本小节将展示如何在[Kubernetes](http://kubernetes.io/)上使用[StatefulSet](http://kubernetes.io/docs/concepts/abstractions/controllers/statefulsets/)，去编排部署、管理和监控一个由3个节点组成的安全模式的CockroachDB集群。

如果集群只是用做测试环境而非生产环境，对安全性没有要求，可以选择非安全模式。

**TIPS:** 了解更多CockroachDB运行在Kubernetes上潜在的性能瓶颈，以及如何进行优化，查看：[CockroachDB Performance on Kubernetes](http://doc.cockroachchina.baidu.com/#deploy/orchestrated-deployment/kubernetes-performance-optimization/)

前言

Kubernetes术语

| **英文** | **中文** | **简介** |
| --- | --- | --- |
| instance | 实例 | 物理或虚拟的机器，本小节将使用GCE或AWS机器，组成一个Kubernetes集群。 |
| [pod](http://kubernetes.io/docs/user-guide/pods/) | - | 一个或多个容器作为一个单元方便管理。本小节一个pod运行一台独立的机器上，每个pod包含一个容器，每个容器运行一个CockroachDB节点。 |
| [StatefulSet](http://kubernetes.io/docs/concepts/abstractions/controllers/statefulsets/) | - | StatefulSet是一组有状态的pod，其中每个pod具有不同的网络身份，在重启的时候绑定同一个物理存储位置。该功能自Kubernetes 1.5版本推出beta版本以来，在1.9版本达到稳定。 |
| [persistent volume](http://kubernetes.io/docs/user-guide/persistent-volumes/) | 持久卷 | 持久卷是mount到一个pod的一块存储空间，其生命周期与pod的生命周期分离，以确保每个CockroachDB节点在重启的时候能够找到其原来的存储空间。  本小节默认使用动态卷配置，若不支持则需要手动配置持久卷。 |
| [CSR](https://kubernetes.io/docs/tasks/tls/managing-tls-in-a-cluster/) | 证书签名请求 | 请求获取有Kubernetes集群内置CA证书签名的TLS证书。每个pod创建的时候，pod会为内部CockroachDB节点发送一个CSR，需要手动检查和批准。同理，访问集群的客户端也需要做上述操作。 |
| [RBAC](https://kubernetes.io/docs/admin/authorization/rbac/) | 基于角色的访问权限控制 | Kubernetes用于管理集群访问控制权限的系统，用户产生的动作都需要获得相应的角色权限。本小节配置了CockroachDB创建和访问证书所需的RBAC。 |

限制

Kubernetes版本

本小节使用的配置文件需要Kubernetes版本不低于1.8，更早的版本存在部分配置不支持的问题。如需更早版本的配置文件，可以在[以前的归档文档](https://github.com/cockroachdb/cockroach/tree/master/cloud/kubernetes)中尝试寻找对应版本（例如v1.7）。

存储

目前，在Kubernetes上CockroachDB使用的外部持久卷，服务提供商往往会提供冗余备份服务。由于CockroachDB已经具备数据冗余的特性，因此服务提供商的备份冗余是不必要的，且往往会影响性能。因此在StatefulSets在支持节点本地存储之前，若追求高性能，可以考虑使用DaemonSet部署方式。

Step 1: 启动Kubernetes

根据使用机器类型，采用不同的配置方式：

托管的GKE

（ Hosted Google Kubernetes Engine）

* 完成[Google Kubernetes Engine Quickstart](https://cloud.google.com/kubernetes-engine/docs/quickstart)文档的**Before You Begin**步骤，包括安装用于创建和删除Kubernetes引擎节点的gcloud和方便本地机器上管理Kubernetes集群的命令行工具kubectl。

**TIPS:** 文档提供了使用Google云shell产品和在本地机器上使用shell两种选择，如果需要查看CockroachDB Admin界面可以选择后者。 - 在本地机器上启动Kubernetes集群

gcloud container clusters create cockroachdb

Creating cluster cockroachdb...done.

该命令创建了GKE实例机器并将他们加入到Kubernetes集群管理，集群名字为cockroachdb。

* 获取Google云账号相关的邮箱地址

gcloud info | grep Account

Account: [your.google.cloud.email@example.org]

**WARNING:** 该命令返回的邮箱地址是小写的，而接下来的步骤输入邮箱是大小写敏感，需要邮箱地址大小写一致。

* [创建CockroachDB运行GKE需要的RBAC角色](https://cloud.google.com/kubernetes-engine/docs/how-to/role-based-access-control#prerequisites_for_using_role-based_access_control)，邮箱地址为上一步获取的内容。

kubectl create clusterrolebinding $USER-cluster-admin-binding --clusterrole=cluster-admin --user=<your.google.cloud.email@example.org>

clusterrolebinding "cluster-admin-binding" created

手动管理的GCE

根据[Running Kubernetes on Google Compute Engine](https://kubernetes.io/docs/setup/turnkey/gce/)文档，安装软件依赖，启动Kubernetes集群。包括：

* 创建Google云平台账户，安装gcloud和其他依赖
* 下载安装最新版Kubernetes程序
* 创建GCE实例机器并加入到Kubernetes集群当中
* 安装kubectl工具用于远程管理集群

AWS

根据[Running Kubernetes on AWS EC2](https://kubernetes.io/docs/setup/turnkey/aws/)文档，安装软件依赖，启动Kubernetes集群。

Step 2: 启动CockroachDB节点

在本地机器上使用cockroachdb-statefulset-secure.yaml配置文件去创建StatefuleSet，后者自动创建3个pods，每个pod运行一个CockroachDB节点。

kubectl create -f https://raw.githubusercontent.com/cockroachdb/cockroach/master/cloud/kubernetes/cockroachdb-statefulset-secure.yaml

serviceaccount "cockroachdb" created

role "cockroachdb" created

clusterrole "cockroachdb" created

rolebinding "cockroachdb" created

clusterrolebinding "cockroachdb" created

service "cockroachdb-public" created

service "cockroachdb" created

poddisruptionbudget "cockroachdb-budget" created

statefulset "cockroachdb" created

Step 3: 确认节点验证

每个pod创建的时候，pod会为内部CockroachDB节点发送一个CSR，获取有Kubernetes集群内置CA证书签名的TLS证书，这需要手动检查和批准。

* 获取pod 1 Pending状态的CSR名字

kubectl get csr

NAME AGE REQUESTOR CONDITION

default.node.cockroachdb-0 1m system:serviceaccount:default:default Pending

node-csr-0Xmb4UTVAWMEnUeGbW4KX1oL4XV\_LADpkwjrPtQjlZ4 4m kubelet Approved,Issued

node-csr-NiN8oDsLhxn0uwLTWa0RWpMUgJYnwcFxB984mwjjYsY 4m kubelet Approved,Issued

node-csr-aU78SxyU69pDK57aj6txnevr7X-8M3XgX9mTK0Hso6o 5m

如果存在Pending状态的CSR，需要等待几分钟并再次查询。

* 检查pod 1的CSR

kubectl describe csr **default**.node.cockroachdb-0

Name: **default**.node.cockroachdb-0

Labels: <none>

Annotations: <none>

CreationTimestamp: Thu, 09 Nov 2017 13:39:37 -0500

Requesting User: system:serviceaccount:default:**default**

Status: Pending

Subject:

Common Name: node

Serial Number:

Organization: Cockroach

Subject Alternative Names:

DNS Names: localhost

cockroachdb-0.cockroachdb.**default**.svc.cluster.local

cockroachdb-**public**

IP Addresses: 127.0.0.1

10.48.1.6

Events: <none>

* 确认无误后接受pod 1的CSR

kubectl certificate approve default.node.cockroachdb-0

certificatesigningrequest "default.node.cockroachdb-0" approved

* 为其他pod同样执行上述步骤

Step 4: 初始化集群

* 确认所有pods是否成功运行，避免初始化集群的时候存在ready状态的的pod。

kubectl get pods

**NAME** READY **STATUS** RESTARTS AGE

cockroachdb-0 0/1 Running 0 m

cockroachdb-1 0/1 Running 0 m

cockroachdb-2 0/1 Running 0 m

* 确认为所有pod成功创建持久卷和响应链。

kubectl get persistentvolumes

NAME CAPACITY ACCESSMODES RECLAIMPOLICY STATUS CLAIM REASON AGE

pvc-52f51ecf-8bd5-11e6-a4f4-42010a800002 1Gi RWO Delete Bound default/datadir-cockroachdb-0 26s

pvc-52fd3a39-8bd5-11e6-a4f4-42010a800002 1Gi RWO Delete Bound default/datadir-cockroachdb-1 27s

pvc-5315efda-8bd5-11e6-a4f4-42010a800002 1Gi RWO Delete Bound default/datadir-cockroachdb-2 27s

* 使用cluster-init-secure.yaml配置文件一次性初始化，将节点添加到集群。

kubectl create -f https://raw.githubusercontent.com/cockroachdb/cockroach/master/cloud/kubernetes/cluster-init-secure.yaml

job "cluster-init-secure" created

* 确认用户执行初始化操作的一次性pod的CSR。

kubectl certificate approve default.client.root

certificatesigningrequest "default.client.root" approved

* 确认初始化集群成功

kubectl get job cluster-init-secure

NAME DESIRED SUCCESSFUL AGE

cluster-init-secure 1 1 2m

kubectl get pods

NAME READY STATUS RESTARTS AGE

cockroachdb-0 1/1 Running 0 3m

cockroachdb-1 1/1 Running 0 3m

cockroachdb-2 1/1 Running 0 3m

**TIPS:** 该StatefulSet配置CockroachD节点的标准输出重定向到stderr，如果你需要通过pod/node的日志去排除故障，请用kubectl logs <podname>命令，而不是检查持久卷的日志。

Step 5: 使用内置SQL客户端

启动一个用于运行cockroach命令以开启内置的SQL客户端的pod，需要检查和确认该pod的CSR。

* 在本地机器上使用client-secure.yaml配置文件去加载pod，并保持该pod一直运行。该pod使用先前创建的root客户端验证去初始化集群，因此不需要CSR确认。

kubectl create -f https://raw.githubusercontent.com/cockroachdb/cockroach/master/cloud/kubernetes/client-secure.yaml

pod "cockroachdb-client-secure" created

* 在pod中开启CockroachDB内置的[SQL客户端](http://doc.cockroachchina.baidu.com/#quick-start/learn-cockroachdb-sql/use-the-built-in-sql-client/)

kubectl exec -it cockroachdb-client-secure -- ./cockroach sql --certs-dir=/cockroach-certs --host=cockroachdb-public

*# Welcome to the cockroach SQL interface.*

*# All statements must be terminated by a semicolon.*

*# To exit: CTRL + D.*

*#*

*# Server version: CockroachDB CCL v1.1.2 (linux amd64, built 2017/11/02 19:32:03, go1.8.3) (same version as client)*

*# Cluster ID: 3292fe08-939f-4638-b8dd-848074611dba*

*#*

*# Enter \? for a brief introduction.*

*#*

root@cockroachdb-public:26257/>

* 执行[SQL语句](https://www.cockroachlabs.com/docs/stable/learn-cockroachdb-sql.html)进行测试

**CREATE** **DATABASE** bank;

**CREATE** **TABLE** bank.accounts (id INT **PRIMARY** **KEY**, balance DECIMAL);

**INSERT** **INTO** bank.accounts **VALUES** (1, 1000.50);

**SELECT** \* **FROM** bank.accounts;

+*----+---------+*

| id | balance |

+*----+---------+*

| 1 | 1000.5 |

+*----+---------+*

(1 row)

**TIPS:** 该pod将一直运行。后续需要重启开启SQL客户端或是执行[cockroach客户端命令](http://doc.cockroachchina.baidu.com/#deploy/cockroach-commands/)（例如cockroach node或cockroach zone）时，可以重复使用该pod。  
用户也可以删除pod并在需要的时候再创建，执行kubectl delete pod cockroachdb-client-secure

Step 6: 访问Admin界面

* 本地机器映射端口到其中一个pod

kubectl port-forward cockroachdb-0 8080

Forwarding from 127.0.0.1:8080 -> 8080

**NOTE:** 该命令需要在登录Admin界面的机器上执行，否则无法登录Admin界面。

* 登录Admin界面，URL地址：https://localhost:8080/
* 在网页上查看集群状况：
  + 点击右边**查看节点列表**标签页，查看是否所有节点正常加入到集群
  + 点击左边**数据库**标签页，查看bank数据库

Step 7: 模拟节点丢失

基于StatefulSet配置文件中replicas: 3，Kubernetes能够保证有3个pods/nodes在集群中运行。一旦其中一个节点丢失，Kubernetes自动为丢失节点创建具有同样网络身份和持久卷的pod/node。

删除CockroachDB节点

kubectl delete pod cockroachdb-2

pod "cockroachdb-2" deleted

查看节点情况

登录Admin界面，切换到Summary界面，可以看到其中一个节点丢失，Kubernetes自动重启新的节点，恢复集群健康。

验证pod

kubectl get pod cockroachdb-2

NAME READY STATUS RESTARTS AGE

cockroachdb-2 1/1 Running 0 12s

Step 8: 设置监控和报警

尽管CockroachDB有多种[保障集群可用性的设计](https://www.cockroachlabs.com/docs/stable/high-availability.html)，对于集群健康和性能的持续监控仍然是有必要的，预先针对需要调查和干预的事件创建报警规则，以便出现问题的时候能够第一时间获知并及时处理。

配置Prometheus

CockroachDB集群的每一个节点导出粒化时间序列度量（granular timeseries metrics），便于[Prometheus](https://prometheus.io/)集成。Prometheus是开源的存储、聚合和查询时间序列数据。

本小节将展示如何将Prometheus视作Kubernetes集群的一部分进行编排，同时将这些指标引入到Prometheus作为外部监控。下述教程基于[Centos's Prometheus Operator](https://github.com/coreos/prometheus-operator/blob/master/Documentation/user-guides/getting-started.md)，允许使用Kubernetes概念管理Prometheus实例。

**NOTE:** 请事先参照 Step 1:启动Kubernetes 的内容，确保将Google云账号关联的邮箱添加到cluster-adminRBAC组。

添加标签

在本地机器上编辑cockroachdb服务，添加prometheus:cockroachdb标签：

kubectl label svc cockroachdb prometheus=cockroachdb

service "cockroachdb" labeled

该命令确保了prometheus工作和只监控cockroachdb服务的数据。

安装prometheus-operator并检查

* 安装[CoreOS's Prometheus Operator](https://raw.githubusercontent.com/coreos/prometheus-operator/release-0.20/bundle.yaml)

kubectl apply -f https://raw.githubusercontent.com/coreos/prometheus-operator/release-0.20/bundle.yaml

clusterrolebinding "prometheus-operator" created

clusterrole "prometheus-operator" created

serviceaccount "prometheus-operator" created

deployment "prometheus-operator" created

* 检查prometheus-operator是否启动

kubectl get deploy prometheus-operator

NAME DESIRED CURRENT UP-TO-DATE AVAILABLE AGE

prometheus-operator 1 1 1 1 1m

创建依赖对象

使用[prometheus.yaml](https://github.com/cockroachdb/cockroach/blob/master/cloud/kubernetes/prometheus/prometheus.yaml)文件创建Prometheus实例所需要的多个对象。

kubectl apply -f https://raw.githubusercontent.com/cockroachdb/cockroach/master/cloud/kubernetes/prometheus/prometheus.yaml

clusterrole "prometheus" created

clusterrolebinding "prometheus" created

servicemonitor "cockroachdb" created

prometheus "cockroachdb" created

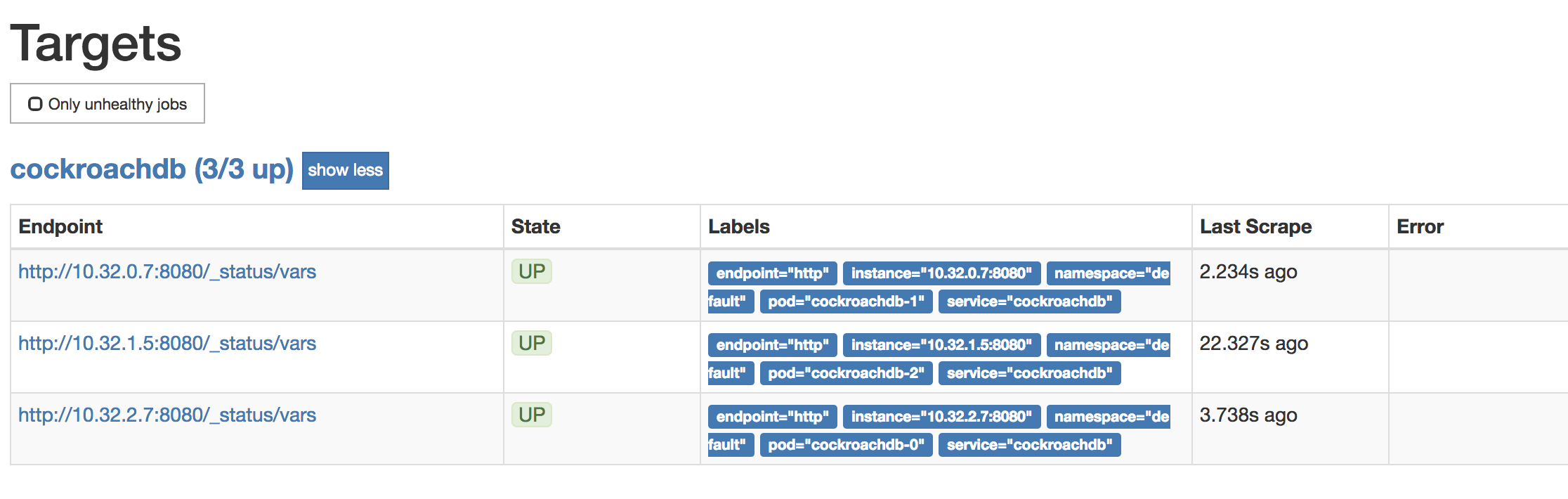
检验数据流

访问Prometheus用户界面，验证CockroachDB是否向Prometheus不停发送数据。

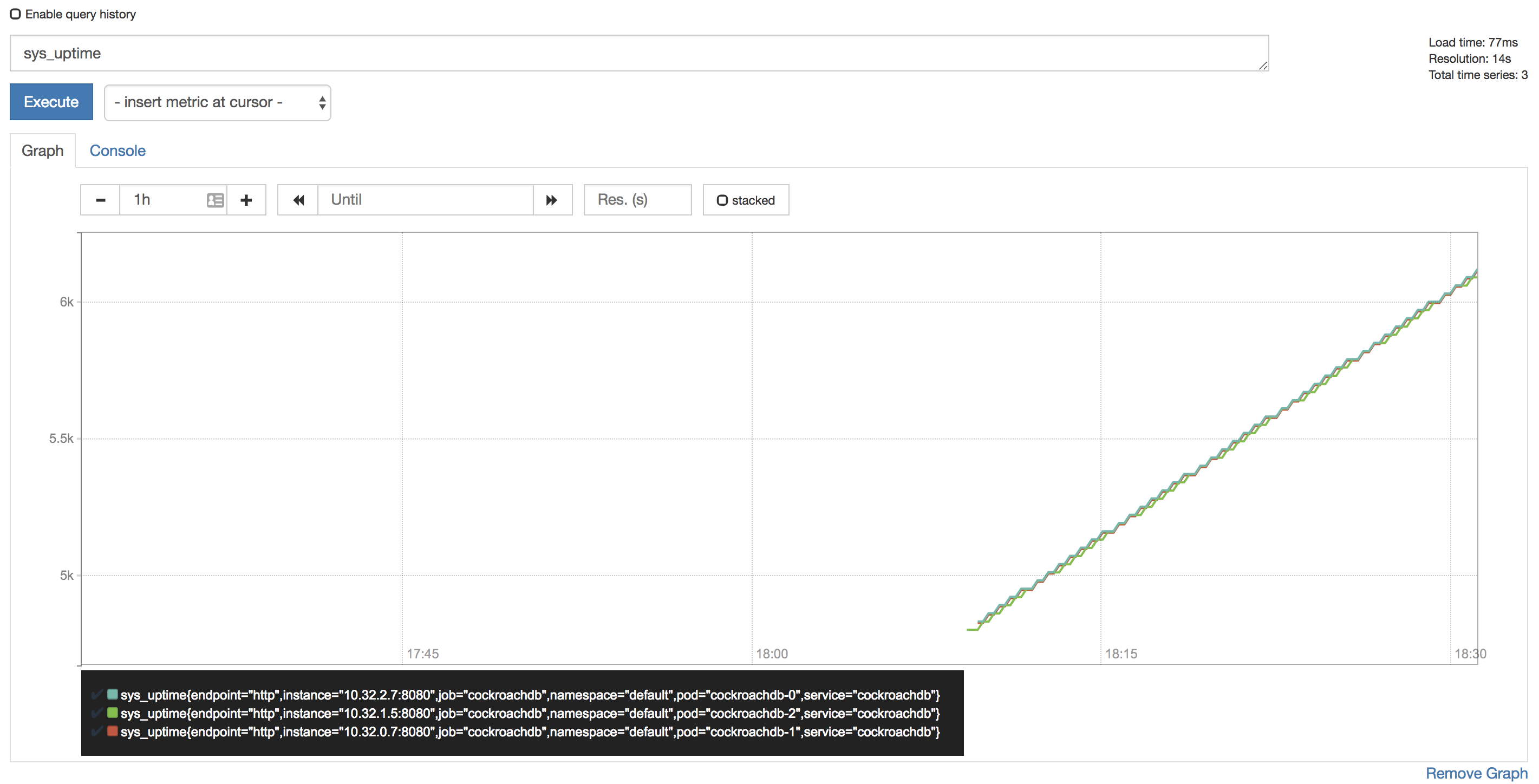
* 操作运行Prometheus的pod与本机做端口映射，映射到本机。

kubectl port-forward prometheus-cockroachdb-0 9090

* 浏览器访问 http://localhost:9090
* 查看**Status > Targets**，确认每个CockroachDB节点与Prometheus相连，如下图所示：



* 查看**Graph**，确认采集数据。输入sys\_uptime变量，**Execute**执行，则在**Graph**输出如下图内容：



**TIPS:** Prometheus自动完成CockroachDB时间序列指标，如果想查看带说明的完整列表，请参照[Access the Admin UI](http://doc.cockroachchina.baidu.com/#deploy/orchestrated-deployment/kubernetes-single-cluster-deployment/#step-6-access-the-admin-ui)进行端口映射后，访问网址：http://localhost:8080/\_status/vars.  
关于Prometheus用户界面使用的更多细节，请查看[官方文档](https://prometheus.io/docs/introduction/getting_started/)。

配置Alertmanager

实时监控能够帮助我们尽早定位问题。与此同时，对可能发生的需要用户调查和干预的事件预先进行设置，并在事件发生时及时报警，也是很重要的。本小节将展示如何使用[Alertmanager](https://prometheus.io/docs/alerting/alertmanager/)和CockroachDB启动器的[alerting rules](https://github.com/cockroachdb/cockroach/blob/master/cloud/kubernetes/prometheus/alert-rules.yaml)来配置报警功能。

下载并编辑配置文件

下载[alertmanager-config.yaml](https://raw.githubusercontent.com/cockroachdb/cockroach/master/cloud/kubernetes/prometheus/alertmanager-config.yaml)配置文件，[配置需要报警的接收器](https://prometheus.io/docs/alerting/configuration/)。

创建Secrets

将此配置作为Secret添加到Kubernetes集群，将其重命名为alertmanager.yaml并对其进行标记以便于查找。

kubectl create secret generic alertmanager-cockroachdb --from-file=alertmanager.yaml=alertmanager-config.yaml

secret "alertmanager-cockroachdb" created

kubectl label secret alertmanager-cockroachdb app=cockroachdb

secret "alertmanager-cockroachdb" labeled

**WARNING:** Secret的名称alertmanager-cockroachdb必须与altermanager.yaml配置文件里一致，不一致则会导致配置文件加载失败，启动无效。

创建依赖对象并验证

使用配置文件[alertmanager.yaml](https://github.com/cockroachdb/cockroach/blob/master/cloud/kubernetes/prometheus/alertmanager.yaml)创建Alertmanager实例所需要的多个对象，包括了集群IP服务（ClusterIP service），其允许Prometheus推送报警信息。

kubectl apply -f https://raw.githubusercontent.com/cockroachdb/cockroach/master/cloud/kubernetes/prometheus/alertmanager.yaml

alertmanager "cockroachdb" created

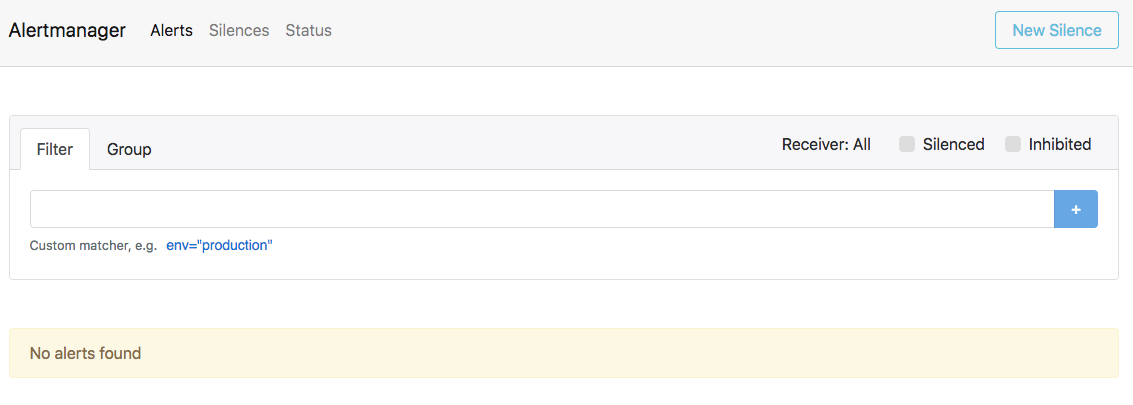
service "alertmanager-cockroachdb" created

查看Alertmanager是否正在运行

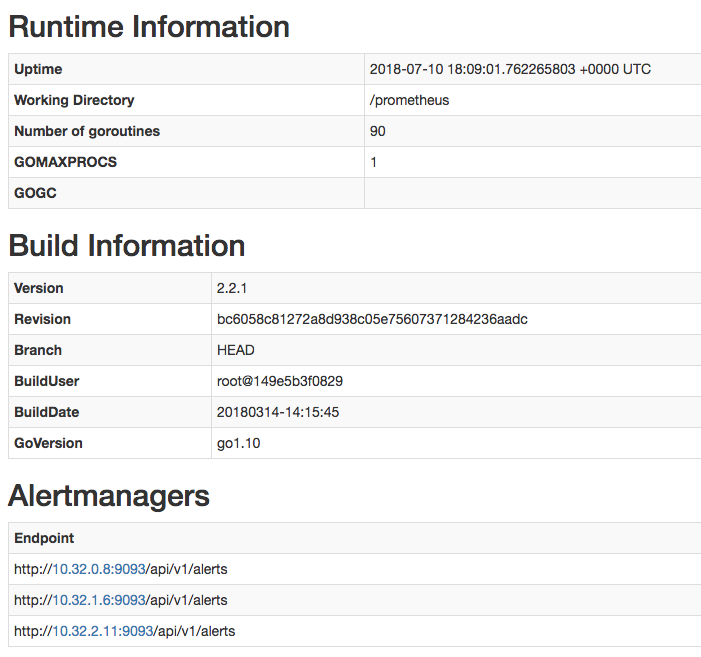
* 操作运行Alertmanager的pod与本机做端口映射，映射到本机。

kubectl port-forward alertmanager-cockroachdb-0 9093

* 打开http://localhost:9093，内容如下：



* 打开http://localhost:9090/status，查看Alertmanager是否连接到Prometheus，如下图：



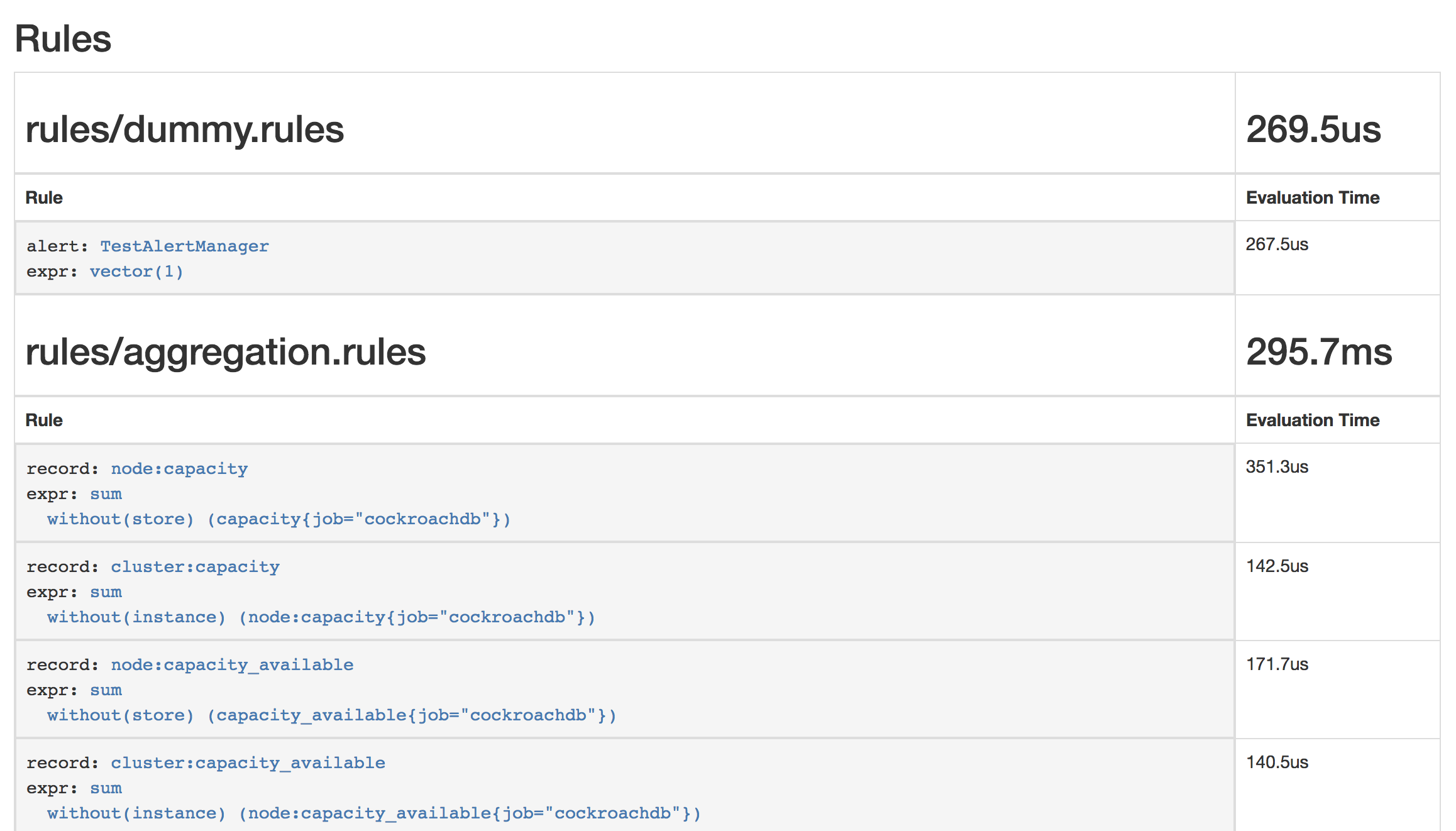
添加规则并验证

* 添加CockroachDB启动报警规则

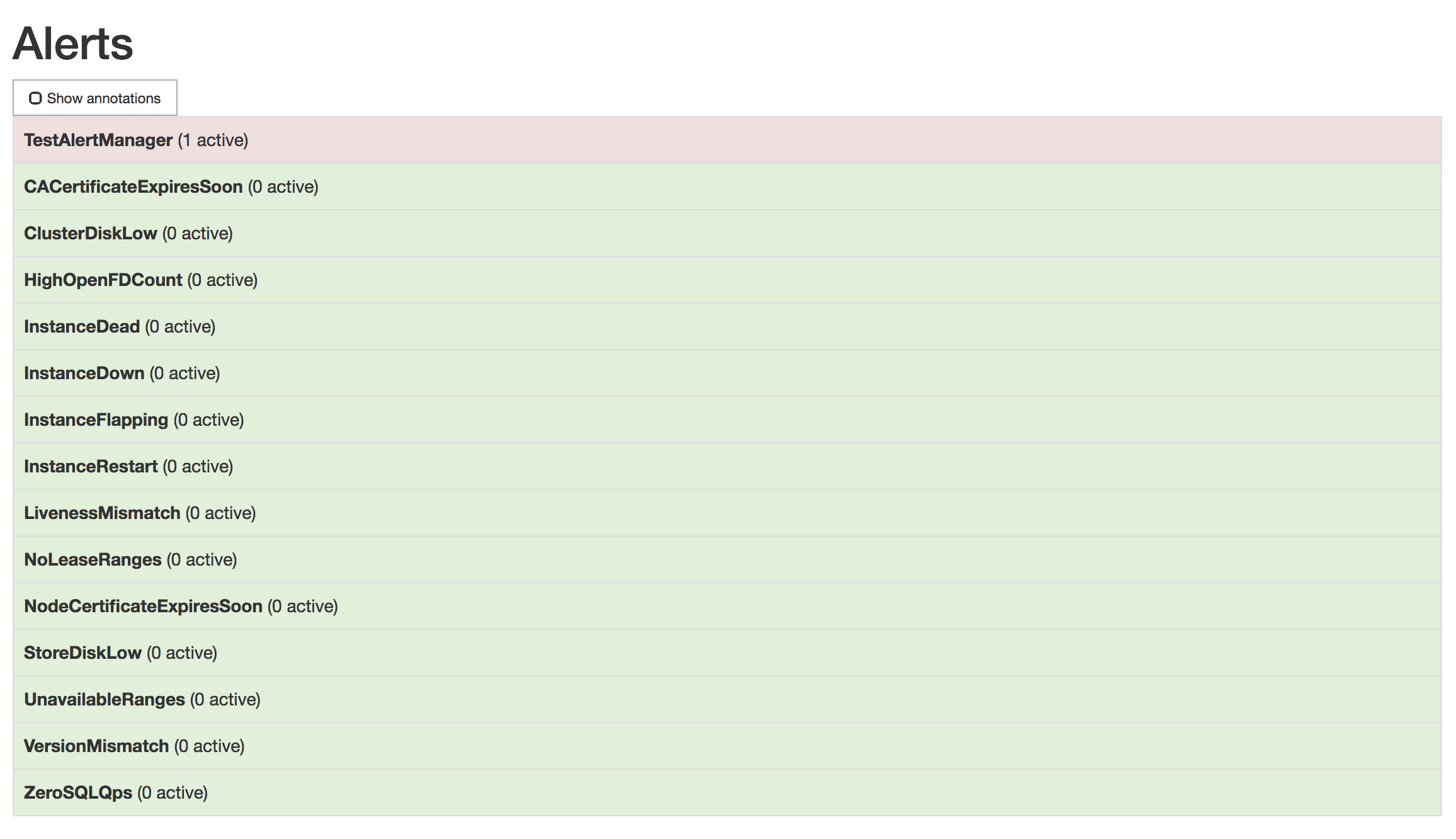
kubectl apply -f https://raw.githubusercontent.com/cockroachdb/cockroach/master/cloud/kubernetes/prometheus/alert-rules.yaml

prometheusrule "prometheus-cockroachdb-rules" created

* 打开http://localhost:9090/rules查看规则是否生效，如下图：



* 打开http://localhost:9090/alerts查看报警触发情况，如下图：



移除规则

* 使用kubectl edit命令编辑规则

kubectl edit prometheusrules prometheus-cockroachdb-rules

* 移除dummy.rules部分并保存文件

- name: rules/dummy.rules

rules:

- alert: TestAlertManager

expr: **vector**(1)

Step 9: 节点维护

拓展集群

Kubernetes集群包含4个节点，其中1个主节点，3个工作节点。pod只部署在工作节点上。为了保证性能，避免2个pods运行在同一个工作节点上，需要添加新的工作节点，再去编辑StatefulSet配置新增pod。

* 添加工作节点
* 如果是GCE，调整[Managed Instance Group](https://cloud.google.com/compute/docs/instance-groups/)
* 如果是AWS，调整[Auto Scaling Group](https://docs.aws.amazon.com/autoscaling/latest/userguide/as-manual-scaling.html)
* 使用kubectl scale命令为StatefulSet新增pod

kubectl scale statefulset cockroachdb --replicas=4

statefulset "cockroachdb" scaled

* 获取处于Pending CSR状态的新pod

kubectl get csr

NAME AGE REQUESTOR CONDITION

default.client.root 1h system:serviceaccount:default:default Approved,Issued

default.node.cockroachdb-0 1h system:serviceaccount:default:default Approved,Issued

default.node.cockroachdb-1 1h system:serviceaccount:default:default Approved,Issued

default.node.cockroachdb-2 1h system:serviceaccount:default:default Approved,Issued

default.node.cockroachdb-3 2m system:serviceaccount:default:default Pending

node-csr-0Xmb4UTVAWMEnUeGbW4KX1oL4XV\_LADpkwjrPtQjlZ4 1h kubelet Approved,Issued

node-csr-NiN8oDsLhxn0uwLTWa0RWpMUgJYnwcFxB984mwjjYsY 1h kubelet Approved,Issued

node-csr-aU78SxyU69pDK57aj6txnevr7X-8M3XgX9mTK0Hso6o 1h kubelet Approved,Issued

* 检查新pod的CSR

kubectl describe csr default.node.cockroachdb-3

Name: default.node.cockroachdb-0

Labels: <none>

Annotations: <none>

CreationTimestamp: Thu, 09 Nov 2017 13:39:37 -0500

Requesting User: system:serviceaccount:default:default

Status: Pending

Subject:

Common Name: node

Serial Number:

Organization: Cockroach

Subject Alternative Names:

DNS Names: localhost

cockroachdb-0.cockroachdb.default.svc.cluster.local

cockroachdb-public

IP Addresses: 127.0.0.1

10.48.1.6

Events: <none>

* 确认无误后接受新pod的CSR

kubectl certificate approve default.node.cockroachdb-3

certificatesigningrequest "default.node.cockroachdb-3" approved

* 查看新pod是否成功运行

kubectl get pods

NAME READY STATUS RESTARTS AGE

cockroachdb-0 1/1 Running 0 51m

cockroachdb-1 1/1 Running 0 47m

cockroachdb-2 1/1 Running 0 3m

cockroachdb-3 1/1 Running 0 1m

cockroachdb-client-secure 1/1 Running 0 15m

* 在Admin界面，查看**Node List**，检查第四个节点是否成功添加到集群。

升级集群

CockroachDB新版本会对旧版本进行bug修复、性能优化以及新功能添加，因此强烈推荐及时更新到最新版本。[General CockroachDB upgrade documentation](http://doc.cockroachchina.baidu.com/#maintain/upgrade-cockroach-version/)文档为用户升级CockroachDB提供了最佳实践方式。如果是在Kubernetes中部署的CockroachDB集群，则在停止和重启进程有一些特别注意的地方。

Kubernetes负责执行CockroachDB节点的安全滚动升级过程。当切换CockroachDB节点的Docker镜像的时候，Kubernetes会依次执行：停止节点，用新镜像重启节点。只有上一个节点重启完成并准备好接受客户端请求，下一个节点才进行操作。详见[Kubernetes documentation](https://kubernetes.io/docs/tutorials/stateful-application/basic-stateful-set/#updating-statefulsets)

* 切换需要的Docker镜像，需要执行以下命令替换“VERSION”到想要的新的版本：

kubectl patch statefulset cockroachdb --type='json' -p='[{"op": "replace", "path": "/spec/template/spec/containers/0/image", "value":"cockroachdb/cockroach:VERSION"}]'

statefulset "cockroachdb" patched

* 如果检查群集的pod的状态，会看到其中一个正在重新启动：

kubectl get pods

NAME READY STATUS RESTARTS AGE

cockroachdb-0 1/1 Running 0 2m

cockroachdb-1 1/1 Running 0 2m

cockroachdb-2 1/1 Running 0 2m

cockroachdb-3 0/1 Terminating 0 1m

* 该过程一直持续到所有pod重启并替换成新的镜像。检查每个pod的镜像可以执行：

kubectl get pods -o jsonpath='{range .items[\*]}{.metadata.name}{"\t"}{.spec.containers[0].image}{"\n"}'

cockroachdb-0 cockroachdb/cockroach:v2.0.5

cockroachdb-1 cockroachdb/cockroach:v2.0.5

cockroachdb-2 cockroachdb/cockroach:v2.0.5

cockroachdb-3 cockroachdb/cockroach:v2.0.5

停止集群

执行以下步骤重启CockroachDB集群：

* 删除跟cockroachDB标签相关联的所有资源，包括日志、远程持久卷、Prometheus和Alertmanager的资源。

kubectl delete pods,statefulsets,services,persistentvolumeclaims,persistentvolumes,poddisruptionbudget,jobs,rolebinding,clusterrolebinding,role,clusterrole,serviceaccount,alertmanager,prometheus,prometheusrule,serviceMonitor -l app=cockroachdb

pod "cockroachdb-0" deleted

pod "cockroachdb-1" deleted

pod "cockroachdb-2" deleted

service "alertmanager-cockroachdb" deleted

service "cockroachdb" deleted

service "cockroachdb-public" deleted

persistentvolumeclaim "datadir-cockroachdb-0" deleted

persistentvolumeclaim "datadir-cockroachdb-1" deleted

persistentvolumeclaim "datadir-cockroachdb-2" deleted

poddisruptionbudget "cockroachdb-budget" deleted

job "cluster-init-secure" deleted

rolebinding "cockroachdb" deleted

clusterrolebinding "cockroachdb" deleted

clusterrolebinding "prometheus" deleted

role "cockroachdb" deleted

clusterrole "cockroachdb" deleted

clusterrole "prometheus" deleted

serviceaccount "cockroachdb" deleted

serviceaccount "prometheus" deleted

alertmanager "cockroachdb" deleted

prometheus "cockroachdb" deleted

prometheusrule "prometheus-cockroachdb-rules" deleted

servicemonitor "cockroachdb" deleted

* 删除用于执行cockroach客户端命令的pod

kubectl delete pod cockroachdb-client-secure

pod "cockroachdb-client-secure" deleted

* 获取集群的CSRs名字

kubectl get csr

NAME AGE REQUESTOR CONDITION

default.client.root 1h system:serviceaccount:default:default Approved,Issued

default.node.cockroachdb-0 1h system:serviceaccount:default:default Approved,Issued

default.node.cockroachdb-1 1h system:serviceaccount:default:default Approved,Issued

default.node.cockroachdb-2 1h system:serviceaccount:default:default Approved,Issued

default.node.cockroachdb-3 12m system:serviceaccount:default:default Approved,Issued

node-csr-0Xmb4UTVAWMEnUeGbW4KX1oL4XV\_LADpkwjrPtQjlZ4 1h kubelet Approved,Issued

node-csr-NiN8oDsLhxn0uwLTWa0RWpMUgJYnwcFxB984mwjjYsY 1h kubelet Approved,Issued

node-csr-aU78SxyU69pDK57aj6txnevr7X-8M3XgX9mTK0Hso6o 1h kubelet Approved,Issued

* 删除创建的CSRs

kubectl delete csr default.client.root default.node.cockroachdb-0 default.node.cockroachdb-1 default.node.cockroachdb-2 default.node.cockroachdb-3

certificatesigningrequest "default.client.root" deleted

certificatesigningrequest "default.node.cockroachdb-0" deleted

certificatesigningrequest "default.node.cockroachdb-1" deleted

certificatesigningrequest "default.node.cockroachdb-2" deleted

certificatesigningrequest "default.node.cockroachdb-3" deleted

* 获取集群Secrets的名字

kubectl get secrets

NAME TYPE DATA AGE

alertmanager-cockroachdb Opaque 1 1h

default-token-d9gff kubernetes.io/service-account-token 3 5h

default.client.root Opaque 2 5h

default.node.cockroachdb-0 Opaque 2 5h

default.node.cockroachdb-1 Opaque 2 5h

default.node.cockroachdb-2 Opaque 2 5h

default.node.cockroachdb-3 Opaque 2 5h

prometheus-operator-token-bpdv8 kubernetes.io/service-account-token 3 3h

* 删除集群Secrets的名字

kubectl delete secrets alertmanager-cockroachdb default.client.root default.node.cockroachdb-0 default.node.cockroachdb-1 default.node.cockroachdb-2 default.node.cockroachdb-3

secret "alertmanager-cockroachdb" deleted

secret "default.client.root" deleted

secret "default.node.cockroachdb-0" deleted

secret "default.node.cockroachdb-1" deleted

secret "default.node.cockroachdb-2" deleted

secret "default.node.cockroachdb-3" deleted

* 停止Kubernetes

gcloud **container** clusters **delete** cockroachdb

**WARNING:** 若停止Kubernetes之前没有删除持久卷，则该资源会在云端一直占用着。