

O'REILLY®

NGINX

Livro de receitas

Receitas avançadas para alto desempenho

Balanceamento de carga

Elogios de



NGINX®

Part of F5



Derek De Jonghe



NGINX Cookbook

NGINX is one of the most widely used web servers available today, in part because of its capabilities as a load balancer and reverse proxy server for HTTP and other network protocols. This cookbook provides easy-to-follow examples to real-world problems in application delivery. The practical recipes will help you set up and use either the open source or commercial offering to solve problems in various use cases.

For professionals who understand modern web architectures, such as n-tier or microservice designs, and common web protocols including TCP and HTTP, these recipes provide proven solutions for security, software load balancing, and monitoring and maintaining NGINX's application delivery platform. You'll also explore advanced features of both NGINX and NGINX Plus, the free and licensed versions of this server.

You'll find recipes for:

- High-performance load balancing with HTTP, TCP, and UDP
- Securing access through encrypted traffic, secure links, HTTP authentication subrequests, and more
- Deploying NGINX to Google Cloud, AWS, and Azure cloud computing services
- Setting up and configuring NGINX Controller
- Installing and configuring the NGINX Plus App Protect module
- Enabling WAF through Controller ADC

"NGINX is one of the most powerful and complete tools out there today, and this book is the ultimate instrumentation guide for large architectures. The use cases presented can help anyone resolve almost all difficulties that appear while working in a microservice environment without losing focus on the business."

—Gonzalo Spina
Software Engineer, Brubank

Derek DeJonghe specializes in cloud migrations and operations of all sizes. He leads a team of cloud architects and solution engineers to produce self-healing, autoscaling infrastructure for different applications. His in-depth background and experience in web development, system administration, and networking give him a well-rounded understanding of modern web architecture.

SYSTEM ADMINISTRATION

Twitter: @oreillymedia
facebook.com/oreilly

ISBN: 978-1-492-08702-1



9 781492 087021



Experimente o NGINX Plus e NGINX App Protect Gratuito



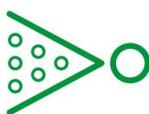
Obtenha entrega de aplicativos de alto desempenho e segurança para microsserviços. O NGINX Plus é um平衡ador de carga de software, gateway de API e proxy de microsserviços. O NGINX App Protect é um firewall de aplicativo da Web (WAF) leve e rápido, desenvolvido com a comprovada tecnologia F5 e projetado para aplicativos modernos e ambientes DevOps.



Custo

Poupança

Mais de 80% de economia de custos em comparação com controladores de entrega de aplicativos de hardware e WAFs, com todo o desempenho e recursos você espera.



Reduzido

Complexidade

O único balanceador de carga completo, gateway de API, proxy de microsserviços e firewall de aplicativo da Web ajuda a reduzir a expansão da infraestrutura.



Empreendimento

Preparar

NGINX Plus e NGINX App Protect oferece requisitos corporativos de segurança, escalabilidade e resiliência enquanto se integra a ambientes DevOps e CI/CD.



Avançado

Segurança

O NGINX App Protect oferece 4x o desempenho e 10x a taxa de transferência como alternativas de código aberto como ModSecurity, ao mesmo tempo em que fornece controles ainda mais abrangentes.

Baixe em nginx.com/freetrial



NGINX
Part of F5

Livro de receitas do NGINX

Receitas avançadas para alto desempenho

Balanceamento de carga

Derek De Jonghe

Beijing • Boston • Farnham • Sebastopol • Tokyo

O'REILLY®

Livro de receitas

NGINX por Derek DeJonghe

Copyright © 2021 O'Reilly Media, Inc. Todos os direitos reservados.

Impresso nos Estados Unidos da América.

Publicado por O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

Os livros da O'Reilly podem ser adquiridos para uso educacional, comercial ou promocional de vendas. Edições online também estão disponíveis para a maioria dos títulos (<http://oreilly.com>). Para mais informações, entre em contato com nosso departamento de vendas corporativo/institucional: 800-998-9938 ou corporate@oreilly.com.

Editor de aquisições: Mary Preap

Indexador: Sue Klefstad

Editor de desenvolvimento: Gary

Designer de Interiores: David

O'Brien **Editor de produção:** Christopher

Futato **Designer de Capa:** Karen

Faucher **Editor de texto:** Piper Editorial,

Montgomery **Ilustrador:** O'Reilly Media, Inc.

LLC **Revisor:** nSight, Inc.

Novembro de 2020: Primeira edição

Histórico de revisões para a primeira edição

28-10-2020: Primeiro lançamento

Consulte <http://oreilly.com/catalog/errata.csp?isbn=9781492078487> para detalhes do lançamento.

O logotipo O'Reilly é uma marca registrada da O'Reilly Media, Inc. NGINX Cookbook, a imagem da capa e a imagem comercial relacionada são marcas registradas da O'Reilly Media, Inc.

As opiniões expressas neste trabalho são de responsabilidade do autor e não representam a opinião da editora.

Embora o editor e o autor tenham feito esforços de boa fé para garantir que as informações e instruções contidas neste trabalho sejam precisas, o editor e o autor se isentam de qualquer responsabilidade por erros ou omissões, incluindo, sem limitação, responsabilidade por danos resultantes do uso ou confiança neste trabalho. O uso das informações e instruções contidas neste trabalho é por sua conta e risco. Se qualquer amostra de código ou outra tecnologia que este trabalho contém ou descreve está sujeita a licenças de código aberto ou direitos de propriedade intelectual de terceiros, é sua responsabilidade garantir que seu uso esteja em conformidade com tais licenças e/ou direitos.

Este trabalho é parte de uma colaboração entre O'Reilly e NGINX. Veja nossa declaração de [independência editorial](#).

978-1-492-08702-1

[LSI]

Índice

Prefácio.....	XI
Prefácio.....	xiii
1. Noções básicas.....	1
1.0 introdução	1
1.1 Instalando no Debian/Ubuntu 1.2	1
Instalando no RedHat/CentOS 1.3	2
Instalando o NGINX Plus 1.4 Verificando	3
sua instalação 1.5 Arquivos de chave,	3
diretórios e comandos 1.6 Servindo conteúdo	4
estático 1.7 Graceful Reload	6
	7
2. Balanceamento de carga de alto desempenho.....	9
2.0 Introdução	9
2.1 Balanceamento de carga	10
HTTP 2.2 Balanceamento de	11
carga TCP 2.3 Balanceamento	13
de carga UDP 2.4 Métodos de	14
balanceamento de carga 2.5 Sticky Cookie	16
com NGINX Plus 2.6 Sticky Learn com	17
NGINX Plus 2.7 Roteamento fixo com	18
NGINX Plus 2.8 Drenagem de conexão com NGINX	19
Plus 2.9 Verificações de integridade passivas 2.10	20
Verificações de integridade ativas com NGINX Plus	21
2.11 Início lento com NGINX Plus	23

3. Entre e Gestão.....	25
3.0 Introdução	25
3.1 Teste A/B	25
Usando o Módulo Geolocalização e Banco de Dados	27
Restringindo o Acesso Baseado no País	29
Encontrando o Cliente Original	30
3.5 Limitando as Conexões	30
3.6 Limitando a Taxa	31
3.7 Limitando a Largura de Banda	32
	34
4. Cache de Conteúdo Massivamente Escalável.....	37
4.0 Introdução	37
4.1 Zonas de Cache	37
4.2 Bloqueio de Cache	38
4.3 Cache de Chaves de Hash	39
4.4 Desvio de Cache	40
4.5 Desempenho de Cache	41
4.6 Limpeza de Cache com NGINX Plus	41
4.7 Cache Slicing	42
5. Programabilidade e Automação.....	45
5.0 Introdução	45
5.1 API do NGINX Plus	45
Usando o armazenamento de valores-chave com NGINX	49
Plus 5.3 Estendendo o NGINX com uma linguagem de programação comum	51
5.4 Instalando com Puppet	54
5.5 Instalando com Chef	54
5.6 Instalando com Ansible	55
5.7 Instalando com SaltStack	56
5.8 Automatizando configurações com Consul Templating	56
	58
	59
6. Autenticação.....	61
6.0 Introdução	61
6.1 Autenticação básica HTTP	61
6.2 Subsolicitações de autenticação	63
Validando JWTs com NGINX Plus	64
Criando chaves Web JSON	65
6.5 Validando tokens Web JSON com NGINX Plus	66
6.6 Obtendo e armazenando em cache conjuntos de chaves Web JSON com NGINX Mais	66
	67
6.7 Autenticar usuários por meio de SSO OpenID Connect existente com NGINX Plus	68

7. Controles de Segurança.....	71
7.0 Introdução 7.1	71
Acesso baseado no endereço IP 7.2	71
Permitindo compartilhamento de recursos entre origens 7.3 Criptografia do lado do cliente 7.4	72
Criptografia avançada do lado do cliente 7.5	74
Criptografia upstream 7.6 Protegendo um local 7.7	75
Gerando um link seguro com um segredo 7.8	77
Protegendo um local com uma data de expiração 7.9	77
Gerando um Link Expirando 80	79
7.10 Redirecionamentos HTTPS 82	
7.11 Redirecionando para HTTPS onde o SSL/TLS é encerrado antes do NGINX 82	83
7.12 Segurança de transporte restrito HTTP	84
7.13 Satisfazendo qualquer número de métodos de segurança 7.14 Mitigação de DDoS da camada de aplicativo dinâmica	85
NGINX Plus 7.15 Instalando e configurando o módulo de proteção de aplicativo NGINX Plus	86
8. HTTP/2.....	91
8.0 Introdução 8.1	91
Configuração básica 8.2	91
gRPC 8.3 HTTP/2 Server	92
Push	94
9. Streaming de mídia sofisticado.....	97
9.0 Introdução 9.1	97
Servindo MP4 e FLV 9.2	97
Streaming com HLS com NGINX Plus 9.3 Streaming com HDS com NGINX Plus 9.4 Limites de largura de banda com NGINX Plus	98
	99
	100
10. Implantações em Nuvem.....	101
10.0 Introdução 10.1	101
Provisionamento automático no AWS	101
10.2 Roteamento para nós NGINX sem um AWS ELB 10.3 O NLB Sandwich 10.4 Implantação do AWS Marketplace	103
10.5	104
Criando uma imagem de máquina virtual NGINX no Azure 10.6	106
Balanceamento de carga sobre conjuntos de escala NGINX no Azure 10.7 Implantação por meio do Azure Marketplace 10.8 Implantando no Google Compute Engine	107
	109
	109
	110

10.9 Criando uma imagem do Google Compute	111
10.10 Criando um proxy do Google App Engine	112
11. Contêineres/Microsserviços.	115
11.0 Introdução	115
11.1 Usando o NGINX como um API Gateway	116
11.2 Usando registros SRV DNS com NGINX Plus 11.3	120
Usando a imagem oficial do NGINX 11.4 Criando um	121
Dockerfile NGINX 11.5 Criando uma imagem Docker	122
NGINX Plus 11.6 Usando variáveis de ambiente no NGINX	124
11.7 Kubernetes Ingress Controller 11.8 Prometheus	126
Exporter Module	127
	129
12. Modos de implantação de alta disponibilidade.	131
12.0 Introdução	131
Modo NGINX Plus HA	131
12.2 Balanceamento de carga Balanceadores de carga com	132
DNS 12.3 Balanceamento de carga no EC2 12.4 Sincronização	132
de configuração do NGINX Plus 12.5 Compartilhamento de	133
estado com NGINX Plus e sincronização de zona	136
13. Monitoramento Avançado de Atividades.	139
13.0 Introdução	139
Habilitar o status de stub de código aberto do NGINX	139
13.2 Habilitar o painel de monitoramento do NGINX Plus	140
Coletar métricas usando a API do NGINX Plus	143
14. Depuração e solução de problemas com logs de acesso, logs de erros e rastreamento de solicitações.	147
14.0 Introdução	147
Configurando logs de acesso	147
Configurando logs de erro	149
Encaminhamento para Syslog	150
Rastreamento de solicitação	151
OpenTracing para NGINX	152
15. Ajuste de desempenho.	155
15.0 Introdução	155
15.1 Automatizando Testes com Drivers de Carga	155
15.2 Mantendo Conexões Abertas para Clientes	156
15.3 Mantendo Conexões Abertas Upstream	157

15.4 Buffer de respostas 15.5	158
Buffer de logs de acesso 15.6	159
Ajuste do SO	159
16. Introdução ao Controlador NGINX.....	161
16.0 Introdução 16.1	161
Visão geral da configuração	161
16.2 Conectando o NGINX Plus com o controlador 16.3	163
Conduzindo o controlador NGINX com a API 16.4 Habilitar	164
WAF por meio do aplicativo Controller Security	165
17. Dicas de Operações Práticas e Conclusão.....	169
17.0 Introdução 169	
17.1 Usando inclusões para configurações limpas	169
17.2 Depurando configurações	170
Conclusão.....	173
Índice.....	175

Prefácio

Bem-vindo à edição 2020 do NGINX Cookbook. O'Reilly publica o NGINX Cookbook há quase cinco anos e muita coisa mudou. No entanto, uma coisa não mudou: a cada dia, mais e mais sites do mundo optam por rodar no NGINX. Hoje existem mais de 445 milhões, quase o dobro do número de quando o livro de receitas foi lançado – e milhões a mais do que quando o NGINX foi lançado.

Desde que escrevi a primeira versão do NGINX em 2002, ele cresceu e se tornou o balanceador de carga, servidor web, proxy reverso e gateway de API de escolha para muitos sites e organizações – e ainda está crescendo. O NGINX é um canivete suíço: pode ser usado em praticamente qualquer cenário de plano de dados, inclusive como cache de conteúdo, firewall de aplicativo da Web (WAF) e rede de entrega de conteúdo (CDN). Sem mencionar o fato de que é rápido e confiável.

O NGINX Cookbook mostra como tirar o máximo proveito do NGINX, seja NGINX Open Source ou NGINX Plus. Você encontrará mais de 170 páginas de receitas fáceis de seguir sobre como instalar o NGINX, como configurá-lo para praticamente qualquer caso de uso, além de depuração e solução de problemas.

Esta versão inclui atualizações para muitas seções para cobrir novas funcionalidades no NGINX, bem como seções e capítulos inteiramente novos. Com a introdução do NGINX App Protect, expandimos a segurança no NGINX, juntamente com a cobertura expandida do NGINX Controller em paralelo aos novos recursos impressionantes introduzidos no NGINX Controller em 2020 (com mais por vir).

Espero que você goste do livro de receitas do NGINX e que ele contribua para o seu sucesso na criação e implantação de aplicativos com o NGINX, cumprindo a visão que eu tinha anos atrás.

—Igor Sysoev,
Autor e Fundador do NGINX

Prefácio

O NGINX Cookbook visa fornecer exemplos fáceis de seguir para problemas do mundo real na entrega de aplicativos. Ao longo deste livro, você explorará os muitos recursos do NGINX e como usá-los. Este guia é bastante abrangente e aborda a maioria dos principais recursos do NGINX.

Ao longo deste livro, haverá referências ao software NGINX gratuito e de código aberto, bem como aos produtos comerciais da NGINX, Inc., NGINX Plus e NGINX Controller. Recursos e diretivas que estão disponíveis apenas como parte da assinatura paga do NGINX Plus serão indicados como tal. Como o NGINX Plus é um controlador de entrega de aplicativos (ADC) e fornece muitos recursos avançados, é importante destacar esses recursos para obter uma visão completa das possibilidades da plataforma.

O livro começará explicando o processo de instalação do NGINX e do NGINX Plus, bem como algumas etapas básicas de introdução para leitores novos no NGINX. A partir daí, as seções irão progredir para平衡amento de carga em todas as formas, acompanhadas de capítulos sobre gerenciamento de tráfego, cache e automação. O capítulo de segurança cobre muito, mas é importante, porque o NGINX geralmente é o primeiro ponto de entrada para o tráfego da Web para seu aplicativo e a primeira linha de defesa da camada de aplicativo. Há vários capítulos que cobrem tópicos de ponta, como HTTP/2, websockets, streaming de mídia, ambientes de nuvem e contêineres – encerrando com tópicos operacionais mais tradicionais, como monitoramento, depuração, desempenho e dicas operacionais. . O livro terminará com uma introdução ao NGINX Controller, uma plataforma de gerenciamento centrada em aplicativos.

Pessoalmente, uso o NGINX como uma multiferramenta e acredito que este livro permitirá que você faça o mesmo. É um software em que acredito e gosto de trabalhar. Fico feliz em compartilhar esse conhecimento com você e espero que, ao ler este livro, você relate as receitas com seus cenários do mundo real e empregue essas soluções.

Convenções utilizadas neste livro

As seguintes convenções tipográficas são usadas neste livro:

Itálico

Indica novos termos, URLs, endereços de e-mail, nomes de arquivo e extensões de arquivo.

Largura constante

Usado para listagens de programas, bem como dentro de parágrafos para se referir a elementos de programa, como nomes de variáveis ou funções, bancos de dados, tipos de dados, variáveis de ambiente, instruções e palavras-chave.



Este elemento significa uma nota geral.



Este elemento indica um aviso ou cuidado.

Usando exemplos de código

Se você tiver uma pergunta técnica ou um problema ao usar os exemplos de código, envie um e-mail para bookquestions@oreilly.com.

Este livro está aqui para ajudá-lo a fazer o seu trabalho. Em geral, se um código de exemplo for oferecido com este livro, você poderá usá-lo em seus programas e documentação. Você não precisa entrar em contato conosco para obter permissão, a menos que esteja reproduzindo uma parte significativa do código. Por exemplo, escrever um programa que usa vários pedaços de código deste livro não requer permissão. Vender ou distribuir exemplos de livros da O'Reilly requer permissão. Responder a uma pergunta citando este livro e citando um código de exemplo não requer permissão. Incorporar uma quantidade significativa de código de exemplo deste livro na documentação do seu produto requer permissão.

Apreciamos, mas geralmente não exigimos, atribuição. Uma atribuição geralmente inclui o título, autor, editora e ISBN. Por exemplo: "NGINX Cookbook por Derek DeJonghe (O'Reilly). Copyright 2021 O'Reilly Media, Inc., 978-1-492-07848-7."

Se você achar que o uso de exemplos de código está fora do uso justo ou da permissão dada acima, sinta-se à vontade para nos contatar em permissions@oreilly.com.

Aprendizagem on-line O'Reilly



Por mais de 40 anos, a **O'Reilly Media** forneceu treinamento em tecnologia e negócios, conhecimento e percepção para ajudar as empresas a serem bem-sucedidas.

Nossa rede exclusiva de especialistas e inovadores compartilha seu conhecimento e experiência por meio de livros, artigos e nossa plataforma de aprendizado online. A plataforma de aprendizado on-line da O'Reilly oferece acesso sob demanda a cursos de treinamento ao vivo, caminhos de aprendizado aprofundados, ambientes de codificação interativos e uma vasta coleção de texto e vídeo da O'Reilly e mais de 200 outras editoras. Para obter mais informações, visite <http://oreilly.com>.

Como entrar em contato conosco

Envie comentários e perguntas sobre este livro à editora:

O'Reilly Media, Inc.
1005 Gravenstein Highway North
Sebastopol, CA 95472 800-998-9938
(nos Estados Unidos ou Canadá) 707-829-0515
(internacional ou local) 707-829-0104 (fax)

Temos uma página web para este livro, onde listamos erratas, exemplos e qualquer informação adicional. Você pode acessar esta página em <https://oreil.ly/NGINX-cookbook>.

E-mail bookquestions@oreilly.com para comentar ou fazer perguntas técnicas sobre este livro.

Para notícias e informações sobre nossos livros e cursos, visite <http://oreilly.com>.

Encontre-nos no Facebook: <http://facebook.com/oreilly>

Siga-nos no Twitter: <http://twitter.com/oreillymedia>

Assista-nos no YouTube: <http://www.youtube.com/oreillymedia>

Agradecimentos

Gostaria de agradecer a Hussein Nasser e Gonzalo Josue Spina por suas revisões úteis e detalhadas deste título. Também gostaria de agradecer à minha esposa por seu apoio durante o processo de escrita.

CAPÍTULO 1

Fundamentos

1.0 introdução

Para começar com o NGINX Open Source ou NGINX Plus, primeiro você precisa instalá-lo em um sistema e aprender alguns conceitos básicos. Neste capítulo, você aprenderá como instalar o NGINX, onde estão os principais arquivos de configuração e comandos para administração. Você também aprenderá como verificar sua instalação e fazer solicitações ao padrão servidor.

1.1 Instalando no Debian/Ubuntu

Problema

Você precisa instalar o NGINX Open Source em uma máquina Debian ou Ubuntu.

Solução

Crie um arquivo chamado /etc/apt/sources.list.d/nginx.list que contém o seguinte conteúdo:

```
deb http://nginx.org/packages/mainline/OS/ CODENAME nginx deb-
src http://nginx.org/packages/mainline/OS/ CODENAME nginx
```

Altere o arquivo, substituindo o SO no final da URL por ubuntu ou debian, dependendo da sua distribuição. Substitua CODENAME pelo codinome da sua distribuição; jessie ou stretch para o Debian, ou trusty, xenial, artful ou bionic para o Ubuntu. Em seguida, execute os seguintes comandos:

```
wget http://nginx.org/keys/nginx_signing.key apt-key
adicionar nginx_signing.key apt-get update
```

```
apt-get install -y nginx /etc/  
init.d/nginx start
```

Discussão

O arquivo que você acabou de criar instrui o sistema de gerenciamento de pacotes da ferramenta avançada de pacotes (APT) a utilizar o repositório oficial de pacotes NGINX. Modificar o arquivo para fornecer o endpoint e o nome de código corretos para sua distribuição garante que o utilitário APT receba os pacotes .deb corretos para seu sistema. Os comandos a seguir baixam a chave de assinatura do pacote NGINX GPG e a importam para o APT. Fornecer ao APT a chave de assinatura permite que o sistema APT valide pacotes do repositório. O comando apt-get update instrui o sistema APT a atualizar suas listagens de pacotes de seus repositórios conhecidos. Depois que a lista de pacotes for atualizada, você poderá instalar o NGINX Open Source do repositório oficial do NGINX. Depois de instalá-lo, o comando final inicia o NGINX.

1.2 Instalando no RedHat/CentOS

Problema

Você precisa instalar o NGINX Open Source no RedHat ou CentOS.

Solução

Crie um arquivo chamado /etc/yum.repos.d/nginx.repo que contenha o seguinte conteúdo:

```
[nginx]  
name=nginx repo  
baseurl=http://nginx.org/packages/mainline/OS/OSRELEASE/$basearch/ gpgcheck=0  
enabled=1
```

Altere o arquivo, substituindo OS no final da URL por rhel ou centos, dependendo da sua distribuição.

Substitua OSRELEASE por 6 ou 7 para a versão 6.x ou 7.x, respectivamente.

Em seguida, execute os seguintes comandos:

```
yum -y install nginx  
systemctl enable nginx  
systemctl start nginx firewall-  
cmd --permanent --zone=public --add-port=80/tcp firewall-cmd --reload
```

Discussão

O arquivo que você acabou de criar para esta solução instrui o sistema de gerenciamento de pacotes YUM a utilizar o repositório oficial de pacotes NGINX Open Source. Os comandos a seguir instalaram o NGINX Open Source do repositório Oficial, instrui

systemd para habilitar o NGINX no momento da inicialização e diga para iniciá-lo agora. O firewall comanda a porta aberta 80 para o protocolo TCP, que é a porta padrão para HTTP. O último comando recarrega o firewall para confirmar as alterações.

1.3 Instalando o NGINX Plus

Problema

Você precisa instalar o NGINX Plus.

Solução

Visite http://cs.nginx.com/repo_setup. No menu suspenso, selecione o SO que você está instalando e siga as instruções. As instruções são semelhantes à instalação das soluções open source; no entanto, você precisa instalar um certificado para autenticar no repositório NGINX Plus.

Discussão

O NGINX mantém este guia de instalação do repositório atualizado com instruções sobre como instalar o NGINX Plus. Dependendo do seu sistema operacional e versão, essas instruções variam um pouco, mas há uma semelhança. Você deve obter um certificado e uma chave do portal NGINX e fornecê-los ao seu sistema para autenticar no repositório NGINX Plus.

1.4 Verificando sua instalação

Problema

Você deseja validar a instalação do NGINX e verificar a versão.

Solução

Você pode verificar se o NGINX está instalado e verificar sua versão usando o seguinte comando:

```
$ nginx -v  
versão do nginx: nginx/1.15.3
```

Como este exemplo mostra, a resposta exibe a versão.

Você pode confirmar que o NGINX está em execução usando o seguinte comando:

```
$ ps -ef | raiz grep nginx  
1738      1 0 19:54 ? 00:00:00 nginx: processo mestre 1739 1738  
nginx      0 19:54 ? 00:00:00 nginx: processo de trabalho
```

O comando ps lista os processos em execução. Ao canalizá-lo para grep, você pode pesquisar palavras específicas na saída. Este exemplo usa grep para procurar por nginx. O resultado mostra dois processos em execução, um mestre e um trabalhador. Se o NGINX estiver em execução, você sempre verá um mestre e um ou mais processos de trabalho. Observe que o processo mestre está sendo executado como root, pois o NGINX precisa de privilégios elevados para funcionar corretamente. Para obter instruções sobre como iniciar o NGINX, consulte a próxima seção. Para ver como iniciar o NGINX como um daemon, use as metodologias init.d ou systemd.

Para verificar se o NGINX está retornando solicitações corretamente, use seu navegador para fazer uma solicitação à sua máquina ou use curl. Ao fazer a solicitação, use o endereço IP ou o nome do host da máquina. Se instalado localmente, você pode usar localhost da seguinte forma:

```
curl localhost
```

Você verá o site HTML padrão de boas-vindas do NGINX.

Discussão

O comando nginx permite interagir com o binário NGINX para verificar a versão, listar módulos instalados, testar configurações e enviar sinais para o processo mestre. O NGINX deve estar em execução para atender às solicitações. O comando ps é uma maneira infalível de determinar se o NGINX está sendo executado como um daemon ou em primeiro plano. A configuração fornecida por padrão com o NGINX executa um servidor HTTP de site estático na porta 80. Você pode testar esse site padrão fazendo uma solicitação HTTP para a máquina em localhost , bem como o IP e o nome do host do host.

1.5 Arquivos de chave, diretórios e comandos

Problema

Você precisa entender os diretórios e comandos importantes do NGINX.

Solução

Arquivos e diretórios NGINX

/etc/nginx/

O diretório /etc/nginx/ é a raiz de configuração padrão para o servidor NGINX.

Dentro deste diretório você encontrará arquivos de configuração que instruem o NGINX sobre como se comportar.

/etc/nginx/nginx.conf

O arquivo /etc/nginx/nginx.conf é o ponto de entrada de configuração padrão usado pelo serviço NGINX. Este arquivo de configuração define configurações globais para coisas como processo de trabalho, ajuste, registro, carregamento de módulos dinâmicos e referências a

outros arquivos de configuração do NGINX. Em uma configuração padrão, o arquivo /etc/nginx/nginx.conf inclui o bloco http de nível superior , ou contexto, que inclui todos os arquivos de configuração no diretório descrito a seguir.

/etc/nginx/conf.d/ O

diretório /etc/nginx/conf.d/ contém o arquivo de configuração do servidor HTTP padrão. Os arquivos neste diretório que terminam em .conf são incluídos no bloco http de nível superior de dentro do arquivo /etc/nginx/nginx.conf. É uma prática recomendada utilizar instruções de inclusão e organizar sua configuração dessa maneira para manter seus arquivos de configuração concisos. Em alguns repositórios de pacotes, essa pasta é denominada sites-enabled e os arquivos de configuração são vinculados a partir de uma pasta denominada site available; esta convenção está obsoleta.

/var/log/nginx/ O

diretório /var/log/nginx/ é o local de log padrão para NGINX. Dentro deste diretório você encontrará um arquivo access.log e um arquivo error.log. O log de acesso contém uma entrada para cada solicitação que o NGINX atende. O arquivo de log de erros contém eventos de erro e informações de depuração se o módulo de depuração estiver ativado .

Comandos NGINX

nginx -h

Mostra o menu de ajuda do NGINX.

nginx -v

Mostra a versão do NGINX.

nginx -V

Mostra a versão do NGINX, informações de compilação e argumentos de configuração, que mostra os módulos integrados ao binário NGINX.

nginx -t

Testa a configuração do NGINX.

nginx -T

Testa a configuração do NGINX e imprime a configuração validada na tela. Este comando é útil ao buscar suporte.

nginx -s signal O

sinalizador -s envia um sinal para o processo mestre do NGINX. Você pode enviar sinais como parar, sair, recarregar e reabrir. O sinal de parada interrompe o processo NGINX imediatamente. O sinal de saída interrompe o processo do NGINX após concluir o processamento de solicitações em andamento. O sinal de recarga recarrega a configuração. O sinal de reabertura instrui o NGINX a reabrir os arquivos de log.

Discussão

Com uma compreensão desses arquivos, diretórios e comandos principais, você está em uma boa posição para começar a trabalhar com o NGINX. Com esse conhecimento, você pode alterar os arquivos de configuração padrão e testar suas alterações usando o comando `nginx -t`. Se seu teste for bem sucedido, você também sabe como instruir o NGINX a recarregar sua configuração usando o comando `nginx -s reload`.

1.6 Servindo conteúdo estático

Problema

Você precisa fornecer conteúdo estático com NGINX.

Solução

Substitua a configuração padrão do servidor HTTP localizada em `/etc/nginx/conf.d/default.conf` com o seguinte exemplo de configuração NGINX:

```
servidor
{
    escuta 80 default_server;
    nome_do_servidor www.exemplo.com;

    local / {
        root /usr/share/nginx/html; #
        alias /usr/share/nginx/html; índice
        índice.html índice.htm;
    }
}
```

Discussão

Essa configuração serve arquivos estáticos por HTTP na porta 80 do diretório `/usr/share/nginx/html/`. A primeira linha nesta configuração define um novo bloco de servidor. Isso define um novo contexto para o NGINX escutar. A linha dois instrui o NGINX a listar ten na porta 80, e o parâmetro `default_server` instrui o NGINX a usar este servidor como o contexto padrão para a porta 80. A diretiva `listen` também pode receber um intervalo de portas. A diretiva `server_name` define o hostname ou nomes dos quais os pedidos devem ser direcionados para este servidor. Se a configuração não tivesse definido esse contexto como o `default_server`, o NGINX direcionaria solicitações para esse servidor somente se o cabeçalho do host HTTP corresponesse ao valor fornecido à diretiva `server_name`. Com o contexto `default_server` definido, você pode omitir a diretiva `server_name` se ainda não tiver um nome de domínio para usar.

O bloco de localização define uma configuração com base no caminho na URL. O caminho, ou parte da URL após o domínio, é chamado de identificador de recurso uniforme (URI). O NGINX corresponderá melhor ao URI solicitado a um bloco de localização . O exemplo usa / para corresponder a todas as solicitações. A diretiva root mostra ao NGINX onde procurar arquivos estáticos ao fornecer conteúdo para o contexto fornecido. O URI da solicitação é anexado ao valor da diretiva raiz ao procurar o arquivo solicitado. Se tivéssemos fornecido um prefixo de URI para a diretiva de localização , isso seria incluído no caminho anexado, a menos que usássemos a diretiva de alias em vez de raiz. A diretiva de localização é capaz de corresponder a uma ampla variedade de expressões. Visite o link na seção Veja também para obter mais informações. Por fim, a diretiva index fornece ao NGINX um arquivo padrão ou uma lista de arquivos a serem verificados, caso nenhum caminho adicional seja fornecido no URI.

Veja também

[Documentação da Diretiva de Localização HTTP NGINX](#)

1.7 Recarga Graciosa

Problema

Você precisa recarregar sua configuração sem descartar pacotes.

Solução

Use o método reload do NGINX para obter um recarregamento normal da configuração sem interromper o servidor:

```
nginx -s recarregar
```

Este exemplo recarrega o sistema NGINX usando o binário NGINX para enviar um sinal ao processo mestre.

Discussão

Recarregar a configuração do NGINX sem interromper o servidor oferece a capacidade de alterar as configurações em tempo real sem descartar nenhum pacote. Em um ambiente dinâmico de alto tempo de atividade, você precisará alterar sua configuração de平衡amento de carga em algum momento. O NGINX permite que você faça isso enquanto mantém o balanceador de carga online. Esse recurso permite inúmeras possibilidades, como executar novamente o gerenciamento de configuração em um ambiente ativo ou criar um módulo com reconhecimento de aplicativo e cluster para configurar e recarregar dinamicamente o NGINX para atender às necessidades do ambiente.

CAPÍTULO 2

Balanceamento de carga de alto desempenho

2.0 Introdução

A experiência atual do usuário da Internet exige desempenho e tempo de atividade. Para conseguir isso, várias cópias do mesmo sistema são executadas e a carga é distribuída sobre elas. À medida que a carga aumenta, outra cópia do sistema pode ser colocada online. Essa técnica de arquitetura é chamada de escala horizontal. A infraestrutura baseada em software está aumentando em popularidade devido à sua flexibilidade, abrindo um vasto mundo de possibilidades.

Quer o caso de uso seja tão pequeno quanto um conjunto de dois para alta disponibilidade ou tão grande quanto milhares em todo o mundo, há uma necessidade de uma solução de balanceamento de carga que seja tão dinâmica quanto a infraestrutura. O NGINX atende a essa necessidade de várias maneiras, como平衡amento de carga HTTP, TCP e protocolo de datagrama do usuário (UDP), que abordamos neste capítulo.

Ao equilibrar a carga, é importante que o impacto na experiência do cliente seja totalmente positivo. Muitas arquiteturas da Web modernas empregam camadas de aplicativos sem estado, armazenando o estado em memória compartilhada ou bancos de dados. No entanto, esta não é a realidade para todos. O estado da sessão é imensamente valioso e vasto em aplicativos interativos. Esse estado pode ser armazenado localmente no servidor de aplicativos por vários motivos; por exemplo, em aplicativos para os quais os dados que estão sendo trabalhados são tão grandes que a sobrecarga da rede é muito cara em termos de desempenho. Quando o estado é armazenado localmente em um servidor de aplicativos, é extremamente importante para a experiência do usuário que as solicitações subsequentes continuem sendo entregues ao mesmo servidor. Outra faceta da situação é que os servidores não devem ser liberados até que a sessão termine. Trabalhar com aplicativos com estado em escala requer um平衡ador de carga inteligente. O NGINX Plus oferece várias maneiras de resolver esse problema rastreando cookies ou roteamento. Este capítulo aborda a persistência de sessão no que se refere ao平衡amento de carga com NGINX e NGINX Plus.

É importante garantir que o aplicativo que o NGINX está atendendo seja íntegro. Por vários motivos, as solicitações upstream podem começar a falhar. Pode ser devido à conectividade de rede, falha de servidor ou falha de aplicativo, para citar alguns. Proxies e balanceadores de carga devem ser inteligentes o suficiente para detectar falhas de servidores upstream (servidores atrás do balanceador de carga ou proxy) e parar de passar tráfego para eles; caso contrário, o cliente estará esperando, apenas para ser entregue um timeout. Uma maneira de mitigar a degradação do serviço quando um servidor falha é fazer com que o proxy verifique a integridade dos servidores upstream. O NGINX oferece dois tipos diferentes de verificações de integridade: passiva, disponível na versão de código aberto; e ativo, disponível apenas no NGINX Plus. As verificações de integridade ativas em intervalos regulares farão uma conexão ou solicitação ao servidor upstream e poderão verificar se a resposta está correta. As verificações de integridade passivas monitoram a conexão ou as respostas do servidor upstream à medida que os clientes fazem a solicitação ou a conexão. Você pode querer usar verificações de integridade passivas para reduzir a carga de seus servidores upstream e pode querer usar verificações de integridade ativas para determinar a falha de um servidor upstream antes que um cliente receba uma falha. O final deste capítulo examina o monitoramento da integridade dos servidores de aplicativos upstream para os quais você está fazendo o平衡amento de carga.

2.1 Balanceamento de carga HTTP

Problema

Você precisa distribuir a carga entre dois ou mais servidores HTTP.

Solução

Use o módulo HTTP do NGINX para balancear a carga em servidores HTTP usando o bloco upstream :

```
back-end upstream {
    servidor 10.10.12.45:80      peso=1;
    servidor app.example.com:80 peso=2; server
    spare.example.com:80 backup;

} servidor
{ local {
    proxy_pass http://backend;
}
}
```

Essa configuração equilibra a carga em dois servidores HTTP na porta 80 e define um como um backup que é usado quando os dois principais não estão disponíveis. O parâmetro de peso instrui o NGINX a passar duas vezes mais solicitações para o segundo servidor e o parâmetro de peso é padronizado como 1.

Discussão

O módulo upstream HTTP controla o balanceamento de carga para HTTP. Este módulo define um conjunto de destinos — qualquer combinação de soquetes Unix, endereços IP e registros DNS, ou uma combinação. O módulo upstream também define como qualquer solicitação individual é atribuída a qualquer um dos servidores upstream.

Cada destino upstream é definido no pool upstream pela diretiva do servidor .

A diretiva do servidor é fornecida com um soquete Unix, endereço IP ou um nome de domínio totalmente qualificado (FQDN), juntamente com vários parâmetros opcionais. Os parâmetros opcionais dão mais controle sobre o roteamento de solicitações. Esses parâmetros incluem o peso do servidor no algoritmo de平衡amento; se o servidor está em modo de espera, disponível ou indisponível; e como determinar se o servidor está indisponível. O NGINX Plus fornece vários outros parâmetros convenientes, como limites de conexão ao servidor, controle avançado de resolução de DNS e a capacidade de aumentar lentamente as conexões a um servidor após o início.

2.2 Balanceamento de carga TCP

Problema

Você precisa distribuir a carga entre dois ou mais servidores TCP.

Solução

Use o módulo stream do NGINX para balancear a carga em servidores TCP usando o bloco upstream :

```
fluxo
{ upstream mysql_read {
    servidor read1.example.com:3306 peso=5;
    servidor read2.example.com:3306; servidor
    10.10.12.34:3306
        cópia de segurança;
}
servidor
{ escuta 3306;
proxy_pass mysql_read;
}
}
```

O bloco do servidor neste exemplo instrui o NGINX a escutar na porta TCP 3306 e balancear a carga entre duas réplicas de leitura do banco de dados MySQL e lista outra como um backup que receberá o tráfego se as primárias estiverem inativas.

Essa configuração não deve ser adicionada à pasta conf.d, pois essa pasta está incluída em um bloco http ; em vez disso, você deve criar outra pasta chamada stream.conf.d,

abra o bloco de stream no arquivo nginx.conf e inclua a nova pasta para configurações de stream. Segue um exemplo: No arquivo de configuração /etc/nginx/nginx.conf:

```

usuário
nginx; trabalhador_processos
auto; pid /run/nginx.pid;

fluxo
{ include /etc/nginx/stream.conf.d/*.conf;
}

```

Em um arquivo chamado /etc/nginx/stream.conf.d/mysql_reads.conf arquivo de configuração:

```

upstream mysql_read {
    servidor read1.example.com:3306 peso=5;
    servidor read2.example.com:3306; servidor
    10.10.12.34:3306
}

servidor
{ escuta 3306;
proxy_pass mysql_read;
}

```

Discussão

A principal diferença entre o contexto http e stream é que eles operam em diferentes camadas do modelo OSI. O contexto http opera na camada de aplicação, 7, e o fluxo opera na camada de transporte, 4. Isso não significa que o contexto de fluxo não possa se tornar sensível ao aplicativo com algum script inteligente, mas que o contexto http é projetado especificamente para entender completamente o protocolo HTTP e o contexto de fluxo , por padrão, simplesmente roteiam e equilibram a carga dos pacotes.

O balanceamento de carga TCP é definido pelo módulo de fluxo NGINX . O módulo stream , como o módulo HTTP , permite definir pools upstream de servidores e configurar um servidor de escuta. Ao configurar um servidor para escutar em uma determinada porta, você deve definir a porta em que ele escutará ou, opcionalmente, um endereço e uma porta. A partir daí, um destino deve ser configurado, seja um proxy reverso direto para outro endereço ou um pool upstream de recursos.

Várias opções que alteram as propriedades do proxy reverso da conexão TCP estão disponíveis para configuração. Algumas delas incluem limitações de validação SSL/TLS, tempos limite e keepalives. Alguns dos valores dessas opções de proxy podem ser (ou conter) variáveis, como a taxa de download e o nome usado para verificar um certificado SSL/TLS.

O upstream para balanceamento de carga TCP é muito parecido com o upstream para HTTP, pois define recursos upstream como servidores, configurados com soquete Unix, IP ou FQDN, bem como peso do servidor, número máximo de conexões, resolvedores de DNS e conexões, períodos de aceleração; e se o servidor está ativo, inativo ou em modo de backup.

O NGINX Plus oferece ainda mais recursos para balanceamento de carga TCP. Esses recursos avançados oferecidos no NGINX Plus podem ser encontrados em todo este livro. As verificações de integridade de todo o balanceamento de carga serão abordadas posteriormente neste capítulo.

2.3 Balanceamento de carga UDP

Problema

Você precisa distribuir a carga entre dois ou mais servidores UDP.

Solução

Use o módulo stream do NGINX para balancear a carga em servidores UDP usando o bloco upstream definido como udp:

```
fluxo
{
    upstream ntp
        { servidor ntp1.example.com:123 weight=2;
          servidor ntp2.example.com:123;
        }
    servidor
        { escuta 123 udp;
          proxy_pass ntp;
        }
}
```

Esta seção de configuração equilibra a carga entre dois servidores de protocolo de tempo de rede (NTP) upstream usando o protocolo UDP. Especificar o balanceamento de carga UDP é tão simples quanto usar o parâmetro udp na diretiva listen .

Se o serviço sobre o qual você está fazendo o balanceamento de carga exigir que vários pacotes sejam enviados entre cliente e servidor, você poderá especificar o parâmetro reutilizaport .

Exemplos desses tipos de serviços são OpenVPN, Voice over Internet Protocol (VoIP), soluções de desktop virtual e Datagram Transport Layer Security (DTLS).

Veja a seguir um exemplo de uso do NGINX para lidar com conexões OpenVPN e fazer proxy delas para o serviço OpenVPN em execução localmente:

```
fluxo
{
    server
        { escuta 1195 udp reutilização;
          proxy_pass 127.0.0.1:1194;
        }
}
```

```

    }
}

```

Discussão

Você pode perguntar: "Por que preciso de um balanceador de carga quando posso ter vários hosts em um DNS A ou registro de serviço (registro SRV)?" A resposta é que não apenas existem algoritmos de平衡amento alternativos com os quais podemos balancear, mas podemos balancear a carga nos próprios servidores DNS. Os serviços UDP compõem muitos dos serviços dos quais dependemos em sistemas de rede, como DNS, NTP, QUIC, HTTP/3 e VoIP. O balanceamento de carga UDP pode ser menos comum para alguns, mas igualmente útil no mundo da escala.

Você pode encontrar o balanceamento de carga UDP no módulo de fluxo , assim como o TCP, e configurá-lo principalmente da mesma maneira. A principal diferença é que a diretiva listen especifica que o soquete aberto é para trabalhar com datagramas. Ao trabalhar com datagramas, existem algumas outras diretivas que podem ser aplicadas onde não seriam no TCP, como a diretiva proxy_response , que especifica ao NGINX quantas respostas esperadas podem ser enviadas do servidor upstream. Por padrão, isso é ilimitado até que o limite proxy_timeout seja atingido. A diretiva proxy_timeout define o tempo entre duas operações de leitura ou gravação sucessivas em conexões de cliente ou servidor proxy antes que a conexão seja fechada.

O parâmetro de reutilização instrui o NGINX a criar um soquete de escuta individual para cada processo de trabalho. Isso permite que o kernel distribua conexões de entrada entre processos de trabalho para lidar com vários pacotes enviados entre cliente e servidor. O recurso de porta de reutilização funciona apenas em kernels Linux 3.9 e superior, DragonFly BSD e FreeBSD 12 e superior.

2.4 Métodos de balanceamento de carga

Problema

O balanceamento de carga round-robin não se adequa ao seu caso de uso porque você tem cargas de trabalho heterogêneas ou pools de servidores.

Solução

Use um dos métodos de balanceamento de carga do NGINX, como menos conexões, menos tempo, hash genérico, aleatório ou hash de IP:

```

back-end upstream
{
    less_conn;
    servidor backend.example.com;
    servidor backend1.example.com;
}

```

Este exemplo define o algoritmo de balanceamento de carga para o pool upstream de back-end para ter menos conexões. Todos os algoritmos de balanceamento de carga, com exceção de hash genérico, aleatório e de menor tempo, são diretivas independentes, como o exemplo anterior. Os parâmetros para essas diretivas são explicados na discussão a seguir.

Discussão

Nem todas as solicitações ou pacotes têm o mesmo peso. Diante disso, o round robin, ou mesmo o round robin ponderado usado nos exemplos anteriores, não atenderá à necessidade de todas as aplicações ou fluxo de tráfego. O NGINX fornece vários algoritmos de平衡amento de carga que você pode usar para se adequar a casos de uso específicos. Além de poder escolher esses algoritmos ou métodos de平衡amento de carga, você também pode configurá-los. Os métodos de平衡amento de carga a seguir estão disponíveis para pools de upstream HTTP, TCP e UDP.

Rodízio

Este é o método de平衡amento de carga padrão, que distribui solicitações na ordem da lista de servidores no pool upstream. Você também pode levar em consideração o peso para um round robin ponderado, que pode ser usado se a capacidade dos servidores upstream variar. Quanto maior o valor inteiro para o peso, mais favorecido será o servidor no round robin. O algoritmo por trás do peso é simplesmente a probabilidade estatística de uma média ponderada.

Menos conexões

Esse método equilibra a carga fazendo proxy da solicitação atual para o servidor upstream com o menor número de conexões abertas. Conexões mínimas, como round robin, também levam em consideração os pesos ao decidir para qual servidor enviar a conexão. O nome da diretiva é least_conn.

Menor

tempo Disponível apenas no NGINX Plus, o menor tempo é semelhante ao menor tempo de conexões, pois faz proxy para o servidor upstream com o menor número de conexões atuais, mas favorece os servidores com os menores tempos médios de resposta. Esse método é um dos algoritmos de平衡amento de carga mais sofisticados e atende às necessidades de aplicativos da Web de alto desempenho. Este algoritmo é um valor agregado sobre conexões mínimas porque um pequeno número de conexões não significa necessariamente a resposta mais rápida. Ao usar este algoritmo, é importante levar em consideração a variação estatística dos tempos de solicitação dos serviços. Algumas requisições podem naturalmente levar mais processamento e, portanto, ter um tempo de requisição maior, aumentando o alcance da estatística. Longos tempos de solicitação nem sempre significam um servidor com menos desempenho ou sobrecarregado. No entanto, solicitações que exigem mais processamento podem ser candidatas a fluxos de trabalho assíncronos. Um parâmetro de cabeçalho ou last_byte deve ser especificado para esta diretiva. Quando o cabeçalho é especificado, o tempo para receber o cabeçalho de resposta é usado. Quando last_byte é especificado, o tempo para receber a resposta completa é usado. O nome da diretiva é least_time.

hash genérico

O administrador define um hash com o texto fornecido, variáveis da solicitação ou tempo de execução, ou ambos. O NGINX distribui a carga entre os servidores produzindo um hash para a solicitação atual e colocando-o nos servidores upstream. Esse método é muito útil quando você precisa de mais controle sobre onde as solicitações são enviadas ou para determinar qual servidor upstream provavelmente terá os dados armazenados em cache.

Observe que quando um servidor é adicionado ou removido do pool, as solicitações com hash serão redistribuídas. Este algoritmo possui um parâmetro opcional, consistente, para minimizar o efeito da redistribuição. O nome da diretiva é hash.

Aleatório

Este método é usado para instruir o NGINX a selecionar um servidor aleatório do grupo, levando em consideração os pesos do servidor. O parâmetro opcional dois [método] direciona o NGINX para selecionar aleatoriamente dois servidores e, em seguida, usar o método de平衡amento de carga fornecido para balancear entre esses dois. Por padrão, o método least_conn é usado se dois forem passados sem um método. O nome da diretiva para平衡amento de carga aleatório é aleatório.

hash de IP

Este método funciona apenas para HTTP. O hash IP usa o endereço IP do cliente como o hash. Um pouco diferente de usar a variável remota em um hash genérico, esse algoritmo usa os três primeiros octetos de um endereço IPv4 ou todo o endereço IPv6. Esse garante que os clientes sejam encaminhados para o mesmo servidor upstream enquanto esse servidor estiver disponível, o que é extremamente útil quando o estado da sessão é preocupante e não é tratado pela memória compartilhada do aplicativo. Esse método também leva em consideração o parâmetro de peso ao distribuir o hash. O nome da diretiva é ip_hash.

2.5 Sticky Cookie com NGINX Plus

Problema

Você precisa vincular um cliente downstream a um servidor upstream usando o NGINX Plus.

Solução

Use a diretiva sticky cookie para instruir o NGINX Plus a criar e rastrear um cookie:

```
backend upstream
  { server backend1.example.com;
    servidor backend2.example.com;
    A afinidade de sticky cookie
      expira=1h
      domínio=.exemplo.com
```

```

httpsomente
seguro
caminho=/;
}

```

Essa configuração cria e rastreia um cookie que vincula um cliente downstream a um servidor upstream. Neste exemplo, o cookie é denominado afinidade, é definido para example.com, expira em uma hora, não pode ser consumido no lado do cliente, pode ser enviado apenas por HTTPS e é válido para todos os caminhos.

Discussão

Usar o parâmetro cookie na diretiva sticky cria um cookie na primeira solicitação que contém informações sobre o servidor upstream. O NGINX Plus rastreia esse cookie, permitindo que ele continue direcionando solicitações subsequentes para o mesmo servidor. O primeiro parâmetro posicional para o parâmetro cookie é o nome do cookie a ser criado e rastreado. Outros parâmetros oferecem controle adicional, informando ao navegador o uso apropriado – como o tempo de expiração, domínio, caminho e se o cookie pode ser consumido no lado do cliente ou se pode ser passado por protocolos não seguros.

2.6 Sticky Learn com NGINX Plus

Problema

Você precisa vincular um cliente downstream a um servidor upstream usando um cookie existente com o NGINX Plus.

Solução

Use a diretiva sticky learn para descobrir e rastrear cookies que são criados pelo aplicativo upstream:

```

back-end upstream {
    servidor backend1.example.com:8080;
    servidor backend2.example.com:8081;

    aprendizado
        fixo criar=$upstream_cookie_cookieName
        lookup=$cookie_cookieName
        zona=client_sessions:2m;
}

```

Este exemplo instrui o NGINX a procurar e rastrear sessões procurando um cookie chamado COOKIE NAME nos cabeçalhos de resposta e procurando sessões existentes procurando o mesmo cookie nos cabeçalhos de solicitação. Essa afinidade de sessão é armazenada em uma zona de memória compartilhada de 2 MB que pode rastrear aproximadamente 16.000 sessões. O nome do

cookie será sempre específico do aplicativo. Nomes de cookies comumente usados, como `jsessionid` ou `phpsessionid`, normalmente são padrões definidos no aplicativo ou na configuração do servidor de aplicativos.

Discussão

Quando os aplicativos criam seus próprios cookies de estado de sessão, o NGINX Plus pode descobri-los nas respostas das solicitações e rastreá-los. Este tipo de rastreamento de cookies é realizado quando a diretiva de aderência é fornecida: o parâmetro `learn`. A memória compartilhada para rastreamento de cookies é especificada com o parâmetro `zone`, com nome e tamanho. O NGINX Plus é direcionado para procurar cookies na resposta do servidor upstream através da especificação do parâmetro `create`, e procura por afinidade de servidor previamente registrado usando o parâmetro `lookup`. O valor desses parâmetros são variáveis expostas por o módulo HTTP .

2.7 Roteamento fixo com NGINX Plus

Problema

Você precisa de controle granular sobre como suas sessões persistentes são roteadas para o servidor upstream com o NGINX Plus.

Solução

Use a diretiva `sticky` com o parâmetro de rota para usar variáveis sobre a solicitação para rotear:

```
map $cookie_jsessionid $route_cookie { ~.+.(?  
    P<route>\w+)\$ $route;  
}  
  
map $request_uri $route_uri  
{ ~jsessionid=.+\.(?P<rota>\w+)\$ $rota;  
}  
  
backend upstream  
{ server backend1.example.com route=a;  
    servidor backend2.example.com rota=b;  
  
    rota fixa $route_cookie $route_uri;  
}
```

Este exemplo tenta extrair um ID de sessão Java, primeiro de um cookie mapeando o valor do cookie de ID de sessão Java para uma variável com o primeiro bloco de mapa e, segundo, procurando no URI de solicitação um parâmetro chamado `jsessionid`, mapeando o valor para uma variável usando o segundo bloco de mapa . A diretiva pegajosa com a rota

parâmetro é passado qualquer número de variáveis. O primeiro valor diferente de zero ou não vazio é usado para a rota. Se um cookie jsessionid for usado, a solicitação será roteada para backend1; se um parâmetro de URI for usado, a solicitação será roteada para backend2. Embora este exemplo seja baseado no ID de sessão comum do Java, o mesmo se aplica a outra tecnologia de sessão como phpsessionid, ou qualquer identificador exclusivo garantido que seu aplicativo gere para o ID de sessão.

Discussão

Às vezes, utilizando um controle um pouco mais granular, você pode querer direcionar o tráfego para um servidor específico. O parâmetro de rota para a diretiva de aderência é criado para atingir esse objetivo. A rota fixa oferece melhor controle, rastreamento real e aderência, em oposição ao algoritmo genérico de平衡amento de carga de hash. O cliente é roteado primeiro para um servidor upstream com base na rota especificada e, em seguida, as solicitações subsequentes transportarão as informações de roteamento em um cookie ou URI. A rota fixa leva vários parâmetros posicionais que são avaliados. A primeira variável não vazia é usada para rotear para um servidor.

Blocos de mapa podem ser usados para analisar seletivamente variáveis e salvá-las como outras variáveis a serem usadas no roteamento. Essencialmente, a diretiva sticky route cria uma sessão dentro da zona de memória compartilhada do NGINX Plus para rastrear qualquer identificador de sessão do cliente que você especificar para o servidor upstream, entregando consistentemente solicitações com esse identificador de sessão para o mesmo servidor upstream que sua solicitação original.

2.8 Drenagem de conexão com NGINX Plus

Problema

Você precisa remover servidores para manutenção ou outros motivos, enquanto ainda atende sessões com o NGINX Plus.

Solução

Use o parâmetro de drenagem por meio da API do NGINX Plus, descrito com mais detalhes em [Capítulo 5](#), para instruir o NGINX a parar de enviar novas conexões que ainda não foram rastreadas:

```
$ curl -X POST -d '{"drenar":true}' \
  'http://nginx.local/api/3/http/upstreams/backend/servers/0'

{
  "id":0,
  "server":"172.17.0.3:80",
  "weight":1, "max_conns":0,
  "maxfails":1, "fail_timeout":
  "10s", "slow_start":
```

```

    "0s",
    "route":"",
    "backup":false,
    "down":false,
    "drain":true
}

```

Discussão

Quando o estado da sessão é armazenado localmente em um servidor, as conexões e sessões persistentes devem ser drenadas antes que o servidor seja removido do pool. A drenagem de conexões é o processo de permitir que as sessões para um servidor expirem nativamente antes de remover o servidor do pool upstream. Você pode configurar a drenagem para um servidor específico adicionando o parâmetro de drenagem à diretiva do servidor . Quando o parâmetro de drenagem é definido, o NGINX Plus para de enviar novas sessões para este servidor, mas permite que as sessões atuais continuem sendo atendidas pela duração da sessão. Você também pode alternar essa configuração adicionando o parâmetro de drenagem a uma diretiva de servidor upstream e, em seguida, recarregando a configuração do NGINX Plus.

2.9 Verificações de integridade passivas

Problema

Você precisa verificar passivamente a integridade dos servidores upstream.

Solução

Use as verificações de integridade do NGINX com balanceamento de carga para garantir que apenas servidores upstream inteiros sejam utilizados:

```

back-end upstream
{
    server backend1.example.com:1234 max_fails=3 fail_timeout=3s;
    server backend2.example.com:1234 max_fails=3 fail_timeout=3s;
}

```

Essa configuração monitora passivamente a integridade do upstream monitorando a resposta das solicitações do cliente direcionadas ao servidor upstream. O exemplo define a diretiva max_fails para três e fail_timeout para 3 segundos. Esses parâmetros de diretiva funcionam da mesma maneira em servidores de fluxo e HTTP.

Discussão

A verificação de integridade passiva está disponível na versão de código aberto do NGINX e é configurada usando os mesmos parâmetros de servidor para平衡amento de carga HTTP, TCP e UDP. O monitoramento passivo observa conexões com falha ou com tempo limite à medida que passam pelo NGINX conforme solicitado por um cliente. As verificações de integridade passivas são habilitadas por padrão;

os parâmetros mencionados aqui permitem que você ajuste seu comportamento. O valor padrão de max_fails é 1 e o valor padrão de fail_timeout é 10s. O monitoramento da integridade é importante em todos os tipos de平衡amento de carga, não apenas do ponto de vista da experiência do usuário, mas também para a continuidade dos negócios. O NGINX monitora passivamente servidores HTTP, TCP e UDP upstream para garantir que estejam íntegros e funcionando.

Veja também

- [Guia de administração de verificação de integridade HTTP](#)
- [Guia de administração de verificação de integridade TCP](#)
- [Guia de administração da verificação de integridade do UDP](#)

2.10 Verificações de integridade ativas com NGINX Plus

Problema

Você precisa verificar ativamente a integridade de seus servidores upstream com o NGINX Plus.

Solução

Para HTTP, use a diretiva health_check em um bloco de localização:

```
http
{
    servidor
    {
        #...
        local / {
            proxy_pass http://backend;
            health_check interval=2s falha=2

            passes=5
            uri=/
            match=welcome;
        }

        } # status é 200, tipo de conteúdo é "text/html", # e
        # corpo contém "Welcome to nginx!" correspondência de
        # boas-vindas { status 200; cabeçalho Content-Type =
        # text/html; body ~ "Bem-vindo ao nginx!";

    }
}
```

Essa configuração de verificação de integridade para servidores HTTP verifica a integridade dos servidores upstream fazendo uma solicitação HTTP GET para o URI '/' a cada 2 segundos. O método HTTP não pode ser definido para verificações de integridade, apenas solicitações GET são executadas, pois outros métodos podem alterar o estado dos sistemas de back-end. Os servidores upstream devem passar por cinco verificações de integridade consecutivas para serem considerados íntegros. Eles são considerados

não saudável se falharem em duas verificações consecutivas. A resposta do servidor upstream deve corresponder ao bloco de correspondência definido, que define o código de status como 200, o valor do cabeçalho Content-Type como 'text/html' e a string "Welcome to nginx!" no corpo da resposta. O bloco de correspondência HTTP tem três diretivas: status, cabeçalho e corpo. Todas essas três diretivas também têm sinalizadores de comparação.

As verificações de integridade de fluxo para serviços TCP/UDP são muito semelhantes:

```
stream {
    # ...
    servidor
        { escuta 1234;
        proxy_pass stream_backend;
        health_check interval=10s
            passa=2 falha=3;
            health_check_timeout 5s;

        }
    # ...
}
```

Neste exemplo, um servidor TCP está configurado para escutar na porta 1234 e fazer proxy para um conjunto upstream de servidores, para os quais ele verifica ativamente a integridade. A diretiva stream health_check recebe todos os mesmos parâmetros do HTTP, com exceção de uri, e a versão do stream tem um parâmetro para alternar o protocolo de verificação para udp. Neste exemplo, o intervalo é definido como 10 segundos, requer que duas passagens sejam consideradas íntegras e três não sejam consideradas não íntegras. A verificação de integridade do fluxo ativo também pode verificar a resposta do servidor upstream. O bloco de correspondência para servidores de fluxo, no entanto, tem apenas duas diretivas: send e expect. A diretiva send são dados brutos a serem enviados e expect é uma resposta exata ou uma expressão regular para corresponder.

Discussão

No NGINX Plus, verificações de integridade passivas ou ativas podem ser usadas para monitorar os servidores de origem. Essas verificações de integridade podem medir mais do que apenas o código de resposta. No NGINX Plus, as verificações de integridade HTTP ativas monitoram com base em vários critérios de aceitação da resposta do servidor upstream. Você pode configurar o monitoramento de verificação de integridade ativa para a frequência com que os servidores upstream são verificados, quantas vezes um servidor deve passar nessa verificação para ser considerado íntegro, quantas vezes ele pode falhar antes de ser considerado não íntegro e qual deve ser o resultado esperado. Para uma lógica mais complexa, uma diretiva require para o bloco match permite o uso de variáveis cujo valor não deve ser vazio ou zero. O parâmetro de correspondência aponta para um bloco de correspondência que define os critérios de aceitação para a resposta. O bloco de correspondência também define os dados a serem enviados ao servidor upstream quando usado no contexto de fluxo para TCP/UDP.

Esses recursos permitem que o NGINX garanta que os servidores upstream estejam sempre íntegros.

Veja também

[Guia de administração de verificação de integridade HTTP](#)
[Guia de administração de verificação de integridade TCP](#)
[Guia de administração da verificação de integridade do UDP](#)

2.11 Início lento com NGINX Plus

Problema

Seu aplicativo precisa aumentar antes de assumir a carga total de produção.

Solução

Use o parâmetro slow_start na diretiva do servidor para aumentar gradualmente o número de conexões ao longo de um tempo especificado à medida que um servidor é reintroduzido no pool de平衡amento de carga upstream:

```
upstream
    { zona backend 64k;

        servidor server1.example.com slow_start=20s;
        servidor server2.example.com slow_start=15s;
    }
```

As configurações da diretiva do servidor aumentarão lentamente o tráfego para os servidores upstream depois que eles forem reintroduzidos no pool. server1 aumentará lentamente seu número de conexões em 20 segundos e server2 em 15 segundos.

Discussão

O início lento é o conceito de aumentar lentamente o número de solicitações enviadas por proxy para um servidor durante um período de tempo. A inicialização lenta permite que o aplicativo aqueça preenchendo caches, iniciando conexões de banco de dados sem ser sobrecarregado por conexões assim que é iniciado. Esse recurso entra em vigor quando um servidor que falhou nas verificações de integridade começa a passar novamente e entra novamente no pool de balanceamento de carga, e está disponível apenas no NGINX Plus. A inicialização lenta não pode ser usada com métodos hash, ip_hash ou de balanceamento de carga aleatório .

CAPÍTULO 3

Entre e Gestão

3.0 Introdução

NGINX e NGINX Plus também são classificados como controladores de tráfego da web. Você pode usar o NGINX para rotear o tráfego de forma inteligente e controlar o fluxo com base em muitos atributos. Este capítulo aborda a capacidade do NGINX de dividir solicitações de clientes com base em porcentagens; utilizar a localização geográfica dos clientes; e controlar o fluxo de tráfego na forma de limitação de taxa, conexão e largura de banda. Ao ler este capítulo, lembre-se de que você pode misturar e combinar esses recursos para permitir inúmeras possibilidades.

3.1 Teste A/B

Problema

Você precisa dividir os clientes entre duas ou mais versões de um arquivo ou aplicativo para testar a aceitação ou o engajamento.

Solução

Use o módulo `split_clients` para direcionar uma porcentagem de seus clientes para um pool `upstream` diferente:

```
split_clients "${remote_addr}AAA" $variant { "backendv2";
    20,0%      "backendv1";
    *
}
```

A diretiva `split_clients` faz um hash da string fornecida por você como o primeiro parâmetro e divide esse hash pelas porcentagens fornecidas para mapear o valor de uma variável fornecida como o segundo parâmetro. A adição de “AAA” ao primeiro parâmetro é para

demonstram que esta é uma string concatenada que pode incluir muitas variáveis, conforme mencionado no algoritmo genérico de balanceamento de carga de hash. O terceiro parâmetro é um objeto contendo pares chave-valor onde a chave é o peso percentual e o valor é o valor a ser atribuído. A chave pode ser uma porcentagem ou um asterisco.

O asterisco indica o resto do todo depois que todas as porcentagens são tomadas. O valor da variável \$variant será backendv2 para 20% dos endereços IP do cliente e backendv1 para os 80% restantes.

Neste exemplo, backendv1 e backendv2 representam pools de servidores upstream e podem ser usados com a diretiva proxy_pass assim:

```
local / {
    proxy_pass http://$variant
}
```

Usando a variável \$variant, nosso tráfego será dividido entre dois pools de servidores de aplicativos diferentes.

Para demonstrar a ampla variedade de usos que split_clients pode ter, a seguir está um exemplo de divisão entre duas versões de um site estático.

```
http
{
    split_clients "${remote_addr}" $site_root_folder {
        "/var/www/
        33,3%      sitev2"; "/var/www/sitev1";
        *
    }

    } servidor
    { escuta 80 _;
        raiz $site_root_folder;
        localização / { índice
            index.html;
        }
    }
}
```

Discussão

Esse tipo de teste A/B é útil ao testar diferentes tipos de recursos de marketing e front-end para taxas de conversão em sites de comércio eletrônico. É comum que os aplicativos usem um tipo de implantação chamado versão canário. Nesse tipo de implantação, o tráfego é transferido lentamente para a nova versão, aumentando gradualmente a porcentagem de usuários roteados para a nova versão. Dividir seus clientes entre diferentes versões de seu aplicativo pode ser útil ao lançar novas versões de código, para limitar o raio de explosão em caso de erro. Ainda mais comum é o estilo de implantação azul-verde, onde os usuários são transferidos para uma nova versão e a versão antiga ainda está disponível enquanto a implantação é validada. Seja qual for o motivo para dividir os clientes entre dois conjuntos de aplicativos diferentes, o NGINX simplifica isso por causa desse módulo split_clients .

[Veja também](#)

[split_clients Documentação](#)

3.2 Usando o Módulo GeolP e Banco de Dados

Problema

Você precisa instalar o banco de dados GeolP e habilitar suas variáveis incorporadas dentro NGINX para utilizar a localização física de seus clientes no log do NGINX, solicitações com proxy ou roteamento de solicitações.

Solução

O repositório oficial de pacotes NGINX Open Source, configurado no [Capítulo 2](#) ao instalar o NGINX, fornece um pacote chamado nginx-module-geoip. Ao usar o repositório de pacotes NGINX Plus, esse pacote é denominado nginx-plus-module-geoip. Esses pacotes instalaram a versão dinâmica do módulo GeolP.

RHEL/CentOS NGINX Open Source:

```
# yum install nginx-module-geoip
```

Debian/Ubuntu NGINX Open Source:

```
# apt-get install nginx-module-geoip
```

RHEL/CentOS NGINX Plus:

```
# yum install nginx-plus-module-geoip
```

Debian/Ubuntu NGINX Plus:

```
# apt-get install nginx-plus-module-geoip
```

Baixe os bancos de dados GeolP de países e cidades e descompacte-os:

```
# mkdir /etc/nginx/geoip # cd /
etc/nginx/geoip # wget "http://
geolite.maxmind.com/\ download/geoip/
database/GeoLiteCountry/GeoIP.dat.gz" # gunzip GeoIP.dat. gz
# wget "http://geolite.maxmind.com/\ download/geoip/database/
GeoLiteCity.dat.gz" # gunzip GeoLiteCity.dat.gz
```

Este conjunto de comandos cria um diretório geoip no diretório /etc/nginx, move para este novo diretório e baixa e descompacta os pacotes.

Com o banco de dados GeolP para países e cidades no disco local, agora você pode instruir o módulo NGINX GeolP a usá-los para expor variáveis incorporadas com base no endereço IP do cliente:

```
load_module "/usr/lib64/nginx/modules/ngx_http_geoip_module.so";  
  
http  
{  
    geoip_country /etc/nginx/geoip/GeolP.dat;  
    geoip_city /etc/nginx/geoip/GeoLiteCity.dat;  
    ...  
}
```

A diretiva `load_module` carrega dinamicamente o módulo de seu caminho no sistema de arquivos. A diretiva `load_module` só é válida no contexto principal. A diretiva `geoip_country` leva um caminho para o arquivo `GeolP.dat` que contém os endereços IP de mapeamento do banco de dados para códigos de país e é válida apenas no contexto HTTP.

Discussão

Para usar essa funcionalidade, você deve ter o módulo NGINX GeolP instalado e um banco de dados local GeolP do condado e da cidade. A instalação e a recuperação desses pré-requisitos foram demonstradas nesta seção.

As diretivas `geoip_country` e `geoip_city` expõem várias variáveis incorporadas disponíveis neste módulo. A diretiva `geoip_country` habilita variáveis que permitem distinguir o país de origem do seu cliente. Essas variáveis incluem `$geoip_country_code`, `$geoip_country_code3` e `$geoip_country_name`. A variável do código do país retorna o código do país de duas letras e a variável com um 3 no final retorna o código do país de três letras. A variável nome do país retorna o nome completo do país.

A diretiva `geoip_city` habilita algumas variáveis. A diretiva `geoip_city` habilita todas as mesmas variáveis que a diretiva `geoip_country`, apenas com nomes diferentes, como `$geoip_city_country_code`, `$geoip_city_country_code3` e `$geoip_city_country_name`. Outras variáveis incluem `$geoip_city`, `$geoip_latitude`, `$geoip_longitude`, `$geoip_city_continent_code` e `$geoip_postal_code`, todas descriptivas do valor que retornam. `$geoip_region` e `$geoip_region_name` descrevem a região, território, estado, província, terra federal e similares. Região é o código de duas letras, onde o nome da região é o nome completo. `$geoip_area_code`, válido apenas nos EUA, retorna o código de área do telefone de três dígitos.

Com essas variáveis, você pode registrar informações sobre seu cliente. Opcionalmente, você pode passar essas informações para seu aplicativo como um cabeçalho ou variável ou usar o NGINX para rotear seu tráfego de maneiras específicas.

Veja também[Módulo NGINX GeoIP](#)[Atualização GeoIP](#)

3.3 Restringindo o acesso com base no país

Problema

Você precisa restringir o acesso de determinados países para requisitos contratuais ou de aplicação.

Solução

Mapeie os códigos de país que você deseja bloquear ou permitir para uma variável:

```
load_module
    "/usr/lib64/nginx/modules/ngx_http_geoip_module.so";

http
    { mapa $ geoip_country_code $ country_access {
        "NÓS"      0;
        "RU"       0;
        padrão 1;
    }
    # ...
}
```

Esse mapeamento definirá uma nova variável \$country_access como 1 ou 0. Se o endereço IP do cliente for originário dos EUA ou da Rússia, a variável será definida como 0. Para qualquer outro país, a variável será definida como 1 .

Agora, dentro do nosso bloco de servidor , usaremos uma instrução if para negar acesso a qualquer pessoa que não seja originária dos EUA ou da Rússia:

```
servidor
    { if ($country_access = '1') { return
        403;
    }
    # ...
}
```

Esta instrução if avaliará True se a variável \$country_access estiver definida como 1. Quando True, o servidor retornará um 403 não autorizado. Caso contrário, o servidor funciona normalmente. Então, isso se o bloco está lá apenas para negar pessoas que não são dos EUA ou da Rússia.

Discussão

Este é um exemplo curto, mas simples, de como permitir o acesso apenas de alguns países. Este exemplo pode ser explicado para atender às suas necessidades. Você pode utilizar esta mesma prática para permitir ou bloquear com base em qualquer uma das variáveis incorporadas disponibilizadas no módulo GeoIP.

3.4 Encontrando o Cliente Original

Problema

Você precisa encontrar o endereço IP do cliente original porque há proxies na frente do servidor NGINX.

Solução

Use a diretiva geoip_proxy para definir o intervalo de endereços IP do seu proxy e a diretiva geoip_proxy_recursive para procurar o IP original:

```
load_module "/usr/lib64/nginx/modules/ngx_http_geoip_module.so";  
  
http  
{  
    geoip_country /etc/nginx/geoip/GeoIP.dat;  
    geoip_city /etc/nginx/geoip/GeoLiteCity.dat;  
    geoip_proxy 10.0.16.0/26; geoip_proxy_recursive  
    ativado;  
    ...  
    # }  
}
```

A diretiva geoip_proxy define um intervalo de roteamento entre domínios sem classes (CIDR) no qual nossos servidores proxy residem e instrui o NGINX a utilizar o cabeçalho X-Forwarded-For para encontrar o endereço IP do cliente. A diretiva geoip_proxy_recursive instrui o NGINX a procurar recursivamente no cabeçalho X-Forwarded-For o último IP do cliente conhecido.



Padronização do cabeçalho encaminhado

Um cabeçalho denominado Encaminhado tornou-se o cabeçalho padrão para adicionar informações de proxy para solicitações com proxy. O cabeçalho usado pelo módulo NGINX GeoIP é X-Forwarded-For e não pode ser conyimaginado de outra forma no momento da escrita. Embora o X-Forwarded-For não seja um padrão oficial, ele ainda é amplamente usado, aceito e definido pela maioria dos proxies.

Discussão

Você pode descobrir que, se estiver usando um proxy na frente do NGINX, o NGINX pegará o endereço IP do proxy em vez do cliente. Para isso você pode usar a diretiva geoip_proxy para instruir o NGINX a usar o cabeçalho X-Forwarded-For quando as conexões

são abertos a partir de um determinado intervalo. A diretiva geoip_proxy recebe um endereço ou um intervalo CIDR. Quando há vários proxies passando tráfego na frente do NGINX, você pode usar a diretiva geoip_proxy_recursive para pesquisar recursivamente nos endereços X Forwarded-For para encontrar o cliente de origem. Você desejará usar algo assim ao utilizar平衡adores de carga como o Amazon Web Services Elastic Load Balancing (AWS ELB), o balanceador de carga do Google ou o balanceador de carga do Microsoft Azure na frente do NGINX.

3.5 Limitando Conexões

Problema

Você precisa limitar o número de conexões com base em uma chave predefinida, como o endereço IP do cliente.

Solução

Construa uma zona de memória compartilhada para manter as métricas de conexão e use a diretiva limit_conn para limitar as conexões abertas:

```
http
{
    limit_conn_zone $binary_remote_addr zone=limitbyaddr:10m;
    limit_conn_status 429; # servidor {
        ...
        # ...
        limit_conn limitbyaddr 40;
        # ...
    }
}
```

Essa configuração cria uma zona de memória compartilhada denominada limitbyaddr. A chave predefinida utilizada é o endereço IP do cliente em formato binário. O tamanho da zona de memória compartilhada é definido para 10 MB. A diretiva limit_conn recebe dois parâmetros: um nome limit_conn_zone e o número de conexões permitidas. O limit_conn_status define a resposta quando as conexões estão limitadas a um status de 429, indicando muitas solicitações. As diretivas limit_conn e limit_conn_status são válidas no contexto HTTP, servidor e local.

Discussão

Limitar o número de conexões com base em uma chave pode ser usado para se defender contra abusos e compartilhar seus recursos de maneira justa entre todos os seus clientes. É importante ter cuidado com a sua chave predefinida. Usar um endereço IP, como estamos no exemplo anterior, pode ser perigoso se muitos usuários estiverem na mesma rede que se origina do mesmo IP, como quando atrás de uma tradução de endereço de rede (NAT).

Todo o grupo de clientes será limitado. A diretiva `limit_conn_zone` é válida apenas no contexto HTTP. Você pode utilizar qualquer número de variáveis disponíveis para NGINX dentro do contexto HTTP para construir uma string na qual limitar. Utilizar uma variável que possa identificar o usuário no nível do aplicativo, como um cookie de sessão, pode ser uma solução mais limpa, dependendo do caso de uso. O `limite_conn_status` é padronizado para 503, serviço indisponível. Você pode achar preferível usar um 429, pois o serviço está disponível, e respostas de nível 500 indicam erro do servidor, enquanto respostas de nível 400 indicam cíli erro.

As limitações de teste podem ser complicadas. Muitas vezes, é difícil simular o tráfego ao vivo em um ambiente alternativo para teste. Nesse caso, você pode definir a diretiva `limit_req_dry_run` para `on` e, em seguida, usar a variável `$limit_req_status` em seu log de acesso. A variável `$limit_req_status` será avaliada como `PASSED`, `DELAYED`, `REJECTED`, `DELAYED_DRY_RUN` ou `REJECTED_DRY_RUN`. Com a simulação habilitada, você poderá analisar os logs de tráfego ao vivo e ajustar seus limites conforme necessário antes de habilitar; fornecendo a você a garantia de que sua configuração de limite está correta.

3.6 Taxa Limitante

Problema

Você precisa limitar a taxa de solicitações por uma chave predefinida, como o endereço IP do cliente.

Solução

Utilize o módulo de limitação de taxa para limitar a taxa de solicitações:

```
http
{
    limit_req_zone $binary_remote_addr
        zone=limitbyaddr:10m rate=3r/s;
    limit_req_status 429;
    #
    servidor
    {
        #
        limit_req zona=limitbyaddr;
        #
    }
}
```

Esta configuração de exemplo cria uma zona de memória compartilhada denominada limitbyaddr. A chave predefinida usada é o endereço IP do cliente em formato binário. O tamanho da zona de memória compartilhada é definido como 10 MB. A zona define a taxa com um argumento de palavra-chave. A diretiva limit_req recebe um argumento de palavra-chave obrigatório: zona. zone instrui a diretiva sobre qual zona de limite de solicitação de memória compartilhada usar. As solicitações que excedem a taxa expressa são retornadas com um código HTTP 429, conforme definido pela diretiva limit_req_status. Aconselho a definir um status na faixa de 400 níveis, pois o padrão é 503, implicando em um problema com o servidor, quando na verdade o problema é com o cliente.

Use argumentos de palavra-chave opcionais para a diretiva limit_req para habilitar a limitação de taxa de dois estágios:

```
servidor
{ local / {
    limit_req zona=limitbyaddr burst=12 delay=9;
}
}
```

Em alguns casos, um cliente precisará fazer muitas solicitações de uma só vez e, em seguida, reduzirá sua taxa por um período de tempo antes de fazer mais. Você pode usar o argumento de palavra-chave burst para permitir que o cliente exceda seu limite de taxa, mas não tenha requisições rejeitadas. As solicitações de taxa excedida terão um atraso no processamento para corresponder ao limite de taxa até o valor configurado. Um conjunto de argumentos de palavras-chave altera esse comportamento: delay e nodelay. O argumento nodelay não aceita um valor e simplesmente permite que o cliente consuma o valor de intermitência de uma só vez, no entanto, todas as solicitações serão rejeitadas até que tenha passado tempo suficiente para satisfazer o limite de taxa. Neste exemplo, se usássemos nodelay, o cliente poderia consumir 12 requisições no primeiro segundo, mas teria que esperar 4 segundos após a inicial para fazer outra. O argumento de palavra-chave delay define quantas solicitações podem ser feitas antecipadamente sem limitação. Nesse caso, o cliente pode fazer nove solicitações antecipadamente sem demora, as próximas três serão estranguladas e qualquer outra dentro de um período de 4 segundos será rejeitada.

Discussão

O módulo de limitação de taxa é muito poderoso para proteger contra solicitações rápidas abusivas, ao mesmo tempo em que oferece um serviço de qualidade a todos. Há muitas razões para limitar a taxa de solicitação, sendo uma delas a segurança. Você pode negar um ataque de força bruta colocando um limite muito estreito em sua página de login. Você pode definir um limite sensato para todas as solicitações, desabilitando assim os planos de usuários mal-intencionados que podem tentar negar serviço ao seu aplicativo ou desperdiçar recursos. A configuração do módulo de limite de taxa é muito parecida com o módulo de limitação de conexão anterior descrito na [Receita 3.5](#), e muitas das mesmas preocupações se aplicam. Você pode especificar a taxa na qual as solicitações são limitadas em solicitações por segundo ou solicitações por minuto. Quando o limite de taxa é atingido, o incidente é registrado. Há também uma diretiva que não está no exemplo, limit_req_log_level,

cujo padrão é error, mas pode ser definido como info, notice ou warning. No NGINX Plus, a limitação de taxa agora reconhece o cluster (consulte a [Receita 12.5](#) para obter um exemplo de sincronização de zona).

As limitações de teste podem ser complicadas. Muitas vezes, é difícil simular o tráfego ao vivo em um ambiente alternativo para teste. Nesse caso, você pode definir a diretiva `limit_conn_dry_run` para on e usar a variável `$limit_conn_status` em seu log de acesso. A variável `$limit_conn_status` será avaliada como PASSED, REJECTED ou REJECTTED_DRY_RUN. Com a simulação habilitada, você poderá analisar os logs de tráfego ao vivo e ajustar seus limites conforme necessário antes de habilitar; fornecendo a você a garantia de que sua configuração de limite está correta.

3.7 Limitando a largura de banda

Problema

Você precisa limitar a largura de banda de download por cliente para seus ativos.

Solução

Utilize as diretivas `limit_rate` e `limit_rate_after` do NGINX para limitar a largura de banda de resposta a um cliente:

```
local /baixar/ {
    taxa_limite_após 10m;
    taxa_limite 1m;
}
```

A configuração deste bloco de localização especifica que para URIs com o prefixo `download`, a taxa na qual a resposta será servida ao cliente será limitada após 10 MB a uma taxa de 1 MB por segundo. O limite de largura de banda é por conexão, portanto, convém instituir um limite de conexão, bem como um limite de largura de banda, quando aplicável.

Discussão

Limitar a largura de banda para conexões específicas permite que o NGINX compartilhe sua largura de banda de upload em todos os clientes da maneira que você especificar. Essas duas diretivas fazem tudo: `limit_rate_after` e `limit_rate`. A diretiva `limit_rate_after` pode ser definida em quase qualquer contexto: HTTP, servidor, local e if quando o if estiver dentro de um local. A diretiva `limit_rate` é aplicável nos mesmos contextos que `limit_rate_after`; no entanto, ele pode ser definido alternativamente por uma variável chamada `$limit_rate`.

A diretiva `limit_rate_after` especifica que a conexão não deve ser limitada por taxa até que uma quantidade especificada de dados tenha sido transferida. A diretiva `limit_rate` especifica o limite de taxa para um determinado contexto em bytes por segundo por padrão.

No entanto, você pode especificar `m` para megabytes ou `g` para gigabytes. Ambas as diretivas têm como padrão o valor 0. O valor 0 significa não limitar as taxas de download. Este módulo permite que você altere programaticamente o limite de taxa de clientes.

CAPÍTULO 4

Cache de conteúdo massivamente escalável

4.0 Introdução

O armazenamento em cache acelera o fornecimento de conteúdo armazenando respostas de solicitação para serem atendidas novamente no futuro. O cache de conteúdo reduz a carga dos servidores upstream, armazenando em cache a resposta completa em vez de executar cálculos e consultas novamente para a mesma solicitação. O armazenamento em cache aumenta o desempenho e reduz a carga, o que significa que você pode servir mais rapidamente com menos recursos. O dimensionamento e a distribuição de servidores de cache em locais estratégicos podem ter um efeito dramático na experiência do usuário. É ideal hospedar conteúdo próximo ao consumidor para obter o melhor desempenho. Você também pode armazenar em cache seu conteúdo perto de seus usuários. Este é o padrão das redes de entrega de conteúdo, ou CDNs. Com o NGINX, você pode armazenar seu conteúdo em cache onde quer que possa colocar um servidor NGINX, permitindo que você crie seu próprio CDN. Com o cache do NGINX, você também pode armazenar passivamente em cache e servir respostas em cache no caso de uma falha de upstream. Os recursos de cache estão disponíveis apenas no contexto http .

4.1 Zonas de Cache

Problema

Você precisa armazenar o conteúdo em cache e definir onde o cache é armazenado.

Solução

Use a diretiva proxy_cache_path para definir zonas de cache de memória compartilhada e um local para o conteúdo:

```
proxy_cache_path /var/nginx/cache  
    keys_zone=CACHE:60m  
    niveis=1:2
```

```

inativo=3h
max_size=20g;
proxy_cache CACHE;
```

O exemplo de definição de cache cria um diretório para respostas em cache no sistema de arquivos em /var/nginx/cache e cria um espaço de memória compartilhado chamado CACHE com 60 MB de memória. Este exemplo define os níveis da estrutura de diretórios, define a liberação de respostas em cache após elas não terem sido solicitadas em 3 horas e define um tamanho máximo do cache de 20 GB. A diretiva proxy_cache informa um contexto particular para usar a zona de cache. O proxy_cache_path é válido no contexto HTTP e a diretiva proxy_cache é válida nos contextos HTTP, servidor e local.

Discussão

Para configurar o cache no NGINX, é necessário declarar um caminho e uma zona a ser utilizada. Uma zona de cache no NGINX é criada com a diretiva proxy_cache_path. O proxy_cache_path designa um local para armazenar as informações em cache e um espaço de memória compartilhado para armazenar chaves ativas e metadados de resposta. Parâmetros opcionais para esta diretiva fornecem mais controle sobre como o cache é mantido e acessado.

O parâmetro levels define como a estrutura do arquivo é criada. O valor é um valor separado por dois pontos que declara o comprimento dos nomes de subdiretórios, com um máximo de três níveis. O NGINX armazena em cache com base na chave de cache, que é um valor com hash. O NGINX armazena o resultado na estrutura de arquivo fornecida, usando a chave de cache como um caminho de arquivo e dividindo os diretórios com base no valor dos níveis . O parâmetro inativo permite controlar por quanto tempo um item de cache ficará hospedado após seu último uso. O tamanho do cache também é configurável com o uso do parâmetro max_size .

Outros parâmetros estão relacionados ao processo de carregamento de cache, que carrega as chaves de cache na zona de memória compartilhada dos arquivos armazenados em cache no disco.

4.2 Bloqueio de cache

Problema

Você não quer que o NGINX faça proxy de solicitações que estão sendo gravadas no cache de um servidor upstream.

Solução

Use a diretiva proxy_cache_lock para garantir que apenas uma solicitação seja capaz de gravar no cache por vez, onde as solicitações subsequentes aguardarão a gravação da resposta:

```

proxy_cache_lock
ativado; proxy_cache_lock_age
10s; proxy_cache_lock_timeout 3s;
```

Discussão

A diretiva proxy_cache_lock instrui o NGINX a manter solicitações, destinadas a um elemento em cache, que está sendo preenchido no momento. A requisição proxy que está preenchendo o cache é limitada na quantidade de tempo que ela tem antes que outra requisição tente preencher o elemento, definido pela diretiva proxy_cache_lock_age , cujo padrão é 5 segundos. O NGINX também pode permitir que solicitações que estavam esperando um determinado período de tempo passem para o servidor proxy, que não tentará preencher o cache usando a diretiva proxy_cache_lock_timeout , que também tem o padrão de 5 segundos. Você pode pensar na diferença entre as duas diretivas como proxy_cache_lock_age : "Você está demorando muito, vou preencher o cache para você" e proxy_cache_lock_timeout: "Você está demorando muito para eu esperar, estou vou conseguir o que preciso e deixar você preencher o cache no seu próprio tempo."

4.3 Cache de chaves de hash

Problema

Você precisa controlar como seu conteúdo é armazenado em cache e recuperado.

Solução

Use a diretiva proxy_cache_key junto com as variáveis para definir o que constitui um acerto ou erro de cache:

```
proxy_cache_key "$host$request_uri $cookie_user";
```

Essa chave de hash de cache instruirá o NGINX a armazenar em cache as páginas com base no host e no URI solicitados, bem como um cookie que define o usuário. Com isso, você pode armazenar em cache páginas dinâmicas sem veicular conteúdo gerado para um usuário diferente.

Discussão

O proxy_cache_key padrão, que se encaixa na maioria dos casos de uso, é "\$scheme\$proxy_host\$request_uri". As variáveis usadas incluem o esquema, HTTP ou HTTPS, o proxy_host, para onde a solicitação está sendo enviada e o URI da solicitação. Tudo junto, isso reflete a URL para a qual o NGINX está fazendo proxy da solicitação. Você pode descobrir que existem muitos outros fatores que definem uma solicitação exclusiva por aplicativo - como argumentos de solicitação, cabeçalhos, identificadores de sessão e assim por diante - para os quais você deseja criar sua própria chave de hash.¹

¹ Qualquer combinação de texto ou variáveis expostas ao NGINX pode ser usada para formar uma chave de cache. Uma lista de variáveis é disponível no NGINX.

A seleção de uma boa chave de hash é muito importante e deve ser pensada com a compreensão do aplicativo. Selecionar uma chave de cache para conteúdo estático normalmente é bastante simples; usar o nome do host e o URI será suficiente. Selecionar uma chave de cache para conteúdo bastante dinâmico, como páginas para um aplicativo de painel, requer mais conhecimento sobre como os usuários interagem com o aplicativo e o grau de variação entre as experiências do usuário. Devido a questões de segurança, talvez você não queira apresentar dados em cache de um usuário para outro sem entender completamente o contexto. A diretiva proxy_cache_key configura a string a ser hash para a chave de cache. O proxy_cache_key pode ser definido no contexto de HTTP, servidor e blocos de localização, fornecendo controle flexível sobre como as solicitações são armazenadas em cache.

4.4 Desvio de Cache

Problema

Você precisa da capacidade de contornar o cache.

Solução

Use a diretiva proxy_cache_bypass com um valor não vazio ou diferente de zero. Uma maneira de fazer isso é definir uma variável dentro dos blocos de localização que você não deseja armazenar em cache para igual a 1:

```
proxy_cache_bypass $http_cache_bypass;
```

A configuração diz ao NGINX para ignorar o cache se o cabeçalho de solicitação HTTP chamado cache_bypass estiver definido para qualquer valor que não seja 0. Este exemplo usa um cabeçalho como variável para determinar se o cache deve ser ignorado—o cliente precisaria definir especificamente esse cabeçalho por seu pedido.

Discussão

Há vários cenários que exigem que a solicitação não seja armazenada em cache. Para isso, o NGINX expõe uma diretiva proxy_cache_bypass para que, quando o valor for não vazio ou diferente de zero, a solicitação seja enviada para um servidor upstream em vez de ser extraída do cache. Diferentes necessidades e cenários para ignorar o cache serão ditados pelo caso de uso de seus aplicativos. As técnicas para contornar o cache podem ser tão simples quanto usar um cabeçalho de solicitação ou resposta ou tão complexas quanto vários blocos de mapa trabalhando juntos.

Por muitas razões, você pode querer ignorar o cache. Um motivo importante é a solução de problemas e a depuração. A reprodução de problemas pode ser difícil se você estiver constantemente puxando páginas em cache ou se sua chave de cache for específica para um identificador de usuário. Ter a capacidade de contornar o cache é vital. As opções incluem, mas não estão limitadas a, ignorar o cache quando um determinado cookie, cabeçalho ou argumento de solicitação é definido. Você também pode

desligue o cache completamente para um determinado contexto, como um bloco de localização, definindo proxy_cache off;.

4.5 Desempenho do Cache

Problema

Você precisa aumentar o desempenho armazenando em cache no lado do cliente.

Solução

Use cabeçalhos de controle de cache do lado do cliente:

```
local ~* \.(css|js)$ { expira 1 ano;
    add_header Cache-Control
        "público";
}
```

Este bloco de localização especifica que o cliente pode armazenar em cache o conteúdo de arquivos CSS e Java-Script. A diretiva expires instrui o cliente que seu recurso em cache não será mais válido após um ano. A diretiva add_header adiciona o cabeçalho de resposta HTTP Cache-Control à resposta, com um valor de public, que permite que qualquer servidor de cache ao longo do caminho armazene o recurso em cache. Se especificarmos private, somente o cliente poderá armazenar o valor em cache.

Discussão

O desempenho do cache tem muitos fatores, sendo a velocidade do disco no topo da lista. Há muitas coisas na configuração do NGINX que você pode fazer para ajudar no desempenho do cache. Uma opção é definir os cabeçalhos da resposta de tal forma que o cliente realmente armazene a resposta em cache e não faça a solicitação ao NGINX, mas simplesmente a sirva de seu próprio cache.

4.6 Limpeza de cache com NGINX Plus

Problema

Você precisa invalidar um objeto do cache.

Solução

Use o recurso purge do NGINX Plus, a diretiva proxy_cache_purge e uma variável não vazia ou de valor zero:

```
map $request_method $purge_method {
    PURGE 1;
```

```

    padrão 0;

} servidor
{ #...
local / {
# ...
proxy_cache_purge $purge_method;
}
}

```

Neste exemplo, o cache de um objeto específico será limpo se for solicitado com um método PURGE. Veja a seguir um exemplo de curl de limpeza do cache de um arquivo chamado main.js:

```
$ curl -XPURGE localhost/main.js
```

Discussão

Uma maneira comum de lidar com arquivos estáticos é colocar um hash do arquivo no nome do arquivo. Isso garante que, à medida que você implementa novo código e conteúdo, seu CDN o reconheça como um novo arquivo porque o URI foi alterado. No entanto, isso não funciona exatamente para conteúdo dinâmico para o qual você definiu chaves de cache que não se encaixam nesse modelo. Em cada cenário de cache, você deve ter uma maneira de limpar o cache. O NGINX Plus forneceu um método simples de limpeza de respostas em cache. A diretiva proxy_cache_purge , ao passar um valor diferente de zero ou não vazio, limpará os itens armazenados em cache correspondentes à solicitação. Uma maneira simples de configurar a limpeza é mapeando o método de solicitação para PURGE. No entanto, você pode querer usar isso em conjunto com o módulo geo_ip ou autenticação simples para garantir que ninguém possa limpar seus preciosos itens de cache. O NGINX também permitiu o uso de * , que limpará itens de cache que correspondem a um prefixo de URI comum. Para usar curingas, você precisará configurar sua diretiva proxy_cache_path com o argumento purger=on .

4.7 Divisão de Cache

Problema

Você precisa aumentar a eficiência do cache segmentando o arquivo em fragmentos.

Solução

Use a diretiva slice NGINX e suas variáveis incorporadas para dividir o resultado do cache em fragmentos:

```

proxy_cache_path /tmp/mycache keys_zone=mycache:10m;
servidor {
# ...
proxy_cache mycache;

```

```
fatia 1m;
proxy_cache_key $host$uri$is_args$args$slice_range;
proxy_set_header Faixa $slice_range; proxy_http_version 1.1;
proxy_cache_valid 200 206 1h;

local /
    { proxy_pass http://origin:80;
}
}
```

Discussão

Essa configuração define uma zona de cache e a habilita para o servidor. A diretiva slice é então usada para instruir o NGINX a dividir a resposta em segmentos de arquivo de 1 MB. Os arquivos de cache são armazenados de acordo com a diretiva proxy_cache_key . Observe o uso da variável incorporada denominada slice_range . Essa mesma variável é usada como cabeçalho ao fazer a solicitação para a origem, e essa versão HTTP da solicitação é atualizada para HTTP/1.1 porque a versão 1.0 não suporta solicitações de intervalo de bytes. A validade do cache é definida para códigos de resposta de 200 ou 206 por 1 hora e, em seguida, a localização e as origens são definidas.

O módulo Cache Slice foi desenvolvido para entrega de vídeo HTML5, que usa solicitações de intervalo de bytes para pseudotransmitir conteúdo para o navegador. Por padrão, o NGINX é capaz de atender solicitações de intervalo de bytes de seu cache. Se uma solicitação de intervalo de bytes for feita para conteúdo não armazenado em cache, o NGINX solicitará o arquivo inteiro da origem. Quando você usa o módulo Cache Slice, o NGINX solicita apenas os segmentos necessários da origem. As solicitações de intervalo maiores que o tamanho da fatia, incluindo o arquivo inteiro, acionam subsolicitações para cada um dos segmentos necessários e, em seguida, esses segmentos são armazenados em cache. Quando todos os segmentos são armazenados em cache, a resposta é montada e enviada ao cliente, permitindo que o NGINX armazene em cache e atenda com mais eficiência o conteúdo solicitado em intervalos. O módulo Cache Slice deve ser usado apenas em arquivos grandes que não são alterados. O NGINX valida a ETag cada vez que recebe um segmento da origem. Se a ETag na origem for alterada, o NGINX abortará a população de cache do segmento porque o cache não é mais válido. Se o conteúdo mudar e o arquivo for menor, ou sua origem puder lidar com picos de carga durante o processo de preenchimento de cache, é melhor usar o módulo Cache Lock descrito no blog listado na seção Veja também a seguir. Este módulo não é compilado por padrão e precisa ser habilitado pela configuração --with-http_slice_module ao compilar o NGINX.

Veja também

[Cache de intervalo de bytes inteligente e eficiente com NGINX e NGINX Plus](#)

CAPÍTULO 5

Programabilidade e Automação

5.0 Introdução

Programabilidade refere-se à habilidade de interagir com algo através da programação. A API para NGINX Plus oferece exatamente isso: a capacidade de interagir com a configuração e o comportamento do NGINX Plus por meio de uma interface HTTP. Essa API oferece a capacidade de reconfigurar o NGINX Plus adicionando ou removendo servidores upstream por meio de solicitações HTTP. O recurso de armazenamento de valor-chave no NGINX Plus permite outro nível de configuração dinâmica - você pode utilizar chamadas HTTP para injetar informações que o NGINX Plus pode usar para rotear ou controlar o tráfego dinamicamente. Este capítulo abordará a API do NGINX Plus e o módulo de armazenamento de valor-chave exposto por essa mesma API.

As ferramentas de gerenciamento de configuração automatizam a instalação e configuração de servidores, o que é um utilitário inestimável na era da nuvem. Os engenheiros de aplicativos da Web em grande escala não precisam mais configurar servidores manualmente; em vez disso, eles podem usar uma das muitas ferramentas de gerenciamento de configuração disponíveis. Com essas ferramentas, os engenheiros precisam escrever configurações e códigos apenas uma vez para produzir muitos servidores com a mesma configuração de forma repetível, testável e modular. Este capítulo aborda algumas das ferramentas de gerenciamento de configuração mais populares disponíveis e como usá-las para instalar o NGINX e modelar uma configuração básica. Esses exemplos são extremamente básicos, mas demonstram como iniciar um servidor NGINX com cada plataforma.

5.1 API NGINX Plus

Problema

Você tem um ambiente dinâmico e precisa reconfigurar o NGINX Plus rapidamente.

Solução

Configure a API do NGINX Plus para permitir a adição e remoção de servidores por meio de chamadas de API:

```
back-end upstream
    { zona http_backend 64k;

} servidor {
    ...
    # local /api { api
        [write=on];
        # Diretivas que limitam o acesso à API
        # Veja o capítulo 7
    }

    local = /dashboard.html {
        root /usr/share/nginx/html;
    }
}
```

Essa configuração do NGINX Plus cria um servidor upstream com uma zona de memória compartilhada, habilita a API no bloco de localização /api e fornece um local para o painel do NGINX Plus.

Você pode utilizar a API para adicionar servidores quando eles ficarem online:

```
$ curl -X POST -d '{"server":"172.17.0.3"}' \ 'http://
nginx.local/api/3/http/upstreams/backend/servers/'

{
    "id":0,
    "server":"172.17.0.3:80",
    "weight":1, "max_conns":0,
    "maxfails":1,
    "fail_timeout":"10s",
    "slow_start":"0s ", "route":"",
    "backup":false, "down":false
}
```

A chamada curl neste exemplo faz uma solicitação ao NGINX Plus para adicionar um novo servidor à configuração upstream de back-end. O método HTTP é um POST, um objeto JSON é passado como corpo e uma resposta JSON é retornada. A resposta JSON mostra a configuração do objeto do servidor, observe que um novo id foi gerado e outras configurações foram definidas com valores padrão.

A API NGINX Plus é RESTful; portanto, existem parâmetros no URI de solicitação.

O formato do URI é o seguinte:

```
/api/{version}/http/upstreams/{httpUpstreamName}/servers/
```

Você pode utilizar a API NGINX Plus para listar os servidores no pool upstream:

```
$ curl 'http://nginx.local/api/3/http/upstreams/backend/servers/' [
  {
    "id":0,
    "server":"172.17.0.3:80",
    "weight":1, "max_conns":0,
    "maxfails":1, "fail_timeout":"10s",
    "slow_start":"0s ", "route":"",
    "backup":false, "down":false
  }
]
```

A chamada curl neste exemplo faz uma solicitação ao NGINX Plus para listar todos os servidores no pool upstream denominado back- end. Atualmente, temos apenas um servidor que adicionamos na chamada curl anterior à API. A solicitação retornará um objeto de servidor upstream que contém todas as opções configuráveis para um servidor.

Use a API do NGINX Plus para drenar conexões de um servidor upstream, preparando-o para uma remoção normal do pool upstream. Você pode encontrar detalhes sobre a drenagem de conexão na [Receita 2.8](#):

```
$ curl -X PATCH -d '{"drain":true}' \
  'http://nginx.local/api/3/http/upstreams/backend/servers/0'
{
  "id":0,
  "server":"172.17.0.3:80",
  "weight":1, "max_conns":0,
  "maxfails":1, "fail_timeout":
  "10s", "slow_start": "0s ",
  "route": "", "backup":false,
  "down":false, "drain":true
}
```

Neste curl, especificamos que o método de solicitação é PATCH, passamos um corpo JSON instruindo-o a drenar conexões para o servidor e especificamos o ID do servidor anexando-o ao URI. Encontramos o ID do servidor listando os servidores no pool upstream no comando curl anterior.

O NGINX Plus começará a drenar as conexões. Esse processo pode demorar tanto quanto a duração das sessões do aplicativo. Para verificar quantas conexões ativas estão sendo atendidas pelo servidor que você começou a drenar, use a seguinte chamada e procure o atributo ativo do servidor que está sendo drenado:

```
$ curl 'http://nginx.local/api/3/http/upstreams/backend' {

    "zone" : "http_backend",
    "keepalive" : 0, "peers" : [ {

        "backup" : false, "id" :
        0, "unavail" : 0,
        "name" : "172.17.0.3",
        "requests" : 0, "recebido" :
        0, "state" : "drenagem",
        "servidor" : "172.17.0.3:80",
        "ativo" : 0, "peso" : 1,
        "falha" : 0, "enviado" : 0,
        "respostas" : { "4xx" : 0, "total" :
        0 , "3xx": 0, "5xx": 0, "2xx": 0,
        "1xx": 0

    },
    "health_checks":
        { "checks": 0,
        "unhealthy": 0,
        "fails": 0
    },
    "tempo de inatividade": 0
}
],
"zumbis" : 0
}
```

Depois que todas as conexões forem drenadas, utilize a API NGINX Plus para remover completamente o servidor do pool upstream:

```
$ curl -X DELETE \
'http://nginx.local/api/3/http/upstreams/backend/servers/0' []
```

O comando curl faz uma solicitação do método DELETE para o mesmo URI usado para atualizar o estado dos servidores. O método DELETE instrui o NGINX a remover o servidor. Esse

A chamada da API retorna todos os servidores e seus IDs que ainda permanecem no pool. Como começamos com um pool vazio, adicionamos apenas um servidor por meio da API, drenamos e depois o removemos, agora temos um pool vazio novamente.

Discussão

A API exclusiva do NGINX Plus permite que os servidores de aplicativos dinâmicos se adicionem e se removam da configuração do NGINX em tempo real. À medida que os servidores ficam online, eles podem se registrar no pool e o NGINX começará a enviar carga para ele.

Quando um servidor precisa ser removido, o servidor pode solicitar ao NGINX Plus para drenar suas conexões e, em seguida, remover-se do pool upstream antes de ser desligado.

Isso permite que a infraestrutura, por meio de alguma automação, aumente e diminua sem intervenção humana.

Veja também

[Documentação do Swagger da API NGINX Plus](#)

5.2 Usando o armazenamento de valores-chave com NGINX Plus

Problema

Você precisa do NGINX Plus para tomar decisões de gerenciamento de tráfego dinâmico com base na entrada de aplicativos.

Solução

Esta seção usará o exemplo de uma lista de bloqueio dinâmica como uma decisão de gerenciamento de tráfego.

Configure o armazenamento de valor-chave com reconhecimento de cluster e a API e, em seguida, adicione chaves e valores:

```
keyval_zone zona=blocklist:1M; keyval
$remote_addr $blocked zone=blocklist;

servidor {
    # ...
    local / { if ($
        bloqueado) { return
            403 'Proibido';

        } return 200 'OK';
    }

} servidor {
    # ...
    # Diretrivas que limitam o acesso à API
    # Veja o capítulo 6
```

```

localização /api
    { api write=on;
    }
}

```

Essa configuração do NGINX Plus usa o diretório keyval_zone para criar uma zona de memória compartilhada de armazenamento de chave-valor chamada blocklist e define um limite de memória de 1 MB. A diretiva keyval então mapeia o valor da chave, combinando o primeiro parâmetro \$remote_addr com uma nova variável chamada \$blocked da zona. Essa nova variável é usada para determinar se o NGINX Plus deve atender à solicitação ou retornar um código 403 Forbidden.

Depois de iniciar o servidor NGINX Plus com esta configuração, você pode enrolar a máquina local e esperar receber uma resposta 200 OK.

```
$ curl 'http://127.0.0.1/'
OK
```

Agora adicione o endereço IP da máquina local ao armazenamento de valor-chave com um valor de 1:

```
$ curl -X POST -d '{"127.0.0.1":"1"}' \
'http://127.0.0.1/api/3/http/keyvals/blocklist'
```

Esse comando curl envia uma solicitação HTTP POST com um objeto JSON contendo um objeto de valor-chave a ser enviado à zona de memória compartilhada da lista de bloqueio. O URI da API de armazenamento de chave-valor é formatado da seguinte forma:

```
/api/{version}/http/keyvals/{httpKeyvalZoneName}
```

O endereço IP da máquina local agora é adicionado à zona de valor-chave chamada lista de bloqueio com um valor de 1. Na próxima solicitação, o NGINX Plus procura o \$remote_addr na zona de valor-chave, encontra a entrada e mapeia o valor para a variável \$blocked. Essa variável é então avaliada na instrução if . Quando a variável tem um valor, o if é avaliado como True e o NGINX Plus retorna o código de retorno 403 Forbidden:

```
$ curl 'http://127.0.0.1/'
Proibido
```

Você pode atualizar ou excluir a chave fazendo uma solicitação do método PATCH :

```
$ curl -X PATCH -d '{"127.0.0.1":null}' \
'http://127.0.0.1/api/3/http/keyvals/blocklist'
```

O NGINX Plus exclui a chave se o valor for nulo e as solicitações retornarão novamente 200 OK.

Discussão

O armazenamento de valor-chave, um recurso exclusivo do NGINX Plus, permite que os aplicativos injetem informações no NGINX Plus. No exemplo fornecido, a variável \$remote_addr é usada para criar uma lista de bloqueio dinâmica. Você pode preencher o armazenamento de valor-chave com qualquer chave

que o NGINX Plus pode ter como variável – um cookie de sessão, por exemplo – e fornecer um valor externo ao NGINX Plus. No NGINX Plus R16, o armazenamento de valor-chave tornou-se compatível com cluster, o que significa que você precisa fornecer sua atualização de valor-chave para apenas um servidor NGINX Plus e todos eles receberão as informações.

No NGINX Plus R19, o armazenamento de chave-valor ativou um parâmetro de tipo , que permite a indexação de tipos específicos de chaves. Por padrão, o tipo é de valor string, onde ip e prefix também são opções. O tipo string não cria um índice e todas as solicitações de chave devem ser correspondências exatas, onde prefix permitirá correspondências parciais de chave, desde que o prefixo da chave seja uma correspondência. Um tipo de ip permite o uso da notação CIDR.

Em nosso exemplo, se tivéssemos especificado type=ip como um parâmetro para nossa zona, poderíamos ter fornecido um intervalo CIDR inteiro para bloquear, como 192.168.0.0/16 para bloquear todo o bloco de intervalo privado RFC 1918 ou 127.0. 0.1/32 para localhost que renderia o mesmo efeito demonstrado no exemplo.

Veja também

[Limites de largura de banda dinâmica](#)

5.3 Estendendo o NGINX com uma linguagem de programação comum

Problema

Você precisa do NGINX para executar alguma extensão personalizada usando uma linguagem de programação comum.

Solução

Antes de se preparar para escrever um módulo NGINX personalizado do zero em C, primeiro avalie se um dos outros módulos de linguagem de programação se adequará ao seu caso de uso. A linguagem de programação C é extremamente poderosa e performática. Existem, no entanto, muitos outros idiomas disponíveis como módulos que podem permitir a customização necessária. O NGINX introduziu o NGINScript (njs), que expõe o poder do Java Script na configuração do NGINX simplesmente habilitando um módulo. Módulos Lua e Perl também estão disponíveis.

Para começar com njs, instale o módulo njs e use o seguinte script njs, hello_world.js, para retornar “Hello World” quando chamado:

```
function hello(request)
    { request.return(200, "Olá mundo!");
}
```

Chame o script njs usando a seguinte configuração NGINX mínima:

```
módulos load_module/ngx_http_js_module.so;

eventos {}

http
{ js_include hello_world.js;

servidor
{ escuta 8000;

localização /
{ js_content olá;
}

}

}
```

A configuração NGINX acima habilita o módulo njs, inclui a biblioteca njs que construímos chamada hello_world.js e usamos a função hello para retornar uma resposta ao cliente. A função hello é chamada pela diretiva NGINX js_content. O objeto de solicitação fornecido à função njs possui muitos atributos que descrevem a solicitação e são capazes de manipular a resposta. O módulo njs é escrito e suportado pelo NGINX e é atualizado a cada versão do NGINX. Para uma referência atualizada, veja o link njs Documentation na seção Veja também.

Com esses módulos de linguagem, você importa um arquivo incluindo código ou define um bloco de código diretamente na configuração.

Para usar Lua, instale o módulo Lua e a seguinte configuração NGINX para definir um script Lua inline.

```
módulos load_module/ngx_http_lua_module.so;

eventos {}

http
{ servidor
{ escuta 8080;
local /
{ default_type text/html;
content_by_lua_block
{ ngx.say("olá, mundo")
}
}

}
```

O módulo Lua fornece sua própria API NGINX através de um objeto definido pelo módulo chamado ngx. Assim como o objeto de solicitação em njs, o objeto ngx possui atributos e métodos para descrever a solicitação e manipular a resposta.

Com o módulo Perl instalado, este exemplo usará Perl para definir uma variável NGINX do ambiente de tempo de execução.

```
módulos load_module/ngx_http_perl_module.so;

eventos {}

http
{ perl_set $app_endpoint 'sub { return $ENV{"APP_DNS_ENDPOINT"}; }'; servidor
 { escuta 8080; local / { proxy_pass http://$app_endpoint

}

}
}

}
```

O exemplo anterior demonstra que esses módulos de linguagem expõem mais funcionalidades do que apenas retornar uma resposta. A diretiva perl_set define uma variável NGINX para dados retornados de um script Perl. Este exemplo limitado simplesmente retorna uma variável de ambiente do sistema, que é usada como o terminal para o qual as solicitações de proxy.

Discussão

Os recursos habilitados pela extensibilidade do NGINX são infinitos. O NGINX é extensível com código personalizado por meio de módulos C, que podem ser compilados no NGINX ao compilar a partir da fonte ou carregados dinamicamente na configuração.

Módulos existentes que expõem a funcionalidade e a sintaxe de JavaScript (njs), Lua e Perl já estão disponíveis. Em muitos casos, a menos que distribua a funcionalidade NGINX personalizada para outros, esses módulos pré-existentes podem ser suficientes. Muitos scripts criados para esses módulos já existem na comunidade de código aberto.

Esta solução demonstrou o uso básico das linguagens de script njs, Lua e Perl disponíveis no NGINX e NGINX Plus. Seja para responder, definir uma variável, fazer uma subsolicitação ou definir uma reescrita complexa, esses módulos NGINX fornecem a capacidade.

Veja também

[Documentação do NGINScript](#)

[Instalação do módulo NGINX Plus njs](#)

[Instalação do módulo NGINX Plus Lua](#)

[Instalação do módulo NGINX Plus Perl](#)

[Documentação do módulo NGINX Lua](#)

[Documentação do módulo NGINX Perl](#)

5.4 Instalando com Puppet

Problema

Você precisa instalar e configurar o NGINX com Puppet para gerenciar as configurações do NGINX como código e estar em conformidade com o restante das configurações do Puppet.

Solução

Crie um módulo que instale o NGINX, gerencie os arquivos necessários e garanta que NGINX está em execução:

```
class nginx
  { package {"nginx": garantir => 'instalado',} serviço
    {"nginx": garantir => 'true', hasrestart => 'true',
     restart => '/etc/init.d/nginx reload ',

    } file { "nginx.conf": path
      => '/etc/nginx/nginx.conf', require =>
      Package['nginx'], notify => Service['nginx'],
      content => template('nginx /templates/
      nginx.conf.erb'), user=>'root', group=>'root', mode='0644';

    }
  }
```

Este módulo usa o utilitário de gerenciamento de pacotes para garantir que o pacote NGINX seja instalado. Ele também garante que o NGINX esteja em execução e habilitado no momento da inicialização. A configuração informa ao Puppet que o serviço tem um comando de reinicialização com a diretiva `hasrestart`, e podemos substituir o comando de reinicialização por um recarregamento NGINX.

O recurso de arquivo gerenciará e modelará o arquivo `nginx.conf` com a linguagem de modelagem Embedded Ruby (ERB). A modelagem do arquivo acontecerá após a instalação do pacote NGINX devido à diretiva `require`. No entanto, o recurso de arquivo notificará o serviço NGINX para recarregar devido à diretiva de notificação. O arquivo de configuração modelo não está incluído. No entanto, pode ser simples instalar um arquivo de configuração NGINX padrão ou muito complexo se estiver usando ERB ou Embedded Puppet (EPP) loops de linguagem de modelagem e substituição de variáveis.

Discussão

Puppet é uma ferramenta de gerenciamento de configuração baseada na linguagem de programação Ruby. Os módulos são criados em uma linguagem específica do domínio e chamados por meio de um arquivo de manifesto que define a configuração de um determinado servidor. Puppet pode ser executado em um servidor cliente

relacionamento ou configuração independente. Com o Puppet, o manifesto é executado no servidor e depois enviado ao agente. Isso é importante porque garante que o agente receba apenas a configuração destinada a ele e nenhuma configuração extra destinada a outros servidores. Existem muitos módulos públicos extremamente avançados disponíveis para o Puppet. Começar a partir destes módulos irá ajudá-lo a dar um salto inicial na sua configuração. Um módulo NGINX público do Vox Pupuli no GitHub modelará as configurações do NGINX para você.

Veja também

[Documentação da Marionete](#)
[Documentação do Pacote Puppet](#)
[Documentação do serviço de marionetes](#)
[Documentação do arquivo de marionetes](#)
[Documentação de modelagem de marionetes](#)
[Módulo Vox Pupuli NGINX](#)

5.5 Instalando com o Chef

Problema

Você precisa instalar e configurar o NGINX com o Chef para gerenciar as configurações do NGINX como código e estar em conformidade com o restante das configurações do Chef.

Solução

Crie um livro de receitas com uma receita para instalar o NGINX e defina os arquivos de configuração por meio de modelos e garanta que o NGINX seja recarregado após a configuração ser implementada. Segue um exemplo de receita:

```
pacote 'nginx' faz
  ação: install end

serviço 'nginx' fazer
  suporta :status => true, :restart => true, :reload => true action
  [ :start, :enable ] end

modelo 'nginx.conf' faça
  caminho "/etc/nginx.conf"
  fonte "nginx.conf.erb"
  proprietário 'raiz'
  o modo 'root'
  do grupo '0644'
  notifica :reload, 'service[nginx]', :delayed end
```

O bloco de pacotes instala o NGINX. O bloco de serviço garante que o NGINX seja iniciado e habilitado na inicialização e, em seguida, declara ao resto do Chef o que o serviço nginx suportará em relação às ações. O bloco de modelo modela um arquivo ERB e o coloca em /etc/nginx.conf com um proprietário e grupo de root. O bloco de modelo também define o modo como 644 e notifica o serviço nginx para recarregar, mas aguarda até o final da execução do Chef declarada pela instrução :delayed . O arquivo de configuração modelo não está incluído. No entanto, pode ser tão simples quanto um arquivo de configuração NGINX padrão ou muito complexo com loops de linguagem de modelagem ERB e substituição de variáveis.

Discussão

Chef é uma ferramenta de gerenciamento de configuração baseada em Ruby. O Chef pode ser executado em uma relação cliente-servidor ou configuração individual, agora conhecida como Chef Zero. Chef tem uma comunidade muito grande com muitos livros de receitas públicos chamados de Supermercado. Livros de receitas públicos do Supermercado podem ser instalados e mantidos por meio de um utilitário de linha de comando chamado Berkshelf. O Chef é extremamente capaz, e o que demonstramos é apenas uma pequena amostra. O livro de receitas público NGINX no supermercado é extremamente flexível e oferece as opções para instalar facilmente o NGINX a partir de um gerenciador de pacotes ou da fonte, e a capacidade de compilar e instalar muitos módulos diferentes, bem como modelar as configurações básicas.

Veja também

[Documentação do Chef](#)

[Pacote Chef](#)

[Serviço de Cozinheiro](#)

[Modelo Chef](#)

[Chef Supermercado para NGINX](#)

5.6 Instalando com o Ansible

Problema

Você precisa instalar e configurar o NGINX com o Ansible para gerenciar as configurações do NGINX como código e estar em conformidade com o restante das configurações do Ansible.

Solução

Crie um manual do Ansible para instalar o NGINX e gerenciar o arquivo nginx.conf. O seguinte é um arquivo de tarefa de exemplo para o manual para instalar o NGINX. Verifique se ele está em execução e modele o arquivo de configuração:

- nome: NGINX | Instalando o NGINX
- pacote: nome=nginx estado=presente

```
- nome: NGINX | Iniciando o NGINX
  serviço:
    nome: nginx
    estado: iniciado
    habilitado: sim

- name: Copie a configuração do nginx no local. template: src:
  nginx.conf.j2 dest: "/etc/nginx/nginx.conf" proprietário: root
```

```
grupo: modo
raiz: 0644
notificar: - recarregar
nginx
```

O bloco de pacotes instala o NGINX. O bloco de serviço garante que o NGINX seja iniciado e habilitado na inicialização. O bloco template modela um arquivo Jinja2 e coloca o resultado em /etc/nginx.conf com um proprietário e grupo de root. O bloco de modelo também define o modo para 644 e notifica o serviço nginx para recarregar. O arquivo de configuração modelo não está incluído. No entanto, pode ser tão simples quanto um arquivo de configuração NGINX padrão ou muito complexo com loops de linguagem de modelagem Jinja2 e substituição de variáveis.

Discussão

Ansible é uma ferramenta de gerenciamento de configuração amplamente utilizada e poderosa baseada em Python. A configuração das tarefas está em YAML e você usa a linguagem de modelagem Jinja2 para modelagem de arquivos. O Ansible oferece um servidor chamado Ansible Tower em um modelo de assinatura. No entanto, é comumente usado em máquinas locais ou para construir servidores diretamente para o cliente ou em um modelo autônomo. O Ansible fará o SSH em massa nos servidores e executará a configuração. Assim como outras ferramentas de gerenciamento de configuração, há uma grande comunidade de funções públicas. Ansible chama isso de Ansible Galaxy. Você pode encontrar funções muito sofisticadas para utilizar em seus manuais.

Veja também

[Função Ansible fornecida pelo NGINX](#)

[Documentação Ansible](#)

[Pacotes Ansible](#)

[Serviço Ansible](#)

[Modelo Ansible](#)

[Galáxia Ansible](#)

5.7 Instalando com SaltStack

Problema

Você precisa instalar e configurar o NGINX com SaltStack para gerenciar as configurações do NGINX como código e estar em conformidade com o restante de suas configurações do SaltStack.

Solução

Instale o NGINX através do módulo de gerenciamento de pacotes e gerencie os arquivos de configuração desejados. Veja a seguir um arquivo de estado de exemplo (arquivo de estado de salt [SLS]) que instalará o pacote nginx e garantirá que o serviço esteja em execução, habilitado na inicialização e recarregado se for feita uma alteração no arquivo de configuração:

```
nginx:  
  
pacote: - instalado  
serviço:  
    - name: nginx -  
      running - enable:  
      True - recarregar:  
      True - watch:  
  
      - arquivo: /etc/nginx/nginx.conf  
  
/etc/nginx/nginx.conf: arquivo: -  
gerenciado - fonte: salt://path/  
to/nginx.conf - usuário: root  
  
    - grupo: root -  
    template: jinja - modo: 644  
  
    - requer: -  
    pacote: nginx
```

Este é um exemplo básico de instalação do NGINX por meio de um utilitário de gerenciamento de pacotes e gerenciamento do arquivo nginx.conf. O pacote NGINX está instalado e o serviço está em execução e habilitado na inicialização. Com SaltStack, você pode declarar um arquivo gerenciado pelo Salt, como visto no exemplo, e modelado por muitas linguagens de template diferentes. O arquivo de configuração do modelo não está incluído. No entanto, pode ser tão simples quanto um arquivo de configuração NGINX padrão ou muito complexo com os loops de linguagem de modelagem Jinja2 e substituição de variáveis. Essa configuração também especifica que o NGINX deve ser instalado antes de gerenciar o arquivo devido à instrução require . Depois que o arquivo está no lugar, o NGINX é recarregado devido à diretiva watch no serviço e recarrega, em vez de reiniciar, porque a diretiva reload está definida como True.

Discussão

SaltStack é uma poderosa ferramenta de gerenciamento de configuração que define os estados do servidor em YAML. Módulos para SaltStack podem ser escritos em Python. Salt expõe a linguagem de modelagem Jinja2 para estados e também para arquivos. No entanto, para arquivos, existem muitas outras opções, como Mako, o próprio Python e outros.

SaltStack usa terminologia master minion para representar o relacionamento cliente-servidor. O lacaio também pode ser executado sozinho. A comunicação de transporte de lacaios mestre, no entanto, difere das outras e diferencia o SaltStack. Com Salt, você pode escolher ZeroMQ, TCP ou Reliable Asynchronous Event Transport (RAET) para transmissões ao agente Salt; ou você não pode usar um agente e o mestre pode usar SSH. Como a camada de transporte é, por padrão, assíncrona, o SaltStack é construído para poder entregar sua mensagem a um grande número de minions com baixa carga para o servidor mestre.

Veja também

[Módulo SaltStack NGINX](#)

[Documentação SaltStack](#)

[Pacotes Instalados SaltStack](#)

[Arquivos Gerenciados SaltStack](#)

[Modelagem SaltStack com Ninja](#)

5.8 Automatizando Configurações com Modelo Consul

Problema

Você precisa automatizar sua configuração do NGINX para responder às mudanças em seu ambiente por meio do uso do Consul.

Solução

Use o daemon consul-template e um arquivo de modelo para modelar o arquivo de configuração NGINX de sua escolha:

```
back-end upstream {{{range service "app.backend"}}} servidor
    {{.Address}};{{end}}
}
```

Este exemplo é um arquivo Consul Template que modela um bloco de configuração upstream.

Este modelo fará um loop pelos nós no Consul identificados como app.backend.

Para cada nó no Consul, o modelo produzirá uma diretiva de servidor com o endereço IP desse nó.

O daemon consul-template é executado por meio da linha de comando e pode ser usado para recarregar o NGINX toda vez que o arquivo de configuração for modelado com uma alteração:

```
# consul-template -consul consul.example.internal -template \ template:/etc/
nginx/conf.d/upstream.conf:"nginx -s reload"
```

Este comando instrui o daemon consul-template a se conectar a um Consul cluster em consul.example.internal e a usar um arquivo chamado template no diretório de trabalho atual para modelar o arquivo e enviar o conteúdo gerado para /etc/nginx/conf.d/upstream.conf e, em seguida, para recarregar o NGINX toda vez que o arquivo de modelo for alterado. O sinalizador -template recebe uma sequência do arquivo de modelo, o local de saída e o comando a ser executado após a ocorrência do processo de modelagem. Estas três variáveis são separadas por dois pontos. Se o comando que está sendo executado tiver espaços, certifique-se de colocá-lo entre aspas duplas. O sinalizador -consul informa ao daemon qual cluster do Consul deve ser conectar a.

Discussão

O Consul é uma poderosa ferramenta de descoberta de serviços e armazenamento de configuração. O Consul armazena informações sobre nós, bem como pares de valores-chave em uma estrutura semelhante a um diretório e permite a interação tranquila da API. O Consul também fornece uma interface DNS em cada cliente, permitindo pesquisas de nomes de domínio de nós conectados ao cluster. Um projeto separado que utiliza clusters Consul é o daemon consul-template ; esta ferramenta modela arquivos em resposta a mudanças em nós Consul, serviços ou pares chave-valor. Isso torna o Consul uma opção muito poderosa para automatizar o NGINX. Com o modelo cônsl , você também pode instruir o daemon a executar um comando depois que uma alteração no modelo ocorrer. Com isso, podemos recarregar a configuração do NGINX e permitir que sua configuração do NGINX ganhe vida junto com seu ambiente. Com Consul e consul-template, sua configuração NGINX pode ser tão dinâmica quanto seu ambiente. As informações de infraestrutura, configuração e aplicativo são armazenadas centralmente, e o consul-template pode se inscrever e refazer o modelo conforme necessário de maneira baseada em eventos. Com essa tecnologia, o NGINX pode se reconfigurar dinamicamente em reação à adição e remoção de servidores, serviços, versões de aplicativos e assim por diante.

Veja também

[Integração NGINX Plus Consul](#)

[Integração NGINX Consul](#)

[Página inicial do cônsl](#)

[Introdução ao Modelo de Cônsul](#)

[Modelo de Cônsl GitHub](#)

CAPÍTULO 6

Autenticação

6.0 Introdução

O NGINX é capaz de autenticar clientes. A autenticação de solicitações de clientes com descarregamentos do NGINX funciona e oferece a capacidade de impedir que solicitações não autenticadas cheguem aos servidores de aplicativos. Os módulos disponíveis para NGINX Open Source incluem autenticação básica e subsolicitações de autenticação. O módulo exclusivo NGINX Plus para verificação de JSON Web Tokens (JWTs) permite a integração com provedores de autenticação de terceiros que usam o padrão de autenticação OpenID Connect.

6.1 Autenticação básica HTTP

Problema

Você precisa proteger seu aplicativo ou conteúdo por meio de autenticação básica HTTP.

Solução

Gere um arquivo no seguinte formato, onde a senha é criptografada ou hash com um dos formatos permitidos:

```
#  
comentário  
nome1:senha1  
nome2:senha2:comentário nome3:senha3
```

O nome de usuário é o primeiro campo, a senha é o segundo campo e o delimitador é dois-pontos. Há um terceiro campo opcional, que você pode usar para comentar sobre cada usuário. O NGINX pode entender alguns formatos diferentes de senhas, um dos quais é se a senha é criptografada com a função C crypt(). Esta função é

exposto à linha de comando pelo comando openssl passwd . Com o openssl instalado, você pode criar strings de senha criptografadas usando o seguinte comando:

```
$ openssl passwd MinhaSenha1234
```

A saída será uma string que o NGINX pode usar em seu arquivo de senha.

Use as diretivas auth_basic e auth_basic_user_file em sua configuração NGINX para habilitar a autenticação básica:

```
local /  
    { auth_basic "Site privado";  
        auth_basic_user_file conf.d/passwd;  
    }
```

Você pode usar as diretivas auth_basic nos contextos HTTP, servidor ou local.

A diretiva auth_basic recebe um parâmetro de string, que é exibido na janela pop-up de autenticação básica quando um usuário não autenticado chega. O auth_basic_user_file especifica um caminho para o arquivo do usuário.

Para testar sua configuração, você pode usar curl com -u ou --user para criar um cabeçalho de autorização para a solicitação.

```
$ curl --user myuser:MyPassword1234 https://localhost
```

Discussão

Você pode gerar senhas de autenticação básicas de algumas maneiras e em alguns formatos diferentes, com vários graus de segurança. O comando htpasswd do Apache também pode gerar senhas. Ambos os comandos openssl e htpasswd podem gerar senhas com o algoritmo apr1 , que o NGINX também pode entender. A senha também pode estar no formato SHA-1 salgado que o Lightweight Directory Access Protocol (LDAP) e o Dovecot usam. NGINX suporta mais formatos e algoritmos de hash; no entanto, muitos deles são considerados inseguros porque podem ser facilmente derrotados por ataques de força bruta.

Você pode usar a autenticação básica para proteger o contexto de todo o host NGINX, servidores virtuais específicos ou até mesmo apenas blocos de local específicos. A autenticação básica não substituirá a autenticação do usuário para aplicativos da Web, mas pode ajudar a manter as informações privadas seguras. Nos bastidores, a autenticação básica é feita pelo servidor que retorna um código HTTP 401 não autorizado com o cabeçalho de resposta WWW-Authenticate. Este cabeçalho terá um valor de Basic realm="your string". Essa resposta faz com que o navegador solicite um nome de usuário e uma senha. O nome de usuário e a senha são concatenados e delimitados com dois pontos, depois codificados em base64 e enviados em um cabeçalho de solicitação chamado Autorização. O cabeçalho da solicitação de autorização especificará uma string codificada Basic e user:password . O servidor decodifica

o cabeçalho e verifica o auth_basic_user_file fornecido. Como a string de senha do nome de usuário é meramente codificada em base64, é recomendável usar HTTPS com autenticação básica.

6.2 Sub-solicitações de autenticação

Problema

Você tem um sistema de autenticação de terceiros para o qual deseja que as solicitações sejam autenticadas.

Solução

Use o http_auth_request_module para fazer uma solicitação ao serviço de autenticação para verificar a identidade antes de atender à solicitação:

```
local /privado/
    { auth_request /auth;
        auth_request_set $auth_status $upstream_status;
    }

local = /auth { interno;

    proxy_pass http://auth-server;
    proxy_pass_request_body desativado; proxy_set_header
        Comprimento do conteúdo
    ""; proxy_set_header X-Original-URI $request_uri;
}
```

A diretiva auth_request recebe um parâmetro URI que deve ser um local interno local. A diretiva auth_request_set permite que você defina variáveis da subsolicitação de autenticação.

Discussão

O http_auth_request_module habilita a autenticação em cada solicitação tratada pelo servidor NGINX. O módulo usará uma subsolicitação para determinar se a solicitação está autorizada a prosseguir. Uma subsolicitação ocorre quando o NGINX passa a solicitação para um local interno alternativo e observa sua resposta antes de encaminhar a solicitação para seu destino. O local de autenticação passa a solicitação original, incluindo o corpo e os cabeçalhos, para o servidor de autenticação. O código de status HTTP da subsolicitação é o que determina se o acesso é concedido ou não. Se a subsolicitação retornar com um código de status HTTP 200, a autenticação será bem-sucedida e a solicitação será atendida. Se a subsolicitação retornar HTTP 401 ou 403, o mesmo será devolvido para a solicitação original.

Se o seu serviço de autenticação não solicitar o corpo da solicitação, você poderá descartar o corpo da solicitação com a diretiva proxy_pass_request_body , conforme demonstrado. Esse

prática reduzirá o tamanho e o tempo da solicitação. Como o corpo da resposta é descartado, o cabeçalho Content-Length deve ser definido como uma string vazia. Se seu serviço de autenticação precisar saber o URI que está sendo acessado pela solicitação, você desejará colocar esse valor em um cabeçalho personalizado que seu serviço de autenticação verifica e verifica. Se houver coisas que você deseja manter da subsolicitação para o serviço de autenticação, como cabeçalhos de resposta ou outras informações, você pode usar a diretiva auth_request_set para criar novas variáveis a partir dos dados de resposta.

6.3 Validando JWTs com NGINX Plus

Problema

Você precisa validar um JWT antes que a solicitação seja processada com o NGINX Plus.

Solução

Use o módulo de autenticação HTTP JWT do NGINX Plus para validar a assinatura do token e incorporar declarações e cabeçalhos JWT como variáveis NGINX:

```
local /api/ { auth_jwt
    "api"; auth_jwt_key_file conf/
    keys.json;
}
```

Essa configuração permite a validação de JWTs para esse local. A diretiva auth_jwt recebe uma string, que é usada como o domínio de autenticação. O auth_jwt recebe um parâmetro de token opcional de uma variável que contém o JWT. Por padrão, o cabeçalho Authentication é usado de acordo com o padrão JWT. A diretiva auth_jwt também pode ser usada para cancelar os efeitos da autenticação JWT necessária de configurações herdadas. Para desativar a autenticação, passe a palavra-chave para a diretiva auth_jwt sem mais nada. Para cancelar os requisitos de autenticação herdados, passe a palavra-chave off para a diretiva auth_jwt com nada mais. O auth_jwt_key_file usa um único parâmetro. Esse parâmetro é o caminho para o arquivo de chave no formato JSON Web Key (JWK) padrão.

Discussão

O NGINX Plus é capaz de validar os tipos de tokens de assinatura da Web JSON, em oposição ao tipo de criptografia da Web JSON, em que todo o token é criptografado. O NGINX Plus é capaz de validar assinaturas assinadas com os algoritmos HS256, RS256 e ES256. Fazer com que o NGINX Plus valide o token pode economizar tempo e recursos necessários para fazer uma subsolicitação a um serviço de autenticação. O NGINX Plus decifra o cabeçalho JWT e a carga útil e captura os cabeçalhos e declarações padrão em

ded variáveis para seu uso. A diretiva auth_jwt pode ser usada nos contextos http, server, location e limit_except .

Veja também

[Documentação padrão RFC da assinatura da Web JSON](#)

[Documentação padrão RFC de algoritmos da Web JSON](#)

[Documentação padrão RFC do JSON Web Token](#)

[Autenticação NGINX Plus JWT](#)

[Blog NGINX detalhado](#)

6.4 Criando chaves da Web JSON

Problema

Você precisa de uma chave da Web JSON (JWK) para o NGINX Plus usar.

Solução

O NGINX Plus utiliza o formato JWK conforme especificado no padrão RFC. Esse padrão permite uma matriz de objetos-chave no arquivo JWK.

Veja a seguir um exemplo de como o arquivo de chave pode ser:

```
{"chaves": [
    {
        "kty": "octetSequence",
        "kid": "0001",
        "k": "OctetSequenceKeyValue"
    },
    {
        "kty": "EC",
        "kid": "0002",
        "crv": "P-256",
        "x": "XCoordinateValue", "y": "YCoordinateValue", "d": "PrivateExponent", "use": "sig"
    },
    {
        "kty": "RSA",
        "criança": "0003",
        "n": "Módulo", "e": "Expoente", "d": "Exponente Privado"
    }
]
```

O arquivo JWK mostrado demonstra os três tipos iniciais de chaves observados no padrão RFC. O formato dessas chaves também faz parte do padrão RFC. O atributo kty é o tipo de chave. Este arquivo mostra três tipos de chaves: a sequência de octetos (oct), a curva elíptica (EC) e o tipo RSA . O atributo kid é o ID da chave. Outros atributos para essas chaves são especificados no padrão para esse tipo de chave. Consulte a documentação RFC desses padrões para obter mais informações.

Discussão

Existem inúmeras bibliotecas disponíveis em muitas linguagens diferentes para gerar o JWK. É recomendável criar um serviço de chave que seja a autoridade central do JWK para criar e alternar seus JWKs em intervalos regulares. Para maior segurança, é recomendado tornar seus JWKs tão seguros quanto suas certificações SSL/TLS. Proteja seu arquivo de chave com permissões adequadas de usuário e grupo. Mantê-los na memória do seu host é a melhor prática. Você pode fazer isso criando um sistema de arquivos na memória como ramfs. Girar as teclas em intervalos regulares também é importante; você pode optar por criar um serviço de chaves que cria chaves públicas e privadas e as oferece ao aplicativo e ao NGINX por meio de uma API.

Veja também

[Documentação padrão RFC da chave da Web JSON](#)

6.5 Validar tokens da Web JSON com NGINX Plus

Problema

Você deseja validar os tokens da Web JSON com o NGINX Plus.

Solução

Use o módulo JWT que vem com o NGINX Plus para proteger um local ou servidor e instrua a diretiva auth_jwt a usar \$cookie_auth_token como o token a ser validado:

```
location /private/ { auth_jwt
    "Google Oauth" token=$cookie_auth_token; auth_jwt_key_file /
        etc/nginx/google_certs.jwk;
}
```

Essa configuração direciona o NGINX Plus para proteger o caminho /private/ URI com validação JWT. O Google OAuth 2.0 OpenID Connect usa o cookie auth_token em vez do token de portador padrão. Assim, você deve instruir o NGINX a procurar o token neste cookie em vez do local padrão do NGINX Plus. o

auth_jwt_key_file location é definido como um caminho arbitrário, que é uma etapa que abordamos na [Receita 6.6](#).

Discussão

Esta configuração demonstra como você pode validar um JWT do Google OAuth 2.0 OpenID Connect com NGINX Plus. O módulo de autenticação NGINX Plus JWT para HTTP é capaz de validar qualquer JWT que adere à especificação RFC para JSON Web Signature, permitindo instantaneamente que qualquer autoridade SSO que utilize JWTs seja validada na camada NGINX Plus. O protocolo OpenID 1.0 é uma camada sobre o protocolo de autenticação OAuth 2.0 que adiciona identidade, permitindo o uso de JWTs para comprovar a identidade do usuário que envia a solicitação. Com a assinatura do token, o NGINX Plus pode validar que o token não foi modificado desde que foi assinado. Dessa forma, o Google está usando um método de assinatura assíncrono e possibilita a distribuição de JWKS públicos, mantendo seu segredo JWK privado.

Veja também

[Autenticação de clientes de API com JWT e NGINX Plus](#)

6.6 Obtendo e armazenando em cache conjuntos de chaves da Web JSON automaticamente com o NGINX Plus

Problema

Você deseja que o NGINX Plus solicite automaticamente o JSON Web Key Set (JWKS) de um provedor e o armazene em cache.

Solução

Utilize uma zona de cache e a diretiva auth_jwt_key_request para manter sua chave atualizada automaticamente:

```
proxy_cache_path /data/nginx/cache levels=1 keys_zone=foo:10m;  
  
servidor  
{ # ...  
  
local /  
{ auth_jwt "site fechado";  
auth_jwt_key_request /jwks_uri;  
}  
  
local = /jwks_uri { interno;  
proxy_cache foo;
```

```
        proxy_pass https://idp.example.com/keys;
    }
}
```

Neste exemplo, a diretiva auth_jwt_key_request instrui o NGINX Plus a recuperar o JWKS de uma subsolicitação interna. A subsolicitação é direcionada para /jwks_uri, que fará o proxy da solicitação para um provedor de identidade. A solicitação é armazenada em cache por um padrão de 10 minutos para limitar a sobrecarga.

Discussão

No NGINX Plus R17, a diretiva auth_jwt_key_request foi introduzida. Esse recurso permite que o servidor NGINX Plus atualize dinamicamente seus JWKs quando uma solicitação é feita. Um método de subsolicitação é usado para buscar os JWKs, o que significa que o local para o qual a diretiva aponta deve ser local para o servidor NGINX Plus. No exemplo, o local da subsolicitação foi bloqueado para garantir que apenas as solicitações internas do NGINX Plus seriam atendidas. Um cache também foi usado para garantir que a solicitação de recuperação de JWKs seja feita apenas com a frequência necessária e não sobrecarregue o provedor de identidade. A diretiva auth_jwt_key_request é válida nos contextos http, server, location e limit_except .

Veja também

[Autenticação de clientes de API com JWT e NGINX Plus](#)

6.7 Autenticar usuários por meio de SSO OpenID Connect existente com NGINX Plus

Problema

Você deseja integrar o NGINX Plus com um provedor de identidade OpenID Connect (OIDC).

Solução

Esta solução consiste em vários aspectos de configuração e um pouco de código NGINScript. O provedor de identidade (IdP) deve oferecer suporte ao OpenID Connect 1.0. O NGINX Plus atuará como uma parte de retransmissão de seu OIDC em um fluxo de código de autorização.

NGINX Inc., mantém um repositório público GitHub contendo configuração e código como uma implementação de referência da integração OIDC com NGINX Plus. O link a seguir para o repositório contém instruções atualizadas sobre como configurar a implementação de referência com seu próprio IdP.

Discussão

Esta solução está simplesmente vinculada a uma implementação de referência para garantir que você, leitor, tenha a solução mais atualizada. A referência fornecida configura o NGINX Plus como uma parte de retransmissão para um fluxo de código de autorização para OpenID Connect 1.0. Quando solicitações não autenticadas de recursos protegidos são feitas ao NGINX Plus nessa configuração, o NGINX Plus primeiro redireciona a solicitação ao IdP. O IdP leva o cliente por meio de seu próprio fluxo de login e retorna o cliente ao NGINX Plus com um código de autenticação. O NGINX Plus então se comunica diretamente com o IdP para trocar o código de autenticação por um conjunto de tokens de identificação. Esses tokens são validados usando JWTs e armazenados no armazenamento de valor-chave do NGINX Plus. Ao usar o armazenamento de valor-chave, os tokens são disponibilizados para todos os nós NGINX Plus em uma configuração de alta disponibilidade (HA). Durante esse processo, o NGINX Plus gera um cookie de sessão para o cliente que é usado como chave para pesquisar o token no armazenamento de valor-chave. O cliente recebe um redirecionamento com o cookie para o recurso inicial solicitado. As solicitações subsequentes são validadas usando o cookie para pesquisar o ID Token no armazenamento de valor-chave do NGINX Plus.

Esse recurso permite a integração com a maioria dos principais provedores de identidade, incluindo CA Single Sign On (anteriormente SiteMinder), ForgeRock OpenAM, Keycloak, Okta, OneLogin e Ping Identity. O OIDC como padrão é extremamente relevante na autenticação – os provedores de identidade mencionados acima são apenas um subconjunto das integrações possíveis.

Veja também

- [Blog NGINX detalhado no OpenID Connect](#)
- [OpenID Connect](#)
- [Implementação de referência do NGINX OpenID Connect](#)

CAPÍTULO 7

Controles de segurança

7.0 Introdução

A segurança é feita em camadas e deve haver várias camadas em seu modelo de segurança para que ele seja realmente reforçado. Neste capítulo, passamos por muitas maneiras diferentes de proteger seus aplicativos da web com NGINX e NGINX Plus. Você pode usar muitos desses métodos de segurança em conjunto uns com os outros para ajudar a fortalecer a segurança. A seguir estão várias seções que exploram os recursos de segurança do NGINX e do NGINX Plus que podem ajudar a fortalecer seu aplicativo. Você pode notar que este capítulo não aborda um dos maiores recursos de segurança do NGINX, o módulo ModSecurity 3.0 NGINX, que transforma o NGINX em um firewall de aplicativo da Web (WAF). Para saber mais sobre os recursos do WAF, baixe o [ModSecurity 3.0](#) e [NGINX: Guia de início rápido](#).

7.1 Acesso baseado no endereço IP

Problema

Você precisa controlar o acesso com base no endereço IP do cliente.

Solução

Use o módulo de acesso HTTP ou stream para controlar o acesso a recursos protegidos:

```
local /admin/ { nega
    10.0.0.1; permitir
    10.0.0.0/20; permitir
    2001:0db8::/32; negar
    tudo;
}
```

O bloco de localização fornecido permite acesso de qualquer endereço IPv4 em 10.0.0.0/20, exceto 10.0.0.1, permite acesso de endereços IPv6 na sub-rede 2001:0db8::/32 e retorna um 403 para solicitações originadas de qualquer outro endereço. As diretivas de permissão e negação são válidas nos contextos HTTP, servidor e local, bem como no contexto de fluxo e servidor para TCP/UDP. As regras são verificadas em sequência até que uma correspondência seja encontrada para o endereço remoto.

Discussão

A proteção de recursos e serviços valiosos na Internet deve ser feita em camadas. A funcionalidade NGINX fornece a capacidade de ser uma dessas camadas. A diretiva deny bloqueia o acesso a um determinado contexto, enquanto a diretiva allow pode ser usada para permitir subconjuntos do acesso bloqueado. Você pode usar endereços IP, IPv4 ou IPv6, intervalos de blocos de roteamento entre domínios sem classes (CIDR), a palavra-chave all e um soquete Unix. Normalmente, ao proteger um recurso, pode-se permitir um bloqueio de endereços IP internos e negar o acesso de todos.

7.2 Permitindo Compartilhamento de Recursos de Origem Cruzada

Problema

Você está servindo recursos de outro domínio e precisa permitir o compartilhamento de recursos entre origens (CORS) para permitir que os navegadores utilizem esses recursos.

Solução

Altere os cabeçalhos com base no método de solicitação para habilitar o CORS:

```
map $request_method $cors_method {
    OPÇÕES 11;
    PEGUE 1;
    POST 1;
    padrão 0;

} server { #
    location /
    { if ($cors_method
        ~ '1') { add_header 'Access-Control-
        Allow-Methods'
            'OBTER, POSTAR, OPÇÕES';
        add_header 'Acesso-Controle-Permitir-Origem'
            '*.*.example.com'; add_header 'Access-Control-Allow-
        Headers'
            'DNT,
            Mantenha vivo,
            Agente de usuário,
            X-Solicitado-Com,
```

```
Se-Modificado-Desde,  
Controle de Cache,  
Tipo de conteúdo';  
  
} if ($cors_method = '11')  
{ add_header 'Access-Control-Max-Age' 1728000;  
add_header 'Tipo de conteúdo' 'texto/simples; conjunto de  
caracteres=UTF-8'; add_header 'Content-Length' 0; retornar  
204;  
}  
}  
}
```

Há muita coisa acontecendo neste exemplo, que foi condensado usando um mapa para agrupar os métodos GET e POST . O método de solicitação OPTIONS retorna uma solicitação de comprovação ao cliente sobre as regras CORS desse servidor. Os métodos OPTIONS, GET e POST são permitidos em CORS. A configuração do cabeçalho Access-Control-Allow-Origin permite que o conteúdo servido por este servidor também seja usado em páginas de origens que correspondam a este cabeçalho. A solicitação de comprovação pode ser armazenada em cache no cliente por 1.728.000 segundos ou 20 dias.

Discussão

Recursos como JavaScript fazem CORS quando o recurso que eles estão solicitando é de um domínio diferente do seu. Quando uma solicitação é considerada de origem cruzada, o navegador é obrigado a obedecer às regras do CORS. O navegador não usará o recurso se não tiver cabeçalhos que permitam especificamente seu uso. Para permitir que nossos recursos sejam usados por outros subdomínios, temos que definir os cabeçalhos CORS, o que pode ser feito com a diretiva add_header . Se a solicitação for GET, HEAD ou POST com tipo de conteúdo padrão e a solicitação não tiver cabeçalhos especiais, o navegador fará a solicitação e verificará apenas a origem. Outros métodos de solicitação farão com que o navegador faça a solicitação de comprovação para verificar os termos do servidor ao qual obedecerá para esse recurso. Se você não definir esses cabeçalhos adequadamente, o navegador apresentará um erro ao tentar utilizar esse recurso.

7.3 Criptografia do lado do cliente

Problema

Você precisa criptografar o tráfego entre o servidor NGINX e o cliente.

Solução

Utilize um dos módulos SSL, como `ngx_http_ssl_module` ou `ngx_stream_ssl_module` para criptografar o tráfego:

```
http { # Todas as diretivas usadas abaixo também são válidas no stream
    servidor
        { escuta 8443 ssl;
            ssl_certificate /etc/nginx/ssl/example.crt;
            ssl_certificate_key /etc/nginx/ssl/example.key;
        }
}
```

Essa configuração configura um servidor para escutar em uma porta criptografada com SSL/TLS, 8443. A diretiva `ssl_certificate` define o certificado e a cadeia opcional que é fornecida ao cliente. A diretiva `ssl_certificate_key` define a chave usada pelo NGINX para descriptografar solicitações e criptografar respostas. Várias configurações de negociação SSL/TLS são padronizadas para predefinições seguras para a data de lançamento da versão NGINX.

Discussão

Camadas de transporte seguro são a maneira mais comum de criptografar informações em trânsito. No momento em que este artigo foi escrito, o protocolo TLS é preferível ao protocolo SSL. Isso porque as versões 1 a 3 do SSL agora são consideradas inseguras. Embora o nome do protocolo possa ser diferente, o TLS ainda estabelece uma camada de soquete segura. O NGINX permite que seu serviço proteja as informações entre você e seus clientes, o que, por sua vez, protege o cliente e sua empresa. Ao usar um certificado assinado por CA, você precisa concatenar o certificado com a cadeia de autoridade de certificação. Quando você concatena seu certificado e a cadeia, seu certificado deve estar acima do arquivo da cadeia concatenada. Se sua autoridade de certificação forneceu vários arquivos como certificados intermediários para a cadeia, há uma ordem na qual eles são colocados em camadas. Consulte a documentação do fornecedor de certificados para o pedido.

Veja também

[Página TLS do lado do servidor Mozilla](#)

[Gerador de configuração SSL Mozilla](#)

[Teste sua configuração SSL com o teste de servidor SSL do SSL Labs](#)

7.4 Criptografia avançada do lado do cliente

Problema

Você tem necessidades avançadas de configuração de criptografia cliente-servidor.

Solução

Os módulos HTTP e stream SSL para NGINX permitem o controle completo do handshake SSL/TLS aceito. Certificados e chaves podem ser fornecidos ao NGINX, por meio de caminho de arquivo ou valor de variável. O NGINX apresenta ao cliente uma lista aceita de protocolos, cifras e tipos de chave, de acordo com sua configuração. O padrão mais alto entre o cliente e o servidor NGINX é negociado. O NGINX pode armazenar em cache o resultado da negociação SSL/TLS do servidor cliente por um período de tempo.

O seguinte demonstra intencionalmente muitas opções ao mesmo tempo para ilustrar a complexidade disponível da negociação cliente-servidor:

```
http { # Todas as diretivas usadas abaixo também são válidas no stream
    servidor
        { escuta 8443 ssl; #
            Definir protocolo aceito e cifra ssl_protocols
            TLSv1.2 TLSv1.3; ssl_ciphers HIGH:!aNULL!:
            MD5;

            # Cadeia de certificados RSA carregada do arquivo
            ssl_certificate /etc/nginx/ssl/example.crt; # Chave de
            criptografia RSA carregada do arquivo ssl_certificate_key /
            etc/nginx/ssl/example.pem;

            # Certificado de curva elíptica do valor da variável
            ssl_certificate $ecdsa_cert;
            # Chave de curva elíptica como variável de caminho
            de arquivo ssl_certificate_key data:$ecdsa_key_path;

            # Cache de negociação cliente-servidor
            ssl_session_cache shared:SSL:10m;
            ssl_session_timeout 10m;
        }
    }
}
```

O servidor aceita as versões do protocolo SSL TLSv1.2 e TLSv1.3. As cifras aceitas são definidas como HIGH, que é uma macro para o padrão mais alto, negações explícitas são demonstradas para aNULL e MD5 pela denotação do !.

Dois conjuntos de pares de chave de certificado são usados. Os valores passados para as diretivas NGINX demonstram diferentes maneiras de fornecer valores de chave de certificado NGINX. Uma variável é interpretada como um caminho para um arquivo. Quando prefixado com dados: o valor de uma variável é interpretado como um valor direto. Vários formatos de chave de certificado podem ser fornecidos para oferecer

compatibilidade reversa para o cliente. O padrão mais forte capaz pelo cliente e aceito pelo servidor será o resultado da negociação.



Se a chave SSL/TLS for exposta como uma variável de valor direto, ela poderá ser registrada ou exposta pela configuração. Certifique-se de ter controles rígidos de alteração e acesso ao expor o valor da chave como uma variável.

O cache de sessão SSL e o tempo limite permitem que os processos de trabalho do NGINX armazenem em cache e armazenem parâmetros de sessão por um determinado período de tempo. Os processos de trabalho do NGINX compartilham esse cache entre si como processos em uma única instância, mas não entre máquinas. Existem muitas outras opções de cache de sessão que podem ajudar no desempenho ou na segurança de todos os tipos de casos de uso. Você pode usar as opções de cache de sessão em conjunto. No entanto, especificar um sem o padrão desativará o cache de sessão interno padrão.

Discussão

Neste exemplo avançado, o NGINX fornece ao cliente as opções SSL/TLS do TLS versão 1.2 ou 1.3, algoritmos de cifra altamente conceituados e a capacidade de usar chaves formatadas RSA ou ECC (Elliptic Curve Cryptography). O mais forte dos protocolos, cifras e formatos de chave que o cliente é capaz é o resultado da negociação. A configuração instrui o NGINX a armazenar em cache a negociação por um período de 10 minutos com a alocação de memória disponível de 10 MB.

Nos testes, os certificados ECC foram considerados mais rápidos do que os certificados RSA de força equivalente. O tamanho da chave é menor, o que resulta na capacidade de servir mais conexões SSL/TLS e com handshakes mais rápidos. O NGINX permite configurar vários certificados e chaves e, em seguida, fornecer o certificado ideal para o navegador do cliente. Isso permite que você aproveite a tecnologia mais recente, mas ainda atenda clientes mais antigos.



O NGINX está criptografando o tráfego entre ele e o cliente neste exemplo. A conexão com servidores upstream também pode ser criptografada, no entanto. A negociação entre o NGINX e o servidor upstream é demonstrada na receita Upstream Encryption.

Veja também

[Página TLS do lado do servidor Mozilla](#)

[Gerador de configuração SSL Mozilla](#)

[Teste sua configuração SSL com o SSL Labs SSL Server Test](#)

7.5 Criptografia Upstream

Problema

Você precisa criptografar o tráfego entre o NGINX e o serviço upstream e definir regras de negociação específicas para regulamentos de conformidade ou o serviço upstream está fora de sua rede segura.

Solução

Use as diretivas SSL do módulo proxy HTTP para especificar as regras SSL:

```
local / {  
    proxy_pass https://upstream.example.com;  
    proxy_ssl_verify ativado; proxy_ssl_verify_depth  
    2; proxy_ssl_protocols TLSv1.2;  
}
```

Essas diretivas de proxy definem regras SSL específicas para o NGINX obedecer. As diretivas configuradas garantem que o NGINX verifique se o certificado e a cadeia no serviço upstream são válidos com até dois certificados de profundidade. A diretiva proxy_ssl_protocols especifica que o NGINX usará apenas o TLS versão 1.2. Por padrão, o NGINX não verifica os certificados upstream e aceita todas as versões do TLS.

Discussão

As diretivas de configuração para o módulo de proxy HTTP são vastas e, se você precisar criptografar o tráfego upstream, deverá pelo menos ativar a verificação. Você pode fazer proxy sobre HTTPS simplesmente alterando o protocolo no valor passado para a diretiva proxy_pass . No entanto, isso não valida o certificado upstream. Outras diretivas, como proxy_ssl_certificate e proxy_ssl_certificate_key, permitem bloquear a criptografia upstream para maior segurança. Você também pode especificar proxy_ssl_crl ou uma lista de revogação de certificados, que lista os certificados que não são mais considerados válidos. Essas diretivas de proxy SSL ajudam a fortalecer os canais de comunicação do seu sistema em sua própria rede ou na Internet pública.

7.6 Protegendo um local

Problema

Você precisa proteger um bloco de localização usando um segredo.

Solução

Use o módulo de link seguro e a diretiva `secure_link_secret` para restringir o acesso aos recursos aos usuários que possuem um link seguro:

```
localização /recursos
{ secure_link_secret mySecret; if
  ($secure_link = "") { return 403; }

  reescrever ^ /secured/$secure_link;
}

local /seguro/ { interno;
  raiz /var/www;

}
```

Essa configuração cria um bloco de local interno e voltado para o público. O bloco de localização público /resources retornará um 403 Forbidden, a menos que o URI de solicitação inclua uma string de hash md5 que possa ser verificada com o segredo fornecido à diretiva `secure_link_secret`. A variável `$secure_link` é uma string vazia, a menos que o hash no URI seja verificado.

Discussão

Proteger recursos com um segredo é uma ótima maneira de garantir que seus arquivos estejam protegidos. O segredo é usado em conjunto com o URI. Essa string é então md5 hash e o resumo hexadecimal desse md5 hash é usado no URI. O hash é colocado no link e avaliado pelo NGINX. O NGINX conhece o caminho para o arquivo que está sendo solicitado porque está no URI após o hash. O NGINX também conhece seu segredo, pois ele é fornecido pela diretiva `secure_link_secret`. O NGINX é capaz de validar rapidamente o hash md5 e armazenar o URI na variável `$secure_link`. Se o hash não puder ser validado, a variável será definida como uma string vazia. É importante observar que o argumento passado para o `secure_link_secret` deve ser uma string estática; não pode ser uma variável.

7.7 Gerando um Link Seguro com um Segredo

Problema

Você precisa gerar um link seguro do seu aplicativo usando um segredo.

Solução

O módulo de link seguro no NGINX aceita o resumo hexadecimal de uma string com hash md5 , onde a string é uma concatenação do caminho URI e do segredo. Com base na última seção, [Receita 7.6](#), criaremos o link seguro que funcionará com o anterior.

exemplo de configuração dado que existe um arquivo presente em /var/www/secured/index.html. Para gerar o resumo hexadecimal do hash md5 , podemos usar o comando Unix openssl :

```
$ echo -n 'index.htmlmeuSegredo' | openssl md5 -hex (stdin) =
a53bee08a4bf0bbea978ddf736363a12
```

Aqui mostramos o URI que estamos protegendo, index.html, concatenado com nosso segredo, mySecret. Essa string é passada para o comando openssl para gerar um resumo hexadecimal md5 .

Veja a seguir um exemplo do mesmo resumo de hash sendo construído em Python usando a biblioteca hashlib incluída na biblioteca padrão do Python:

```
import hashlib
hashlib.md5(b'index.htmlmySecret').hexdigest()
'a53bee08a4bf0bbea978ddf736363a12'
```

Agora que temos esse resumo de hash, podemos usá-lo em uma URL. Nossa exemplo será www.example.com fazendo uma solicitação para o arquivo /var/www/secured/index.html por meio de nosso local /resources. Nossa URL completa será o seguinte:

```
www.example.com/resources/a53bee08a4bf0bbea978ddf736363a12/
index.html
```

Discussão

A geração do resumo pode ser feita de várias maneiras, em vários idiomas. Coisas para lembrar: o caminho do URI vai antes do segredo, não há retornos de carro na string e use o resumo hexadecimal do hash md5 .

7.8 Protegendo um local com data de expiração

Problema

Você precisa proteger um local com um link que expire em algum momento futuro e seja específico para um cliente.

Solução

Utilize as outras diretivas incluídas no módulo de link seguro para definir um tempo de expiração e usar variáveis em seu link seguro:

```
localização /recursos { raiz /
    var/www; link_seguro
    $arg_md5,$arg_expires; secure_link_md5
    "$secure_linkExpires$uri$remote_addrmySecret"; if ($secure_link = "") { return
    403; } if ($secure_link = "0") { return 410; }

}
```

A diretiva `secure_link` recebe dois parâmetros separados por vírgula. O primeiro parâmetro é a variável que contém o hash md5 . Este exemplo usa um argumento HTTP de md5. O segundo parâmetro é uma variável que contém a hora em que o link expira no formato Unix epoch time. A diretiva `secure_link_md5` usa um único parâmetro que declara o formato da string que é usada para construir o hash md5 . Como a outra configuração, se o hash não for validado, a variável `$secure_link` é definida como uma string vazia. No entanto, com esse uso, se o hash corresponder, mas o tempo expirou, a variável `$secure_link` será definida como 0.

Discussão

Esse uso de proteger um link é mais flexível e parece mais limpo do que o `secure_link_secret` mostrado na [Receita 7.6](#). Com essas diretivas, você pode usar qualquer número de variáveis disponíveis para o NGINX na string com hash. O uso de variáveis específicas do usuário na string de hash fortalecerá sua segurança, pois os usuários não poderão negociar links para recursos protegidos. É recomendado usar uma variável como `$remote_addr` ou

`$http_x_forwarded_for`, ou um cabeçalho de cookie de sessão gerado pelo aplicativo. Os argumentos para `secure_link` podem vir de qualquer variável que você preferir, e eles podem ser nomeados da forma que melhor se adequar. As condições são: Você tem acesso? Você está acessando dentro do prazo? Se não tiver acesso: Proibido. Se você tiver acesso, mas estiver atrasado: Desaparecido. O HTTP 410, Gone, funciona muito bem para links expirados porque a condição deve ser considerada permanente.

7.9 Gerando um Link Expirando

Problema

Você precisa gerar um link que expira.

Solução

Gere um carimbo de data/hora para o tempo de expiração no formato Unix epoch. Em um sistema Unix, você pode testar usando a data conforme demonstrado a seguir:

```
$ date -d "2020-12-31 00:00" +%s --utc  
1609372800
```

Em seguida, você precisará concatenar sua string de hash para corresponder à string configurada com a diretiva `secure_link_md5` . Nesse caso, nossa string a ser usada será 1293771600/resources/index.html127.0.0.1 mySecret. O hash md5 é um pouco diferente de apenas um resumo hexadecimal. É um hash md5 em formato binário, codificado em base64, com sinais de mais (+) convertidos em hífens (-), barras (/) traduzidos em sublinhados (_) e sinais de igual (=) removidos. O seguinte é um exemplo em um sistema Unix:

```
$ echo -n '1609372800/resources/index.html127.0.0.1 meuSegredo' \
| openssl md5 -binário \
openssl base64 | tr +/ -_ \
tr -d =
```

TG6ck3OpAttQ1d7jW3JOcw

Agora que temos nosso hash, podemos usá-lo como argumento junto com a data de expiração:

```
/resources/index.html?md5=TG6ck3OpAttQ1d7jW3JOcw&expires=1609372
800'
```

Veja a seguir um exemplo mais prático em Python utilizando um tempo relativo para a expiração, configurando o link para expirar uma hora após a geração. No momento da escrita, este exemplo funciona com Python 2.7 e 3.x utilizando a Python Standard Library:

```
de datetime import datetime, timedelta de base64
import b64encode import hashlib

# Definir ambiente vars
resource = b'/resources/index.html'
remote_addr = b'127.0.0.1' host =
b'www.example.com' mysecret = b'mySecret'

# Gerar timestamp de expiração
agora = datetime.utcnow() expire_dt =
= agora + timedelta(horas=1) expire_epoch =
str.encode(expire_dt.strftime('%s'))

# md5 hash a string não
codificada = expire_epoch + resource + remote_addr + mysecret md5hashed
= hashlib.md5(uncoded).digest()

# Base64 codifica e transforma a string b64 =
b64encode(md5hashed) unpadded_b64url =
b64.replace(b'+', b'-') .replace(b'/', b'_') .replace(b'=',
b')

# Formate e gere o link linkformat = " {}
{}?md5={}?expires={}" securelink =
linkformat.format( host.decode(),
resource.decode(), unpadded_b64url.decode(),
expire_epoch. decodificar()

) imprimir (link seguro)
```

Discussão

Com esse padrão, podemos gerar um link seguro em um formato especial que pode ser usado em URLs. O segredo fornece segurança por meio do uso de uma variável que nunca é enviada ao cliente. Você pode usar quantas outras variáveis forem necessárias para proteger o local. md5 hashing e codificação base64 são comuns, leves e estão disponíveis em quase todos os idiomas.

7.10 Redirecionamentos HTTPS

Problema

Você precisa redirecionar solicitações não criptografadas para HTTPS.

Solução

Use uma regravação para enviar todo o tráfego HTTP para HTTPS:

```
servidor
{
    escuta 80 default_server;
    escute [::]:80 default_server;
    nome do servidor
    _; return 301 https://$host$request_uri;
}
```

Essa configuração atende na porta 80 como o servidor padrão para IPv4 e IPv6 e para qualquer nome de host. A instrução return retorna um redirecionamento permanente 301 para o servidor HTTPS no mesmo host e URI de solicitação.

Discussão

É importante sempre redirecionar para HTTPS quando apropriado. Você pode descobrir que não precisa redirecionar todas as solicitações, mas apenas aquelas com informações confidenciais sendo transmitidas entre cliente e servidor. Nesse caso, você pode querer colocar a instrução return apenas em locais específicos, como /login.

7.11 Redirecionando para HTTPS onde o SSL/TLS é encerrado antes do NGINX

Problema

Você precisa redirecionar para HTTPS, no entanto, você encerrou o SSL/TLS em uma camada anterior ao NGINX.

Solução

Use o cabeçalho X-Forwarded-Proto comum para determinar se você precisa redirecionar:

```
servidor
{
    escuta 80 default_server;
    escute [::]:80 default_server;
    nome do servidor
    _; if ($http_x_forwarded_proto = 'http') {
        return 301 https://$host$request_uri;
    }
}
```

Essa configuração é muito parecida com redirecionamentos HTTPS. Porém, nesta configuração estamos redirecionando apenas se o cabeçalho X-Forwarded-Proto for igual a HTTP.

Discussão

É um caso de uso comum que você pode encerrar o SSL/TLS em uma camada na frente do NGINX.

Uma razão pela qual você pode fazer algo assim é economizar nos custos de computação. No entanto, você precisa garantir que cada solicitação seja HTTPS, mas a camada que termina com SSL/TLS não tem a capacidade de redirecionar. Ele pode, no entanto, definir cabeçalhos de proxy. Essa configuração funciona com camadas como o Amazon Web Services Elastic Load Balancer (AWS ELB), que descarregará SSL/TLS sem custo adicional. Este é um truque útil para garantir que seu tráfego HTTP esteja protegido.

7.12 Segurança de Transporte Estrita HTTP

Problema

Você precisa instruir os navegadores a nunca enviar solicitações por HTTP.

Solução

Use o aprimoramento HTTP Strict Transport Security (HSTS) definindo o Strict

Cabeçalho de segurança de transporte :

```
add_header Strict-Transport-Security max-age=31536000;
```

Essa configuração define o cabeçalho Strict-Transport-Security para uma idade máxima de um ano. Isso instruirá o navegador a sempre fazer um redirecionamento interno quando houver tentativas de solicitações HTTP para este domínio, para que todas as solicitações sejam feitas por HTTPS.

Discussão

Para alguns aplicativos, uma única solicitação HTTP capturada por um ataque man-in-the-middle pode ser o fim da empresa. Se uma postagem de formulário contendo informações confidenciais for

enviado por HTTP, o redirecionamento HTTPS do NGINX não salvará você; o dano está feito. Esse aprimoramento de segurança opcional informa ao navegador para nunca fazer uma solicitação HTTP e, portanto, a solicitação nunca é enviada sem criptografia.

Veja também

[RFC-6797 HTTP Strict Transport Security](#)
[Folha de dicas do OWASP HSTS](#)

7.13 Satisfazendo qualquer número de métodos de segurança

Problema

Você precisa fornecer várias maneiras de passar a segurança para um site fechado.

Solução

Use a diretiva satisfazer para instruir o NGINX de que você deseja satisfazer um ou todos os métodos de segurança usados:

```
localização /  
    { satisfaz qualquer;  
  
        permitir 192.168.1.0/24;  
        negar tudo;  
  
        auth_basic "site fechado";  
        auth_basic_user_file conf/htpasswd;  
    }
```

Essa configuração informa ao NGINX que o usuário que solicita a localização / precisa satisfazer um dos métodos de segurança: ou a solicitação precisa se originar do bloco CIDR 192.168.1.0/24 ou ser capaz de fornecer um nome de usuário e senha que possam ser encontrados no arquivo conf/htpasswd. A diretiva satisfazer aceita uma das duas opções: qualquer ou todos.

Discussão

A diretiva satisfazer é uma ótima maneira de oferecer várias maneiras de autenticação em seu aplicativo da web. Ao especificar qualquer para a diretiva satisfazer , o usuário deve atender a um dos desafios de segurança. Ao especificar tudo para a diretiva satisfazer , o usuário deve atender a todos os desafios de segurança. Esta diretiva pode ser usada em conjunto com o http_access_module detalhado na [Receita 7.1](#), o http_auth_basic_module detalhado na [Receita 6.1](#), o http_auth_request_module detalhado na [Receita 6.2](#) e o http_auth_jwt_module detalhado na [Receita 6.3](#). A segurança só é verdadeiramente segura se for feita

em várias camadas. A diretiva satisfazer o ajudará a conseguir isso para locais e servidores que exigem regras de segurança profundas.

7.14 DDoS de camada de aplicativo dinâmico NGINX Plus Mitigação

Problema

Você precisa de uma solução dinâmica de mitigação de negação de serviço distribuída (DDoS).

Solução

Use o NGINX Plus para criar um limite de taxa com reconhecimento de cluster e uma lista de bloqueio automática:

```
limit_req_zone $remote_addr zone=per_ip:1M taxa=100r/s sincronização;
    # Limite de taxa com reconhecimento de cluster

limit_req_status 429;

keyval_zone zona=sinbin:1M tempo limite=600 sincronização;
    # "sin bin" com reconhecimento de cluster com
    # TTL de 10 minutos

keyval $remote_addr $in_sinbin zona=sinbin;
    # Preencha $in_sinbin com #
    # endereços IP de clientes correspondentes

servidor {
    ouça 80;
    local / {
        if ($in_sinbin) {
            defina $limit_rate 50; # Restringir a largura de banda de clientes ruins
        }

        limit_req zona=per_ip;
            # Aplique o limite de taxa aqui
        error_page 429 = @send_to_sinbin;
            # Excesso de clientes são movidos para
            # este local
        proxy_pass http://my_backend;
    }

    localização @send_to_sinbin {
        reescrever ^ /api/3/http/keyvals/sinbin break;
            # Defina o URI do valor de
            chave # "sin bin"
        proxy_method POST;
        proxy_set_body {'$remote_addr':'1'};
        proxy_pass http://127.0.0.1:80;
    }

    local /api/
}
```

```

    api write=on; #
    diretivas para controlar o acesso à API
}
}

```

Discussão

Essa solução usa um limite de taxa sincronizado por meio de um armazenamento de chave-valor sincronizado para responder dinamicamente a ataques DDoS e mitigar seus efeitos. O parâmetro sync fornecido para as diretivas limit_req_zone e keyval_zone sincroniza a zona de memória compartilhada com outras máquinas no cluster NGINX Plus ativo-ativo. Este exemplo identifica clientes que enviam mais de 100 solicitações por segundo, independentemente de qual nó NGINX Plus recebe a solicitação. Quando um cliente excede o limite de taxa, seu endereço IP é adicionado a um armazenamento de valor-chave “sin bin” fazendo uma chamada para a API NGINX Plus. O bin sin é sincronizado em todo o cluster. Solicitações adicionais de clientes no bin sin estão sujeitas a um limite de largura de banda muito baixo, independentemente de qual nó NGINX Plus as recebe. Limitar a largura de banda é preferível a rejeitar solicitações porque não sinaliza claramente ao cliente que a mitigação de DDoS está em vigor. Após 10 minutos, o cliente é automaticamente removido da lixeira.

7.15 Instalando e configurando o módulo NGINX Plus App Protect

Problema

Você precisa instalar e configurar o módulo NGINX Plus App Protect.

Solução

Siga o [guias de instalação do NGINX Plus App Protect](#) para sua plataforma. Certifique-se de não pular a parte sobre a instalação de assinaturas do App Protect do repositório separado.

Certifique-se de que o App Protect Module seja carregado dinamicamente pelo NGINX Plus usando a diretiva load_module no contexto principal e habilitado usando as diretivas app_protect_* .

```

usuário nginx;
trabalhador_processos auto;

load_module modules/ngx_http_app_protect_module.so;

# ... Outras diretivas de contexto principais

http
{
    app_protect_enable ativado;
}

```

```

    app_protect_policy_file "/etc/nginx/AppProtectTransparentPolicy.json";
    app_protect_security_log_enable ativado; app_protect_security_log "/etc/nginx/log-
default.json"
        syslog: servidor=127.0.0.1:515;
    # ... Outras diretivas de contexto http
}

```

Neste exemplo, a diretiva `app_protect_enable` definida como `on` ativou o módulo para o contexto atual. Essa diretiva e todas as seguintes são válidas no contexto HTTP, bem como nos contextos Servidor e Local com HTTP. A diretiva `app_protect_policy_file` aponta para um arquivo de política do App Protect que definiremos a seguir; se não for definida, a política padrão será usada. O log de segurança é configurado em seguida e requer um servidor de log remoto. Para o exemplo, nós o configuramos para o ouvinte `syslog` local. A diretiva `app_protect_security_log` recebe dois parâmetros; o primeiro é um arquivo JSON que define as configurações de log e o segundo é um destino de fluxo de log. O arquivo de configurações de log será mostrado posteriormente nesta seção.

Crie um arquivo de Política de Proteção de Aplicativo e nomeie-o como `/etc/nginx/AppProtectTransparentPolicy.json`:

```
{
    "policy":
        { "name": "transparent_policy",
            "template": { "name": "POLICY_TEMPLATE_NGINX_BASE" },
            "applicationLanguage": "utf-8", "enforcementMode":
                "transparent"
        }
}
```

Esse arquivo de política configura a política padrão do NGINX App Protect usando um modelo, definindo o nome da política como `transparent_policy` e definindo o Modo de imposição como transparente, o que significa que o NGINX Plus registrará, mas não bloqueará. O modo transparente é ótimo para testar novas políticas antes de colocá-las em prática.

Habilite o bloqueio alterando o `enforcementMode` para bloqueio. Esse arquivo de política pode ser denominado `/etc/nginx/AppProtectTransparentPolicy.json`. Para alternar entre os arquivos, atualize a diretiva `app_protect_policy_file` em sua configuração do NGINX Plus.

```
{
    "policy":
        { "name": "blocking_policy",
            "template": { "name": "POLICY_TEMPLATE_NGINX_BASE" },
            "applicationLanguage": "utf-8", "enforcementMode": "blocking"
        }
}
```

Para habilitar alguns dos recursos de proteção do App Protect, habilite algumas violações:

```
{
  "policy": {
    "name": "blocking_policy",
    "template": { "name": "POLICY_TEMPLATE_NGINX_BASE" },
    "applicationLanguage": "utf-8", "enforcementMode": "blocking",
    "blocking-settings" : { "violações": [ {

      "name": "VIOL_JSON_FORMAT",
      "alarm": verdadeiro, "block":
      verdadeiro
    },
    {
      "name": "VIOL_PARAMETER_VALUE_METACHAR",
      "alarm": verdadeiro, "block": falso

    }
  ]
}
}
}
```

O exemplo acima demonstra a adição de duas violações à nossa política. Observe que VIOL_PARAMETER_VALUE_METACHAR não está definido para bloquear, mas apenas alarme; enquanto VIOL_JSON_FORMAT é definido como bloqueio e alarme. Essa funcionalidade permite a substituição do enforcementMode padrão quando definido como bloqueio. Quando enforcementMode é definido como transparente, a configuração de imposição padrão tem precedência.

Configure um arquivo de log do NGINX Plus, chamado /etc/nginx/log-default.json:

```
{
  "filter": [
    { "request_type": "all"
  },
  "content": {
    "format": "default",
    "max_request_size": "qualquer",
    "max_message_size": "5k"
  }
}
```

Esse arquivo foi definido na configuração do NGINX Plus pela diretiva app_protect_security_log e é necessário para o registro do App Protect.

Discussão

Esta solução demonstra a base da configuração do módulo NGINX Plus App Protect. O módulo App Protect permite um conjunto completo de definições de Web Application Firewall (WAF). Essas definições derivam da funcionalidade Advanced F5 Application Security em F5. Este conjunto abrangente de assinaturas de ataque WAF foi amplamente testado e comprovado em campo. Ao adicionar isso a uma instalação do NGINX Plus, ele renderiza o melhor da F5 Application Security com a agilidade da plataforma NGINX.

Uma vez que o módulo esteja instalado e habilitado, a maior parte da configuração é feita em um arquivo de política. Os arquivos de política nesta seção mostraram como habilitar o bloqueio ativo, monitoramento passivo, modo transparente, bem como explicações sobre substituições dessa funcionalidade com violações. As violações são apenas um tipo de proteção oferecida. Outras proteções incluem Conformidade HTTP, Técnicas de Evasão, Assinaturas de Ataque, Tecnologias de Servidor, Data Guard e muito mais. Para recuperar os logs do App Protect, é necessário usar o formato de log NGINX Plus e enviar os logs para um serviço de escuta remota, um arquivo ou /dev/stderr.

Se você estiver usando o NGINX Controller ADC, poderá habilitar os recursos do NGINX App Protect WAF por meio do componente NGINX Controllers App Security e visualizar as métricas do WAF por meio da interface da web.

Veja também

[Guia de administração do controlador NGINX](#)

[Guia de configuração do controlador NGINX](#)

[Documentação de referência de política declarativa do controlador NGINX](#)

CAPÍTULO 8

HTTP/2

8.0 Introdução

HTTP/2 é uma revisão importante do protocolo HTTP. Muito do trabalho feito nesta versão foi focado na camada de transporte, como habilitar a multiplexação completa de solicitação e resposta em uma única conexão TCP. As eficiências foram obtidas com o uso de compactação em campos de cabeçalho HTTP e foi adicionado suporte para priorização de solicitações. Outra grande adição ao protocolo foi a capacidade do servidor de enviar mensagens para o cliente. Este capítulo detalha a configuração básica para habilitar HTTP/2 no NGINX, bem como configurar a chamada de procedimento remoto de código aberto (gRPC) do Google e o suporte a push de servidor HTTP/2.

8.1 Configuração Básica

Problema

Você deseja aproveitar o HTTP/2.

Solução

Ative o HTTP/2 em seu servidor NGINX:

```
servidor
    { escuta 443 ssl http2 default_server;

        ssl_certificate      servidor.crt;
        ssl_certificate_key server.key;
        # ...
    }
```

Discussão

Para ativar o HTTP/2, basta adicionar o parâmetro `http2` à diretiva `listen`. O problema, entretanto, é que, embora o protocolo não exija que a conexão seja encapsulada em SSL/TLS, algumas implementações de clientes HTTP/2 suportam apenas HTTP/2 em uma conexão criptografada. Outra ressalva é que a especificação HTTP/2 bloqueou vários conjuntos de cifras TLS 1.2 e, portanto, falhará no handshake. As cifras que o NGINX usa por padrão não estão na lista de bloqueio. A negociação de protocolo de camada de aplicativo do TLS permite que a camada de aplicativo negocie qual protocolo deve ser usado na conexão segura para evitar viagens de ida e volta adicionais. Para testar se sua configuração está correta, você pode instalar um plug-in para os navegadores Chrome e Firefox que indica quando um site está usando HTTP/2 ou na linha de comando com o utilitário `nghtp`.

Veja também

[Cifras bloqueadas HTTP/2 RFC](#)

[Plug-in do indicador Chrome HTTP/2 e SPDY](#)

[Complemento do indicador HTTP/2 do Firefox](#)

8,2 gRPC

Problema

Você precisa encerrar, inspecionar, rotear ou balancear a carga das chamadas de método gRPC.

Solução

Use NGINX para fazer proxy de conexões gRPC.

```
servidor
  { escuta 80 http2;
    local / {
      grpc_pass grpc://backend.local:50051;
    }
  }
```

Nessa configuração, o NGINX está escutando na porta 80 o tráfego HTTP/2 não criptografado e fazendo proxy desse tráfego para uma máquina chamada `backend.local` na porta 50051. A diretiva `grpc_pass` instrui o NGINX a tratar a comunicação como uma chamada gRPC. O `grpc://` na frente da localização do nosso servidor de backend não é necessário; no entanto, indica diretamente que a comunicação de back-end não está criptografada.

Para utilizar a criptografia TLS entre o cliente e o NGINX e encerrar essa criptografia antes de passar as chamadas para o servidor de aplicativos, ative SSL e HTTP/2, como você fez na primeira seção:

```
servidor
    { escuta 443 ssl http2 default_server;

        ssl_certificate server.crt;
        ssl_certificate_key server.key; local / {

            grpc_pass grpc://backend.local:50051;
        }
    }
```

Essa configuração encerra o TLS no NGINX e passa a comunicação gRPC para o aplicativo por meio de HTTP/2 não criptografado.

Para configurar o NGINX para criptografar a comunicação gRPC com o servidor de aplicativos, fornecendo tráfego criptografado de ponta a ponta, basta modificar a diretiva `grpc_pass` para especificar `grpcs://` antes das informações do servidor (observe a adição do `s` denotando comunicação segura):

```
grpc_pass grpcs://backend.local:50051;
```

Você também pode usar o NGINX para rotear chamadas para diferentes serviços de back-end com base no gRPC URI, que inclui o pacote, o serviço e o método. Para isso, utilize o diretiva de localização .

```
localização /meupacote.serviço1 {
    grpc_pass grpc://$grpc_service1;

} localização /meupacote.serviço2 {
    grpc_pass grpc://$grpc_service2;

} local /
    root /usr/share/nginx/html; índice
    índice.html índice.htm;
}
```

Este exemplo de configuração usa a diretiva `location` para rotear o tráfego HTTP/2 de entrada entre dois serviços gRPC separados, bem como um local para servir conteúdo estático. As chamadas de método para o serviço `mypackage.service1` são direcionadas para o valor da variável `grpc_service1` que pode conter um nome de host ou IP e uma porta opcional. As chamadas para `mypackage.service2` são direcionadas para o valor da variável `grpc_service2`. A localização / captura qualquer outra solicitação HTTP e fornece conteúdo estático. Isso demonstra como o NGINX é capaz de servir gRPC e não gRPC sob o mesmo endpoint HTTP/2 e rotear de acordo.

O balanceamento de carga das chamadas gRPC também é semelhante ao tráfego HTTP não gRPC:

```
grpcservers upstream
    { server backend1.local:50051;
      servidor backend2.local:50051;

    } server
        { escuta 443 ssl http2 default_server;

          ssl_certificate         servidor.crt;
          ssl_certificate_key server.key; local /
          { grpc_pass grpc://grpcservers;

        }
    }
```

O bloco upstream funciona exatamente da mesma maneira para gRPC e para outro tráfego HTTP. A única diferença é que o upstream é referenciado por grpc_pass.

Discussão

O NGINX é capaz de receber, fazer proxy, balancear a carga, rotear e encerrar a criptografia para chamadas gRPC. O módulo gRPC permite que o NGINX defina, altere ou descarte cabeçalhos de chamada gRPC, defina tempos limite para solicitações e defina especificações de SSL/TLS upstream. À medida que o gRPC se comunica pelo protocolo HTTP/2, você pode configurar o NGINX para aceitar tráfego da Web gRPC e não gRPC no mesmo endpoint.

8.3 Envio de servidor HTTP/2

Problema

Você precisa enviar antecipadamente o conteúdo para o cliente.

Solução

Use o recurso de envio de servidor HTTP/2 do NGINX.

```
servidor
    { escuta 443 ssl http2 default_server;

      ssl_certificate server.crt;
      ssl_certificate_key server.key; root /usr/
      share/nginx/html;

      local = /demo.html {
          http2_push/style.css;
          http2_push/image1.jpg;
      }
    }
```

Discussão

Para usar o push do servidor HTTP/2, seu servidor deve estar configurado para HTTP/2, como é feito na [Receita 7.1](#). A partir daí, você pode instruir o NGINX a enviar arquivos específicos preventivamente com a diretiva `http2_push`. Essa diretiva usa um parâmetro, o caminho URI completo do arquivo para enviar para o cliente.

O NGINX também pode enviar recursos automaticamente para os clientes se os aplicativos proxy incluírem um cabeçalho de resposta HTTP chamado Link. Este cabeçalho é capaz de instruir o NGINX a pré-carregar os recursos especificados. Para habilitar esse recurso, adicione `http2_push_preload` em; para a configuração do NGINX.

CAPÍTULO 9

Streaming de mídia sofisticado

9.0 Introdução

Este capítulo abrange streaming de mídia com NGINX nos formatos MPEG-4 (MP4) ou Flash Video (FLV). O NGINX é amplamente usado para distribuir e transmitir conteúdo para as massas. O NGINX oferece suporte a formatos padrão do setor e tecnologias de streaming, que serão abordados neste capítulo. O NGINX Plus permite a capacidade de fragmentar conteúdo em tempo real com o módulo HTTP Live Stream (HLS), bem como a capacidade de fornecer HTTP Dynamic Streaming (HDS) de mídia já fragmentada. O NGINX permite limites de largura de banda nativamente, e os recursos avançados do NGINX Plus oferecem limitação de taxa de bits, permitindo que seu conteúdo seja entregue da maneira mais eficiente, reservando os recursos dos servidores para alcançar a maioria dos usuários.

9.1 Servindo MP4 e FLV

Problema

Você precisa transmitir mídia digital, originada em MP4 ou FLV.

Solução

Designe um bloco de localização HTTP para veicular vídeos .mp4 ou .flv. O NGINX transmitirá a mídia usando downloads progressivos ou pseudostreaming HTTP e buscará suporte:

```
http
{ servidor
  { #...
    local /vídeos/ {
```

```

        mp4;

    } localização ~ \.flv$ { flv;
}

}
}
}

```

O bloco de localização de exemplo informa ao NGINX que os arquivos no diretório de vídeos estão no tipo de formato MP4 e podem ser transmitidos com suporte a download progressivo. O segundo bloco de localização instrui o NGINX de que todos os arquivos que terminam em .flv estão no formato FLV e podem ser transmitidos com suporte a pseudostreaming HTTP.

Discussão

O streaming de arquivos de vídeo ou áudio no NGINX é tão simples quanto uma única diretiva. O download progressivo permite que o cliente inicie a reprodução da mídia antes que o arquivo tenha terminado o download. O NGINX suporta a busca de uma parte não baixada da mídia em ambos os formatos.

9.2 Transmissão com HLS com NGINX Plus

Problema

Você precisa oferecer suporte ao HTTP Live Streaming (HLS) para conteúdo codificado em H.264/AAC empacotado em arquivos MP4.

Solução

Utilize o módulo HLS do NGINX Plus com segmentação, empacotamento e multiplexação em tempo real, com controle sobre o buffer de fragmentação e muito mais, como encaminhamento
Argumentos HLS:

```

localização /hls/{hls;
    # Use o manipulador HLS para gerenciar solicitações

    # Servir conteúdo do seguinte alias de localização /var/
    www/video;

    # Parâmetros HLS
    hls_fragment 4s; hls_buffers 10
    10m; hls_mp4_buffer_size 1m;
    hls_mp4_max_buffer_size 5m;

}

```

Esse bloco de localização direciona o NGINX para transmitir mídia HLS para fora do diretório /var/www/video, fragmentando a mídia em segmentos de 4 segundos. O número de HLS buffs

fers é definido como 10 com um tamanho de 10 MB. O tamanho inicial do buffer MP4 é definido como 1 MB com um máximo de 5 MB.

Discussão

O módulo HLS disponível no NGINX Plus oferece a capacidade de transmultiplexar arquivos de mídia MP4 em tempo real. Existem muitas diretivas que lhe dão controle sobre como sua mídia é fragmentada e armazenada em buffer. O bloco de localização deve ser configurado para servir a mídia como um fluxo HLS com o manipulador HLS. A fragmentação HLS é definida em número de segundos, instruindo o NGINX a fragmentar a mídia por duração. A quantidade de dados armazenados em buffer é definida com a diretiva `hls_buffers` especificando o número de buffers e o tamanho. O cliente tem permissão para iniciar a reprodução da mídia após uma certa quantidade de buffer acumulada, especificada pelo `hls_mp4_buffer_size`. No entanto, um buffer maior pode ser necessário porque os metadados sobre o vídeo podem exceder o tamanho do buffer inicial. Esse valor é limitado pelo `hls_mp4_max_buffer_size`.

Essas variáveis de buffer permitem que o NGINX otimize a experiência do usuário final. Escolher os valores corretos para essas diretrizes requer conhecer o público-alvo e sua mídia. Por exemplo, se a maior parte de sua mídia for grandes arquivos de vídeo e seu público-alvo tiver alta largura de banda, você pode optar por um tamanho máximo de buffer maior e uma fragmentação de comprimento maior. Isso permitirá que os metadados sobre o conteúdo sejam baixados inicialmente sem erros e que seus usuários recebam fragmentos maiores.

9.3 Transmissão com HDS com NGINX Plus

Problema

Você precisa oferecer suporte ao HTTP Dynamic Streaming (HDS) da Adobe que já foi fragmentado e separado dos metadados.

Solução

Use o suporte do NGINX Plus para o módulo de arquivos FLV fragmentados (F4F) para oferecer Streaming adaptável para seus usuários:

```
local /video/ { alias /
    var/www/transformed_video; f4f;
    f4f_buffer_size 512k;
}
```

O exemplo instrui o NGINX Plus a servir mídia fragmentada anteriormente de um local no disco para o cliente usando o módulo NGINX Plus F4F. O tamanho do buffer para o arquivo de índice (.f4x) é definido como 512 KB.

Discussão

O módulo NGINX Plus F4F permite que o NGINX forneça mídia anteriormente fragmentada para usuários finais. A configuração de tal é tão simples quanto usar o manipulador f4f dentro de um bloco de localização HTTP. A diretiva f4f_buffer_size configura o tamanho do buffer para o arquivo de índice desse tipo de mídia.

9.4 Limites de largura de banda com NGINX Plus

Problema

Você precisa limitar a largura de banda para clientes de streaming de mídia downstream sem afetar a experiência de visualização.

Solução

Utilize o suporte limitador de taxa de bits do NGINX Plus para arquivos de mídia MP4:

```
localização /vídeo/
{
    mp4;
    mp4_limit_rate_após 15s;
    mp4_limit_rate 1.2;
}
```

Essa configuração permite que o cliente downstream faça download por 15 segundos antes de aplicar um limite de taxa de bits. Após 15 segundos, o cliente pode baixar mídia a uma taxa de 120% da taxa de bits, o que permite que o cliente baixe sempre mais rápido do que reproduz.

Discussão

A limitação de taxa de bits do NGINX Plus permite que seu servidor de streaming limite a largura de banda dinamicamente, com base na mídia que está sendo servida, permitindo que os clientes baixem o quanto eles precisam para garantir uma experiência de usuário perfeita. O manipulador MP4 descrito na [Receita 9.1](#) designa esse bloco de localização para transmitir formatos de mídia MP4. As diretivas de limitação de taxa, como mp4_limit_rate_after, informam ao NGINX para limitar o tráfego apenas após um período de tempo especificado, em segundos. A outra diretiva envolvida na limitação de taxa de MP4 é mp4_limit_rate, que especifica a taxa de bits na qual os clientes podem fazer download em relação à taxa de bits da mídia. Um valor de 1 fornecido à diretiva mp4_limit_rate especifica que o NGINX deve limitar a largura de banda (1 para 1) à taxa de bits da mídia. Fornecer um valor de mais de 1 para a diretiva mp4_limit_rate permitirá que os usuários baixem mais rápido do que assistem, para que possam armazenar em buffer a mídia e assistir sem problemas durante o download.

CAPÍTULO 10

Implantações de nuvem

O advento dos provedores de nuvem mudou o cenário da hospedagem de aplicativos da web.

Um processo como o provisionamento de uma nova máquina costumava levar horas a meses; agora, você pode criar um com apenas um clique ou uma chamada de API. Esses provedores de nuvem alugam suas máquinas virtuais, chamadas de infraestrutura como serviço (IaaS), ou soluções de software gerenciadas, como bancos de dados, por meio de um modelo de pagamento por uso, o que significa que você paga apenas pelo que usa. Isso permite que os engenheiros construam ambientes inteiros para testes a qualquer momento e os destruam quando não forem mais necessários.

Esses provedores de nuvem também permitem que os aplicativos sejam dimensionados horizontalmente com base na necessidade de desempenho a qualquer momento. Este capítulo abrange implantações básicas do NGINX e NGINX Plus em algumas das principais plataformas de provedores de nuvem.

Problema

Você precisa automatizar a configuração dos servidores NGINX na Amazon Web Services (AWS) para que as máquinas possam se provisionar automaticamente.

Solução

Utilize EC2 UserData , bem como uma Amazon Machine Image (AMI) pré-preparada. Crie uma imagem de máquina da Amazon com NGINX e qualquer pacote de software de suporte instalado. Utilize UserData do Amazon Elastic Compute Cloud (EC2) para configurar qualquer configuração específica do ambiente em tempo de execução.

Discussão

Existem três padrões de pensamento ao provisionar na AWS:

Provisionar na

inicialização Inicie a partir de uma imagem comum do Linux e execute o gerenciamento de configuração ou scripts de shell no momento da inicialização para configurar o servidor. Esse padrão é lento para iniciar e pode ser propenso a erros.

AMIs totalmente

preparadas Configure totalmente o servidor e grave uma AMI para usar. Este padrão inicializa muito rápido e com precisão. No entanto, é menos flexível ao ambiente ao seu redor e a manutenção de muitas imagens pode ser complexa.

AMIs parcialmente preparadas

É uma mistura dos dois mundos. Parcialmente preparado é onde os requisitos de software são instalados e gravados em uma AMI, e a configuração do ambiente é feita no momento da inicialização. Esse padrão é flexível em comparação com um padrão totalmente preparado e rápido em comparação com uma solução de provisionamento na inicialização.

Independentemente de você optar por preparar parcialmente ou totalmente suas AMIs, você desejará automatizar esse processo. Para construir um pipeline de compilação de AMI, é recomendável usar algumas ferramentas:

Gerenciamento de configuração As

ferramentas de gerenciamento de configuração definem o estado do servidor no código, como qual versão do NGINX deve ser executada e qual usuário deve ser executado, qual resolvedor de DNS usar e para quem o proxy upstream. Esse código de gerenciamento de configuração pode ser controlado por fonte e ter versão como um projeto de software. Algumas ferramentas populares de gerenciamento de configuração são Puppet, Chef, Ansible e SaltStack, que foram descritas no [Capítulo 5](#).

O Packer da HashiCorp Packer é

usado para automatizar a execução do gerenciamento de configuração em praticamente qualquer plataforma de virtualização ou nuvem e para gravar uma imagem de máquina se a execução for bem-sucedida.

O Packer basicamente cria uma máquina virtual (VM) na plataforma de sua escolha, então o SSH na VM, executa qualquer provisionamento que você especificar e grava uma imagem.

Você pode utilizar o Packer para executar a ferramenta de gerenciamento de configuração e gravar de forma confiável uma imagem de máquina de acordo com sua especificação.

Para provisionar configurações ambientais no momento da inicialização, você pode utilizar o Amazon EC2 UserData para executar comandos na primeira vez que a instância for inicializada. Se você estiver usando o método parcialmente preparado, poderá utilizá-lo para configurar itens baseados em ambiente no momento da inicialização. Exemplos de configurações baseadas em ambiente podem ser quais nomes de servidor escutar, resolver usar, nome de domínio para proxy ou conjunto de servidores upstream para começar.

UserData é uma string codificada em base64 que é baixada na primeira inicialização e execução. UserData pode ser tão simples quanto um arquivo de ambiente acessado por

outros processos de bootstrap em sua AMI, ou pode ser um script escrito em qualquer idioma existente na AMI. É comum que UserData seja um script bash que especifica variáveis, ou faz download de variáveis, para passar ao gerenciamento de configuração. O gerenciamento de configuração garante que o sistema esteja configurado corretamente, modela os arquivos de configuração com base nas variáveis de ambiente e recarrega os serviços. Após a execução do UserData, sua máquina NGINX deve estar completamente configurada, de maneira muito confiável.

10.2 Roteamento para nós NGINX sem um AWS ELB

Problema

Você precisa rotear o tráfego para vários nós NGINX ativos ou criar um conjunto de failover ativo-passivo para obter alta disponibilidade sem um balanceador de carga na frente do NGINX.

Solução

Use o serviço DNS do Amazon Route 53 para rotear para vários nós NGINX ativos ou configurar verificações de integridade e failover entre um conjunto ativo-passivo de nós NGINX.

Discussão

O DNS equilibra a carga entre os servidores há muito tempo; mudar para a nuvem não muda isso. O serviço Route 53 da Amazon fornece um serviço DNS com muitos recursos avançados, todos disponíveis por meio de uma API. Todos os truques típicos de DNS estão disponíveis, como vários endereços IP em um único registro A e registros A ponderados.

Ao executar vários nós NGINX ativos, você desejará usar um desses recursos de registro A para distribuir a carga por todos os nós. O algoritmo round-robin é usado quando vários endereços IP são listados para um único registro A. Uma distribuição ponderada pode ser usada para distribuir a carga de forma desigual, definindo pesos para cada endereço IP do servidor em um registro A.

Um dos recursos mais interessantes do Route 53 é sua capacidade de verificação de integridade. Você pode configurar o Route 53 para monitorar a integridade de um endpoint estabelecendo uma conexão TCP ou fazendo uma solicitação com HTTP ou HTTPS. A verificação de integridade é altamente configurável com opções para IP, nome de host, porta, caminho de URI, taxas de intervalo, monitoramento e geografia. Com essas verificações de integridade, o Route 53 pode tirar um IP de rotação se começar a falhar. Você também pode configurar o Route 53 para fazer failover para um registro secundário em caso de falha, o que alcançaria uma configuração ativa-passiva e altamente disponível.

O Route 53 possui um recurso de roteamento baseado em geografia que permitirá que você roteie seus clientes para o nó NGINX mais próximo a eles, com a menor latência. Ao fazer o roteamento por geografia, seu cliente é direcionado para o local físico saudável mais próximo. Ao executar

vários conjuntos de infraestrutura em uma configuração ativa-ativa, você pode fazer failover automaticamente para outro local geológico por meio do uso de verificações de integridade.

Ao usar o DNS do Route 53 para rotear seu tráfego para nós NGINX em um grupo de Auto Scaling, você desejará automatizar a criação e a remoção de registros DNS. Para automatizar a adição e remoção de máquinas NGINX ao Route 53 à medida que seus nós NGINX são dimensionados, você pode usar os ganchos de ciclo de vida do Auto Scaling da Amazon para acionar scripts na própria caixa NGINX ou scripts executados independentemente no Amazon Lambda. Esses scripts usariam a interface de linha de comando da Amazon (CLI) ou o kit de desenvolvimento de software (SDK) para fazer interface com a API do Amazon Route 53 para adicionar ou remover o IP da máquina NGINX e a verificação de integridade configurada durante a inicialização ou antes de ser encerrada.

Veja também

[Balanceamento de carga de servidor global do Amazon Route 53](#)

10.3 O sanduíche NLB

Problema

Você precisa dimensionar automaticamente sua camada de código aberto NGINX e distribuir a carga de maneira uniforme e fácil entre os servidores de aplicativos.

Solução

Crie um平衡ador de carga de rede (NLB). Durante a criação do NLB através do console, você é solicitado a criar um novo grupo-alvo. Se você não fizer isso por meio do console, precisará criar esse recurso e anexá-lo a um ouvinte no NLB.

Você cria um grupo de Auto Scaling com uma configuração de execução que provisão uma instância do EC2 com o NGINX Open Source instalado. O grupo de Auto Scaling possui uma configuração para vincular ao grupo de destino, que registra automaticamente qualquer instância do grupo de Auto Scaling ao grupo de destino configurado na primeira inicialização. O grupo de destino é referenciado por um ouvinte no NLB. Coloque seus aplicativos upstream atrás de outro balanceador de carga de rede e grupo de destino e, em seguida, configure o NGINX como proxy para o NLB do aplicativo.

Discussão

Esse padrão comum é chamado de sanduíche NLB (consulte a [Figura 10-1](#)), colocando o NGINX Open Source em um grupo de Auto Scaling atrás de um NLB e o grupo de Auto Scaling do aplicativo atrás de outro NLB. A razão para ter NLBs entre cada camada é porque o NLB funciona muito bem com grupos de Auto Scaling; eles registram automaticamente novos nós e removem aqueles que estão sendo encerrados, além de executar verificações de integridade e passar o tráfego apenas para nós íntegros. Pode ser necessário construir um segundo NLB interno

para a camada NGINX Open Source porque permite que os serviços em seu aplicativo chamem outros serviços por meio do grupo NGINX Auto Scaling sem sair da rede e entrar novamente pelo NLB público. Isso coloca o NGINX no meio de todo o tráfego de rede em seu aplicativo, tornando-o o centro do roteamento de tráfego de seu aplicativo. Esse padrão costumava ser chamado de sanduíche do balanceador de carga elástico; no entanto, o NLB é preferido ao trabalhar com NGINX porque o NLB é um balanceador de carga de camada 4, enquanto ELBs e ALBs são平衡adores de carga de camada 7. Os平衡adores de carga da camada 7 transformam a solicitação por meio do protocolo PROXY e são redundantes com o uso do NGINX. Esse padrão é necessário apenas para NGINX Open Source porque o conjunto de recursos fornecido pelo NLB está disponível no NGINX Plus.

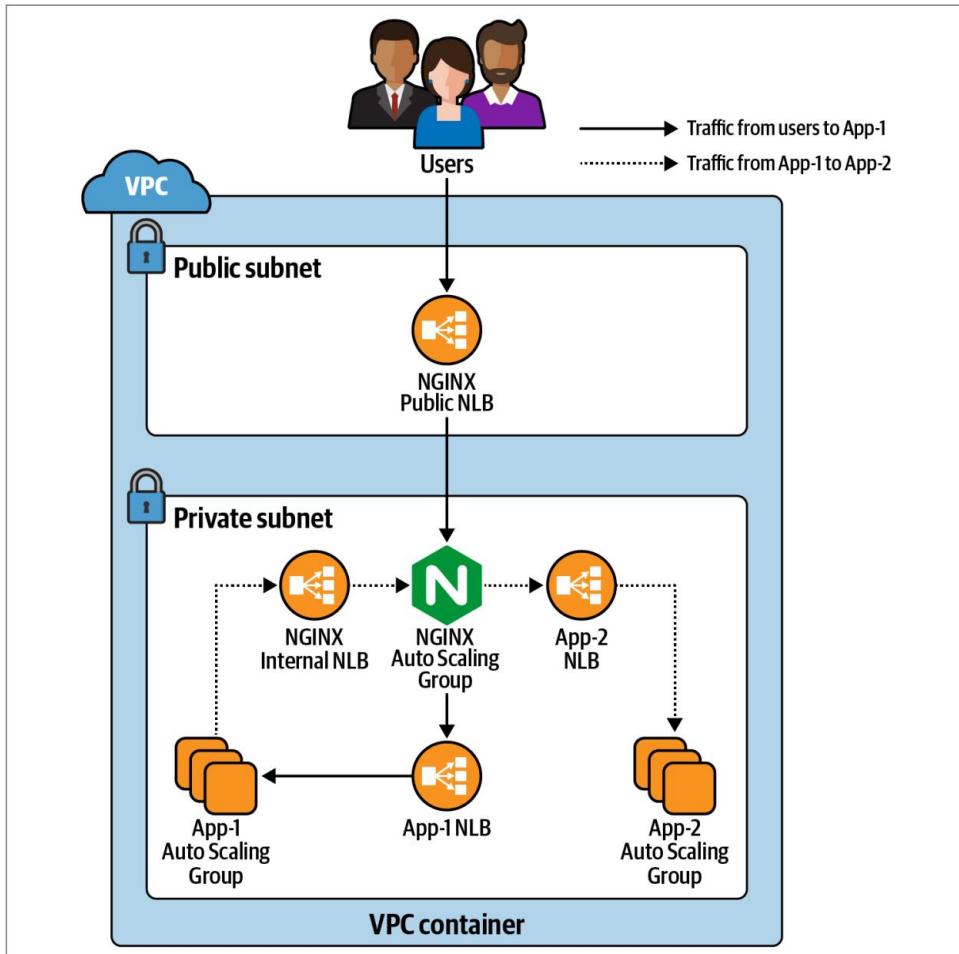


Figura 10-1. Esta imagem mostra o NGINX em um padrão de sanduíche NLB com um NLB interno para uso de aplicativos internos. Um usuário faz uma solicitação ao App-1 e o App-1 faz uma solicitação ao App-2 por meio do NGINX para atender à solicitação do usuário.

10.4 Implantação do AWS Marketplace

Problema

Você precisa executar o NGINX Plus na AWS com facilidade, com uma licença de pagamento conforme o uso.

Solução

Implante por meio do AWS Marketplace. Visite o [AWS Marketplace](#) e pesquise “NGINX Plus” (veja a [Figura 10-2](#)). Selecione a AMI baseada na distribuição Linux de sua escolha; revisar os detalhes, termos e preços; em seguida, clique no link Continuar. Na próxima página, você poderá aceitar os termos e implantar o NGINX Plus com um único clique ou aceitar os termos e usar a AMI.

The screenshot shows the AWS Marketplace interface with the search term "NGINX Plus" entered. The results page lists four available AMIs:

- NGINX Plus Basic - Amazon Linux AMI** (Version 1.4) | Sold by Nginx Software, Inc. | **Free Trial** | ★★★★☆ 10 AWS reviews | Starting from \$0.34/hr or from \$2,500.00/yr (16% savings) for software + AWS usage fees | Description: NGINX helps the world's most innovative companies deliver their sites and applications with performance, reliability, security, and scale. NGINX offers an award-winning, comprehensive application delivery platform in use on more than 315 million sites worldwide. Companies around the world rely on... | Linux/Unix, Amazon Linux 2017.09 - 64-bit Amazon Machine Image (AMI)
- NGINX Plus Basic - Ubuntu AMI** (Version 1.6) | Sold by Nginx Software, Inc. | **Free Trial** | ★★★★☆ 6 AWS reviews | Starting from \$0.34/hr or from \$2,500.00/yr (16% savings) for software + AWS usage fees | Description: NGINX helps the world's most innovative companies deliver their sites and applications with performance, reliability, security, and scale. NGINX offers an award-winning, comprehensive application delivery platform in use on more than 315 million sites worldwide. Companies around the world rely on... | Linux/Unix, Ubuntu Ubuntu 18.04 LTS - 64-bit Amazon Machine Image (AMI)
- NGINX Plus Basic - Red Hat Enterprise Linux 7 AMI** (Version 1.3) | Sold by Nginx Software, Inc. | **Free Trial** | ★★★★☆ & up (2) | Starting from \$0.34/hr or from \$2,500.00/yr (16% savings) for software + AWS usage fees | Description: Deliver applications with performance, reliability, security, and scale with NGINX Plus on AWS. Deploy quickly and save more than 80% compared to hardware ADCs with the all-in-one (yet surprisingly lightweight) load balancer, web server, content cache, and API gateway. Try NGINX Plus for free... | Linux/Unix, Red Hat Enterprise Linux 7.7 - 64-bit Amazon Machine Image (AMI)
- NGINX Plus - Amazon Linux 2 (LTS) AMI** (Version 1.2) | Sold by Nginx Software, Inc. | **Free Trial** | ★★★★☆ & up (2) | Starting from \$0.34/hr or from \$2,500.00/yr (16% savings) for software + AWS usage fees | Description: NGINX helps the world's most innovative companies deliver their sites and applications with performance, reliability, security, and scale. NGINX offers an award-winning, comprehensive

Figura 10-2. Pesquisando por NGINX Plus no AWS Marketplace

Discussão

A solução do AWS Marketplace para implantar o NGINX Plus oferece facilidade de uso e uma licença de pagamento conforme o uso. Você não apenas não tem nada para instalar, mas também tem uma licença sem pular obstáculos, como obter uma ordem de compra para uma licença de um ano. Esta solução permite que você experimente o NGINX Plus facilmente sem compromisso. Você também pode usar a AMI do NGINX Plus Marketplace como capacidade de estouro. É uma prática comum comprar sua carga de trabalho esperada de licenças e usar o Marketplace AMI em um grupo de Auto Scaling como capacidade de estouro. Essa estratégia garante que você pague apenas pelo licenciamento que usar.

10.5 Criando uma imagem de máquina virtual NGINX no Azure

Problema

Você precisa criar uma imagem de máquina virtual (VM) de seu próprio servidor NGINX configurado como achar melhor para criar rapidamente mais servidores ou usar em conjuntos de dimensionamento.

Solução

Crie uma VM a partir de um sistema operacional baseado em Linux de sua escolha. Depois que a VM for inicializada, faça login e instale o NGINX ou NGINX Plus da maneira que preferir, seja da fonte ou por meio da ferramenta de gerenciamento de pacotes da distribuição que você está executando. Configure o NGINX conforme desejado e crie uma nova imagem de VM. Para criar uma imagem de VM, você deve primeiro generalizar a VM. Para generalizar sua VM, você precisa remover o usuário provisionado pelo Azure, conectar-se a ele por SSH e executar o seguinte comando:

```
$ sudo waagent -deprovision+user -force
```

Este comando desprovisiona o usuário que o Azure provisionou ao criar a VM. A opção `-force` simplesmente pula uma etapa de confirmação. Depois de instalar o NGINX ou NGINX Plus e remover o usuário provisionado, você pode sair da sessão.

Conecte sua CLI do Azure à sua conta do Azure usando o comando de logon do Azure e verifique se está usando o modo Azure Resource Manager. Agora desaloque seu VM:

```
$ azure vm deallocate -g <ResourceGroupName> \
-n <VirtualMachineName>
```

Depois que a VM for desalocada, você poderá generalizá-la com o comando `azure vm generalize` :

```
$ azure vm generalize -g <ResourceGroupName> \
-n <VirtualMachineName>
```

Depois que sua VM for generalizada, você poderá criar uma imagem. O comando a seguir criará uma imagem e também gerará um modelo do Azure Resources Manager (ARM) para você usar para inicializar esta imagem:

```
$ azure vm capture <ResourceGroupName> <VirtualMachineName> \
    <ImageNamePrefix> -t <TemplateName>.json
```

A linha de comando produzirá uma saída informando que sua imagem foi criada, que está salvando um modelo ARM no local que você especificou e que a solicitação foi concluída. Você pode usar este modelo ARM para criar outra VM a partir da imagem recém-criada. No entanto, para usar este modelo que o Azure criou, você deve primeiro criar uma nova interface de rede:

```
$ azure network nic create <ResourceGroupName> \
    <NetworkInterfaceName> \
    <Region> \ --
    subnet-name <SubnetName> \ --
    subnet-vnet-name <VirtualNetworkName>
```

Esta saída de comando detalhará as informações sobre a interface de rede recém-criada. A primeira linha dos dados de saída será a ID da interface de rede, que você precisará para utilizar o modelo ARM criado pelo Azure. Depois de ter o ID, você pode criar uma implantação com o modelo ARM:

```
$ implantação do grupo do Azure criar <ResourceGroupName> \
    <DeploymentName> \ -f <TemplateName>.json
```

Você será solicitado a inserir várias variáveis de entrada, como vmName, adminUserName, adminPassword e networkInterfaceId . Insira um nome para a VM e o nome de usuário e a senha do administrador. Use o ID da interface de rede coletado do último comando como entrada para o prompt networkInterfaceId . Essas variáveis serão passadas como parâmetros para o modelo ARM e usadas para criar uma nova VM a partir da imagem NGINX ou NGINX Plus personalizada que você criou. Depois de inserir os parâmetros necessários, o Azure começará a criar uma nova VM a partir de sua imagem personalizada.

Discussão

A criação de uma imagem personalizada no Azure permite que você crie cópias de seu servidor NGINX ou NGINX Plus pré-configurado à vontade. Um modelo do Azure ARM permite que você implante esse mesmo servidor de forma rápida e confiável, sempre que necessário. Com o caminho da imagem de VM que pode ser encontrado no modelo, você pode criar diferentes conjuntos de infraestrutura, como conjuntos de dimensionamento de VM ou outras VMs com configurações diferentes.

Veja também

[Instalando a CLI de plataforma cruzada do Azure](#)

[Logon da CLI de plataforma cruzada do Azure](#)

[Capturando imagens de máquinas virtuais Linux](#)

10.6 Balanceamento de carga sobre conjuntos de dimensionamento NGINX no Azure

Problema

Você precisa dimensionar os nós NGINX por trás de um平衡ador de carga do Azure para obter alta disponibilidade e uso dinâmico de recursos.

Solução

Crie um balanceador de carga do Azure voltado para o público ou interno. Implante a imagem NGINX VM criada na seção anterior ou a imagem NGINX Plus do Marketplace descrito na [Receita 10.7](#), em um conjunto de dimensionamento de máquina virtual do Azure (VMSS). Depois que o balanceador de carga e o VMSS forem implantados, configure um pool de back-end no balanceador de carga para o VMSS. Configure regras de balanceamento de carga para as portas e protocolos nos quais você gostaria de aceitar tráfego e encaminhe-os para o pool de back-end.

Discussão

É comum dimensionar o NGINX para obter alta disponibilidade ou lidar com cargas de pico sem superprovisionar recursos. No Azure, você consegue isso com o VMSS. O uso do balanceador de carga do Azure oferece facilidade de gerenciamento para adicionar e remover nós NGINX ao pool de recursos ao dimensionar. Com os平衡adores de carga do Azure, você pode verificar a integridade de seus pools de back-end e passar tráfego apenas para nós íntegros. Você pode executar平衡adores de carga internos do Azure na frente do NGINX, onde deseja habilitar o acesso apenas por meio de uma rede interna. Você pode usar o NGINX para fazer proxy de um balanceador de carga interno na frente de um aplicativo dentro de um VMSS, usando o balanceador de carga para facilitar o registro e o cancelamento do registro do pool.

10.7 Implantando por meio do Azure Marketplace

Problema

Você precisa executar o NGINX Plus no Azure com facilidade e uma licença de pagamento conforme o uso.

Solução

Implante uma imagem de VM NGINX Plus por meio do Azure Marketplace:

1. No painel do Azure, selecione o ícone Novo e use a barra de pesquisa para pesquisar "NGINX". Os resultados da pesquisa serão exibidos.
2. Na lista, selecione a imagem da máquina virtual NGINX Plus publicada por NGINX, Inc.
3. Quando solicitado a escolher seu modelo de implantação, selecione a opção Resource Manager e clique no botão Criar.
4. Em seguida, você será solicitado a preencher um formulário para especificar o nome de sua VM, o tipo de disco, o nome de usuário e a senha padrão ou a chave pública do par de chaves SSH, a assinatura a ser cobrada, o grupo de recursos desejado usar e a localização.
5. Uma vez que este formulário esteja preenchido, você pode clicar em OK. Seu formulário será validado.
6. Quando solicitado, selecione um tamanho de VM e clique no botão Selecionar.
7. No próximo painel, você tem a opção de selecionar configurações opcionais, que serão o padrão com base na escolha do grupo de recursos feita anteriormente. Após alterar essas opções e aceitá-las, clique em OK.
8. Na próxima tela, revise o resumo. Você tem a opção de baixar essa configuração como um modelo ARM para poder criar esses recursos novamente mais rapidamente por meio de um modelo JSON.
9. Depois de revisar e baixar seu modelo, você pode clicar em OK para ir para a tela de compra. Esta tela irá notificá-lo sobre os custos que você está prestes a incorrer com o uso dessa VM. Clique em Comprar e sua caixa NGINX Plus começará a inicializar.

Discussão

O Azure e o NGINX facilitaram a criação de uma VM NGINX Plus no Azure por meio de apenas alguns formulários de configuração. O Azure Marketplace é uma ótima maneira de obter o NGINX Plus sob demanda com uma licença de pagamento conforme o uso. Com este modelo, você pode experimentar os recursos do NGINX Plus ou usá-lo para capacidade de estouro sob demanda de seus servidores NGINX Plus já licenciados.

10.8 Como implantar no Google Compute Engine

Problema

Você precisa criar um servidor NGINX no Google Compute Engine para balancear a carga ou proxy para o restante de seus recursos no Google Compute ou App Engine.

Solução

Inicie uma nova VM no Google Compute Engine. Selecione um nome para sua VM, zona, tipo de máquina e disco de inicialização. Configure o gerenciamento de identidade e acesso, firewall e qualquer configuração avançada que desejar. Crie a VM.

Depois que a VM for criada, faça login via SSH ou pelo Google Cloud Shell.

Instale o NGINX ou NGINX Plus por meio do gerenciador de pacotes para o tipo de SO fornecido. Configure o NGINX como achar melhor e recarregue.

Como alternativa, você pode instalar e configurar o NGINX por meio do Google Compute Script de inicialização do mecanismo, que é uma opção de configuração avançada ao criar um VM.

Discussão

O Google Compute Engine oferece VMs altamente configuráveis a qualquer momento. Iniciar uma VM requer pouco esforço e possibilita um mundo de possibilidades. O Google Compute Engine oferece rede e computação em um ambiente de nuvem virtualizado. Com uma instância do Google Compute, você tem todos os recursos de um servidor NGINX onde e quando precisar.

10.9 Criando uma imagem do Google Compute

Problema

Você precisa criar uma imagem do Google Compute para instanciar rapidamente uma VM ou criar um modelo de instância para um grupo de instâncias.

Solução

Crie uma VM conforme descrito na [Receita 10.8](#). Depois de instalar e configurar o NGINX em sua instância de VM, defina o estado de exclusão automática do disco de inicialização como false. Para definir o estado de exclusão automática do disco, edite a VM. Na página Editar, na configuração do disco, há uma caixa de seleção chamada "Excluir disco de inicialização quando a instância for excluída". Desmarque esta caixa de seleção e salve a configuração da VM. Depois que o estado de exclusão automática da instância for definido como falso, exclua a instância. Quando solicitado, não marque a caixa de seleção que oferece a exclusão do disco de inicialização. Ao executar essas tarefas, você ficará com um disco de inicialização não anexado com o NGINX instalado.

Depois que sua instância for excluída e você tiver um disco de inicialização não anexado, poderá criar uma imagem do Google Compute. Na seção Imagem do console do Google Compute Engine, selecione Criar imagem. Ser-lhe-á pedido um nome de imagem, família, descrição, tipo de encriptação e a fonte. O tipo de fonte que você precisa usar é disco; e para

o disco de origem, selecione o disco de inicialização NGINX desanexado. Selecione Criar e o Google Compute Cloud criará uma imagem do seu disco.

Discussão

Você pode utilizar o Google Cloud Images para criar VMs com um disco de inicialização idêntico ao servidor que acabou de criar. O valor da criação de imagens é poder garantir que todas as instâncias dessa imagem sejam idênticas. Ao instalar pacotes no momento da inicialização em um ambiente dinâmico, a menos que use bloqueio de versão com repositórios privados, você corre o risco de a versão do pacote e as atualizações não serem validadas antes de serem executadas em um ambiente de produção. Com imagens de máquina, você pode validar se cada pacote executado nesta máquina é exatamente como você testou, fortalecendo a confiabilidade de sua oferta de serviço.

Veja também

[Criar, excluir e depreciação de imagens privadas](#)

10.10 Criando um proxy do Google App Engine

Problema

Você precisa criar um proxy para o Google App Engine para alternar o contexto entre aplicativos ou servir HTTPS em um domínio personalizado.

Solução

Utilize o NGINX no Google Compute Cloud. Crie uma VM no Google Compute Engine ou crie uma imagem de máquina virtual com o NGINX instalado e crie um modelo de instância com essa imagem como seu disco de inicialização. Se você criou um modelo de instância, continue criando um grupo de instâncias que utiliza esse modelo.

Configure o NGINX como proxy para seu endpoint do Google App Engine. Certifique-se de fazer proxy para HTTPS porque o Google App Engine é público, e você deve garantir que não encerre o HTTPS em sua instância do NGINX e permita que as informações trafeguem entre o NGINX e o Google App Engine sem segurança. Como o App Engine fornece apenas um único endpoint DNS, você usará a diretiva proxy_pass em vez de blocos upstream na versão de código aberto do NGINX porque a versão de código aberto não é capaz de resolver nomes DNS de servidores upstream. Ao fazer proxy para o Google App Engine, certifique-se de definir o endpoint como uma variável no NGINX e, em seguida, use essa variável na diretiva proxy_pass para garantir que o NGINX faça a resolução de DNS em todas as solicitações. Para que o NGINX faça qualquer resolução de DNS, você também precisará utilizar a diretiva do resolvedor e apontar para seu resolvedor de DNS favorito. O Google disponibiliza o endereço IP 8.8.8.8 para uso público. Se você estiver usando o NGINX Plus, poderá

use o sinalizador de resolução na diretiva do servidor no bloco upstream, conexões keepalive e outros benefícios do módulo upstream ao fazer proxy para o Google App Engine.

Você pode optar por armazenar seus arquivos de configuração do NGINX no Google Storage e, em seguida, usar o script de inicialização da sua instância para baixar a configuração no momento da inicialização. Isso permitirá que você altere sua configuração sem ter que gravar uma nova imagem. No entanto, isso aumentará o tempo de inicialização do seu servidor NGINX.

Discussão

Você deseja executar o NGINX na frente do Google App Engine se estiver usando seu próprio domínio e quiser disponibilizar seu aplicativo via HTTPS. No momento, o Google App Engine não permite que você faça upload de seus próprios certificados SSL. Portanto, se você quiser veicular seu aplicativo em um domínio diferente de appspot.com com criptografia, será necessário criar um proxy com NGINX para escutar em seu domínio personalizado. O NGINX criptografará a comunicação entre ele e seus clientes, bem como entre ele e o Google App Engine.

Outra razão pela qual você pode querer executar o NGINX na frente do Google App Engine é hospedar muitos aplicativos do App Engine no mesmo domínio e usar o NGINX para fazer a troca de contexto baseada em URI. Os microsserviços são uma arquitetura popular e é comum que um proxy como o NGINX conduza o roteamento de tráfego. O Google App Engine facilita a implantação de aplicativos e, em conjunto com o NGINX, você tem uma plataforma de entrega de aplicativos completa.

CAPÍTULO 11**Contêineres/Microsserviços**

11.0 Introdução

Os contêineres oferecem uma camada de abstração na camada de aplicação, deslocando a instalação de pacotes e dependências do processo de implantação para o de construção. Isso é importante porque os engenheiros agora estão enviando unidades de código que são executadas e implantadas de maneira uniforme, independentemente do ambiente. A promoção de contêineres como unidades executáveis reduz o risco de dependência e confusão de configuração entre ambientes. Diante disso, tem havido um grande impulso para as organizações implantarem seus aplicativos em plataformas de contêineres. Ao executar aplicativos em uma plataforma de contêiner, é comum colocar em contêiner o máximo possível da pilha, incluindo seu proxy ou balanceador de carga. NGINX e NGINX Plus contentorizam e enviam com facilidade. Eles também incluem muitos recursos que tornam a entrega de aplicativos em contêineres fluida. Este capítulo se concentra na criação de imagens de contêiner NGINX e NGINX Plus, recursos que facilitam o trabalho em um ambiente em contêiner e na implantação de sua imagem no Kubernetes e no OpenShift.

Ao fazer a conteinerização, muitas vezes é comum decompor os serviços em aplicativos menores. Ao fazer isso, eles são vinculados por um gateway de API. A primeira seção deste capítulo fornece um cenário de caso comum de uso do NGINX como um gateway de API para proteger, validar, autenticar e rotear solicitações para o serviço apropriado.

Algumas considerações de arquitetura sobre a execução do NGINX ou NGINX Plus em um contêiner devem ser destacadas. Ao conteinerizar um serviço, para usar o driver de log do Docker, os logs devem ser enviados para /dev/stdout e os logs de erros direcionados para /dev/stderr. Ao fazer isso, os logs são transmitidos para o driver de log do Docker, que pode roteá-los para servidores de log consolidados nativamente.

Os métodos de平衡amento de carga também são considerados ao usar o NGINX Plus em um ambiente em contêiner. O método de平衡amento de carga less_time foi projetado com

sobreposições de rede em contêiner em mente. Ao favorecer o baixo tempo de resposta, o NGINX Plus passará a solicitação recebida para o servidor upstream com o tempo médio de resposta mais rápido. Quando todos os servidores possuem平衡amento de carga adequado e desempenho igual, o NGINX Plus pode otimizar por latência de rede, preferindo servidores mais próximos da rede.

11.1 Usando NGINX como um API Gateway

Problema

Você precisa de um gateway de API para validar, autenticar, manipular e rotear solicitações de entrada para seu caso de uso.

Solução

Use NGINX ou NGINX Plus como um gateway de API. Um gateway de API fornece um ponto de entrada para uma ou mais interfaces de programação de aplicativos (APIs). O NGINX se encaixa muito bem nesse papel. Esta seção destacará alguns conceitos centrais e fará referência a outras seções deste livro para obter mais detalhes sobre as especificidades. Também é importante observar que o NGINX publicou um e-book inteiro sobre este tópico: [Implantando o NGINX Plus como um gateway de API](#).

Comece definindo um bloco de servidor para seu gateway de API em seu próprio arquivo. Um nome como /etc/nginx/api_gateway.conf serve.

```
servidor
{
    escuta 443 ssl;
    nome_do_servidor api.empresas.com;
    # Configurações SSL Capítulo 7

    default_type application/json;
}
```

Adicione algumas respostas básicas de tratamento de erros à sua definição de servidor:

```
proxy_intercept_errors ativado;

página_de_erro 400 =
@400; local @400 { return 400 '{"status":400,"message":"Solicitação incorreta"}\n'; }

página_de_erro 401 =
@401; local @401 { return 401 '{"status":401,"message":"Unauthorized"}\n'; }

página_de_erro 403 =
@403; local @403 { return 403 '{"status":403,"message":"Proibido"}\n'; }

página_de_erro 404 =
@404; location @404 { return 404 '{"status":404,"message":"Recurso não encontrado"}\n'; }
```

A seção acima da configuração do NGINX pode ser adicionada diretamente ao bloco do servidor em /etc/nginx/api_gateway.conf ou em um arquivo separado e importado por meio de uma diretiva de inclusão . A diretiva include é abordada na [Receita 17.1](#).

Use uma diretiva include para importar esta configuração de servidor para o arquivo nginx.conf principal dentro do contexto http :

```
incluir /etc/nginx/api_gateway.conf;
```

Agora você precisa definir seus endpoints de serviço upstream. [O Capítulo 2](#) cobre o balanceamento de carga, que discute o bloco upstream . Como lembrete, o upstream é válido no contexto http e não no contexto do servidor . O seguinte deve ser incluído ou definido fora do bloco do servidor .

```
serviço upstream_1 {
    servidor 10.0.0.12:80;
    servidor 10.0.0.13:80;

} serviço upstream_2 {
    servidor 10.0.0.14:80;
    servidor 10.0.0.15:80;
}
```

Dependendo do caso de uso, você pode declarar seus serviços inline, como um arquivo incluído ou incluído por serviços. Também existe um caso em que os serviços devem ser definidos como pontos de extremidade de localização de proxy, neste caso, é sugerido definir o ponto de extremidade como uma variável para uso completo. [O Capítulo 5, Programabilidade e Automação](#), discute maneiras de automatizar a adição e remoção de máquinas de blocos upstream .

Crie um local roteável internamente no bloco do servidor para cada serviço:

```
local = /_service_1 { interno;

    # Configuração comum ao serviço
    proxy_pass http://service_1$request_uri;

} local = /_service_2 { interno;

    # Configuração comum ao serviço
    proxy_pass http://service_2$request_uri;
}
```

Ao definir locais roteáveis internos para esses serviços, a configuração comum ao serviço pode ser definida uma vez, em vez de repetidamente.

A partir daqui, precisamos construir blocos de localização que definam caminhos de URI específicos para um determinado serviço. Esses blocos validarão e encaminharão a solicitação adequadamente. Um gateway de API pode ser tão simples quanto rotear solicitações com base no caminho e tão detalhado quanto definir regras específicas para cada URI de API aceita. Neste último, você vai querer conceber

uma estrutura de arquivos para organização e uso O NGINX inclui para importar seus arquivos de configuração. Este conceito é discutido na [Receita 17.1](#).

Crie um novo diretório para o gateway de API:

```
mkdir /etc/nginx/api_conf.d/
```

Crie uma especificação de um caso de uso de serviço definindo blocos de localização em um arquivo em um caminho que faça sentido para sua estrutura de configuração. Use a diretiva de reescrita para direcionar a solicitação para o bloco de local configurado anteriormente que faz proxy da solicitação para um serviço. A diretiva de reescrita usada abaixo instrui o NGINX a reprocessar a solicitação com um URI alterado. O seguinte define regras específicas para recursos de API, restringe métodos HTTP e usa a diretiva de reescrita para enviar a solicitação para o local de proxy comum interno definido anteriormente para o serviço.

```
local /api/service_1/object {
    limit_except GET PUT { nega tudo; }
    reescrever ^/_service_1 último;

} local /api/service_1/object/[^\$]*\$ { limit_except
    GET POST { deny all; } reescrever ^/
    _service_1 último;
}
```

Repita esta etapa para cada serviço. Empregue separação lógica por meio de estruturas de arquivos e diretórios para organizar efetivamente o caso de uso. Use toda e qualquer informação fornecida neste livro para configurar os blocos de localização da API para serem tão específicos e restritivos quanto possível.

Se arquivos separados foram usados para o local acima ou blocos upstream , verifique se eles estão incluídos no contexto do seu servidor:

```
servidor
{
    escuta 443 ssl;
    nome_do_servidor api.empresia.com;
    # Configurações SSL Capítulo 7

    default_type application/json;

    inclua api_conf.d/*.conf;
}
```

Habilite a autenticação para proteger recursos privados, usando um dos muitos métodos discutidos no [Capítulo 6](#), ou algo tão simples como chaves de API pré-compartilhadas como segue (observe que a diretiva map só é válida no contexto http):

```
map $http_apikey $api_client_name {
    default
    "";
    "j7UqLLB+yRv2VTCXXDZ1M/N4" "client_one";
    "6B2kbyrrTilN8S8JhSAxb63R" "client_two";
```

```
"KcVgIDSY4Nm46m3tXVY3vbgA" "client_three";
}
```

Proteja os serviços de back-end contra ataques com o NGINX empregando os aprendizados do [Capítulo 2](#) para limitar o uso. No contexto http , defina uma ou várias zonas de memória compartilhada de limite de solicitação.

```
limit_req_zone $http_apikey
    zone=limitbyapikey:10m rate=100r/s;
limit_req_status 429;
```

Proteja um determinado contexto com limites de taxa e autenticação:

```
local /api/service_2/object { limit_req
    zona=limitbyapikey;

    # Considere escrever esses if's em um arquivo #
    # e usar uma inclusão fosse necessária. if
    ($http_apikey = "") { return 401;

    }

    } if ($api_client_name = "") { return
        403;
    }

    limit_except GET PUT { nega tudo; }
    reescrever ^/_service_2 último;
}
```

Teste algumas chamadas para seu gateway de API:

```
curl -H "apikey: 6B2kbyrrTilN8S8JhSAxb63R" \
https://api.company.com/api/service_2/object
```

Discussão

Os gateways de API fornecem um ponto de entrada para uma interface de programação de aplicativos (API). Isso soa vago e básico, então vamos nos aprofundar. Os pontos de integração acontecem em muitas camadas diferentes. Quaisquer dois serviços independentes que precisem se comunicar (integrar) devem ter um contrato de versão de API. Tais contratos de versão definem a compatibilidade dos serviços. Um gateway de API reforça esses contratos – autenticando, autorizando, transformando e roteando solicitações entre serviços.

Esta seção demonstrou como o NGINX pode funcionar como um gateway de API validando, autenticando e direcionando solicitações recebidas para serviços específicos e limitando seu uso. Essa tática é popular em arquiteturas de microserviços, onde uma única oferta de API é dividida entre diferentes serviços.

Implore todos os seus aprendizados até agora para construir uma configuração de servidor NGINX com as especificações exatas para seu caso de uso. Ao unir os conceitos centrais demonstrados neste texto, você tem a capacidade de autenticar e autorizar o uso de

Caminhos de URI, rotear ou reescrever solicitações com base em qualquer fator, limitar o uso e definir o que é e o que não é aceito como uma solicitação válida. Nunca haverá uma única solução para um gateway de API, pois cada um é íntima e infinitamente definível para o caso de uso que ele fornece.

Um gateway de API fornece um espaço de colaboração definitivo entre as equipes de operações e de aplicativos para formar uma verdadeira organização DevOps. O desenvolvimento de aplicativos define os parâmetros de validade de uma determinada solicitação. A entrega de tal solicitação é normalmente gerenciada pelo que é considerado TI (equipes de rede, infraestrutura, segurança e middleware). Um gateway de API atua como uma interface entre essas duas camadas. A construção de um gateway de API requer entrada de todos os lados. A configuração de tal deve ser mantida em algum tipo de controle de origem. Muitos repositórios de controle de origem modernos têm o conceito de proprietários de código. Este conceito permite que você exija a aprovação de usuários específicos para determinados arquivos. Dessa forma, as equipes podem colaborar, mas verificar as alterações específicas de um determinado departamento.

Algo a ter em mente ao trabalhar com gateways de API é o caminho do URI. Na configuração de exemplo, todo o caminho do URI é passado para os servidores upstream. Isso significa que o exemplo service_1 precisa ter manipuladores no caminho /api/service_1/* .

Para executar o roteamento baseado em caminho dessa maneira, é melhor que o aplicativo não tenha rotas conflitantes com outro aplicativo.

Se houver rotas conflitantes, há algumas coisas que você pode fazer. Edite o código para resolver os conflitos ou adicione uma configuração de prefixo de URI a um ou ambos os aplicativos para mover um deles para outro contexto. No caso de software de prateleira que não pode ser editado, você pode reescrever o URI de solicitações upstream. No entanto, se o aplicativo retornar links no corpo, você precisará usar expressões regulares (regex) para reescrever o corpo da solicitação antes de fornecê-la ao cliente - isso deve ser evitado.

Veja também

[Implantando o NGINX Plus como um EBook de gateway de API](#)

11.2 Usando registros SRV DNS com NGINX Plus

Problema

Você gostaria de usar sua implementação de registro SRV DNS existente como fonte para servidores upstream com NGINX Plus.

Solução

Especifique a diretiva de serviço com um valor de http em um servidor upstream para instruir NGINX para utilizar o registro SRV como um pool de平衡amento de carga:

```

http
    { resolvedor 10.0.0.2 valid=30s;

        backend upstream
            { backends de zona
                64k; servidor api.example.internal service=http resolve;
            }
        }
    }
}

```

Esse recurso é exclusivo do NGINX Plus. A configuração instrui o NGINX Plus a resolver o DNS de um servidor DNS em 10.0.0.2 e configurar um pool de servidores upstream com uma única diretiva de servidor . Essa diretiva do servidor especificada com o parâmetro resolve é instruída a re-resolver periodicamente a base do nome de domínio no TTL do registro DNS ou o parâmetro de substituição válido da diretiva do resolvedor . O parâmetro e valor service=http informa ao NGINX que este é um registro SRV contendo uma lista de IPs e portas e para balancear a carga sobre eles como se estivessem configurados com a diretiva do servidor .

Discussão

A infraestrutura dinâmica está se tornando cada vez mais popular com a demanda e adoção de infraestrutura baseada em nuvem. Os ambientes de Auto Scaling são dimensionados horizontalmente, aumentando e diminuindo o número de servidores no pool para atender à demanda da carga. O dimensionamento horizontal exige um平衡ador de carga que pode adicionar e remover recursos do pool. Com um registro SRV, você descarrega a responsabilidade de manter a lista de servidores no DNS. Esse tipo de configuração é extremamente atraente para ambientes em contêiner porque você pode ter contêineres executando aplicativos em números de porta variáveis, possivelmente no mesmo endereço IP. É importante observar que a carga útil do registro DNS UDP é limitada a cerca de 512 bytes.

11.3 Usando a imagem oficial do NGINX

Problema

Você precisa começar a trabalhar rapidamente com a imagem NGINX do Docker Hub.

Solução

Use a imagem NGINX do Docker Hub. Esta imagem contém uma configuração padrão. Você precisará montar um diretório de configuração local ou criar um Dockerfile e ADICIONAR em sua configuração à compilação da imagem para alterar a configuração. Aqui montamos um volume onde a configuração padrão do NGINX serve conteúdo estático para demonstrar seus recursos usando um único comando:

```
$ docker run --name meu-nginx -p 80:80 \
-v /path/to/content:/usr/share/nginx/html:ro -d nginx
```

O comando docker extrai a imagem nginx:latest do Docker Hub se ela não for encontrada localmente. O comando então executa essa imagem NGINX como um contêiner do Docker, mapeando localhost:80 para a porta 80 do contêiner NGINX. Ele também monta o diretório local / path/to/content/ como um volume de contêiner em /usr/share/nginx/html/ como somente leitura. A configuração padrão do NGINX servirá este diretório como conteúdo estático. Ao especificar o mapeamento de sua máquina local para um contêiner, a porta ou diretório da máquina local vem primeiro e a porta ou diretório do contêiner vem em segundo lugar.

Discussão

O NGINX disponibilizou uma imagem oficial do Docker via Docker Hub. Essa imagem oficial do Docker facilita a instalação e o funcionamento muito rápido no Docker com sua plataforma de entrega de aplicativos favorita, NGINX. Nesta seção, conseguimos colocar o NGINX em funcionamento em um contêiner com um único comando! A linha principal oficial da imagem NGINX Docker que usamos neste exemplo é construída a partir da imagem Debian Jessie Docker. No entanto, você pode escolher imagens oficiais baseadas no Alpine Linux. O Dockerfile e a fonte dessas imagens oficiais estão disponíveis no GitHub. Você pode estender a imagem oficial criando seu próprio Dockerfile e especificando a imagem oficial no comando FROM . Você também pode montar um diretório de configuração do NGINX como um volume do Docker para substituir a configuração do NGINX sem modificar a imagem oficial.

Veja também

[Imagen oficial do NGINX Docker, NGINX](#)

[Repositório do Docker no GitHub](#)

11.4 Criando um Dockerfile NGINX

Problema

Você precisa criar um NGINX Dockerfile para criar uma imagem do Docker.

Solução

Comece pela imagem do Docker da sua distribuição favorita. Use o comando RUN para instalar o NGINX. Use o comando ADD para adicionar seus arquivos de configuração NGINX. Use o comando EXPOSE para instruir o Docker a expor determinadas portas ou faça isso manualmente ao executar a imagem como um contêiner. Use o CMD para iniciar o NGINX quando a imagem for instanciada como um contêiner. Você precisará executar o NGINX em primeiro plano. Para fazer isso, você precisará iniciar o NGINX com -g "daemon off;" ou adicione o daemon off; à sua configuração. Este exemplo usará o último com o daemon desativado; no arquivo de configuração dentro do contexto principal. Você também desejará alterar sua configuração do NGINX para registrar

para /dev/stdout para logs de acesso e /dev/stderr para logs de erros; fazer isso colocará seus logs nas mãos do daemon do Docker, o que os tornará mais facilmente disponíveis, com base no driver de log que você escolheu usar com o Docker:

DE centos:⁷

Instale o epel repo para obter o nginx e instale o nginx

RUN yum -y install epel-release && \ yum -y
install nginx

adiciona arquivos de configuração local na imagem

ADICIONE /nginx-conf /etc/nginx

EXPOR 80 443

CMD ["nginx"]

A estrutura de diretórios tem a seguinte aparência:

```
..  
  -- Dockerfile  
  -- nginx-conf  
    -- conf.d  
      -- default.conf  
      -- fastcgi.conf  
      -- fastcgi_params  
      -- koi-utf  
      -- koi-win  
      -- mime.types  
      -- nginx.conf  
      -- scgi_params  
      -- uwsgi_params  
    -- win-utf
```

Optei por hospedar toda a configuração do NGINX nesse diretório do Docker para facilitar o acesso a todas as configurações com apenas uma linha no Dockerfile para adicionar todas as minhas configurações do NGINX.

Discussão

Você achará útil criar seu próprio Dockerfile quando precisar de controle total sobre os pacotes instalados e as atualizações. É comum manter seu próprio repositório de imagens para que você saiba que sua imagem base é confiável e testada por sua equipe antes de executá-la em produção.

11.5 Criando uma imagem do Docker NGINX Plus

Problema

Você precisa criar uma imagem do Docker do NGINX Plus para executar o NGINX Plus em um ambiente em contêiner.

Solução

Use este Dockerfile para criar uma imagem do Docker NGINX Plus. Você precisará baixar seus certificados de repositório NGINX Plus e mantê-los no diretório com este Dockerfile chamado nginx-repo.crt e nginx-repo.key, respectivamente. Com isso, este Dockerfile fará o resto do trabalho instalando o NGINX Plus para seu uso e vinculando o acesso NGINX e os logs de erros ao coletor de logs do Docker.

```
DE debian:stretch-slim

mantenedor do LABEL ="NGINX <docker-maint@nginx.com>"

# Baixe o certificado e a chave do portal do cliente # (https://cs.nginx.com) e copie para o contexto de compilação

COPY nginx-repo.crt /etc/ssl/nginx/ COPY
nginx-repo.key /etc/ssl/nginx/

# Instalar NGINX Plus

RUN set -x \&&
      APT_PKG="Adquirir::https::plus-pkgs.nginx.com::" \&&
      REPO_URL="https://plus-pkgs.nginx.com/debian" \&& apt-get
      update \&& apt-get upgrade -y \&& apt-get install \
      --no-install-recommends --no-install-suggests\ -y apt-
      transport-https ca-certificates gnupg1 \&& \
      NGINX_GPGKEY=573BFD6B3D8FBC641079A6ABAF5BD827BD9BF62;\ \
      encontrado=""; \ para servidor em \
      ha.pool.sks-keyservers.net \ hkp://
      keyserver.ubuntu.com:80 \ hkp://
      p80.pool.sks-keyservers.net:80 \ pgp.mit.edu \ ;
      do \ echo "Buscando a chave GPG"
$NGINX_GPGKEY do $server"; \ apt-key adv --
      keyserver "$server" --keyserver-options \ timeout=10 --recv-keys
      "$NGINX_GPGKEY" \
      \&& found=sim \
      \&& break\; feito\;
      test -z "$found" \&&
      echo >&2 \
      "erro: falha ao buscar a chave GPG $NGINX_GPGKEY" \&& exit 1; \
```

```

echo "${APT_PKG}Verify-Peer "true";"\ >> /etc/
apt/apt.conf.d/90nginx \&& echo \ "${APT_PKG}
Verify-Host "true";">>\ /etc/apt/apt.conf.d/90nginx
\\ "/etc/ssl/nginx/nginx-repo.crt;" >> \\ /etc/apt/
apt.conf.d/90nginx \&& echo "${APT_PKG}
&& echo "${APT_PKG}SslCert SslKey
\\ "/etc/ssl/nginx/nginx-repo.key;" >> \\ /etc/apt/
apt.conf.d/90nginx \&& printf \\ "deb ${REPO_URL}
stretch nginx-plus" \ > /etc/apt/sources.list\\ /etc/apt/
&& apt-get update && apt-get install -y nginx-plus
\ && apt-get remove --purge --auto-remove -y
gnupg1 \&& rm -rf /var/lib/apt/lists/*

```

Encaminhe logs de solicitação para o coletor de logs do

Docker RUN In -sf /dev/stdout /var/log/nginx/access.log \&& In
-sf /dev/stderr /var/log/nginx/error.log

EXPOR 80

SINAL DE PARAGEM

CMD ["nginx", "-g", "daemon off;"]

Para criar esse Dockerfile em uma imagem do Docker, execute o seguinte no diretório que contém o Dockerfile e o certificado e a chave do repositório NGINX Plus:

\$ docker build --no-cache -t nginxplus .

Esse comando docker build usa o sinalizador --no-cache para garantir que, sempre que você compilar, os pacotes NGINX Plus sejam extraídos do repositório NGINX Plus para atualizações. Se for aceitável usar a mesma versão no NGINX Plus que a compilação anterior, você pode omitir o sinalizador --no-cache . Neste exemplo, a nova imagem do Docker é marcada como nginxplus.

Discussão

Ao criar sua própria imagem do Docker para o NGINX Plus, você pode configurar seu contêiner NGINX Plus da maneira que achar melhor e soltá-lo em qualquer ambiente do Docker. Isso abre todo o poder e recursos avançados do NGINX Plus para seu ambiente em contêiner. Este Dockerfile não usa a propriedade ADD do Dockerfile para adicionar em sua configuração; você precisará adicionar em sua configuração manualmente.

Veja também

[Blog NGINX em imagens do Docker](#)

11.6 Usando variáveis de ambiente no NGINX

Problema

Você precisa usar variáveis de ambiente dentro da configuração do NGINX para usar a mesma imagem de contêiner para ambientes diferentes.

Solução

Use o `ngx_http_perl_module` para definir variáveis no NGINX do seu ambiente:

```
daemon
desligado; env
APP_DNS; inclua /usr/share/nginx/modules/*.conf;
# ...
http
{ perl_set $upstream_app 'sub { return $ENV{"APP_DNS"}; }'; servidor
{ #
...
local /
{ proxy_pass https://$upstream_app;
}
}
}
```

Para usar `perl_set` você deve ter o `ngx_http_perl_module` instalado; você pode fazer isso carregando o módulo dinamicamente ou estaticamente se estiver compilando a partir da fonte. O NGINX, por padrão, limpa as variáveis de ambiente de seu ambiente; você precisa declarar quaisquer variáveis que não deseja remover com a diretiva `env`. A diretiva `perl_set` recebe dois parâmetros: o nome da variável que você gostaria de definir e uma string Perl que renderiza o resultado.

A seguir está um Dockerfile que carrega o `ngx_http_perl_module` dinamicamente, instalando este módulo do utilitário de gerenciamento de pacotes. Ao instalar módulos do utilitário de pacote para CentOS, eles são colocados no diretório `/usr/lib64/nginx/modules/`, e os arquivos de configuração que carregam dinamicamente esses módulos são colocados no diretório `/usr/share/nginx/modules/` diretório. É por isso que no trecho de configuração anterior incluímos todos os arquivos de configuração nesse caminho:

DE centos:7

```
# Instale o epel repo para obter o nginx e instale o nginx
RUN yum -y install epel-release && \
    yum -y instala nginx nginx-mod-http-perl

# adiciona arquivos de configuração local na imagem
ADICIONE /nginx-conf /etc/nginx
```

EXPOR 80 443

CMD ["nginx"]

Discussão

Uma prática típica ao usar o Docker é utilizar variáveis de ambiente para alterar a maneira como o contêiner opera. Você pode usar variáveis de ambiente em sua configuração NGINX para que seu NGINX Dockerfile possa ser usado em vários ambientes diversos.

omentos.

11.7 Controlador de entrada do Kubernetes

Problema

Você está implantando seu aplicativo no Kubernetes e precisa de um controlador de entrada.

Solução

Certifique-se de ter acesso à imagem do controlador de ingresso. Para NGINX, você pode usar a imagem nginx/nginx-ingress do Docker Hub. Para o NGINX Plus, você precisará criar sua própria imagem e hospedá-la em seu registro privado do Docker. Você pode encontrar instruções sobre como construir e enviar seu próprio NGINX Plus Kubernetes Ingress Controller no [GitHub da NGINX Inc.](#).

Visite as implantações do **controlador de entrada do Kubernetes** pasta no repositório kubernetes-ingress no GitHub. Os comandos a seguir serão executados a partir deste diretório de uma cópia local do repositório.

Crie um namespace e uma conta de serviço para o controlador de entrada; ambos são nomeados nginx-ingress:

```
$ kubectl apply -f common/ns-and-sa.yaml
```

Crie um segredo com um certificado TLS e uma chave para o controlador de entrada:

```
$ kubectl apply -f common/default-server-secret.yaml
```

Este certificado e chave são autoassinados e criados pela NGINX Inc. para fins de teste e exemplo. É recomendável usar a sua própria porque essa chave está disponível publicamente.

Opcionalmente, você pode criar um mapa de configuração para personalizar a configuração do NGINX (o mapa de configuração fornecido está em branco; no entanto, você pode ler mais sobre [personalização e anotação](#)):

```
$ kubectl apply -f common/nginx-config.yaml
```

Se o controle de acesso baseado em função (RBAC) estiver habilitado em seu cluster, crie uma função de cluster e vincule-a à conta de serviço. Você deve ser um administrador de cluster para executar esta etapa:

```
$ kubectl apply -f rbac/rbac.yaml
```

Agora implante o controlador de entrada. Dois exemplos de implantações são disponibilizados neste repositório: um Deployment e um DaemonSet. Use uma implantação se você planeja alterar dinamicamente o número de réplicas do controlador de entrada. Use um DaemonSet para implantar um controlador de entrada em cada nó ou em um subconjunto de nós.

Se você planeja usar os manifestos de implantação do NGINX Plus, deve alterar o arquivo YAML e especificar seu próprio registro e imagem.

Para implantação NGINX:

```
$ kubectl apply -f deployment/nginx-ingress.yaml
```

Para implantação do NGINX Plus:

```
$ kubectl apply -f deployment/nginx-plus-ingress.yaml
```

Para NGINX DaemonSet:

```
$ kubectl apply -f daemon-set/nginx-ingress.yaml
```

Para NGINX Plus DaemonSet:

```
$ kubectl apply -f daemon-set/nginx-plus-ingress.yaml
```

Valide se o controlador de ingresso está em execução:

```
$ kubectl get pods --namespace=nginx-ingress
```

Se você criou um DaemonSet, as portas 80 e 443 do controlador de ingresso são mapeadas para as mesmas portas no nó em que o contêiner está em execução. Para acessar o controlador de ingresso, use essas portas e o endereço IP de qualquer um dos nós nos quais o controlador de ingresso está sendo executado. Se você implantou uma implantação, continue com as próximas etapas.

Para os métodos de implantação, há duas opções para acessar os pods do controlador de ingresso. Você pode instruir o Kubernetes a atribuir aleatoriamente uma porta de nó que mapeia para o pod do controlador de entrada. Este é um serviço com o tipo NodePort. A outra opção é criar um serviço com o tipo LoadBalancer. Ao criar um serviço do tipo LoadBalancer, o Kubernetes cria um平衡ador de carga para a plataforma de nuvem especificada, como Amazon Web Services (AWS), Microsoft Azure e Google Cloud Compute.

Para criar um serviço do tipo NodePort, use o seguinte:

```
$ kubectl create -f service/nodeport.yaml
```

Para configurar estaticamente a porta que está aberta para o pod, altere o YAML e adicione o atributo nodePort: {port} à configuração de cada porta que está sendo aberta.

Para criar um serviço do tipo LoadBalancer para Google Cloud Compute ou Azure, use este código:

```
$ kubectl create -f service/loadbalancer.yaml
```

Para criar um serviço do tipo LoadBalancer para Amazon Web Services:

```
$ kubectl create -f service/loadbalancer-aws-elb.yaml
```

Na AWS, o Kubernetes cria um ELB clássico no modo TCP com o protocolo PROXY habilitado.

Você deve configurar o NGINX para usar o protocolo PROXY. Para fazer isso, você pode adicionar o seguinte ao mapa de configuração mencionado anteriormente em referência ao arquivo common/nginx-con.g.yaml.

```
proxy-protocol: "True" real-
ip-header: "proxy_protocol" set-real-ip-
from: "0.0.0.0/0"
```

Em seguida, atualize o mapa de configuração:

```
$ kubectl apply -f common/nginx-config.yaml
```

Agora você pode endereçar o pod por seu NodePort ou fazendo uma solicitação ao平衡ador de carga criado em seu nome.

Discussão

Até o momento, o Kubernetes é a plataforma líder em orquestração e gerenciamento de contêineres. O controlador de entrada é o pod de borda que roteia o tráfego para o restante do seu aplicativo. O NGINX se encaixa perfeitamente nessa função e simplifica a configuração com suas anotações. O projeto NGINX Ingress oferece um controlador de entrada NGINX Open Source pronto para uso a partir de uma imagem do Docker Hub e NGINX Plus por meio de algumas etapas para adicionar seu certificado e chave de repositório. Habilite seu cluster Kubernetes com um controlador NGINX Ingress fornece todos os mesmos recursos do NGINX, mas com os recursos adicionais de rede Kubernetes e DNS para rotear o tráfego.

11.8 Módulo Exportador Prometheus

Problema

Você está implantando o NGINX em um ambiente usando o monitoramento do Prometheus e precisa de estatísticas do NGINX.

Solução

Use o NGINX Prometheus Exporter para coletar estatísticas do NGINX ou NGINX Plus e enviá-las para o Prometheus.

O NGINX Prometheus Exporter Module é escrito em GoLang e distribuído como um [binário no GitHub](#) e pode ser encontrado como uma [imagem do Docker](#) pré-criada no Docker Hub.

Por padrão, o exportador será iniciado para NGINX e colherá apenas o `stub_status` em formação. Para executar o exportador para NGINX Open Source, certifique-se de que o status do stub esteja ativado. Se não for, há mais informações sobre como fazer isso na [Receita 13.1](#), então use o seguinte comando do Docker:

```
docker run -p 9113:9113 nginx/nginx-prometheus-exporter:0.8.0 \
    -nginx.scrape-uri http://{nginxEndpoint}:8080/stub_status
```

Para usar o exportador com o NGINX Plus, um sinalizador deve ser usado para alternar o contexto do exportador porque muito mais dados podem ser coletados da API do NGINX Plus. Você pode aprender como ativar a API NGINX Plus, na [Receita 13.2](#). Use o seguinte comando do Docker para executar o exportador para um ambiente NGINX Plus.

```
docker run -p 9113:9113 nginx/nginx-prometheus-exporter:0.8.0 \
    -nginx.plus -nginx.scrape-uri http://{nginxPlusEndpoint}:8080/api
```

Discussão

O Prometheus é uma solução de monitoramento métrico extremamente comum e muito prevalente no ecossistema Kubernetes. O NGINX Prometheus Exporter Module é um componente bastante simples, no entanto, permite a integração pré-construída entre o NGINX e as plataformas de monitoramento comuns. Com o NGINX, o status do stub não fornece uma grande quantidade de dados, mas dados importantes para fornecer informações sobre a quantidade de trabalho que um nó NGINX está processando. A API NGINX Plus permite muito mais estatísticas sobre o servidor NGINX Plus, todas as quais o exportador envia para o Prometheus. Em ambos os casos, as informações coletadas são dados de monitoramento valiosos, e o trabalho para enviar esses dados ao Prometheus já está feito; você só precisa conectá-lo e aproveitar as informações fornecidas pelas estatísticas do NGINX.

Veja também

[Exportador NGINX Prometheus GitHub](#)

[Status do stub](#)

[API NGINX Plus](#)

[Painel de monitoramento NGINX Plus](#)

CAPÍTULO 12

Modos de implantação de alta disponibilidade

12.0 Introdução

A arquitetura tolerante a falhas separa os sistemas em pilhas idênticas e independentes. Balanceadores de carga como o NGINX são empregados para distribuir a carga, garantindo que o que é provisionado seja utilizado. Os conceitos centrais de alta disponibilidade são balanceamento de carga em vários nós ativos ou um failover ativo-passivo. Aplicativos altamente disponíveis não têm pontos únicos de falha; cada componente deve usar um desses conceitos, incluindo os próprios平衡adores de carga. Para nós, isso significa NGINX. O NGINX foi projetado para funcionar em qualquer configuração: failover ativo múltiplo ou ativo-passivo. Este capítulo detalha técnicas sobre como executar vários servidores NGINX para garantir alta disponibilidade em sua camada de balanceamento de carga.

12.1 Modo HA NGINX Plus

Problema

Você precisa de uma solução de balanceamento de carga altamente disponível (HA).

Solução

Use o modo HA do NGINX Plus com keepalived instalando o pacote nginx-ha-keepalived do repositório NGINX Plus.

Discussão

O pacote nginx-ha-keepalived é baseado em keepalived e gerencia um endereço IP virtual exposto ao cliente. Outro processo é executado no servidor NGINX que garante que o NGINX Plus e o processo keepalived estejam em execução. Keepalive é um

processo que utiliza o Virtual Router Redundancy Protocol (VRRP), enviando pequenas mensagens, geralmente chamadas de pulsações, para o servidor de backup. Se o servidor de backup não receber a pulsação por três períodos consecutivos, o servidor de backup inicia o failover, movendo o endereço IP virtual para si mesmo e tornando-se o principal. Os recursos de failover do nginx-ha-keepalived podem ser configurados para identificar situações de falha personalizadas.

12.2 Balanceadores de carga com balanceamento de carga com DNS

Problema

Você precisa distribuir a carga entre dois ou mais servidores NGINX.

Solução

Use o DNS para fazer o rodízio entre os servidores NGINX adicionando vários endereços IP a um registro DNS A.

Discussão

Ao executar vários平衡adores de carga, você pode distribuir a carga via DNS. O registro A permite que vários endereços IP sejam listados em um único FQDN. O DNS fará o rodízio automaticamente em todos os IPs listados. O DNS também oferece round robin ponderado com registros ponderados, que funciona da mesma forma que round robin ponderado no NGINX, conforme descrito no [Capítulo 2](#). Essas técnicas funcionam muito bem. No entanto, uma armadilha pode ser remover o registro quando um servidor NGINX encontra uma falha. Existem provedores de DNS – Amazon Route 53 para um e Dyn DNS para outro – que oferecem verificações de integridade e failover com sua oferta de DNS, o que alivia esses problemas. Se você estiver usando o DNS para balancear a carga no NGINX, quando um servidor NGINX estiver marcado para remoção, é melhor seguir os mesmos protocolos que o NGINX faz ao remover um servidor upstream. Primeiro, pare de enviar novas conexões para ele removendo seu IP do registro DNS e, em seguida, permita que as conexões sejam drenadas antes de interromper ou encerrar o serviço.

12.3 Balanceamento de carga no EC2

Problema

Você está usando o NGINX na AWS e o NGINX Plus HA não oferece suporte a IPs da Amazon.

Solução

Coloque o NGINX atrás de um AWS NLB configurando um grupo de Auto Scaling de servidores NGINX e vinculando o grupo de Auto Scaling a um grupo de destino e, em seguida, anexe o grupo de destino ao NLB. Como alternativa, você pode colocar servidores NGINX no grupo de destino manualmente usando o console da AWS, a interface de linha de comando ou a API.

Discussão

A solução de alta disponibilidade do NGINX Plus baseada em keepalived não funcionará na AWS porque não oferece suporte ao endereço IP virtual flutuante, pois os endereços IP do EC2 funcionam de maneira diferente. Isso não significa que o NGINX não possa ser HA na nuvem AWS; Na verdade, o oposto é verdadeiro. O AWS NLB é uma oferta de produto da Amazon que fará o balanceamento de carga nativamente em vários datacenters fisicamente separados chamados zonas de disponibilidade, fornecerá verificações de integridade ativas e um endpoint DNS CNAME. Uma solução comum para HA NGINX na AWS é colocar uma camada NGINX atrás do NLB. Os servidores NGINX podem ser adicionados e removidos automaticamente do grupo de destino, conforme necessário. O NLB não substitui o NGINX; há muitas coisas que o NGINX oferece que o NLB não oferece, como vários métodos de平衡amento de carga, limitação de taxa, armazenamento em cache e roteamento de camada 7. O AWS ALB executa o balanceamento de carga da camada 7 com base no caminho do URI e no cabeçalho do host, mas por si só não oferece recursos que o NGINX oferece, como armazenamento em cache WAF, limitação de largura de banda, envio de servidor HTTP/2 e muito mais. Caso o NLB não atenda a sua necessidade, existem muitas outras opções. Uma opção é a solução DNS: o Route 53 da AWS oferece verificações de integridade e failover de DNS.

12.4 Sincronização de configuração do NGINX Plus

Problema

Você está executando uma camada HA NGINX Plus e precisa sincronizar a configuração entre servidores.

Solução

Use o recurso de sincronização de configuração exclusivo do NGINX Plus. Para configurar esse recurso, siga estas etapas:

Instale o pacote nginx-sync do repositório de pacotes NGINX Plus.

Para RHEL ou CentOS:

```
$ sudo yum install nginx-sync
```

Para Ubuntu ou Debian:

```
$ sudo apt-get install nginx-sync
```

Conceda à máquina primária acesso SSH como root às máquinas de mesmo nível.

Gere um par de chaves de autenticação SSH para root e recupere a chave pública:

```
$ sudo ssh-keygen -t rsa -b 2048 $ sudo
cat /root/.ssh/id_rsa.pub ssh-rsa
AAAAB3Nz4rFgt...vgaD root@node1
```

Obtenha o endereço IP do nó primário:

```
$ ip addr 1:
lo: mtu 65536 qdisc noqueue state UNKNOWN group default link/loopback
    00:00:00:00:00:00 brd 00:00:00:00:00:00 inet 127.0.0.1/8 scope host lo
        valid_lft sempre preferido_lft sempre inet6 ::1/128 escopo host valid_lft
            sempre preferido_lft sempre 2: eth0: mtu 1500 qdisc pfifo_fast estado
                UP grupo padrão qlen \
                    1000
```

```
link/ether 52:54:00:34:6c:35 brd ff:ff:ff:ff:ff:ff inet 192.168.1.2/24
    brd 192.168.1.255 scope global eth0 valid_lft sempre preferido_lft
        sempre inet6 fe80::5054:ff:fe34:6c35/64 link de escopo valid_lft
            para sempre preferido_lft para sempre
```

O comando ip addr despejará informações sobre interfaces na máquina.

Desconsidere a interface de loopback, que normalmente é a primeira. Procure o endereço IP após inet para a interface primária. Neste exemplo, o endereço IP é 192.168.1.2.

Distribua a chave pública para o arquivo authorized_keys do usuário raiz em cada nó de mesmo nível e especifique para autorizar apenas a partir do endereço IP primário:

```
$ sudo echo 'from="192.168.1.2" ssh-rsa AAAAB3Nz4rFgt...vgaD \
root@node1' >> /root/.ssh/authorized_keys
```

Adicione a seguinte linha ao /etc/ssh/sshd_config e recarregue o sshd em todos os nós:

```
$ sudo echo 'PermitRootLogin sem senha' >> \
/etc/ssh/sshd_config $
sudo service sshd recarregar
```

Verifique se o usuário root no nó primário pode ssh para cada um dos nós de mesmo nível sem uma senha:

```
$ sudo ssh root@node2.example.com
```

Crie o arquivo de configuração /etc/nginx-sync.conf na máquina primária com a seguinte configuração:

```
NODES="node2.example.com node3.example.com node4.example.com"  
CONFPATHS="/etc/nginx/nginx.conf /etc/nginx/conf.d"  
EXCLUDE="default.conf"
```

Este exemplo de configuração demonstra os três parâmetros de configuração comuns para este recurso: NODES, CONFIGPATHS e EXCLUDE. O parâmetro NODES é definido como uma sequência de nomes de host ou endereços IP separados por espaços; esses são os nós de peer para os quais o primário enviará suas alterações de configuração. O parâmetro CONFIGPATHS indica quais arquivos ou diretórios devem ser sincronizados. Por fim, você pode usar o parâmetro EXCLUDE para excluir arquivos de configuração da sincronização. Em nosso exemplo, o primário envia por push as alterações de configuração do arquivo de configuração principal do NGINX e inclui o diretório /etc/nginx/nginx.conf e /etc/nginx/conf.d para nós pares denominados node2.example.comnode3.example. com e node4.example.com. Se o processo de sincronização encontrar um arquivo chamado default.conf, ele não será enviado aos peers, pois está configurado como EXCLUDE.

Existem parâmetros de configuração avançados para configurar o local do binário NGINX, binário RSYNC, binário SSH, binário diff, local do arquivo de bloqueio e diretório de backup. Há também um parâmetro que utiliza sed para arquivos fornecidos pelo modelo. Para obter mais informações sobre os parâmetros avançados, consulte [Compartilhamento de configuração](#).

Teste sua configuração:

```
$ nginx-sync.sh -h # exibe informações de uso $  
nginx-sync.sh -c node2.example.com # compara configuração com node2 $ nginx-  
sync.sh -C # compara configuração primária com todos os pares $ nginx-sync.  
sh # sincroniza a configuração e recarrega o NGINX nos pares
```

Discussão

Esse recurso exclusivo do NGINX Plus permite gerenciar vários servidores NGINX Plus em uma configuração de alta disponibilidade atualizando apenas o nó primário e sincronizando a configuração com todos os outros nós de mesmo nível. Ao automatizar a sincronização da configuração, você limita o risco de erros ao transferir configurações. O aplicativo nginx-sync.sh fornece algumas salvaguardas para evitar o envio de configurações incorretas aos peers. Eles incluem testar a configuração no primário, criar backups da configuração nos peers e validar a configuração no peer antes de recarregar. Embora seja preferível sincronizar sua configuração usando ferramentas de gerenciamento de configuração ou Docker, o recurso de sincronização de configuração do NGINX Plus é valioso se você ainda não deu um grande salto para gerenciar ambientes dessa maneira.

12.5 Compartilhamento de estado com NGINX Plus e sincronização de zona

Problema

Você precisa do NGINX Plus para sincronizar suas zonas de memória compartilhada em uma frota de servidores altamente disponíveis.

Solução

Configure a sincronização de zona e use o parâmetro sync ao configurar um Zona de memória compartilhada NGINX Plus:

```

fluxo
    { resolvidor 10.0.0.2 válido=20s;

        servidor
            { escuta 9000;
                zone_sync;
                zone_sync_server nginx-cluster.example.com:9000 resolver; #
                    ... Medidas de segurança
            }

        } http
            { upstream my_backend {
                zona my_backend 64k;
                servidor backends.example.com resolver;
                sticky learn zone=sessions:1m
                    create=$upstream_cookie_session
                    lookup=$cookie_session
                    sincronizar;
            }

            servidor
                { escuta 80;
                    local /
                        { proxy_pass http://my_backend;
                    }
                }
            }
    }
}

```

Discussão

O módulo zone_sync é um recurso exclusivo do NGINX Plus que permite que o NGINX Plus realmente faça um cluster. Conforme mostrado na configuração, você deve configurar um servidor de stream configurado como zone_sync. No exemplo, este é o servidor escutando na porta 9000.

O NGINX Plus se comunica com o restante dos servidores definidos pela diretiva zone_sync_server . Você pode definir essa diretiva para um nome de domínio que resolva vários endereços IP para clusters dinâmicos ou definir estaticamente uma série de

diretivas `zone_sync_server`, para evitar pontos únicos de falha. Você deve restringir o acesso ao servidor de sincronização de zona; existem diretivas SSL/TLS específicas para este módulo para autenticação de máquina. A vantagem de configurar o NGINX Plus em um cluster é que você pode sincronizar zonas de memória compartilhada para limitação de taxa, sessões de aprendizado permanente e armazenamento de chave-valor. O exemplo fornecido mostra o parâmetro de sincronização anexado ao final de uma diretiva de aprendizado fixo. Neste exemplo, um usuário está vinculado a um servidor upstream com base em um cookie denominado `session`. Sem o módulo de sincronização de zona, se um usuário fizer uma solicitação para um servidor NGINX Plus diferente, ele poderá perder a sessão. Com o módulo de sincronização de zona, todos os servidores NGINX Plus estão cientes da sessão e a qual servidor upstream está vinculado.

CAPÍTULO 13

Monitoramento avançado de atividades

13.0 Introdução

Para garantir que seu aplicativo esteja sendo executado com desempenho e precisão ideais, você precisa de informações sobre as métricas de monitoramento sobre sua atividade. O NGINX Plus oferece um painel de monitoramento avançado e um feed JSON para fornecer monitoramento detalhado sobre todas as solicitações que chegam ao coração do seu aplicativo. O monitoramento de atividades do NGINX Plus fornece informações sobre solicitações, pools de servidores upstream, cache, integridade e muito mais. Este capítulo detalha o poder e as possibilidades do painel do NGINX Plus, da API do NGINX Plus e do módulo de status de stub de código aberto.

13.1 Ativar status de stub de código aberto NGINX

Problema

Você precisa habilitar o monitoramento básico para NGINX.

Solução

Habilite o módulo `stub_status` em um bloco de localização dentro de um servidor HTTP NGINX:

```
localização /stub_status {  
    stub_status;  
    permitir 127.0.0.1;  
    negar tudo; # Defina  
    as restrições de IP conforme apropriado  
}
```

Teste sua configuração fazendo uma solicitação para o status:

```
$ curl localhost/stub_status
Conexões ativas: 1 servidor
aceita solicitações tratadas
1 1 1
Leitura: 0 Escrita: 1 Aguardando: 0
```

Discussão

O módulo stub_status permite algum monitoramento básico do servidor NGINX OSS.

As informações retornadas fornecem informações sobre o número de conexões ativas, bem como o total de conexões aceitas, conexões tratadas e solicitações atendidas. O número atual de conexões sendo lidas, gravadas ou em estado de espera também é mostrado. As informações fornecidas são globais e não são específicas do servidor para no qual a diretiva stub_status está definida. Isso significa que você pode hospedar o status em um servidor protegido. Por motivos de segurança, bloqueamos todo o acesso ao recurso de monitoramento, exceto o tráfego local. Este módulo fornece contagens de conexões ativas como variáveis incorporadas para uso em logs e em outros lugares. Essas variáveis são \$connections_active, \$connections_reading, \$connections_writing e \$connections_waiting.

13.2 Habilitando o painel de monitoramento NGINX Plus

Problema

Você precisa de métricas detalhadas sobre o tráfego que flui pelo NGINX Plus servidor.

Solução

Utilize o painel de monitoramento de atividades em tempo real:

```
servidor {
    # ...
    local /api { api
        [write=on];
        # Diretivas que limitam o acesso à API
        # Veja o capítulo 7
    }

    local = /dashboard.html {
        root /usr/share/nginx/html;
    }
}
```

A configuração do NGINX Plus atende ao painel de monitoramento de status do NGINX Plus. Essa configuração configura um servidor HTTP para servir a API e o painel de status. O painel é servido como conteúdo estático fora do diretório /usr/share/nginx/html. O painel faz solicitações à API em /api/ para recuperar e exibir o status em tempo real.

Discussão

O NGINX Plus fornece um painel avançado de monitoramento de status. Esse painel de status fornece um status detalhado do sistema NGINX, como número de conexões ativas, tempo de atividade, informações do pool de servidores upstream e muito mais. Para um vislumbre do console, veja a [Figura 13-1](#).

O NGINX Controller fornece uma visão centrada em aplicativos para monitorar uma frota de servidores NGINX Plus em diferentes locais. A guia Infraestrutura fornece informações e métricas sobre servidores, ambientes e aplicativos, em uma única interface. Para um vislumbre do console, veja a [Figura 13-2](#).

A página inicial do painel de status fornece uma visão geral de todo o sistema. Clicar na guia Zonas HTTP lista detalhes sobre todos os servidores HTTP configurados na configuração do NGINX, detalhando o número de respostas de 1XX a 5XX e um total geral, bem como solicitações por segundo e a taxa de transferência de tráfego atual. A guia HTTP Upstreams detalha o status do servidor upstream: se o servidor estava em um estado de falha, quantas solicitações ele atendeu e um total de quantas respostas foram atendidas pelo código de status, bem como outras estatísticas, como quantas verificações de integridade passou ou falhou. A guia Zonas TCP/UDP detalha a quantidade de tráfego que flui pelos fluxos TCP ou UDP e o número de conexões. A guia Upstreams TCP/UDP mostra informações sobre quanto cada um dos servidores upstream nos pools upstream TCP/UDP está atendendo, bem como detalhes de aprovação e falha da verificação de integridade e tempos de resposta. A guia Caches exibe informações sobre a quantidade de espaço utilizada para o cache; a quantidade de tráfego servido, escrito e ignorado; bem como a taxa de acerto. O painel de status do NGINX é inestimável para monitorar o coração de seus aplicativos e fluxo de tráfego.

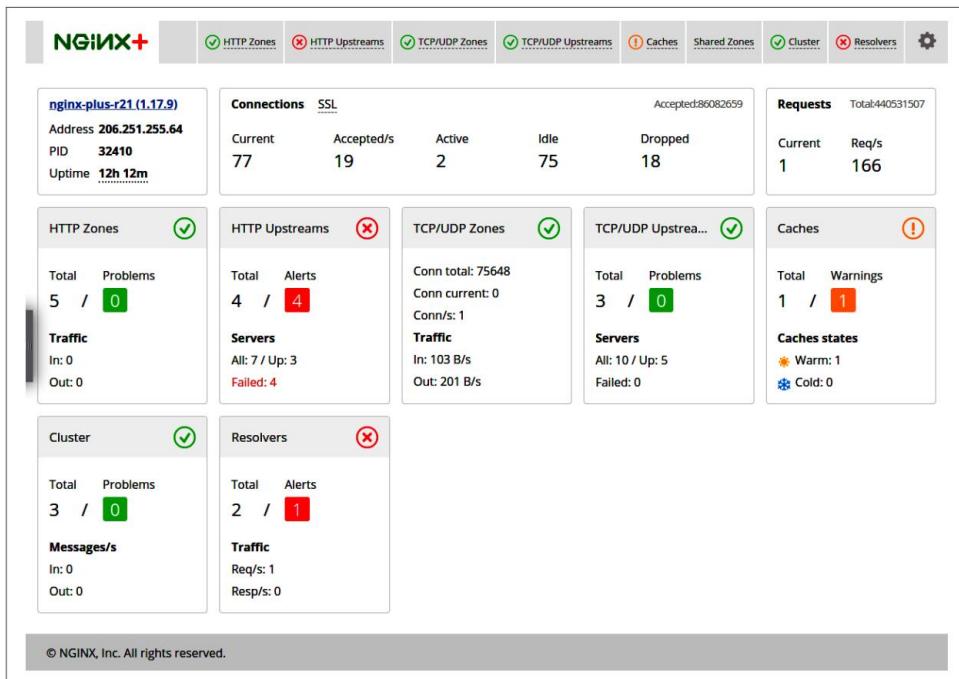


Figura 13-1. O painel de status do NGINX Plus

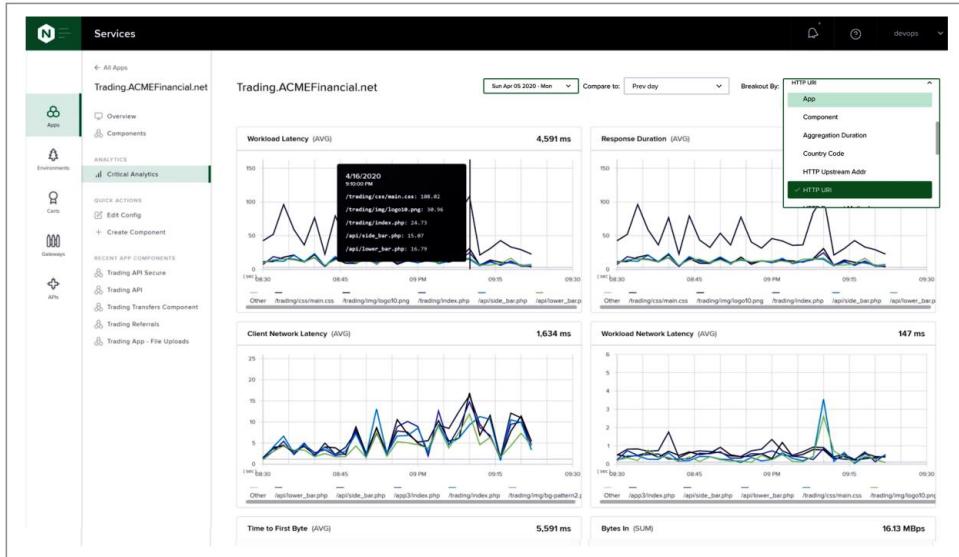


Figura 13-2. Um painel de análise centrado no aplicativo do NGINX Controller

Veja também

[Demonstração do painel de status do NGINX Plus](#)

[Página do produto do controlador NGINX](#)

13.3 Coletando métricas usando a API NGINX Plus

Problema

Você precisa de acesso à API para as métricas de detalhes fornecidas pelo painel de status do NGINX Plus.

Solução

Utilize a API RESTful para coletar métricas. Os exemplos canalizam a saída por meio de json_pp para facilitar a leitura:

```
$ curl "demo.nginx.com/api/3/" | json_pp [  
  
    "nginx",  
    "processos",  
    "conexões", "ssl",  
    "slabs", "http",  
    "stream"  
  
]
```

A chamada curl solicita o nível superior da API, que exibe outras partes da API.

Para obter informações sobre o servidor NGINX Plus, use o /api/{version}/nginx URL:

```
$ curl "demo.nginx.com/api/3/nginx" | json_pp {  
  
    "version" : "1.15.2", "ppid" :  
    79909, "build" : "nginx-plus-  
r16", "pid" : 77242, "address" :  
    "206.251.255.64", "timestamp" :  
    "2018-09-29T23:12:20.525Z",  
    "load_timestamp" : "2018-09-29T10:00:00.404Z",  
    "geração" : 2  
}
```

Para limitar as informações retornadas pela API, use argumentos:

```
$ curl "demo.nginx.com/api/3/nginx?fields=version,build" \  
| json_pp  
{
```

```

    "build" : "nginx-plus-r16", "version" :
    "1.15.2"
}

```

Você pode solicitar estatísticas de conexão do URI /api/{version}/connections :

```

$ curl "demo.nginx.com/api/3/connections" | json_pp {

    "ativo" : 3,
    "inativo" : 34,
    "descartado" : 0,
    "aceito" : 33614951
}

```

Você pode coletar estatísticas de solicitação do URI /api/{version}/http/requests :

```

$ curl "demo.nginx.com/api/3/http/requests" | json_pp {

    "total" : 52107833,
    "atual" : 2
}

```

Você pode recuperar estatísticas sobre uma zona de servidor específica usando o URI /api/{version}/ http/server_zones/{httpServerZoneName} :

```

$ curl "demo.nginx.com/api/3/http/server_zones/hg.nginx.org" \
| json_pp
{
    "respostas" :
        { "1xx" : 0,
          "5xx" : 0, "3xx" :
            938, "4xx" :
            341, "total" :
            25245, "2xx" : 23966
        },
    "solicitações" : 25252,
    "descartadas" : 7,
    "recebidas" : 5758103,
    "processando" : 0,
    "enviadas" : 359428196
}

```

A API pode retornar qualquer dado que você possa ver no painel. Tem profundidade e segue um padrão lógico. Você pode encontrar links para recursos no final desta receita.

Discussão

A API do NGINX Plus pode retornar estatísticas sobre muitas partes do servidor NGINX Plus. Você pode coletar informações sobre o servidor NGINX Plus, seus processos, conexões e slabs. Você também pode encontrar informações sobre servidores http e stream em execução no NGINX, incluindo servidores, upstreams, servidores upstream e

armazenamentos de valores-chave, bem como informações e estatísticas sobre zonas de cache HTTP. Isso fornece a você ou a agregadores de métricas de terceiros uma visão detalhada do desempenho do seu servidor NGINX Plus.

Com a API do NGINX Controller, você pode consultar métricas de vários servidores NGINX Plus de uma só vez. As métricas fornecidas pelo Controller oferecem uma visão diferente e centrada no aplicativo de suas métricas.

Veja também

[Documentação do módulo de API HTTP NGINX](#)

[IU do Swagger da API NGINX](#)

[Página do produto do controlador NGINX](#)

CAPÍTULO 14

Depuração e solução de problemas com logs de acesso, logs de erros e rastreamento de solicitações

14.0 Introdução

O registro em log é a base para entender seu aplicativo. Com o NGINX, você tem grande controle sobre o registro de informações significativas para você e seu aplicativo. O NGINX permite dividir os logs de acesso em diferentes arquivos e formatos para diferentes contextos e alterar o nível de log do log de erros para obter uma compreensão mais profunda do que está acontecendo. A capacidade de transmitir logs para um servidor centralizado é inerente ao NGINX por meio de seus recursos de log Syslog. O NGINX e o NGINX Plus também permitem o rastreamento de solicitações à medida que elas passam por um sistema. Neste capítulo, discutimos logs de acesso e erro, streaming pelo protocolo Syslog e rastreamento de solicitações de ponta a ponta com identificadores de solicitação gerados pelo NGINX e OpenTracing.

14.1 Configurando Logs de Acesso

Problema

Você precisa configurar os formatos de log de acesso para adicionar variáveis incorporadas aos seus logs de solicitação.

Solução

Configure um formato de registro de acesso:

```

http
{ log_format geoproxy
    ['$time_local'] $remote_addr'
    '$realip_remote_addr $remote_user'
    '$proxy_protocol_server_addr $proxy_protocol_server_port'
    '$request_method $server_protocol '$scheme $server_name $uri $status
    '$request_time $body_bytes $sent' geoip_region '"$geoip_city"
    $http_x_forwarded_for' '$upstream_status $upstream_response_time'
    '"$http_referer" "$http_user_agent";

#
# ...
}

```

Essa configuração de formato de log é denominada geoproxy e usa várias variáveis incorporadas para demonstrar o poder do log do NGINX. Essa configuração mostra a hora local no servidor quando a solicitação foi feita, o endereço IP que abriu a conexão e o IP do cliente, conforme o NGINX entende por instruções geoip_proxy ou realip_header .

As variáveis prefixadas com \$proxy_protocol_server_ fornecem informações sobre o servidor do cabeçalho do protocolo PROXY, quando o parâmetro proxy_protocol é usado na diretiva listen do servidor. \$remote_user mostra o nome de usuário do usuário, autenticado por autenticação básica, seguido do método e protocolo de solicitação, bem como o esquema, como HTTP ou HTTPS. A correspondência do nome do servidor é registrada, bem como o URI de solicitação e o código de status de retorno.

As estatísticas registradas incluem o tempo de processamento em milissegundos e o tamanho do corpo enviado ao cliente. As informações sobre o país, região e cidade são registradas. O cabeçalho HTTP X-Forwarded-For é incluído para mostrar se a solicitação está sendo encaminhada por outro proxy. O módulo upstream habilita algumas variáveis incorporadas que usamos que mostram o status retornado do servidor upstream e quanto tempo a solicitação upstream leva para retornar. Por fim, registramos algumas informações sobre de onde o cliente foi encaminhado e qual navegador o cliente está usando.

A diretiva log_format é válida apenas no contexto HTTP. Um parâmetro de escape opcional pode especificar que tipo de escape é feito na string; default, json e none são valores de escape. O valor none desabilita o escape. Para o escape padrão , os caracteres "", "\" e outros caracteres com valores menores que 32 ou acima de 126 são escapados como "\XX". Se o valor da variável não for encontrado, um hífen ("") será registrado .

Para o escape json , todos os caracteres não permitidos em strings JSON serão escapados: ter caracteres "" e "\" são escapados como "\"" e "\\\", caracteres com valores menores que 32 são escapou como "\n", "\r", "\t", "\b", "\f" ou "\u00XX".

Essa configuração de log renderiza uma entrada de log semelhante a esta:

```
[25/Nov/2016:16:20:42 +0000] 10.0.1.16 192.168.0.122 Derek GET
HTTP/1.1 http www.example.com / 200 0.001 370 USA MI "Ann Arbor"
- 200 0.001 "-" "curl /7.47.0"
```

Para usar esse formato de log, use a diretiva `access_log`, fornecendo um caminho do arquivo de log e o nome do formato geoproxy como parâmetros:

```
servidor
{
    access_log /var/log/nginx/access.log geaproxy;
    # ...
}
```

A diretiva `access_log` usa um caminho de arquivo de log e o nome do formato como parâmetros. Esta diretiva é válida em muitos contextos e em cada contexto pode ter um caminho de log e/ou formato de log diferente. Parâmetros nomeados como `buffer`, `flush` e `gzip` configuram com que frequência os logs são gravados no arquivo de log e se o arquivo é compactado ou não. Um parâmetro chamado `if` existe e recebe uma condição; se a condição for avaliada como 0 ou string vazia, o acesso não será registrado.

Discussão

O módulo de log no NGINX permite configurar formatos de log para muitos cenários diferentes para logar em vários arquivos de log conforme você achar melhor. Você pode achar útil configurar um formato de log diferente para cada contexto, onde você usa módulos diferentes e emprega variáveis incorporadas desses módulos, ou um formato único e abrangente que fornece todas as informações que você deseja. Também é possível fazer login em JSON ou XML, desde que você construa a string de formato dessa maneira. Esses logs ajudarão você a entender seus padrões de tráfego, uso de clientes, quem são seus clientes e de onde eles vêm. Os logs de acesso também podem ajudá-lo a encontrar atrasos nas respostas e problemas com servidores upstream ou URIs específicos. Os logs de acesso podem ser usados para analisar e reproduzir padrões de tráfego em ambientes de teste para imitar a interação real do usuário. Há possibilidades ilimitadas de logs ao solucionar problemas, depurar ou analisar seu aplicativo ou mercado.

14.2 Configurando Logs de Erros

Problema

Você precisa configurar o registro de erros para entender melhor os problemas com seu NGINX servidor.

Solução

Use a diretiva `error_log` para definir o caminho do log e o nível do log:

```
error_log /var/log/nginx/error.log advertir;
```

A diretiva `error_log` requer um caminho; no entanto, o nível de log é opcional e o padrão é `error`. Esta diretiva é válida em todos os contextos, exceto para instruções `if`.

Os níveis de log disponíveis são `debug`, `info`, `notice`, `warning`, `error`, `crit`, `alert` ou `emerg`. A ordem em que esses níveis de log foram introduzidos também é a ordem de gravidade do menor para o maior. O nível de log de depuração só estará disponível se o NGINX estiver configurado com o sinalizador `--with-debug`.

Discussão

O log de erros é o primeiro lugar a ser observado quando os arquivos de configuração não estão funcionando corretamente. O log também é um ótimo lugar para encontrar erros produzidos por servidores de aplicativos como FastCGI. Você pode usar o log de erros para depurar conexões para o trabalhador, alocação de memória, IP do cliente e servidor. O log de erros não pode ser formatado. No entanto, segue um formato específico de data, seguido do nível e, em seguida, da mensagem.

14.3 Encaminhamento para Syslog

Problema

Você precisa encaminhar seus logs para um ouvinte Syslog para agregar logs a um serviço centralizado.

Solução

Use as diretivas `error_log` e `access_log` para enviar seus logs para um ouvinte Syslog:

```
error_log syslog:server=10.0.1.42 depurar;  
access_log syslog:server=10.0.1.42,tag=nginx,severity=info geoproxy;
```

O parâmetro `syslog` para as diretivas `error_log` e `access_log` é seguido por dois pontos e várias opções. Essas opções incluem o sinalizador de servidor necessário que denota o IP, nome DNS ou soquete Unix ao qual se conectar, bem como sinalizadores opcionais, como facilidade, gravidade, tag e `nohostname`. A opção de servidor recebe um número de porta, juntamente com endereços IP ou nomes DNS. No entanto, o padrão é UDP 514. A opção de facilidade refere-se à facilidade da mensagem de log definida como uma das 23 definidas no padrão RFC para Syslog; o valor padrão é `local7`. A opção `tag` marca a mensagem com um valor. Esse valor padrão é `nginx`. severidade assume como padrão `info` e denota a severidade da mensagem que está sendo enviada. O sinalizador `nohostname` é desativado, adicionando o campo `hostname` no cabeçalho da mensagem Syslog e não assume um valor.

Discussão

Syslog é um protocolo padrão para enviar mensagens de log e coletar esses logs em um único servidor ou conjunto de servidores. Enviar logs para um local centralizado ajuda na depuração quando você tem várias instâncias do mesmo serviço em execução em vários hosts. Isso é chamado de agregação de logs. A agregação de logs permite que você visualize logs juntos em um só lugar sem ter que pular de servidor para servidor e juntar mentalmente arquivos de log por carimbo de data/hora. Uma pilha de agregação de log comum é o Elastic-search, Logstash e Kibana, também conhecido como ELK Stack. O NGINX facilita o streaming desses logs para seu ouvinte Syslog com as diretivas `access_log` e `error_log`.

14.4 Solicitar Rastreamento

Problema

Você precisa correlacionar os logs do NGINX com os logs do aplicativo para ter uma compreensão completa de uma solicitação.

Solução

Use a variável de identificação de solicitação e passe-a para seu aplicativo para registrar também:

```
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'; back-end upstream { server 10.0.0.42;

} servidor
{ escuta 80;
# Adiciona o cabeçalho X-Request-ID à resposta ao cliente add_header X-
Request-ID $request_id; local / { proxy_pass http://backend; # Envia o
cabeçalho X-Request-ID para o aplicativo proxy_set_header X-Request-ID
$request_id; trace access_log /var/log/nginx/access_trace.log;

}
```

Nesta configuração de exemplo, um rastreamento nomeado `log_format` é configurado e a variável `$request_id` é usada no log. Essa variável `$request_id` também é passada para o aplicativo `upstream` pelo uso da diretiva `proxy_set_header` para adicionar o ID da solicitação a um cabeçalho ao fazer a solicitação `upstream`. O ID da solicitação também é passado de volta ao cliente por meio do uso da diretiva `add_header` definindo o ID da solicitação em um cabeçalho de resposta.

Discussão

Disponível no NGINX Plus R10 e NGINX versão 1.11.0, o \$request_id fornece uma string gerada aleatoriamente de 32 caracteres hexadecimais que podem ser usados para identificar solicitações de forma exclusiva. Ao passar esse identificador para o cliente e também para o aplicativo, você pode correlacionar seus logs com as solicitações feitas. Do cliente front-end, você receberá essa string exclusiva como um cabeçalho de resposta e poderá usá-la para pesquisar em seus logs as entradas correspondentes. Você precisará instruir seu aplicativo a capturar e registrar esse cabeçalho em seus logs de aplicativo para criar um verdadeiro relacionamento de ponta a ponta entre os logs. Com esse avanço, o NGINX possibilita rastrear solicitações por meio de sua pilha de aplicativos.

14.5 OpenTracing para NGINX

Problema

Você tem um servidor de rastreamento compatível com OpenTracing e deseja integrar o NGINX ou o NGINX Plus.

Solução

Verifique se você tem um servidor compatível com OpenTrace disponível e se o cliente correto está instalado no nó NGINX ou NGINX Plus.

Será necessário um arquivo de configuração de plug-in para o servidor compatível com OpenTrace específico. Esta solução irá demonstrar Jaeger e Zipkin.

Um exemplo de configuração de plug-in Jaeger chamado /etc/jaeger/jaeger-con g.json é o seguinte:

```
{
  "service_name": "nginx",
  "sampler": { "type": "const",
    "param": 1 },
  "reporter": {
    "localAgentHostPort": "Jaeger-server-IP-
address:6831"
  }
}
```

Um exemplo de configuração de plug-in Zipkin chamado /etc/zipkin/zipkin-con g.json é o seguinte:

```
{
  "service_name": "nginx",
  "collector_host": "Zipkin-server-IP-address",
```

```

    "collector_port": 9411
}

```

Se o NGINX Plus estiver sendo usado, instale o módulo OpenTracing do repositório NGINX Plus seguindo o [Guia de administração do NGINX Plus](#).

Se o NGINX de código aberto estiver sendo usado, visite o [NGINX OpenTracing Module Releases](#) página para encontrar um módulo dinâmico pré-construído compatível com seu sistema ou compile o módulo junto com o NGINX a partir da fonte. Como alternativa, em um ambiente docker, uma imagem chamada opentracing/nginx-opentracing está disponível no Docker Hub e pode ser usada para iniciar seus testes.

Ao usar um módulo carregado dinamicamente, que inclui a instalação do NGINX Plus, certifique-se de carregá-lo dentro de sua configuração do NGINX adicionando a seguinte diretiva `load_module`, para informar ao NGINX onde encontrar o módulo no sistema de arquivos.

Como lembrete, a diretiva `load_module` é válida apenas no contexto principal (nível superior).

```
módulos load_module/nginx_opentracing_module.so;
```

Quando um servidor compatível com OpenTrace estiver atendendo, um cliente instalado no nó NGINX e a configuração do plug-in em vigor e o módulo NGINX carregado, o NGINX poderá ser configurado para iniciar o rastreamento de solicitações. O exemplo a seguir fornece exemplos de carregamento de um rastreador e configuração do NGINX para marcar solicitações. A diretiva para carregar um plug-in rastreador está incluída; com local padrão para plug-ins Jaeger e Zipkin e os arquivos de configuração fornecidos acima. Remova o comentário do exemplo de fornecedor apropriado para o caso de uso.

```

# Carrega um rastreador de
fornecedor #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;

# Habilite o rastreamento para todas as
requisições em opentracing;

# Definir tags adicionais que capturam o valor das variáveis NGINX 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;

servidor
{ escuta 9001;

```

```
local / {  
    # O nome da operação usado para OpenTracing Spans tem como padrão  
    # o nome do bloco 'location', # mas descomente esta diretiva para  
    # personalizá-la. #opentracing_operation_name $uri;  
  
    # Propagar o contexto Span ativo upstream, # para que  
    # o rastreamento possa ser continuado pelo backend.  
    opentracing_propagate_context;  
  
    # Exemplo de serviço de localização do  
    # aplicativo proxy_pass http://10.0.0.2:8080;  
}  
}
```

Discussão

Uma configuração do OpenTracing não é trivial, mas fornece um enorme valor em áreas de monitoramento distribuído de desempenho e transações. Essas ferramentas permitem que as equipes forneçam efetivamente a causa raiz e a análise de dependência para identificar áreas problemáticas com dados. É natural que o NGINX sirva como um gateway de API, roteando e autorizando requisições entre aplicações e, portanto, possui informações integrais para rastrear requisições através de um sistema complexo.

O NGINX pode marcar uma solicitação com qualquer variável disponível para si mesmo, o que permite que o usuário do sistema de rastreamento tenha uma visão completa de como uma solicitação se comporta. Este exemplo forneceu uma amostra limitada do uso do OpenTracing para uma solicitação com proxy. Pode-se imaginar a quantidade de dados que podem ser coletados do NGINX, pois a diretiva opentracing_tag é válida nos contextos HTTP, servidor e local.

Veja também

[Módulo OpenTracing NGINX](#)

[Guia de administração do módulo dinâmico NGINX Plus OpenTracing](#)

[Guia de plug-in do módulo Datadog OpenTracing NGINX](#)

[NGINX OpenTracing para NGINX e NGINX Plus Blog](#)

[Blog do NGINX OpenTracing para NGINX Ingress Controller](#)

CAPÍTULO 15

Ajuste de desempenho

15.0 Introdução

O ajuste do NGINX fará de você um artista. O ajuste de desempenho de qualquer tipo de servidor ou aplicativo sempre depende de vários itens variáveis, como, mas não limitado a, ambiente, caso de uso, requisitos e componentes físicos envolvidos. É comum praticar o ajuste orientado a gargalos, o que significa testar até atingir um gargalo, determinar o gargalo, ajustar as limitações e repetir até atingir os requisitos de desempenho desejados. Neste capítulo, sugerimos fazer medições ao ajustar o desempenho testando com ferramentas automatizadas e medindo resultados. Este capítulo também cobre o ajuste de conexão para manter as conexões abertas aos clientes, bem como aos servidores upstream, e servir mais conexões ajustando o sistema operacional.

15.1 Automatizando Testes com Drivers de Carga

Problema

Você precisa automatizar seus testes com um driver de carga para ganhar consistência e repetibilidade em seus testes.

Solução

Use uma ferramenta de teste de carga HTTP, como Apache JMeter, Locust, Gatling ou qualquer outra que sua equipe tenha padronizado. Crie uma configuração para sua ferramenta de teste de carga que execute um teste abrangente em seu aplicativo da web. Execute seu teste em relação ao seu serviço. Revise as métricas coletadas da execução para estabelecer uma linha de base. Aumente lentamente a simultaneidade do usuário emulado para imitar o uso típico de produção e identificar

pontos de melhoria. Ajuste o NGINX e repita este processo até alcançar os resultados desejados.

Discussão

Usar uma ferramenta de teste automatizada para definir seu teste oferece um teste consistente para criar métricas ao ajustar o NGINX. Você deve ser capaz de repetir seu teste e medir ganhos ou perdas de desempenho para conduzir a ciência. A execução de um teste antes de fazer qualquer ajuste na configuração do NGINX para estabelecer uma linha de base fornece uma base para trabalhar para que você possa medir se a alteração de configuração melhorou o desempenho ou não. A medição de cada alteração feita o ajudará a identificar de onde vêm suas melhorias de desempenho.

15.2 Mantendo Conexões Abertas aos Clientes

Problema

Você precisa aumentar o número de solicitações permitidas em uma única conexão de clientes e aumentar a quantidade de tempo que as conexões ociosas podem persistir.

Solução

Use as diretivas `keepalive_requests` e `keepalive_timeout` para alterar o número de solicitações que podem ser feitas em uma única conexão e alterar a quantidade de tempo que as conexões ociosas podem permanecer abertas:

```
http
{
    keepalive_requests 320;
    keepalive_timeout 300s; #
    ...
}
```

A diretiva `keepalive_requests` é padronizada para 100, e a diretiva `keepalive_timeout` é padronizada para 75 segundos.

Discussão

Normalmente, o número padrão de solicitações em uma única conexão atenderá às necessidades do cliente porque os navegadores atualmente têm permissão para abrir várias conexões com um único servidor por FQDN. O número de conexões abertas paralelas a um domínio ainda é limitado tipicamente a um número menor que 10, portanto, nesse sentido, muitas solicitações em uma única conexão ocorrerão. Um truque para HTTP/1.1 comumente empregado por redes de entrega de conteúdo é criar vários nomes de domínio apontados para o servidor de conteúdo e alternar qual nome de domínio é usado dentro do código para permitir que o navegador

abrir mais conexões. Você pode achar essas otimizações de conexão úteis se seu aplicativo de front-end pesquisa continuamente seu aplicativo de back-end em busca de atualizações, porque uma conexão aberta que permite um número maior de solicitações e permanece aberta por mais tempo limitará o número de conexões que precisam ser feitas.

15.3 Mantendo as conexões abertas a montante

Problema

Você precisa manter as conexões abertas para os servidores upstream para reutilização e melhorar seu desempenho.

Solução

Use a diretiva keepalive no contexto upstream para manter as conexões abertas para os servidores upstream para reutilização:

```
proxy_http_version 1.1;
proxy_set_header Conexão "";

back-end upstream
{ server 10.0.0.42;
servidor 10.0.2.56;

manter vivo 32;
}
```

A diretiva keepalive no contexto upstream ativa um cache de conexões que permanecem abertas para cada trabalhador NGINX. A diretiva indica o número máximo de conexões inativas para manter abertas por trabalhador. As diretivas de módulos proxy usadas acima do bloco upstream são necessárias para que a diretiva keepalive funcione corretamente para conexões de servidor upstream. A diretiva proxy_http_version instrui o módulo proxy a usar o HTTP versão 1.1, que permite que várias solicitações sejam feitas em série em uma única conexão enquanto estiver aberta. A diretiva proxy_set_header instrui o módulo proxy a remover o cabeçalho padrão de fechamento, permitindo que a conexão permaneça aberta.

Discussão

Você deseja manter as conexões abertas para servidores upstream para economizar o tempo necessário para iniciar a conexão, permitindo que o processo de trabalho passe diretamente para fazer uma solicitação em uma conexão ociosa. É importante observar que o número de conexões abertas pode exceder o número de conexões especificado na diretiva keepalive porque conexões abertas e conexões inativas não são a mesma coisa. O número de conexões keepalive deve ser mantido pequeno o suficiente para permitir outras conexões.

conexões de entrada para seu servidor upstream. Este pequeno truque de ajuste do NGINX pode economizar alguns ciclos e melhorar seu desempenho.

15.4 Respostas de buffer

Problema

Você precisa armazenar respostas em buffer entre servidores upstream e clientes na memória para evitar gravar respostas em arquivos temporários.

Solução

Ajuste as configurações de buffer de proxy para permitir que NGINX a memória armazene corpos de resposta em buffer:

```
servidor
{
    proxy_buffering
        ativado; proxy_buffer_size
        8k; proxy_buffers 8 32k;
        proxy_busy_buffer_size 64k; #
        ...
}
```

A diretiva proxy_buffering está ativada ou desativada; por padrão está ativado. O proxy_buffer_size denota o tamanho de um buffer usado para ler a primeira parte da resposta, cabeçalhos, do servidor proxy e o padrão é 4k ou 8k, dependendo da plataforma. A diretiva proxy_buffers recebe dois parâmetros: o número de buffers e o tamanho dos buffers. Por padrão, a diretiva proxy_buffers é definida para um número de 8 buffers de tamanho 4k ou 8k, dependendo da plataforma. A diretiva proxy_busy_buffer_size limita o tamanho dos buffers que podem estar ocupados, enviando uma resposta ao cliente enquanto a resposta não é totalmente lida. O tamanho padrão do buffer ocupado é o dobro do tamanho de um buffer proxy ou do tamanho do buffer. Se o buffer de proxy estiver desabilitado, a solicitação não poderá ser enviada para o próximo servidor upstream em caso de falha porque o NGINX já começou a enviar o corpo da solicitação.

Discussão

Os buffers de proxy podem melhorar muito o desempenho do proxy, dependendo do tamanho típico de seus corpos de resposta. O ajuste dessas configurações pode ter efeitos adversos e deve ser feito observando o tamanho médio do corpo retornado e realizando testes completos e repetidos. Buffers extremamente grandes, configurados quando não são necessários, podem consumir a memória da sua caixa NGINX. Você pode definir essas configurações para locais específicos que são conhecidos por retornar corpos de resposta grandes para um desempenho ideal.

[Veja também](#)

[Documentação do proxy_request_buffering do NGINX](#)

15.5 Registros de acesso em buffer

Problema

Você precisa armazenar em buffer os logs para reduzir a oportunidade de bloqueios ao processo de trabalho do NGINX quando o sistema estiver sob carga.

Solução

Defina o tamanho do buffer e o tempo de liberação de seus logs de acesso:

```
http
{
    access_log /var/log/nginx/access.log buffer principal=32k
        flush=1m gzip=1;
}
```

O parâmetro buffer da diretiva access_log indica o tamanho de um buffer de memória que pode ser preenchido com dados de log antes de ser gravado no disco. O parâmetro flush da diretiva access_log define a maior quantidade de tempo que um log pode permanecer em um buffer antes de ser gravado no disco. Ao usar o gzip, os logs são compactados antes de serem gravados no log — os valores de nível 1 (mais rápido, menos compactação) a 9 (mais lenta, melhor compactação) são válidos.

Discussão

O armazenamento de dados de log na memória pode ser um pequeno passo para a otimização. No entanto, para sites e aplicativos muito solicitados, isso pode fazer um ajuste significativo no uso do disco e da CPU. Ao usar o parâmetro buffer para a diretiva access_log , os logs serão gravados no disco se a próxima entrada de log não couber no buffer. Se estiver usando o parâmetro flush em conjunto com o parâmetro buffer , os logs serão gravados no disco quando os dados no buffer forem mais antigos que o tempo especificado. Ao seguir o log e com o buffer ativado, você poderá ver atrasos de até o período de tempo especificado pelo parâmetro flush .

15.6 Ajuste do SO

Problema

Você precisa ajustar seu sistema operacional para aceitar mais conexões para lidar com cargas de pico ou sites com alto tráfego.

Solução

Verifique a configuração do kernel para `net.core.somaxconn`, que é o número máximo de conexões que podem ser enfileiradas pelo kernel para o NGINX processar. Se você definir esse número acima de 512, precisará definir o parâmetro backlog da diretiva de escuta em sua configuração do NGINX para corresponder. Um sinal de que você deve olhar para esta configuração do kernel é se o seu log do kernel explicitamente diz para fazer isso. O NGINX lida com conexões muito rapidamente e, para a maioria dos casos de uso, você não precisará alterar essa configuração.

Aumentar o número de descritores de arquivos abertos é uma necessidade mais comum. No Linux, um identificador de arquivo é aberto para cada conexão; e, portanto, o NGINX pode abrir dois se você o estiver usando como proxy ou平衡ador de carga devido ao upstream de conexão aberta. Para atender a um grande número de conexões, pode ser necessário aumentar o limite do descritor de arquivo em todo o sistema com a opção de kernel `sys.fs.file_max` ou, para o usuário do sistema, o NGINX está sendo executado como no arquivo `/etc/security/limits.conf`. Ao fazer isso, você também desejará aumentar o número de `worker_connections` e `worker_rlimit_nofile`. Ambas as configurações são diretivas na configuração do NGINX.

Habilite mais portas efêmeras. Quando o NGINX atua como um proxy reverso ou balanceador de carga, cada conexão upstream abre uma porta temporária para tráfego de retorno. Dependendo da configuração do seu sistema, o servidor pode não ter o número máximo de portas efêmeras abertas. Para verificar, revise a configuração do kernel `net.ipv4.ip_local_port_range`. A configuração é um intervalo de portas de limite inferior e superior. Normalmente, não há problema em definir essa configuração do kernel de 1024 a 65535. 1024 é onde as portas TCP registradas param e 65535 é onde as portas dinâmicas ou efêmeras param.

Lembre-se de que seu limite inferior deve ser maior que a porta de serviço de escuta aberta mais alta.

Discussão

Ajustar o sistema operacional é um dos primeiros lugares que você procura quando começa a ajustar um grande número de conexões. Existem muitas otimizações que você pode fazer em seu kernel para seu caso de uso específico. No entanto, o ajuste do kernel não deve ser feito por capricho, e as alterações devem ser medidas pelo seu desempenho para garantir que as alterações estejam ajudando. Como dito anteriormente, você saberá quando é hora de começar a ajustar seu kernel a partir de mensagens registradas no log do kernel ou quando o NGINX registra explicitamente uma mensagem em seu log de erros.

CAPÍTULO 16

Introdução ao controlador NGINX

16.0 Introdução

O NGINX Controller é um plano de controle centrado na aplicação para seus ambientes de aplicação. O Controller fornece uma interface que permite visualizar e configurar uma frota inteira de servidores NGINX Plus, independentemente de sua localização física. O Controller permite que as equipes se concentrem menos na configuração bruta do NGINX Plus e mais no aplicativo que estão usando o NGINX Plus para entregar.

Neste capítulo, você lerá uma visão geral da configuração do NGINX Controller, conectará uma instância do servidor NGINX Plus e aprenderá a usar a API do NGINX Controller.

O NGINX Controller é um produto corporativo que requer uma licença. Você pode solicitar uma avaliação gratuita na página do [produto do controlador NGINX](#).

16.1 Visão geral da configuração

Problema

Você gostaria de configurar um ambiente de controlador NGINX.

Solução

Use o [guias oficiais de instalação do controlador NGINX](#) para um processo de instalação atualizado. A seguir estão algumas dicas, observações e textos explicativos para itens a serem observados ao longo do guia de configuração.

O NGINX Controller 3.x é instalado como uma pilha do Kubernetes. É importante rever todas as [especificações técnicas](#) antes de começar. É necessário um banco de dados PostgreSQL externo. O instalador do Controller é fornecido como um tarball. Uma vez descompactado, um script install.sh precisará ser executado como um usuário não root.

Devido à forma como algumas imagens do SO são distribuídas, pode haver variação nos repositórios de pacotes, o que pode causar alguma dificuldade na instalação. O Ubuntu 18.04 parece ser o mais consistente em meus testes e é minha recomendação para testes e exploração. Lembre-se de que o NGINX Support está disponível para ajudá-lo a colocar o NGINX Controller em funcionamento rapidamente.

Há uma série de ferramentas que o instalador precisa instalar antes de poder funcionar corretamente. A maioria das ferramentas é padrão em muitos sistemas operacionais, porém a ferramenta jq não é. Você precisará garantir que todas as ferramentas necessárias estejam instaladas em seu sistema antes de executar qualquer um dos scripts do instalador.

Um script helper.sh é fornecido no pacote de instalação que pode ajudar na instalação ou alterar a configuração básica uma vez instalada. Por exemplo, o argumento suporte portpkg construirá um pacote de informações de depuração e log para você enviar ao Suporte NGINX para permitir que eles tenham uma visão geral rápida de sua situação. O argumento prereqs instalará os pacotes necessários e configurará o Kubernetes. Para visualizar os logs do NGINX Controller, você pode usar os logs ./helper.sh.

Quando o comando do instalador for executado, ele verificará os requisitos do sistema e instalará qualquer um que possa precisar adicionalmente. O instalador solicitará informações do banco de dados. Atualmente o PostgreSQL é suportado e deve estar em um servidor remoto. As informações do usuário fornecidas devem ser capazes de criar bancos de dados. Essas informações podem ser passadas como argumentos de linha de comando para o instalador.

Um volume de banco de dados de série temporal precisará ser fornecido. Você pode usar o disco local, um volume NSF ou um volume do AWS EBS. Se você optar por usar um volume do AWS EBS, o sistema precisará de permissões apropriadas do AWS IAM para anexar o volume à instância.

Um contrato de licença de usuário final é apresentado e deve ser aceito para avançar.

Após a leitura, pressione q para sair do acordo e, em seguida, y para aceitar.

Um servidor SMTP é necessário para convidar usuários por e-mail, bem como para notificações por e-mail. Caso um servidor SMTP ainda não esteja disponível, essas configurações podem ser definidas posteriormente usando o script helper.sh. Forneça algum valor genérico para esses prompts, defina o host para localhost, porta para 25, recuse autenticação e TLS. O NGINX Controller não poderá enviar e-mail até que o SMTP seja configurado.

O FQDN é usado ao gerar a configuração do agente e deve ser definido como um domínio confiável. O prompt do nome da organização é um nome amigável usado para rotulagem — um nome de equipe ou empresa será suficiente. Ao fornecer valores para o usuário administrador, observe que o email é usado para login no sistema.

Os caminhos de certificado SSL/TLS podem ser fornecidos por meio de parâmetros de comando do instalador ou como variáveis de ambiente. Se eles não forem encontrados, o instalador solicitará a geração de certificados autoassinados.

Assim que a instalação for concluída, o instalador fornecerá um link para o Controlador. Siga o link e faça login com as credenciais de administrador.

Discussão

O NGINX Controller fornece um plano de controle único para gerenciamento de seus aplicativos. A interface é seccionada em diferentes visualizações, Plataforma, Infraestrutura, Serviços e Análise. Ao fazer isso, a visualização é limpa e concisa para a tarefa específica em mãos.

A visualização da plataforma é usada para gerenciar o acesso do Controlador e do usuário. A visualização de infraestrutura fornece detalhes sobre as máquinas que executam os agentes do NGINX Controller. A próxima seção descreverá a adição de um servidor NGINX Plus ao Controller instalando um agente.

Na visualização de serviços, os atributos centrados no aplicativo do NGINX Controller vêm à tona. O Controller organiza seus aplicativos, ambientes, gateways e APIs para permitir que você reorganize e implante rapidamente.

Veja também

[Guia de instalação do administrador](#)

[Especificação técnica](#)

[Instalando e preparando o banco de dados PostgreSQL para o conhecimento do controlador NGINX](#)
[Artigo base](#)

16.2 Conectando o NGINX Plus com o controlador

Problema

Você instalou o Controller e precisa conectar uma instância do NGINX Plus com um agente.

Solução

Se você ainda não instalou o NGINX Plus, use a [Receita 1.3](#) para obter um nó NGINX Plus online.

A melhor maneira de encontrar a documentação para a instalação do seu controlador é visitar [https://\(Controller-FQDN\)/docs/infrastructure/agent](https://(Controller-FQDN)/docs/infrastructure/agent). Neste local do documento, você pode encontrar informações sobre as especificações técnicas necessárias para executar o NGINX Controller Agent, bem como instalar e gerenciar.

A instalação do Controller Agent em um servidor NGINX Plus existente é simples.

Você precisará recuperar um script do instalador da API do controlador na porta 8443 e

execute-o com uma chave de API. A IU do Controlador fornece instruções simples de copiar e colar para seu ambiente. Após a conclusão da instalação, você deve iniciar o Controller Agent usando o gerenciador de serviço para seu sistema.

Quando o serviço Controller Agent estiver em execução, você verá uma instância em execução na visualização Controller Infrastructure.

Discussão

Nesta seção, você adicionou um servidor NGINX Plus ao NGINX Controller como uma instância. Um inventário dos sistemas NGINX Plus agora é mostrado na visualização de infraestrutura, bem como com uma solicitação de lista para a API. Quando você tem uma ou mais instâncias em execução no NGINX Controller, você pode monitorar métricas valiosas do servidor e do NGINX Plus com a guia Graphs na visualização Infrastructure. Na visualização da plataforma, na guia agentes, há uma configuração para habilitar o analisador de configuração do NGINX. Quando ativada, a visualização Infraestrutura habilita uma guia de análise. A guia de análise fornece informações relevantes para a instalação do NGINX Plus e sua configuração atual.

Agora que você tem um novo nó NGINX Plus com o Controller Agent instalado, você pode querer obter uma imagem inicializável desta máquina ou criar um gerenciamento de configuração para dar suporte a essas instalações, para que você possa replicar a máquina. Com uma instância configurada, você pode começar a configurar serviços, que consistem em aplicativos, seus ambientes e como eles são servidos.

Veja também

[Guia de instalação do agente do controlador NGINX](#)

16.3 Conduzindo o controlador NGINX com a API

Problema

Você aprendeu a configurar entidades do NGINX Controller e deseja automatizar esses processos com a API.

Solução

Certifique-se de ter conectividade de rede com o Controlador na porta da API, que por padrão é 8443.

O NGINX Controller é 100% conduzido inteiramente por meio de sua API. A interface simplesmente usa essa API para fornecer acesso de apontar e clicar e painéis. Use a visão geral da API na documentação da instalação do seu controlador visitando <https://Controller FQDN/docs/api/overview/>. Isso ensinará a você a base dos objetos, permissões,

e como autenticar. A partir daí, a Referência da API pode ser encontrada em <https://{{Controller-FQDN}}/docs/api/api-reference/>.

Uma maneira de impulsionar sua automação usando a API é visualizar entidades já configuradas na interface do NGINX Controller, editar a entidade e visualizar as especificações da API. Esta especificação de API mostrará o método, caminho e carga necessária para criar esse objeto. Com alguma substituição de variável, você já está começando a automatizar seu ambiente Controller.

Discussão

Para alguns engenheiros, a API será a principal interação que eles terão com o NGINX Controller; para outros, será a interface web. Qualquer um é válido e extremamente poderoso. A adição de mostrar a chamada da API na interface da Web diminui a frustração de pesquisar a referência da API e acelera a automação de tarefas. Existe uma coleção Ansible para NGINX Controller para auxiliar na automação do Controller.

Veja também

[Introdução à coleção Ansible para NGINX Controller](#)

16.4 Habilitar WAF por meio da segurança do aplicativo do controlador

Problema

Você está usando o NGINX Controller ADC e gostaria de habilitar os recursos do Web Application Firewall (WAF) para seus aplicativos.

Solução

Se ainda não o fez, siga o [guias de instalação do NGINX Plus App Protect](#) para sua plataforma instalar o módulo App Protect em seu nó NGINX Plus.

Navegue até a configuração de um App Component existente no NGINX Controller.

Na seção Segurança, localize o cabeçalho ou as configurações do WAF. Ative o WAF e Salve ý.

O WAF agora está processando solicitações para o aplicativo por meio da política padrão do WAF. A política padrão é definida para alarme em todas as assinaturas, mas bloqueará as assinaturas consideradas altamente precisas. A precisão é determinada por um algoritmo que determina a probabilidade de falsos positivos. Isso significa que você pode começar a bloquear solicitações prejudiciais imediatamente enquanto coleta dados sobre eventos de segurança relatados pela política. As solicitações sinalizadas e bloqueadas serão exibidas na interface do usuário do NGINX Controller, devidamente rotuladas. O NGINX Controller ADC exibirá estatísticas de WAF e eventos de violação para violações de WAF acionadas.

Certifique-se de que o aplicativo esteja lidando com algum tráfego. Teste uma solicitação que normalmente seria bloqueada ou sinalizada por um WAF. A seguir está uma solicitação de injeção de SQL extremamente básica:

```
curl https://{{appComponentEndpoint}}/?query=99999999%20UNION%20SELECT%201%2C2
```

Após uma solicitação que seria considerada um evento de segurança, o NGINX Controller ADC relatará os dados de análise de segurança. Localize essas métricas na seção Security Analytics para este aplicativo e componente de aplicativo. Na [Figura 16-1](#) você pode ver como o NGINX Controller exibe informações métricas sobre solicitações sinalizadas pelo WAF.

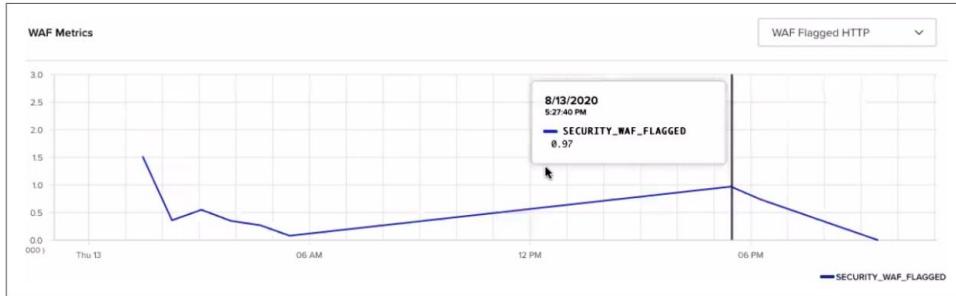


Figura 16-1. Um NGINX App Security WAF sinalizou algumas solicitações

Assim que algumas solicitações sinalizadas começarem a aparecer, você também poderá visualizar o evento na página Eventos de segurança. É aqui que você encontrará informações detalhadas sobre cada solicitação sinalizada ou bloqueada pelo NGINX App Security WAF.

Antes de habilitar políticas mais rígidas, você deve verificar se o tráfego de aplicativo válido normal não está sendo sinalizado. Se o comportamento normal for sinalizado, inspecione o motivo do evento de segurança individual. Se o tráfego normal do aplicativo estiver sendo sinalizado, você pode ter vulnerabilidades em seu aplicativo que precisam ser corrigidas. Na [Figura 16-2](#), todo o tráfego observado pelo WAF é exibido para mostrar um padrão entre o tráfego regular e a solicitação violada pelas regras do WAF.

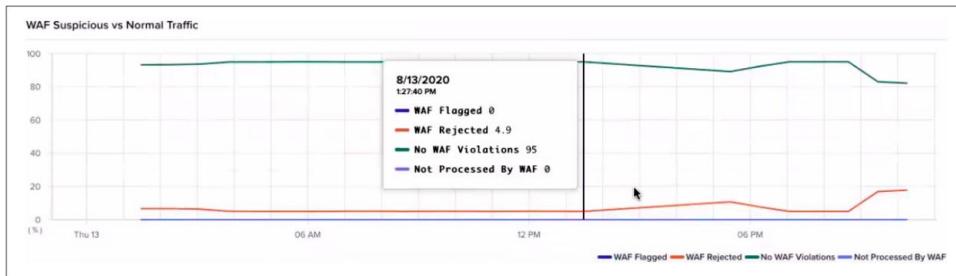


Figura 16-2. Um exemplo de estatísticas relatadas pelo Controller ADC com um WAF habilitado

Discussão

O Controller App Security fornece uma experiência WAF simplista para proteger seus aplicativos. Com as informações fornecidas pelo monitoramento, você pode observar tendências ao longo do tempo para ataques de segurança, investigar mais detalhadamente o evento e decidir qual ação deve ser tomada.

Os firewalls de aplicativos da Web são extremamente importantes na arquitetura de segurança da Web atual. Os aplicativos são constantemente bombardeados com tentativas de comprometer serviços com vulnerabilidades comuns. Ao bloquear essas solicitações antes que elas cheguem aos seus serviços de aplicativo, você não apenas protege o aplicativo da Web, mas também reserva recursos para solicitações de clientes legítimos.

Um WAF não é apenas para clientes externos; você também deve considerar o uso de um WAF no tráfego interno, para que um serviço comprometido não afete outro.

CAPÍTULO 17

Dicas e Conclusão de Operações Práticas

17.0 Introdução

Este último capítulo cobrirá dicas práticas de operações e é a conclusão deste livro. Ao longo deste livro, discutimos muitos conceitos pertinentes aos engenheiros de operações. No entanto, pensei que mais algumas idéias poderiam ser úteis para completar as coisas.

Neste capítulo, abordarei como garantir que seus arquivos de configuração sejam limpos e concisos, bem como depurar arquivos de configuração.

17.1 Usando inclusões para configurações limpas

Problema

Você precisa limpar arquivos de configuração volumosos para manter suas configurações agrupadas logicamente em conjuntos de configuração modulares.

Solução

Use a diretiva include para fazer referência a arquivos de configuração, diretórios ou máscaras:

```
http
{ include config.d/compression.conf;
  include habilitado para sites/*.conf
}
```

A diretiva include usa um único parâmetro de um caminho para um arquivo ou uma máscara que corresponde a muitos arquivos. Esta diretiva é válida em qualquer contexto.

Discussão

Ao usar instruções de inclusão , você pode manter sua configuração do NGINX limpa e concisa. Você poderá agrupar logicamente suas configurações para evitar arquivos de configuração que duram centenas de linhas. Você pode criar arquivos de configuração modulares que podem ser incluídos em vários locais ao longo de sua configuração para evitar a duplicação de configurações. Veja o exemplo de arquivo de configuração fastcgi_param fornecido na maioria das instalações de gerenciamento de pacotes do NGINX. Se você gerencia vários servidores virtuais FastCGI em uma única caixa NGINX, pode incluir esse arquivo de configuração para qualquer local ou contexto em que você precise desses parâmetros para FastCGI sem ter que duplicar essa configuração. Outro exemplo são as configurações de SSL. Se você estiver executando vários servidores que exigem configurações SSL semelhantes, você pode simplesmente escrever essa configuração uma vez e incluí-la sempre que necessário. Ao agrupar logicamente suas configurações, você pode ter certeza de que suas configurações são organizadas. A alteração de um conjunto de arquivos de configuração pode ser feita editando um único arquivo, em vez de alterar vários conjuntos de blocos de configuração em vários locais em um arquivo de configuração massivo. Agrupar suas configurações em arquivos e usar instruções de inclusão é uma boa prática para sua sanidade e a sanidade de seus colegas.

17.2 Configurações de depuração

Problema

Você está obtendo resultados inesperados do seu servidor NGINX.

Solução

Depure sua configuração e lembre-se destas dicas:

- O NGINX processa solicitações procurando a regra correspondente mais específica. Isso torna um pouco mais difícil percorrer as configurações manualmente, mas é a maneira mais eficiente de o NGINX funcionar. Há mais informações sobre como o NGINX processa solicitações no link de documentação na seção “[Consulte também](#)” na página 171. • Você pode ativar o registro de depuração. Para o registro de depuração, você precisará garantir que seu pacote NGINX esteja configurado com o sinalizador `--with-debug` . A maioria dos pacotes comuns tem; mas se você construiu seu próprio pacote ou está executando um pacote mínimo, você pode querer pelo menos verificar novamente. Depois de garantir que você tenha depuração, você pode definir o nível de log da diretiva `error_log` para depurar: `error_log /var/log/nginx/error.log debug`.
- Você pode habilitar a depuração para conexões específicas. A diretiva `debug_connection` é válida dentro do contexto de eventos e usa um intervalo de IP ou CIDR como parâmetro. A diretiva pode ser declarada mais de uma vez para adicionar vários IPs

endereços ou intervalos CIDR a serem depurados. Isso pode ser útil para depurar um problema na produção sem prejudicar o desempenho depurando todas as conexões. • Você pode depurar apenas servidores virtuais específicos. Como a diretiva error_log é válida nos principais contextos HTTP, mail, stream, servidor e local, você pode definir o nível de log de depuração apenas nos contextos que precisar.

- Você pode habilitar dumps principais e obter backtraces deles. Os dumps principais podem ser habilitados por meio do sistema operacional ou do arquivo de configuração NGINX. Você pode ler mais sobre isso no guia de administração na seção “[Ver também](#)” na página 171.
- Você pode registrar o que está acontecendo nas instruções de reescrita com o rewrite_log diretiva em: rewrite_log on.

Discussão

A plataforma NGINX é vasta e a configuração permite que você faça muitas coisas incríveis. No entanto, com o poder de fazer coisas incríveis, há também o poder de atirar no próprio pé. Ao depurar, certifique-se de saber como rastrear sua solicitação por meio de sua configuração; e se você tiver problemas, adicione o nível de log de depuração para ajudar. O log de depuração é bastante detalhado, mas muito útil para descobrir o que o NGINX está fazendo com sua solicitação e onde em sua configuração você errou.

Veja também

[Como o NGINX processa solicitações](#)

[Guia de administração de depuração](#)

[Registro de reescrita](#)

Conclusão

Este livro se concentrou em balanceamento de carga de alto desempenho, segurança e implantação e manutenção de servidores NGINX e NGINX Plus. O livro demonstrou alguns dos recursos mais poderosos da plataforma de entrega de aplicativos NGINX. A NGINX Inc. continua a desenvolver recursos incríveis e ficar à frente da curva.

Este livro demonstrou muitas receitas curtas que permitem que você entenda melhor algumas das diretivas e módulos que fazem do NGINX o coração da web moderna.

O servidor NGINX não é apenas um servidor web, nem apenas um proxy reverso, mas toda uma plataforma de entrega de aplicativos, totalmente capaz de autenticação e ganhando vida com os ambientes em que é empregado.

Índice

UMA

- Testes A/B, [25](#) logs
- de acesso (veja como restringir o acesso)
- de buffer da diretiva access_log, [159](#)
 - logs de encaminhamento para o ouvinte Syslog, [150](#)
- monitoramento de atividade (consulte monitoramento)
- diretiva add_header cache do lado do cliente, [41](#)
 - compartilhamento de recursos entre origens, [73](#)
- Adobe Adaptive Streaming, [99](#) ferramentas
- de pacote avançado (APT) instalação de NGINX, [2](#) logs
- de agregação, [151](#)
 - Pilha ELK, diretiva
- de permissão [151](#), [72](#)
- Ganchos do ciclo de vida do Amazon Auto Scaling, [104](#)
- Amazon Machine Image (AMI) automatizando a construção de, [102](#) automação de configuração, [101](#)
- Amazon Web Services (AWS)
 - DNS do Amazon Route 53
 - Automação de DNS, verificação de integridade [104](#), balanceamento de carga [103](#), [132](#), [133](#), [103](#)
 - AWS NLB, [133](#)
 - automação de configuração, [101](#)
 - implantação do NGINX Plus do Marketplace, [106](#)
 - EC2
 - balanceamento de carga, [132](#) balanceador de carga de rede, [104](#)
 - Dados do usuário, [101](#), [102](#)
- Ansible
- sobre, [57](#)
- documentação, [57](#)
- Recurso do controlador NGINX, [165](#)
- Instalação e configuração do NGINX, [56](#)
- Senhas do Apache (veja htpasswd)
- Gateways de API
 - sobre, [119](#)
 - NGINX as, [116](#)-[120](#)
 - documentação, [116](#), [120](#)
- Bloqueio ativo do App Protect
 - Module, [88](#)
 - Arquivo de política do App Protect, [87](#) documentação, [86](#), [89](#), [165](#)
 - instalação, [86](#) guia de instalação, [86](#), [165](#)
 - Web Application Firewall, [89](#) servidores
- de aplicativos, adicionando/removendo upstream, [49](#)
 - aplicativos de alta disponibilidade, [131](#) (veja também alta disponibilidade)
- NGINX conteinerizado, [115](#) (veja também contêineres) até a carga total de produção, [23](#)
- APT (ferramenta de pacote avançado) instalação de NGINX, [2](#)
- autenticação
 - sobre autenticação NGINX, [61](#) subsolicitações de autenticação, [63](#)
 - HTTP básico
 - sobre, [62](#)
 - HTTP básico (htpasswd), [61](#)
 - Conjunto de chaves da Web JSON, [67](#)
 - Chaves da Web JSON

- sobre, 66
 - criando, 65
 - documentação, 66
 - atualizando dinamicamente, 68
 - Caminho do arquivo de chave de autenticação JWT, 64
 - Padrão RFC, 66
 - Autenticação JSON Web Token, 61, 64, 66
 - documentação, 65, 67, 68 tipos de tokens de assinatura da web, 64
 - Documentação do OpenID Connect, 69
 - OpenID Connect JWT, 61, 66, 68
 - OpenID Connect SSO, 68
 - repositórios, 68 funções de criptografia de senha, 61
 - HTTP básico (htpasswd), 61
 - diretiva satisfazer, 84 autenticação de terceiros, 63 descartando o corpo da solicitação, 63 diretiva auth_basic, 62 (consulte também htpasswd) diretiva auth_jwt, 64 cookie como token a ser validado, 66
 - diretiva auth_jwt_key_request, 67
 - sub solicitações de autenticação da diretiva auth_request, 63 diretiva auth_request_set para cabeçalhos, 63
 - Escala horizontal do grupo de Auto Scaling, 9 balanceamento de carga no EC2, 133 criação de平衡ador de carga de rede, 104 Sanduíche NLB, automação 104 (ver programabilidade)
 - Azure
 - implantação do NGINX Plus do Marketplace, 109
 - balanceamento de carga em conjuntos de dimensionamento, 109 imagens de máquina virtual ativadas, 107 conjuntos de dimensionamento de máquinas virtuais, 109
 - B**
 - balanceamento de carga (veja balanceamento de carga) limitação de largura de banda, 34 streaming de mídia, limitação de taxa de bits de 100, lista de bloqueio de 100, dinâmica, mitigação de DDoS dinâmica de 49, implantação de 85 azul-verde, ajuste de 26 gargalos, 155
 - logs de buffer, 159
 - respostas de buffer, 158
- C**
- Criptografia da função de criptografia da linguagem de programação C, 61 extensões personalizadas, 51, 53 de cache, 37 de desvio, 40 de bloqueio de cache, 38
 - Módulo Cache Slice, 43
 - cache do lado do cliente, 41 diretiva de expiração, 41 público versus privado, 41
 - cache de chave de hash, 39 caminho para o cache, 37 purga de objeto do cache, 41 arquivo de segmentação para eficiência, 42 versão canary, 26
 - CentOS
 - instalação do NGINX, 2
 - Sincronização de configuração do NGINX Plus, 133
 - Instalação dinâmica Perl, 126
 - certificados
 - pares de chave de certificado, 75
 - ECC versus RSA, 76
 - certificados SSL
 - Google App Engine, 113
 - Módulo SSL, 74
 - Chef
 - sobre, 56
 - documentação, 56
 - Instalação e configuração do NGINX, 55
 - Livros de receitas de supermercados, 56
 - Depuração de conexão CIDR (roteamento entre domínios sem classe), 170 lista de bloqueio dinâmico, 51
 - Proxies GeoIP, 30
 - restringindo o acesso, 72
 - roteamento entre domínios sem classes (consulte CIDR) cache do lado do cliente, diretiva 41 expira, 41 público versus privado, 41
 - criptografia do lado do cliente, 74 avançadas, 75
 - implantações de nuvem sobre provedores de nuvem, 101

Balanceador de carga DNS do Amazon Route 53, 103 Amazon Web Services automação de configuração, 101 implantação do NGINX Plus do Marketplace, 106 provisionamento, 102 Azure implantação do NGINX Plus do Marketplace, 109 balanceamento de carga sobre conjuntos de dimensionamento, 109 imagem de máquina virtual ativada, 107 Proxy do Google App Engine, 112 Imagens do Google Cloud, 111 Google Compute Engine implantando NGINX para, 110 Proxy do Google App Engine, criação de 112平衡adores de carga de rede, 104 com reconhecimento de cluster Consul para automação de configuração, 59 mitigação dinâmica de DDoS, 85 armazenamento de chave-valor, 50 sincronização de zona, 136 sobre, 136 visão geral de comandos para NGINX, 5 configuração sobre gerenciamento, 45 formato de log de acesso, 147 geoproxy, 148 automação Cônsl, 59 Packer, 102 número de conexão maximizado, 160 de depuração, 170 log de erros para erros de arquivo de configuração, 150 recarregamento normal, 7 diretivas de inclusão, 169 ferramentas de gerenciamento (consulte ferramentas de gerenciamento de configuração) Arquivos e diretórios de chave NGINX, 4 Armazenamento do Google para, 113 OpenTracing Jaeger, 152 Zipkin, 152 aplicativos de aceleração, 23 servindo conteúdo estático, 6 ferramentas de gerenciamento de configuração aproximadamente, 45, 102 Servidores NGINX da Amazon Web Services, 101 Ansible, 56	automação Cônsl, 59 Empacotador, 102 Chef, armazenamento de 55 valores-chave (consulte armazenamento de valores-chave) Sincronização de configuração NGINX Plus HA, 133 aproximadamente, 135 Marionete, 54 SaltStack, 58 servidores adicionados ou removidos em tempo real, 46 depuração de conexão, 170 drenagem de conexão, 19 remoção de um servidor, 47 conexões, número máximo, 160 (consulte também limitando conexões) Consul para automação de configuração, 59 sobre Consul, 60 documentação, 60 containers aproximadamente, 115 Gateway de API sobre conteinerização, 115 NGINX como, 116-120 criando Dockerfile para criar imagem do Docker NGINX, 122 NGINX Plus, 124-125 Registro SRV DNS para balanceamento de carga, 120 variáveis de ambiente, 126 Controlador de entrada do Kubernetes, 127-129 Conteinerização NGINX, 115 Imagem NGINX do Docker Hub, 121 Criação de imagem NGINX Plus Docker, 124-125 Módulo exportador Prometheus, 129 cache de conteúdo (veja cache) redes de entrega de conteúdo (CDNs), 37 Cookies do controlador (consulte NGINX Controller) Validação JSON Web Token, 66, 69 sticky cookies com NGINX Plus, 16 sticky load com cookie existente, 17 compartilhamento de recursos de origem cruzada (CORS), 72 criptografia de função crypt, 61 curl adicionando um servidor, 46 testes de autorização, 62 métricas de painel via API, 143 conexões de servidor de drenagem, 47
---	--

atualização de armazenamento de valor-chave via PATCH, 50 servidores de listagem, 47 limpando arquivos do cache, 42 removendo um servidor, 48 testes de solicitação, 4

extensões personalizadas módulos de idioma disponíveis, 51, 53 Lua Olá Mundo, 52 njs Olá Mundo, 51 Ambiente de leitura Perl, 53

Datagramas D (ver UDP)

DDoS (consulte Negação de Serviço Distribuída) Debian sincronização de configuração, 134 instalação do NGINX, 1 Imagem do NGINX Docker, 122 configurações de depuração, 170 conexões, 170 registros de depuração, 170

Guia de administração de depuração, 171 log de erros para, 150 (consulte também logging) rastreamento de solicitação, 151 OpenTracing, 152 diretivas debug_connection, 170 parâmetros default_server, 6 diretivas deny, 72 deployment azul-verde, 26 canary release, 26 cloud (ver cloud deployments) containers como unidades executáveis de código, 115

Controlador de entrada do Kubernetes, 128 diretórios e arquivos Arquivo de política do App Protect, 87 arquivos de usuário de autenticação básica, 62 locais de cache, 37 Instalação dinâmica do CentOS Perl, saída do driver de log do Docker de 126 contêineres, caminho do log de erro 115, 149 Arquivo de chave JSON Web Tokens, 64, armazenamento de 66 valores-chave, 50 painéis de monitoramento para NGINX Plus, 141 Imagem NGINX do Docker Hub, 122 Arquivos de chave NGINX, 4 Armazenamento do Google para, 113

streaming de mídia, 98 Mitigação de negação de serviço distribuída (DDoS), 85

DNS Amazon Route 53, 103 automatizando a criação e remoção de registros DNS, 104 Verificação de integridade do DNS Dyn, 132 Proxy do Google App Engine, 112 balanceamento de carga entre平衡eadores de carga, 132 registros SRV, 120

Janela de encaixe criando Dockerfile para criar imagem do Docker NGINX, 122 NGINX Plus, 124-125 variáveis de ambiente, 127 imagens do controlador de entrada, 127 saídas do driver de log para stdout e stderr, 115 Imagem NGINX do Docker Hub, 121 Instalação dinâmica Perl, 126 Formato de senha Dovecot, parâmetro de drenagem 62, 19 Verificação de integridade do DNS Dyn, 132 listas de bloqueio dinâmicas, 49 mitigação dinâmica de DDoS, 85

E Automação de configuração do EC2 (Amazon Web Services) via UserData, balanceamento de carga 101, 102, balanceador de carga de rede 132, 104

Certificados ECC versus RSA, 76 Pilha ELK (Elasticsearch, Logstash e Kibana), 151

Embedded Ruby (ERB) linguagem de modelagem de Puppet, 54 variáveis incorporadas em logs, 147 criptografia do lado do cliente, 74 avançadas, 75 gRPC, 93

HTTP/2 habilitado, função de criptografia de 92 senhas, 61 HTTP básico, 61 tráfego upstream, 77 diretiva env, 126 variáveis de ambiente

- Amazon EC2 para gerenciamento de configuração, 102
 imagens de contêiner em diferentes ambientes, 126
 Docker e, 127
 Script Perl configurando a variável NGINX de, 53, 126
 apagado do ambiente por padrão, 126 portas efêmeras, 160 log de erro sobre log de erro, 150 configuração, 149 logs de encaminhamento para o ouvinte Syslog, 150 sobre Syslog, 151 log de depuração da diretiva error_log, 170 logs de encaminhamento para o ouvinte Syslog, 150 caminho de log definido, 149 data de expiração para proteger um local, 79 expira diretiva para armazenamento em cache do lado do cliente, 41 link expirando, 80
- Módulo e banco de dados GeoIP, 27
 proxies, 30 cabeçalhos encaminhados padronizados, 30 geoproxy, 149
- Proxy do Google App Engine, 112
 Imagem do Google Cloud, 112
 Google Compute Engine
 implantando NGINX para, 110
 Proxy do Google App Engine, 112
 Google Compute Image, 111 Google
 OAuth 2.0 OpenID Connect (consulte OpenID Connect)
- Armazenamento do Google para arquivos de configuração, cerca de 113 chamadas de método gRPC, 94 roteamentos de back-end via diretiva de localização, 93
 balanceamento de carga, 94 conexões proxy gRPC, 92
- F**
- Módulo F4F, 99
 failover
 exame de saúde
 DNS do Amazon Route 53, 103, 132, 133
 Dyn DNS, 132
 keepalives do modo NGINX Plus HA, 131 arquivos e diretórios
 Arquivo de política do App Protect, 87 arquivos de usuário de autenticação básica, 62 locais de cache, 37
 Instalação dinâmica do CentOS Perl, saída do driver de log do Docker de 126 contêineres, caminho do log de erro 115 , 149
 Arquivo de chave JSON Web Tokens, 64, armazenamento de 66 valores-chave, 50
 painéis de monitoramento para NGINX Plus, 141
 Imagem NGINX do Docker Hub, 122
 Arquivos de chave NGINX, 4
 Google Storage for, 113
 streaming media, 98 firewall quick
 start guide, 71
 Mídia de streaming FLV (Flash Video), 97 cabeçalhos encaminhados padronizados para proxies, 30
- Cache de chave de hash H , 39 criando chaves de hash, 39
 Mídia de streaming HDS (HTTP Dynamic Streaming), 99 cabeçalhos
 encaminhados padronizados para proxies, 30
 Segurança de Transporte Estrita HTTP, 83
 X-Forwarded-Proto, 83 health
 check active check com NGINX Plus, 21
 DNS do Amazon Route 53, 103, 132, 133
 Dyn DNS, 132
 pulsações de keepalived, 131
 verificação passiva, 20 pulsações de keepalived, 131
 Olá Mundo
 JavaScript, 51
 Lua, 52
 modos de implantação de alta disponibilidade (HA) sobre alta disponibilidade, 131 balanceamento de carga
 EC2, 132
 平衡adores de carga via DNS, 132
 Modo NGINX Plus HA, 131
 IPs da Amazon e 132, 133 pacotes keepalived, 131, 133 configurações de sincronização, 133 sincronização de zona, 136
- G**
- balanceamento de carga de hash genérico, 16

- Mídia de streaming HLS (HTTP Live Streaming), 98
- escala horizontal, 9
- htpasswd para senhas, 62
- HTTP
- redirecionando para HTTPS, 82
 - SSL/TLS encerrado antes do NGINX, 82
- Segurança Estrita de Transporte, 83
- Servidores HTTP
- recursos de cache, 37
 - arquivos e diretórios de configuração, 4 servindo conteúdo estático, 6 testes de instalação, 4 balanceamento de carga, 10 verificação de integridade ativa, 21
 - Hash de IP, 16 verificações de integridade passiva, 20 monitoramento básico habilitado, 139
 - Métricas de retorno da API NGINX Plus, 144
- Modelo OSI, 12
- HTTP Strict Transport Security (HSTS), 83 documentação, 84
- HTTP/2
- sobre, 91
 - configuração básica, 91
 - método gRPC chama roteamento de back-end via diretiva de localização, 93
 - proxies de conexão, 92
 - servidores push, 94 testes via plug-in do navegador, 92
- Autenticação básica
- HTTPS upstream, 63 entrega de aplicativos de domínio personalizado, 113 proxy, 77 redirecionando solicitações não criptografadas para, 82
 - SSL/TLS encerrado antes do NGINX, 82 criptografia upstream, 77 http_auth_request_module, 63
- incluir diretivas em arquivos de configuração, 169, 170
- infrastructure as a service (IaaS), 101 controlador de entrada para Kubernetes, 127-129 aproximadamente, 129 contêineres de instalação como unidades de código executáveis, 115
- NGINX**
- Ansible, 56
 - CentOS, 2
 - Cozinhiero, 55
 - Debian, 1
 - imagem do Docker Hub, 121
 - Marionete, 54
 - Red Hat, 2
 - Pilha de Sal, 58
 - Ubuntu, 1
 - validação, 3
- Controlador NGINX, 161
- NGINX Plus, 3
- Módulo App Protect, 86
 - sincronização de configuração, 133 implantação do AWS Marketplace, 106 implantação do Azure Marketplace, 109 validação, 3 comando ip addr, 134
- endereço de IP
- acesso baseado em, 71
 - restringindo por país, 29 depuração de conexão, 170
 - DNS para balancear a carga de平衡adores de carga, 132 lista de bloqueio dinâmico, 49 encontrando cliente original, 30 keepalive do modo NGINX Plus HA, 131, 133
 - limitando conexões com base em, 31
 - Tradução de Endereço de Rede e, 32 taxa limite de solicitações, 32
 - NGINX Plus HA e IPs da Amazon, 132, 133
- Balanceamento de carga de hash IP, 16
- J**
- Configuração do plug-in Jaeger para OpenTrace, 152
- Compartilhamento de recursos de origem cruzada JavaScript, 73 introdução, 51
- Linguagem de modelagem Jinja2 do SaltStack, 59
- ferramenta jq, 162

Conjunto de chaves da Web JSON (JWKS), 67

JSON Web Keys (JWKs) sobre, 66

- criando, 65 documentação,
- 66 atualizando dinamicamente,
- 68

Caminho do arquivo de chave de autenticação JWT, 64

Padrão RFC, 66

Autenticação JSON Web Tokens (JWTs) via módulo NGINX Plus, 61, 64, 66, 68

- tipos de tokens de assinatura da web, 64, 67
- documentação, 65, 67

Provedor de identidade OpenID Connect, 68 diretiva js_content, 52

K

diretiva keepalive, 157 keepalived

- Endereços IP EC2, 133
- Modo NGINX Plus HA, diretiva 131

keepalive_requests, diretiva keepalive_timeout 156, configuração do kernel 156 para conexões máximas, armazenamento de 160 valores-chave aproximadamente, 45, 50 reconhecimento de cluster, 50 lista de bloqueio dinâmico, 49 atualização ou exclusão de chave, 50 mitigação dinâmica de DDoS, 86

Validação do JSON Web Token, 69 diretório keyval_zone, 50

Kubernetes

- aproximadamente,
- 129 controlador de entrada, 127-129
- aproximadamente, 129

Implantação versus DaemonSet, 128 repositório, 127

Controle de Acesso Baseado em Função, 128

NGINX Controller instalando como pilha, 161

Prevalência de Prometheus, 130

Módulos de linguagem L

- disponíveis, 51, 53
- JavaScript Olá mundo, 51
- Lua Olá mundo, 52

Ambiente de leitura Perl, menos 53

conexões de平衡amento de carga, 15

menos tempo de balanceamento de carga, 15

Lightweight Directory Access Protocol (LDAP), 62

limitando a largura de banda, 34

conexões limitando por chave predefinida, 31

- Endereço IP, 32
- testes complicados, 32

limitando a taxa de solicitações

- com base na chave predefinida, 32
- testes complicados, 34 argumentos de palavra-chave de explosão, 33
- aspectos de segurança, 33 diretivas de escuta número de conexão maximizado, 160
- proxies de conexão gRPC, 92 verificações de integridade de servidores, 22

HTTP/2 habilitado, 92 portas

criptografadas com SSL/TLS, 74, 75 portas para escutar, 6 redirecionam para HTTPS, 82, 83

porta TCP, 11

- Balanceamento de carga TCP, 12
- Balanceamento de carga UDP, 13, 14

balanceamento de carga aproximadamente, 9, 131

Balanceador de carga do Azure, implantação de 109 nuvens

- DNS do Amazon Route 53, criação do balanceador de carga de rede 103 , 104
- ambiente conteinerizado, 115
- EC2, 132
- portas efêmeras, 160 hash
- genéricos, 16 chamadas
- gRPC, 94 verificações de integridade de verificação ativa com NGINX Plus, 21
- Amazon Route 53 DNS, 103 verificação passiva, 20 alta disponibilidade

aproximadamente, 131

EC2, 132

balanceadores de carga via DNS, 132

servidores HTTP, 10

Hash de IP, 16

menos conexões, 15 menos tempo, 15 balanceadores de carga balanceados, 132 métodos para, 14

NGINX Plus

verificação de integridade
ativa, 21 ambientes em contêineres, 115
Registros DNS SRV, 120 de
menor tempo, 15 aleatórios,
16 round-robin, 15 estados de
sessão e 9

servidores TCP, 11
Servidores UDP, 13
falhas de servidor upstream, 10
teste de carga automatizado,
diretiva load_module 155
Instalação do módulo App Protect, 86
GeoIP, 28
OpenTracing, 153
blocos de localização
restringindo o acesso ao endereço IP, 71
datas de expiração de segurança, 79
métodos múltiplos, 84 secretos para, 77
servindo conteúdo estático, 7 streaming
media, 98, 98

Mídia HLS, 99
diretivas de localização, 93
logs, 147 configuração de
formato de log de
acesso, 147 geoproxy, 148 logs de agregação,
151

pilha ELK, 151
Log de segurança do App Protect, 87 logs
de buffer, 159 logs de depuração, 170
locais padrão para arquivos de log, 5

Saída do driver de log do Docker, 115
variáveis incorporadas, 147 log de
erro sobre log de erro, 150
configuração, 149 encaminhamento
para o ouvinte Syslog, 150 sobre
Syslog, 151 log do kernel e número de
conexões, 160 rastreamento de
solicitação, 151

OpenTracing, 152
identificadores de solicitação,
152 log de reescrita, 171 logs de
visualização no NGINX Controller, 162 diretiva
log_format, 148

Dois
introdução , 52 documentação,
53 módulos de linguagem
disponíveis, 51 documentação do
módulo, 53 objetos ngx para NGINX
API, 52

M
ataque man-in-the-middle, 83
autenticação de diretiva de mapa, 118
limpeza de cache, 41 códigos de
país, 29 compartilhamento de
recursos entre origens, 72

Agrupamento GET e POST, 73
controle de sessão persistente, 18
Módulo de firewall ModSecurity, 71
monitoramento sobre, 139

Registro de segurança do App Protect, 87
Firewall de aplicativo da Web, 167
monitoramento básico habilitado, 139
logs do kernel e número de conexões, 160 testes de
carga automatizados, 155
Painel de monitoramento NGINX Plus sobre,
139-141
Acesso à API para métricas detalhadas, 143
demonstração online, 143 habilitação, 140
rastreamento de solicitação, 151

OpenTracing, 152
identificadores de solicitação, 152
Mídia de streaming MP4 (MPEG-4), 97 limites
de largura de banda, 100

N
Network Address Translation (NAT), balanceador
de carga de rede 32 (NLB)
AWS NLB, 133
criando, 104
NGINX por trás do AWS NLB, 133
servidores de protocolo de tempo de rede (NTP), 13
NGINScript (veja njs)
NGINX
sobre,
visão geral do comando xi ,
5 configuração
Cônsul para automação, 59
depuração, 170

arquivos e diretórios, 4
 recarregamento normal, 7
 diretivas de inclusão, 169
 servindo conteúdo estático, 6
 conteinerizando com facilidade, 115
 (veja também containers)
 instalação
 Ansible, 56
 CentOS, 2
 Cozinheiro, 55
 Debian, 1
 imagem do Docker Hub, 121
 Marionete, 54
 Red Hat, 2
 Pilha de Sal, 58
 Ubuntu, 1
 validação, 3
 arquivos e diretórios de chave, 4
 processos mestre como root, 4
 Sanduíche NLB,
 processamento de 104 solicitações, 170, 171
 Controlador NGINX
 sobre, 141, 145, 161, 163 página
 do produto, 143
 ADC para habilitar o Web Application Firewall, 89, 165
 agente conectando NGINX Plus, 163
 API automatizando processos, 164
 documentação
 Referências de API, 164
 guia de instalação, 161, 163
 Agente do Controlador NGINX, 163, 164
 Banco de dados PostgreSQL,
 163 especificações técnicas, 161,
 163 instalação, 161
 Pilha do Kubernetes, 161
 Banco de dados PostgreSQL necessário, 161, 162
 documentação, 163
 configuração, 161
 Servidor SMTP necessário, 162
 NGINX Plus
 API
 coleta de métricas, 143
 RESTful, 47
 controlador de entrega de aplicativos, xiii
 cookie de autenticação auth_token, 66
 Criação de chave da Web JSON, 65
 Conjunto de chaves da Web JSON, 67
 Validação de token da Web JSON, 61, 64, 66
 Documentação de autenticação JWT, 65
 OpenID Connect JWT, 61, 66
 OpenID Connect SSO, 68 tipos
 de tokens de assinatura na web, 64, 67
 implantação na nuvem
 AWS Marketplace, 106
 Azure Marketplace, 109
 sincronização de configuração, 133 sobre,
 135 conteinerização com facilidade, 115
 (veja também contêineres)
 Modo de alta disponibilidade do controlador
 (consulte Controlador NGINX), 131
 IPs da Amazon e, 132, 133 pacote
 keepalived, 131, 133 configurações
 de sincronização, 133 sincronização de
 zona, 136 instalação, 3
 App Protect Module, 86
 sincronização de configuração, 133 de
 implantação do AWS Marketplace, 106 de
 implantação do Azure Marketplace, 109 de validação,
 3 armazenamento de chave-valor, 50
 Kubernetes Ingress Controller, 127 verificações
 de integridade ativa de balanceamento de
 carga, 21 ambientes em contêiner, 115
 Registros DNS SRV, 120
 menos tempo, 15 painéis de
 monitoramento sobre, 139-141
 Acesso de API a métricas detalhadas, 143
 demo online, 143 habilitação, 140
 processamento de solicitação, 170, 171
 servidores adicionados ou removidos em
 tempo real, 46 aplicativos com estado em escala
 drenando conexões normalmente, 19, 47
 balanceamento de carga e 9 controle de sessão
 persistente, 18 sticky cookies, 16 sticky learn com
 cookie existente, 17 streaming media largura de
 banda limitando dinamicamente, 100 streaming
 dinâmico, 97, 99 fragmentação em tempo real, 97,
 98 ngx para Lua NGINX API, 52

- documentação njs, 53
 - introdução , 51 documentação, 53
 - Ativador de JavaScript, 51
 - js_content, 52
 - NGINScript anteriormente, 51
 - OpenID Connect para autenticação, 68
 - sanduíche NLB, 104
- O**
 - descritores de arquivo aberto, 160
 - Autenticação OpenID Connect
 - (OIDC) via provedor de identidade, 68 repositório, 68
 - documentação, 69
 - Autenticação JSON Web Token, 61, 66, 68 openssl command md5 hex digest, 79 passwd, 61
- OpenTracing, 152
 - instalação, 153
 - configuração de plug-in
 - Exemplo Jaeger, 152
 - Exemplo de Zipkin, 152
 - diretiva opentracing_tag, 153, 154 sistema operacional para conexões descritores de arquivo aberto, 160 ajuste do sistema operacional para conexões, 159 portas efêmeras, 160
- Modelo OSI, http versus stream, 12
- Sistema de gerenciamento de pacotes P
 - APT para instalação, 2
 - Instalação do NGINX
 - Cozinheiro, 56
 - Marionete, 54
 - Pilha de Sal, 58
 - Packer (HashiCorp), função de criptografia de 102 senhas, 61
 - HTTP básico (htpasswd), arquivo de usuário de autenticação básica de 61 caminhos, 62 locais de cache, 37 saídas do driver de log do Docker de contêiner, 115 logs de erros, 149
 - Arquivo de chave JSON Web Tokens, 64
- painel de monitoramento para NGINX Plus, 141
- Imagem NGINX do Docker Hub, 122 modelo de pagamento por uso, 101
 - Implantação do AWS Marketplace NGINX Plus, 106
 - Implantação do Azure Marketplace NGINX Plus, 109
- ajuste de desempenho
 - sobre, 155 ajustes
 - orientados a gargalos, 155 logs de buffer, 159 respostas de buffer, 158 tempo de persistência de conexão ociosa, 156 testes de carga automatizados, 155 sistema operacional para conexões, 159 portas efêmeras, 160 descritores de arquivos abertos, 160 solicitações em uma única conexão , 156 conexões de servidor upstream mantidas abertas, 157
- Instalação dinâmica Perl, 126
 - documentação de instalação, 53 módulos de linguagem disponíveis, 51
 - documentação, 53 perl_set para variável NGINX, 53, 126 ambiente de tempo de execução para variável NGINX, 53, 126 localização física de clientes, 27
- Banco de dados PostgreSQL para Controlador, 161, 162
 - documentação, 163 solicitações de comprovação, 73
- sobre programação, 45
- Consul para automação de configuração, 59 extensões personalizadas, 51 módulos de idioma disponíveis, 51, 53
 - Lua Olá Mundo, 52 njs Olá Mundo, 51
 - Ambiente de leitura Perl, 53 listas de bloqueio dinâmicas, 49 testes de carga automatizados, 155
 - Controlador NGINX via API, 164
- Instalação e configuração do NGINX
 - Ansible, 56
 - Cozinheiro, 55
 - Marionete, 54
 - Pilha de Sal, 58
 - servidores upstream adicionados/removidos em tempo real, 46

download progressivo, 98

Módulo exportador do Prometheus, 129 sobre o Prometheus, 130 proxies encontrando o endereço IP do cliente original, 30 cabeçalhos encaminhados padronizados, 30

Proxy do Google App Engine, 112 falhas no servidor upstream, 10 diretivas

proxy_buffering, 158 diretivas proxy_buffers, 158 diretivas proxy_cache_bypass, 40 diretivas proxy_cache_key, 39 diretivas proxy_cache_lock, 38 diretivas proxy_cache_path, 37 diretivas proxy_cache_purge, 41 diretivas proxy_pass, 112 diretivas proxy_pass_request_body, 63 diretivas proxy_ssl_protocols Validação de instalação de comando de 77 ps, 4

NGINX como daemon ou primeiro plano, 4

Marionete sobre, 54 documentação, 55 Linguagem de modelagem Embedded Puppet (EPP), 54

Instalação e configuração do NGINX, 54 ferramentas de modelagem, 55

Pitão Ansible usando, 57 construção de resumo de hash, 79 link que expira, 81 SaltStack usando, 59

R

balanceamento de carga aleatório, limite de taxa de 16 para mitigação de DDoS, 85 Instalação RedHat de NGINX, 2 redirecionando para HTTPS, 82, 82 método de recarga, 7 repositórios

Implantações do controlador de entrada do Kubernetes, 127

Instalação NGINX, 2

NGINX Plus instalação, 3

Integração OpenID Connect, 68 corpos de solicitação descartados, processamento de 63 solicitações por NGINX, 170, 171

solicitação de rastreamento, 151

OpenTracing, 152 identificadores de solicitação, 152

solicitações permitidas em conexão única, diretiva resolver 156 , 112, 121

Recursos Documentação Ansible, 57 Controlador NGINX, 165

Documentação do gateway de API, 116, 120 Documentação do chef, 56 Documentação do cônsl, 60 Guia de administração de depuração, 171 Documentação da chave da Web JSON, 66 Documentação do JSON Web Token, 65, 67, 68 documentação do módulo de linguagem, 53 Controlador NGINX Referências de API, 164 guia de instalação, 161, 163 Agente do Controlador NGINX, 163, 164 Banco de dados PostgreSQL, 163 especificações técnicas, 161, 163

Documentação do OpenID Connect, 69

Ferramenta de modelagem de configuração de marionetes, 55 documentação, 55

Documentação do SaltStack, 59

SSL gerador de configuração, 74, 76 configuração de teste, 74, 76 documentação de status de stub na habilitação, 130 Protocolo TLS, 74, 76 Guia de firewall de aplicativos da Web, 71

Reposante Acesso da API às métricas do painel, 143 API NGINX Plus, 47

restringindo o acesso com base no endereço IP, 71 por país, 29 métodos múltiplos, 84 links seguros, 78 links seguros que expiram, 80

padrão RFC Chaves da Web JSON, 66 Assinatura da Web JSON, 67 Sincronização de configuração do RHEL, 133 Controle de acesso baseado em função (RBAC), 128 privilégios de root para o processo mestre, 4 balanceamento de carga round-robin, 15 balanceadores de carga de balanceamento de carga, 132

parâmetro de rota com diretiva pegajosa, 18

Certificados RSA versus ECC, 76

linguagem de programação ruby

Chef usando, 56

Linguagem de modelagem Embedded Ruby (ERB),
54

Marionete usando, 54

Segurança de Transporte Estrita HTTP, 83

Redirecionamentos HTTPS, 82

SSL/TLS encerrado antes do NGINX, 82 taxa

limitante de solicitações, 33 data de expiração do bloco de localização, 79 segredo para, 77 várias maneiras de

passar a segurança, 84 restringindo o acesso com base no endereço IP, 71 por país, 29 links seguros, 78 links seguros que expirar, 80

S

SaltStack

sobre, 59

documentação, 59

Instalação e configuração do NGINX, 58

 ferramentas de modelagem, 58

satisfazer a diretiva, 84

conjuntos de escala

 dimensionamento horizontal, 9

 balanceamento de carga, 109

 conjuntos de dimensionamento de máquina virtual, 109

segredos

 sobre como proteger recursos, 78

 bloqueio de localização via segurança,

 77 links seguros gerados com, 78 links

 seguros data de expiração, 79 gerando um

 link que expira, 80 gerando com um

 segredo, 78 restringindo o acesso a ter,

 78 diretiva link_seguro, 80 diretiva

 link_seguro_secreto, 78 segurança sobre,

 71 acessos baseados em endereço IP, 71 por

 país, 29

Diretivas SSL para especificar regras SSL, 77

Firewall de aplicativo da Web

 Módulo de proteção de aplicativos, 89

 Ativação do controlador NGINX, 165 guias

 de início rápido, 71 blocos de localização de blocos de servidor, 7 servindo conteúdo estático,

 6 diretivas de servidor, 121

servidores

 adicionar ou remover em tempo real, 46

 drenagem de conexão, 19 envio de conteúdo

 para o cliente, 94 veiculação de conteúdo

 estático, 6 estado de sessão

drenando conexões graciosamente, 19

 removendo um servidor, 47 balanceamento

 de carga e, 9 controle de sessão persistente,

 18 sticky cookies com NGINX Plus, 16 sticky

 learn com cookie existente, 17 diretiva slice,

 42 parâmetro slow_start, 23

Módulo de proteção de aplicativos

 Arquivo de política do App Protect,

 87 documentação, 89 instalação,

 86

 Firewall de aplicativo da Web, 71, 89, 165

autenticação (consulte autenticação) pares

de chave de certificado, 75 criptografia do

lado do cliente, 74 avançadas, 75

SSL versus protocolo TLS, 74

 Controle de handshake SSL/TLS, 75

compartilhamento de recursos entre

origens, 72 soluções dinâmicas de mitigação de

DDoS, 85 criptografando o tráfego upstream, 77

encontrando o cliente original, 30

Servidor SMTP para NGINX Controller, 162 módulo

split_client, 26

registros SRV, 120

certificados SSL

 Google App Engine, 113

 Módulo SSL para, 74

Módulos SSL

 criptografia do lado do cliente, 74

 geradores de configuração, 74, 76 sites

 de teste de configuração, 74, 76 diretivas

 para especificar regras SSL, 77 proxies de

 conexão gRPC, 92

 HTTP/2 habilitado, 92

Cache de sessão SSL e tempo limite, 76
 Controle de handshake SSL/TLS, 75
 SSL/TLS encerrado antes do NGINX, 82
 Protocolo TLS versus SSL, 74
 diretiva `ssl_certificate`, 74 diretiva
`ssl_certificate_key`, 74 padronização do cabeçalho encaminhado, 30
 Estado
 estado da sessão
 drenando conexões graciosamente, 19, 47
 balanceamento de carga e, 9 controle de sessão persistente, 18 cookies fixos com NGINX Plus, 16
 aprendizado fixo com cookie existente, 17
 compartilhamento de estado e sincronização de zona, 136
 conteúdo estático servido, 6
 stderr para saída do driver de log do Docker, 115 stdout para saída do driver de log do Docker, 115 cookies fixos com NGINX Plus, 16 parâmetros de rota de diretivas adesivas, 18 aprendizado fixo com cookie existente, 17
 servidores de stream
 monitoramento com NGINX Plus API, sincronização de 144 zonas, 136 streaming de mídia, 97 limites de largura de banda, 100
 Segmentação de arquivo Cache Slice, 43
 FLV (Video Flash), 97
 HDS (Streaming Dinâmico HTTP), 99
 HLS (transmissão ao vivo HTTP), 98
 MP4 (MPEG-4), 97 download progressivo, 98
 Cabeçalho Strict-Transport-Security, 83
 documentação de status de stub na habilitação, 130
 habilitação de monitoramento básico, 139
 Exportador Prometheus, 130
 Ouvinte Syslog
 sobre, 151 logs
 de encaminhamento para, 150

T
 Balanceamento de
 carga de servidores TCP, 11
 verificações de integridade ativas, 22 verificações de integridade passivas, 20
 Modelo OSI, 12
 carimbos de data/hora

tempo relativo, 81
 Formato de época Unix, 80
 protocolo TLS
 Negociação de protocolo de camada de aplicativo, 92
 documentação, 74, 76 proxies de conexão gRPC, 92
 HTTP/2 habilitado, 92
 SSL versus, 74
 Controle de handshake SSL/TLS, 75
 SSL/TLS encerrado antes do NGINX, 82 balanceadores de carga DNS na nuvem de gerenciamento de tráfego, 103
 listas de bloqueio dinâmicas, 49
 Sanduíche NLB, 104
 controladores de tráfego na web
 Teste A/B, 25
 aproximadamente, 25 ajustes (veja ajuste de desempenho)

Ubuntu
 sincronização de configuração, 134 instalação do NGINX, 1
UDP
 datagramas, 14
 balanceamento de carga,
 13 verificação de integridade ativa, 22 verificação de integridade passiva, 20
 Data e hora do formato Unix epoch, 80
 URI (identificador de recurso uniforme) para bloco de localização, 7
V
 validando a instalação, 3 variáveis
 variáveis incorporadas em logs, 147 variáveis de ambiente (consulte variáveis de ambiente) perl_set to NGINX variable, 53, 126 video via slicing, 43 virtual machine (VM)
 implantação do NGINX Plus do Azure Marketplace, 109
 implantação do NGINX no Google Compute Motor, 110
 altamente configurável com pouco esforço, 111
 imagens do Google Cloud para criar, 112
 Google Compute Image, 111 balanceamento de carga sobre conjuntos de dimensionamento do Azure, 109

conjuntos de dimensionamento de máquinas virtuais, 109

Imagen de VM no Azure, 107

conjuntos de dimensionamento de máquinas virtuais (VMSS), 109

Protocolo de redundância de roteador virtual (VRRP),

131

Dentro

Firewall de aplicativo da Web (WAF)

Módulo de proteção de aplicativos, 89

Ativação do controlador NGINX, 165 guia de início rápido, 71

controladores de tráfego web

Teste A/B, 25

aproximadamente, 25

roteamento dinâmico via armazenamento de valor-chave, 45

balanceamento de carga round-robin ponderado, 15

X

Cabeçalho X-Forwarded-For (veja padronização

do cabeçalho encaminhado)

Cabeçalho X-Forwarded-Proto, 83

Configuração do plug-in Zipkin para OpenTrace,
152

sincronização de zona, 136

aproximadamente, 136

Sobre o autor

Derek DeJonghe é apaixonado por tecnologia. Sua formação e experiência em desenvolvimento web, administração de sistemas e redes lhe dão uma compreensão completa da arquitetura web moderna. Derek lidera uma equipe de engenheiros de soluções em nuvem e confiabilidade do site e produz infraestrutura de autocorreção e dimensionamento automático para vários aplicativos. Ao projetar, construir e manter aplicativos altamente disponíveis para clientes, ele presta consultoria para organizações maiores à medida que embarcam em sua jornada para a nuvem. Derek e sua equipe estão na vanguarda de uma onda de tecnologia e estão desenvolvendo as melhores práticas de nuvem todos os dias. Com um histórico comprovado de arquitetura de nuvem resiliente, Derek é pioneiro em implantações de nuvem para segurança e manutenção que são do melhor interesse de seus clientes.

Colofão

O animal na capa do NGINX Cookbook é o lince euro-asiático (*Lynx lynx*), a maior das espécies de lince, encontrada em uma ampla faixa geográfica da Europa Ocidental à Ásia Central.

Este gato selvagem tem tufos verticais impressionantes de pêlo escuro no topo de suas orelhas, com cabelos longos e ásperos em seu rosto. Sua pelagem é cinza-amarelada a marrom-acinzentada, com coloração branca no ventre. Este lince é salpicado de manchas escuras, com variantes do norte que tendem a ser mais cinzentas e menos manchadas do que suas contrapartes do sul.

Ao contrário de outras espécies de lince, o lince eurasiano se alimenta de ungulados maiores – animais com cascos – como veados selvagens, alces e até mesmo ovelhas domesticadas. Os adultos precisam de dois a cinco quilos de carne por dia e se alimentam de uma única fonte de alimento por até uma semana.

O lince euro-asiático chegou perto da extinção em meados do século XX, mas os esforços de conservação subsequentes trouxeram o status de conservação do gato para a menor preocupação. Muitos dos animais nas capas da O'Reilly estão ameaçados de extinção; todos eles são importantes para o mundo.

Ilustração colorida de Karen Montgomery, baseada em uma gravura em preto e branco da Zoologia de Shaw. As fontes da capa são Gilroy Semibold e Guardian Sans. A fonte do texto é Adobe Minion Pro; a fonte do título é Adobe Myriad Condensed; e a fonte do código é Ubuntu Mono da Dalton Maag.