Spring Cloud on k8s

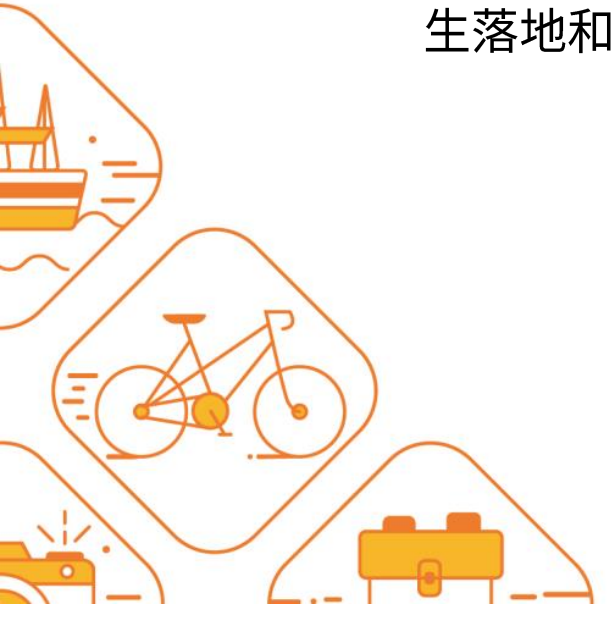
王 琼

UCLOUD 优刻得


自我介绍

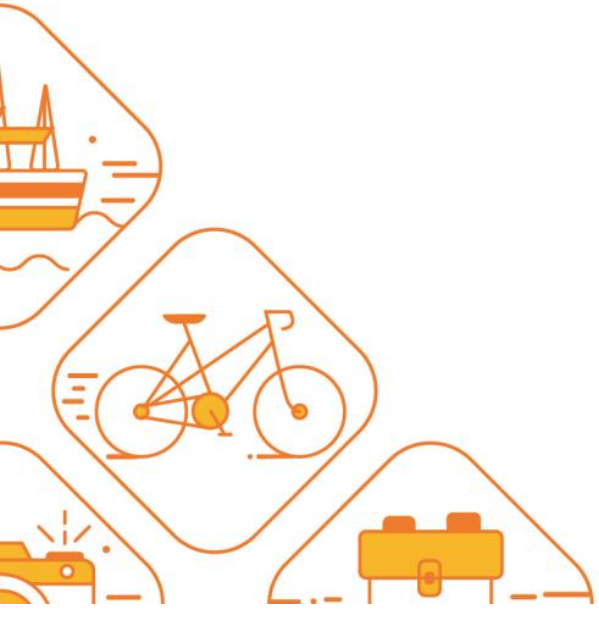
王琼 / 要出发周边游 / 运维经理

2016年开始接触k8s，在K8S以及Service Mesh领域持续深耕，致力于搭建生产级可用的容器服务平台。目前是要出发周边游运维架构师兼运维经理，负责公司云原生落地和企业容器化改造。




目 录


Spring Cloud 迁移至 K8S




Spring Cloud 介绍

Spring Cloud最早在功能层面为为服务治理定义了一系列的标准，例如智能路由、熔断机制、服务注册与发现等，并提供了对应的库和组件来实现这些标准特性。到目前为止，这些库和组件都已被广泛应用。

但是Spring cloud 也有一些缺点，例如：

- 即博采众家之长，也导致一些散乱的局面，即用户需要学习成本和熟悉各组件的“方言”并分别加以运维，这在客观上提高了应用门槛。
- 需要代码级别对诸多组件进行控制，包括Sidecar在内的组件都被依赖JAVA的实现，这和微服务的多语言协作目标是背道而驰的。
- 自身没有调度系统、资源、DevOps等提供相关支持，需要借助其他平台来完成，然后目前的容器编排事实标准是K8S，二者的部分功能存在重合或者冲突。

K8S 介绍

Kubernetes主要由以下几个核心组件组成：

- etcd保存了整个集群的状态；
- apiserver提供了资源操作的唯一入口，并提供认证、授权、访问控制、API注册和发现等机制
- controller manager负责维护集群的状态，比如故障检测、自动扩展、滚动更新等
- scheduler负责资源的调度，按照预定的调度策略将Pod调度到相应的机器上
- kubelet负责维护容器的生命周期，同时也负责Volume（CSI）和网络（CNI）的管理
- Container runtime负责镜像管理以及Pod和容器的真正运行（CRI）
- kube-proxy负责为Service提供Cluster内部的服务发现和负载均衡

除了核心组件，还有一些推荐的Add-ons：

- kube-dns负责为整个集群提供DNS服务
- Ingress Controller为服务提供外网入口
- metric-server提供资源监控

Spring cloud vs k8s

microervices concern	spring cloud & netfix oss	kubernetes
configuration management	conifg server,consul,netflix archaius	kubernetes configmap & secrets
service discovery	netflix eureka,hashicorp consul	kubernetes service & ingress resources
load balancing	netflix ribbon	kubernetes service
api gateway	netflix zuuk	kubernetes service & ingress resources
service security	spring cloud security	-
centralized logging	elk stack (logstash)	efk stach (fluentd)
centralized metrics	netflix spectator & atlas	heapster,promethues,grafana
centralized tracing	spring cloud sleuth,zipkin	opentracing,zipkin
resilience & fault tolerance	netflix hystrix,turbine & ribbon	kubernetes health check & resource isolation
auto scaling & self healing	-	kubernetes health check,self healing autoscaling
packaging,deployment & scheduling	spring boot	docker/rkt,kubernetes scheduler & deployment
job management	spring batch	kubernetes jobs & scheduled jobs
singleton application	spring cloud cluster	kubernetes pods

Spring Cloud 框架

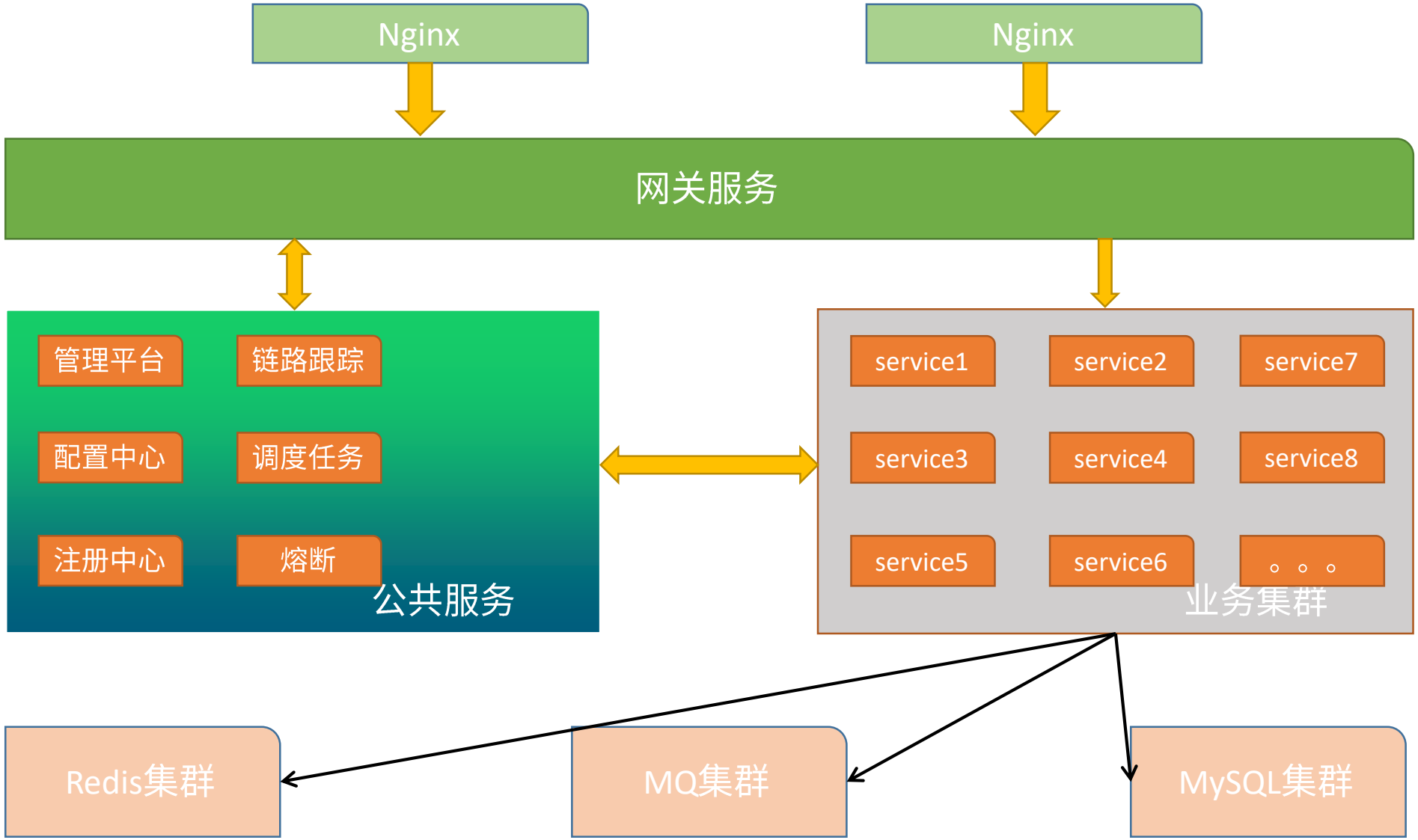
服务发现
Netflix Eureka

断路器
Netflix Hystrix

服务网关
Netflix Zuul
Spring Cloud gateway

分布式配置—Spring Cloud Config

客服端负载均衡
feign



改造

去掉

- spring-cloud-starter-eureka
- spring-cloud-starter-config

引入

- spring-cloud-kubernetes-discovery
- spring-cloud-kubernetes-ribbon

Property Key	Type	Default Value
spring.cloud.kubernetes.ribbon.enabled	boolean	true
spring.cloud.kubernetes.ribbon.mode	KubernetesRibbonMode	POD
spring.cloud.kubernetes.ribbon.cluster-domain	string	cluster.local

参考文档

https://cloud.spring.io/spring-cloud-static/spring-cloud-kubernetes/1.1.0.RC1/reference/html/#propertysource-reload

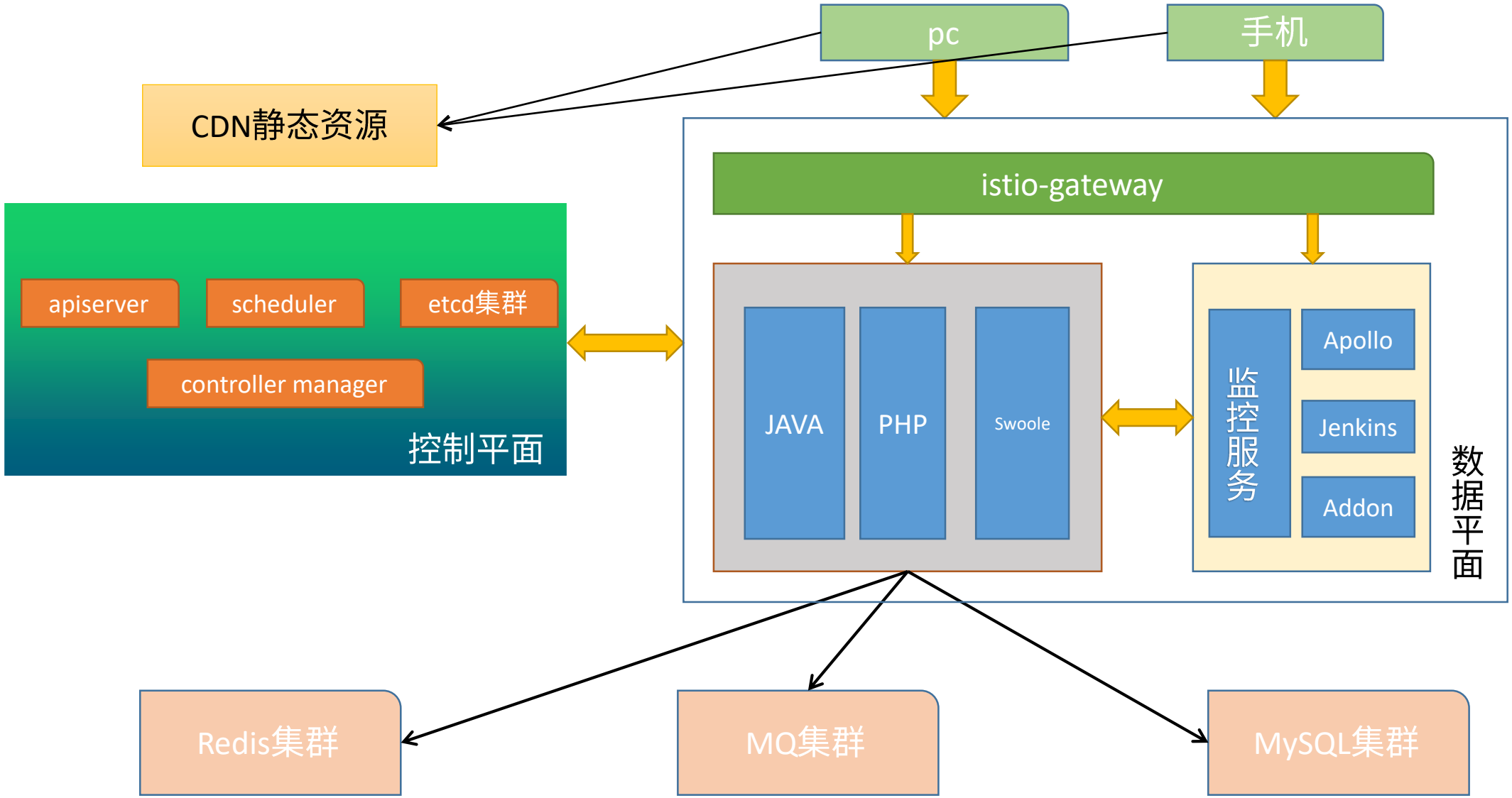
改造

新增

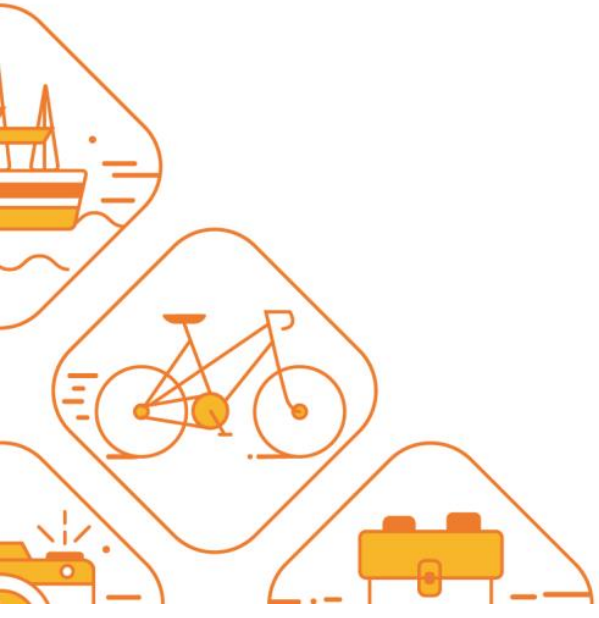
- Apollo配置中心
- Prometheus监控
- HPA自定义指标
- Istio服务治理

参考文档
https://cloud.spring.io/spring-cloud-static/spring-cloud-kubernetes/1.1.0.RC1/reference/html/#propertysource-reload

K8S框架


pc
手机
CDN静态资源
istio-gateway
apiserver
scheduler
etcd集群
controller manager
控制平面
JAVA
PHP
Swoole
监控服务
Apollo
Jenkins
Addon
数据平面
Redis集群
MQ集群
MySQL集群






Istio使用场景

微服务代理选型

7个关键标准

- 应用程序安全性
- 可观察性
- 持续部署
- 弹性伸缩和性能
- 对开源工具的集成
- Istio对开源控制平面的支持
- 所需的IT技术栈

参考文档
https://mp.weixin.qq.com/s/An6DBi_QsM9yoNpDvYCEww

微服务代理选型

- ingress-nginx
- haproxy
- traefik
- contour
- kong
- istio

网关

Istio-gateway: Ingress Gateway 在逻辑上相当于网格边缘的一个负载均衡器，用于接收和处理网格边缘出站和入站的网络连接，其中包含开放端口和TLS的配置等内容

Mesh: Istio内部的虚拟Gateway，代表网格内部的所有Sidecar，所有网格内部服务之间的互相通信，都是同个这个网关进行的。

http流量管理

定义目标规则:

- 使用Pod标签对具体的服务进程进行分组
- 可以定义服务的负载均衡策略
- 可以为服务指定TLS要求
- 可以为服务设置连接池的大小
- 流量的拆分和迁移

http流量管理

路由:

- 根据来源服务进行路由
- 根据URL进行重定向实现目标路由分流

超时控制&&重试:

- 在应用无感知的情况下，根据修改vs的配置，动态调整调用过程中的超时上限，从而到达控制故障范围的目的
- 可以直接在运行环境中对故障重试行为进行调整，能够极大的增强系统弹性和健壮性

熔断&&故障注入:

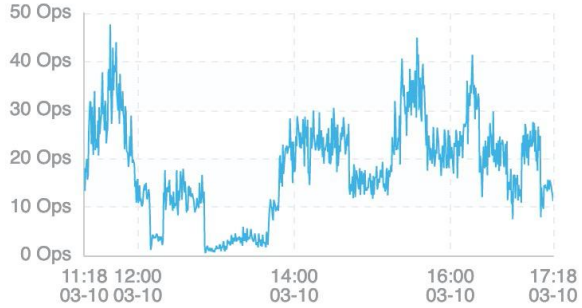
- 服务实例无法正常提供服务的情况下，将其从负载均衡池中摘除
- 请求中注入中断或者延迟，来阻止某些情况下的服务访问

限流:

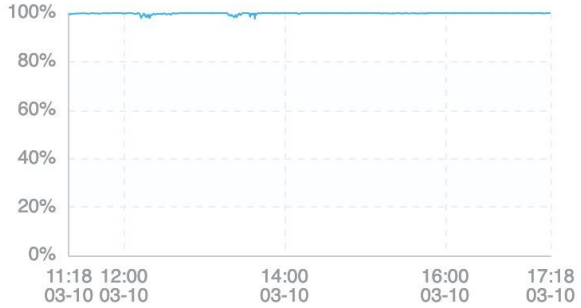
- MenQuota
- RedisQuota

遥测

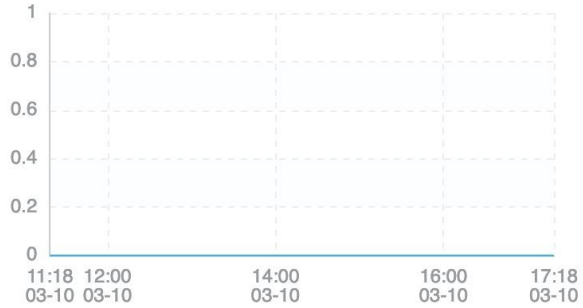
Project Request Volume



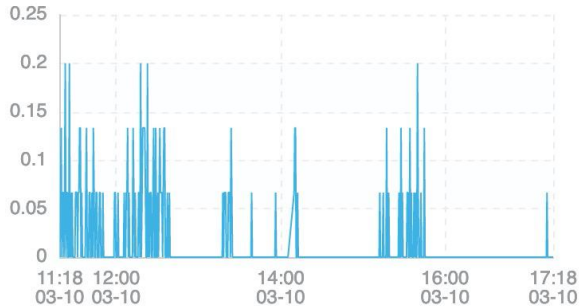
项目成功率



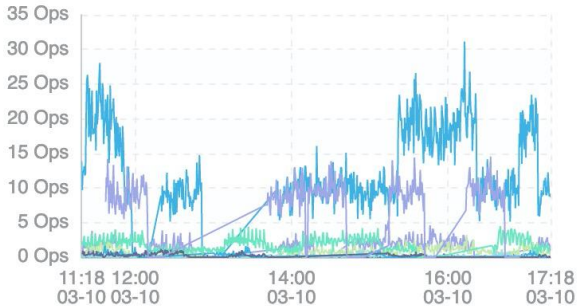
项目4xx错误代码计数



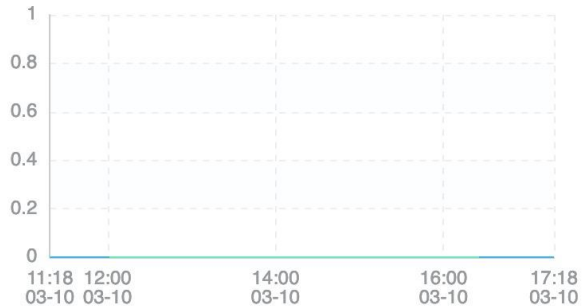
项目5xx错误代码计数



Request Volume by Service

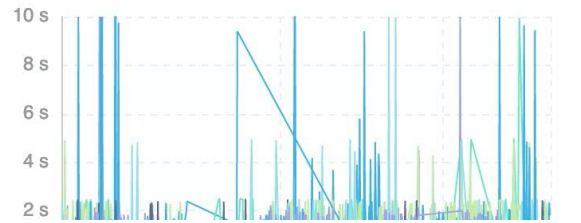


服务4xx错误代码计数

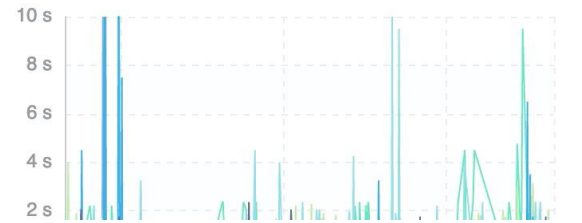


服务5xx错误代码计数

P99 Request Latency by Service



P90 Request Latency by Service



加密

MTLS:
Istio提供了无侵入的安全解决方案，能够提供网格内部、网格和边缘之间的安全通信和访问控制能力

JVM参数优化 && 配置介绍

JVM 参数优化

-XX:+UnlockExperimentalVMOptions
-XX:+UseCGroupMemoryLimitForHeap

-XX:MinRAMPercentage=20.0
-XX:MaxRAMPercentage=50.0

-XX:ActiveProcessorCount=$limitscpu

-XX:+HeapDumpOnOutOfMemoryError

UseContainerSupport

```
resources:
  limits:
    cpu: $limitscpu
    memory: $limitsmemory
  requests:
    cpu: $requestscpu
    memory: $requestsmemory
```

参考文档
https://merikan.com/2019/04/jvm-in-a-container/

K8S 配置介绍

```
apiVersion: apps/v1beta1
kind: Deployment
metadata:
  name: program_name-v1
  namespace: namespace_env
spec:
  replicas: replicaset
  strategy:
    rollingUpdate:
      maxSurge: 25%
      maxUnavailable: 25%
    type: RollingUpdate
  template:
    metadata:
      labels:
        app: program_name
        version: v1
```

K8S 配置介绍

```
affinity:
 nodeAffinity: requiredDuringSchedulingIgnoredDuringExecution:
    nodeSelectorTerms:
    - matchExpressions:
      - {key: node.application.java/type, operator: In, values: ["java"]}
 podAntiAffinity: preferredDuringSchedulingIgnoredDuringExecution:
  - weight: 100
    podAffinityTerm:
     labelSelector:
       matchExpressions:
         - {key: app, operator: In, values: ["program_name"]}
     topologyKey: kubernetes.io/hostname
```

K8S 配置介绍

```
livenessProbe:
  failureThreshold: 3
  httpGet:
    path: /actuator/info
    port: 9108
    scheme: HTTP
  initialDelaySeconds: 30
  periodSeconds: 15
  successThreshold: 1
  timeoutSeconds: 60
```

```
readinessProbe:
  failureThreshold: 3
  httpGet:
    path: /actuator/info
    port: 9108
    scheme: HTTP
  initialDelaySeconds: 30
  periodSeconds: 15
  successThreshold: 1
  timeoutSeconds: 60
```

K8S 配置介绍

```
        lifecycle:
          preStop:
            exec:
              command: ["/bin/bash", "-c", "PID=`pidof java` && kill -SIGTERM $PID && while ps -p $PID >
/dev/null; do sleep 1; done;"]
```

K8S 配置介绍

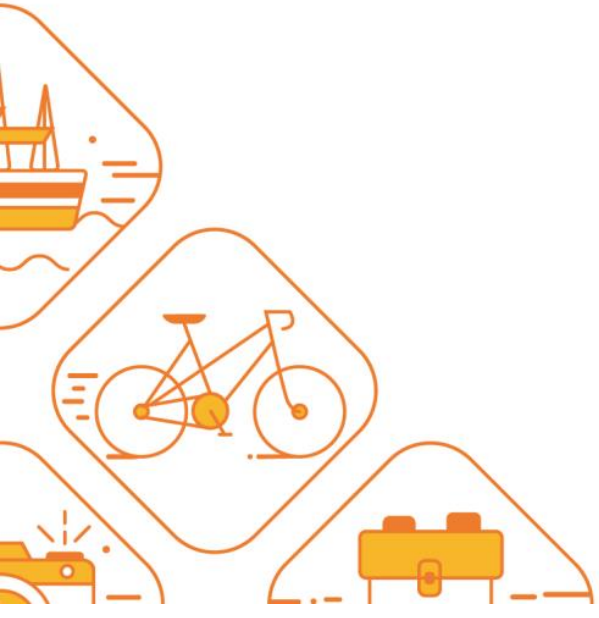
HPA自定义指标:

通过Prometheus获取http_server_requests_seconds_count实时动态调整Pod数量

需要安装Prometheus-adapt将Prometheus获取到参数转化成apiserver能识别的参数

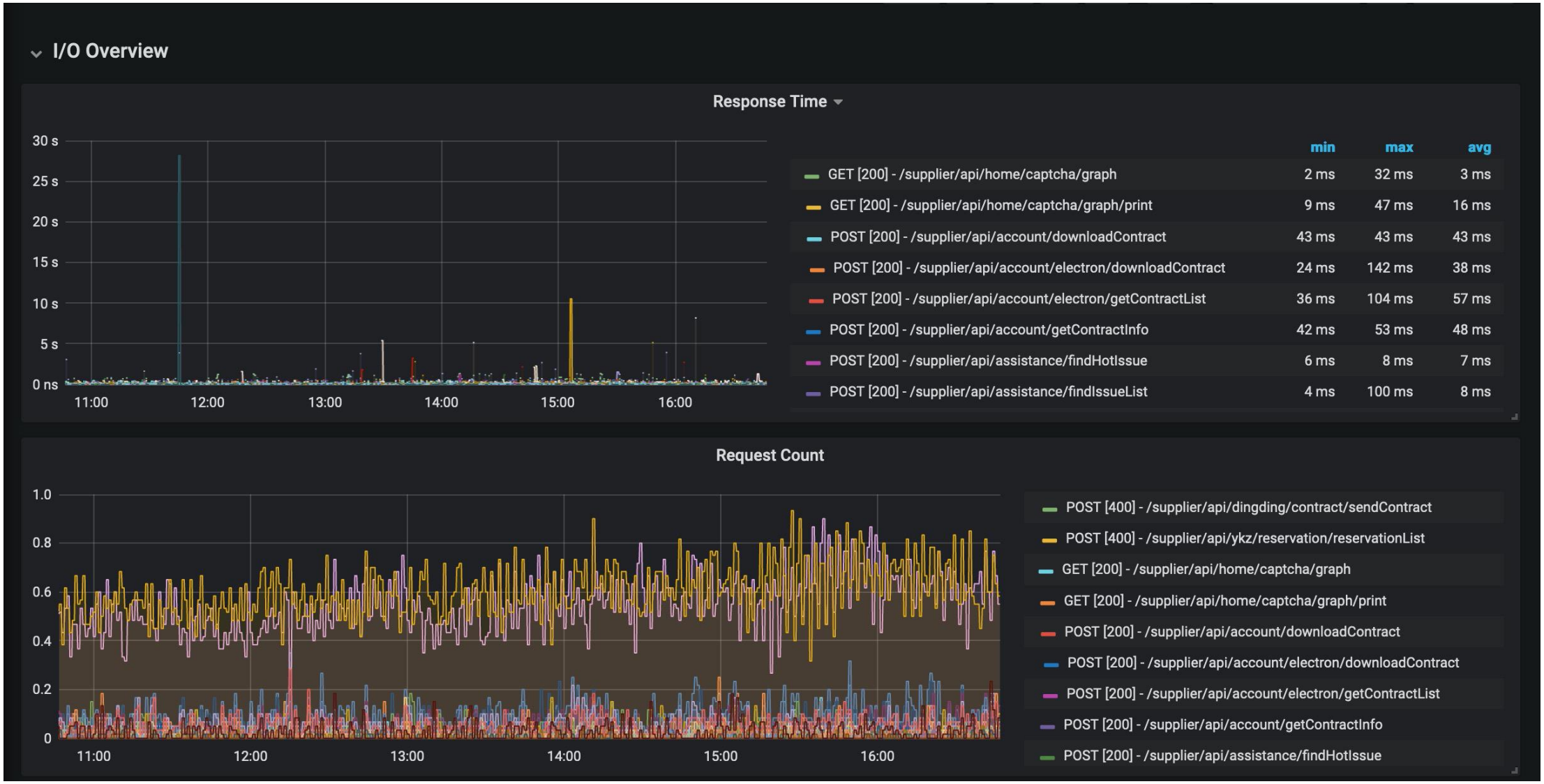
```
rules:
- seriesQuery: '{__name__=~"^http_server_requests_.*",pod!=""}'
  seriesFilters: []
  resources:
    overrides:
      namespace:
        resource: namespace
      pod:
        resource: pod
  name:
    matches: "^http_server_requests_(.*)_count$"
    as: "http_server_requests_${1}_count"
  metricsQuery: 'sum(irate(<<.Series>>{<<.LabelMatchers>>,pod!="",uri!~".*actuator.*"}[5m])) by (<<.G
```



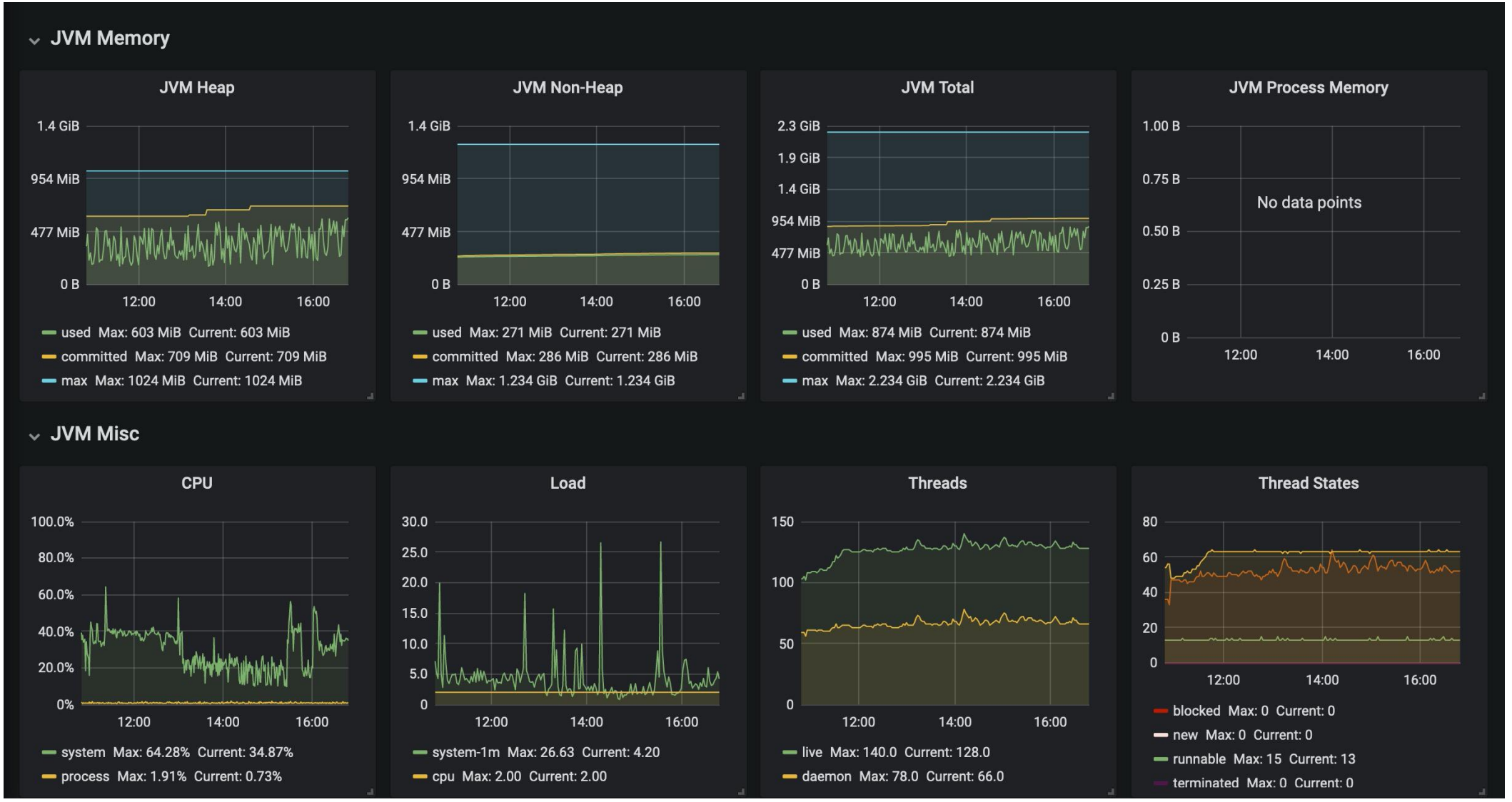



监控 && 链路跟踪

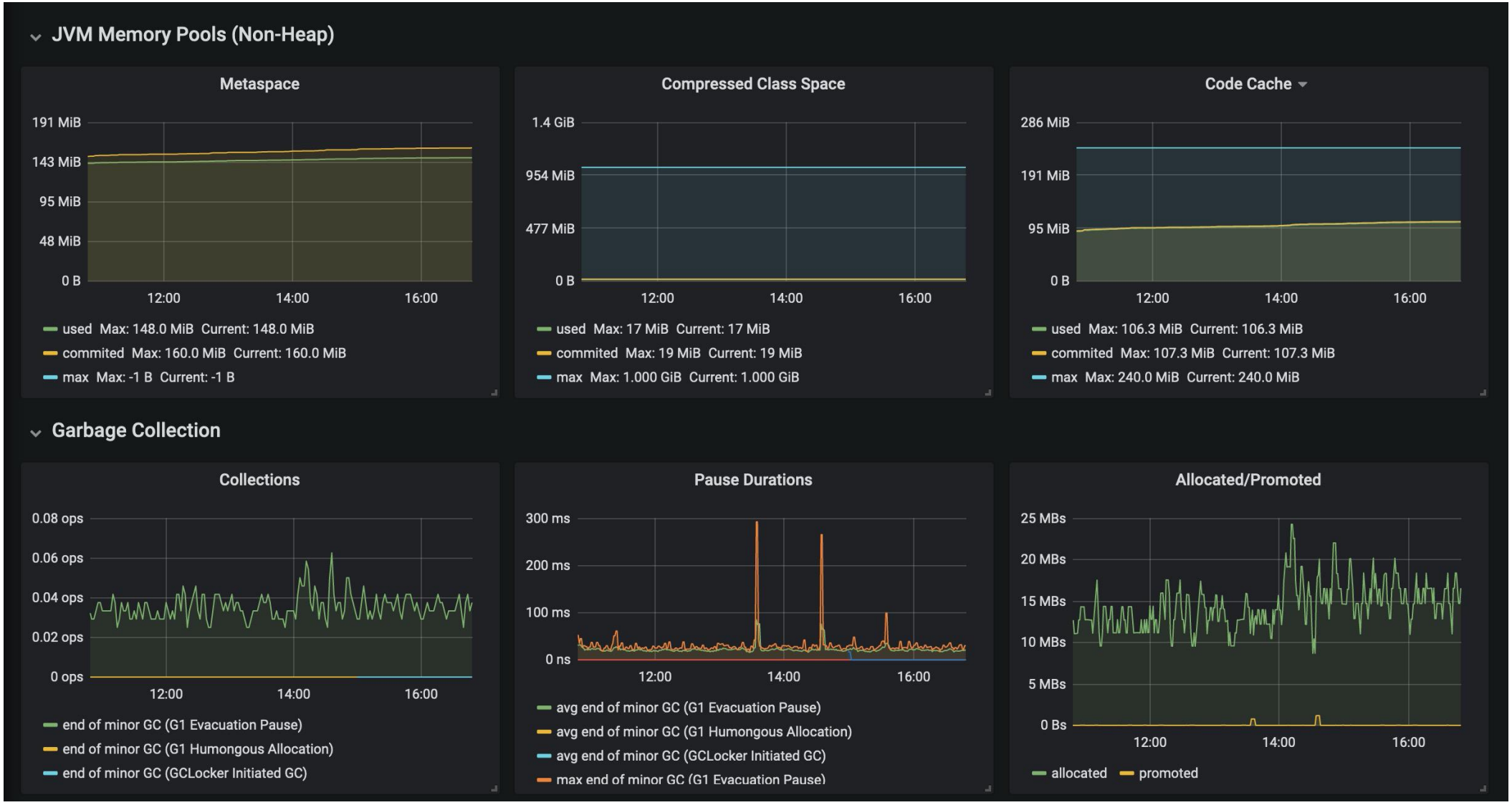
基于Prometheus的JVM监控



基于Prometheus的JVM监控



基于Prometheus的JVM监控



链路跟踪

	pinpoint	zipkin	jaeger	skywalking
OpenTracing兼容	否	是	是	是
客户端支持语言	java、php	java,c#,go,php等	java,c#,go,php等	Java, .NET Core, NodeJS and PHP
存储	hbase	ES， mysql,Cassandra,内存	ES， kafka,Cassandra,内存	ES， H2,mysql,TIDB,sharding sphere
传输协议支持	thrift	http,MQ	udp/http	gRPC
ui丰富程度	高	低	中	中
实现方式-代码侵入性	字节码注入， 无侵入	拦截请求， 侵入	拦截请求， 侵入	字节码注入， 无侵入
扩展性	低	高	高	中
trace查询	不支持	支持	支持	支持
告警支持	支持	不支持	不支持	支持
jvm监控	支持	不支持	不支持	支持
性能损失	高	中	中	低

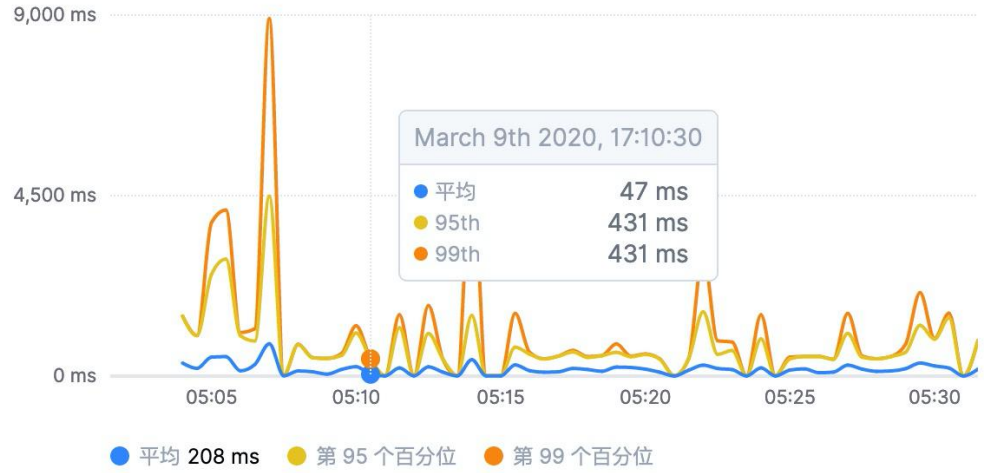
基于Elastic的APM

搜索事务和错误.....（例如 transaction.duration.us > 300000 AND context.resp

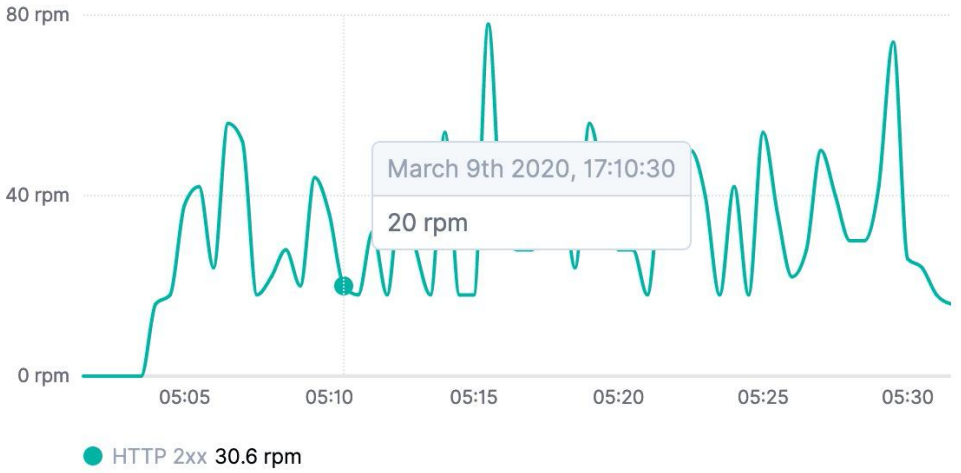
Last 30 minutes　Show dates　Refresh

事务　错误　指标

事务持续时间



每分钟请求数



名称	平均持续时间	第 95 个百分位 ↓	每分钟事务数	影响
CrmSaleDataController#getSaleResultDataList	2,139 ms	5,199 ms	0.4 tpm	
CrmDataStatisticsController#getHumanList	3,983 ms	4,141 ms	0.1 tpm	
DataStatisticalController#getSaleInfo	688 ms	1,272 ms	1.4 tpm	
CrmSaleDataController#getSaleProcessDataList	579 ms	1,256 ms	0.1 tpm	

基于Elastic的APM

基于Elastic的APM

UCLOUD 优刻得

THANKS