Instrumenting Java apps in Amazon EKS and non-EKS environments

In your organization, you use Kubernetes to automate deployments, scale, and manage containers. However, maintaining the system can be difficult. For that reason, cloud platforms offer Kubernetes as a managed service like Amazon Elastic Kubernetes Service (EKS). Amazon EKS helps deliver highly available and secure Kubernetes clusters in a way that is both fast and simple.

While you can still instrument applications that are running in EKS in a similar way to non-EKS applications, you'll need to make some changes. You'll need to first understand how to instrument apps running in non-EKS environments in order to see what changes are needed.

Deploy a Java application without Kubernetes

1. From the hamburger menu within APM, navigate to the Data Setup wizard.
2. From the APM Instrumentation menu select Java, and click Add Connection.
3. In the Configure Integration section, provide the following details about the environment where the application is deployed:
   • Service: The name of the service you are instrumenting
   • Endpoint: Reflects the endpoint with the Splunk OpenTelemetry Connector installed. By default, the setting is set to `http://localhost:4317`. This can be changed to reflect a remote machine hosting the Splunk OpenTelemetry Connector.
   • Environment: The name for the environment that you are instrumenting
   • Collect Metrics: (Experimental) If enabled, gathers additional JVM-specific metrics
   • Kubernetes: If the application is running within Kubernetes.Legacy
   • Smart Agent: Whether or not you would like to use the legacy SignalFx agent
4. If you have not already done so, click Install the Splunk OpenTelemetry Connector and follow the steps shown on the guided wizard.
5. Install the integration by downloading the latest version of the Splunk Distribution of OpenTelemetry for Java with this curl command to add to your runtime command:

   ```bash
   ```
6. Do one of the following:
   • Copy and paste the environmental variables to help identify your application (service) and associate it to the environment of your choice.

The information provided in Splunk Lantern is intended for informational and educational purposes only. All information is provided in good faith, however, Splunk disclaims any and all representations and warranties, express and implied, regarding the information provided, including without limitation any warranties and representations regarding the completeness, adequacy or accuracy of the information. You agree to take full responsibility for the results arising from the use of the information provided.
You can also skip adding the environmental variables on your host system and add them to your service and environment details of your application in the next step when updating your runtime:

```java
java -Dexec.executable="java"
-Dotel.resource.attributes=service.name=<SERVICENAMEHERE>,deployment.environment=<ENVIRONMENTNAMEHERE>
-Dsplunk.metrics.enabled=false -javaagent:splunk-otel-javaagent.jar -jar
curlapp-1.1-SNAPSHOT.jar
```

7. Update your runtime command to include the Splunk Distribution of OpenTelemetry Java. Here is an example of how you can update this command in a non-EKS (Kubernetes) deployment:

```java
java -javaagent:./splunk-otel-javaagent.jar -jar curlapp-1.1-SNAPSHOT.jar
```

This runtime reflects manually adding your service (application) name and environment name manually, skipping the addition of environmental variables on the host system.

You can now interact with your data by clicking **Explore Service Spans**.

The environment name is visible in this area.

---

**Deploy a Java application with Amazon EKS**

The process to instrument your Java application with Amazon EKS only changes slightly from deploying without Kubernetes, due to how applications are deployed with Kubernetes.

Deploying with Amazon EKS will also work the same when deploying your application to a Kubernetes cluster outside of EKS.

1. From the hamburger menu within APM, navigate to the Data Setup wizard.
2. In the Configure Integration section, select **Yes** for Kubernetes, then click **Next**.
3. In the Install Integration section, install the Splunk OpenTelemetry Connector on your Kubernetes cluster if you have not done this already. To verify if the Splunk OpenTelemetry Connector is installed on your Amazon EKS cluster, **update kubeconfig** to work with your Amazon EKS cluster and run the following command:

   ```bash
   kubectl get pods
   ```

   The output should show both the agent and cluster receiver pods if the Splunk OpenTelemetry Collector is installed.

4. Use the curl command listed previously to download the splunk-otel-javaagent.jar to include with your runtime command within your containerized application. Here is an example Dockerfile, which copies the splunk-otel-javaagent.jar of the Java application container and executes the application within the run.sh bash script.

   ```bash
   FROM openjdk:8-jre-alpine3.9
   COPY target/curlapp-1.1-SNAPSHOT.jar /curlapp-1.1-SNAPSHOT.jar
   COPY splunk-otel-javaagent.jar /splunk-otel-javaagent.jar
   ```

---

The information provided in Splunk Lantern is intended for informational and educational purposes only. All information is provided in good faith, however, Splunk disclaims any and all representations and warranties, express and implied, regarding the information provided, including without limitation any warranties and representations regarding the completeness, adequacy or accuracy of the information. You agree to take full responsibility for the results arising from the use of the information provided.
COPY run.sh /run.sh
ENTRYPOINT ["sh", "/run.sh"]

Contents of run.sh:

```bash
java -javaagent:/splunk-otel-javaagent.jar -jar /curlapp-1.1-SNAPSHOT.jar
```

5. Update your deployment file to inject the environment variables necessary to auto instrument your application for your Java application. Here is an example Kubernetes deployment file that reflects the required environmental variables to include the service (application) name and the environment in which it is associated to:

```yaml
apiVersion: v1
derg==
kind: Pod
metadata:
  name: httpgooglechecker
spec:
  containers:
  - name: httpgooglechecker
    image: docker.io/astro7982/curlappstatic:lastest
    env:
      - name: SPLUNK_OTEL_AGENT
        valueFrom:
          fieldRef:
            fieldPath: status.hostIP
      - name: OTEL_RESOURCE_ATTRIBUTES
        value: service.name=<SERVICENAMEHERE>,deployment.environment=<ENVIRONMENTNAMEHERE>
      - name: SPLUNK_METRICS_ENDPOINT
        value: http://$(SPLUNK_OTEL_AGENT):9943
      - name: OTEL_EXPORTER_OTLP_ENDPOINT
        value: http://$(SPLUNK_OTEL_AGENT):4317
    command: ["/bin/sh"]
  args: ["run.sh"]
```

6. Deploy your Java application pod using `kubectl create`.

Within 90 seconds, Splunk APM begins capturing all traces from your application. Both the service name and environment are reflected in the ones entered in your Kubernetes configuration file.

Now that the service is instrumented, you can see RED metrics created automatically, and you can also view every transaction’s full trace through your infrastructure.

Next steps

The content in this guide comes from a previously published blog, one of the thousands of Splunk resources available to help users succeed. In addition, these Splunk resources might help you understand and implement this use case:
- Tech Talk: How to instrument Java applications
- .Conf Talk: Real-world strategies for Kubernetes & Docker integrations with Splunk
- Blog: Strategies for monitoring Docker and Kubernetes environments
- Splunk Docs: Kubernetes cluster
- Tech Talk: Splunk Operator for Kubernetes: Effortlessly deploy, scale, and manage Splunk