA","Request.Namespace":"myproject","Request.Name":"consumer1-topic0001-group-0001","error":"the server could not find the requested resource (put watermarkpodautoscalers.datadoghq.com consumer1-topic0001-group-0001)","stacktrace":"github.com/go-logr/zapr.(*zapLogger).Error\n\twatermarkpodautoscaler/vendor/github.com/go-logr/zapr/zapr.go:128\ngithub.com/DataDog/watermarkpodautoscaler/pkg/controller/watermarkpodautoscaler.(*ReconcileWatermarkPodAutoscaler).Reconcile\n\twatermarkpodautoscaler/pkg/controller/watermarkpodautoscaler/watermarkpodautoscaler_controller.go:345\nsigs.k8s.io/controller-runtime/pkg/internal/controller.(*Controller).reconcileHandler\n\twatermarkpodautoscaler/vendor/sigs.k8s.io/controller-runtime/pkg/internal/controller/controller.go:216\nsigs.k8s.io/controller-runtime/pkg/internal/controller.(*Controller).processNextWorkItem\n\twatermarkpodautoscaler/vendor/sigs.k8s.io/controller-runtime/pkg/internal/controller/controller.go:192\nsigs.k8s.io/controller-runtime/pkg/internal/controller.(*Controller).worker\n\twatermarkpodautoscaler/vendor/sigs.k8s.io/controller-runtime/pkg/internal/controller/controller.go:171\nk8s.io/apimachinery/pkg/util/wait.JitterUntil.func1\n\twatermarkpodautoscaler/vendor/k8s.io/apimachinery/pkg/util/wait/wait.go:152\nk8s.io/apimachinery/pkg/util/wait.JitterUntil\n\twatermarkpodautoscaler/vendor/k8s.io/apimachinery/pkg/util/wait/wait.go:153\nk8s.io/apimachinery/pkg/util/wait.Until\n\twatermarkpodautoscaler/vendor/k8s.io/apimachinery/pkg/util/wait/wait.go:88"}

- Reason
  - WPA is incompatible with k8s 1.11
  - Error message when installing WPA on k8s 1.11

```
must only have "properties", "required" or "description" at the root if the
status subresource is enabled
```

- Workaround
  - In WPA helm template, we need to mark subresources key in CRD WatermarkPodAutoscaler

```
~# cd
datadog_wpa/watermarkpodautoscaler_for_k8s_1.11/chart/watermarkpodautoscaler/templ
ates
~# vi datadoghq.com_watermarkpodautoscalers_crd.yaml
...
...
    shortNames:
    - wpa
    singular: watermarkpodautoscaler
  scope: Namespaced
  #subresources:  ß mark
  #  status: {}         ß mark
  validation:
    openAPIV3Schema:
      description: WatermarkPodAutoscaler is the Schema for the
watermarkpodautoscalers
        API
      properties:
        apiVersion:
          description: 'APIVersion defines the versioned schema of this
representation
...
...
```

Note: It can auto-scale monitored application well, but raise some error during update status

- Related Datadog WPA ticket
  - https://github.com/DataDog/watermarkpodautoscaler/issues/50

2. Data Adapter reports errors

- Error message in Data Adapter

```
~# oc exec -it $(oc get pods|grep federatorai-data-adapter|grep Running|awk '{print $1}') -- cat /var/log/telegraf.log
> telegraf.log
~# cat telegraf.log | grep "E\!"
2020-05-15T09:59:33Z E! [datadog][application_aware] Failed to get kafka consumer spec replicas
2020-05-15T09:59:33Z E! [inputs.datadog_application_aware] Error in plugin:
[url=https://api.datadoghq.com/api/v1/query][kafka]: Failed to get consumer information.
```

- Reason
  - Datadog does not work with *kube-state-metrics* comes with OpenShift
- Solution
  - Install another *kube-state-metrics*

```
If there is another kube-state-metrics running on openshift, rename all the clusterrole and clusterrolebinding name
of kube-state-metrics to prevent kube-state-metrics clusterrole name collision

restart datadog agent and make sure agent integrate with kube-state-metrics properly.
check all the node agent status by following command
~# oc exec <datadog-agent-pod-name> agent status
```