链滴

golang 链路追踪之 jaeger(http)

作者：wx-shi

原文链接：https://ld246.com/article/1564238427681

来源网站：链滴

OpenTracing

- 开放式分布式追踪规范，详细介绍请自行百度 google。
- 常见实现：
  - jaeger
  - zipkin

jaeger

- Jaeger是Uber开发的一套分布式追踪系统。详细介绍请自行百度google。
- docker简单启动：

```
docker run \
  -p 5775:5775/udp \
  -p 16686:16686 \
  -p 6831:6831/udp \
  -p 6832:6832/udp \
  -p 5778:5778 \
  -p 14268:14268 \
  jaegertracing/all-in-one:latest
```

- 进入 ui
  - http://localhost:16686

golang 接入 jaeger

- tracer创建

```
package tracer

import (
    "io"
    "time"

    "github.com/opentracing/opentracing-go"
    "github.com/uber/jaeger-client-go"
    jaegercfg "github.com/uber/jaeger-client-go/config"
)

var Tracer opentracing.Tracer

// NewTracer 创建一个jaeger Tracer
func NewTracer(servicename string, addr string) (opentracing.Tracer, io.Closer, error) {
    cfg := jaegercfg.Configuration{
        ServiceName: servicename,
        Sampler: &jaegercfg.SamplerConfig{
            Type:  jaeger.SamplerTypeConst,
            Param: 1,
```

原文链接：golang 链路追踪之 jaeger(http)

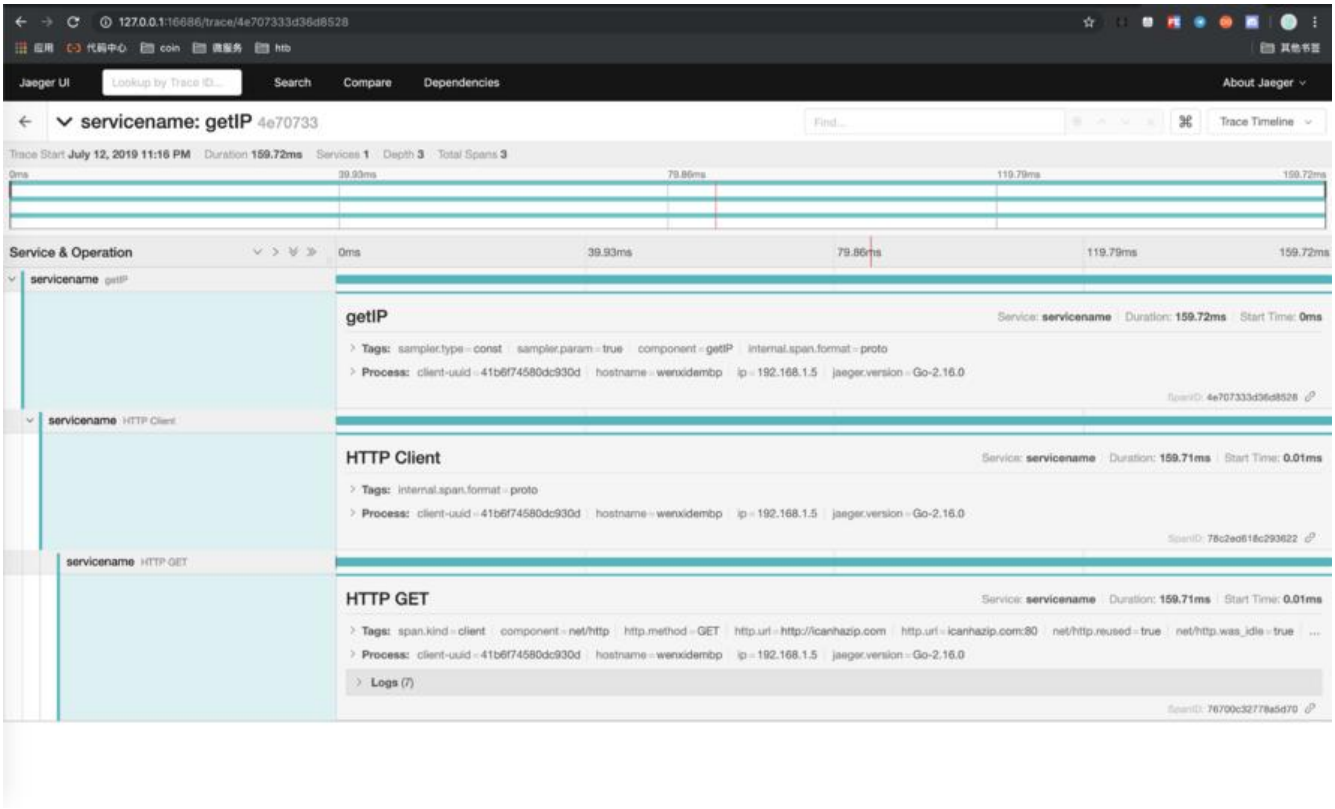
```
        },
        Reporter: &jaegercfg.ReporterConfig{
            LogSpans:            true,
            BufferFlushInterval: 1 * time.Second,
        },
    }

    sender, err := jaeger.NewUDPTransport(addr, 0)
    if err != nil {
        return nil, nil, err
    }

    reporter := jaeger.NewRemoteReporter(sender)
    // Initialize tracer with a logger and a metrics factory
    tracer, closer, err := cfg.NewTracer(
        jaegercfg.Reporter(reporter),
    )

    return tracer, closer, err
}
```

- http调用

```
package main

import (
    "context"
    "fmt"
    "github.com/opentracing-contrib/go-stdlib/nethttp"
    "github.com/opentracing/opentracing-go"
    "github.com/opentracing/opentracing-go/ext"
    otlog "github.com/opentracing/opentracing-go/log"
    "io"
    "io/ioutil"
    "jaeger_http/tracer"
    "log"
    "net/http"
)

func main() {
    var (
        err error
        io  io.Closer
    )
    //创建tracer对象
    tracer.Tracer, io, err = tracer.NewTracer("servicename", "127.0.0.1:6831")
    if err != nil {
        log.Fatal("tracer.NewTracer error(%v)", err)
    }
    defer io.Close()
    opentracing.SetGlobalTracer(tracer.Tracer)

    //server
    http.HandleFunc("/getIP", getIP)
```

```
    log.Printf("Starting server on port %d", 8002)
    err = http.ListenAndServe(
        fmt.Sprintf(":%d", 8002),
        // use nethttp.Middleware to enable OpenTracing for server
        nethttp.Middleware(tracer.Tracer, http.DefaultServeMux))
    if err != nil {
        log.Fatalf("Cannot start server: %s", err)
    }
}

func getIP(w http.ResponseWriter, r *http.Request) {
    log.Print("Received getIP request")

    //client
    client := &http.Client{Transport: &nethttp.Transport{}}
    span := tracer.Tracer.StartSpan("getIP")
    span.SetTag(string(ext.Component), "getIP")
    defer span.Finish()
    ctx := opentracing.ContextWithSpan(context.Background(), span)
    req, err := http.NewRequest(
        "GET",
        "http://icanhazip.com",
        nil,
    )
    if err != nil {
        log.Fatal(err)
    }

    req = req.WithContext(ctx)
    // wrap the request in nethttp.TraceRequest
    req, ht := nethttp.TraceRequest(tracer.Tracer, req)
    defer ht.Finish()

    res, err := client.Do(req)
    if err != nil {
        onError(span, err)
        return
    }
    defer res.Body.Close()
    body, err := ioutil.ReadAll(res.Body)
    if err != nil {
        onError(span, err)
        return
    }
    log.Printf("Received result: %s\n", body)
    io.WriteString(w, fmt.Sprintf("ip %s", body))
}

func onError(span opentracing.Span, err error) {
    span.SetTag(string(ext.Error), true)
    span.LogKV(otlog.Error(err))
    log.Fatal("client(%v)", err)
}
```

- 效果演示



原文链接：golang 链路追踪之 jaeger(http)