## 一、创建EKS集群

1. 安装helm 3

|  |
| --- |
| curl -LO https://get.helm.sh/helm-v3.4.0-linux-amd64.tar.gz |
| tar -zxvf helm-v3.0.0-linux-amd64.tar.gz |
| mv linux-amd64/helm /usr/local/bin/helm |
| helm help |

1. 安装kubectl

curl -o kubectl <https://amazon-eks.s3.us-west-2.amazonaws.com/1.19.6/2021-01-05/bin/linux/amd64/kubectl>

chmod +x ./kubectl

mkdir -p $HOME/bin && cp ./kubectl $HOME/bin/kubectl && export PATH=$PATH:$HOME/bin

cp ./kubectl /usr/local/bin/

which kubectl

kubectl version

1. eksctl安装

curl --silent --location "https://github.com/weaveworks/eksctl/releases/latest/download/eksctl\_$(uname -s)\_amd64.tar.gz" | tar xz -C /tmp

1. 创建cluster

Vi Cluster-m5.yaml

apiVersion: eksctl.io/v1alpha5

kind: ClusterConfig

metadata:

name: eks11902

region: cn-northwest-1

nodeGroups:

- name: admin

desiredCapacity: 1

labels:

dedicated: admin

instanceType: c5.xlarge

ssh: # use existing EC2 key

publicKeyName: zhy2018

allow: true

- name: tidb

desiredCapacity: 2

labels:

dedicated: tidb

taints:

dedicated: tidb:NoSchedule

instanceType: m5.2xlarge

ssh: # use existing EC2 key

publicKeyName: zhy2018

allow: true

- name: pd

desiredCapacity: 3

labels:

dedicated: pd

taints:

dedicated: pd:NoSchedule

instanceType: m5.xlarge

- name: tikv

desiredCapacity: 3

labels:

dedicated: tikv

taints:

dedicated: tikv:NoSchedule

instanceType: m5.4xlarge

ssh: # use existing EC2 key

publicKeyName: zhy2018

allow: true

eksctl create cluster -f cluster-m5.yaml

大概20分钟后，集群创建完成

## 二、部署TiDB on EKS

### 2.1部署 TiDB Operator

<https://docs.pingcap.com/tidb-in-kubernetes/stable/get-started#deploy-tidb-operator>

1.安装TiDB Operator CRD

kubectl apply -f <https://raw.githubu> sercontent.com/pingcap/tidb-operator/v1.1.10/manifests/crd.yaml

2.安装TiDB Operator

1.1Add the PingCAP repository:

helm repo add pingcap https://charts.pingcap.org/

【already exist】

1.2 Create a namespace for TiDB Operator:

kubectl create namespace tidb-admin

1.3 Install TiDB Operator 【ZHY的命令和链接不同】

helm install --namespace tidb-admin tidb-operator pingcap/tidb-operator --version v1.1.6 --set operatorImage=registry.cn-beijing.aliyuncs.com/tidb/tidb-operator:v1.1.6 --set tidbBackupManagerImage=registry.cn-beijing.aliyuncs.com/tidb/tidb-backup-manager:v1.1.6 --set scheduler.kubeSchedulerImageName=registry.cn-hangzhou.aliyuncs.com/google\_containers/kube-scheduler

确认运行

kubectl get pods --namespace tidb-admin -l app.kubernetes.io/instance=tidb-operator

输出：

AME READY STATUS RESTARTS AGE

tidb-controller-manager-7b65b54d6d-d5tw7 1/1 Running 0 24s

tidb-scheduler-7696878976-sjrfp 2/2 Running 0 24s

### 2.2部署TiDB

1.创建namespace

kubectl create namespace tidb-cluster

2.下载配置文件 TidbCluster and TidbMonitor :

curl -O https://raw.githubusercontent.com/pingcap/tidb-operator/master/examples/aws/tidb-cluster.yaml && \

curl -O https://raw.githubusercontent.com/pingcap/tidb-operator/master/examples/aws/tidb-monitor.yaml

3.部署

[参数修改： https://github.com/kubernetes-sigs/aws-ebs-csi-driver]

修改tikv的磁盘容量为1000G

kubectl apply -f tidb-cluster.yaml -n tidb-cluster && \

kubectl apply -f tidb-monitor.yaml -n tidb-cluster

4.查看部署运行状态

kubectl get pods -n tidb-cluster

|  |
| --- |
| NAME READY STATUS RESTARTS AGE |
| basic-discovery-6c6b77f78-qhq4c 1/1 Running 0 8m27s |
| basic-monitor-6497584cf6-q8d8w 3/3 Running 0 8m26s |
| basic-pd-0 1/1 Running 0 8m27s |
| basic-pd-1 1/1 Running 1 8m27s |
| basic-pd-2 1/1 Running 0 8m27s |
| basic-tidb-0 2/2 Running 0 6m48s |
| basic-tidb-1 2/2 Running 0 6m48s |
| basic-tikv-0 1/1 Running 0 7m32s |
| basic-tikv-1 1/1 Running 0 7m32s |
| basic-tikv-2 1/1 Running 0 7m32s |

5.查看集群vpc

eksctl get cluster -n eks11901

6.添加storageclass

kubectl apply -f https://raw.githubusercontent.com/pingcap/tidb-operator/master/manifests/eks/local-volume-provisioner.yaml

在tidb-cluster.yaml 添加 storageClassName: "local-storage"

### 2.3连接到TiDB数据库

1.安装 MySQL 客户端并连接

在EKS集群同一子网，创建一个EC2，安装eksctl，kubectl，awscli，配置aksk，

2.查看集群的 VPC 和 Subnet

#eksctl get cluster -n ekscn002

3.安装mysql客户端

#sudo yum install mysql -y

4.连接到 TiDB 集群：

查看ELB的dns名称作为tidb的服务器主机

mysql -h a533623e8f8214b9f812e82196e6e34c-758b049077df2f7c.elb.cn-northwest-1.amazonaws.com.cn -P 4000 -u root

mysql> show status;

+--------------------+--------------------------------------+

| Variable\_name      | Value                                |

+--------------------+--------------------------------------+

| Ssl\_cipher         |                                      |

| Ssl\_cipher\_list    |                                      |

| Ssl\_verify\_mode    | 0                                    |

| Ssl\_version        |                                      |

| ddl\_schema\_version | 22                                   |

| server\_id          | 8239b631-26c0-4227-af8d-ae1c97f1392e |

+--------------------+--------------------------------------+

### 2.4 Grafana 监控

1.查看service

#kubectl -n tidb-cluster get svc basic-grafana

2.查看监控页面

打开浏览器：

输入： http://<grafana-lb>:3000

## 三、不同配置的性能测试

### 3.1安装sysbench

启动一个同VPC的ubuntu实例

1.安装sysbench on ubuntu

sudo apt-get update

sudo apt-get install -y sysbench

2.安装mysql客户端

sudo apt-get install -y mysql-client-core-5.7

### 3.2准备测试数据

1.登陆Tidb

mysql -h ae3f04f2b91b442dba6e76b27db6bf3b-69f43f400153a570.elb.cn-northwest-1.amazonaws.com.cn -P 4000 -u root

>create database sbtest;

2.初始化数据

sysbench \

--mysql-host=a94bc00ea0b7b4dfe9cc27db0e306a7c-653b2958e6fbedc4.elb.cn-northwest-1.amazonaws.com.cn \

--mysql-port=4000 \

--mysql-user=root \

--mysql-db=sbtest2 \

--time=600 \

--threads=16 \

--report-interval=10 \

--db-driver=mysql \

--rand-type=uniform \

--rand-seed=$RANDOM \

--tables=16 \

--table-size=10000000 \

oltp\_common \

prepare

3.数据预热

sysbench \

--mysql-host=a94bc00ea0b7b4dfe9cc27db0e306a7c-653b2958e6fbedc4.elb.cn-northwest-1.amazonaws.com.cn \

--mysql-port=4000 \

--mysql-user=root \

--mysql-db=sbtest2 \

--time=600 \

--threads=16 \

--report-interval=10 \

--db-driver=mysql \

--rand-type=uniform \

--rand-seed=$RANDOM \

--tables=16 \

--table-size=10000000 \

oltp\_common \

prewarm

4开始测试

sysbench \

--mysql-host=a94bc00ea0b7b4dfe9cc27db0e306a7c-653b2958e6fbedc4.elb.cn-northwest-1.amazonaws.com.cn \

--mysql-port=4000 \

--mysql-user=root \

--mysql-db=sbtest2 \

--time=600 \

--threads=32 \

--report-interval=10 \

--db-driver=mysql \

--rand-type=uniform \

--rand-seed=$RANDOM \

--tables=16 \

--table-size=10000000 \

oltp\_point\_select \

run

3.3测试结果

一个测试终端，数据库32thread，16 table下的结果

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Sysbench测试机配置和参数 | | | Tikv配置 | Storage配置 | | point\_select | update\_index | read\_only | oltp\_read\_write |
| 1\*M5.4x | 32 thread | m5.4xlarge | m5.large\*3 | 100G\*3 | EBS GP2 | 14K, 3ms | 2.3K, 13ms | 587,54ms |  |
| 1\*M5.4x | 32 thread | m5.4xlarge | m5.large\*3 | 1000G\*3 | EBS GP2 | 12k，2.4ms | 1.5k，21ms | 538,59ms | 294,108ms |
| 1\*M5.4x | 64thread | m5.4xlarge | m5.large\*3 | 1000G\*3 | EBS GP2 | 11k,15ms | 2.3k,44ms | 587,167ms | 398,200ms |
| 2\*M5.4x | 32 thread | m5.4xlarge | m5.large\*3 | 1000G\*3 | EBS GP2 | 8.3k,3ms+8.6k,3.6ms |  |  | 200,200ms+200,240ms |
| 1\*c5.4x | 32 thread | c5.4xlarge | i3.2x\*3 | 100G\*3 | nvme | 15k,2.0ms | 1.9k,16ms | 1k/17k,29 | 600/12k,52ms |
| 1\*c5.4x | 32 thread | c5.4xlarge | i3.2x\*3 | 1000G\*3 | nvme | 18.5k, 1.73ms | 3.1k，10.2ms | 1k/17k,29 | 600/12k,52ms |
| 1\*c5.4x | 32 thread | c5.4xlarge | i3.2x\*3 | 1741G\*3 | nvme | 19.2k,1.66ms | 3.1k，10.3ms | 1k/17k,29 | 600/12k,52ms |
| 2\*c5.4x | 32 thread | c5.4xlarge | i3.2x\*3 | 1741G\*3 | nvme | 18.8k+18.9k, 1.7ms | 2.5k+2.5k,12ms | 856+930/13.6k+14.9k,37ms | 480+500,9k+10k,66ms |
| 3\*c5.4x | 32 thread | c5.4xlarge | i3.2x\*3 | 1741G\*3 | nvme | 17k+19K+17k,2ms | 2k+2k+2k,14ms | 730+728+790,11k+11k+12k,43ms | 396+420+400,7.9k+8.2k+8k,80ms |

三个测试终端，不同thread下的结果

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 3\*c5.4x | 32 thread\*3 | c5.4xlarge | i3.2x\*3 | 1741G\*3 | nvme | 16K+16K+15k,2ms | 2k+2k+2k,14ms | 730+728+790,11k+11k+12k,43ms | 396+420+400,7.9k+8.2k+8k,80ms |
| 3\*c5.4x | 64 thtread\*3 | c5.4xlarge | i3.2x\*3 | 1741G\*3 | nvme | 25K+25K+26K，2.52ms | 2.7k+2.9K+2.8K,23ms | 900+849+916,14K+13.5K+14.6K,66ms | 517+500+536,10k+10K+10K,125ms |
| 3\*c5.4x | 128 thread\*3 | c5.4xlarge | i3.2x\*3 | 1741G\*3 | nvme | 31k+29k+30k,4ms | 3K+3.1k+2,9K,42ms | 928+944+892,14.8K+15.1K+14.2K,140ms | 598+635+588,11.9K+12.7K+11.7k,220ms |
| 3\*c5.4x | 64\*3 | [tidb\*2](mailto:tidb*@) | i3.2x\*6 | 1741G\*6 | nvme | 25+28.7+30,2.3ms | 3.4+3.4+3.6k,18ms | 1.1+1.2+1.1k,19+17+20,49ms | 520+600+660,12+13+10,100ms |
| 3\*c5.4x | 128\*3 | [tidb\*2](mailto:tidb*@) | i3.2x\*6 | 1741G\*6 | nvme | 43+48+46,2.8ms | 4.4+4.4+4.6,27ms | 1.2+1.2+1.3,20+20+21,94ms | 937+772+795,16+16+15k,130ms |
|  |  |  |  |  |  |  |  |  |  |
| 3\*c5.4x | 128\*3 | [tidb\*2 m5.2x](mailto:tidb*@) | m5.4x\*3 | 1741G\*3 ? | gp2 | 40+40+40k,3.14ms | 2.9+3+3,41ms | 920+940+910,14+15\_15K,140ms | 600+575+576,11k+11K+12k,220ms |
| 3\*c5.4x | 128\*3 | [tidb\*2 m5.2x](mailto:tidb*@) | m5.4x\*3 | 3000G\*3 | gp2 | 40+41+40k,3.13ms | 4.2k+4.2k+4.2k,30 | 900+900+886,14+14+14,140ms | 620+660+620, 12k+12k+13K,200ms |