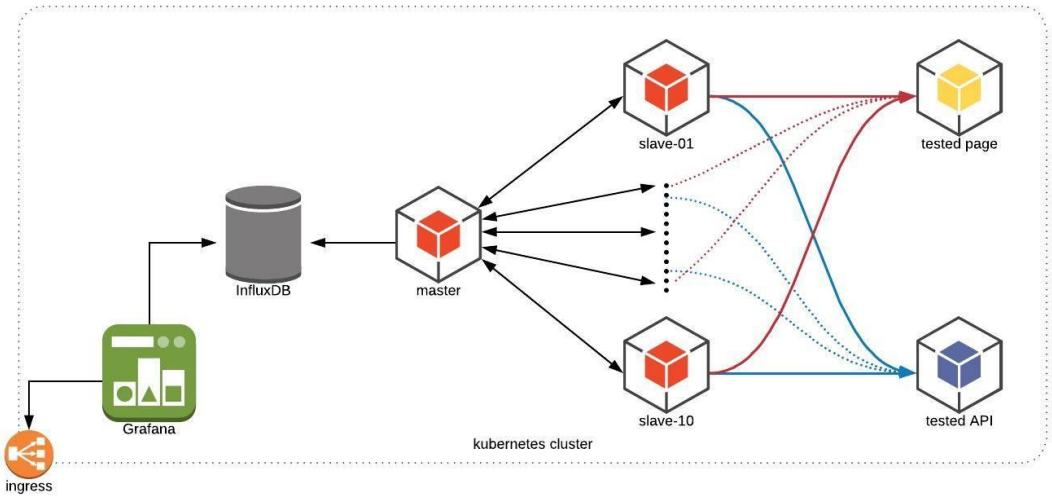
一、 前提条件

Kubernetes > 1.16

二、 部署拓扑



可以从 master 节点启动测试，master 节点把对应的测试脚本发送到对应的 slaves 节点，slave 节点的 pod/nodes 主要作用即发压。

部署文件清单：

- `jmeter_cluster_create.sh` — 此脚本将要求一个唯一的 namespace，然后它将继续创建命名空间和所有组件（jmeter master，slaves，influxdb 和 grafana）。
- 注意：在启动前，请在 `jmeter_slaves_deploy.yaml` 文件中设置要用于 slaves 服务器的副本数，通常副本数应与拥有的 worker nodes 相匹配。
- `jmeter_master_configmap.yaml` — Jmeter master 的应用配置。
- `jmeter_master_deployment.yaml` — Jmeter master 的部署清单。
- `jmeter_slaves_deploy.yaml` — Jmeter slave 的部署清单。
- `jmeter_slave_svc.yaml` — jmeter slave 的服务清单。使用 headless service，这使我们能够直接获取 jmeter slave 的 POD IP 地址，而我们不需要 DNS 或轮询。创建此文件是为了使 slave Pod IP 地址更容易直接发送到 jmeter master。
- `jmeter_influxdb_configmap.yaml` — influxdb 部署的应用配置。如果要在默认的 influxdb 端口之外使用 graphite 存储方法，这会将 influxdb 配置为暴露端口 2003，以便支持 graphite 。因此，可以使用 influxdb 部署来支持jmeter 后置监听器方法（graphite 和 influxdb）。
- `jmeter_influxdb_deploy.yaml` — Influxdb 的部署清单
- `jmeter_influxdb_svc.yaml` — Influxdb 的服务清单。
- `jmeter_grafana_deploy.yaml` — grafana 部署清单。
- `jmeter_grafana_svc.yaml` — grafana 部署的服务清单，默认情况下使用 NodePort，如果公有云中运行它，则可以将其更改为 LoadBalancer（并且可以设置 CNAME 以使用 FQDN 缩短名称）。
- `dashboard.sh` — 该脚本用于自动创建以下内容：
  - （1）influxdb pod 中的一个 influxdb 数据库（Jmeter）
  - （2）grafana 中的数据源（jmeterdb）
- `start_test.sh` —此脚本用于自动运行 Jmeter 测试脚本，而无需手动登录 Jmeter 主 shell，它将询问 Jmeter 测试脚本的位置，然后将其复制到 Jmeter master pod 并启动自动对 Jmeter slave 进行测试。

- `jmeter_stop.sh` - 停止测试
- `GrafanaJMeterTemplate.json` — 预先构建的 Jmeter grafana 仪表板。
- `Dockerfile-base` - 构建 Jmeter 基础镜像
- `Dockerfile-master` - 构建 Jmeter master 镜像
- `Dockerfile-slave` - 构建 Jmeter slave 镜像
- `Dockerimages.sh` - 批量构建 docker 镜像

三、docker 镜像

1、构建 docker 镜像

执行脚本，构建镜像：

```
./dockerimages.sh
```

查看镜像：

```
$ docker images
```

将镜像推送到 Registry：

```
$ sudo docker login --username=xxxx registry.cn-beijing.aliyuncs.com
$ sudo docker tag [ImageId] registry.cn-beijing.aliyuncs.com/7d/jmeter-base:[镜像版本号]
$ sudo docker push registry.cn-beijing.aliyuncs.com/7d/jmeter-base:[镜像版本号]
```

2、部署清单

Dockerfile-base （构建 Jmeter 基础镜像）：

```
FROM alpine:latest
LABEL MAINTAINER 7DGroup

ARG JMETER_VERSION=5.2.1

#定义时区参数
ENV TZ=Asia/Shanghai

RUN apk update && \
    apk upgrade && \
    apk add --update openjdk8-jre wget tar bash && \
    mkdir /jmeter  && cd /jmeter/ && \
    wget https://mirrors.tuna.tsinghua.edu.cn/apache/jmeter/binaries/apache-jmeter-${JMETER_VERSION}.tgz &
    tar -xzf apache-jmeter-$JMETER_VERSION.tgz  && rm apache-jmeter-$JMETER_VERSION.tgz  && \
    cd /jmeter/apache-jmeter-$JMETER_VERSION/ && \
    wget -q -O /tmp/JMeterPlugins-Standard-1.4.0.zip https://jmeter-plugins.org/downloads/file/JMeterPlugi
    wget -q -O /jmeter/apache-jmeter-$JMETER_VERSION/lib/ext/pepper-box-1.0.jar https://github.com/raladev
    cd /jmeter/apache-jmeter-$JMETER_VERSION/ && \
    wget -q -O /tmp/bzm-parallel-0.7.zip https://jmeter-plugins.org/files/packages/bzm-parallel-0.7.zip &&
    ln -snf /usr/share/zoneinfo/$TZ /etc/localtime && echo "$TZ" > /etc/timezone

ENV JMETER_HOME /jmeter/apache-jmeter-$JMETER_VERSION/

ENV PATH $JMETER_HOME/bin:$PATH
#JMeter 主配置文件
ADD jmeter.properties $JMETER_HOME/bin/jmeter.properties
```

Dockerfile-master（构建 Jmeter master 镜像）：

```
FROM registry.cn-beijing.aliyuncs.com/7d/jmeter-base:latest
MAINTAINER 7DGroup

EXPOSE 60000
```

Dockerfile-slave（构建 Jmeter slave 镜像）：

```
Dockerfile-slave:
FROM registry.cn-beijing.aliyuncs.com/7d/jmeter-base:latest
MAINTAINER 7DGroup

EXPOSE 1099 50000

ENTRYPOINT $JMETER_HOME/bin/jmeter-server \
-Dserver.rmi.localport=50000 \
-Dserver_port=1099 \
-Jserver.rmi.ssl.disable=true
```

Dockerimages.sh（批量构建 docker 镜像）：

```
#!/bin/bash -e
docker build --tag="registry.cn-beijing.aliyuncs.com/7d/jmeter-base:latest" -f Dockerfile-base .
docker build --tag="registry.cn-beijing.aliyuncs.com/7d/jmeter-master:latest" -f Dockerfile-master .
docker build --tag="registry.cn-beijing.aliyuncs.com/7d/jmeter-slave:latest" -f Dockerfile-slave .
```

四、Kubernetes 部署

1、部署组件

执行 `jmeter_cluster_create.sh` ,并输入一个唯一的 namespace

```
./jmeter_cluster_create.sh
```

等待一会，查看pods安装情况：

```
$ kubectl get pods -n 7dgroup
NAME                                READY   STATUS    RESTARTS   AGE
influxdb-jmeter-584cf69759-j5m85    1/1     Running   2          5m
jmeter-grafana-6d5b75b7f6-57dxj     1/1     Running   1          5m
jmeter-master-84bfd5d96d-kthzm      1/1     Running   0          5m
jmeter-slaves-b5b75757-dxkxz        1/1     Running   0          5m
jmeter-slaves-b5b75757-n58jw        1/1     Running   0          5m
```

2、部署清单

2.1、主执行脚本

jmeter_cluster_create.sh（创建命名空间和所有组件（jmeter master，slaves，influxdb 和 grafana））：

```
#!/usr/bin/env bash
#Create multiple Jmeter namespaces on an existing kuberntes cluster
#Started On January 23, 2018
working_dir=`pwd`
echo "checking if kubectl is present"
if ! hash kubectl 2>/dev/null
then
echo "'kubectl' was not found in PATH"
echo "Kindly ensure that you can acces an existing kubernetes cluster via kubectl"
exit
fi
kubectl version --short
echo "Current list of namespaces on the kubernetes cluster:"
echo
kubectl get namespaces | grep -v NAME | awk '{print $1}'
echo
echo "Enter the name of the new tenant unique name, this will be used to create the namespace"
read tenant
echo
#Check If namespace exists
kubectl get namespace $tenant > /dev/null 2>&1
if [ $? -eq 0 ]
then
echo "Namespace $tenant already exists, please select a unique name"
echo "Current list of namespaces on the kubernetes cluster"
sleep 2
kubectl get namespaces | grep -v NAME | awk '{print $1}'
exit 1
fi
echo
echo "Creating Namespace: $tenant"
kubectl create namespace $tenant
echo "Namspace $tenant has been created"
echo
echo "Creating Jmeter slave nodes"
nodes=`kubectl get no | egrep -v "master|NAME" | wc -l`
```

```
echo
echo "Number of worker nodes on this cluster is " $nodes
echo
#echo "Creating $nodes Jmeter slave replicas and service"
echo
kubectl create -n $tenant -f $working_dir/jmeter_slaves_deploy.yaml
kubectl create -n $tenant -f $working_dir/jmeter_slaves_svc.yaml
echo "Creating Jmeter Master"
kubectl create -n $tenant -f $working_dir/jmeter_master_configmap.yaml
kubectl create -n $tenant -f $working_dir/jmeter_master_deploy.yaml

echo "Creating Influxdb and the service"
kubectl create -n $tenant -f $working_dir/jmeter_influxdb_configmap.yaml
kubectl create -n $tenant -f $working_dir/jmeter_influxdb_deploy.yaml
kubectl create -n $tenant -f $working_dir/jmeter_influxdb_svc.yaml
echo "Creating Grafana Deployment"
kubectl create -n $tenant -f $working_dir/jmeter_grafana_deploy.yaml
kubectl create -n $tenant -f $working_dir/jmeter_grafana_svc.yaml
echo "Printout Of the $tenant Objects"
echo
kubectl get -n $tenant all
echo namespace = $tenant > $working_dir/tenant_export
```

2.2、jmeter_slaves

jmeter_slaves_deploy.yaml（Jmeter slave 的部署清单）：

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: jmeter-slaves
  labels:
    jmeter_mode: slave
spec:
  replicas: 2
  selector:
    matchLabels:
      jmeter_mode: slave
  template:
    metadata:
      labels:
        jmeter_mode: slave
    spec:
      containers:
      - name: jmslave
        image: registry.cn-beijing.aliyuncs.com/7d/jmeter-slave:latest
        imagePullPolicy: IfNotPresent
        ports:
        - containerPort: 1099
        - containerPort: 50000
        resources:
          limits:
            cpu: 4000m
            memory: 4Gi
          requests:
            cpu: 500m
            memory: 512Mi
```

jmeter_slaves_svc.yaml（ Jmeter slave 的服务清单）：

```yaml
apiVersion: v1
kind: Service
metadata:
  name: jmeter-slaves-svc
  labels:
    jmeter_mode: slave
spec:
  clusterIP: None
  ports:
    - port: 1099
      name: first
      targetPort: 1099
    - port: 50000
      name: second
      targetPort: 50000
```

2.3、jmeter_master

jmeter_master_configmap.yaml（jmeter_master 应用配置）：

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: jmeter-load-test
  labels:
    app: influxdb-jmeter
data:
  load_test: |
    #!/bin/bash
    #Script created to invoke jmeter test script with the slave POD IP addresses
    #Script should be run like: ./load_test "path to the test script in jmx format"
    /jmeter/apache-jmeter-*/bin/jmeter -n -t $1 `getent ahostsv4 jmeter-slaves-svc | cut -d' ' -f1 | sort
```

jmeter_master_deploy.yaml（jmeter_master 部署清单）：

```yaml
apiVersion: apps/v1 # for versions before 1.9.0 use apps/v1beta2
kind: Deployment
metadata:
  name: jmeter-master
  labels:
    jmeter_mode: master
spec:
  replicas: 1
  selector:
    matchLabels:
      jmeter_mode: master
  template:
    metadata:
      labels:
        jmeter_mode: master
    spec:
      containers:
      - name: jmmaster
        image: registry.cn-beijing.aliyuncs.com/7d/jmeter-master:latest
        imagePullPolicy: IfNotPresent
        command: [ "/bin/bash", "-c", "--" ]
        args: [ "while true; do sleep 30; done;" ]
        volumeMounts:
          - name: loadtest
            mountPath: /load_test
            subPath: "load_test"
        ports:
        - containerPort: 60000
        resources:
          limits:
            cpu: 4000m
            memory: 4Gi
          requests:
            cpu: 500m
            memory: 512Mi
      volumes:
      - name: loadtest
        configMap:
          name: jmeter-load-test
```

2.4、influxdb

jmeter_influxdb_configmap.yaml（influxdb 的应用配置）：

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: influxdb-config
  labels:
    app: influxdb-jmeter
data:
  influxdb.conf: |
    [meta]
      dir = "/var/lib/influxdb/meta"

    [data]
      dir = "/var/lib/influxdb/data"
      engine = "tsm1"
      wal-dir = "/var/lib/influxdb/wal"

    # Configure the graphite api
    [[graphite]]
    enabled = true
    bind-address = ":2003" # If not set, is actually set to bind-address.
    database = "jmeter"  # store graphite data in this database
```

jmeter_influxdb_deploy.yaml（influxdb 部署清单）：

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: influxdb-jmeter
  labels:
    app: influxdb-jmeter
spec:
  replicas: 1
  selector:
    matchLabels:
      app: influxdb-jmeter
  template:
    metadata:
      labels:
        app: influxdb-jmeter
    spec:
      containers:
        - image: influxdb
          imagePullPolicy: IfNotPresent
          name: influxdb
          volumeMounts:
          - name: config-volume
            mountPath: /etc/influxdb
          ports:
            - containerPort: 8083
              name: influx
            - containerPort: 8086
              name: api
            - containerPort: 2003
              name: graphite
      volumes:
      - name: config-volume
        configMap:
         name: influxdb-config
```

jmeter_influxdb_svc.yaml（influxdb 部署服务清单）：

```yaml
apiVersion: v1
kind: Service
metadata:
  name: jmeter-influxdb
  labels:
    app: influxdb-jmeter
spec:
  ports:
    - port: 8083
      name: http
      targetPort: 8083
    - port: 8086
      name: api
      targetPort: 8086
    - port: 2003
      name: graphite
      targetPort: 2003
  selector:
    app: influxdb-jmeter
```

2.5、grafana

jmeter_grafana_deploy.yaml（grafana 部署清单）：

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: jmeter-grafana
  labels:
    app: jmeter-grafana
spec:
  replicas: 1
  selector:
    matchLabels:
      app: jmeter-grafana
  template:
    metadata:
      labels:
        app: jmeter-grafana
    spec:
      containers:
      - name: grafana
        image: grafana/grafana:5.2.0
        imagePullPolicy: IfNotPresent
        ports:
        - containerPort: 3000
          protocol: TCP
        env:
        - name: GF_AUTH_BASIC_ENABLED
          value: "true"
        - name: GF_USERS_ALLOW_ORG_CREATE
          value: "true"
        - name: GF_AUTH_ANONYMOUS_ENABLED
          value: "true"
        - name: GF_AUTH_ANONYMOUS_ORG_ROLE
          value: Admin
        - name: GF_SERVER_ROOT_URL
          # If you're only using the API Server proxy, set this value instead:
          # value: /api/v1/namespaces/kube-system/services/monitoring-grafana/proxy
          value: /
```

jmeter_grafana_svc.yaml（grafana 部署服务清单）：

```yaml
apiVersion: v1
kind: Service
metadata:
  name: jmeter-grafana
  labels:
    app: jmeter-grafana
spec:
  ports:
    - port: 3000
      targetPort: 3000
  selector:
    app: jmeter-grafana
  type: NodePort
---
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  annotations:
    nginx.ingress.kubernetes.io/service-weight: 'jmeter-grafana: 100'
  name: jmeter-grafana-ingress
spec:
  rules:
  # 配置七层域名
  - host: grafana-jmeter.7d.com
    http:
      paths:
      # 配置Context Path
      - path: /
        backend:
          serviceName: jmeter-grafana
          servicePort: 3000
```

五、初始化 dashboard

1、启动 dashboard 脚本

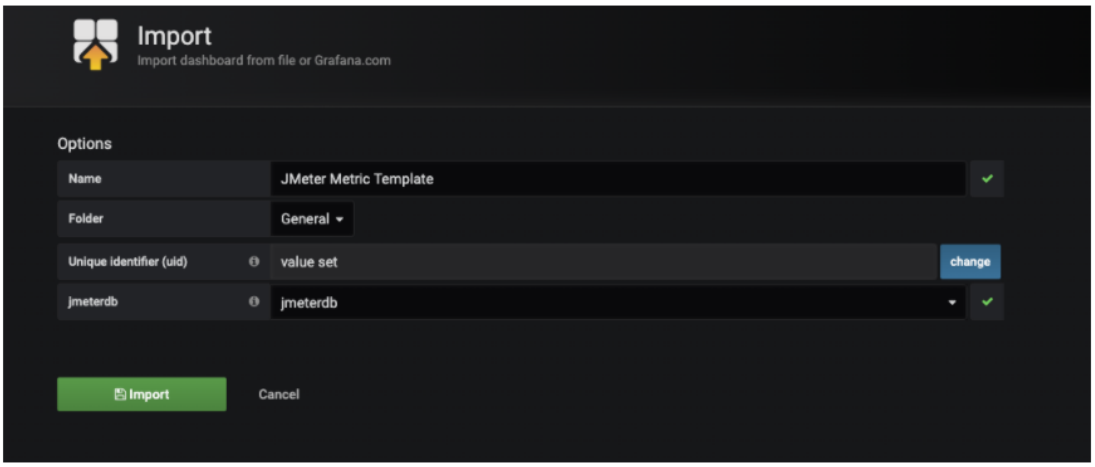
```
$ ./dashboard.sh
```

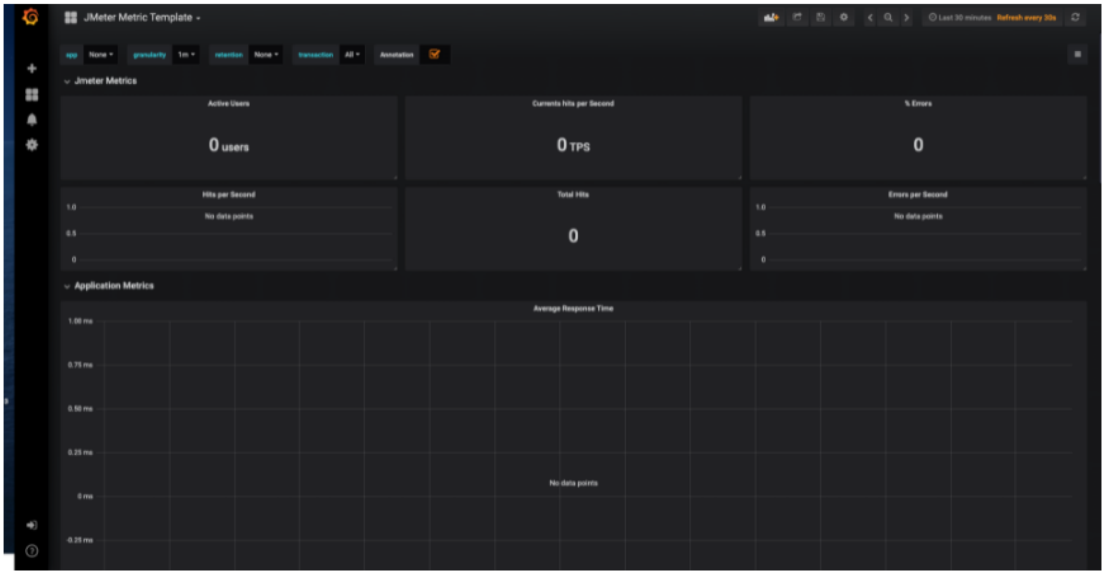
检查 service 部署情况：

```
$ kubectl get svc -n 7dgroup
NAME                TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)                    AGE
jmeter-grafana      NodePort    10.96.6.201    <none>        3000:31801/TCP             10m
jmeter-influxdb     ClusterIP   10.96.111.60   <none>        8083/TCP,8086/TCP,2003/TCP 10m
jmeter-slaves-svc   ClusterIP   None           <none>        1099/TCP,50000/TCP         10m
```

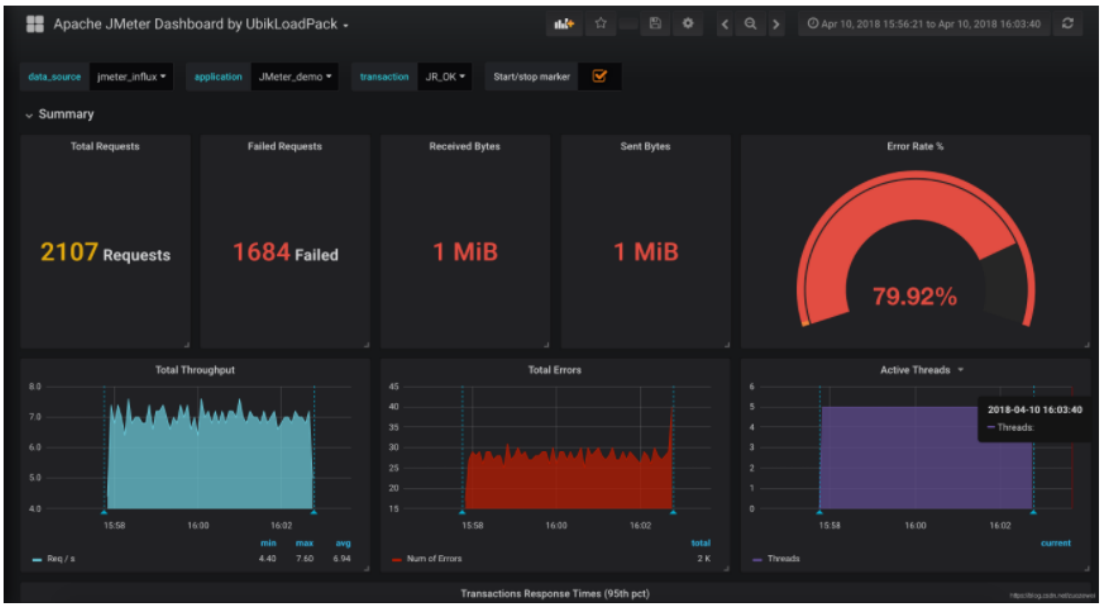
我们可以通过 `http://任意 node_ip:31801/` 访问 grafana

最后，我们在 grafana 导入 dashborad 模版:





如果你不喜欢这个模版，也可以导入热门模版：**5496**


Apache JMeter Dashboard by UbikLoadPack
Apr 10, 2018 15:56:21 to Apr 10, 2018 16:03:40
data_source
jmeter_influx
application
JMeter_demo
transaction
JR_OK
Start/stop marker
Summary
Total Requests
2107 Requests
Failed Requests
1684 Failed
Received Bytes
1 MiB
Sent Bytes
1 MiB
Error Rate %
79.92%
Total Throughput
Req / s
min max avg
4.40 7.60 6.94
Total Errors
Num of Errors
total
2 K
Active Threads
2018-04-10 16:03:40
Threads:
Threads
current
Transactions Response Times (95th pct)


Apache JMeter Dashboard by UbikLoadPack
Apr 10, 2018 15:56:21 to Apr 10, 2018 16:03:40
Req / s
4.40 7.60 6.94
Num of Errors
2 K
Threads
Transactions Response Times (95th pct)
max avg
JR_KO_404 553.80 ms 543.89 ms
JR_KO_500 453.95 ms 443.86 ms
JR_KO_Assertion 652.00 ms 638.48 ms
JR_KO_Assertion2 649.95 ms 644.39 ms
JR_OK 350.80 ms 343.61 ms
all 648.65 ms 625.49 ms
Network Traffic
max avg
JR_KO_404 4.73 KiB 3.28 KiB
JR_KO_500 2.36 KiB 1.65 KiB
JR_KO_Assertion 8.51 KiB 6.54 KiB
JR_KO_Assertion2 18.91 KiB 13.08 KiB
JR_OK 480.00 B 334.40 B
all 30.06 KiB 24.87 KiB
JR_KO_404 4.28 KiB 2.97 KiB
JR_KO_500 4.71 KiB 3.28 KiB
JR_KO_Assertion 405.00 B 311.25 B
JR_KO_Assertion2 450.00 B 311.25 B

2、部署清单

`dashboard.sh` 该脚本用于自动创建以下内容：

- （1）influxdb pod 中的一个 influxdb 数据库（Jmeter）
- （2）grafana 中的数据源（jmeterdb）

```
#!/usr/bin/env bash
working_dir=`pwd`
#Get namesapce variable
tenant=`awk '{print $NF}' $working_dir/tenant_export`
## Create jmeter database automatically in Influxdb
echo "Creating Influxdb jmeter Database"
##Wait until Influxdb Deployment is up and running
##influxdb_status=`kubectl get po -n $tenant | grep influxdb-jmeter | awk '{print $2}' | grep Running
influxdb_pod=`kubectl get po -n $tenant | grep influxdb-jmeter | awk '{print $1}'`
kubectl exec -ti -n $tenant $influxdb_pod -- influx -execute 'CREATE DATABASE jmeter'
## Create the influxdb datasource in Grafana
echo "Creating the Influxdb data source"
grafana_pod=`kubectl get po -n $tenant | grep jmeter-grafana | awk '{print $1}'`
## Make load test script in Jmeter master pod executable
#Get Master pod details
master_pod=`kubectl get po -n $tenant | grep jmeter-master | awk '{print $1}'`
kubectl exec -ti -n $tenant $master_pod -- cp -r /load_test /![]()jmeter/load_test
kubectl exec -ti -n $tenant $master_pod -- chmod 755 /jmeter/load_test
##kubectl cp $working_dir/influxdb-jmeter-datasource.json -n $tenant $grafana_pod:/influxdb-jmeter-datasou
kubectl exec -ti -n $tenant $grafana_pod -- curl 'http://admin:admin@127.0.0.1:3000/api/datasources' -X PO
```

六、启动测试

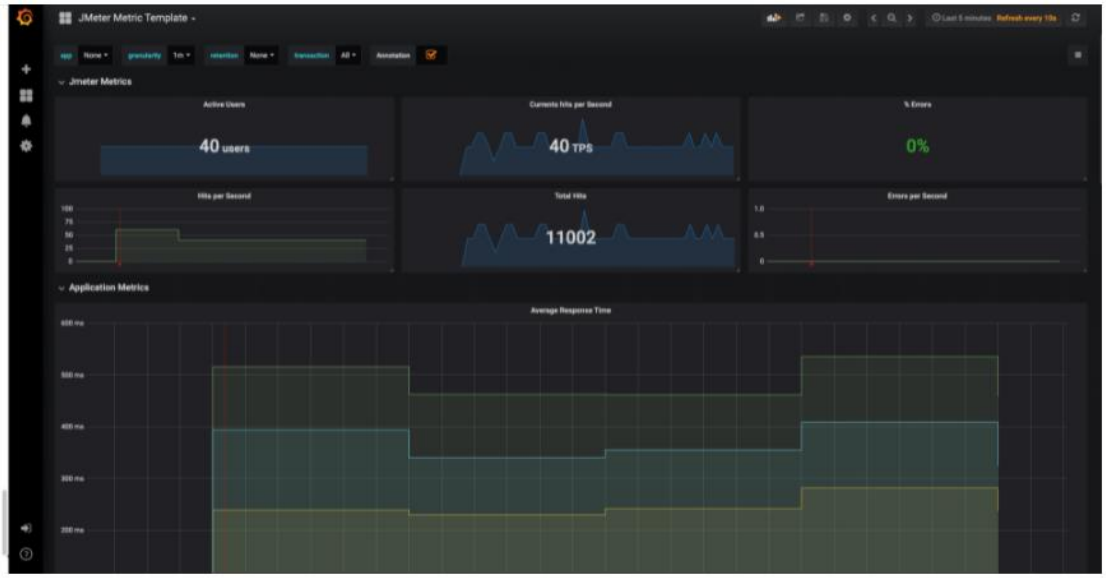
1、执行脚本

```
$ ./start_test.sh
```

需要一个测试脚本，本例为：`web-test.jmx`

```
$  ./start_test.sh
Enter path to the jmx file web-test.jmx
''SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/jmeter/apache-jmeter-5.0/lib/log4j-slf4j-impl-2.11.0.jar!/org/slf4j/imp
SLF4J: Found binding in [jar:file:/jmeter/apache-jmeter-5.0/lib/ext/pepper-box-1.0.jar!/org/slf4j/impl/Sta
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
Jul 25, 2020 11:30:58 AM java.util.prefs.FileSystemPreferences$1 run
INFO: Created user preferences directory.
Creating summariser <summary>
Created the tree successfully using web-test.jmx
Configuring remote engine: 10.100.113.31
Configuring remote engine: 10.100.167.173
Starting remote engines
Starting the test @ Sat Jul 25 11:30:59 UTC 2020 (1595676659540)
Remote engines have been started
Waiting for possible Shutdown/StopTestNow/Heapdump message on port 4445
summary +    803 in 00:00:29 =   27.5/s Avg:   350 Min:   172 Max:  1477 Err:     0 (0.00%) Active: 40 Sta
summary +   1300 in 00:00:29 =   45.3/s Avg:   367 Min:   172 Max:  2729 Err:     0 (0.00%) Active: 40 Sta
summary =   2103 in 00:00:58 =   36.4/s Avg:   361 Min:   172 Max:  2729 Err:     0 (0.00%)
summary +   1400 in 00:00:31 =   45.4/s Avg:   342 Min:   160 Max:  2145 Err:     0 (0.00%) Active: 40 Sta
summary =   3503 in 00:01:29 =   39.5/s Avg:   353 Min:   160 Max:  2729 Err:     0 (0.00%)
summary +   1400 in 00:00:31 =   45.2/s Avg:   352 Min:   169 Max:  2398 Err:     0 (0.00%) Active: 40 Sta
summary =   4903 in 00:02:00 =   41.0/s Avg:   353 Min:   160 Max:  2729 Err:     0 (0.00%)
summary +   1400 in 00:00:30 =   46.8/s Avg:   344 Min:   151 Max:  1475 Err:     0 (0.00%) Active: 40 Sta
summary =   6303 in 00:02:30 =   42.1/s Avg:   351 Min:   151 Max:  2729 Err:     0 (0.00%)
summary +   1200 in 00:00:28 =   43.5/s Avg:   354 Min:   163 Max:  2018 Err:     0 (0.00%) Active: 40 Sta
summary =   7503 in 00:02:57 =   42.3/s Avg:   351 Min:   151 Max:  2729 Err:     0 (0.00%)
summary +   1300 in 00:00:30 =   43.7/s Avg:   456 Min:   173 Max:  2401 Err:     0 (0.00%) Active: 40 Sta
summary =   8803 in 00:03:27 =   42.5/s Avg:   367 Min:   151 Max:  2729 Err:     0 (0.00%)
summary +   1400 in 00:00:31 =   44.9/s Avg:   349 Min:   158 Max:  2128 Err:     0 (0.00%) Active: 40 Sta
summary =  10203 in 00:03:58 =   42.8/s Avg:   364 Min:   151 Max:  2729 Err:     0 (0.00%)
summary +   1400 in 00:00:32 =   44.3/s Avg:   351 Min:   166 Max:  1494 Err:     0 (0.00%) Active: 40 Sta
```

查看测试数据：



2、部署清单

`start_test.sh`（此脚本用于自动运行 Jmeter 测试脚本，而无需手动登录 Jmeter 主 shell，它将询问 Jmeter 测试脚本的位置，然后将其复制到 Jmeter master pod 并启动自动对 Jmeter slave 进行测试）：

```
#!/usr/bin/env bash
#Script created to launch Jmeter tests directly from the current terminal without accessing the jmeter mas
#It requires that you supply the path to the jmx file
#After execution, test script jmx file may be deleted from the pod itself but not locally.

#直接从当前终端启动 Jmeter 测试而创建的脚本，无需访问 Jmeter master pod。
#要求提供 jmx 文件的路径
#执行后，测试脚本 jmx 文件可能会从 pod 本身删除，但不会在本地删除。

working_dir="`pwd`"

# 获取 namesapce 变量
tenant=`awk '{print $NF}' "$working_dir/tenant_export"`

jmx="$1"
[ -n "$jmx" ] || read -p 'Enter path to the jmx file ' jmx

if [ ! -f "$jmx" ];
then
    echo "Test script file was not found in PATH"
    echo "Kindly check and input the correct file path"
    exit
fi

test_name="$(basename "$jmx")"

# 获取 master pod 详细信息
master_pod=`kubectl get po -n $tenant | grep jmeter-master | awk '{print $1}'`
kubectl cp "$jmx" -n $tenant "$master_pod:/$test_name"

## 启动 Jmeter 压测
kubectl exec -ti -n $tenant $master_pod -- /bin/bash /load_test "$test_name"
kubectl exec -ti -n $tenant $master_pod -- /bin/bash /load_test "$test_name"
```

`jmeter_stop.sh`（停止测试）：

```
#!/usr/bin/env bash
#Script writtent to stop a running jmeter master test
#Kindly ensure you have the necessary kubeconfig

#编写脚本来停止运行的 jmeter master 测试
#请确保你有必要的 kubeconfig
working_dir=`pwd`

#获取 namesapce 变量
tenant=`awk '{print $NF}' $working_dir/tenant_export`
master_pod=`kubectl get po -n $tenant | grep jmeter-master | awk '{print $1}'`
kubectl -n $tenant exec -it $master_pod -- bash -c "./jmeter/apache-jmeter-5.0/bin/stoptest.sh"
```

七、小结

传统 Jmeter 存在的问题：

- 并发数超过单节点承载能力时，多节点环境配置、维护复杂；
- 默认配置下无法并行运行多个测试，需要更改配置启动额外进程；
- 难以支持云环境下测试资源的弹性伸缩需求。

Kubernetes-Jmeter 带来的改变：

- 压测执行节点一键安装；
- 多个项目、多个测试可并行使用同一个测试资源池（最大并发数允许情况下， Kubernetes 也提供了 RBAC、namespace 等管理能力，支持多用户共享一个集群，并实现资源限制），提高资源利用率；
- 对接 Kubernetes HPA 根据并发数自动启动、释放压测执行节点。

源码地址：

- https://github.com/zuozewei/blog-example/tree/master/Kubernetes/k8s-jmeter-cluster

参考资料：

- [1]：https://github.com/kubernauts/jmeter-kubernetes