Enable Serverless Metrics in Apache OpenWhisk on Kubernetes with Prometheus

Ying Chun Guo, IBM
About me

• GUO, Ying Chun (WeChat: daisy-yuego)
  • 10+ years IBMer
  • Open source developer
  • Focus on Serverless on Kubernetes
  • Committer of Apache OpenWhisk, contributor of Knative
Agenda

• Apache OpenWhisk Overview
• Metrics defined and collected with Kamon
• Metrics stored and displayed with Prometheus
• Demo
Serverless

Increasing focus on business logic

Decreasing concern (and control) over infrastructure implementation

- Faster start-up times
- Better resource utilization
- Finer-grained management
- Splitting up the monolith

- Functions

Virtual machines

Containers

Bare Metal
Properties

- Stateless
- Event Driven
- Auto-scaled / Scale-to-zero
- Short Lived
- Reduced Cost
- Faster Time to Market
To make your own serverless platform

Open Source Serverless Project

Kubernetes
Apache OpenWhisk

A serverless, open source cloud platform that executes functions in response to events at any scale.

Apache OpenWhisk offers:

- **Apache Software Foundation (ASF)**
  - True, community-driven open source (Apache 2 License)
- **Proven on IBM Cloud**
  - Exact, same code in open source
Deploy Apache OpenWhisk on Kubernetes with Helm

- https://github.com/apache/incubator-openwhisk-deploy-kube
<table>
<thead>
<tr>
<th>Use cases</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>micro-service</td>
<td>Easily implement fine-grained, micro-service APIs.</td>
</tr>
<tr>
<td>IoT</td>
<td>Power various mobile, web and IoT app use cases by scaling and simplifying</td>
</tr>
<tr>
<td></td>
<td>the programming model of orchestrating various services.</td>
</tr>
<tr>
<td>Batch and Stream</td>
<td>Automate and control batch and stream processing</td>
</tr>
<tr>
<td>Processing</td>
<td></td>
</tr>
<tr>
<td>DevOps</td>
<td>Automate DevOps pipeline based on events triggered from successful builds</td>
</tr>
<tr>
<td></td>
<td>or completed staging or a go-live event.</td>
</tr>
<tr>
<td>IT/Ops</td>
<td>Allow an easier deployment model for administrative functions (bots) to</td>
</tr>
<tr>
<td></td>
<td>run for IT/Ops.</td>
</tr>
</tbody>
</table>
Metrics is important on Serverless

- Serverless is a new experience to developers, hiding developers from infrastructure details.
- Metrics, or telemetry, is the only way for developers to understand what happens on the server.
- Metrics is useful to understand system health condition.
- Metrics is necessary to enable metering and billing.
OpenWhisk distinguishes between system and user metrics

- **System metrics** typically contain information about system performance
  - collected by Kamon
  - usually used by providers/operators.

- **User metrics** encompass information about action performance.
  - Sent to Kafka in a form of events
  - Consumed by OpenWhisk users
  - could also used for billing or audit purposes
Kamon is an open source monitoring framework for applications running on the Java Virtual Machine (JVM) with integrations for Scala and Akka.

- Powerful metrics, distributed tracing and context propagation APIs in a single library.
- Provide different metric recording instruments in its core metrics API.
- Switch reporters at any time without having to change your instrumentation.
- Works with Prometheus, Zipkin, InfluxDB, Kamon APM and other commercial and OSS solutions.
Kamon Metrics Instruments

- **Counter**: counts how many times it was incremented during a reporting period. Good for counting errors or occurrences of specific events in your service.
- **Gauges**: track a single value that can be incremented, decremented or explicitly set. Good for slow moving variables, like available memory and disk usage.
- **Histograms**: track the entire value distribution of a given metric.
- **Timer**: allows you to start() the timer and later stop() the StartedTimer instance.
- **Range Samplers**: internally tracks three variables: the current value, the minimum and the maximum observed value.
OpenWhisk system metrics are emitted from within Controller and Invoker, monitoring activations, memory usage, Kafka, database, HTTP requests and etc. There are 60+ metrics till now.

- **Counter**
  - Records the count of activations sent to Kafka.
  - Records the count of non blocking activations started.

- **Gauges**
  - Records the number of activations being worked upon for a given controller
  - Records the amount of RAM memory in use for in flight activations.

- **Histograms**
  - Current memory capacity for all usable managed and blackbox invokers
  - Kafka topic to receive activations to complete.
Prometheus, a Cloud Native Computing Foundation project, is a systems and service monitoring system. It collects metrics from configured targets at given intervals, evaluates rule expressions, displays the results, and can trigger alerts if some condition is observed to be true.

- a multi-dimensional data model (timeseries defined by metric name and set of key/value dimensions)
- a flexible query language to leverage this dimensionality
- no dependency on distributed storage; single server nodes are autonomous
- timeseries collection happens via a pull model over HTTP
- pushing timeseries is supported via an intermediary gateway
- targets are discovered via service discovery or static configuration
- multiple modes of graphing and dashboarding support
- support for hierarchical and horizontal federation
Kamon-Prometheus Exporter

- An open source project to provide the integration Kamon with Prometheus
- With a very simple statements, the metrics collected by Kamon will be converted to Prometheus format and exported to a http address.

```java
import kamon.prometheus.PrometheusReporter
Kamon.addReporter(new PrometheusReporter());
```

```yaml
global:
  scrape_interval: 1s
scrape_configs:
- job_name: 'kamon'
  static_configs:
  - targets: ['owdev-controller.openwhisk.svc.cluster.local:8080']
```

https://github.com/kamon-io/kamon-prometheus
Kamon and Prometheus

OpenWhisk Controller

- Controller
- Kamon
- Kamon-Prometheus Exporter

Prometheus

- Retrieval
- PromQL
- Storage

Port 8080
## Targets

**kamon (2/2 up)**

<table>
<thead>
<tr>
<th>Endpoint</th>
<th>State</th>
<th>Labels</th>
<th>Last Scrape</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://owdev-invoker.openwhisk.svc.cluster.local:8080/metrics">http://owdev-invoker.openwhisk.svc.cluster.local:8080/metrics</a></td>
<td>UP</td>
<td>instance=&quot;owdev-invoker.openwhisk.svc.cluster.local:8080&quot;</td>
<td>11.27s ago</td>
</tr>
</tbody>
</table>