Seamless Integration - Take Kubernetes into your Existing Monitoring Stack
Agenda

• About Me
• Background
• Problem
• Solution
• Solution scenarios
About Me

• Iris Ding (丁少君- dingsj@cn.ibm.com)

• IBMer

• Rich Experience on Middleware production development experience

• Rich Experience on Open Source development

• Apache CXF committer

• Current focus on Service Mesh, Istio
Background
Background
- Short-lived vs Long-running
- Dynamic
- Monolithic applications  --  Microservices
- Large amount of Metrics
- Central view of whole system status
Solution 1 – Self-maintained
Solution 1 – Self Maintained

- Resource usages for whole cluster
- Resource usage for pod
- Resource usage for every containers in the cluster
- Resource usage for nodes
- Node status, Pod status, Node Number, Pod Number
- Cluster events
- Specific metrics via executing command
Solution 1 – Self Maintained

- Available metrics:
  - Specific container’s resource usage includes CPU, memory, disk and load average of CPU.
  - Active container counts
  - Resource usage for whole containers on a host
<table>
<thead>
<tr>
<th>Issue</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>No cAdvisor</td>
<td>Install cAdvisor as daemon set</td>
</tr>
<tr>
<td>cAdvisor port is disabled</td>
<td>Enable cAdvisor port (KUBELET_CADVISOR_ARGS=--cadvisor-port=0)</td>
</tr>
</tbody>
</table>
Solution 1 - Dashboard
Solution 2 - Prometheus

Prometheus

PromQL

PromAdapter

Web hook metrics collection

Metrics Bus (Kafka)

streaming

consumer

Monitoring Service

```json
{
  "prometheus": {
    "max_read": 5,
    "max_conn": 5,
    "customLabel": {
      "org": "monitor_org",
      "env": "monitor_env",
      "cluster": "monitor_cluster"
    }
  },
  "promMetrics": [
    {
      "metric": "container_cpu_usage_seconds_total",
      "prefix": "container_cpu_usage_seconds_total",
    },
    {
      "metric": "container_memory_working_set_bytes",
      "prefix": "container_memory_working_set_bytes"
    },
    {
      "metric": "container_network_transmit_bytes_total",
      "prefix": "container_network_transmit_bytes_total"
    },
    {
      "metric": "container_fs_usage_bytes",
      "prefix": "container_fs_usage_bytes"
    }
  ],
  "interval": 60
}
```
Solution 3 - Prometheus

Prometheus

Remote_write → PromAdapter

Raw Metrics

Web hook metrics collection

Metrics Bus (Kafka)

Consumer

Monitoring Service

```
global:  
  scrape_interval: 5s  
  remote_write:  
```
Solution 4 – Exporter Only

API Server

Custom Controller

Scrape targets

Worker

Worker

Web hook metrics collection

Metrics Bus (Kafka)

streaming consumer

Monitoring Service

```
if _, ok := svc.GetAnnotations()["prometheus.io/scrape"]; ok {
    annotations := svc.GetAnnotations()
    promScrape := annotations["prometheus.io/scrape"]
    if promScrape == "true" {
        var {
            subpath = "/metrics"
            port = "9090"
        }
        if _subpath, ok := svc.GetAnnotations()["prometheus.io/path"]; ok {
            subpath = _subpath
        }
        if _port, ok := svc.GetAnnotations()["prometheus.io/port"]; ok {
            port = _port
        }
        return &scrape.Target{
            Source: key,
            URL: fmt.Sprintf("http://%v:%v", name, namespace, port, subpath),
        }, true
    }
}
```
<table>
<thead>
<tr>
<th>Solution</th>
<th>Pros</th>
<th>Cons</th>
<th>User cases</th>
</tr>
</thead>
<tbody>
<tr>
<td>Self-maintained</td>
<td>• No extra Maintenance effort</td>
<td>• Self development</td>
<td>No Prometheus</td>
</tr>
<tr>
<td>Prometheus with Remote-write</td>
<td>• No performance penalty</td>
<td>• Need change config</td>
<td>Prometheus</td>
</tr>
<tr>
<td>Prometheus with Query</td>
<td>• Has Performance penalty</td>
<td>• No need to change config</td>
<td>Prometheus</td>
</tr>
<tr>
<td>Exporters Only</td>
<td>• No performance penalty • No Maintenance effort</td>
<td>• Rely on exporters</td>
<td>No Prometheus But with exporters</td>
</tr>
</tbody>
</table>
Alert Integration

- Monitoring/Auto-resolve
  - Service1
  - Service2

Alerts Flow

Prometheus

Web hook for downstream

Alert TIP (Kafka)

consumer

Event correlation

Auto-healing engine

Alert sending engine

KubeCon CloudNativeCon China 2018
THANKS!

Iris Ding
dingsj@cn.ibm.com
irisdingbj@gmail.com