Citrix Observability Exporter
## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Citrix ADC Observability Exporter</td>
<td>3</td>
</tr>
<tr>
<td>Deploy Citrix ADC Observability Exporter</td>
<td>5</td>
</tr>
<tr>
<td>Citrix ADC Observability Exporter with Zipkin as endpoint</td>
<td>7</td>
</tr>
<tr>
<td>Citrix ADC Observability Exporter with Prometheus and Grafana</td>
<td>12</td>
</tr>
<tr>
<td>Citrix ADC Observability Exporter with Elasticsearch as endpoint</td>
<td>18</td>
</tr>
<tr>
<td>Citrix ADC Observability Exporter with Kafka as endpoint</td>
<td>28</td>
</tr>
<tr>
<td>Citrix ADC Observability Exporter with Splunk Enterprise as endpoint</td>
<td>37</td>
</tr>
<tr>
<td>Citrix ADC Observability Exporter troubleshooting</td>
<td>42</td>
</tr>
</tbody>
</table>
Citrix ADC Observability Exporter

Citrix ADC Observability Exporter is a container which collects metrics and transactions from Citrix ADCs and transforms them to suitable formats (such as JSON, AVRO) for supported endpoints. You can export the data collected by Citrix ADC Observability Exporter to the desired endpoint. By analyzing the data exported to the endpoint, you can get valuable insights at a microservices level for applications proxied by Citrix ADCs.

Supported Endpoints

Citrix ADC Observability Exporter currently supports the following endpoints:

- Zipkin
- Kafka
- Elasticsearch
- Prometheus
- Splunk Enterprise

Overview

Distributed tracing support with Zipkin

In a microservice architecture, a single end-user request may span across multiple microservices and tracking a transaction and fixing sources of errors is challenging. In such cases, traditional ways for performance monitoring cannot accurately pinpoint where failures occur and what is the reason behind poor performance. You need a way to capture data points specific to each microservice which is handling a request and analyze them to get meaningful insights.

Distributed tracing addresses this challenge by providing a way to track a transaction end-to-end and understand how it is being handled across multiple microservices. OpenTracing is a specification and standard set of APIs for designing and implementing distributed tracing. Distributed tracers allow you to visualize the data flow between your microservices and helps to identify the bottlenecks in your microservices architecture.

Citrix ADC Observability Exporter implements distributed tracing for Citrix ADC and currently supports Zipkin as the distributed tracer.

Currently, you can monitor performance at the application level using Citrix ADC. Using Citrix ADC Observability Exporter with Citrix ADC, you can get tracing data for microservices of each application proxied by your Citrix ADC CPX, MPX, or VPX.
Transaction collection and streaming support

Citrix ADC Observability Exporter supports collecting transactions and streaming them to endpoints. Currently, Citrix ADC Observability Exporter supports Elasticsearch and Kafka as transaction endpoints.

Time series data support

Citrix ADC Observability Exporter supports collecting time series data (metrics) from Citrix ADC instances and exports them to Prometheus. Prometheus is a monitoring solution for storing time series data like metrics. You can then add Prometheus as a data source to Grafana and graphically view the Citrix ADC metrics and analyze the metrics.

How does Citrix ADC Observability Exporter work

Distributed tracing with Zipkin using Citrix ADC Observability Exporter

Logstream is a Citrix-owned protocol that is used as one of the transport modes to efficiently transfer transactions from Citrix ADC instances. Citrix ADC Observability Exporter collects tracing data as Logstream records from multiple Citrix ADCs and aggregates them. Citrix ADC Observability Exporter converts the data into a format understood by the tracer and then uploads to the tracer (Zipkin in this case). For Zipkin, the data is converted into JSON, with Zipkin-specific key values.

You can view the traces using the Zipkin user interface. However, you can also enhance the trace analysis by using Elasticsearch and Kibana with Zipkin. Elasticsearch provides long-term retention of the trace data and Kibana allows you to get much deeper insight into the data.

Citrix ADC Observability Exporter with Elasticsearch as the transaction endpoint

When Elasticsearch is specified as the transaction endpoint, Citrix ADC Observability Exporter converts the data to JSON format. On the Elasticsearch server, Citrix ADC Observability Exporter creates Elasticsearch indexes for each ADC on an hourly basis. These indexes are based on data, hour, UUID of the ADC, and the type of HTTP data (http_event or http_error). Then, Citrix ADC Observability Exporter uploads the data in JSON format under Elasticsearch indexes for each ADC. All regular transactions are placed into the http_event index and any anomalies are placed into the http_error index.

Citrix ADC Observability Exporter with Kafka as the transaction endpoint

When Kafka is specified as the transaction endpoint, Citrix ADC Observability Exporter converts the transaction data to Avro format and streams them to Kafka.
**Citrix ADC Observability Exporter with Prometheus as the endpoint for time series data**

When Prometheus is specified as the format for time series data, Citrix ADC Observability Exporter collects various metrics from Citrix ADCs and converts them to appropriate Prometheus format and exports them to the Prometheus server. These metrics include counters of the virtual servers, services to which the analytics profile is bound and global counters of HTTP, TCP and so on.

**Citrix ADC Observability Exporter with Splunk Enterprise as the endpoint**

When Splunk Enterprise is specified as the transaction endpoint, Citrix ADC Observability Exporter collects indexes, audit logs, and events and exports to Splunk Enterprise. Splunk Enterprise captures indexes and correlates real-time data in a repository from which it can generate reports, graphs, dashboards, and visualizations. Splunk Enterprise provides a graphical representation of these data.

**Deployment**

You can deploy Citrix ADC Observability Exporter using Kubernetes YAML. To deploy Citrix ADC Observability Exporter using Kubernetes YAML, see Deployment. To deploy Citrix ADC Observability Exporter using Helm charts, see Deploy using Helm charts.

**Features**

**Custom header logging**

Custom header logging enables logging of all HTTP headers of a transaction and currently supported on the Kafka endpoint.

For more information, see Custom header logging.

**Elasticsearch support enhancements**

Effective with the Citrix ADC Observability Exporter release 1.2.001, when the Citrix ADC Observability Exporter sends the data to the Elasticsearch server some of the fields are available in the string format. Also, index configuration options are also added for Elasticsearch. For more information on fields which are in the string format and how to configure the Elasticsearch index, see Elasticsearch support enhancements.

**Deploy Citrix ADC Observability Exporter**

February 3, 2022
This topic provides information on how to deploy Citrix ADC Observability Exporter using Kubernetes YAML files.

**Note:**
You can deploy Citrix Observability Exporter using Kubernetes YAML files or using Helm charts.

Based on your Citrix ADC deployment, you can use Citrix ADC Observability Exporter to export metrics and transactions from Citrix ADC CPX, MPX, or VPX.

The following diagram shows a deployment of Citrix Observability Exporter with all the supported endpoints.

Citisx ADC Observability Exporter supports the following endpoints: Kafka, Elasticsearch, Prometheus, and Zipkin. Depending on the endpoint that you require, you can deploy Citrix ADC Observability Exporter with that endpoint.

You can use one of the following deployment procedures based on the endpoint that you require:

- Deploy Citrix ADC Observability Exporter with Zipkin
- Deploy Citrix ADC Observability Exporter with Prometheus
- Deploy Citrix ADC Observability Exporter with Elasticsearch
- Deploy Citrix ADC Observability Exporter with Kafka
- Deploy Citrix ADC Observability Exporter with Splunk
Citrix ADC Observability Exporter with Zipkin as endpoint

February 3, 2022

Citrix ADC Observability Exporter supports OpenTracing (OpenTracing is a part of OpenTelemetry now) using Zipkin as the endpoint. Citrix ADC Observability Exporter transforms the tracing data collected from Citrix ADCs into supported formats suitable for OpenTracing and exports them to Zipkin. Zipkin is a distributed tracing system that helps to gather the timing data required to troubleshoot latency problems in microservice architectures. Elasticsearch is used for long-term retention of trace data and the traces can be visualized using the Zipkin UI or Kibana.

The following diagram illustrates how the Zipkin architecture works:

1. When the tracing is enabled, initially, it adds additional open-tracing headers: x-trace-id, x-span-id, and x-parent-span-id to HTTP packet, before it forwards the packet to the next microservice pod.

2. The information about this communication or transaction is pushed to Citrix ADC Observability Exporter. The information includes the details about the headers, the timestamp (time when this request is initiated and the entire duration of the process), and annotations (annotations include HTTP, SSL, and TCP associated with that request).

3. Then, Citrix ADC Observability Exporter receives multiple trace messages from all the Citrix ADCs and aggregates them into Zipkin understandable JSON format, and push that to Zipkin through the API.

4. Similarly, if microservices are enabled with tracing, then that trace is sent to Zipkin through the API.

5. Zipkin API stores the trace data in the Elasticsearch database, and finally stitch the complete trace to the given HTTP request and visualize it in the visualization tool such as Kibana. You can view the time that the request spent on each microservices.
Deploy Citrix ADC Observability Exporter

Based on your Citrix ADC deployment, you can deploy Citrix ADC Observability Exporter either outside or inside Kubernetes clusters. You can deploy Citrix Observability Exporter as a pod inside the Kubernetes cluster or enable the configuration on Citrix ADC MPX or VPX form factor outside the cluster. You can deploy Citrix ADC Observability Exporter using the Kubernetes YAML file provided by Citrix.

The following diagram illustrates Citrix ADC as an ingress gateway with the Citrix ingress controller as a sidecar. Citrix ADC Observability Exporter sends the tracing data collected from Citrix ADCs to Zipkin API. The tracing data is, then, uploaded to the Elasticsearch server. From Elasticsearch, the data is sent to Zipkin UI or Kibana UI for visualization.

Prerequisites

- Ensure that you have a Kubernetes cluster with kube-dns or CoreDNS addon enabled.

To deploy Citrix ADC Observability Exporter with Zipkin, you must perform the following tasks:

1. Deploy the required application with the tracing support enabled.
2. Deploy Citrix ADC CPX enabled with the Citrix Observability Exporter support.
3. Deploy Zipkin, Elasticsearch, and Kibana using the YAML files.

Deploy application with tracing enabled

The following is a sample application deployment with tracing enabled.
Note:
If you have a pre-deployed web application, skip the steps 1 and 2.

1. Create a secret `ingress.crt` and key `ingress.key` using your own certificate and key.
   In this example, a secret, called `ing` in the default namespace, is created.

   ```bash
   kubectl create secret tls ing --cert=ingress.crt --key=ingress.key
   ```

2. Access the YAML file from `watches-app-tracing.yaml` to deploy the application.

   ```bash
   kubectl create -f watches-app-tracing.yaml
   ```

3. Define the specific parameters that you must import by specifying it in the ingress annotations of the application’s YAML file, using the smart annotations in the ingress.

   ```yaml
   ingress.citrix.com/analyticsprofile: '{
   "webinsight": {
   "httpurl":"ENABLED", "httpuseragent":"ENABLED", "httpHost":"
   ENABLED","httpMethod":"ENABLED","httpContentType":"ENABLED"
   }
   }
   '           
   ```

   Note: The parameters are predefined in the `watches-app-tracing.yaml` file.
   For more information about annotations, see Ingress annotations documentation.

**Deploy Citrix ADC CPX with the Citrix ADC Observability Exporter support**

You can deploy Citrix ADC CPX enabled with the Citrix ADC Observability Exporter support.

While deploying Citrix ADC CPX, you can modify the deployment YAML file `cpx-ingress-tracing.yaml` to include the configuration information that is required for the Citrix ADC Observability Exporter support.

Perform the following steps to deploy a Citrix ADC CPX instance with the Citrix ADC Observability Exporter support:

1. Download the `cpx-ingress-tracing.yaml` and `cic-configmap.yaml` file.
2. Create a ConfigMap with the required key-value pairs and deploy the ConfigMap. You can use the `cic-configmap.yaml` file that is available, for the specific endpoint, in the directory.

3. Modify Citrix ADC CPX related parameters, as required. For example, add lines under `args` in the `cpx-ingress-tracing.yaml` file as following:

```yaml
args:
- --configmap
  default/cic-configmap
```

4. Edit the `cic-configmap.yaml` file to specify the following variables for Citrix ADC Observability Exporter in the `NS_ANALYTICS_CONFIG` endpoint configuration.

```yaml
server: 'coe-.zipkin.default.svc.cluster.local' # COE service FQDN
```

5. Deploy Citrix ADC CPX with the Citrix ADC Observability Exporter support using the following commands:

```bash
kubectl create -f cpx-ingress-tracing.yaml
kubectl create -f cic-configmap.yaml
```

**Note:**
If you have used a namespace other than `default`, change `coe-.zipkin.default.svc.cluster.local` to `coe-zipkin.<desired-namespace>.svc.cluster.local`. If ADC is outside the Kubernetes cluster, then you must specify IP address and Nodeport address of Citrix ADC Observability Exporter.

### Deploy Zipkin, Elasticsearch, and Kibana using YAML files

To deploy Zipkin, Elasticsearch, and Kibana using YAML, perform the following steps:

1. Download the following YAML files:
   - `zipkin.yaml`
   - `Elasticsearch.yaml`
   - `kibana.yaml`

2. Edit the namespace definition, if you want to use a custom namespace other than the `default`. 

---

© 1999–2022 Citrix Systems, Inc. All rights reserved.
3. Run the following commands to deploy Zipkin, Elasticsearch, and Kibana:

```bash
1. kubectl create -f zipkin.yaml
2. kubectl create -f elasticsearch.yaml
3. kubectl create -f kibana.yaml
```

**Note:**
Zipkin, Elasticsearch, and Kibana are deployed in the default namespace of the same Kubernetes cluster.

### Deploy Citrix ADC Observability Exporter using the YAML file

You can deploy Citrix ADC Observability Exporter using the YAML file. Download the `coe-zipkin.yaml` file.

To deploy Citrix ADC Observability Exporter using the Kubernetes YAML, run the following command in the Elasticsearch endpoint:

```bash
1. kubectl create -f coe-zipkin.yaml
```

**Note:**
Modify the YAML file for Citrix ADC Observability Exporter if you have a custom namespace other than the default.

### Verify the Citrix ADC Observability Exporter deployment

To verify the Citrix ADC Observability Exporter deployment, perform the following:

1. Verify the deployment by sending a request to the application using the following command.

```bash
```

2. Open the Zipkin user interface using the Kubernetes node IP address and nodeport.

```bash
1. http://*k8-node-ip-address*:*:node-port*/
```
In the following image, you can view the traces of the *Watches* application. The *Watches* application has multiple microservices for each watches type, communicating with each other to serve the application data. The trace data shows application FASTTRACK took more time to serve when compare to other micro services. In this way, you can identify the slow performing workloads and troubleshoot it.

You can view raw data on your Kibana dashboard too. Open Kibana using the `http://<node-ip>:<node-port>` and commence with defining a `zipkin` index pattern.

Use the `timestamp_millis` field as the timestamp field. After creating the index pattern, click the Discover tab and you can view the trace information collected by Zipkin.

For information on troubleshooting related to Citrix ADC Observability Exporter, see Citrix ADC CPX troubleshooting.

**Citrix ADC Observability Exporter with Prometheus and Grafana**

February 8, 2022

You can configure Prometheus as an endpoint to pull data from Citrix ADC Observability Exporter. You can also configure Grafana to visualize the same data graphically.
Citrix ADC Observability Exporter has a push-gateway server that listens to port 5563 to serve metrics based on pull requests from Prometheus. Citrix ADC Observability Exporter exports time series data to Prometheus.

**Deploy Citrix ADC Observability Exporter**

You can deploy Citrix ADC Observability Exporter using the YAML file. Based on your Citrix ADC deployment, deploy Citrix ADC Observability Exporter either outside or inside Kubernetes clusters. You can deploy Citrix ADC Observability Exporter as a pod inside the Kubernetes cluster or on Citrix ADC MPX or VPX appliance outside the cluster.

**Prerequisites**

- Ensure that you have a Kubernetes cluster with kube-dns or CoreDNS addon enabled.

Deploying Citrix ADC Observability Exporter with the Prometheus endpoint includes the following tasks:

- Deploy a sample application
- Deploy Citrix ADC CPX with support enabled for Citrix ADC Observability Exporter
- Deploy Prometheus and Grafana using YAML files
- Deploy Citrix ADC Observability Exporter using the YAML file
- Configure Citrix ADC to export metrics (optional)
- Configure Prometheus (optional) to pull telemetry data
- Configure Grafana
- Create Grafana visualization

**Deploy a sample application**

The following is an example procedure for deploying a sample webserver application.

*Note:*

If you have a pre-deployed web application, skip the steps from step 1 to step 3.
1. Create a secret `ingress.crt` and key `ingress.key` using your own certificate and key.

   In this example, a secret, called `ing` in the default namespace, is created.

   ```bash
   kubectl create secret tls ing --cert=ingress.crt --key=ingress.key
   ```

2. Access the YAML file from `webserver-es.yaml` to deploy the application.

   ```bash
   kubectl create -f webserver-es.yaml
   ```

3. Define the specific parameters that you must import by specifying it in the ingress annotations of the application's YAML file, using the smart annotations in the ingress.

   ```yaml
   ingress.citrix.com/analyticsprofile: '{
     "webinsight": {
       "httpurl":"ENABLED", "httpuseragent":"ENABLED", "httpHost":"ENABLED","httpMethod":"ENABLED","httpContentType":"ENABLED" }
     }
   ' 
   ```

   **Note:**
   The parameters are predefined in the `webserver-es.yaml` file.

   For more information about Annotations, see Ingress annotations documentation.

---

**Deploy Citrix ADC CPX with support enabled for Citrix ADC Observability Exporter**

You can deploy Citrix ADC CPX as a side car with the Citrix ADC Observability Exporter support enabled along with Citrix ingress controller. You can modify the Citrix ADC CPX YAML file `cpx-ingress-es.yaml` to include the configuration information that is required for the Citrix ADC Observability Exporter support.

The following is a sample application deployment procedure.

1. Download the `cpx-ingress-prometheus.yaml` and `cic-configmap.yaml` file.

2. Create a ConfigMap with the required key-value pairs and deploy the ConfigMap. You can use the `cic-configmap.yaml` file that is available, for the specific endpoint, in the directory.

3. Modify Citrix ADC CPX related parameters, as required.
4. Edit the `cic-configmap.yaml` file and specify the following variables for Citrix ADC Observability Exporter in the `NS_ANALYTICS_CONFIG` endpoint configuration.

```yaml
server: 'coe-prometheus.default.svc.cluster.local' # COE service FQDN
```

**Note:**
If you have used a namespace other than `default`, change `coe-prometheus.default.svc.cluster.local` to `coe-prometheus.<desired-namespace>.svc.cluster.local`.

5. Deploy Citrix ADC CPX with the Citrix ADC Observability Exporter support using the following commands:

```bash
kubectl create -f cpx-ingress-prometheus.yaml
kubectl create -f cic-configmap.yaml
```

**Deploy Prometheus and Grafana using YAML files**

To deploy Prometheus and Grafana using YAML files, perform the following steps:

1. Download the Prometheus-Grafana YAML file from `prometheus-grafana.yaml`.
2. Edit the namespace definition if you want to use a different namespace other than `default`.
3. Run the following commands to deploy Prometheus and Grafana:

```bash
kubectl create -f prometheus-grafana.yaml
```

**Note:**
Prometheus and Grafana are deployed in the default namespace of the same Kubernetes cluster.

**Deploy Citrix ADC Observability Exporter using the YAML file**

You can deploy Citrix ADC Observability Exporter using the YAML file. Download the YAML file from `coe-prometheus.yaml`.

To deploy Citrix ADC Observability Exporter using the Kubernetes YAML, run the following command:
Citrix Observability Exporter

```bash
kubectl create -f coe-prometheus.yaml
```

**Note:**
Modify the YAML file for Citrix ADC Observability Exporter if you have a custom namespace.

**Configure Citrix ADC to export metrics (optional)**

**Note:**
If you do not use Citrix ingress controller to configure Citrix ADC, then you can do the following manual configuration on your Citrix ADC.

You can manually configure Citrix ADCs to export metrics to the Citrix ADC Observability exporter. Specify the Citrix ADC Observability Exporter IP/FQDN address as an HTTP service and combine it to the default `ns_analytics_time_series_profile` analytics profile. Enable the metrics export and set the output mode to Prometheus.

The following is a sample configuration:

```bash
1 add server COE_instance 192.168.1.102
2 add service coe_metric_collector_svc_192.168.1.102 COE_instance HTTP 5563
3 set analytics profile ns_analytics_time_series_profile -collector coe_metric_collector_svc_192.168.1.102 -Metrics ENABLED -OutputMode Prometheus
```

**Configure Prometheus (optional) to pull telemetry data**

Prometheus services are available as Docker images on Quay container registry and Docker Hub.

To launch Prometheus and expose it on port 9090, run the following command:

```bash
docker run -p 9090:9090 prom/prometheus
```

To manually add Citrix ADC Observability Exporter as scrape target, edit the `prometheus.yaml` file. Specify the Citrix ADC Observability Exporter IP/FQDN address and the port 5563 as the scrape target in the YAML file.
Citrix Observability Exporter

```yaml
scrape_configs:
  - job_name: coe
    static_configs:
      - targets: ['192.168.1.102:5563']
```

Configure Grafana

In the current deployment, a Prometheus server has already been added as a data source. If you use an existing Prometheus server for the deployment, ensure to add the same as a data source on your Grafana. For more information, see Grafana support for Prometheus.

Create Grafana visualization

You can create a Grafana dashboard and select the key metrics and the visualization type that is suitable for the data.

The following procedure shows adding of the ADC CPU metric to a Grafana panel:

1. Specify the Panel Title as **ADC CPU**.
2. In the Query tab, for the query A, specify the metric as **cpu_use**.
3. In the Settings tab, select the **Visualization type**.

You can modify the data and its representation in Grafana. For more information, see Grafana Documentation.

![Grafana Dashboard](image)

Import pre-built dashboards for Grafana

You can also import pre-built dashboards to Grafana. See the available Dashboards.

For information on troubleshooting related to Citrix ADC Observability Exporter, see Citrix ADC CPX troubleshooting.
Citrix ADC Observability Exporter with Elasticsearch as endpoint

February 3, 2022

Citrix ADC Observability Exporter is a container that collects metrics and transactions from Citrix ADC. It transforms the data into the supported format (such as JSON) and exports data to Elasticsearch as an endpoint. Elasticsearch is a search engine based on the Lucene library. It provides a distributed, multitenant-capable, and full-text search engine with an HTTP web interface and schema-free JSON documents.

Deploy Citrix ADC Observability Exporter

You can deploy Citrix ADC Observability Exporter using the YAML file. Based on your Citrix ADC deployment, you can deploy Citrix ADC Observability Exporter either outside or inside Kubernetes clusters. You can deploy Citrix ADC Observability Exporter as a pod inside the Kubernetes cluster or on Citrix ADC MPX or VPX appliance outside the cluster.

The following diagram illustrates a Citrix ADC as an Ingress Gateway with the Citrix ingress controller and Citrix ADC Observability Exporter as sidecars. Citrix ADC Observability Exporter sends Citrix ADC application metrics and transaction data to Elasticsearch and the same data exports to Kibana. Kibana provides a graphical representation of the data.

Prerequisites

- Ensure that you have a Kubernetes cluster with kube-dns or CoreDNS addon enabled.

In the following procedure, the YAML file is used to deploy Citrix ADC Observability Exporter in the Kubernetes default namespace. If you want to deploy in a private namespace other than the default,
edit the YAML file to specify the namespace.

The following is a sample application deployment procedure.

**Note:**
If you have a pre-deployed web application, skip the steps 1 and 2.

1. Create a secret **ingress.crt** and key **ingress.key** using your own certificate and key.
   
   In this example, a secret, called **ing** in the default namespace, is created.

   ```shell
   kubectl create secret tls ing --cert=ingress.crt --key=ingress.key
   ```

2. Access the YAML file from **webserver-es.yaml** to deploy the application.

   ```shell
   kubectl create -f webserver-es.yaml
   ```

3. Define the specific parameters that you must import by specifying it in the ingress annotations of the application’s YAML file, using the smart annotations in the ingress.

   ```yaml
   ingress.citrix.com/analyticsprofile: '{
     "webinsight": {
       "httpurl":"ENABLED", "httpuseragent":"ENABLED", "httpHost":"ENABLED","httpMethod":"ENABLED","httpContentType":"ENABLED" }
     }
   '
   ```

   **Note:**
   The parameters are predefined in the **webserver-es.yaml** file.

   For more information about Annotations, see [Ingress annotations documentation](#).

---

**Deploy Citrix ADC CPX with the Citrix ADC Observability Exporter support**

You can deploy Citrix ADC CPX as a side car with the Citrix ADC Observability Exporter support enabled along with Citrix ingress controller. You can modify the Citrix ADC CPX YAML file **cpx-ingress-es.yaml** to include the configuration information that is required for the Citrix ADC Observability Exporter support.

Perform the following steps to deploy a Citrix ADC CPX instance with the Citrix ADC Observability Exporter support:
1. Download the `cpx-ingress-es.yaml` and `cic-configmap.yaml` file.

2. Create a ConfigMap with the required key-value pairs and deploy the ConfigMap. You can use the `cic-configmap.yaml` file that is available, for the specific endpoint, in the directory.

3. Modify Citrix ADC CPX related parameters, as required.

4. Edit the `cic-configmap.yaml` file and specify the following variables for Citrix ADC Observability Exporter in the `NS_ANALYTICS_CONFIG` endpoint configuration.

   ```
   server: 'coe-es.default.svc.cluster.local' # COE service FQDN
   ```

   **Note:**
   If you have used a namespace other than `default`, change `coe-es.default.svc.cluster.local` to `coe-es.<desired-namespace>.svc.cluster.local`. If ADC is outside the Kubernetes cluster, then you must specify IP address and nodeport address of Citrix ADC Observability Exporter.

5. Deploy Citrix ADC CPX with the Citrix ADC Observability Exporter support using the following commands:

   ```
   kubectl create -f cpx-ingress-es.yaml
   kubectl create -f cic-configmap.yaml
   ```

**Deploy Elasticsearch and Kibana using YAML files**

1. Download the Elasticsearch YAML file from `elasticsearch.yaml` and the Kibana YAML file from `kibana.yaml`.

2. Edit the namespace definition, if you want to use a different namespace other than `default`.

3. Run the following commands to deploy Elasticsearch and Kibana:

   ```
   kubectl create -f elasticsearch.yaml
   kubectl create -f kibana.yaml
   ```

   **Note:** Elasticsearch and Kibana are deployed in the default namespace of the same Kubernetes cluster.
Citrix Observability Exporter

**Deploy Citrix ADC Observability Exporter using the YAML file**

You can deploy Citrix Observability Exporter using the YAML file. Download the YAML file from `coes.yaml`.

To deploy Citrix Observability Exporter using the Kubernetes YAML, run the following command in the Elasticsearch endpoint:

```
kubectl create -f coe-es.yaml
```

**Note:**

Modify the YAML file for Citrix ADC Observability Exporter if you have a custom namespace.

**Verify the Citrix ADC Observability Exporter deployment**

To verify the Citrix ADC Observability Exporter deployment, perform the following:

1. Verify the deployment using the following command:

```
kubectl get deployment,pods,svc -o wide
```


For example, from Step 1, access `http://10.102.40.41:30176/` in which, `10.102.40.41` is one of the Kubernetes node IPs.

For example, from step 1, access \texttt{http://10.102.40.41:32529/} in which, \texttt{10.102.40.41} is one of the Kubernetes node IPs.

a) Click \textit{Explore on my own}.

![Welcome to Elastic](image)

b) Click \textit{Connect to your Elasticsearch index}. 

© 1999–2022 Citrix Systems, Inc. All rights reserved.
c) Click **Saved Objects**.

d) Download and import the Kibana Dashboard from **KibanaAppTrans.ndjson**.

e) Click **App Transaction dashboard**.
Integrate Citrix ADC with multiple Citrix ADC Observability Exporter instances manually

You can also configure Citrix ADC Observability Exporter manually. We recommend deploying Citrix ADC Observability Exporter in an automated way with the YAML file as described in the preceding sections. You can also perform manual configuration for Citrix ADC in the MPX and VPX form factors.

```
1  enable feature appflow
2  enable ns mode ULFD
3  add dns nameserver <KUBE-CoreDNS>
4  add server COEsvr <FQDN/IP>
5  add servicegroup COEsvcgrp LOGSTREAM -autoScale DNS
6  bind servicegroup COEsvcgrp COEsvr <PORT>
7  add lb vserver COE LOGSTREAM 0.0.0.0 0
8  bind lb vserver COE COEsvcgrp
9  add analytics profile web_profile -collectors COE -type webinsight -httpURL ENABLED -httpHost ENABLED -httpMethod ENABLED -httpUserAgent ENABLED -httpContentType ENABLED
```
Add Citrix ADC Observability Exporter using FQDN

```bash
enable feature appflow
enable ns mode ULFD
add dns nameserver <KUBE-CoreDNS>
add server COEsvr <FQDN>
add servicegroup COEsvcgrp LOGSTREAM -autoScale DNS
bind servicegroup COEsvcgrp COEsvr <PORT>
add lb vserver COE LOGSTREAM 0.0.0.0 0
bind lb vserver COE COEsvcgrp
add analytics profile web_profile -collectors COE -type webinsight -httpURL ENABLED -httpHost ENABLED -httpMethod ENABLED -httpUserAgent ENABLED -httpContentType ENABLED
add analytics profile tcp_profile -collectors COE -type tcpinsight
bind lb vserver <WEB-VSERVER> -analyticsProfile web_profile
bind lb vserver <WEB-VSERVER> -analyticsProfile tcp_profile

# To enable metrics push to prometheus
add service metrichost_SVC <IP> HTTP <PORT>
set analyticsprofile ns_analytics_time_series_profile -collectors metrichost_SVC -metrics ENABLED -outputMode prometheus
```

To verify if Citrix ADC sends application data logs to Citrix ADC Observability Exporter:

```bash
nsconmsg -g listream_tot_trans_written -d current
```

The counter value indicates that the number of application transactions (for example, HTTP transactions) which have been sent to Citrix ADC Observability Exporter.
If the application traffic rate (for example, HTTP req/sec) that is sent to Citrix ADC Observability Exporter is not equal to `lstream_tot_trans_written`, you can verify the same using the following command:

```
nsconmsg -g nsstream_err_ulf_data_not_sendable -d current
```

The counter value indicates that Citrix ADC cannot send the data to Citrix ADC Observability Exporter due to network congestion, unavailability of network bandwidth, and so on. The data is stored in the available buffers.

Information about various transaction data and individual fields, and their datatype are available in the following location on the Citrix ADC:

```
shell/netscaler/appflow/ns_ipfix.yaml
```

To verify that if application transaction records are exported from Citrix ADC to Citrix ADC Observability Exporter, use the following command:

```
nsconmsg -g appflow_tmpl -d current
```
Location of metrics data export logs to Citrix ADC for time series data:

```
/var/nslog/metrics_prom.log
```

To verify Elasticsearch related counters, run the following command:

```
kubectl exec -it <cpx-pod-name> [-c <cpx-container-name>] [-n <namespace-name>] -- bash
tail -f /var/ulflog/counters/lstrmd_counters_codes.log | grep -iE "\(http_reqs_done|elk\)"
```

Find the logs in the following location to verify that the Citrix ADC Observability Exporter configuration is applied correctly:

```
vi /var/logproxy/lstreamd/conf/lstreamd.conf
```

If Citrix ADC Observability Exporter fails, you can collect logs and files available at the following location and contact Citrix Support.

```
/cores/ (Location of the coredump files, if any.)
/var/ulflog/ (Location of the `libulfd` logs and counter details.)
/var/log (Location of the console logs, lstreamd logs and so on.)
```

For information on troubleshooting related to Citrix ADC Observability Exporter, see [Citrix ADC CPX troubleshooting](https://support.citrix.com/article/CTX115238).
Citrix ADC Observability Exporter with Kafka as endpoint

February 3, 2022

Citrix ADC Observability Exporter is a container that collects metrics and transactions from Citrix ADC. It also transforms the data into the formats (such as AVRO) that are supported in Kafka and exports the data to the endpoint. Kafka is an open-source and distributed event streaming platform for high-performance data pipelines and streaming analytics.

Deploy Citrix ADC Observability Exporter

You can deploy Citrix ADC Observability Exporter using the YAML file. Based on the Citrix ADC deployment, you can use Citrix ADC Observability Exporter to export metrics and transaction data from Citrix ADC. You can deploy Citrix ADC CPX either as a pod inside the Kubernetes cluster or on Citrix ADC MPX or VPX form factor outside the cluster.

The following diagram illustrates a Citrix ADC as an Ingress Gateway with Citrix ADC Observability Exporter as a sidecar. It sends Citrix ADC application transaction data to Kafka.

Prerequisites

- Ensure that you have a Kubernetes cluster with kube-dns or CoreDNS addon enabled.
- Ensure that the Kafka server is installed and configured.
- You must have a Kafka broker IP or FQDN address.
- You must have defined a Kafka topic HTTP.
- Ensure that you have Kafka Consumer to verify the data.
Note:
In this example scenario, the YAML file is used to deploy Citrix ADC Observability Exporter in the Kubernetes default namespace. If you want to deploy in a private Kubernetes namespace other than default, edit the YAML file to specify the namespace.

The following is a sample application deployment procedure.

Note:
If you have a pre-deployed web application, skip the step 1 and 2.

1. Create a secret ingress.crt and key ingress.key using your own certificate and key.
   In this example, a secret, called ing in the default namespace, is created.

```bash
kubectl create secret tls ing --cert=ingress.crt --key=ingress.key
```

2. Access the YAML file from webserver-kafka.yaml to deploy a sample application.

```bash
ekubectl create -f webserver-kafka.yaml
```

3. Define the specific parameters that you must import by specifying it in the ingress annotations of the application's YAML file using the smart annotations in ingress.

```yaml
ingress.citrix.com/analyticsprofile: '{
  "webinsight": {
    "httpurl":"ENABLED", "httpuseragent":"ENABLED", "httpHost":"
      ENABLED","httpMethod":"ENABLED","httpContentType":"ENABLED" }
  }
}
```

Note:
The parameters are predefined in the webserver-kafka.yaml file.

For more information about Annotations, see ingress annotations documentation.

**Deploy Citrix ADC CPX with the Citrix ADC Observability Exporter support**

You can deploy Citrix ADC CPX as a side car with the Citrix ADC Observability Exporter support. You can edit the Citrix ADC CPX YAML file, cpx-ingress-kafka.yaml, to include the configuration infor-
Citrix Observability Exporter

Information that is required for Citrix ADC Observability Exporter support.

Perform the following steps to deploy a Citrix ADC CPX instance with the Citrix ADC Observability Exporter support:

1. Download the `cpx-ingress-kafka.yaml` and the `cic-configmap.yaml` files.
2. Create a ConfigMap with the required key-value pairs and deploy the ConfigMap. You can use the `cic-configmap.yaml` file that is available, for the specific endpoint, in the directory.
3. Modify Citrix ADC CPX related parameters, as required.
4. Edit the `cic-configmap.yaml` file and specify the following variables for Citrix ADC Observability Exporter in the `NS_ANALYTICS_CONFIG` endpoint configuration.

```yaml
server: 'coe-kafka.default.svc.cluster.local' # COE service FQDN
```

5. Deploy Citrix ADC CPX with the Citrix ADC Observability Exporter support using the following commands:

```bash
kubectl create -f cpx-ingress-kafka.yaml
kubectl create -f cic-configmap.yaml
```

**Note:**

If you have used a different namespace, other than `default`, then you must change from `coe-kafka.default.svc.cluster.local` to `coe-kafka.<desired-namespace>.svc.cluster.local`.

**Deploy Citrix ADC Observability Exporter using YAML**

You can deploy Citrix ADC Observability Exporter using the YAML file. Download the `coe-kafka.yaml` file that you can use for the Citrix ADC Observability Exporter deployment.

To deploy Citrix ADC Observability Exporter using the Kubernetes YAML, run the following command in the Kafka endpoint:

```bash
kubectl create -f coe-kafka.yaml
```

To edit the YAML file for the required changes, perform the following steps:

1. Edit the ConfigMap using the following YAML definition:
Note:
Ensure that you specify the Kafka broker IP and the Kafka desired topic.

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: coe-config-kafka
data:
  lstreamd_default.conf: |
  |
    "Endpoints": {
      "KAFKA": {
        "ServerUrl": "X.X.X.X:9092", #Specify the Kafka broker IP
        "KafkaTopic": "HTTP", #Specify the desired kafka topic
        "RecordType": {
          "HTTP": "all",
          "TCP": "all",
          "SWG": "all",
          "VPN": "all",
          "NGS": "all",
          "ICA": "all",
          "APPFW": "none",
          "BOT": "none",
          "VIDEOOPT": "none",
          "BURST_CQA": "none",
          "SLA": "none",
          "MONGO": "none"
        }
      ,
      "ProcessAlways": "yes",
      "FileSizeMax": "40",
      "ProcessYieldTimeOut": "500",
      "FileStorageLimit": "1000",
      "SkipAvro": "no",
      "AvroCompress": "yes"
    }
```
2. Specify the host name and IP or FQDN address of the Kafka nodes. Use the following YAML definition for a three node Kafka cluster:

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: coe-kafka
labels:
  app: coe-kafka
spec:
  replicas: 1
  selector:
    matchLabels:
      app: coe-kafka
  template:
    metadata:
      name: coe-kafka
      labels:
        app: coe-kafka
    spec:
      hostAliases:
        - ip: "X.X.X.X" # Here we specify kafka node1 Ipaddress
          hostnames:
            - "kafka-node1"
        - ip: "Y.Y.Y.Y" # Here we specify kafka node2 Ipaddress
          hostnames:
            - "kafka-node2"
        - ip: "Z.Z.Z.Z" # Here we specify kafka node3 Ipaddress
          hostnames:
            - "kafka-node3"
      containers:
        - name: coe-kafka
          image: "quay.io/citrix/citrix-observability-exporter:1.3.001"
          imagePullPolicy: Always
          ports:
            - containerPort: 5557
              name: lstream
          volumeMounts:
```
3. If necessary, edit the service configuration for exposing the Citrix ADC Observability Exporter port to Citrix ADC using the following YAML definition:

Citrix-observability-exporter headless service:

```yaml
apiVersion: v1
kind: Service
metadata:
  name: coe-kafka
labels:
  app: coe-kafka
spec:
  clusterIP: None
  ports:
    - port: 5557
      protocol: TCP
  selector:
    app: coe-kafka
<!-- NeedCopy-->
```

Citrix-observability-exporter NodePort service

```yaml
apiVersion: v1
kind: Service
metadata:
  name: coe-kafka-nodeport
labels:
  app: coe-kafka
```
Citrix Observability Exporter

```yaml
spec:
  type: NodePort
  ports:
  - port: 5557
    protocol: TCP
  selector:
    app: coe-kafka
```

Verify the Citrix ADC Observability Exporter deployment

To verify the Citrix ADC Observability Exporter deployment, perform the following:

1. Verify the deployment using the following command:

```
kubectl get deployment,pods,svc -o wide
```

2. Access the application with a browser using the URL: `https://<kubernetes-node-IP>:<cpx-ingress-kafka nodeport>`. For example, from step 1, access `<http://10.102.61.56:31202>` in which, `10.102.61.56` is one of the Kubernetes node IPs.
3. Use Kafka Consumer to view the transaction data. Access kafka Consumer from PythonKafka-Consumer.

The following image shows sample data from Kafka Consumer.

Integrate Citrix ADC with multiple Citrix ADC Observability Exporter instances manually

You can configure Citrix ADC Observability Exporter manually in Citrix ADC. Manual configuration is suitable for Citrix ADC in MPX and VPX form factors. Citrix recommends deploying Citrix ADC Observability Exporter in the automated way using the YAML file as described in the preceding sections.

For information about deploying Citrix ADC Observability Exporter (coe-kafka.yaml) and web application (webserver-kafka.yaml), see the preceding sections.
Add Citrix ADC Observability Exporter using FQDN

```bash
enable feature appflow
enable ns mode ULFD
add dns nameserver <KUBE-CoreDNS>
add server COEsvr <FQDN>
add servicegroup COEsvcgrp LOGSTREAM -autoScale DNS
bind servicegroup COEsvcgrp COEsvr <PORT>
add lb vserver COE LOGSTREAM 0.0.0.0 0
bind lb vserver COE COEsvr <PORT>
add analytics profile web_profile -collectors COE -type webinsight -
  httpURL ENABLED -httpHost ENABLED -httpMethod ENABLED -httpUserAgent
  ENABLED -httpContentType ENABLED
add analytics profile tcp_profile -collectors COE -type tcpinsight
bind lb vserver <WEB-VSERVER> -analyticsProfile web_profile
bind lb vserver <WEB-VSERVER> -analyticsProfile tcp_profile
# To enable metrics push to prometheus
add service metrichost_SVC <IP> HTTP <PORT>
set analyticsprofile ns-analytics_time_series_profile -collectors
  metrichost_SVC -metrics ENABLED -outputMode prometheus
```

© 1999–2022 Citrix Systems, Inc. All rights reserved.
Citrix Observability Exporter

For information on troubleshooting related to Citrix ADC Observability Exporter, see Citrix ADC CPX troubleshooting.

**Citrix ADC Observability Exporter with Splunk Enterprise as endpoint**

August 18, 2022

Citrix ADC Observability Exporter is a container that collects metrics and transactions from Citrix ADC and sends the data to various endpoints. Citrix ADC Observability Exporter supports Splunk Enterprise as an endpoint.

Splunk Enterprise is a data platform for searching, monitoring, and analyzing machine-generated big data. Splunk Enterprise captures indexes and correlates real-time data in a repository from which it can generate reports, graphs, dashboards, and visualizations.

You can add Splunk Enterprise as an endpoint to receive audit logs, events, and transactions from Citrix ADC for analysis. Splunk Enterprise provides a graphical representation of these data. You can enable or disable the type of transactions, events, and audit logs which are to be sent to Splunk Enterprise.

**Deploy Citrix ADC Observability Exporter**

You can deploy Citrix ADC Observability Exporter using the YAML file. Based on your Citrix ADC deployment, you can deploy Citrix ADC Observability Exporter either outside or inside Kubernetes clusters. You can deploy Citrix ADC Observability Exporter as a pod inside the Kubernetes cluster or on the Citrix ADC MPX or VPX appliance outside the cluster.

The following diagram illustrates a Citrix ADC as an Ingress Gateway with the Citrix ingress controller and Citrix ADC Observability Exporter as sidecars. Citrix ADC Observability Exporter sends Citrix ADC application metrics and transaction data to Splunk Enterprise. Splunk Enterprise provides a graphical representation of the data.
Prerequisites

- Ensure that you have a Kubernetes cluster with kube-dns or CoreDNS addon enabled.

Note:
In the following procedure, the YAML file is used to deploy Citrix ADC Observability Exporter in the Kubernetes default namespace. If you want to deploy in a private namespace other than the default, edit the YAML file to specify the namespace.

Perform the following steps to deploy Citrix ADC Observability Exporter:

Note:
If you have a pre-deployed web application, skip the steps 1 and 2.

1. Create a secret ingress.crt and key ingress.key using your own certificate and key.
   In this example, a secret, called ing in the default namespace, is created.

   ```bash
   kubectl create secret tls ing --cert=ingress.crt --key=ingress.key
   ```

2. Access the YAML file from webserver-splunk.yaml to deploy the application.

   ```bash
   kubectl create -f webserver-splunk.yaml
   ```

3. Define the specific parameters that you must import by specifying it in the ingress annotations of the application’s YAML file, using the smart annotations in the ingress.
Deploy Citrix ADC CPX with the Citrix ADC Observability Exporter support

You can deploy Citrix ADC CPX as a side car with the Citrix ADC Observability Exporter support enabled along with Citrix ingress controller. You can modify the Citrix ADC CPX YAML file `cpx-ingress-splunk.yaml` to include the configuration information that is required for the Citrix ADC Observability Exporter support.

The following is a sample application deployment procedure.

1. Download the `cpx-ingress-splunk.yaml` and `cic-configmap.yaml` file.
2. Create a ConfigMap with the required key-value pairs and deploy the ConfigMap. You can use the `cic-configmap.yaml` file that is available, for the specific endpoint, in the directory.
3. Modify Citrix ADC CPX related parameters, as required.
4. Edit the `cic-configmap.yaml` file and specify the following variables for Citrix ADC Observability Exporter in the `NS_ANALYTICS_CONFIG` endpoint configuration.

   ```yaml
   server: 'coe-splunk.default.svc.cluster.local' # COE service FQDN
   
   Note:
   If you have used a namespace other than default, change `coe-splunk.default.svc.cluster.local` to `coe-splunk.<desired-namespace>.svc.cluster.local`. If Citrix ADC is outside the Kubernetes cluster, then you must specify IP address and nod-port address of Citrix ADC Observability Exporter.
   
   5. Deploy Citrix ADC CPX with the Citrix ADC Observability Exporter support using the following commands:
Citrix Observability Exporter

```
1  kubectl create -f cpx-ingress-splunk.yaml
2  kubectl create -f cic-configmap.yaml
```

**Deploy Citrix ADC Observability Exporter using the YAML file**

You can deploy Citrix ADC Observability Exporter using the YAML file. Download the YAML file from `coe-splunk.yaml`. Ensure to specify the Splunk server address for the right namespace.

**Note:**

While deploying Citrix ADC Observability Exporter using the YAML file, along with the Splunk server address, you can provide the **Index** name to which the data to be sent in Splunk Enterprise. By default, this **IndexPrefix** option is empty and the data is uploaded to the default index, that is `main`, in Splunk Enterprise.

To deploy Citrix ADC Observability Exporter using the Kubernetes YAML, run the following command in the Splunk Enterprise endpoint:

```
1  kubectl create -f coe-splunk.yaml
```

**Note:**

Modify the YAML file for Citrix ADC Observability Exporter if you have a custom namespace.

**Verify the Citrix ADC Observability Exporter deployment**

You can verify the deployment after deploying Citrix ADC Observability Exporter, web application, Citrix ADC CPX, and Citrix ingress controller.

To verify the deployment, perform the following steps:

1. **Verify the deployment using the following command:**

   ```
   1  kubectl get deployment,pods,svc -o wide
   ```

2. **Access the application using a browser with the URL.**
   
   For example:

   ```
   1  https://kubernetes-node-IP:cpx-ingress-splunk_nodeport/
   ```
3. Access the Splunk server using a browser with the URL. For example:

```
1 https://splunk-node-IP:splunk nodeport/
```

Import pre-built dashboards for Splunk

You can import pre-built Splunk dashboards provided by Citrix. The JSON files for importing the dashboards are available at the GitHub repository. These dashboards provide you the option to filter the transactions based on parameters such as an instance IP address, application name, or client and server IP address and so on.

Following is a sample HTTP dashboard. This dashboard shows data such as HTTP header-based charts, transactional latency, response type distribution, and so on.
Following is a sample TCP dashboard for Splunk. This dashboard shows data such as bandwidth distribution for each application, TCP Jitter, client and server RTT, and so on.

Citrix ADC Observability Exporter troubleshooting

December 23, 2021

This document explains how to troubleshoot issues that you may encounter while using Citrix ADC Observability Exporter.

- How do I verify that Citrix ADC sends application data logs to Citrix ADC Observability Exporter?
Run the following command to verify that Citrix ADC sends application data logs to Citrix ADC Observability Exporter:

```
nsconmsg -g lstream_tot_trans_written -d current
```

The counter value indicates that the number of application transactions (for example, HTTP transactions) which have been sent to Citrix ADC Observability Exporter.

If the application traffic rate (for example, HTTP req/sec) that is sent to Citrix ADC Observability Exporter is not equal to `lstream_tot_trans_written`, you can verify the same using the following command:

```
nsconmsg -g nslstream_err_ulf_data_not_sendable -d current
```

The counter value indicates that Citrix ADC cannot send the data to Citrix ADC Observability Exporter due to network congestion, unavailability of network bandwidth, and so on and the data is stored in the available buffers.

Information about various transaction data and individual fields, and their datatype are available in the following location on the Citrix ADC:

```
shell
2 /netscaler/appflow/ns_ipfix.yaml
```

To verify the current record type exported from Citrix ADC to Citrix ADC Observability Exporter, use the following command:

```
nsconmsg -g appflow_tmpl -d current
```

Location of metrics data export logs to Citrix ADC for time series data:

```
/var/nslog/metrics_prom.log
```

To verify kafka related counters, run the following command:
Find the logs in the following location to verify that the Citrix ADC Observability Exporter configuration is applied correctly:

```
vi /var/logproxy/lstreamd/conf/lstreamd.conf
```

If Citrix ADC Observability Exporter fails, you can collect logs available at the following location and contact Citrix Support.

```
1 /cores/ (Location of coredump files, if any.)
2 /var/ulflog/ (Location of `libulfd` logs and counter details.)
3 /var/log (Location of console logs, lstreamd logs and so on.)
```

For information on Citrix ADC CPX related troubleshooting, see Citrix ADC CPX Troubleshooting.