链滴

k8s 使用 jmx_prometheus_javaagent 采集 jvm 指标

作者：fish2018

原文链接：https://ld246.com/article/1603166987077

来源网站：链滴

许可协议：署名-相同方式共享 4.0 国际 (CC BY-SA 4.0)

背景

skywalking采集的jvm要自己在页面选择endpoint来查看，不合符开发者使用习惯

前置知识

prometheus-operator的四个CRD作用

● Prometheus: 由 Operator 依据一个自定义资源kind: Prometheus类型中，所描述的内容而部署的 Prometheus Server 集群，可以将这个自定义资源看作是一种特别用来管理Prometheus Server的StatefulSets资源。

● ServiceMonitor: 一个Kubernetes自定义资源(和kind: Prometheus一样是CRD)，该资源描述了Prometheus Server的Target列表，Operator 会监听这个资源的变化来动态的更新Prometheus Server Scrape targets并让prometheus server去reload配置(prometheus有对应reload的http接口/-/reload)。而该资源主要通过Selector来依据 Labels 选取对应的Service的endpoints，并让 Prometheus Server 通过 Service 进行拉取（拉）指标资料(也就是metrics信息),metrics信息要在http的url输出符合metrics格式的信息,ServiceMonitor也可以定义目标的metrics的url.

● Alertmanager：Prometheus Operator 不只是提供 Prometheus Server 管理与部署，也包含了 AlertManager，并且一样通过一个 kind: Alertmanager 自定义资源来描述信息，再由 Operator 依据描述内容部署 Alertmanager 集群。

● PrometheusRule:对于Prometheus而言，在原生的管理方式上，我们需要手动创建Prometheus告警文件，并且通过在Prometheus配置中声明式的加载。而在Prometheus Operator模式中，告警则也编程一个通过Kubernetes API 声明式创建的一个资源.告警规则创建成功后，通过在Prometheus中使用想servicemonitor那样用ruleSelector通过label匹配选择需要关联的PrometheusRule即可

什么是 JMX Exporter ?

JMX Exporter 利用 Java 的 JMX 机制来读取 JVM 运行时的一些监控数据，然后将其转换为 Prometeus 所认知的 metrics 格式，以便让 Prometheus 对其进行监控采集。

那么，JMX 又是什么呢？它的全称是：Java Management Extensions。 顾名思义，是管理 Java 的种扩展框架，JMX Exporter 正是基于此框架来读取 JVM 的运行时状态的。

如何使用 JMX Exporter 暴露 JVM 监控指标？

JMX-Exporter 提供了两种用法:

1. 启动独立进程。JVM 启动时指定参数，暴露 JMX 的 RMI 接口，JMX-Exporter 调用 RMI 获取 JV 运行时状态数据，转换为 Prometheus metrics 格式，并暴露端口让 Prometheus 采集。

2. JVM 进程内启动(in-process)。JVM 启动时指定参数，通过 javaagent 的形式运行 JMX-Exporter 的 jar 包，进程内读取 JVM 运行时状态数据，转换为 Prometheus metrics 格式，并暴露端口让 Prometheus 采集。

实施

1. 下载

地址：https://github.com/prometheus/jmx_exporter

```
wget https://repo1.maven.org/maven2/io/prometheus/jmx/jmx_prometheus_javaagent/0.14./jmx_prometheus_javaagent-0.14.0.jar
```

2. 配置文件

jmx-cfg.yml

```
startDelaySeconds: 0
lowercaseOutputName: false
lowercaseOutputLabelNames: false
whitelistObjectNames: ["org.apache.cassandra.metrics:*"]
blacklistObjectNames: ["org.apache.cassandra.metrics:type=ColumnFamily,*"]
rules:
  - pattern: 'org.apache.cassandra.metrics<type=(\w+), name=(\w+)><>Value: (\d+)'
    name: cassandra_$1_$2
    value: $3
    valueFactor: 0.001
    labels: {}
    help: "Cassandra metric $1 $2"
    type: GAUGE
    attrNameSnakeCase: false
```

3. javaagent启动方式

启动参数格式: -javaagent:<jar>=<port>:<config>

```
-javaagent:jmx_prometheus_javaagent-0.14.0.jar=12345:jmx-cfg.yml
```

4. 在rancher中启用集群级别监控prometheus-operator

5. 给prometheus-operator的service account配置RBAC

如果监控的服务不在同一个namespace 那么会出现无权限获取的问题

jxm-rbac.yml

```
apiVersion: rbac.authorization.k8s.io/v1beta1
kind: ClusterRole
metadata:
  name: prometheus
rules:
- apiGroups: [""]
  resources:
  - nodes
  - services
  - endpoints
  - pods
  verbs: ["get", "list", "watch"]
- apiGroups: [""]
  resources:
  - configmaps
  verbs: ["get"]
- nonResourceURLs: ["/metrics"]
  verbs: ["get"]
---
apiVersion: rbac.authorization.k8s.io/v1beta1
kind: ClusterRoleBinding
metadata:
  name: prometheus
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: prometheus
subjects:
- kind: ServiceAccount
  name: prometheus-k8s
```

```
  namespace: monitoring
```

6. 如果你想直接利用集群的Prometheus，那么你得把ServiceMonitor建在 cattle-prometheus下并且设置namespaceSelector属性

jmx-servicemonitor.yml

```
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
  name: jmx-servicemonitor
  namespace: cattle-prometheus # rancher集群级别监控的namespace
spec:
  endpoints:
  - port: jmx-metrics
    path: /metrics
    interval: 5s
  namespaceSelector:
    matchNames:
    - dev-demo # 采集指定namespace
  selector:
    matchLabels:
      jmx: "true" # service中需要配置相同label才会被采集
```

如果希望ServiceMonitor可以关联任意命名空间下的标签，则通过以下方式定义：

```
spec:
  namespaceSelector:
    any: true
```

如果不使用prometheus-operator

```
- job_name: dev-demo
  scrape_interval: 5s
  kubernetes_sd_configs:
  - role: endpoints
    namespaces:
      names:
      - default
      - dev-demo
  relabel_configs:
  - action: keep
    source_labels:
    - __meta_kubernetes_service_label_app
    regex: tomcat
  - action: keep
    source_labels:
    - __meta_kubernetes_endpoint_port_name
    regex: jmx-metrics
```

7. 配置service

demo-service.yml

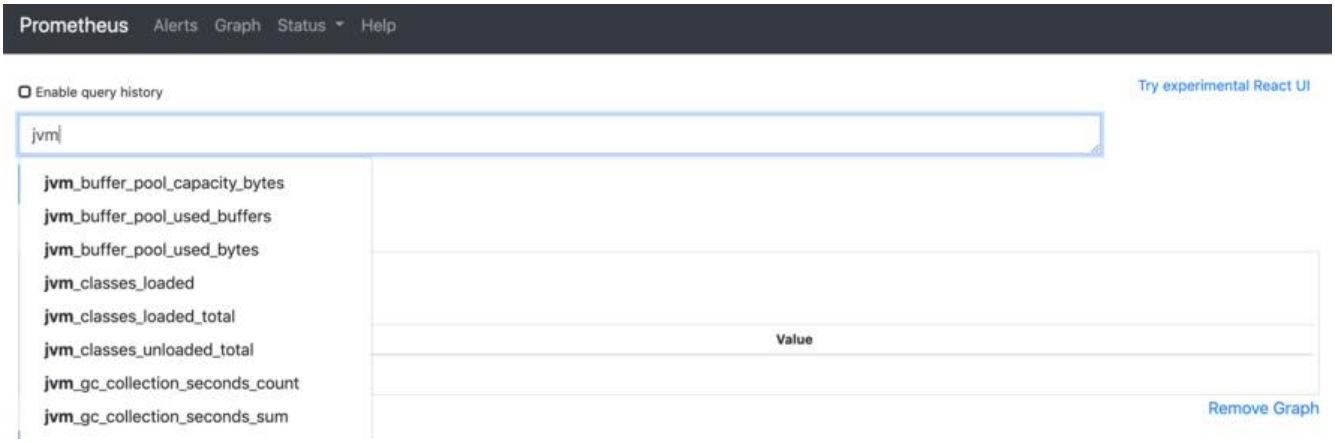
添加和servicemonitor相匹配的label，暴露metrics端口

```yaml
apiVersion: v1
kind: Service
metadata:
  name: dev-demo
  namespace: dev-demo
  labels:
    jmx: "true" # 和servicemonitor的matchLabels一致
spec:
  ports:
  - name: jmx-metrics # 与jmx启动时端口一致
    port: 12345
    protocol: TCP
    targetPort: 12345
  - name: app-port
    port: 8888
    protocol: TCP
    targetPort: 8888
  selector:
    app: dev-demo
  type: NodePort
```

或者通过注解配置

```yaml
apiVersion: v1
kind: Service
metadata:
  name: dev-demo
  namespace: dev-demo
  annotations:
        prometheus.io/scrape: "true"
        prometheus.io/jvm-scrape: "true"
        prometheus.io/jvm-port: "12345"
        prometheus.io/jvm-path: "/metrics"
spec:
  ports:
  - port: 12345
  selector:
    app: dev-demo
```

可以在prometheus查看指标是否采集成功



总的来说就是创建自己的监控数据需要ServiceMonitor，通过ServiceMonitor去对应的label里找对应

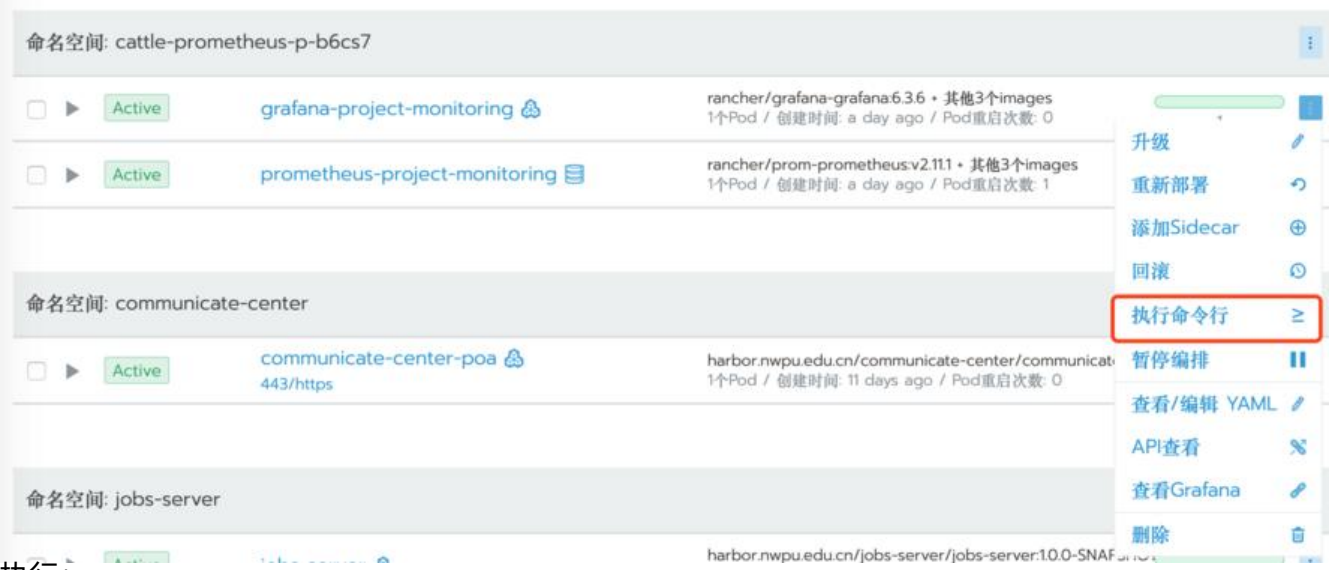
原文链接：k8s 使用 jmx_prometheus_javaagent 采集 jvm 指标

vc下的metrics的port。并且不同的namespace下的监控需要设置好rbac权限

8. 给Grafana添加JVM Dashboard

地址：https://grafana.com/grafana/dashboards/8878

你需要给Grafana添加JVM Dashboard，在这之前你需要设置Grafana的admin密码，进入项目找到Grafana，进入其Shell：

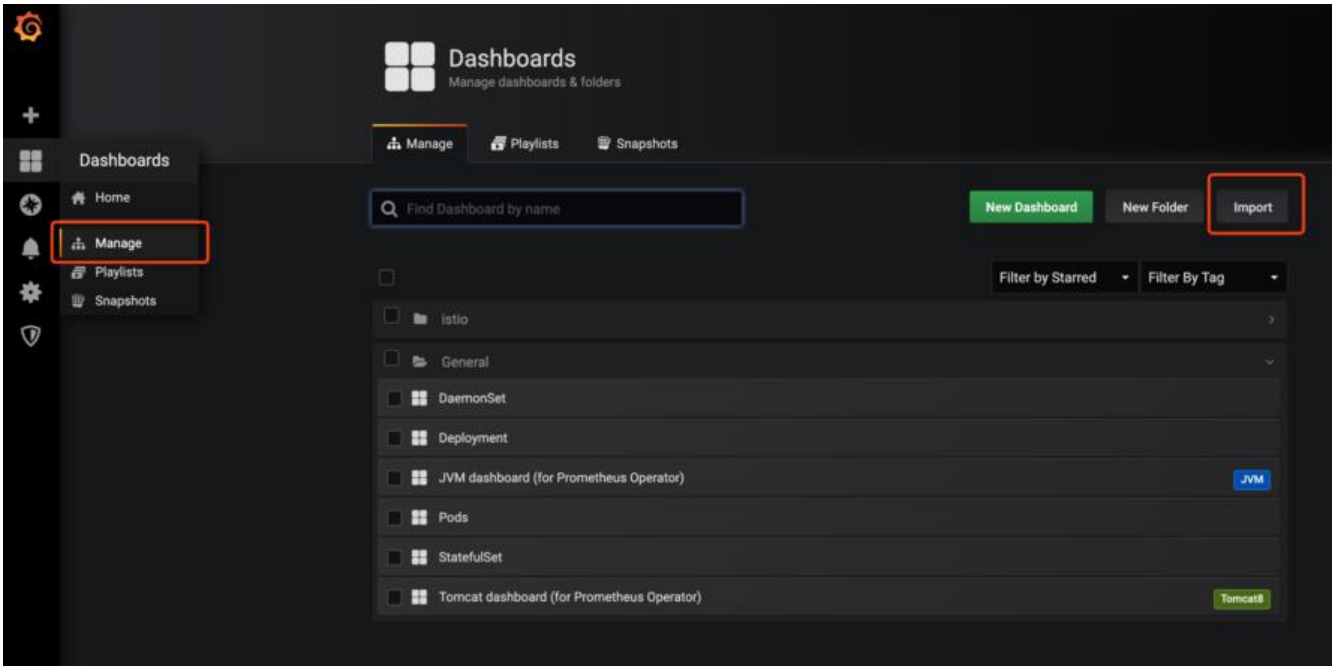


执行：

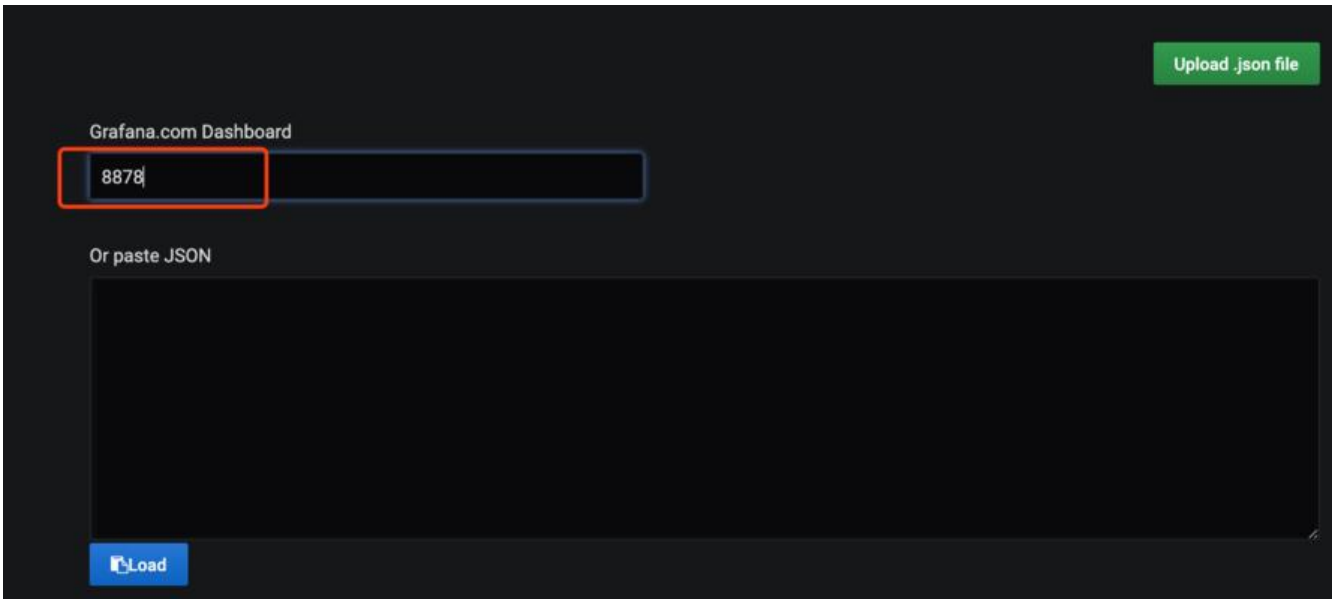
grafana-cli admin reset-admin-password <新密码>

然后随便进入一个Deployment/StatefulSets，进入Grafana：

用admin账号和你刚才设置的密码登录进去，进入管理页面导入Dashboard：



到 https://grafana.com/orgs/chanjarster/dashboards找到 JVM dashboard (for Prometheus Operator)，看到它的编号是8878。把这个编号填到导入页面：



直接导入的面板有些地方无法正常显示，需要进行修正和定制

已修正定制微服务面板、集群节点面板

微服务.json.zip

集群节点.json.zip

参考链接

https://chanjarster.github.io/post/k8s/rancher-p8s-jvm/

https://blog.csdn.net/qq_22543991/article/details/88405322

https://blog.csdn.net/yunxiao6/article/details/109045399