CEM MiNiFi C++ Agent 1.24.05

Using MiNiFi as a Log Collector Pod in Kubernetes

Date published: 2020-10-14
Date modified: 2024-07-30

CLOUDERA

https://docs.cloudera.com/

Legal Notice

Contents

Using MiNiFi as a log collector pod in Kubernetes

Learn how to use MiNiFi as a log collector pod in Kubernetes.

If you have a Kubernetes deployment, which contains some pods, you can see the logs from these pods in a centralized location by using MiNiFi. To do so, you need to set up a log collector pod (in a daemon set) which runs MiNiFi. MiNiFi collects the logs from the other pods, and pushes those logs to the central location you want (for example, Kafka). After the logs are in the central location, they can be searched, archived and so on.

To set up a log collector pod in Kubernetes, you at least need the following:

- A KubernetesControllerService controller service

  You can configure which pods to collect logs from by setting the Namespace Filter, Pod Name Filter, and Cont ainer Name Filter attributes on the KubernetesControllerService. If none of these are set, the default is to collect logs from all pods in the default namespace.
- A TailFile processor with the following properties set:
  - The Attribute Provider Service property set to the name of the KubernetesControllerService
  - The tail-mode property set to Multiple file
  - The File to Tail property set to .*\.log
  - The tail-base-directory property set to

    ```
    /var/log/pods/${namespace}_${pod}_${uid}/${container}
    ```
- Some other processor which uploads the log lines output by the TailFile processor somewhere (for example, PublishKafka)

You can find a sample config.yml file, which contains all these settings, at https://github.com/apache/nifi-minifi-cpp/blob/main/examples/kubernetes_tailfile_config.yml.

The output of the TailFile processor contains single stream flow files, each containing one line of text, from all the pods. On each flow file, the following attributes are set:

- kubernetes.namespace

  The namespace of the pod.
- kubernetes.pod

  The name of the pod.
- kubernetes.uid

  The unique ID of the pod.
- kubernetes.container

  The name of the container inside the pod.
- absolute.path

  The location of the log file on the node; usually something like:

  ```
  /var/log/pods/default_mypod_dd5befc8-5573-40c3-a136-8daf6eb77b01/myconta
  iner/0.log
  ```

You can further use the following processors to modify the output of the TailFile processor:

- The RouteOnAttribute processor to separate the flow files by any of the attributes above.
- The DefragmentText processor to merge multi-line log messages into a single flow file.
- The MergeContent processor to batch multiple log lines into a single flow file.
- The UpdateAttribute processor to create further attributes based on the existing ones.

The log collector pod which runs MiNiFi needs permissions to list namespaces and pods, and MiNiFi needs to run as root so that it can read the log files from other pods. Some volume mounts need to be set up, as well. See https://github.com/apache/nifi-minifi-cpp/tree/main/docker/test/integration/resources/kubernetes/pods-etc for a working example of Kubernetes YAML files which ensure all of this.

Note:

- MiNiFi runs in the log-collector pod; the two hello-world pods are for testing only, and should not be included in your deployment.
- In the list of volume mounts, /var/log/pods is required; the other two are for testing only, and should not be included in your deployment.

As you probably want the log collector pod to run on all nodes in your cluster, Cloudera recommends to run it as a Daemon Set. For more information, see https://kubernetes.io/docs/concepts/workloads/controllers/daemonset/.

Different configurations can be applied in Kubernetes for different log collection use cases. For more information, see https://github.com/apache/nifi-minifi-cpp/tree/main/examples/kubernetes.

For more information about collecting and processing data at the edge, check out the video on the Cloudera Edge Management YouTube playlist: