NGINX 宣布支持 gRPC，可在下个版本 1.13.10 中使用

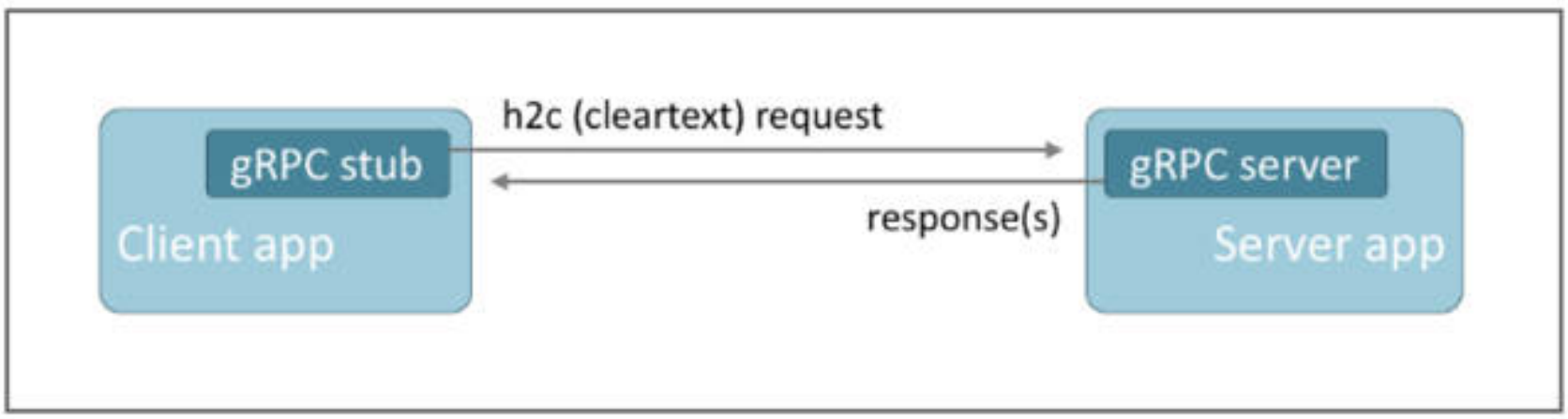
近日，NGINX 在其博客宣布，NGINX 已完成对 gRPC 的原生支持，并将在下一个 OSS 版本 1.13.10 中提供使用，如果迫不及待希望尝鲜，可下载 snapshot 快照来体验一把，也可以给开发团队反馈意见。

而下一个 NGINX Plus 版本将引入对 gRPC 和 HTTP/2 服务器推送的支持。

有了对 gRPC 的支持，NGINX 可以代理 gRPC TCP 连接，还可以终止、检查和跟踪 gRPC 的方法调用。你可以：

关于 gRPC

gRPC是一种远程过程调用协议，用于客户端和服务器应用程序之间的通信。他具有紧凑（节省空间）和可跨多种语言移植的特点，并且支持请求响应和流式交互。 由于其广泛的语言支持和简单的面向用户的设计，该协议越来越受欢迎，其中包括服务网格实现。



一个简单的基于 gRPC 的应用程序

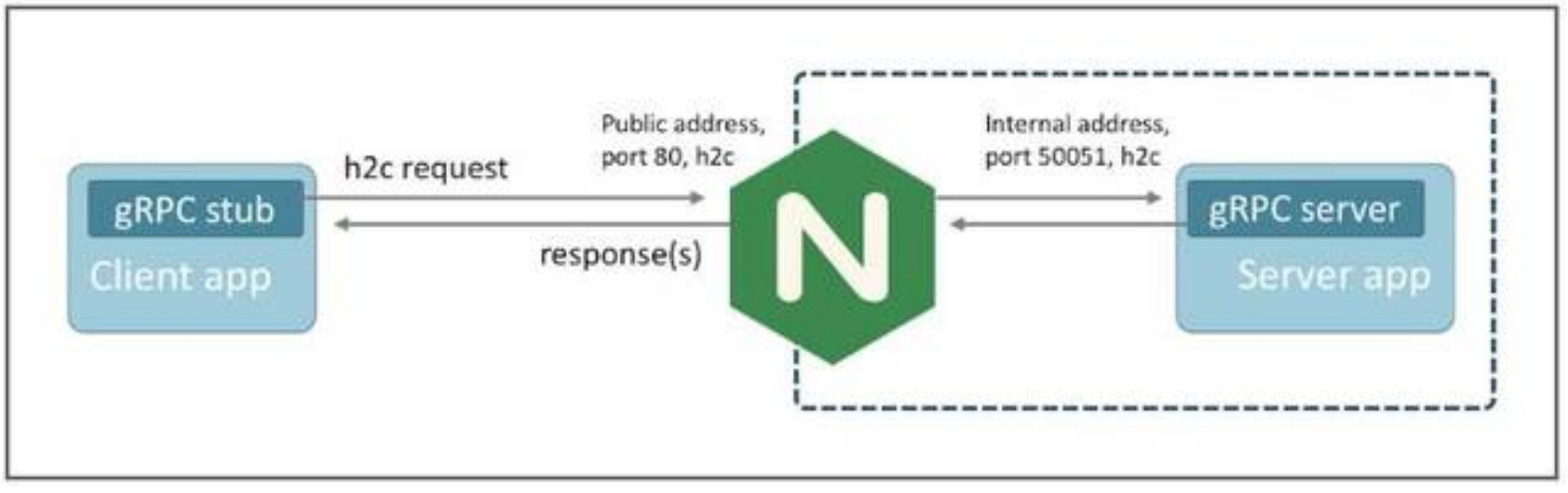
gRPC 通过 HTTP / 2 传输，无论是明文还是 TLS 加密。 gRPC 调用被实现为具有高效编码主体的 HTTP POST 请求（协议缓冲区是标准编码）。 gRPC 响应使用类似的编码体，并在响应结束时使用 HTTP trailers 发送状态码。

按照设计，gRPC 协议不能通过 HTTP 传输。 gRPC 协议规定使用 HTTP / 2，是为了利用 HTTP / 2 连接的复用和流式传输功能。

使用 NGINX 管理 gRPC 服务

公开简单的 gRPC 服务

首先，我们在客户端和服务器应用程序之间插入 NGINX。 NGINX 为服务器应用程序提供了一个稳定可靠的网关。



NGINX 代理 gRPC 流量

先通过 gRPC 更新部署 NGINX。 如果您想从源代码构建 NGINX，记得包含 http_ssl 和 http_v2 模块：

```
$ auto/configure --with-http_ssl_module --with-http_v2_module
```

NGINX 使用 HTTP 服务器监听 gRPC 流量，并使用 grpc_pass 指令代理流量。 为 NGINX 创建以下代理配置，在端口 80 上侦听未加密的 gRPC 流量并将请求转发到端口 50051 上的服务器：

```
http {
    log_format  main  '$remote_addr - $remote_user [$time_local] "$request" '
                      '$status $body_bytes_sent "$http_referer" '
                      '"$http_user_agent"';

    server {
        listen 80 http2;

        access_log logs/access.log main;

        location / {
            # Replace localhost:50051 with the address and port of your gRPC server
            grpc_pass grpc://localhost:50051;
        }
    }
}
```

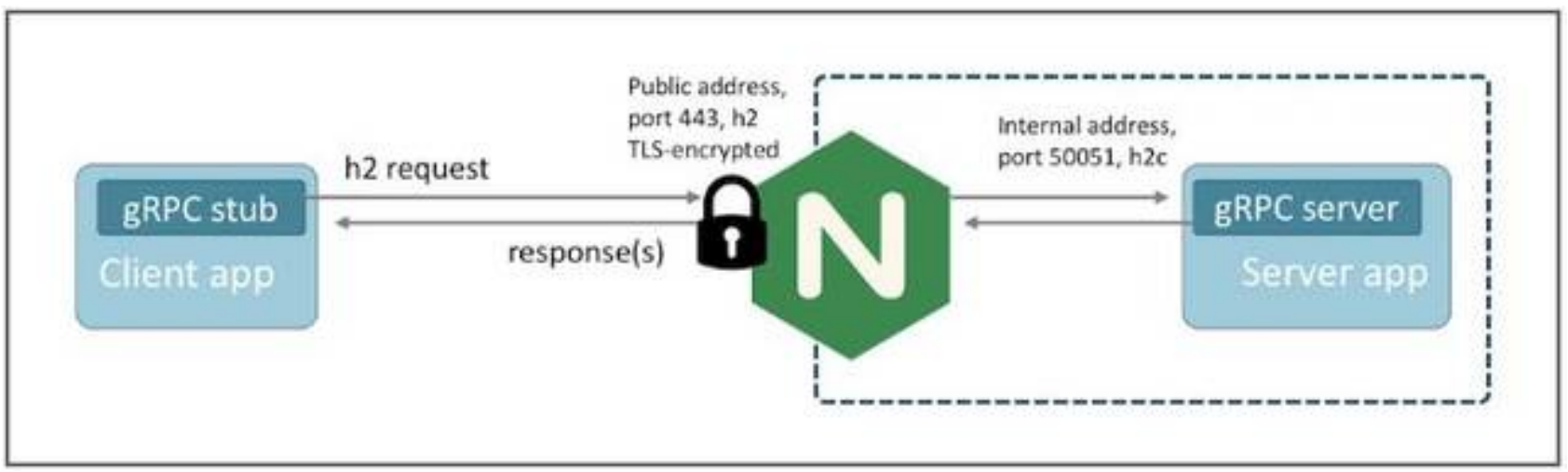
确保 grpc_pass 指令中的地址是正确的。 重新编译客户端以指向 NGINX 的 IP 地址并监听端口。

当您运行修改后的客户端时，您会看到与之前相同的响应，但事务将由 NGINX 终止并转发。 您可以在您配置的访问日志中看到它们：

```
$ tail logs/access.log
192.168.20.1 - - [01/Mar/2018:13:35:02 +0000] "POST /helloworld.Greeter/SayHello HTTP/2.0"
200 18 "-" "grpc-go/1.11.0-dev"
192.168.20.1 - - [01/Mar/2018:13:35:02 +0000] "POST /helloworld.Greeter/SayHelloAgain
HTTP/2.0" 200 24 "-" "grpc-go/1.11.0-dev"
```

使用 TLS 加密发布 gRPC 服务

Hello World 快速入门示例使用未加密的 HTTP / 2（明文）进行通信。 这对测试和部署来说非常简单，但它不提供生产部署所需的加密。 你可以使用 NGINX 来添加这个加密层。



NGINX SSL 终止 gRPC 流量

创建一个自签名证书对并修改您的 NGINX 服务器配置，如下所示：

```
server {
    listen 1443 ssl http2;

    ssl_certificate ssl/cert.pem;
    ssl_certificate_key ssl/key.pem;

    #...
}
```

修改 gRPC 客户端以使用 TLS，连接到端口 1443，并禁用证书检查 - 使用自签名或不可信证书时必需这么做。 例如，如果您使用 Go 示例，则需要：

```
creds := credentials.NewTLS( &tls.Config{ InsecureSkipVerify: true } )

// remember to update address to use the new NGINX listen port
conn, err := grpc.Dial( address, grpc.WithTransportCredentials( creds ) )
```

这就是使用 NGINX 来确保 gRPC 流量所需要做的。 在生产部署中，您需要将自签名证书替换为受信任的证书颁发机构（CA）颁发的证书。 客户端将被配置为信任该 CA。

代理加密的 gRPC 服务

您可能还想要在内部加密 gRPC 流量。 您首先需要修改服务器应用程序以侦听 TLS 加密（grpcs）而不是未加密（grpc）连接：

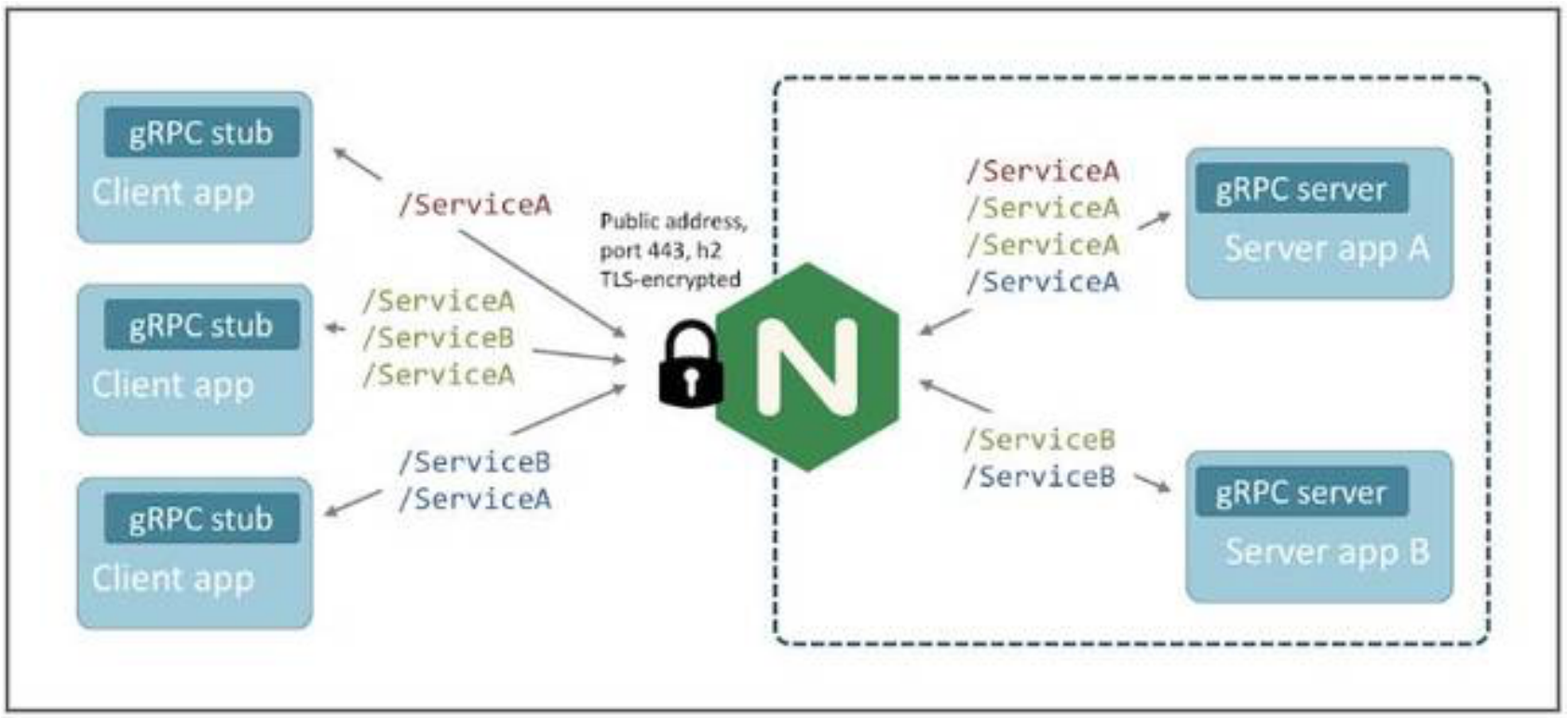
```
cer, err := tls.LoadX509KeyPair( "cert.pem", "key.pem" )
config := &tls.Config{ Certificates: []tls.Certificate{cer} }
lis, err := tls.Listen( "tcp", port, config )
```

在 NGINX 配置中，您需要更改用于将 gRPC 流量代理到上游服务器的协议：

```
# Use grpcs for TLS-encrypted gRPC traffic
grpc_pass grpcs://localhost:50051;
```

路由 gRPC 流量

如果您有多个 gRPC 服务，每个服务都由不同的服务器应用程序实现，你可以做什么？ 如果您可以通过单个 TLS 加密的端点发布所有这些服务，岂不是很棒？



NGINX 路由和 SSL 终止 gRPC 流量

使用 NGINX，您可以识别服务和方法，然后使用位置指令路由流量。 您可能已经推断出每个 gRPC 请求的 URL 是从 proto 规范中的包，服务和方法名称派生的。 考虑这个示例 SayHello RPC 方法：

```
package helloworld;

service Greeter {
    rpc SayHello (HelloRequest) returns (HelloReply) {}
}
```

调用 SayHello RPC 方法为 /helloworld.Greeter/SayHello 发出 POST 请求，如以下日志条目所示：

192.168.20.1 - - [01/Mar/2018:13:35:02 +0000] "POST /helloworld.Greeter/SayHello HTTP/2.0" 200 18 "-" "grpc-go/1.11.0-dev"

使用 NGINX 路由流量非常简单：

```
location /helloworld.Greeter {
    grpc_pass grpc://192.168.20.11:50051;
}

location /helloworld.Dispatcher {
    grpc_pass grpc://192.168.20.21:50052;
}

location / {
    root html;
    index index.html index.htm;
}
```

你可以自己尝试一下。 我们扩展了示例 Hello World 包（在 helloworld.proto 中）以添加一个名为 Dispatcher 的新服务，然后创建了一个实现 Dispatcher 方法的新服务器应用程序。 客户端使用单个 HTTP / 2 连接为 Greeter 和 Dispatcher 服务发出 RPC 调用。 NGINX 将呼叫分开并路由到合适的 gRPC 服务器。

请注意“catch - all” / location 块。 该块处理与已知 gRPC 调用不匹配的请求。 您可以使用像这样的位置块从相同的 TLS 加密端点提供 Web 内容和其他非 gRPC 服务。

负载平衡 gRPC 呼叫

您现在如何扩展 gRPC 服务以增加容量并提供高可用性？ NGINX 的上游团

队是这样做的：

```
upstream grpcservers {
    server 192.168.20.21:50051;
    server 192.168.20.22:50052;
}

server {
    listen 1443 ssl http2;

    ssl_certificate     ssl/certificate.pem;
    ssl_certificate_key ssl/key.pem;

    location /helloworld.Greeter {
        grpc_pass grpc://grpcservers;
        error_page 502 = /error502grpc;
    }

    location = /error502grpc {
        internal;
        default_type application/grpc;
        add_header grpc-status 14;
        add_header grpc-message "unavailable";
        return 204;
    }
}
```

当然，如果您的上游正在侦听 TLS，则可以使用 grpc_pass grpcs：//upstreams。

NGINX 可以采用一系列负载平衡算法来分配上游 gRPC 服务器上的 gRPC 呼叫。 NGINX 的内置运行状况检查将检测服务器是否无法响应或产生错误，然后使服务器停止运转。 如果没有服务器可用，则 / error502grpc 位置将返回符合 gRPC 的错误消息。

以上是 gRPC 代理支持的最初版本。原文链接：

有任何问题欢迎留言探讨，或通过以上链接进行反馈。

声明：本文由入驻搜狐号作者撰写，除搜狐官方账号外，观点仅代表作者本人，不代表搜狐立场。