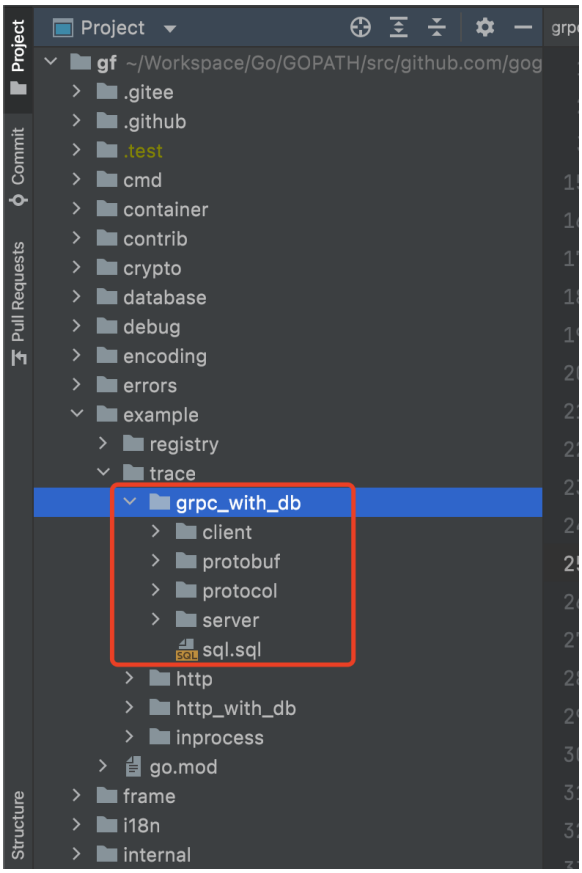
-GRPC

`HTTP Client&ServerGRPCGoFrameGRPCGRPC`

https://github.com/gogf/gf/tree/master/example/trace/grpc_with_db

Content Menu



Protobuf

```
syntax = "proto3";

package user;

option go_package = "protobuf/user";

// User service for tracing demo.
service User {
  rpc Insert(InsertReq) returns (InsertRes) {}
  rpc Query(QueryReq) returns (QueryRes) {}
  rpc Delete(DeleteReq) returns (DeleteRes) {}
}

message InsertReq {
  string Name = 1; // v: required#Please input user name.
}
message InsertRes {
  int32 Id = 1;
}

message QueryReq {
  int32 Id = 1; // v: min:1#User id is required for querying.
}
message QueryRes {
  int32  Id = 1;
  string Name = 2;
}

message DeleteReq {
  int32 Id = 1; // v:min:1#User id is required for deleting.
}
message DeleteRes {}
```

```
gf gen pbprotogrpc
```

GRPC Server

```
package main

import (
        _ "github.com/gogf/gf/contrib/drivers/mysql/v2"
        _ "github.com/gogf/gf/contrib/nosql/redis/v2"
        "github.com/gogf/gf/contrib/registry/etcd/v2"
        "github.com/gogf/gf/example/trace/grpc_with_db/protobuf/user"

        "context"
        "fmt"
        "time"

        "github.com/gogf/gf/contrib/rpc/grpcx/v2"
        "github.com/gogf/gf/contrib/trace/otlpgrpc/v2"
        "github.com/gogf/gf/v2/database/gdb"
        "github.com/gogf/gf/v2/frame/g"
        "github.com/gogf/gf/v2/os/gcache"
        "github.com/gogf/gf/v2/os/gctx"
)

type Controller struct {
        user.UnimplementedUserServer
}

const (
    serviceName = "otlp-grpc-server"
        endpoint    = "tracing-analysis-dc-bj.aliyuncs.com:8090"
        traceToken  = "******_******"
)
```

```
func main() {
        grpcx.Resolver.Register(etcd.New("127.0.0.1:2379"))

        var ctx = gctx.New()
    shutdown, err := otlpgrpc.Init(serviceName, endpoint, traceToken)
        if err != nil {
                g.Log().Fatal(ctx, err)
        }
        defer shutdown()

    // Set ORM cache adapter with redis.
        g.DB().GetCache().SetAdapter(gcache.NewAdapterRedis(g.Redis()))

        s := grpcx.Server.New()
        user.RegisterUserServer(s.Server, &Controller{})
        s.Run()
}

// Insert is a route handler for inserting user info into database.
func (s *Controller) Insert(ctx context.Context, req *user.InsertReq) (res
*user.InsertRes, err error) {
        result, err := g.Model("user").Ctx(ctx).Insert(g.Map{
                "name": req.Name,
        })
        if err != nil {
                return nil, err
        }
        id, _ := result.LastInsertId()
        res = &user.InsertRes{
                Id: int32(id),
        }
        return
}

// Query is a route handler for querying user info. It firstly retrieves
the info from redis,
// if there's nothing in the redis, it then does db select.
func (s *Controller) Query(ctx context.Context, req *user.QueryReq) (res
*user.QueryRes, err error) {
        err = g.Model("user").Ctx(ctx).Cache(gdb.CacheOption{
                Duration: 5 * time.Second,
                Name:     s.userCacheKey(req.Id),
                Force:    false,
        }).WherePri(req.Id).Scan(&res)
        if err != nil {
                return nil, err
        }
        return
}

// Delete is a route handler for deleting specified user info.
func (s *Controller) Delete(ctx context.Context, req *user.DeleteReq) (res
*user.DeleteRes, err error) {
        err = g.Model("user").Ctx(ctx).Cache(gdb.CacheOption{
                Duration: -1,
                Name:     s.userCacheKey(req.Id),
                Force:    false,
        }).WherePri(req.Id).Scan(&res)
        return
}

func (s *Controller) userCacheKey(id int32) string {
        return fmt.Sprintf(`userInfo:%d`, id)
}
```

1jaeger.InitJaeger

2HTTPGRPC

3Redis

```
g.DB().GetCache().SetAdapter(gcache.NewAdapterRedis(g.Redis()))
```

5CacheORM

GRPC Client

```
package main

import (
        "github.com/gogf/gf/contrib/registry/etcd/v2"
        "github.com/gogf/gf/contrib/rpc/grpcx/v2"
        "github.com/gogf/gf/contrib/trace/otlpgrpc/v2"
        "github.com/gogf/gf/example/trace/grpc_with_db/protobuf/user"
        "github.com/gogf/gf/v2/frame/g"
        "github.com/gogf/gf/v2/net/gtrace"
        "github.com/gogf/gf/v2/os/gctx"
)

const (
         serviceName = "otlp-grpc-client"
        endpoint    = "tracing-analysis-dc-bj.aliyuncs.com:8090"
        traceToken  = "******_******"
)

func main() {
        grpcx.Resolver.Register(etcd.New("127.0.0.1:2379"))

        var ctx = gctx.New()
    shutdown, err := otlpgrpc.Init(serviceName, endpoint, traceToken)
        if err != nil {
                g.Log().Fatal(ctx, err)
        }
        defer shutdown()

    StartRequests()
}

func StartRequests() {
        ctx, span := gtrace.NewSpan(gctx.New(), "StartRequests")
        defer span.End()

        client := user.NewUserClient(grpcx.Client.MustNewGrpcClientConn
("demo"))

        // Baggage.
        ctx = gtrace.SetBaggageValue(ctx, "uid", 100)

        // Insert.
        insertRes, err := client.Insert(ctx, &user.InsertReq{
                Name: "john",
        })
        if err != nil {
                g.Log().Fatalf(ctx, `%+v`, err)
        }
        g.Log().Info(ctx, "insert id:", insertRes.Id)

        // Query.
        queryRes, err := client.Query(ctx, &user.QueryReq{
                Id: insertRes.Id,
        })
        if err != nil {
                g.Log().Errorf(ctx, `%+v`, err)
                return
        }
        g.Log().Info(ctx, "query result:", queryRes)
```

```
        // Delete.
        _, err = client.Delete(ctx, &user.DeleteReq{
                Id: insertRes.Id,
        })
        if err != nil {
                g.Log().Errorf(ctx, `%+v`, err)
                return
        }
        g.Log().Info(ctx, "delete id:", insertRes.Id)

        // Delete with error.
        _, err = client.Delete(ctx, &user.DeleteReq{
                Id: -1,
        })
        if err != nil {
                g.Log().Errorf(ctx, `%+v`, err)
                return
        }
        g.Log().Info(ctx, "delete id:", -1)
}
```

1jaeger.InitJaeger

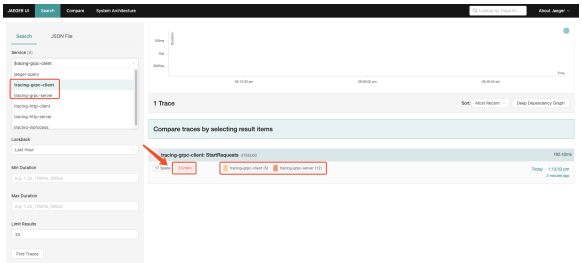
2Katyusha

```
➜  gf-tracing git:(master) go run grpc+db+redis+log/server/main.go
2021-01-29 13:13:28.564 grpc server starts listening on :8080
2021-01-29 13:13:33.281 [DEBU] {TraceID:d78bb60bfc15c3ca5cb1564be84e511e} [122 ms] [default] SHOW FULL COLUMNS FROM `user`
2021-01-29 13:13:33.312 [DEBU] {TraceID:d78bb60bfc15c3ca5cb1564be84e511e} [ 31 ms] [default] INSERT INTO `user`(`name`) VALUES('john')
2021-01-29 13:13:33.318 [DEBU] {TraceID:d78bb60bfc15c3ca5cb1564be84e511e} [  3 ms] [default] SELECT * FROM `user` WHERE `uid`=5 LIMIT 1
2021-01-29 13:13:33.335 [DEBU] {TraceID:d78bb60bfc15c3ca5cb1564be84e511e} [ 14 ms] [default] DELETE FROM `user` WHERE `uid`=5
```

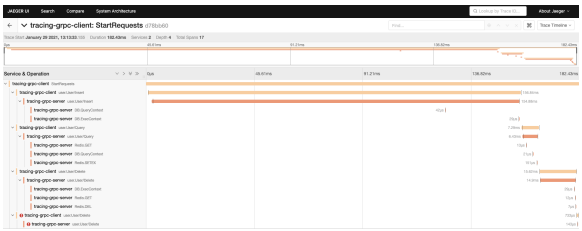
```
➜  gf-tracing git:(master) go run grpc+db+redis+log/client/main.go
2021-01-29 13:13:33.313 {TraceID:d78bb60bfc15c3ca5cb1564be84e511e} insert: 5
2021-01-29 13:13:33.321 {TraceID:d78bb60bfc15c3ca5cb1564be84e511e} query: Name:"john"
2021-01-29 13:13:33.336 {TraceID:d78bb60bfc15c3ca5cb1564be84e511e} delete: 5
2021-01-29 13:13:33.337 {TraceID:d78bb60bfc15c3ca5cb1564be84e511e} User id is required for deleting.
1. User id is required for deleting.
    1).  github.com/gogf/katyusha/krpc.(*krpcClient).UnaryError
         /Users/john/Workspace/Go/GOPATH/pkg/mod/github.com/gogf/katyusha@v0.1.2-0.20210120868058-6fff85f91cc9/krpc/krpc_interceptor_client.go:28
    2).  google.golang.org/grpc.chainUnaryClientInterceptors.func1
         /Users/john/Workspace/Go/GOPATH/pkg/mod/google.golang.org/grpc@v1.29.1/clientconn.go:348
    3).  google.golang.org/grpc.(*ClientConn).Invoke
         /Users/john/Workspace/Go/GOPATH/pkg/mod/google.golang.org/grpc@v1.29.1/call.go:35
    4).  gftracing/grpc+db+redis+log/protobuf/user.(*userClient).Delete
         /Users/john/Workspace/Go/GOPATH/src/github.com/gogf/gf-tracing/grpc+db+redis+log/protobuf/user/user.pb.go:394
    5).  main.StartRequests
         /Users/john/Workspace/Go/GOPATH/src/github.com/gogf/gf-tracing/grpc+db+redis+log/client/main.go:98
    6).  main.main
         /Users/john/Workspace/Go/GOPATH/src/github.com/gogf/gf-tracing/grpc+db+redis+log/client/main.go:23

➜  gf-tracing git:(master)
```

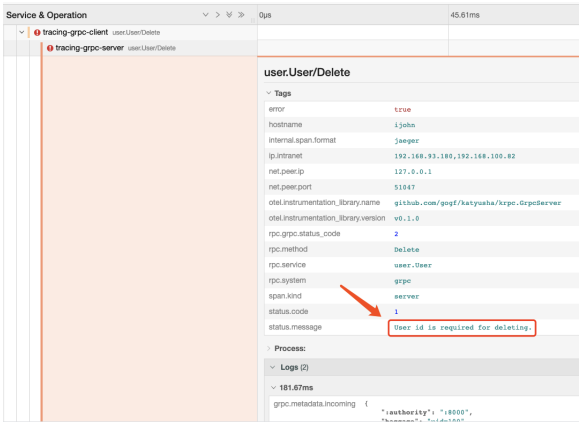
GRPCjaeger



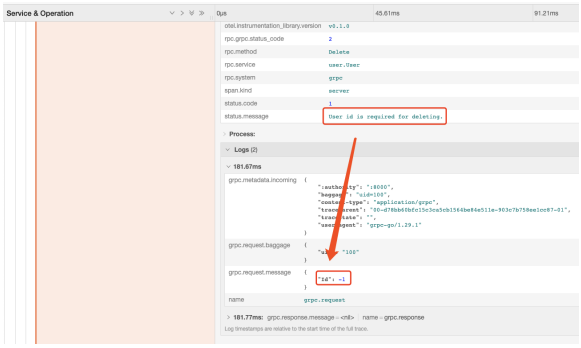
tracing-grpc-clienttracing-grpc-server17span5span12span2



`span`



`Events/Logs`

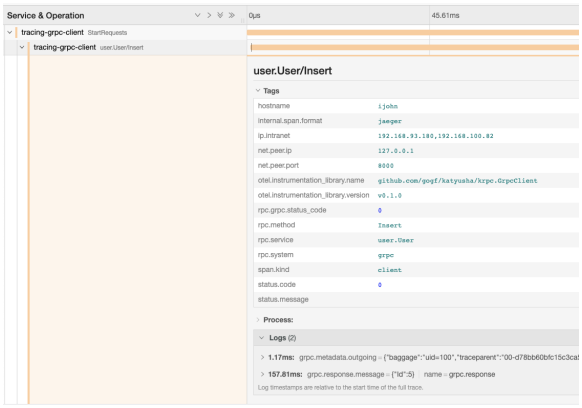


`ProcessLog-1`

GRPC Client

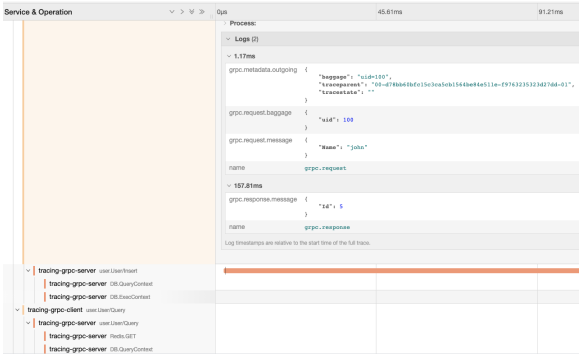
`ormredisloggingGRPC Client&Server`

Attributes



Attribute/Tag	
net.peer.ip	IP
net.peer.port	
rpc.grpc. status_code	GRPC00
rpc.service	RPCRPCGRPCRPCGRPC
rpc.method	RPC
rpc.system	RPCgrpc, thrift

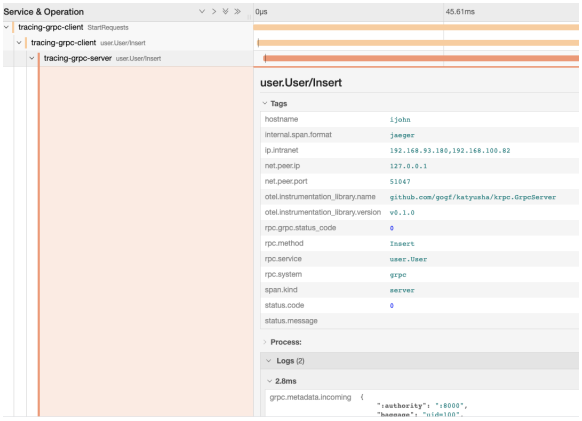
Events/Logs



Event/Log	
grpc.metadata.outgoing	GRPCMetadata
grpc.request.baggage	GRPCBaggage
grpc.request.message	GRPCMessage512KBUnary
grpc.response.message	GRPCMessageUnary

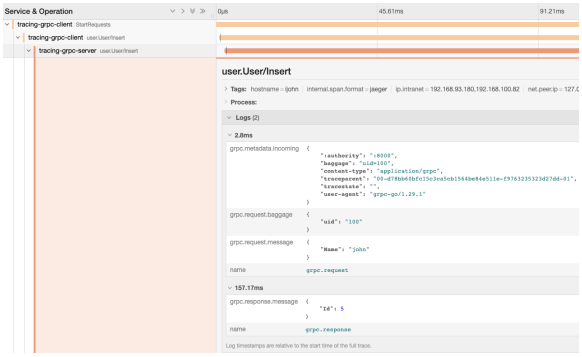
GRPC Server

Attributes



GRPC ServerAttributesGRPC Client

Events



GRPC ServerEventsGRPC Clientmetadatagrpc.metadata.incomingGRPC Client