云原生报告 – 第 6 组

组员和工作分配:

181250037 顾正昕 – Prometheus 接口、Grafana 监控

181250054 霍卓健 – dockerfile 编写

181250090 刘育麟 – rest 接口编写、接口限流、k8s 部署、Jenkins 流水线

GitHub 仓库

https://github.com/1Lucifer1/Cloud_Native_Project、

rest 接口

```java
@RestController
public class GreetingController {

    @Autowired
    private GreetingService greetingService;

    @GetMapping("/greeting")
    @RequestLimit(count=100)
    public Object greeting() {
        return greetingService.greeting();
    }
}
```

```java
@Service
public class GreetingService {
    public Object greeting(){
        return new Greeting( msg: "Hello");
    }
}
```

本地运行

{"msg":"Hello"}

限流功能

- 使用自定义注解

```java
public @interface RequestLimit {
    /**
     * 允许访问的次数，默认值MAX_VALUE
     */
    int count() default Integer.MAX_VALUE;

    /**
     * 时间段，单位为毫秒，默认值一秒钟
     */
    long time() default 1000;
}
```

- 算法（使用 hash 表储存每个接口对的被访问次数，并添加 synchronized 描述避免冲突）

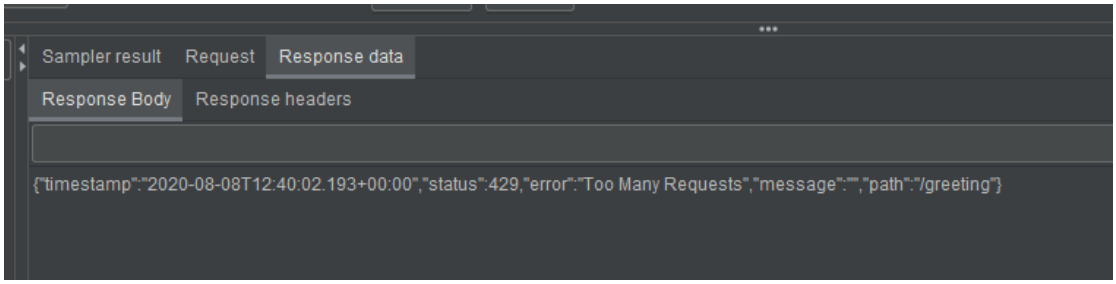
```java
String key = "req_limit_".concat(url); //hash的key
if (!redisTemplate.containsKey(key)) { //接口未访问过
    redisTemplate.put(key, 1);
    System.out.println("1:" + key);
} else {
    redisTemplate.put(key, redisTemplate.get(key) + 1);
    int count = redisTemplate.get(key);
    System.out.println(count + ":" + key);
    if (count > rateLimiter.count()) {
        //logger.info("超过了限定的次数[" + limit.count() + "]");\
//          return new RequestLimitException("429: Too many requests");
        throw new RequestLimitException();
    }else {
        Timer timer = new Timer();
        TimerTask task = new TimerTask() {    //创建一个新的计时器任务。
            @Override
            public synchronized void run() { redisTemplate.remove(key); }
        };
        timer.schedule(task, rateLimiter.time());
        //安排在指定延迟后执行指定的任务。task ： 所要安排的任务。time ： 执行任务前的延迟时间，单位是毫秒。
    }
}
return joinPoint.proceed();
```

- 当 value 大于 100 时返回前端的异常

```java
@ResponseStatus(value = HttpStatus.TOO_MANY_REQUESTS)
public class RequestLimitException extends Exception{

}
```

- 运行结果



Dockerfile

- 按照 demo 进行配置，将 maven build 的结果存进 cloud-native-project-0.0.1-SNAPSHOT.jar

```
FROM openjdk:8-jre-alpine

RUN ln -sf /usr/share/zoneinfo/Asia/Shanghai /etc/localtime
RUN echo 'Asia/Shanghai' >/etc/timezone

ENV JAVA_OPTS ''

WORKDIR /app
ADD target/cloud-native-project-0.0.1-SNAPSHOT.jar .


ENTRYPOINT ["sh", "-c", "set -e && java -XX:+PrintFlagsFinal \
                                        -XX:+HeapDumpOnOutOfMemoryError \
                                        -XX:HeapDumpPath=/heapdump/heapdump.hprof \
                                        -XX:+UnlockExperimentalVMOptions \
                                        -XX:+UseCGroupMemoryLimitForHeap \
                                        $JAVA_OPTS -jar cloud-native-project-0.0.1-SNAPSHOT.jar"]
```

K8s 编排文件

- 原始端口：8080
- 分配端口：40000
- 项目：cloud-native-project
- 类型：NodePort
- 镜像：harbor.edu.cn/cn202006/cloud-native-project:{VERSION}
- Namespace：cn202006

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: cloud-native-project
  name: cloud-native-project
  namespace: cn202006
spec:
  replicas: 1
  selector:
    matchLabels:
      app: cloud-native-project
  template:
    metadata:
      annotations:
        prometheus.io/path: /actuator/prometheus
        prometheus.io/port: "8080"
        prometheus.io/scheme: http
        prometheus.io/scrape: "true"
      labels:
        app: cloud-native-project
    spec:
      containers:
      - image: harbor.edu.cn/cn202006/cloud-native-project:{VERSION}
        name: cloud-native-project
      imagePullSecrets:
        - name: cn202006
```

```yaml
---
apiVersion: v1
kind: Service
metadata:
  name: cloud-native-project
  namespace: cn202006
  labels:
    app: cloud-native-project
spec:
  type: NodePort
  selector:
    app: cloud-native-project
  ports:
  - name: tcp
    nodePort: 40000
    protocol: TCP
    port: 8080
    targetPort: 8080
```

持续集成流水线

- 代码

```
pipeline {
    agent none
    stages {
        stage('Clone to master') {
            agent {
                label 'master'

            }
            steps {
                echo "1.Git Clone Stage"
                git url: "https://github.com/1Lucifer1/Cloud_Native_Project.git"

            }

        }
        stage('Maven Build') {
            agent {
                docker {
                    image 'maven:latest'
                    args '-v /root/.m2:/root/.m2'

                }

            }
            steps {
                echo "2.Maven Build Stage"
                sh 'mvn -B clean package -Dmaven.test.skip=true'

            }

        }
        stage('Image Build') {
            agent {
                label 'master'

            }
            steps {
                echo "3.Image Build Stage"
                sh 'docker build -f Dockerfile --build-arg jar_name=target/cloud-native-
project-0.0.1-SNAPSHOT.jar -t cloud-native-project:${BUILD_ID} . '
                sh 'docker tag cloud-native-project:${BUILD_ID}
harbor.edu.cn/cn202006/cloud-native-project:${BUILD_ID}'
```

```
                }

            }
            stage('Push') {
                agent {
                    label 'master'

                }
                steps {
                    echo "4.Push Docker Image Stage"
                    sh "docker login --username=cn202006 harbor.edu.cn -p cn202006"
                    sh "docker push harbor.edu.cn/cn202006/cloud-native-
project:${BUILD_ID}"

                }

            }

        }

}
```

- 运行结果

持续部署流水线

- 代码

```
node('slave') {
      container('jnlp-kubectl') {
            stage('connect'){
                sh 'curl "http://p.nju.edu.cn/portal_io/login" --data
"username=181250090&password=willy229liu"'
            }
            stage('Git Clone') {
                git url: "https://github.com/1Lucifer1/Cloud_Native_Project.git"

            }
            stage('YAML') {
```

```
                echo "5. Change YAML File Stage"
                sh 'sed -i "s#{VERSION}#${BUILD_ID}#g" ./jenkins/scripts/cloud-native-
project.yaml'

            }
            stage('Deploy') {
                echo "6. Deploy To K8s Stage"
                sh 'kubectl apply -f ./jenkins/scripts/cloud-native-project.yaml -n cn202006'
                sh 'kubectl apply -f ./jenkins/scripts/cloud-native-project-
serviceMonitor.yaml'

            }

        }

}
```

- 运行结果

持续测试流水线

- 代码

```
            stage('RTF Test'){
                echo "RTF Test Stage"
                sh 'kubectl apply -f ./jenkins/scripts/rtf.yaml -n cn202006'

            }
```

- 脚本

```
        *** Settings ***
        Library                 requests

        *** Test Cases ***
        case 1
            ${res}      requests.get        http://172.29.4.47:40000/greeting
            should contain      ${res.text}     Hello
```

- 目的

  将 python 部署到 k8s 上并且运行 rtf 测试脚本

```
      app: rtf
  spec:
    containers:
      - name: rtf
        image: python
        command: ["/bin/sh"]
        args: ["-c", 'curl "http://p.nju.edu.cn/portal_io/login" --data
    imagePullSecrets:
      - name: cn202006
```

- 运行结果


```
$ kubectl get pods -n cn202006
NAME                                  READY   STATUS      RESTARTS   AGE
cloud-native-project-6986d57c7d-dsssn   1/1     Running     0          44s
rtf-56d8b6cc5c-kmfh6                  0/1     Completed   1          35s
```

```
Installing collected packages: robotframework
Successfully installed robotframework-3.2.1
==============================================================================
Rtf
==============================================================================
case 1                                                                | PASS |
------------------------------------------------------------------------------
Rtf                                                                   | PASS |
1 critical test, 1 passed, 0 failed
1 test total, 1 passed, 0 failed
==============================================================================
Output:  /output.xml
Log:     /log.html
Report:  /report.html
```

Prometheus targets 和应用 pod

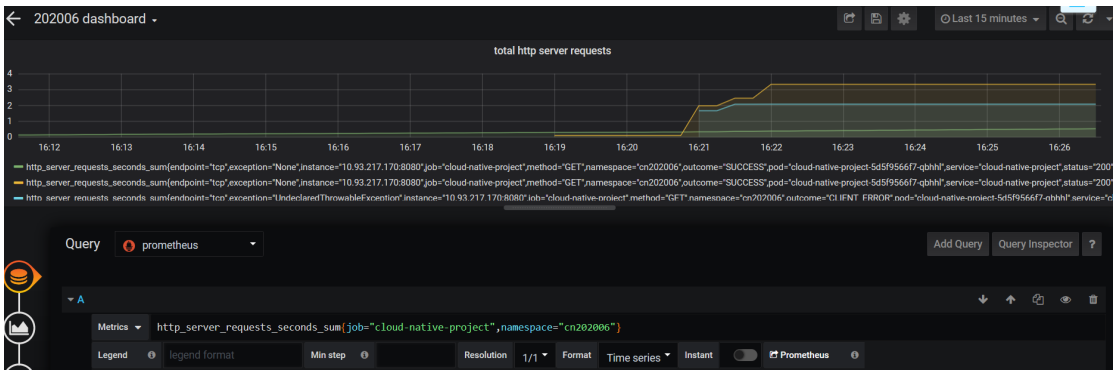
monitoring/cloud-native-project/0 (1/1 up) show less

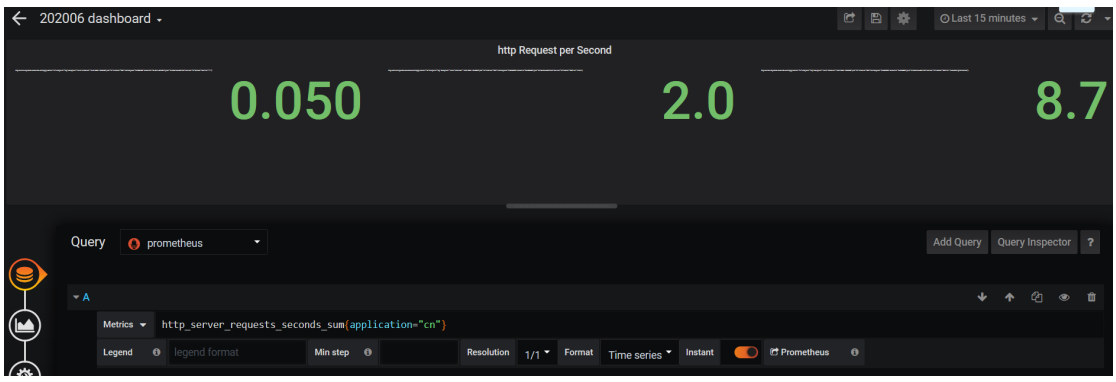
Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://10.87.202.224:8080/actuator/prometheus	UP	endpoint="tcp" instance="10.87.202.224:8080" job="cloud-native-project" namespace="cn202006" pod="cloud-native-project-dcd7675f8-xpnkw" service="cloud-native-project"	3.81s ago	6.481ms	

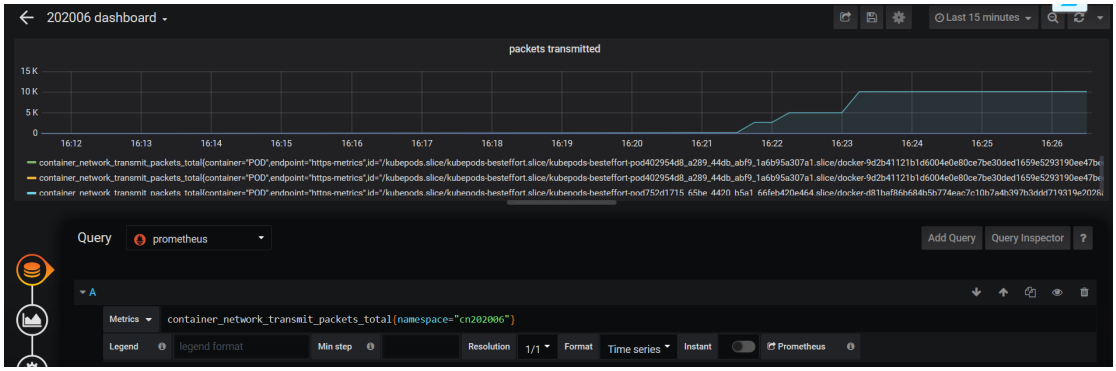
```
$ kubectl get pods -n cn202006
NAME                                  READY   STATUS    RESTARTS   AGE
cloud-native-project-5d5f9566f7-qbhhl   1/1     Running   0          6m41s
```

```
$ kubectl get svc -n cn202006
NAME                  TYPE       CLUSTER-IP      EXTERNAL-IP   PORT(S)          AGE
cloud-native-project   NodePort   10.64.89.99     <none>        8080:40000/TCP   108m
```

Grafana 定制 Dashboard，指标包含 http 请求总数、每秒 http 请求数量、接受包数、处理包数







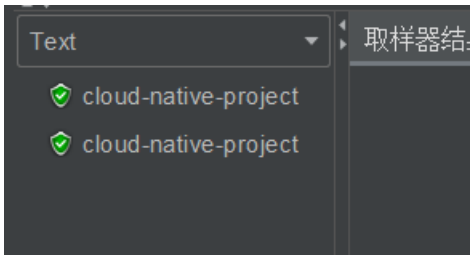
JMeter 配置

线程数（非固定值）

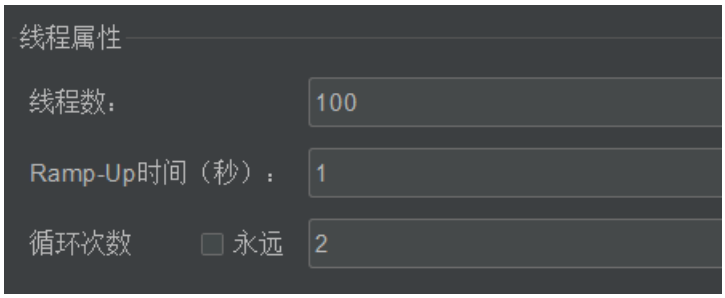
服务器

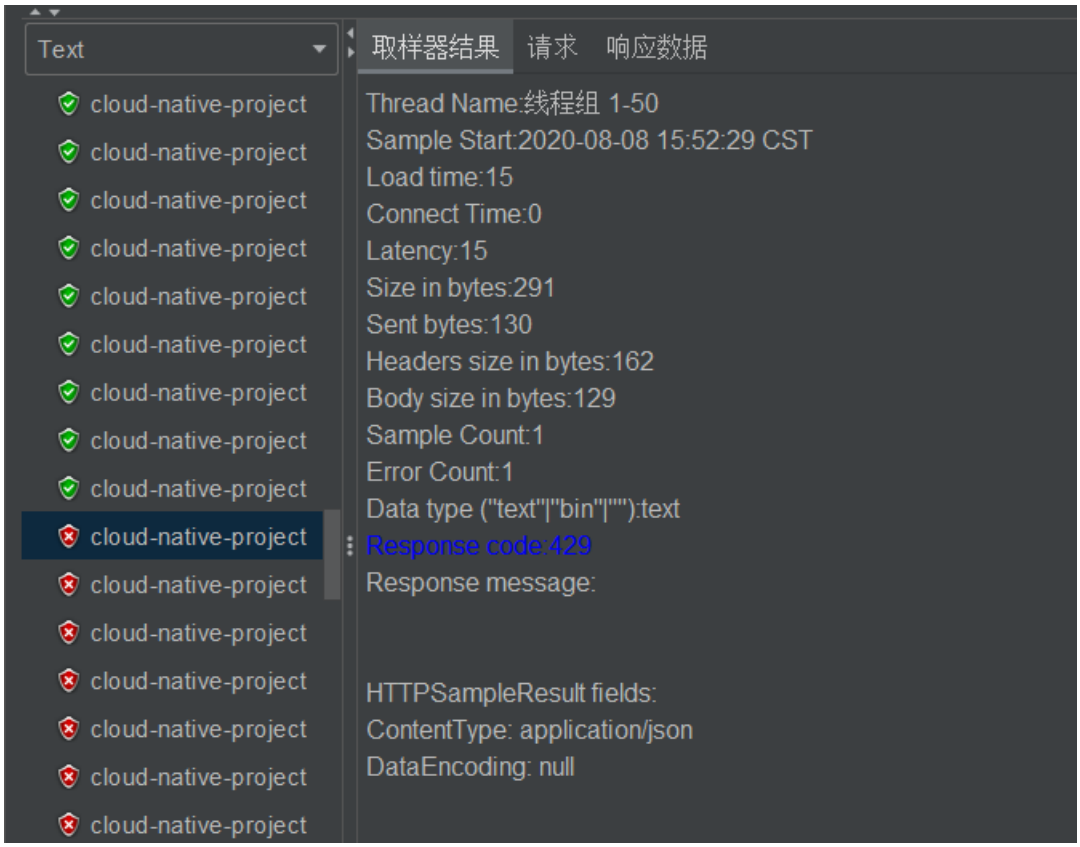
接口和请求

接口测试，成功



压力测试（每秒 100 个请求，共 2 秒），刚开始成功，后来返回 429





压力测试后的 Grafana

手动扩容

```
[cn202006@host-172-29-4-47 ~] eth0 = 172.29.4.47
$ kubectl scale deployment  cloud-native-project --replicas=4 -n cn202006
deployment.apps/cloud-native-project scaled
```

```
[cn202006@host-172-29-4-47 ~] eth0 = 172.29.4.47
$ kubectl  get deployment -n cn202006
NAME                    READY   UP-TO-DATE   AVAILABLE   AGE
cloud-native-project    4/4     4            4           3h37m
rtf                     0/1     1            0           118m
[cn202006@host-172-29-4-47 ~] eth0 = 172.29.4.47
$ kubectl get pods -n cn202006
NAME                                    READY   STATUS         RESTARTS   AGE
cloud-native-project-7885587b8-6xfq2    1/1     Running        0          89s
cloud-native-project-7885587b8-99x6j    1/1     Running        0          90s
cloud-native-project-7885587b8-9tdwf    1/1     Running        0          17m
cloud-native-project-7885587b8-shfjc    1/1     Running        0          89s
rtf-769f7d48f6-75vgp                    0/1     ErrImagePull   0          118m
```

扩容后压力测试