Engineer

nginx

en webserver og omvendt proxy

Hvad det er

nginx er et open source-program, der tager imod HTTP-forbindelser og for hver anmodning enten svarer med filer fra disken eller sender anmodningen videre til en anden server og giver svaret tilbage til klienten. Navnet udtales »engine x«, og projektets egne sider skriver det også NGINX. Igor Sysoev skrev programmet, og den første offentlige version, 0.1.0, udkom den 4. oktober 2004. Det er gratis at bruge under en BSD-licens med to klausuler, og F5, Inc. sælger virksomhedsdistributioner, support og undervisning.

Projektet nævner seks roller: HTTP-server, omvendt proxy, indholdscache, load balancer, TCP- og UDP-proxy samt mailproxy. På en webhost har nginx som regel det ene eller begge af to job: at servere statiske filer fra disken og at stå foran en applikationsserver, hvor det terminerer TLS og sender hver anmodning videre indad.

Når nginx starter, kører det én masterproces og flere workerprocesser. Masteren læser og kontrollerer konfigurationen og holder øje med workerne. Workerne kører som en bruger uden særlige rettigheder og står for al behandling af anmodninger. Hver af dem bruger en hændelsesbaseret løkke, bygget på epoll på Linux og kqueue på FreeBSD, så en håndfuld workere kan holde tusindvis af forbindelser åbne på én gang. Direktivet worker_processes bestemmer, hvor mange der er, enten som et fast tal eller som auto for én pr. CPU-kerne, og projektets liste over funktioner sætter 10.000 inaktive keep-alive-forbindelser til omkring 2,5 MB hukommelse.

Konfigurationen er en tekstfil med direktiver. Et simpelt direktiv er et navn og dets parametre, afsluttet med et semikolon. Et blokdirektiv omslutter andre direktiver i krøllede parenteser, og en blok, der kan rumme direktiver, kaldes en kontekst. Konteksterne events og http ligger på øverste niveau. Inde i http

beskriver hver server-blok én virtuel server, valgt ud fra den adresse og port, den lytter på, og derefter ud fra det værtsnavn, klienten bad om. Inde i en server-blok siger hver location-blok, hvad der skal ske med en familie af anmodningsstier.

Brug begrebet i en beslutning

Brug »Hvad er nginx?« i en konkret beslutning. Vi kan knytte begrebet til det system, I bygger eller køber. Daglig drift af servere, mail, konti og opdateringer, så det, alle er afhængige af, forbliver kedeligt.

Se vores ydelser Kontakt os →

Undersøg antagelserne bag ordet

Brug forklaringen og det tilknyttede arbejde til at formulere spørgsmål til jeres projekt. Vi kan undersøge antagelserne sammen, før et værktøj vælges. Har drevet 10 PostgreSQL- og MS SQL-servere — 10 databaseservere udrullet og holdt i god drift

Har drevet 10 PostgreSQL- og MS SQL-servere Kontakt os →

Hvorfor det betyder noget

Webserveren er det ene program, som alle besøgende er i berøring med, så dens opførsel under belastning og dens standardindstillinger når ud til alle. To egenskaber taler for nginx.

Den første er, hvor billigt det bærer langsomme forbindelser. Buffering af svar fra en server bag proxyen er slået til som standard: nginx tager imod backendens svar så hurtigt, som backenden kan levere det, skriver det til en midlertidig fil, når det bliver for stort til hukommelsen, og sender det så videre til klienten i det tempo, klienten kan klare. Applikationsserveren er fri til sin næste anmodning, mens en telefon med svagt signal stadig henter den forrige.

Den anden er, at ændringer i konfigurationen går i luften uden at tabe anmodninger. En redigeret konfiguration får først virkning ved en reload eller en genstart. Ved en reload kontrollerer masteren den nye konfiguration og forsøger at tage den i brug. Lykkes det, starter nye workere med den, mens de gamle holder op med at tage imod forbindelser, gør de anmodninger færdige, som de har i gang, og afslutter. Mislykkes det, ruller masteren tilbage og fortsætter med den konfiguration, den allerede havde. Projektet nævner også opgradering af selve programfilen uden at afbryde klienterne.

Prisen betales i konfigurationssproget, hvis regler for forrang og nedarvning er der, hvor de fleste overraskelser bor. For en anmodningssti finder nginx først den location med det længste matchende præfiks og husker den, prøver så de locations, der bruger regulære udtryk, i den rækkefølge, de står i filen, og tager det første, der matcher. Slår alle de regulære udtryk fejl, bruges den huskede

præfiks-location. En location markeret med `=` kræver, at stien passer præcist, og afslutter søgningen med det samme, og `^~` på det længste præfiks får nginx til at springe de regulære udtryk over. Nogle direktiver, blandt dem add_header og proxy_set_header, arves kun fra det omsluttende niveau, hvis det aktuelle niveau ikke selv definerer nogen.

Certifikater kom længe fra en ACME-klient, der kørte ved siden af nginx. Projektet tilbyder nu også ngx_http_acme_module, der leveres separat fra kernebygget.

Sådan ser det ud i praksis

To eksempler på hostingarbejde fra grundlæggerens tidligere roller involverer nginx, og i begge delte nginx serverne med Apache frem for at afløse den.

Det første var fyrre kundewebsites, der skulle være hurtige og altid kunne nås, og de lå på servere med Ubuntu Linux, hvor både Apache og nginx var i brug. Opgaven var at skrive deres konfiguration, tune dem for hastighed og passe dem, mens rigtige besøgende brugte dem, og alle fyrre forblev højt tilgængelige. To webservere i én flåde betyder, at hver regel findes på to dialekter: en VirtualHost-blok i Apache er en server-blok i nginx, og en rettelse til den ene skal oversættes til den anden.

Det andet strakte sig fra 2015 til 2022. Et ACME-workflow overtog certifikaterne for hundrede Docker-applikationer, udstedte hvert certifikat, fornyede det inden udløb og installerede det på Ubuntu-værterne uden et menneske i løkken, og også på de værter var både Apache og nginx i brug. På nginx' side er en fornyet fil på disken kun halvdelen af sådan en ændring. Et certifikat, der er angivet med en almindelig sti, læses, når nginx indlæser sin konfiguration, og dokumentationen bemærker, at kun et filnavn bygget af variabler får det til at indlæse certifikatet igen ved hvert TLS-håndtryk. En fornyelse, der skriver nye filer, når derfor først de besøgende efter en reload, så enhver automatiseret fornyelse til nginx skal slutte med én.

Hvor det går galt

En header, der er sat for hele sitet, forsvinder på én sti. En location med bare én egen add_header arver ingen fra server- eller http-niveauet, så én cacheheader tilføjet til en enkelt location fjerner stille alle sikkerhedsheadere over den. proxy_set_header følger samme regel. add_header gælder desuden

kun for succes- og omdirigeringskoder som 200, 301 og 304, medmindre den får `always`, så fejlsider går ud uden. Hold fælles headere i én fil, der hentes ind med include på hvert niveau, der sætter sine egne, eller sæt fra 1.29.3 `add_header_inherit merge;`, tilføj `always`, hvor fejlsvar skal have dem, og kontrollér hver sti med `curl -sI`.

Anmodninger til ét site bliver besvaret af et andet. nginx matcher først adresse og port og derefter Host-headeren mod server_name. Er navnet ukendt for alle server-blokke på den port, går anmodningen til standardserveren, som er den første på listen, medmindre en anden er markeret default_server. En tastefejl i et nyt sites server_name serverer derfor stille en gammel nabos sider. Erklær en eksplicit standardserver, der besvarer ukendte navne med `return 444`, som lukker forbindelsen, så en fejl viser sig som en fejl og ikke som det forkerte site.

Uploads fejler med 413 Request Entity Too Large. client_max_body_size er som standard 1 MB, og nginx afviser en større anmodningskrop, før applikationen ser den. Hæv grænsen i den location, der tager imod uploads, frem for i hele serveren, og hold den i takt med applikationens egen grænse.

Backenden bygger links til den forkerte vært eller får den forkerte sti. Som standard sender nginx serveren bag proxyen en Host-header bygget af adressen i proxy_pass, for eksempel `127.0.0.1:8080`, så applikationen laver omdirigeringer og absolutte links til den adresse. Sæt `proxy_set_header Host $host;` i den location, der videresender. Stier ændrer sig med proxy_pass' form: skrevet med en URI, selv et enkelt `/`, erstatter den den del af stien, der matchede location, mens den uden en URI sender stien uændret videre, så test en indlejret sti efter hver ændring.

Klienterne får stadig den gamle konfiguration eller det gamle certifikat. En redigeret fil eller et certifikat fornyet i nattens løb ændrer intet for en kørende nginx før en reload, og en reload af en defekt konfiguration lader den gamle blive. Test før hver reload, læs så fejlloggen, og kontrollér, hvad klienterne faktisk modtager:

```
nginx -t && nginx -s reload
openssl s_client -connect example.com:443 \
  -servername example.com </dev/null |
  openssl x509 -noout -enddate
```

Læs mere

- nginx documentation — projektets oversigt over vejledninger og direktivreferencer pr. modul.
- Beginner's Guide — master- og workerprocesser, reload og konfigurationens opbygning.
- How nginx processes a request — hvordan en server-blok vælges ud fra adresse, port og Host-header.
- Module ngx_http_core_module — rækkefølgen for location-match, server_name og client_max_body_size.
- Module ngx_http_proxy_module — proxy_pass, standardheadere, timeouts og buffering af svar.
- Module ngx_http_headers_module — add_header, de svarkoder, den gælder for, og dens regel for nedarvning.

Ordliste

Vælg ud fra jeres faktiske vilkår

Medbring udrulningsforløbet, driftsrisiciene og dem, der har vagten. Vi kan afgrænse en ændring med kontroller, en vej tilbage og tydeligt ansvar.

Se, hvad det koster Kontakt os →

Gør begreberne praktiske

Spørg, hvordan »Hvad er nginx?« gælder i jeres organisation. Et konkret eksempel fra jeres nuværende system er nok til at begynde samtalen.

Se vores ydelser

welcome@engineer.company +45 50 10 61 84 København, Danmark

Tag denne side med dig

Visitkort

Lavet med kærlighed af D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Copyright © 2025 – nu · Engineer ApS · Alle rettigheder forbeholdes.

Udskrevet fra https://engineer.company/da/glossary/nginx/