NGINX从入门到精通进阶系列培训

实践篇：现代应用可观测

林静　Software Solution Architect

https://myf5.net

目录

为什么可观测性对现代应用很重要?

什么是可观测性?

可观测性是衡量从外部输出中推断系统内部运行状态的程度 (维基百科)

现代应用的技术特征是什么?

容器化，微服务，PaaS，Cloud

你要回答的问题和你采取的措施

请求通过了哪些服务?

每个微服务在处理请求时做了什么?

如果请求很慢，瓶颈在哪里?

如果请求失败，错误发生在哪里?

请求的执行与系统的正常行为有何不同?

应用容器运行在哪里？那里发生了什么？

容器里工具不全，要啥啥没有！

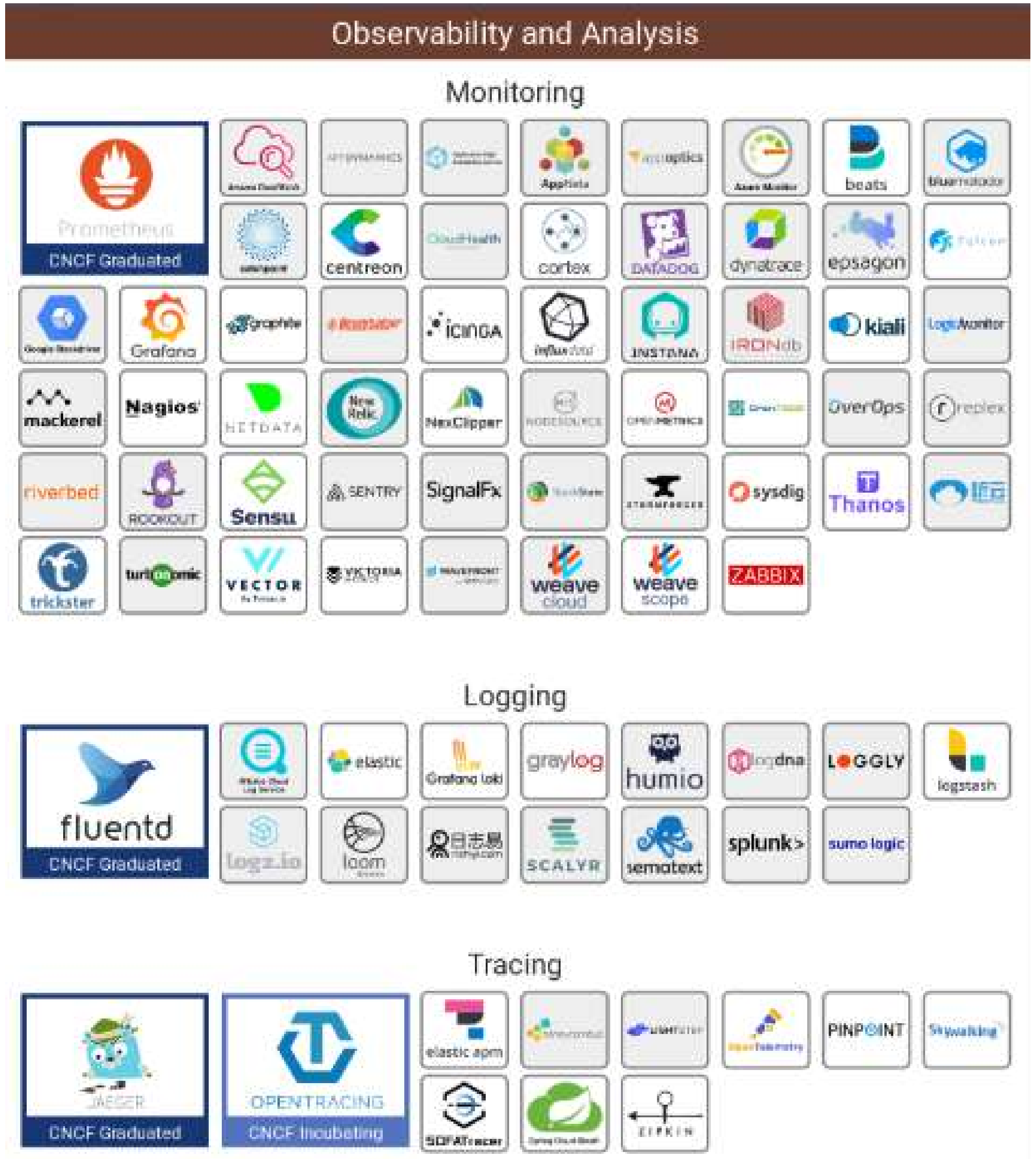
这个错误日志是表达这个应用本身的问题吗？

这个服务和谁依赖，当时到底调用了哪些服务？

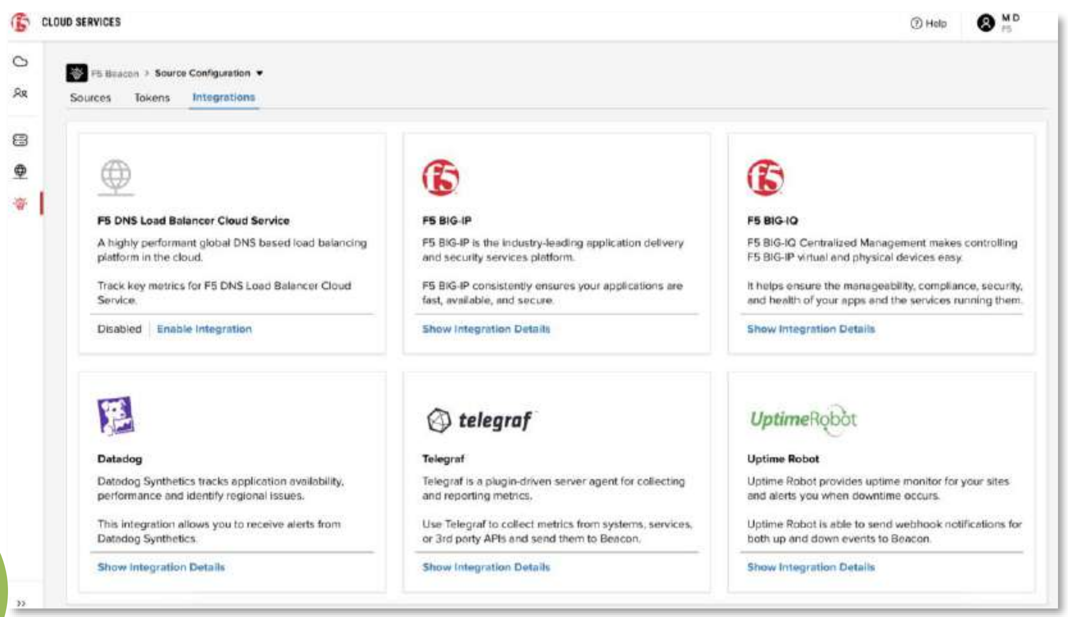
抓包，镜像数据？

到底是网络的问题还是应用的问题?

CNCF


Observability and Analysis
Monitoring
Prometheus
CNCF Graduated
Logging
fluentd
CNCF Graduated
Tracing
JAEGER
CNCF Graduated
OPENTRACING
CNCF Incubating

F5在可观测领域战略



CLOUD SERVICES

Help

M D

F5 Beacon > Source Configuration

Sources | Tokens | Integrations

F5 DNS Load Balancer Cloud Service

A highly performant global DNS based load balancing platform in the cloud.

Track key metrics for F5 DNS Load Balancer Cloud Service.

Disabled | Enable Integration

F5 BIG-IP

F5 BIG-IP is the industry-leading application delivery and security services platform.

F5 BIG-IP consistently ensures your applications are fast, available, and secure.

Show Integration Details

F5 BIG-IQ

F5 BIG-IQ Centralized Management makes controlling F5 BIG-IP virtual and physical devices easy.

It helps ensure the manageability, compliance, security, and health of your apps and the services running them.

Show Integration Details

Datadog

Datadog Synthetics tracks application availability, performance and identify regional issues.

This integration allows you to receive alerts from Datadog Synthetics.

Show Integration Details

Telegraf

Telegraf is a plugin-driven server agent for collecting and reporting metrics.

Use Telegraf to collect metrics from systems, services, or 3rd party APIs and send them to Beacon.

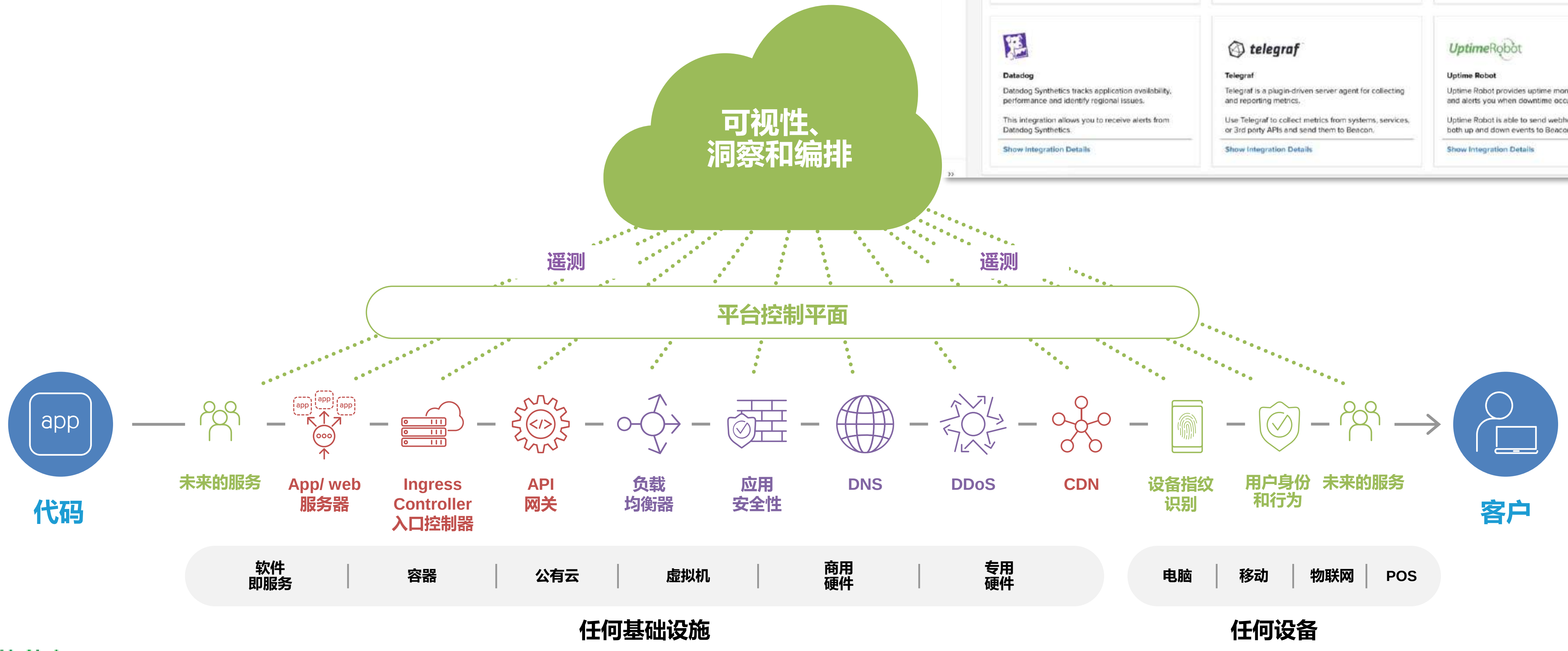
Show Integration Details

Uptime Robot

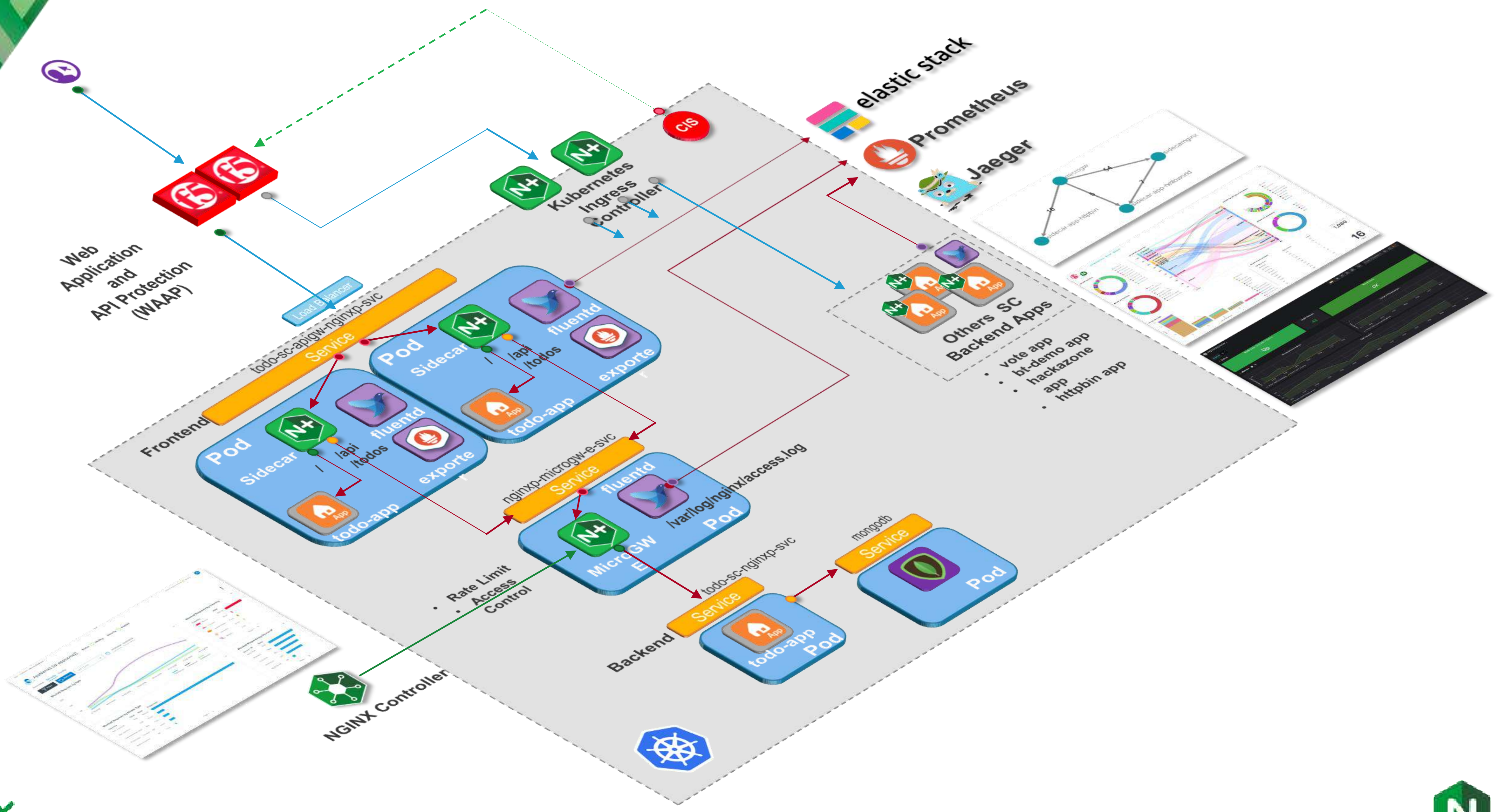
Uptime Robot provides uptime monitor for your sites and alerts you when downtime occurs.

Uptime Robot is able to send webhook notifications for both up and down events to Beacon.

Show Integration Details



🔵 BIG-IP　🔴 NGINX　🟢 未来

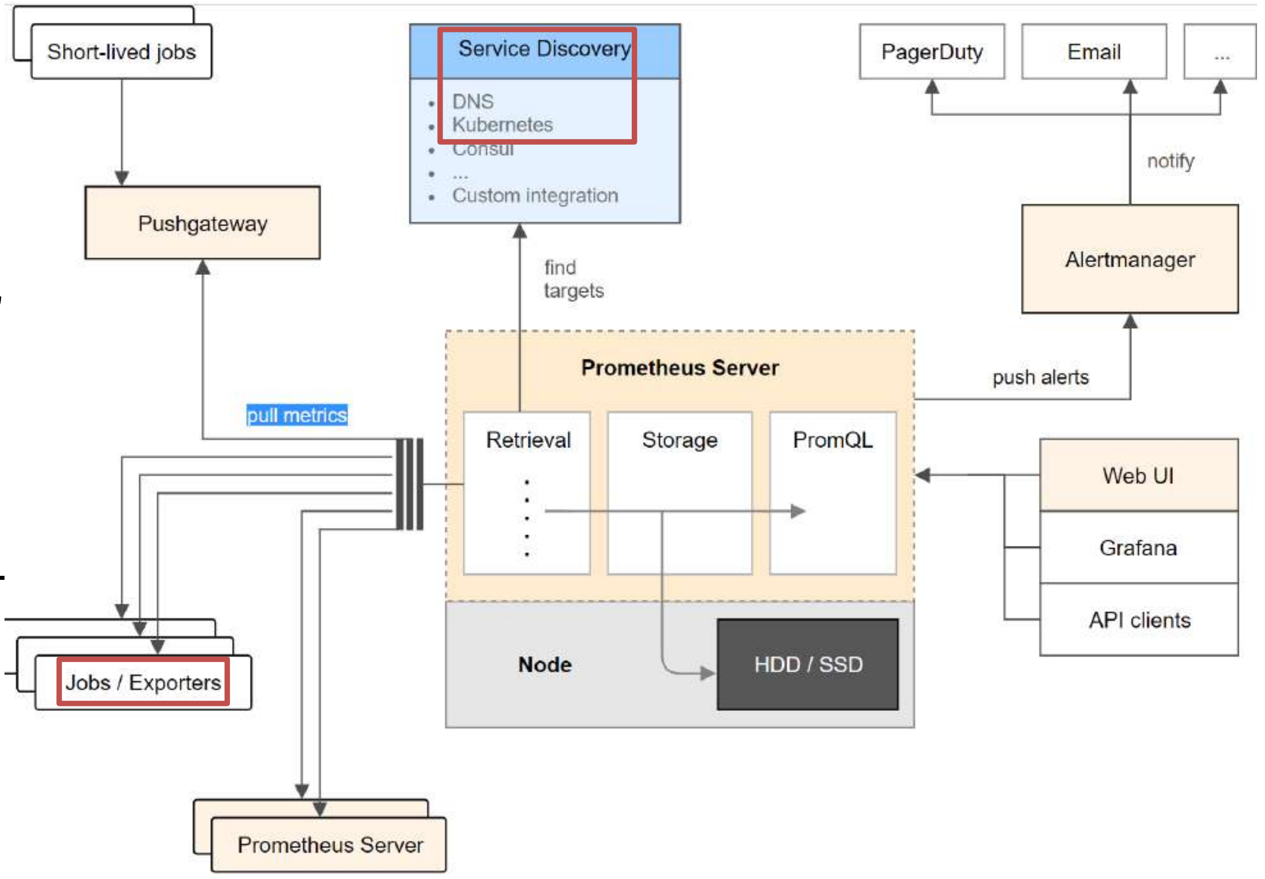

Web Application and API Protection (WAAP)
Load Balancer
CIS
Kubernetes Ingress Controller
elastic stack
Prometheus
Jaeger
todo-sc-apigw-nginxp-svc
Service
Frontend
Pod
Sidecar
fluentd
exporter
/
/api
/todos
App
todo-app
nginxp-microgw-e-svc
MicroGW E
/var/log/nginx/access.log
Others SC Backend Apps
vote app
bt-demo app
hackazone app
httpbin app
Rate Limit
Access Control
todo-sc-nginxp-svc
Backend
mongodb
NGINX Controller

NGINX与Prometheus监控应用性能指标

Prometheus速览

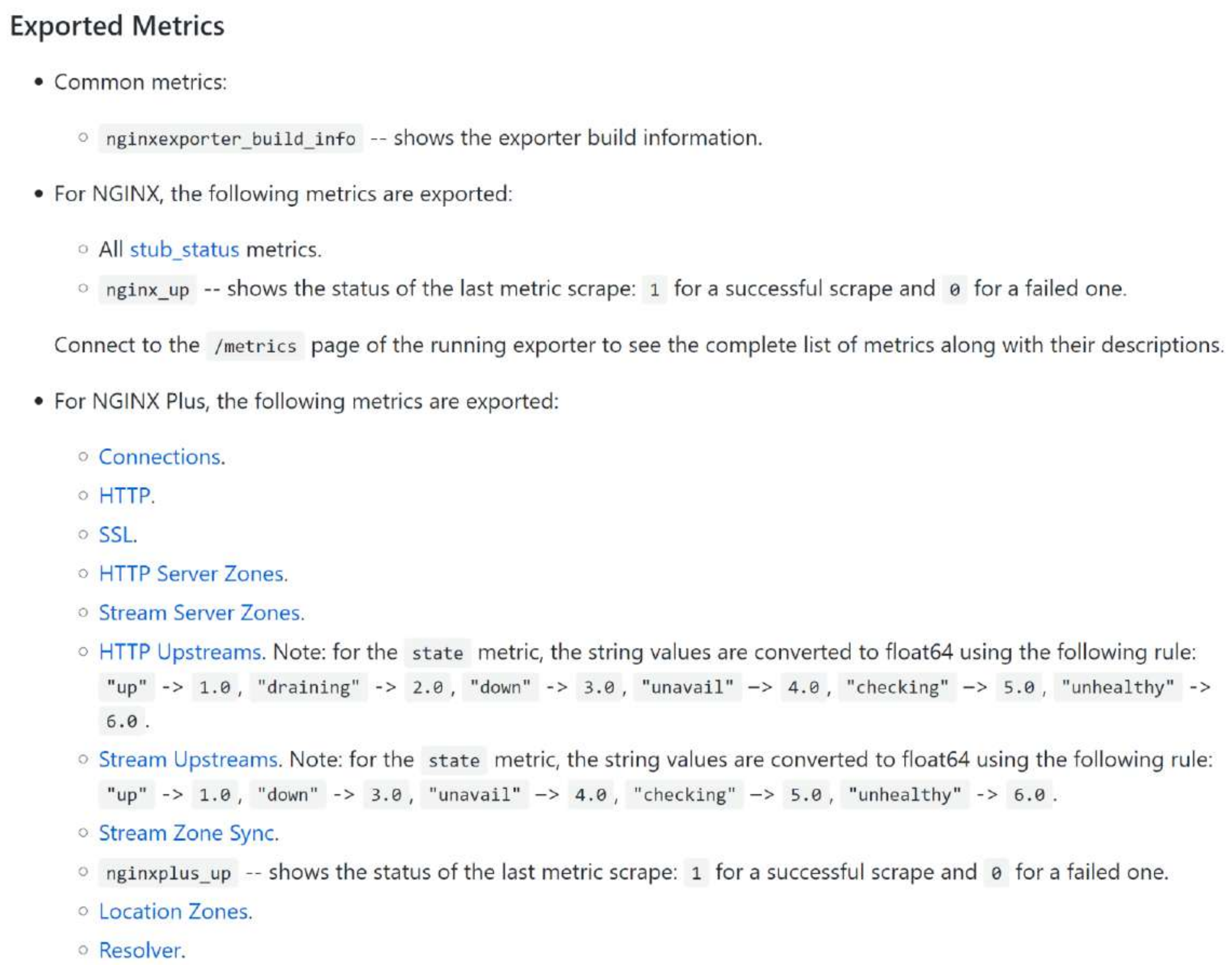
- CNCF 最”火”的项目
- Pull 模式为主，也支持通过pushgateway做push模式
- 目标主机通过 服务发现， 静态配置或者DNS方式发现
- 支持告警管理
- 时序数据库，非常适合实时metric处理
- 基于lables汇聚metrics

- Service discovery—config--- relabel-configs—pulling---metric-relabel-configs



https://github.com/prometheus/prometheus

NGINX如何与Prometheus结合?



Exported Metrics

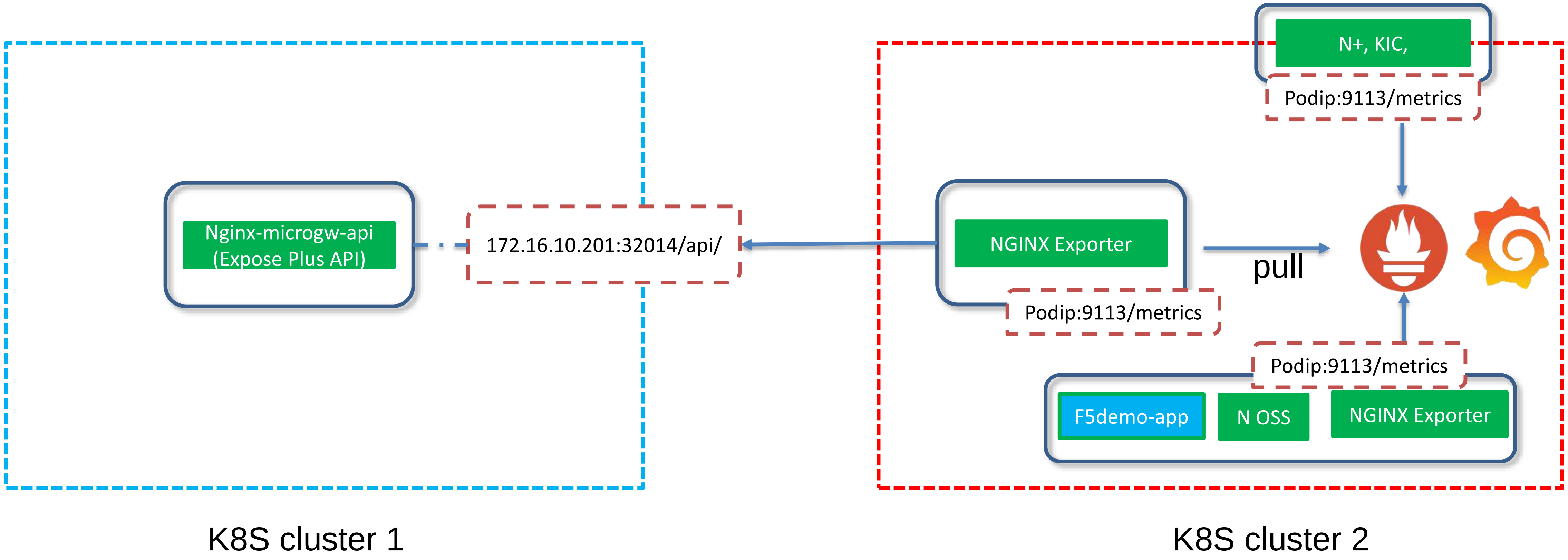
- Common metrics:
  - `nginxexporter_build_info` -- shows the exporter build information.
- For NGINX, the following metrics are exported:
  - All stub_status metrics.
  - `nginx_up` -- shows the status of the last metric scrape: `1` for a successful scrape and `0` for a failed one.

  Connect to the `/metrics` page of the running exporter to see the complete list of metrics along with their descriptions.
- For NGINX Plus, the following metrics are exported:
  - Connections.
  - HTTP.
  - SSL.
  - HTTP Server Zones.
  - Stream Server Zones.
  - HTTP Upstreams. Note: for the `state` metric, the string values are converted to float64 using the following rule: `"up"` -> `1.0`, `"draining"` -> `2.0`, `"down"` -> `3.0`, `"unavail"` –> `4.0`, `"checking"` –> `5.0`, `"unhealthy"` -> `6.0`.
  - Stream Upstreams. Note: for the `state` metric, the string values are converted to float64 using the following rule: `"up"` -> `1.0`, `"down"` -> `3.0`, `"unavail"` –> `4.0`, `"checking"` –> `5.0`, `"unhealthy"` -> `6.0`.
  - Stream Zone Sync.
  - `nginxplus_up` -- shows the status of the last metric scrape: `1` for a successful scrape and `0` for a failed one.
  - Location Zones.
  - Resolver.

NGINX

https://github.com/nginxinc/nginx-prometheus-exporter

NGINX Exporter实验环境介绍



NGINX Exporter Demo

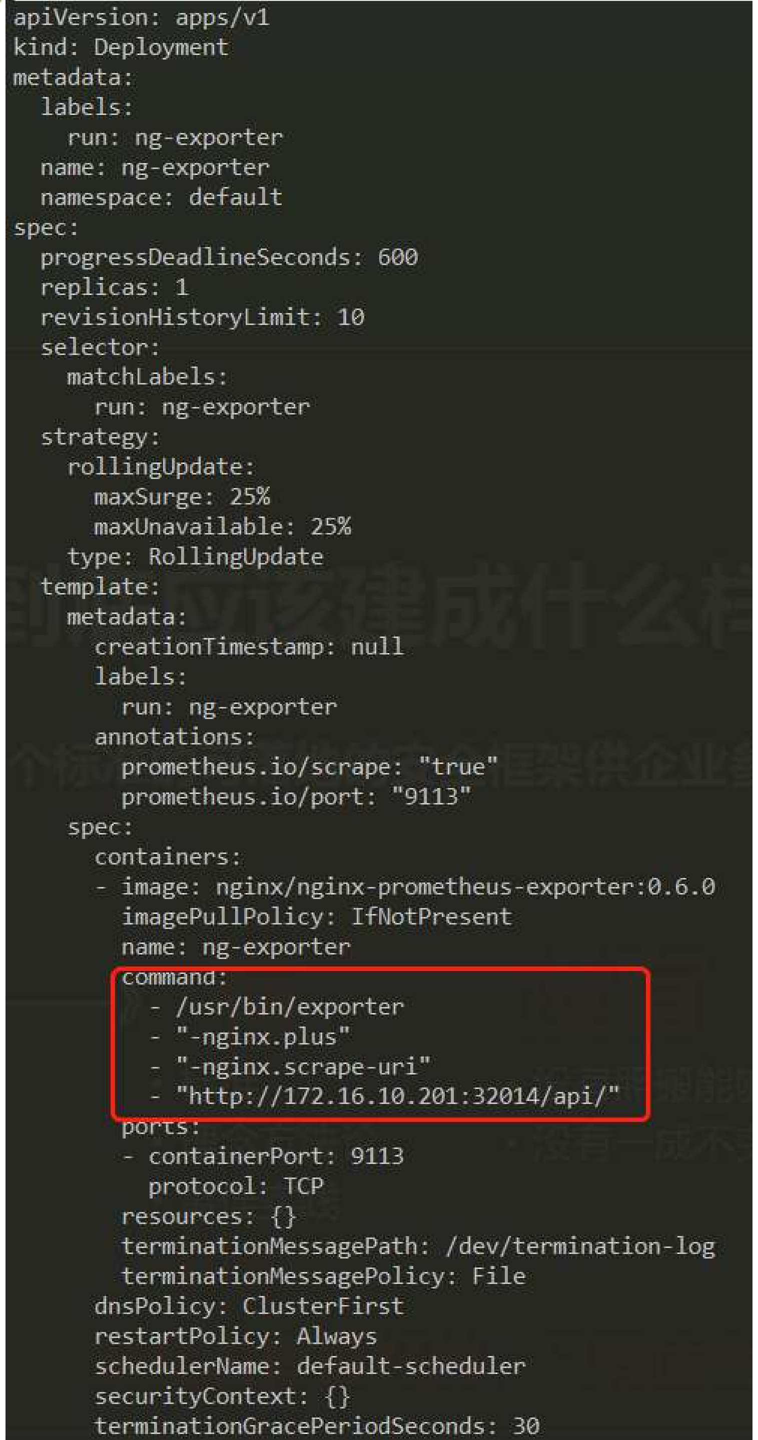
1. 部署一个独立的nginx exporter并让其暴露另一个ks8集群1中的micgw-api NGINX Plus信息
2. 做一些流量访问模拟
   https://apidemo-kic.lab.f5se.io/gotohelloworld
   https://apidemo-kic.lab.f5se.io/#/HTTP_Methods/get_get

3. 在prometheus中查看相关metrics是否出现，并做简单的检索测试
   nginxplus_upstream_server_state
   rate(nginxplus_upstream_server_requests[5m])

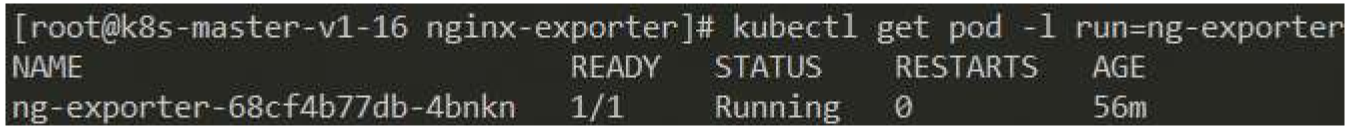
4. 暴露k8s集群2中的Ingress controller的metrics
5. 在prometheus中查看相关ingress controller的metrics
   rate(nginx_ingress_nginxplus_http_requests_total[5m])

6. 将nginx exporter 作为nginx的sidecar，暴露单个pod中的开源nginx信息
7.在prometheus中查看相关开源nginx的metrics
   rate(nginx_connections_accepted[5m])

NGINX Exporter Deployment



```
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    run: ng-exporter
  name: ng-exporter
  namespace: default
spec:
  progressDeadlineSeconds: 600
  replicas: 1
  revisionHistoryLimit: 10
  selector:
    matchLabels:
      run: ng-exporter
  strategy:
    rollingUpdate:
      maxSurge: 25%
      maxUnavailable: 25%
    type: RollingUpdate
  template:
    metadata:
      creationTimestamp: null
      labels:
        run: ng-exporter
      annotations:
        prometheus.io/scrape: "true"
        prometheus.io/port: "9113"
    spec:
      containers:
      - image: nginx/nginx-prometheus-exporter:0.6.0
        imagePullPolicy: IfNotPresent
        name: ng-exporter
        command:
          - /usr/bin/exporter
          - "-nginx.plus"
          - "-nginx.scrape-uri"
          - "http://172.16.10.201:32014/api/"
        ports:
        - containerPort: 9113
          protocol: TCP
        resources: {}
        terminationMessagePath: /dev/termination-log
        terminationMessagePolicy: File
      dnsPolicy: ClusterFirst
      restartPolicy: Always
      schedulerName: default-scheduler
      securityContext: {}
      terminationGracePeriodSeconds: 30
```



```
[root@k8s-master-v1-16 nginx-exporter]# kubectl get pod -l run=ng-exporter
NAME                           READY   STATUS    RESTARTS   AGE
ng-exporter-68cf4b77db-4bnkn   1/1     Running   0          56m
```

Nginx exporter连接k8s 1集群中的microgw的API

留个问题：这样部署，有啥可以改进的地方

同时，如果你有很多nginx实例，该咋办?

Prometheus-njs

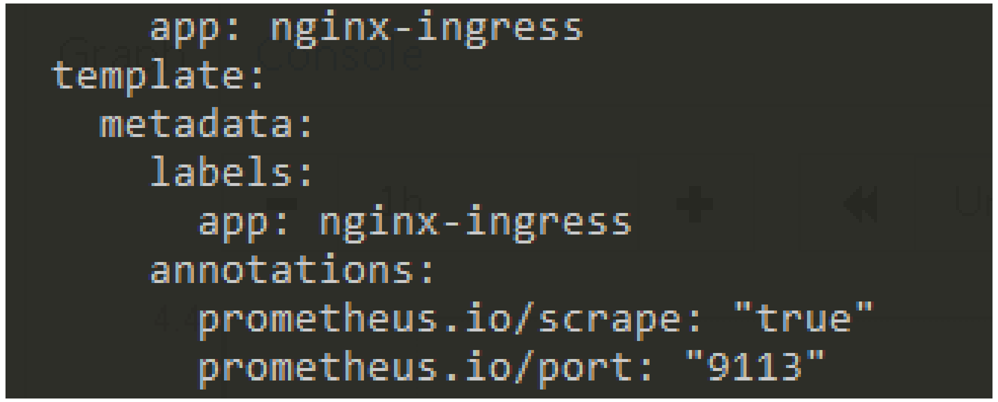
- 每个NGINX Plus暴露自己的metrics给Prometheus
- 避免expoter的 1:1关系，节省部署资源

- Only for Plus，使用API

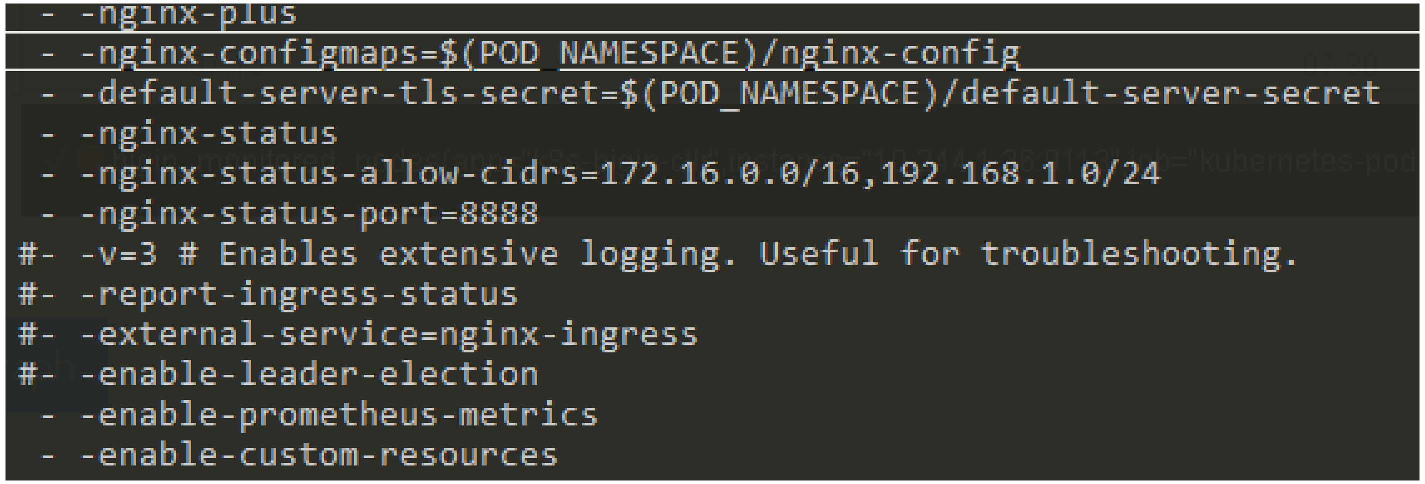
https://docs.nginx.com/nginx/admin-guide/dynamic-modules/prometheus-njs/

暴露k8s-2中的IC metrics

- -enable-prometheus-metrics

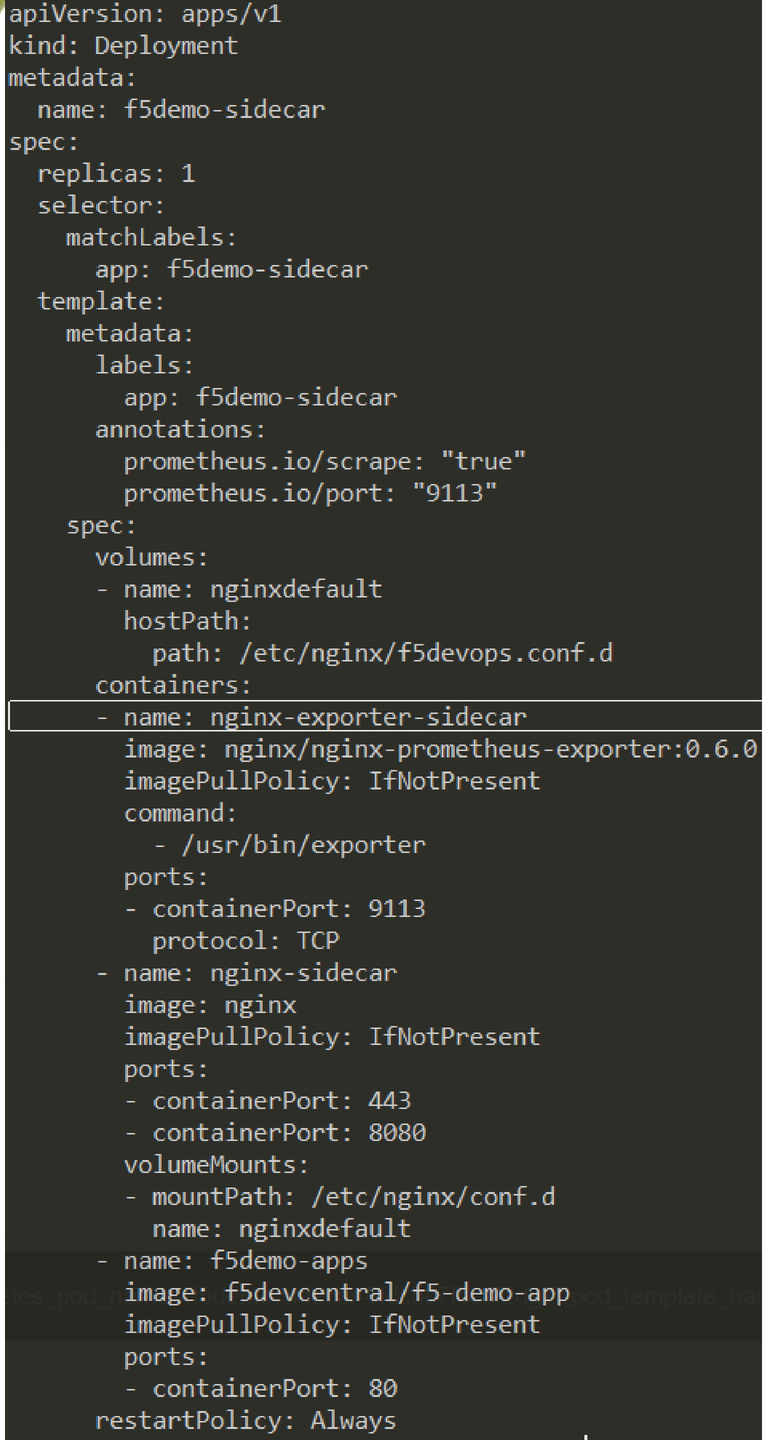


```
    app: nginx-ingress
template:
  metadata:
    labels:
      app: nginx-ingress
    annotations:
      prometheus.io/scrape: "true"
      prometheus.io/port: "9113"
```

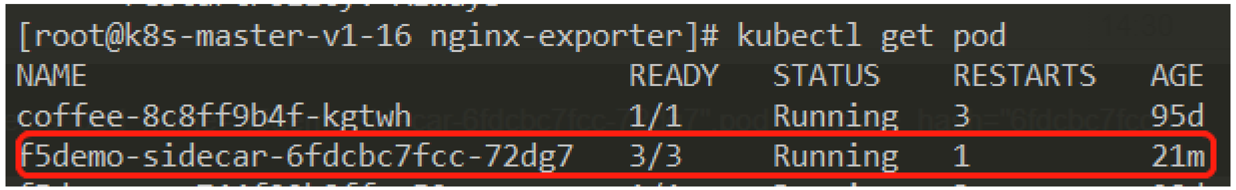


```
 -  -nginx-plus
 -  -nginx-configmaps=$(POD_NAMESPACE)/nginx-config
 -  -default-server-tls-secret=$(POD_NAMESPACE)/default-server-secret
 -  -nginx-status
 -  -nginx-status-allow-cidrs=172.16.0.0/16,192.168.1.0/24
 -  -nginx-status-port=8888
#-  -v=3 # Enables extensive logging. Useful for troubleshooting.
#-  -report-ingress-status
#-  -external-service=nginx-ingress
#-  -enable-leader-election
 -  -enable-prometheus-metrics
 -  -enable-custom-resources
```

NGINX Exporter作为sidecar

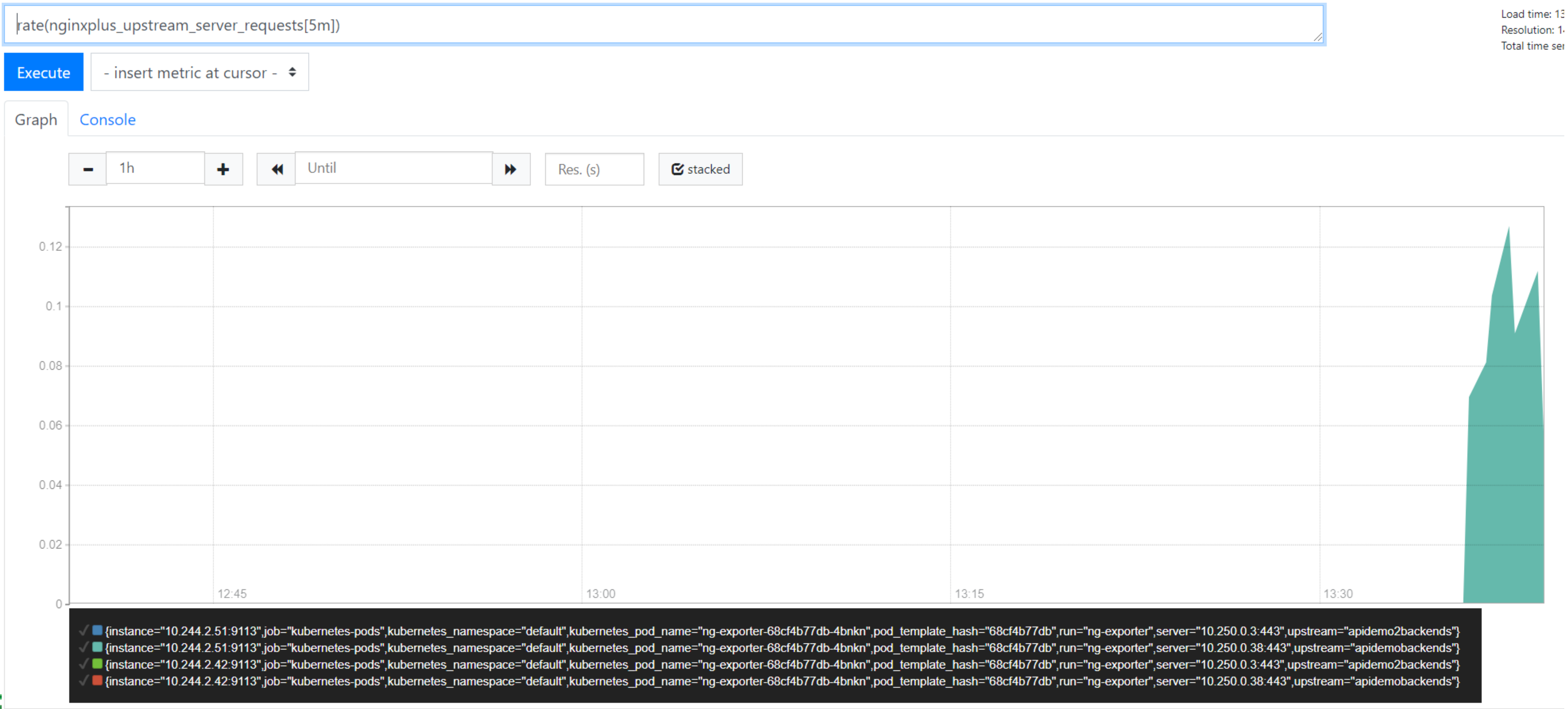


```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: f5demo-sidecar
spec:
  replicas: 1
  selector:
    matchLabels:
      app: f5demo-sidecar
  template:
    metadata:
      labels:
        app: f5demo-sidecar
      annotations:
        prometheus.io/scrape: "true"
        prometheus.io/port: "9113"
    spec:
      volumes:
      - name: nginxdefault
        hostPath:
          path: /etc/nginx/f5devops.conf.d
      containers:
      - name: nginx-exporter-sidecar
        image: nginx/nginx-prometheus-exporter:0.6.0
        imagePullPolicy: IfNotPresent
        command:
          - /usr/bin/exporter
        ports:
        - containerPort: 9113
          protocol: TCP
      - name: nginx-sidecar
        image: nginx
        imagePullPolicy: IfNotPresent
        ports:
        - containerPort: 443
        - containerPort: 8080
        volumeMounts:
        - mountPath: /etc/nginx/conf.d
          name: nginxdefault
      - name: f5demo-apps
        image: f5devcentral/f5-demo-app
        imagePullPolicy: IfNotPresent
        ports:
        - containerPort: 80
      restartPolicy: Always
```



```
[root@k8s-master-v1-16 nginx-exporter]# kubectl get pod
NAME                           READY   STATUS    RESTARTS   AGE
coffee-8c8ff9b4f-kgtwh         1/1     Running   3          95d
f5demo-sidecar-6fdcbc7fcc-72dg7   3/3     Running   1          21m
```

K8s-1的microgw Upsteam requests

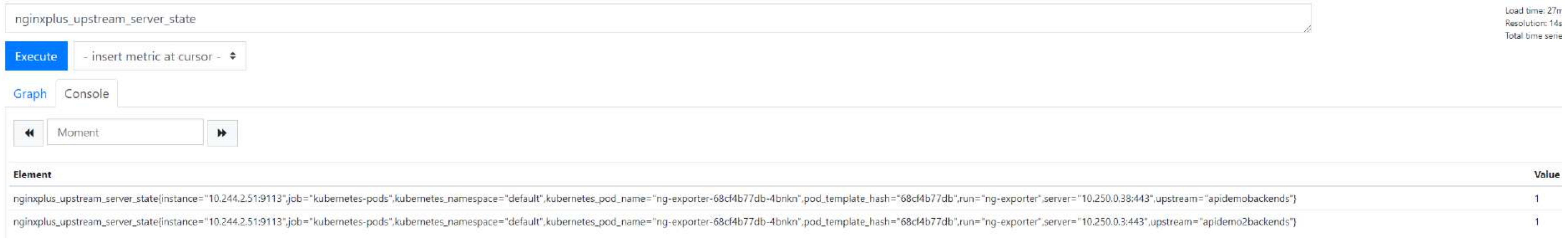


```
rate(nginxplus_upstream_server_requests[5m])
```

Load time: 13
Resolution: 1
Total time ser

```
✓ 🟦 {instance="10.244.2.51:9113",job="kubernetes-pods",kubernetes_namespace="default",kubernetes_pod_name="ng-exporter-68cf4b77db-4bnkn",pod_template_hash="68cf4b77db",run="ng-exporter",server="10.250.0.3:443",upstream="apidemo2backends"}
✓ 🟩 {instance="10.244.2.51:9113",job="kubernetes-pods",kubernetes_namespace="default",kubernetes_pod_name="ng-exporter-68cf4b77db-4bnkn",pod_template_hash="68cf4b77db",run="ng-exporter",server="10.250.0.38:443",upstream="apidemobackends"}
✓ 🟩 {instance="10.244.2.42:9113",job="kubernetes-pods",kubernetes_namespace="default",kubernetes_pod_name="ng-exporter-68cf4b77db-4bnkn",pod_template_hash="68cf4b77db",run="ng-exporter",server="10.250.0.3:443",upstream="apidemo2backends"}
✓ 🟥 {instance="10.244.2.42:9113",job="kubernetes-pods",kubernetes_namespace="default",kubernetes_pod_name="ng-exporter-68cf4b77db-4bnkn",pod_template_hash="68cf4b77db",run="ng-exporter",server="10.250.0.38:443",upstream="apidemobackends"}
```

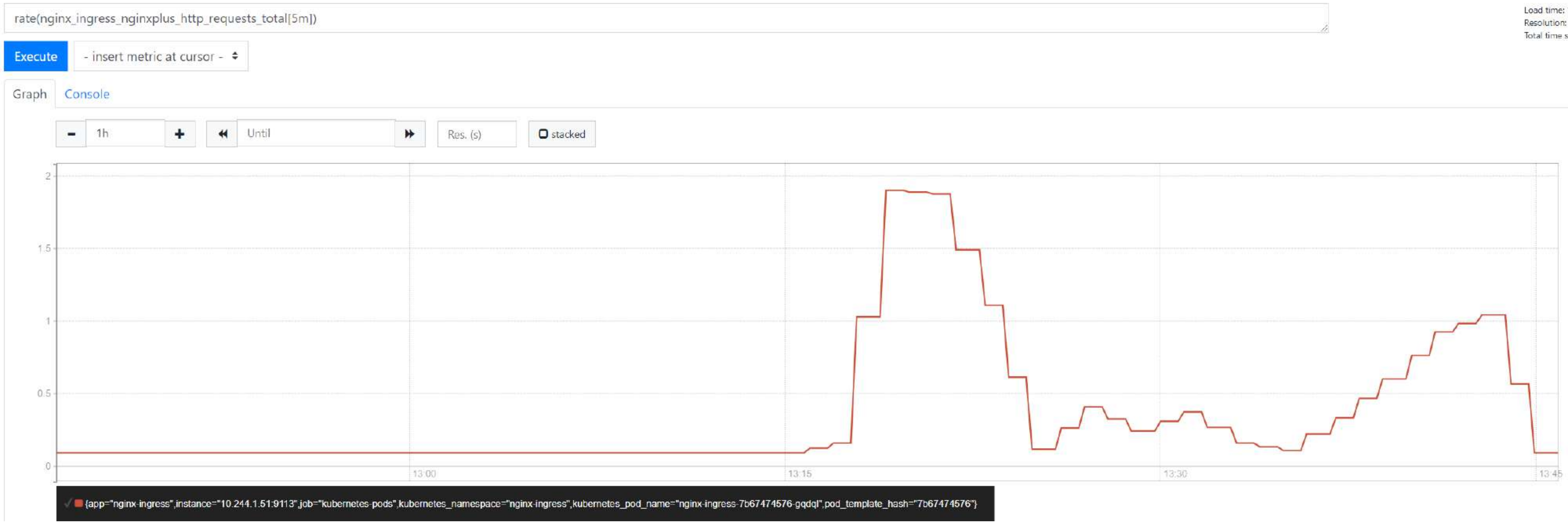
K8s-1的microgw Upsteam状态



```
nginxplus_upstream_server_state
```

Element	Value
nginxplus_upstream_server_state{instance="10.244.2.51:9113",job="kubernetes-pods",kubernetes_namespace="default",kubernetes_pod_name="ng-exporter-68cf4b77db-4bnkn",pod_template_hash="68cf4b77db",run="ng-exporter",server="10.250.0.38:443",upstream="apidemobackends"}	1
nginxplus_upstream_server_state{instance="10.244.2.51:9113",job="kubernetes-pods",kubernetes_namespace="default",kubernetes_pod_name="ng-exporter-68cf4b77db-4bnkn",pod_template_hash="68cf4b77db",run="ng-exporter",server="10.250.0.3:443",upstream="apidemo2backends"}	1

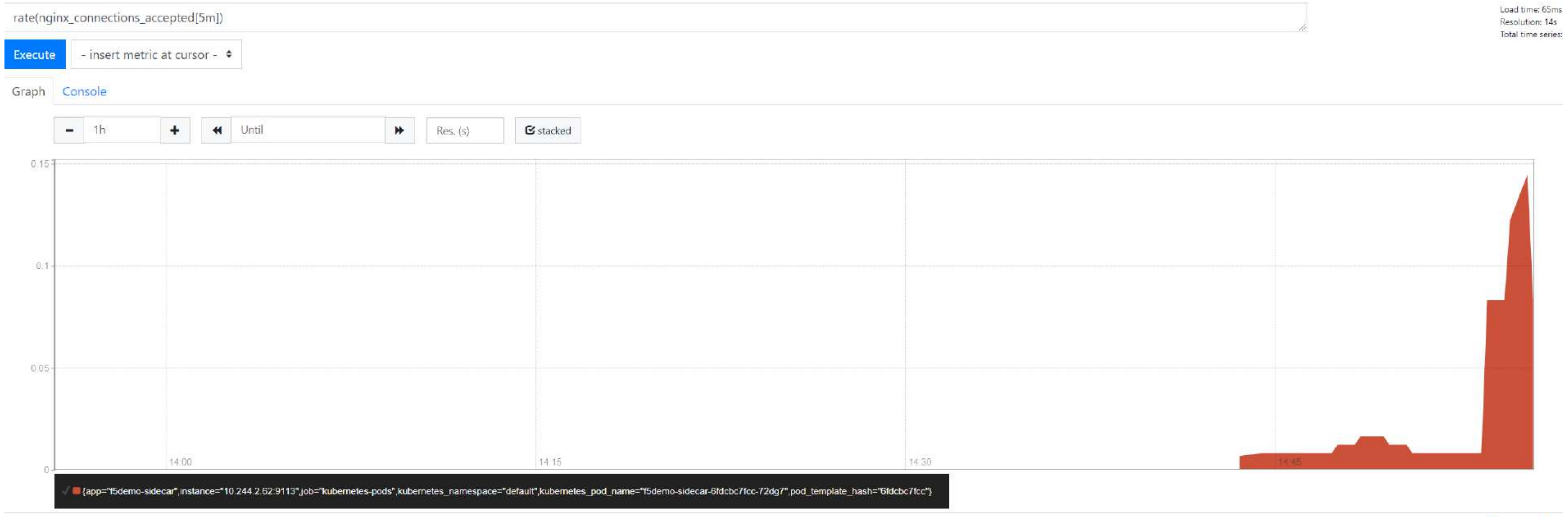
K8s-2 IC的total request



```
rate(nginx_ingress_nginxplus_http_requests_total[5m])
```

```
🟥 {app="nginx-ingress",instance="10.244.1.51:9113",job="kubernetes-pods",kubernetes_namespace="nginx-ingress",kubernetes_pod_name="nginx-ingress-7b67474576-gqdql",pod_template_hash="7b67474576"}
```

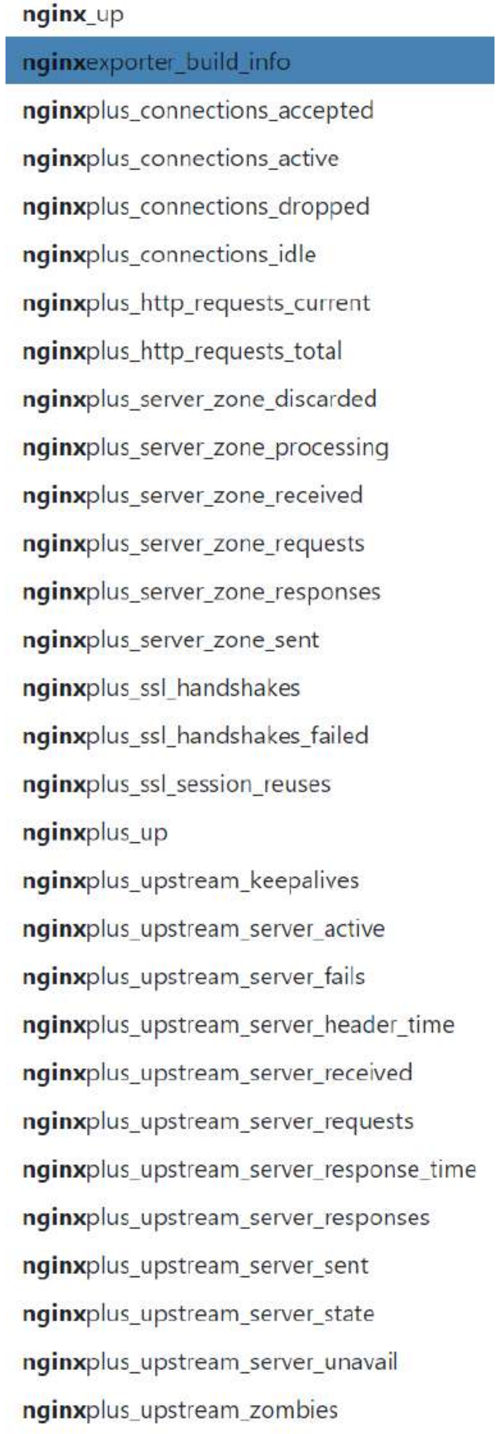
开源nginx 作为sidecar的指标



```
rate(nginx_connections_accepted[5m])
```

```
Load time: 65ms
Resolution: 14s
Total time series:
```

```
✓ 🟥 {app="f5demo-sidecar",instance="10.244.2.62:9113",job="kubernetes-pods",kubernetes_namespace="default",kubernetes_pod_name="f5demo-sidecar-6fdcbc7fcc-72dg7",pod_template_hash="6fdcbc7fcc"}
```

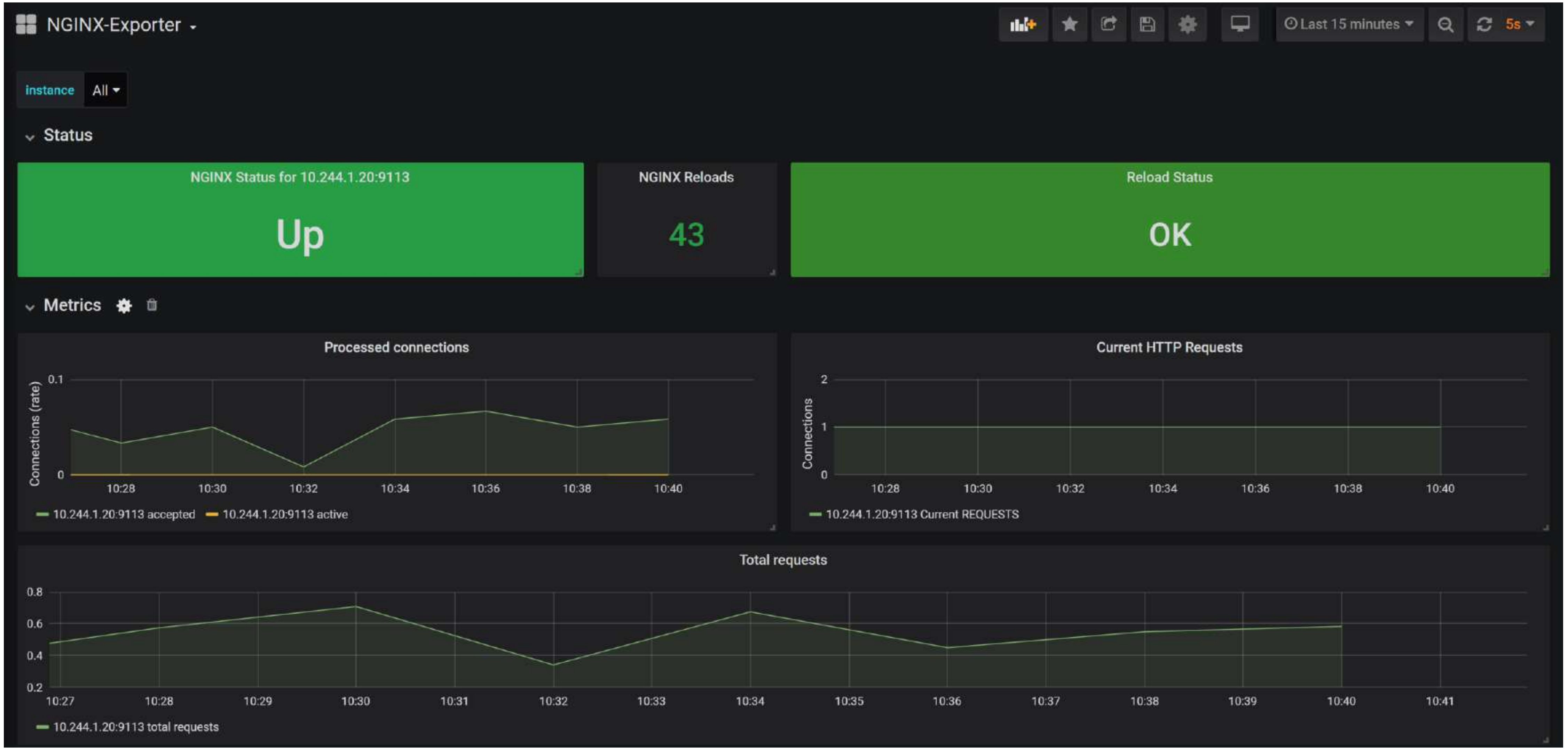


```
nginx_up
nginxexporter_build_info
nginxplus_connections_accepted
nginxplus_connections_active
nginxplus_connections_dropped
nginxplus_connections_idle
nginxplus_http_requests_current
nginxplus_http_requests_total
nginxplus_server_zone_discarded
nginxplus_server_zone_processing
nginxplus_server_zone_received
nginxplus_server_zone_requests
nginxplus_server_zone_responses
nginxplus_server_zone_sent
nginxplus_ssl_handshakes
nginxplus_ssl_handshakes_failed
nginxplus_ssl_session_reuses
nginxplus_up
nginxplus_upstream_keepalives
nginxplus_upstream_server_active
nginxplus_upstream_server_fails
nginxplus_upstream_server_header_time
nginxplus_upstream_server_received
nginxplus_upstream_server_requests
nginxplus_upstream_server_response_time
nginxplus_upstream_server_responses
nginxplus_upstream_server_sent
nginxplus_upstream_server_state
nginxplus_upstream_server_unavail
nginxplus_upstream_zombies
```

Exporter输出的部分指标

https://github.com/nginxinc/nginx-prometheus-exporter

开源NGINX指标的Grafana dashboard


NGINX-Exporter

instance All

Status

NGINX Status for 10.244.1.20:9113

Up

NGINX Reloads

43

Reload Status

OK

Metrics

Processed connections

— 10.244.1.20:9113 accepted — 10.244.1.20:9113 active

Current HTTP Requests

— 10.244.1.20:9113 Current REQUESTS

Total requests

— 10.244.1.20:9113 total requests

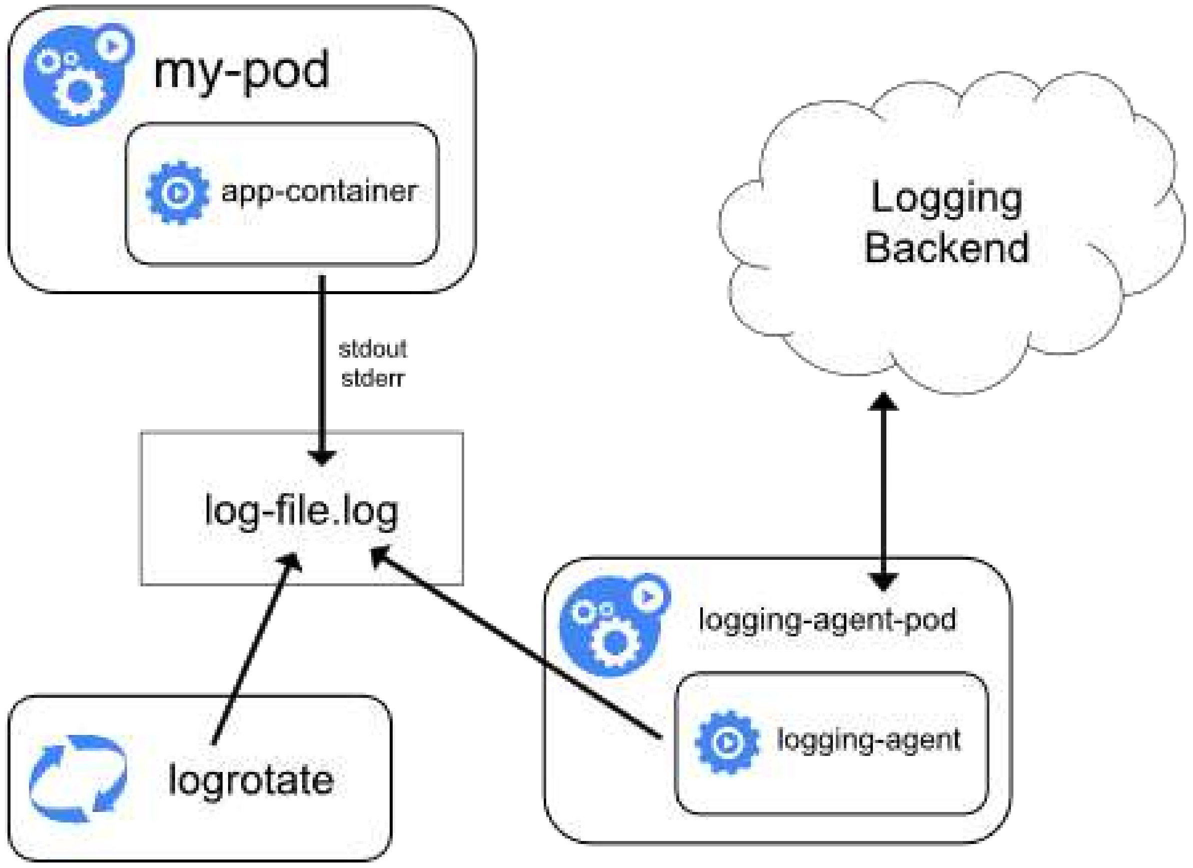
NGINX

结合EFK通过NGINX采集业务访问日志

一个问题

为什么要采集业务访问日志？？ 请抢答！

K8s下日志的采集方式-节点采集

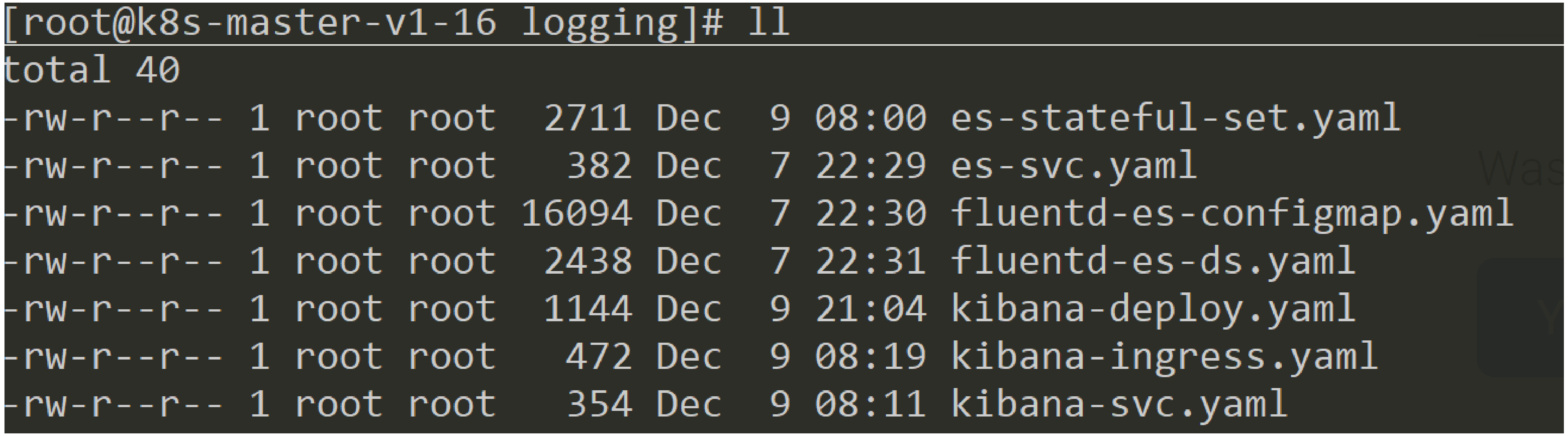


- K8s node节点上安装日志采集程序
- 默认情况下，dockers以json-file驱动来存储容器日志到宿主机目录:
- /var/lib/docker/containers/**containersID**/**.json
- 日志采集程序例如fluentd采集宿主机上文件
- 采集方案难度以及灵活性相对适中
- *应用日志需要输出到stdout/stderr

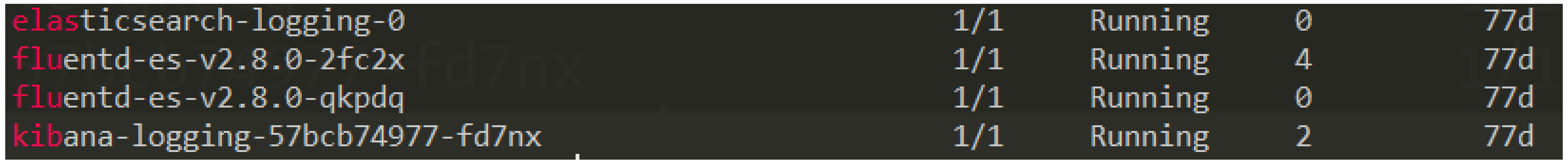
节点日志采集方式demo

1. 部署EFK套件，fluentd采集日志，elasticsearch负责存储，Kibana负责可视化
2. Kibana简单检索日志，并形成可视化
3. Fluentd 在node节点采集nginx的access log
4. 当NGINX作为容器化方式使用，如何使得日志发送到stdout/stderr

部署EFK



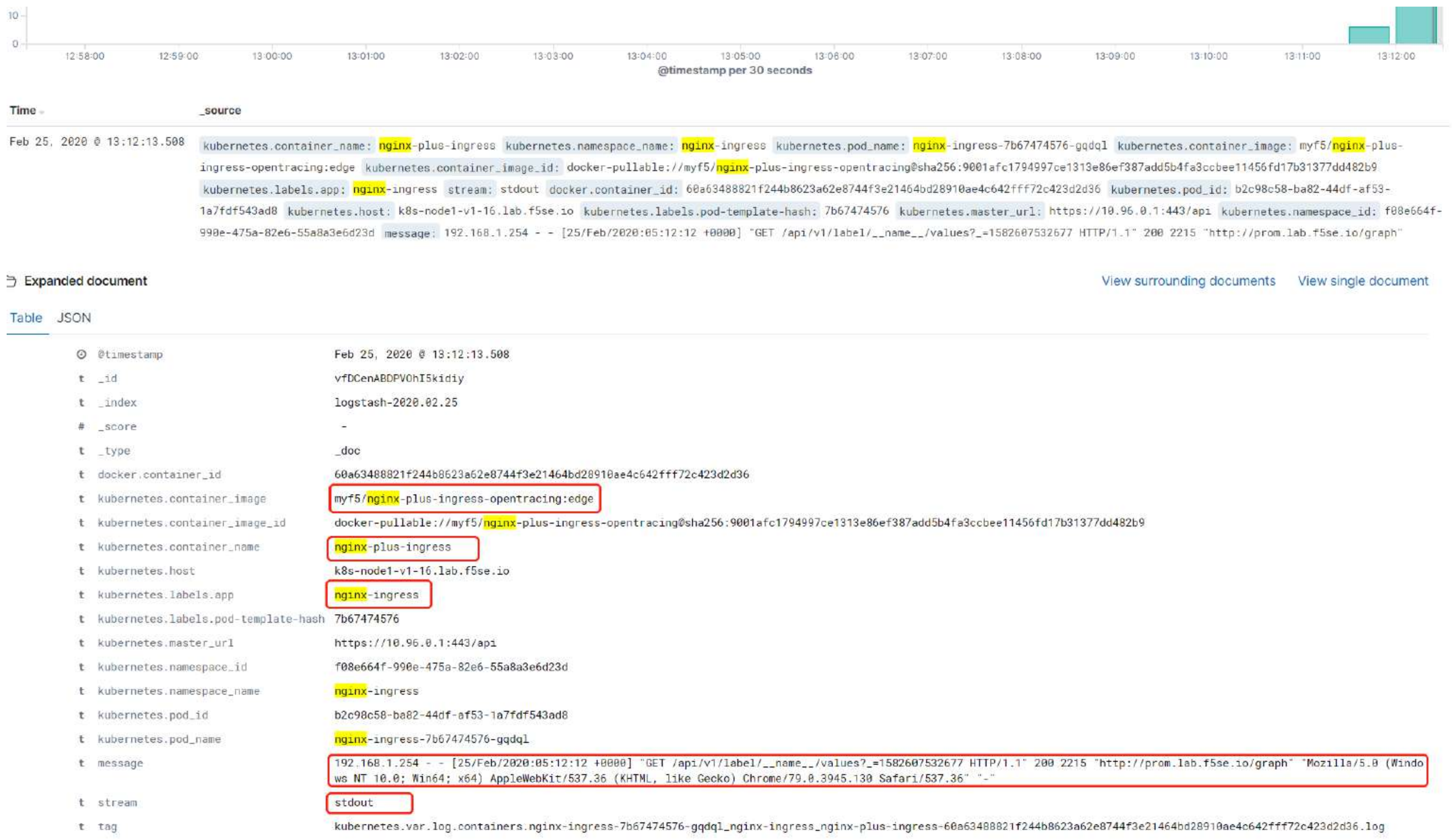
```
[root@k8s-master-v1-16 logging]# ll
total 40
-rw-r--r-- 1 root root  2711 Dec  9 08:00 es-stateful-set.yaml
-rw-r--r-- 1 root root   382 Dec  7 22:29 es-svc.yaml
-rw-r--r-- 1 root root 16094 Dec  7 22:30 fluentd-es-configmap.yaml
-rw-r--r-- 1 root root  2438 Dec  7 22:31 fluentd-es-ds.yaml
-rw-r--r-- 1 root root  1144 Dec  9 21:04 kibana-deploy.yaml
-rw-r--r-- 1 root root   472 Dec  9 08:19 kibana-ingress.yaml
-rw-r--r-- 1 root root   354 Dec  9 08:11 kibana-svc.yaml
```



```
elasticsearch-logging-0                    1/1     Running     0          77d
fluentd-es-v2.8.0-2fc2x                    1/1     Running     4          77d
fluentd-es-v2.8.0-qkpdq                    1/1     Running     0          77d
kibana-logging-57bcb74977-fd7nx            1/1     Running     2          77d
```

https://github.com/kubernetes/kubernetes/tree/master/cluster/addons/fluentd-elasticsearch

Kibana检索与展示



Time	_source
Feb 25, 2020 @ 13:12:13.508	kubernetes.container_name: nginx-plus-ingress kubernetes.namespace_name: nginx-ingress kubernetes.pod_name: nginx-ingress-7b67474576-gqdql kubernetes.container_image: myf5/nginx-plus-ingress-opentracing:edge kubernetes.container_image_id: docker-pullable://myf5/nginx-plus-ingress-opentracing@sha256:9001afc1794997ce1313e86ef387add5b4fa3ccbee11456fd17b31377dd482b9 kubernetes.labels.app: nginx-ingress stream: stdout docker.container_id: 60a63488821f244b8623a62e8744f3e21464bd28910ae4c642fff72c423d2d36 kubernetes.pod_id: b2c98c58-ba82-44df-af53-1a7fdf543ad8 kubernetes.host: k8s-node1-v1-16.lab.f5se.io kubernetes.labels.pod-template-hash: 7b67474576 kubernetes.master_url: https://10.96.0.1:443/api kubernetes.namespace_id: f08e664f-990e-475a-82e6-55a8a3e6d23d message: 192.168.1.254 - - [25/Feb/2020:05:12:12 +0000] "GET /api/v1/label/__name__/values?_=1582607532677 HTTP/1.1" 200 2215 "http://prom.lab.f5se.io/graph"

Expanded document

View surrounding documents View single document

Table JSON

	@timestamp	Feb 25, 2020 @ 13:12:13.508
t	_id	vfDCenABDPVOhI5kidiy
t	_index	logstash-2020.02.25
#	_score	-
t	_type	_doc
t	docker.container_id	60a63488821f244b8623a62e8744f3e21464bd28910ae4c642fff72c423d2d36
t	kubernetes.container_image	myf5/nginx-plus-ingress-opentracing:edge
t	kubernetes.container_image_id	docker-pullable://myf5/nginx-plus-ingress-opentracing@sha256:9001afc1794997ce1313e86ef387add5b4fa3ccbee11456fd17b31377dd482b9
t	kubernetes.container_name	nginx-plus-ingress
t	kubernetes.host	k8s-node1-v1-16.lab.f5se.io
t	kubernetes.labels.app	nginx-ingress
t	kubernetes.labels.pod-template-hash	7b67474576
t	kubernetes.master_url	https://10.96.0.1:443/api
t	kubernetes.namespace_id	f08e664f-990e-475a-82e6-55a8a3e6d23d
t	kubernetes.namespace_name	nginx-ingress
t	kubernetes.pod_id	b2c98c58-ba82-44df-af53-1a7fdf543ad8
t	kubernetes.pod_name	nginx-ingress-7b67474576-gqdql
t	message	192.168.1.254 - - [25/Feb/2020:05:12:12 +0000] "GET /api/v1/label/__name__/values?_=1582607532677 HTTP/1.1" 200 2215 "http://prom.lab.f5se.io/graph" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/79.0.3945.130 Safari/537.36" "-"
t	stream	stdout
t	tag	kubernetes.var.log.containers.nginx-ingress-7b67474576-gqdql_nginx-ingress_nginx-plus-ingress-60a63488821f244b8623a62e8744f3e21464bd28910ae4c642fff72c423d2d36.log

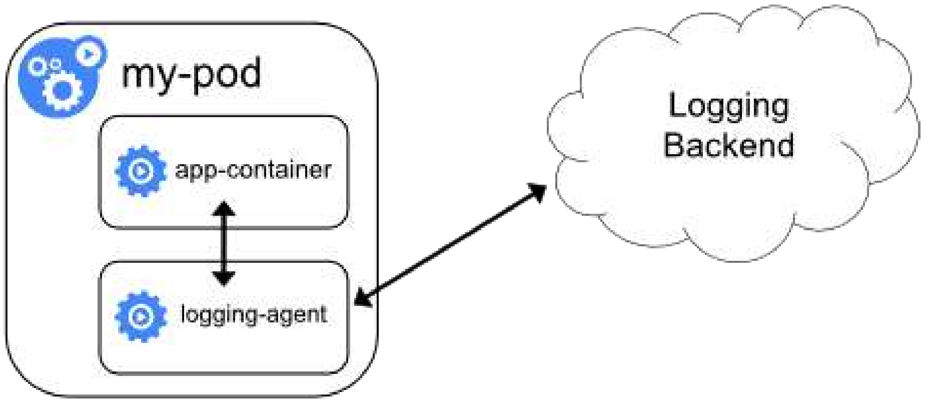
一个问题

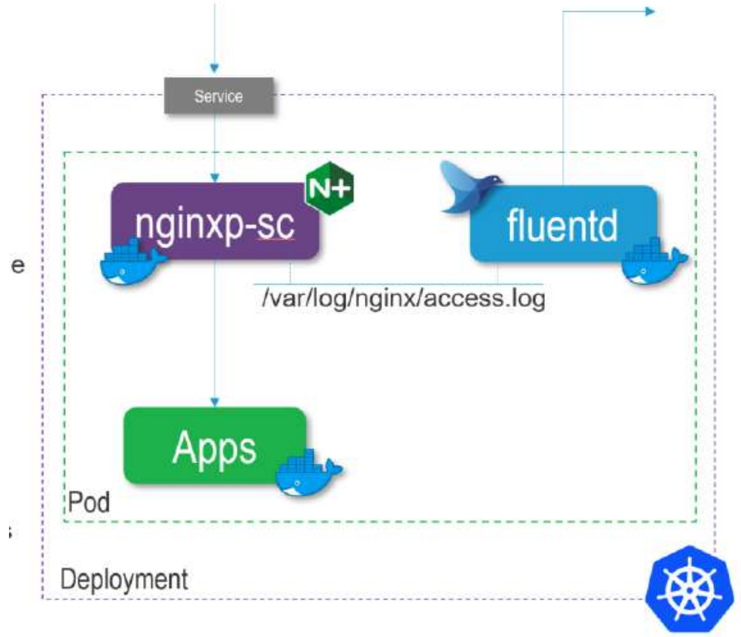
怎么让nginx容器的access log日志输出到stdout?

制作nginx容器镜像时候，在dockerfile中：

```
RUN ln -sf /proc/1/fd/1 /var/log/nginx/access.log \
    && ln -sf /proc/1/fd/1 /var/log/nginx/stream-access.log \
    && ln -sf /proc/1/fd/2 /var/log/nginx/error.log
```

K8s下日志的采集方式-sidecar采集

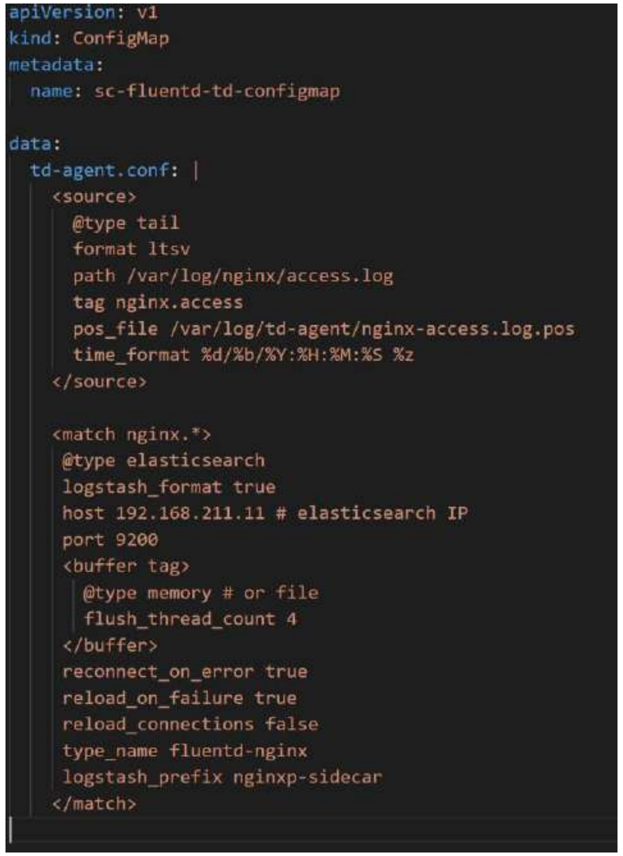




- 当业务日志只能输出到容器内的文件时采用
- 在pod中运行一个专门的日志采集容器
- 两个容器共享相同的文件
- 采集配置自定义灵活性较高，但是技术难度及成本相对较大

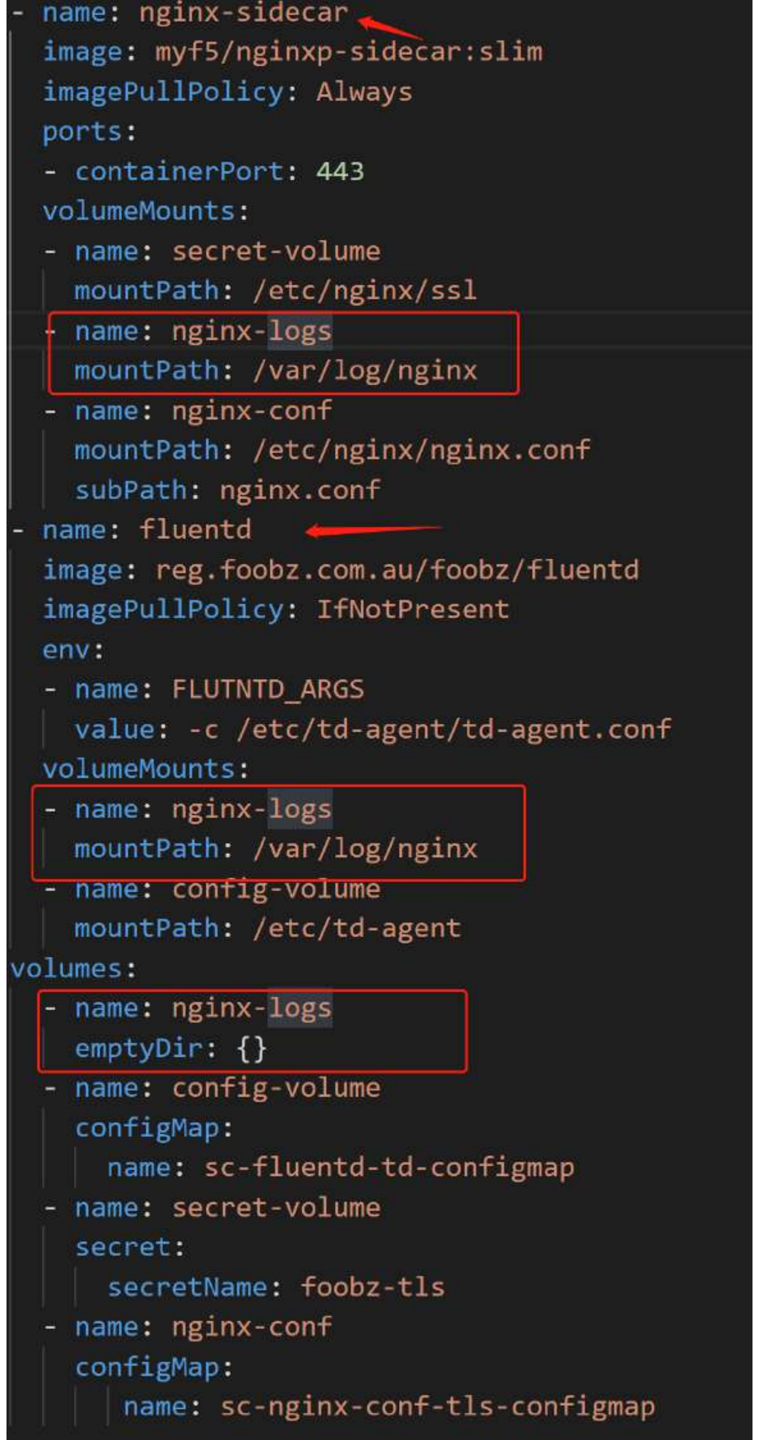
Sidecar 采集方式

NGINX 和fluentd 共用同一个本地文件mount



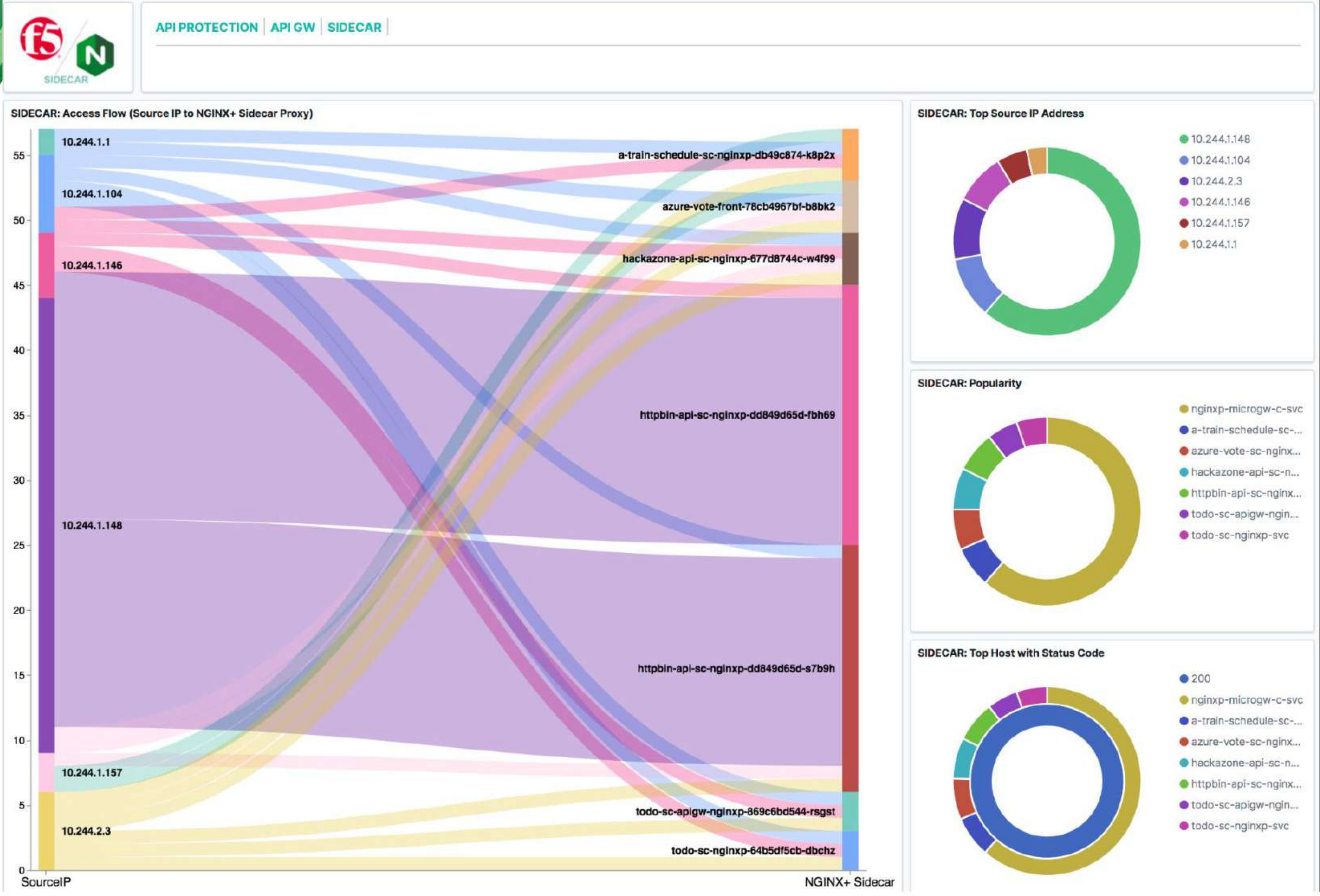
```
apiVersion: v1
kind: ConfigMap
metadata:
  name: sc-fluentd-td-configmap

data:
  td-agent.conf: |
    <source>
      @type tail
      format ltsv
      path /var/log/nginx/access.log
      tag nginx.access
      pos_file /var/log/td-agent/nginx-access.log.pos
      time_format %d/%b/%Y:%H:%M:%S %z
    </source>

    <match nginx.*>
     @type elasticsearch
     logstash_format true
     host 192.168.211.11 # elasticsearch IP
     port 9200
     <buffer tag>
       @type memory # or file
       flush_thread_count 4
     </buffer>
     reconnect_on_error true
     reload_on_failure true
     reload_connections false
     type_name fluentd-nginx
     logstash_prefix nginxp-sidecar
    </match>
```

```
log_format ltsv 'time:$time_local\t'
    'status:$status\t'
    'request_time:$request_time\t'
    'upstream_addr:$upstream_addr\t'
    'upstream_response_time:$upstream_response_time\t'
    'upstream_cache_status:$upstream_cache_status\t'
    'body_bytes_sent:$body_bytes_sent\t'
    'remote_addr:$remote_addr\t'
    'host:$host\t'
    'hostname:$hostname\t'
    'request_method:$request_method\t'
    'request_uri:$request_uri\t'
    'protocol:$server_protocol\t'
    'x-forwarded-for: $proxy_add_x_forwarded_for\t'
    'http_referer:$http_referer\t'
    'http_user_agent:$http_user_agent';
```



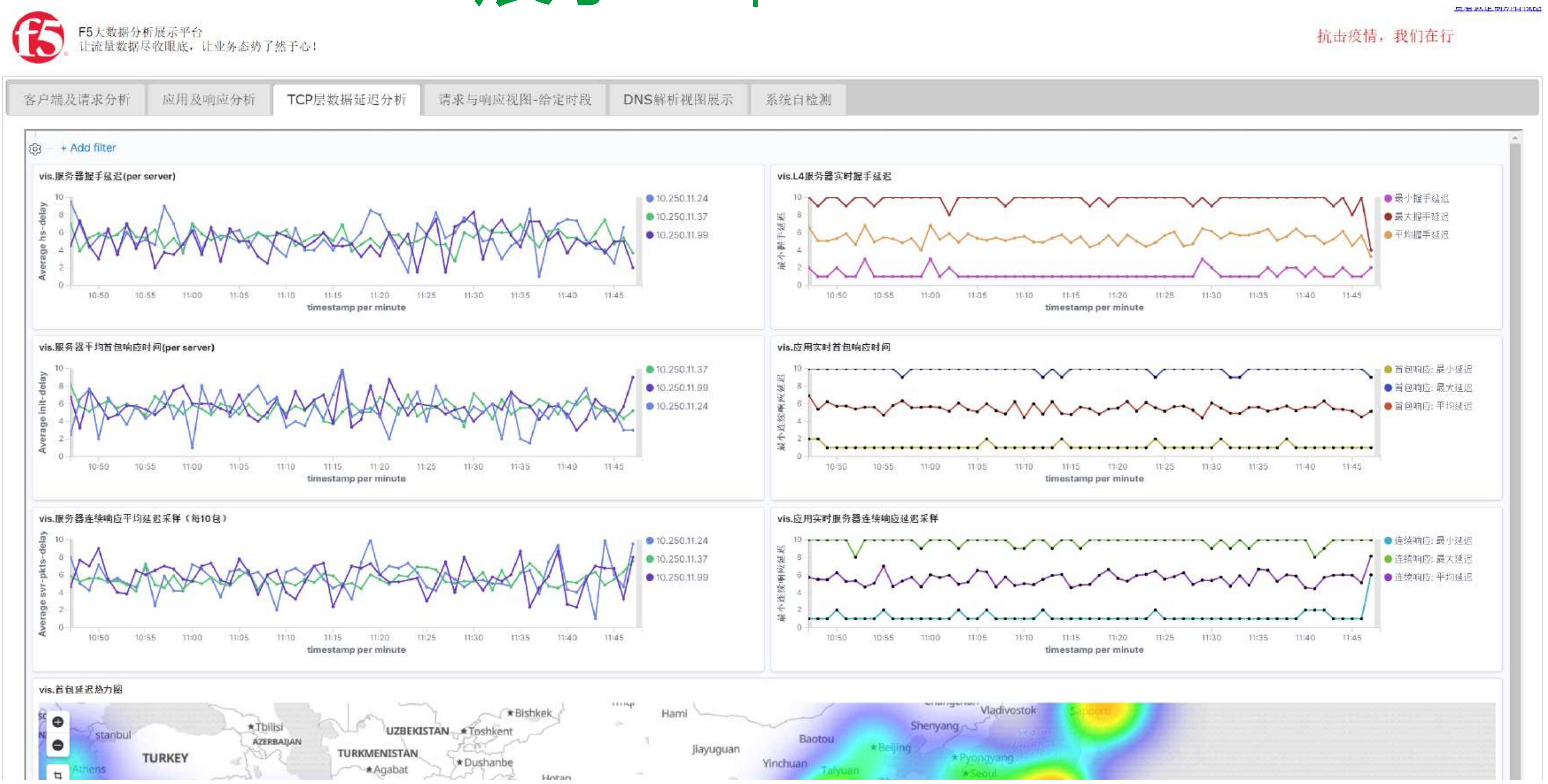
```
- name: nginx-sidecar
  image: myf5/nginxp-sidecar:slim
  imagePullPolicy: Always
  ports:
  - containerPort: 443
  volumeMounts:
  - name: secret-volume
    mountPath: /etc/nginx/ssl
  - name: nginx-logs
    mountPath: /var/log/nginx
  - name: nginx-conf
    mountPath: /etc/nginx/nginx.conf
    subPath: nginx.conf
- name: fluentd
  image: reg.foobz.com.au/foobz/fluentd
  imagePullPolicy: IfNotPresent
  env:
  - name: FLUTNTD_ARGS
    value: -c /etc/td-agent/td-agent.conf
  volumeMounts:
  - name: nginx-logs
    mountPath: /var/log/nginx
  - name: config-volume
    mountPath: /etc/td-agent
volumes:
  - name: nginx-logs
    emptyDir: {}
  - name: config-volume
    configMap:
      name: sc-fluentd-td-configmap
  - name: secret-volume
    secret:
      secretName: foobz-tls
  - name: nginx-conf
    configMap:
      name: sc-nginx-conf-tls-configmap
```

Fluentd配置对接es，并做日志格式化


SIDECAR
API PROTECTION | API GW | SIDECAR
SIDECAR: Access Flow (Source IP to NGINX+ Sidecar Proxy)
10.244.1.1
10.244.1.104
10.244.1.146
10.244.1.148
10.244.1.157
10.244.2.3
a-train-schedule-sc-nginxp-db49c874-k8p2x
azure-vote-front-78cb4967bf-b8bk2
hackazone-api-sc-nginxp-677d8744c-w4f99
httpbin-api-sc-nginxp-dd849d65d-fbh69
httpbin-api-sc-nginxp-dd849d65d-s7b9h
todo-sc-apigw-nginxp-869c6bd544-rsgst
todo-sc-nginxp-64b5df5cb-dbchz
0
5
10
15
20
25
30
35
40
45
50
55
SourceIP
NGINX+ Sidecar
SIDECAR: Top Source IP Address
● 10.244.1.148
● 10.244.1.104
● 10.244.2.3
● 10.244.1.146
● 10.244.1.157
● 10.244.1.1
SIDECAR: Popularity
● nginxp-microgw-c-svc
● a-train-schedule-sc-...
● azure-vote-sc-nginx...
● hackazone-api-sc-n...
● httpbin-api-sc-nginx...
● todo-sc-apigw-ngin...
● todo-sc-nginxp-svc
SIDECAR: Top Host with Status Code
● 200
● nginxp-microgw-c-svc
● a-train-schedule-sc-...
● azure-vote-sc-nginx...
● hackazone-api-sc-n...
● httpbin-api-sc-nginx...
● todo-sc-apigw-ngin...
● todo-sc-nginxp-svc

NGINX

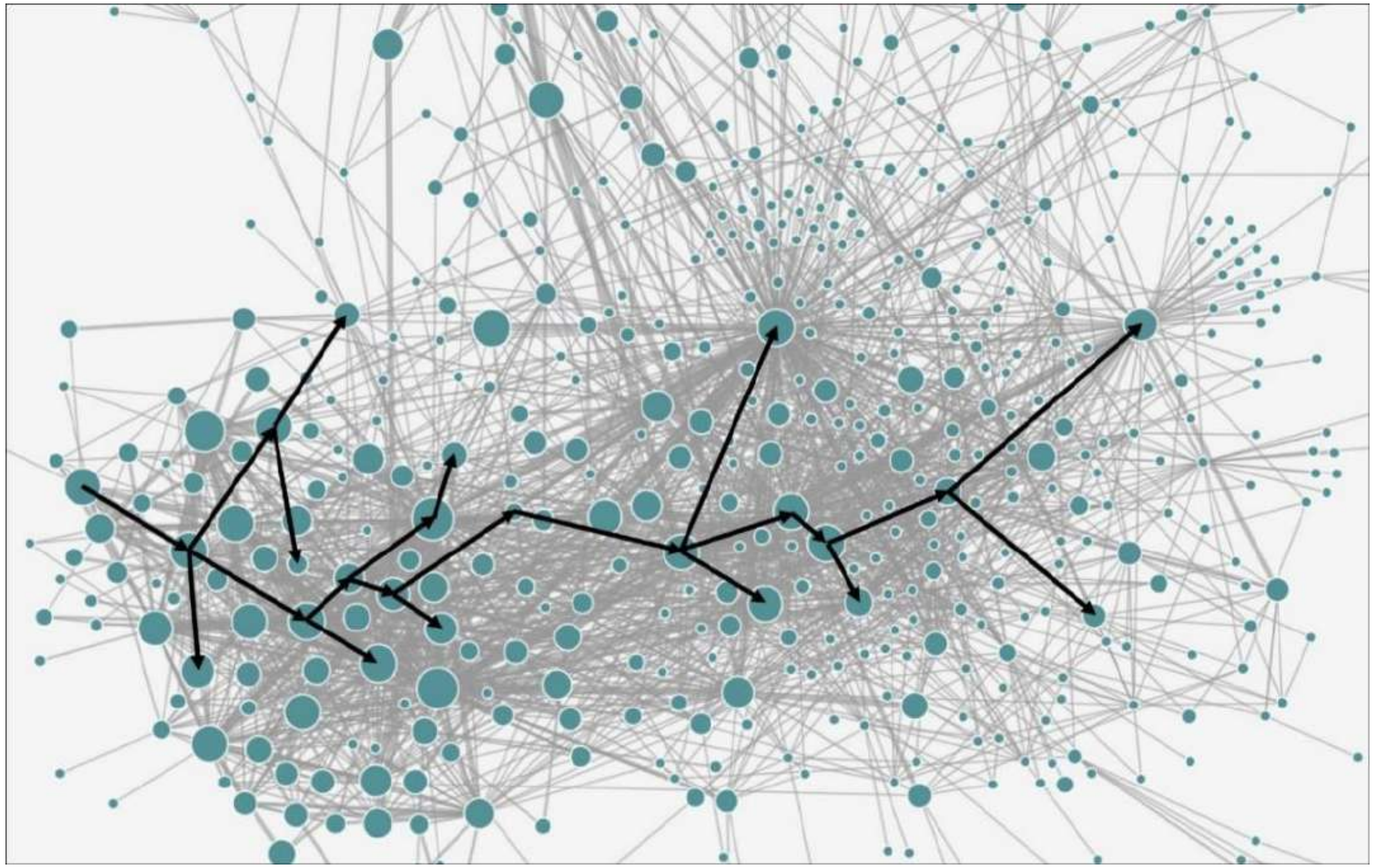
Kibana展示 http://bde.f5se.io



https://github.com/zongzw/bde-over-bigip/releases

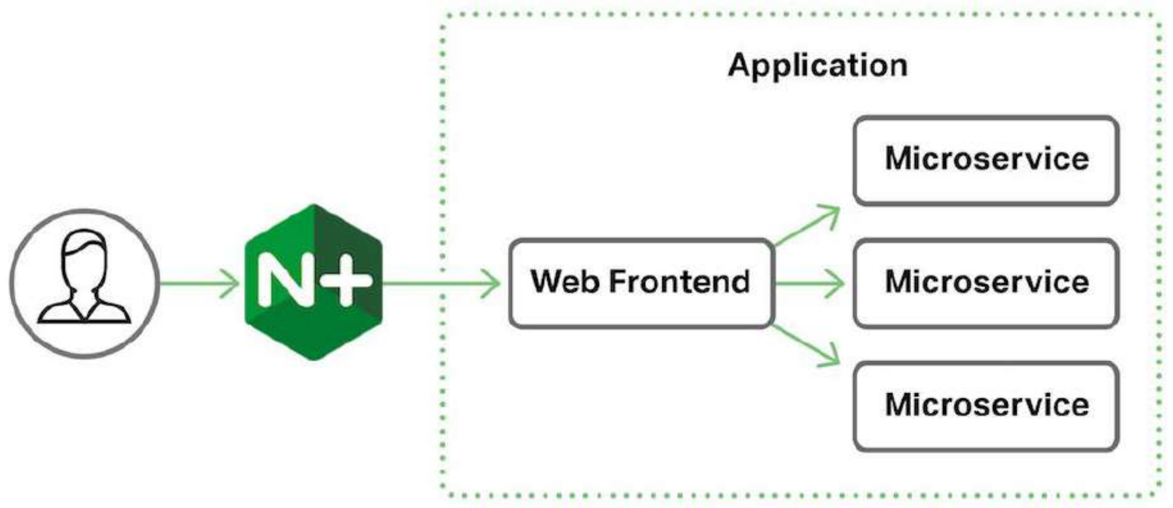
NGINX与Jaeger实现应用访问路径跟踪可视化

为什么要搞tracing



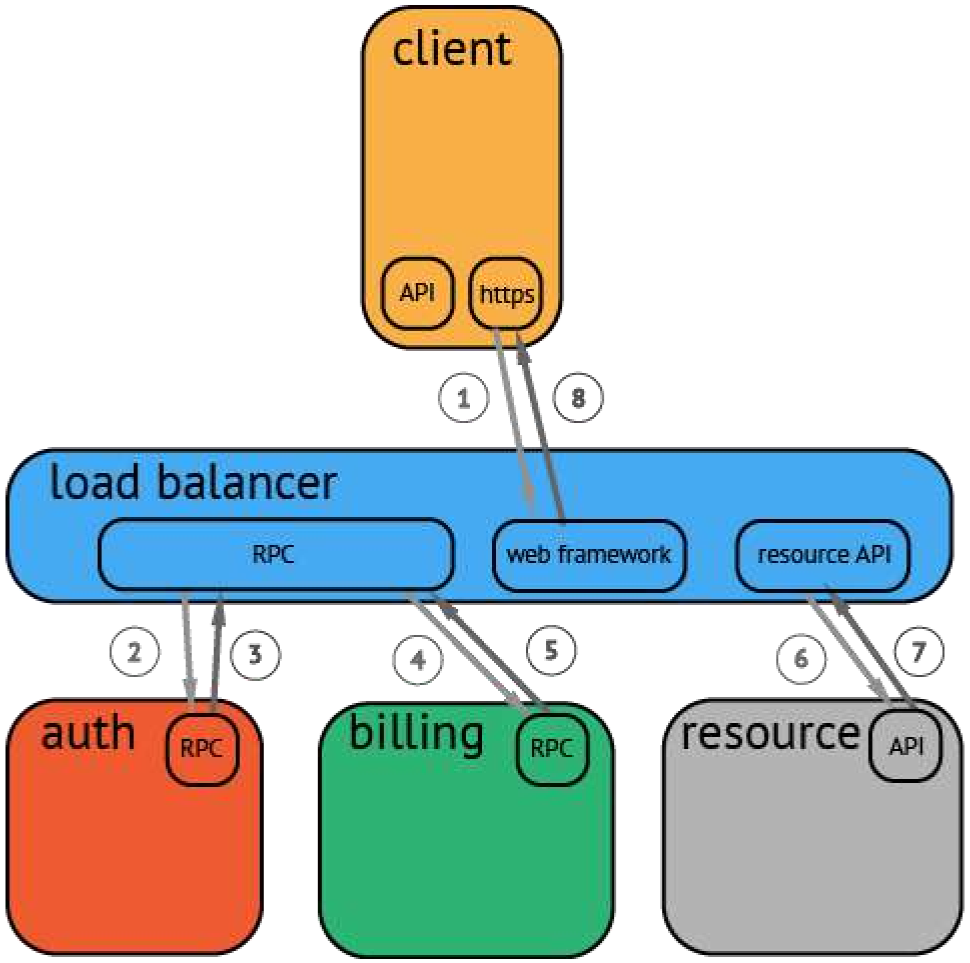
大家用了哪些方法去做访问路径跟踪?

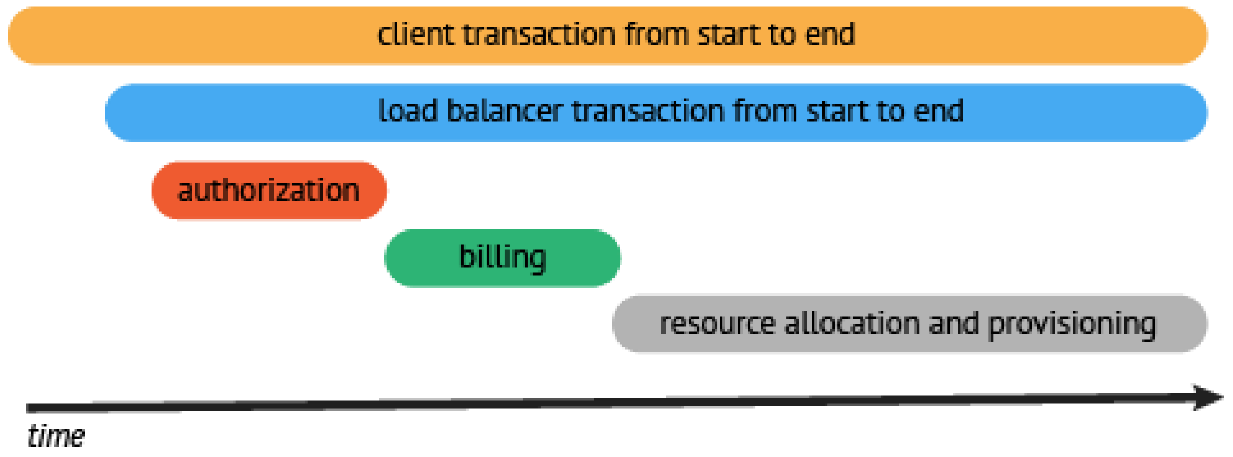
还记得$request_id吗



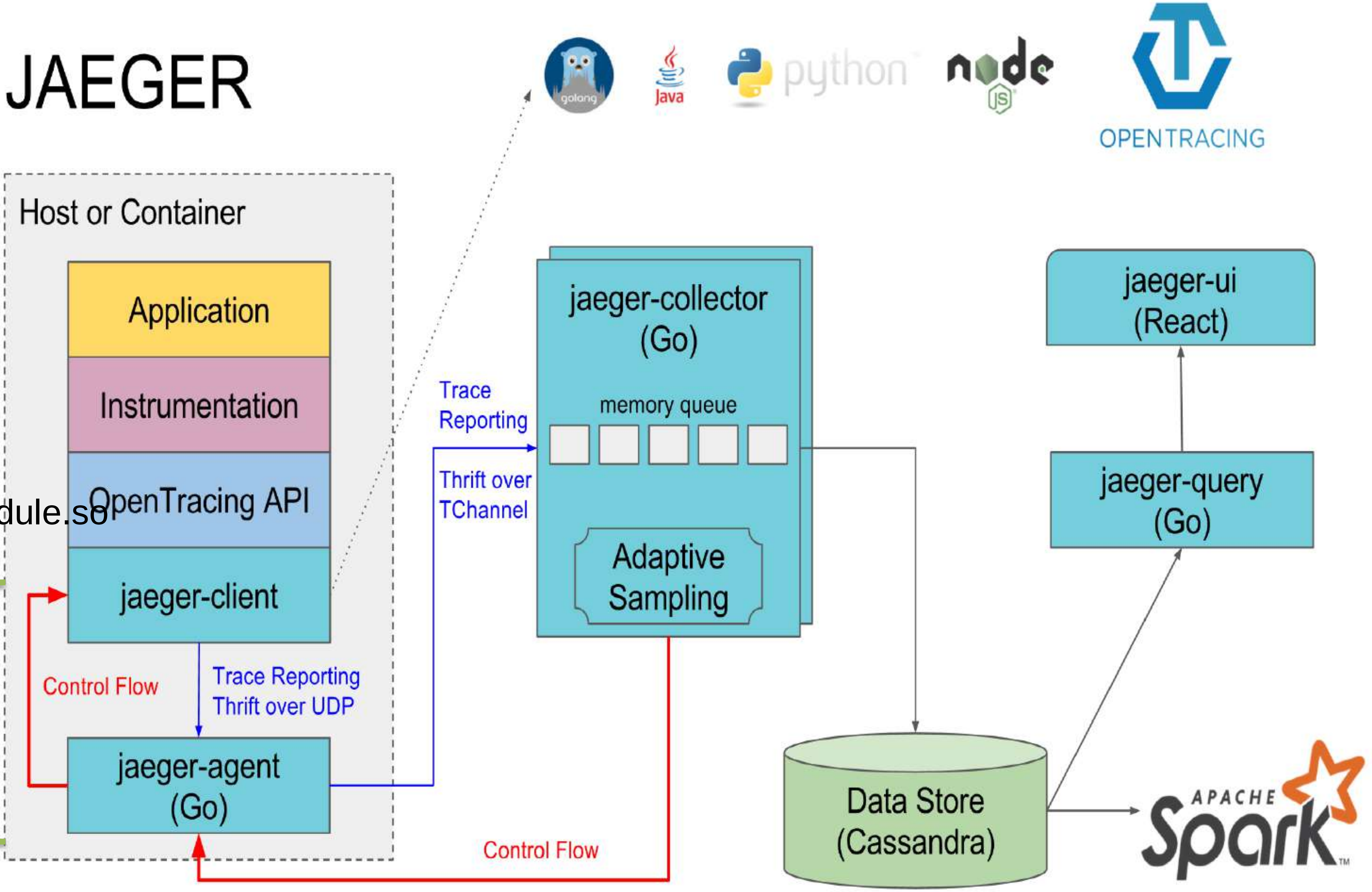
```
log_format trace '$remote_addr - $remote_user [$time_local] "$request" '
                 '$status $body_bytes_sent "$http_referer" "$http_user_agent" '
                 '"$http_x_forwarded_for" $request_id';

upstream app_server {
    server 10.0.0.1;
}

server {
    listen 80;
    add_header X-Request-ID $request_id; # Return to client
    location / {
        proxy_pass http://app_server;
        proxy_set_header X-Request-ID $request_id;        # Pass to app server
        access_log /var/log/nginx/access_trace.log trace; # Log $request_id
    }
}
```

Opentracing



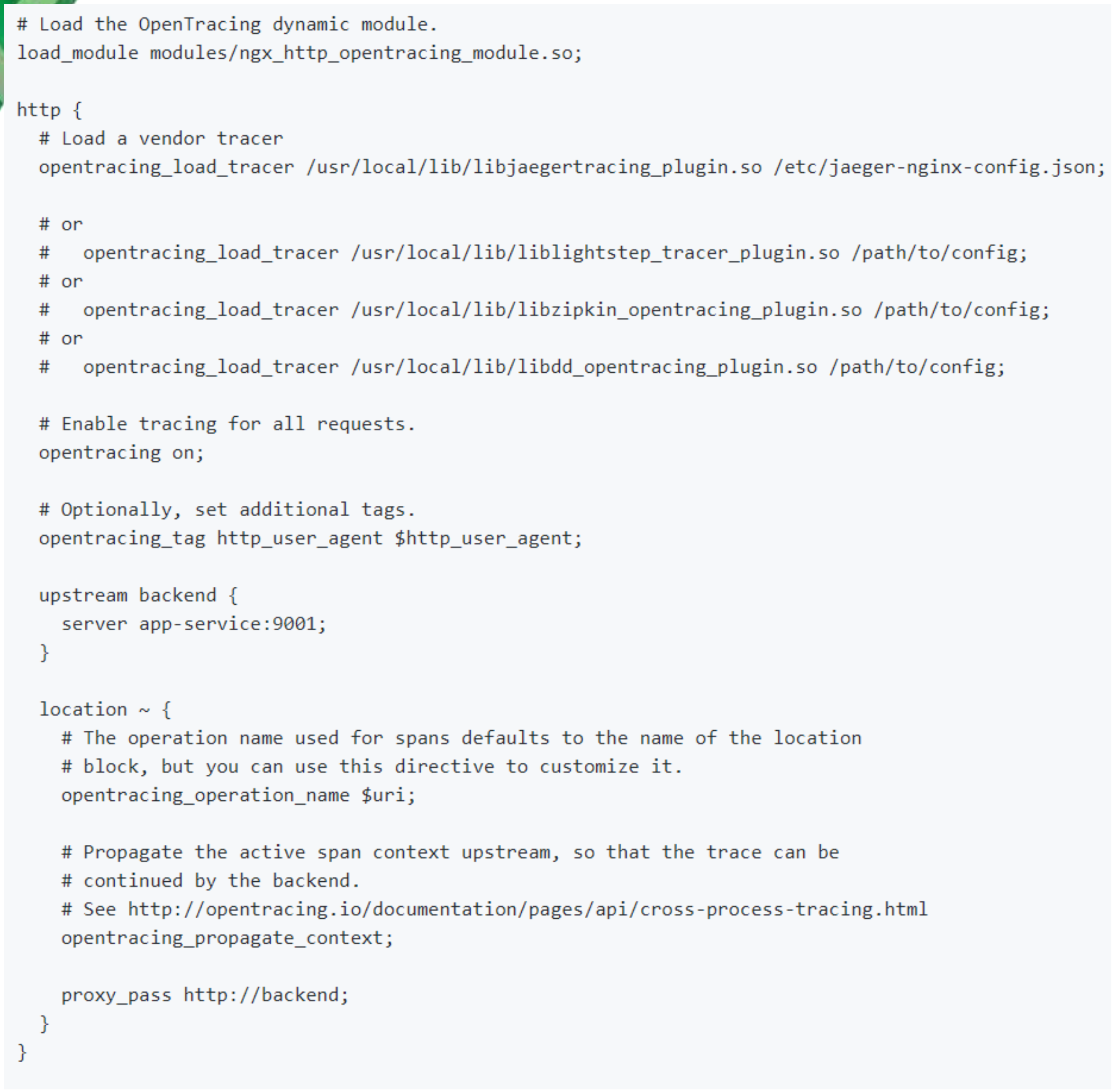

client transaction from start to end
load balancer transaction from start to end
authorization
billing
resource allocation and provisioning
time

JAEGER

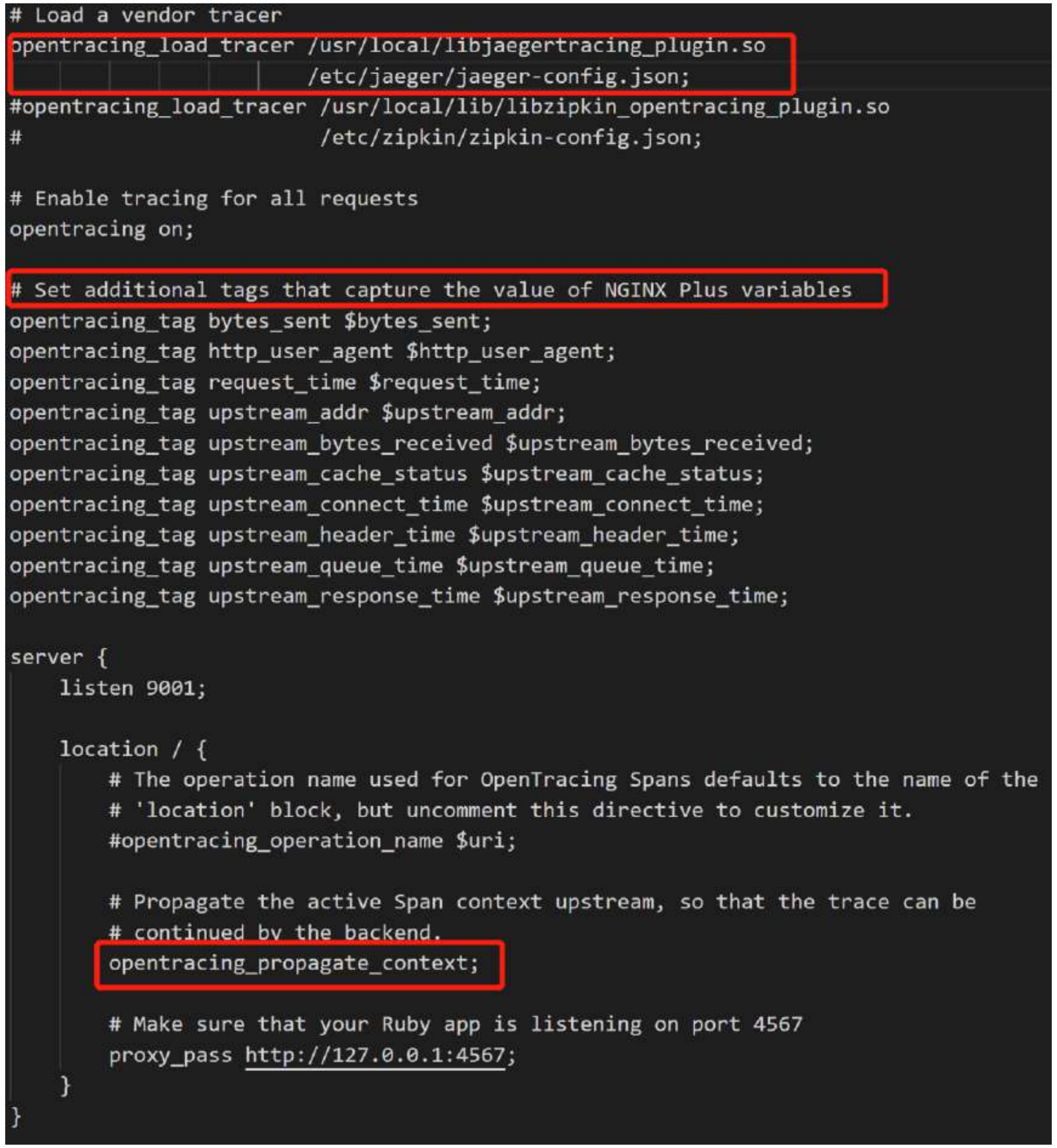


ngx_http_opentracing_module.so

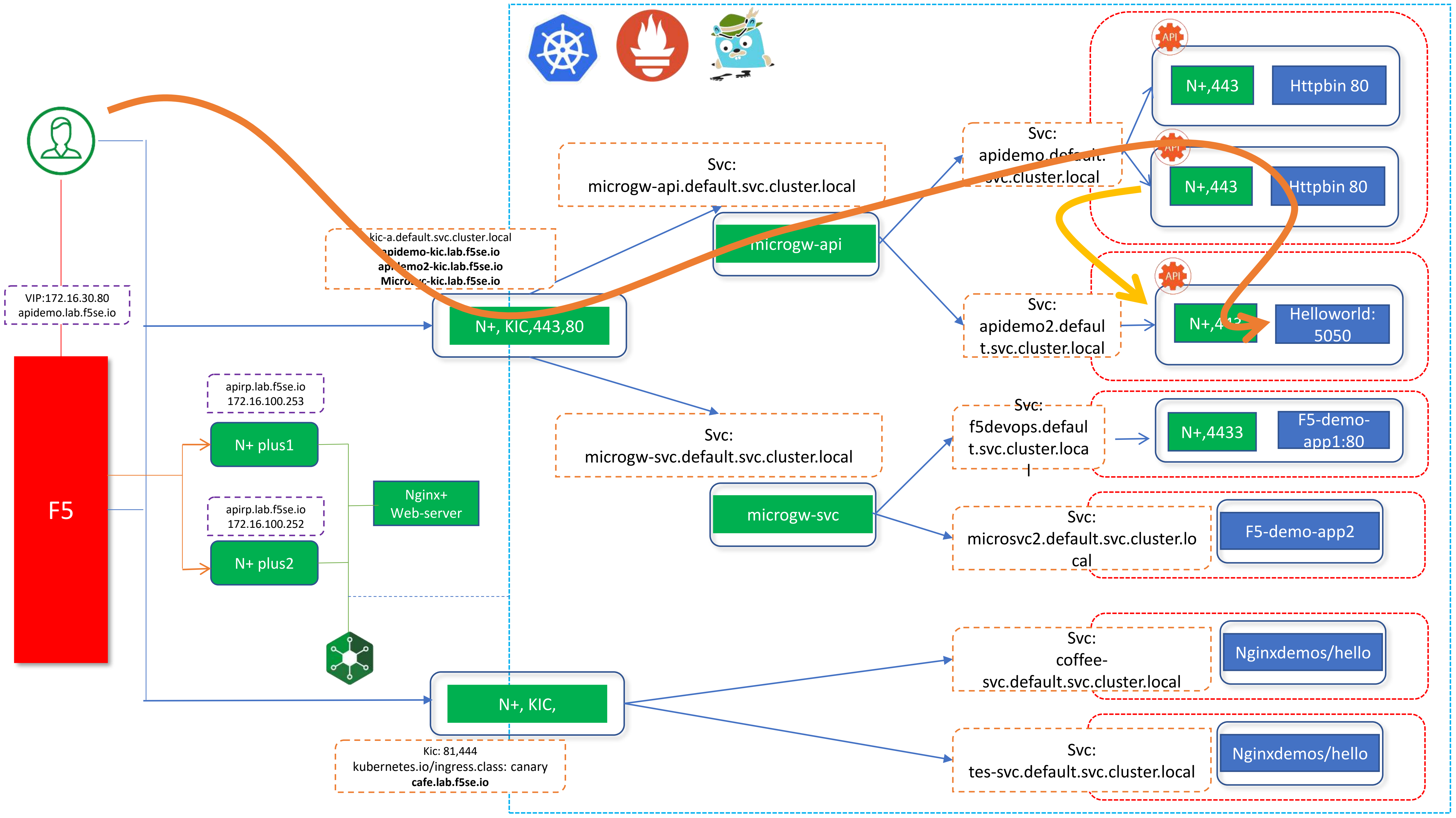
Jaeger plugin
vendor tracer



```
# Load the OpenTracing dynamic module.
load_module modules/ngx_http_opentracing_module.so;

http {
  # Load a vendor tracer
  opentracing_load_tracer /usr/local/lib/libjaegertracing_plugin.so /etc/jaeger-nginx-config.json;

  # or
  #   opentracing_load_tracer /usr/local/lib/liblightstep_tracer_plugin.so /path/to/config;
  # or
  #   opentracing_load_tracer /usr/local/lib/libzipkin_opentracing_plugin.so /path/to/config;
  # or
  #   opentracing_load_tracer /usr/local/lib/libdd_opentracing_plugin.so /path/to/config;

  # Enable tracing for all requests.
  opentracing on;

  # Optionally, set additional tags.
  opentracing_tag http_user_agent $http_user_agent;

  upstream backend {
    server app-service:9001;
  }

  location ~ {
    # The operation name used for spans defaults to the name of the location
    # block, but you can use this directive to customize it.
    opentracing_operation_name $uri;

    # Propagate the active span context upstream, so that the trace can be
    # continued by the backend.
    # See http://opentracing.io/documentation/pages/api/cross-process-tracing.html
    opentracing_propagate_context;

    proxy_pass http://backend;
  }
}
```



```
# Load a vendor tracer
opentracing_load_tracer /usr/local/libjaegertracing_plugin.so
                        /etc/jaeger/jaeger-config.json;
#opentracing_load_tracer /usr/local/lib/libzipkin_opentracing_plugin.so
#                        /etc/zipkin/zipkin-config.json;

# Enable tracing for all requests
opentracing on;

# Set additional tags that capture the value of NGINX Plus variables
opentracing_tag bytes_sent $bytes_sent;
opentracing_tag http_user_agent $http_user_agent;
opentracing_tag request_time $request_time;
opentracing_tag upstream_addr $upstream_addr;
opentracing_tag upstream_bytes_received $upstream_bytes_received;
opentracing_tag upstream_cache_status $upstream_cache_status;
opentracing_tag upstream_connect_time $upstream_connect_time;
opentracing_tag upstream_header_time $upstream_header_time;
opentracing_tag upstream_queue_time $upstream_queue_time;
opentracing_tag upstream_response_time $upstream_response_time;

server {
    listen 9001;

    location / {
        # The operation name used for OpenTracing Spans defaults to the name of the
        # 'location' block, but uncomment this directive to customize it.
        #opentracing_operation_name $uri;

        # Propagate the active Span context upstream, so that the trace can be
        # continued by the backend.
        opentracing_propagate_context;

        # Make sure that your Ruby app is listening on port 4567
        proxy_pass http://127.0.0.1:4567;
    }
}
```


Httpbin 80
N+,443
Httpbin 80
N+,443
Svc:
apidemo.default.svc.cluster.local
Svc:
microgw-api.default.svc.cluster.local
microgw-api
kic-a.default.svc.cluster.local
apidemo-kic.lab.f5se.io
apidemo2-kic.lab.f5se.io
Microsvc-kic.lab.f5se.io
VIP:172.16.30.80
apidemo.lab.f5se.io
N+, KIC,443,80
Svc:
apidemo2.default.svc.cluster.local
N+,443
Helloworld: 5050
apirp.lab.f5se.io
172.16.100.253
N+ plus1
Svc:
f5devops.default.svc.cluster.local
N+,4433
F5-demo-app1:80
Svc:
microgw-svc.default.svc.cluster.local
F5
Nginx+ Web-server
microgw-svc
apirp.lab.f5se.io
172.16.100.252
N+ plus2
Svc:
microsvc2.default.svc.cluster.local
F5-demo-app2
Svc:
coffee-svc.default.svc.cluster.local
Nginxdemos/hello
N+, KIC,
Kic: 81,444
kubernetes.io/ingress.class: canary
cafe.lab.f5se.io
Svc:
tes-svc.default.svc.cluster.local
Nginxdemos/hello

```
kubectl get cm -n nginx-ingress  nginx-config -o yaml
```

观察jaeger配置

模拟一个问题

kubectl get pod,svc -o wide

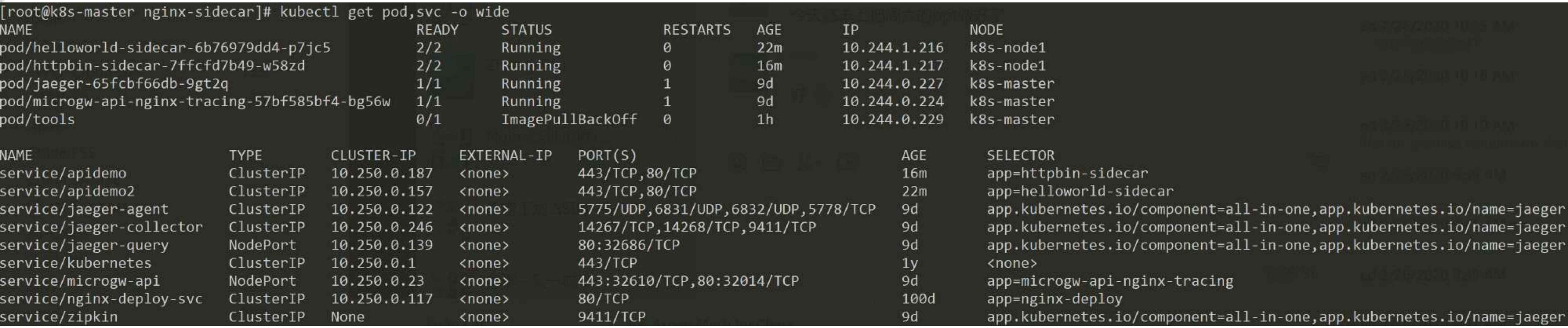
访问 https://apidemo-kic.lab.f5se.io/gotohelloworld/d
得到正常返回

删除并重建 hello-world service:
kubectl delete -f hello-world-sidecar-svc-deploy.yaml
kubectl create -f hello-world-sidecar-svc-deploy.yaml

再次访问:
访问 https://apidemo-kic.lab.f5se.io/gotohelloworld/d

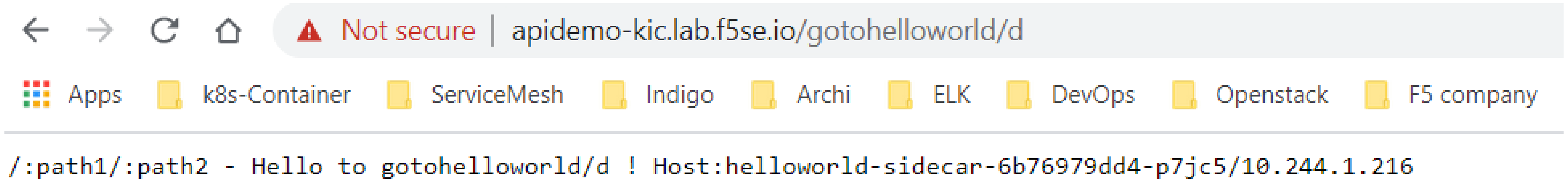
是否可以访问？为什么？查看jaeger可以发现什么?
http://jaeger.lab.f5se.io:32686/

原因是什么？怎么解决?

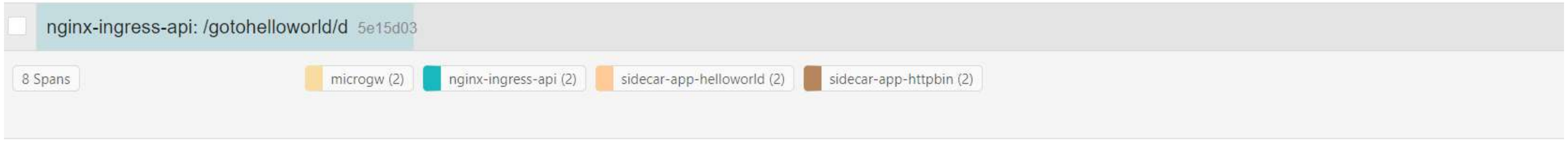


```
[root@k8s-master nginx-sidecar]# kubectl get pod,svc -o wide
NAME                                             READY     STATUS             RESTARTS   AGE       IP             NODE
pod/helloworld-sidecar-6b76979dd4-p7jc5          2/2       Running            0          22m       10.244.1.216   k8s-node1
pod/httpbin-sidecar-7ffcfd7b49-w58zd             2/2       Running            0          16m       10.244.1.217   k8s-node1
pod/jaeger-65fcbf66db-9gt2q                      1/1       Running            1          9d        10.244.0.227   k8s-master
pod/microgw-api-nginx-tracing-57bf585bf4-bg56w   1/1       Running            1          9d        10.244.0.224   k8s-master
pod/tools                                        0/1       ImagePullBackOff   0          1h        10.244.0.229   k8s-master

NAME                       TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)                               AGE       SELECTOR
service/apidemo            ClusterIP   10.250.0.187   <none>        443/TCP,80/TCP                        16m       app=httpbin-sidecar
service/apidemo2           ClusterIP   10.250.0.157   <none>        443/TCP,80/TCP                        22m       app=helloworld-sidecar
service/jaeger-agent       ClusterIP   10.250.0.122   <none>        5775/UDP,6831/UDP,6832/UDP,5778/TCP   9d        app.kubernetes.io/component=all-in-one,app.kubernetes.io/name=jaeger
service/jaeger-collector   ClusterIP   10.250.0.246   <none>        14267/TCP,14268/TCP,9411/TCP          9d        app.kubernetes.io/component=all-in-one,app.kubernetes.io/name=jaeger
service/jaeger-query       NodePort    10.250.0.139   <none>        80:32686/TCP                          9d        app.kubernetes.io/component=all-in-one,app.kubernetes.io/name=jaeger
service/kubernetes         ClusterIP   10.250.0.1     <none>        443/TCP                               1y        <none>
service/microgw-api        NodePort    10.250.0.23    <none>        443:32610/TCP,80:32014/TCP            9d        app=microgw-api-nginx-tracing
service/nginx-deploy-svc   ClusterIP   10.250.0.117   <none>        80/TCP                                100d      app=nginx-deploy
service/zipkin             ClusterIP   None           <none>        9411/TCP                              9d        app.kubernetes.io/component=all-in-one,app.kubernetes.io/name=jaeger
```

正常访问



```
/:path1/:path2 - Hello to gotohelloworld/d ! Host:helloworld-sidecar-6b76979dd4-p7jc5/10.244.1.216
```

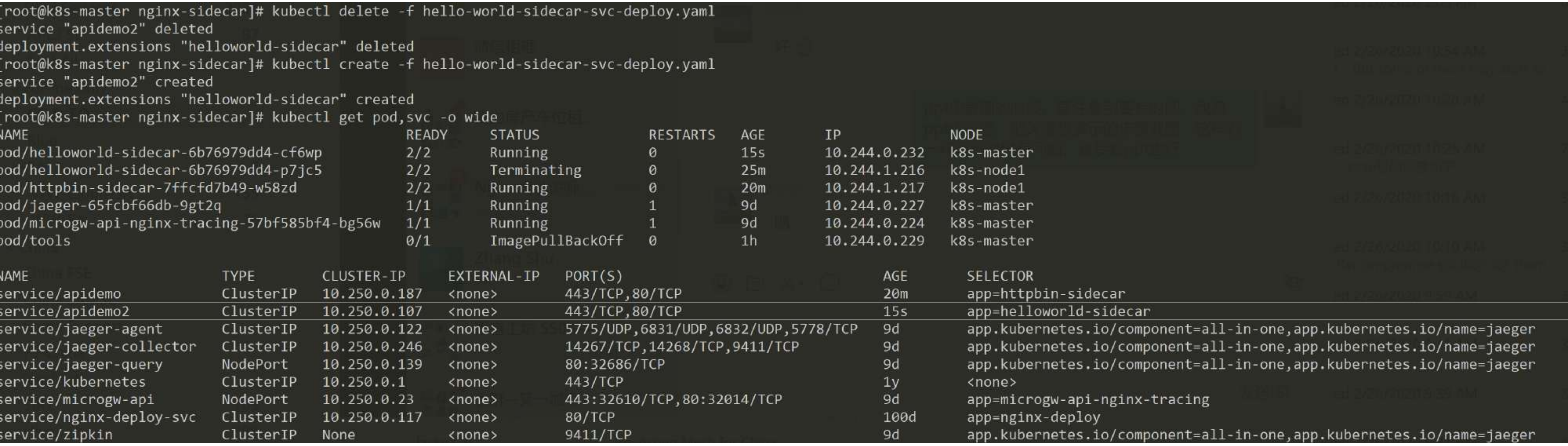


nginx-ingress-api: /gotohelloworld/d 5e15d03

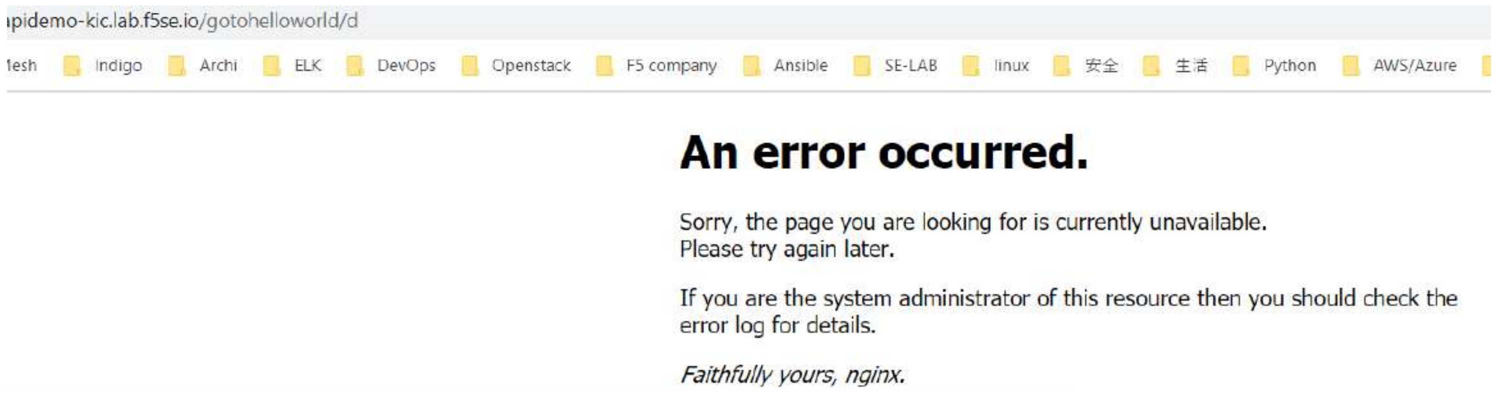
8 Spans microgw (2) nginx-ingress-api (2) sidecar-app-helloworld (2) sidecar-app-httpbin (2)

删除并重建hello world service

kubectl delete -f hello-world-sidecar-svc-deploy.yaml
kubectl create -f hello-world-sidecar-svc-deploy.yaml



```
[root@k8s-master nginx-sidecar]# kubectl delete -f hello-world-sidecar-svc-deploy.yaml
service "apidemo2" deleted
deployment.extensions "helloworld-sidecar" deleted
[root@k8s-master nginx-sidecar]# kubectl create -f hello-world-sidecar-svc-deploy.yaml
service "apidemo2" created
deployment.extensions "helloworld-sidecar" created
[root@k8s-master nginx-sidecar]# kubectl get pod,svc -o wide
NAME                                         READY     STATUS             RESTARTS   AGE       IP             NODE
pod/helloworld-sidecar-6b76979dd4-cf6wp      2/2       Running            0          15s       10.244.0.232   k8s-master
pod/helloworld-sidecar-6b76979dd4-p7jc5      2/2       Terminating        0          25m       10.244.1.216   k8s-node1
pod/httpbin-sidecar-7ffcfd7b49-w58zd         2/2       Running            0          20m       10.244.1.217   k8s-node1
pod/jaeger-65fcbf66db-9gt2q                  1/1       Running            1          9d        10.244.0.227   k8s-master
pod/microgw-api-nginx-tracing-57bf585bf4-bg56w   1/1   Running            1          9d        10.244.0.224   k8s-master
pod/tools                                    0/1       ImagePullBackOff   0          1h        10.244.0.229   k8s-master

NAME                       TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)                                  AGE       SELECTOR
service/apidemo            ClusterIP   10.250.0.187   <none>        443/TCP,80/TCP                           20m       app=httpbin-sidecar
service/apidemo2           ClusterIP   10.250.0.107   <none>        443/TCP,80/TCP                           15s       app=helloworld-sidecar
service/jaeger-agent       ClusterIP   10.250.0.122   <none>        5775/UDP,6831/UDP,6832/UDP,5778/TCP      9d        app.kubernetes.io/component=all-in-one,app.kubernetes.io/name=jaeger
service/jaeger-collector   ClusterIP   10.250.0.246   <none>        14267/TCP,14268/TCP,9411/TCP             9d        app.kubernetes.io/component=all-in-one,app.kubernetes.io/name=jaeger
service/jaeger-query       NodePort    10.250.0.139   <none>        80:32686/TCP                             9d        app.kubernetes.io/component=all-in-one,app.kubernetes.io/name=jaeger
service/kubernetes         ClusterIP   10.250.0.1     <none>        443/TCP                                  1y        <none>
service/microgw-api        NodePort    10.250.0.23    <none>        443:32610/TCP,80:32014/TCP               9d        app=microgw-api-nginx-tracing
service/nginx-deploy-svc   ClusterIP   10.250.0.117   <none>        80/TCP                                   100d      app=nginx-deploy
service/zipkin             ClusterIP   None           <none>        9411/TCP                                 9d        app.kubernetes.io/component=all-in-one,app.kubernetes.io/name=jaeger
```



ıpidemo-kic.lab.f5se.io/gotohelloworld/d

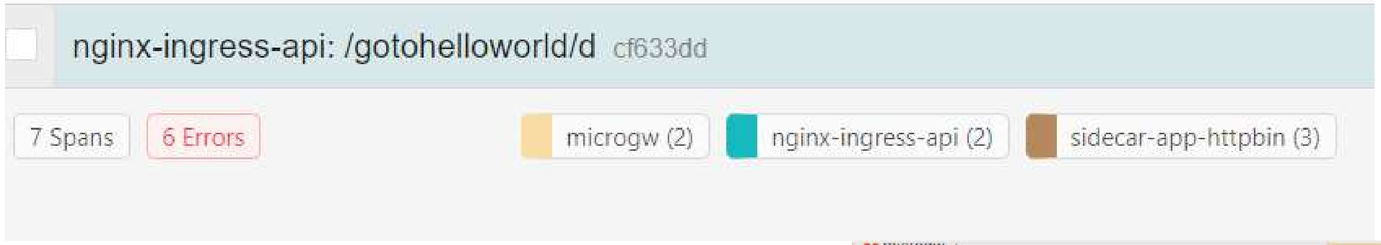
ıesh | Indigo | Archi | ELK | DevOps | Openstack | F5 company | Ansible | SE-LAB | linux | 安全 | 生活 | Python | AWS/Azure

An error occurred.

Sorry, the page you are looking for is currently unavailable.
Please try again later.

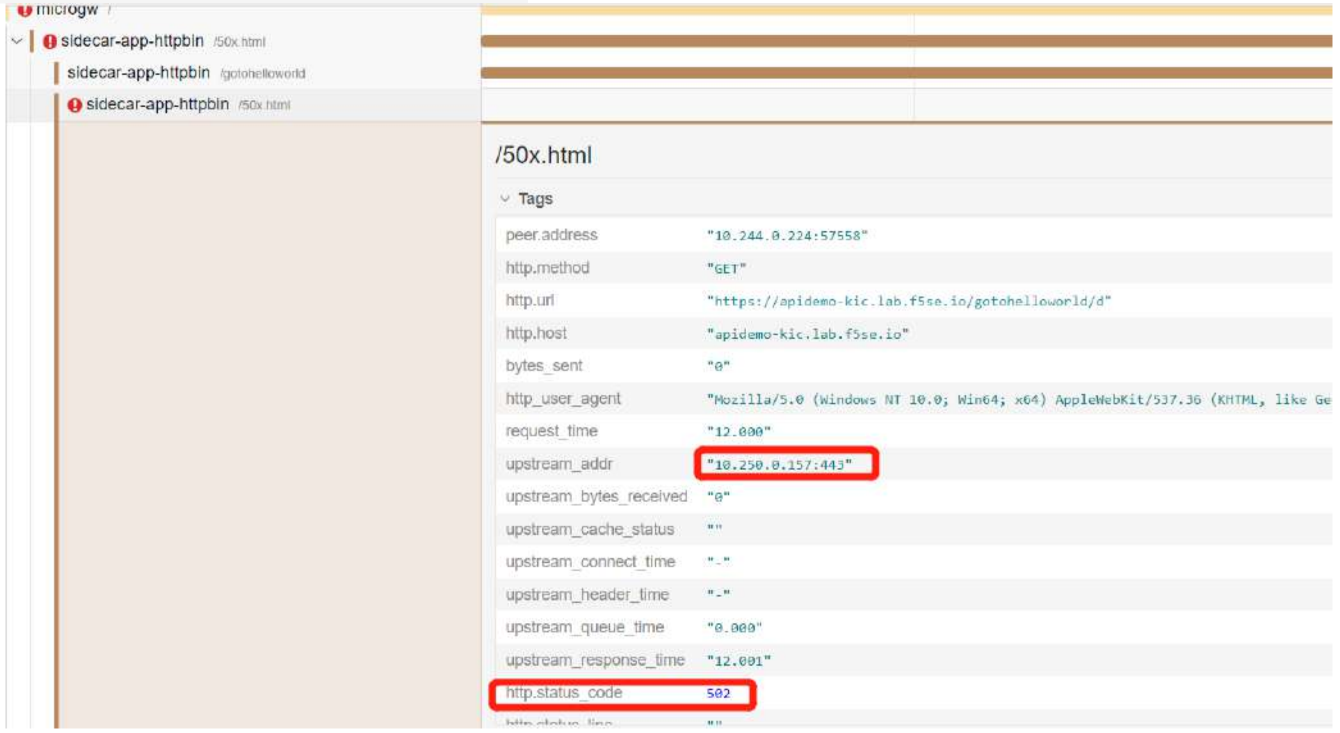
If you are the system administrator of this resource then you should check the error log for details.

Faithfully yours, nginx.



nginx-ingress-api: /gotohelloworld/d cf633dd

7 Spans | 6 Errors | microgw (2) | nginx-ingress-api (2) | sidecar-app-httpbin (3)



microgw /
sidecar-app-httpbin /50x.html
sidecar-app-httpbin /gotohelloworld
sidecar-app-httpbin /50x.html

/50x.html

Tags

peer.address	"10.244.0.224:57558"
http.method	"GET"
http.url	"https://apidemo-kic.lab.f5se.io/gotohelloworld/d"
http.host	"apidemo-kic.lab.f5se.io"
bytes_sent	"0"
http_user_agent	"Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Ge
request_time	"12.000"
upstream_addr	"10.250.0.157:443"
upstream_bytes_received	"0"
upstream_cache_status	""
upstream_connect_time	"-"
upstream_header_time	"-"
upstream_queue_time	"0.000"
upstream_response_time	"12.001"
http.status_code	502

原因

```
proxy_pass https://apidemo2.default.svc.cluster.local;
```

怎么解决?

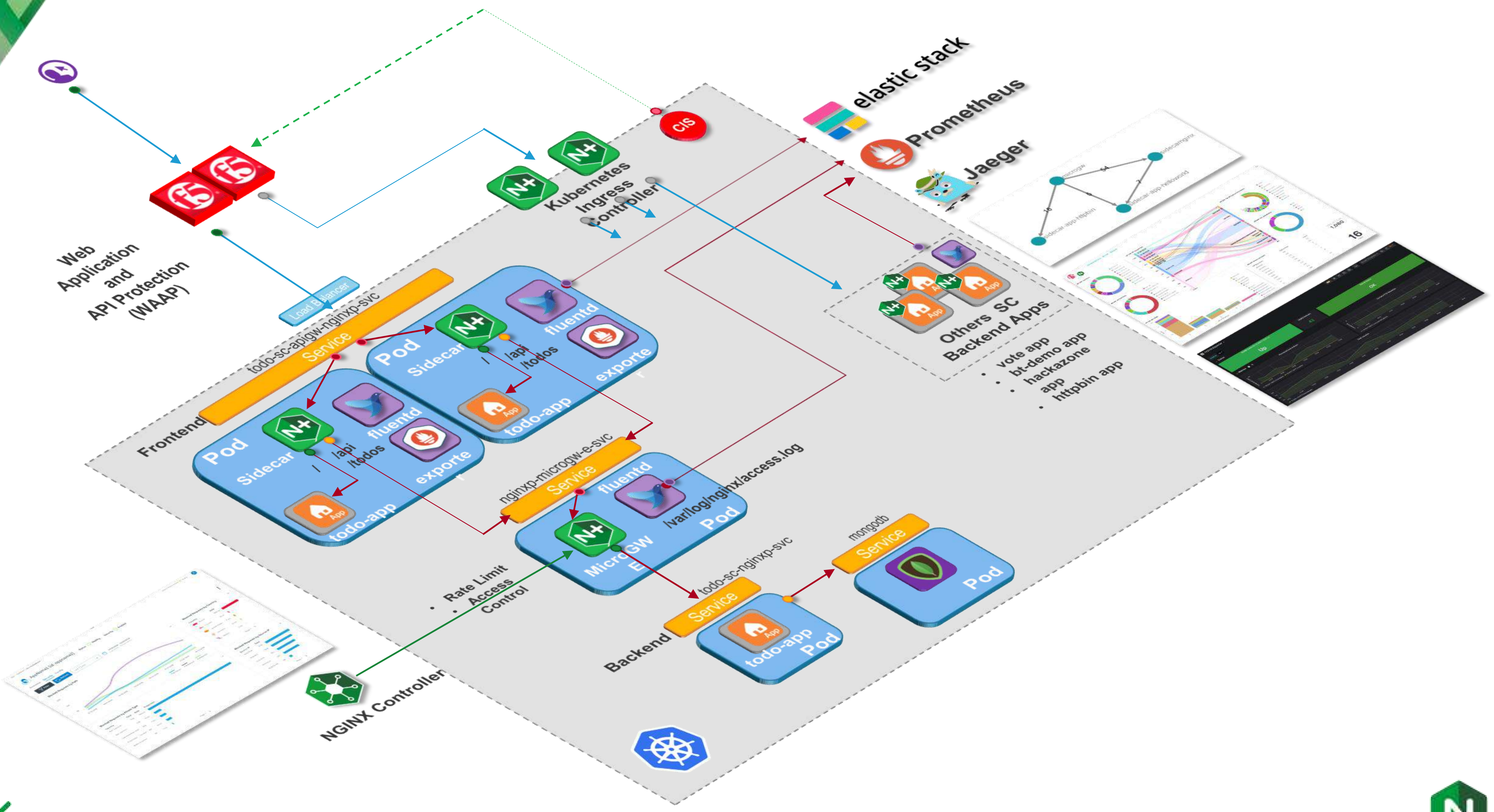
```
resolver 10.250.0.53 valid=3s;
upstream apidemo2backends {
        zone apidemo2backends 32k;
        server apidemo2.default.svc.cluster.local service=_https._tcp resolve;
 }
```

留个问题

Upstream的上游是多个公有云中的服务，例如 svc.aws.com, svc.azure.com,svc.ali.com
且这些服务对应的IP可能会发生变化

NGINX作为反向代理，应如何设置upstream才能确保随时可以正常访问这些业务?

https://www.myf5.net/post/2791.htm


Web Application and API Protection (WAAP)
Load Balancer
CIS
Kubernetes Ingress Controller
elastic stack
Prometheus
Jaeger
todo-sc-apigw-nginxp-svc
Service
Frontend
Pod
Sidecar
/
/api
/todos
fluentd
exporter
todo-app
App
nginxp-microgw-e-svc
Service
fluentd
/var/log/nginx/access.log
Pod
MicroGW
Others SC Backend Apps
• vote app
• bt-demo app
• hackazone app
• httpbin app
• Rate Limit
• Access Control
todo-sc-nginxp-svc
Service
Backend
todo-app
Pod
mongodb
Service
Pod
NGINX Controller
NGINX

Thank You

关注我们

F5 官方微信
(新闻，技术文章)

F5社区
(答疑，吐槽，分享，互动)

NGINX技术群

加入F5社区：关注“F5社区”微信公众号，注册成为社员，随时参加meet up活动，代码共享，讨论，答疑等。只要你有想法，有创造，那么就快来大展身手吧，让我们在社区里尽情分享，交流，吐槽和互动，在这个自由的国度里，发现闪亮的自己。让我们一起来见证“一群有才能的人在一起做有梦想的事！

操作步骤：

1. 扫描二维码并在“入群信息”栏填写姓名
2. 点击下方“我要入群”
3. 长按识别二维码进入群聊