Engineer

Driftstopp

den tid då en tjänst inte svarar någon

Vad det är

Driftstopp är den tid då en tjänst är ur funktion för dem som är beroende av den. Andra ord för samma sak är avbrott och otillgänglighet; motsatsen är upptid, och den andel av tiden som en tjänst är uppe är dess tillgänglighet, som oftast anges i procent över en månad eller ett år.

Det rena fallet är totalt: webbplatsen laddas inte, API:t vägrar ta emot anslutningar, databasen ligger nere. Det vanliga fallet är rörigare. En kassa som får timeout medan produktkatalogen fungerar, en inloggningssida som laddas men avvisar varje lösenord, ett API som svarar på vart tredje anrop: vart och ett är ett driftstopp för den som behöver just den vägen, trots att en hälsokontroll riktad mot startsidan skulle visa grönt. En användbar definition räknar vad användarna kan göra, och bryr sig mindre om huruvida en process körs.

Driftstopp finns i två slag. Det oplanerade driftstoppet är avbrottet som ingen valde: en disk, en misslyckad driftsättning, ett certifikat som har gått ut, ett beroende som gick ner först. Det planerade driftstoppet väljs i förväg, oftast inom ett servicefönster, för att uppgradera, migrera eller flytta något som inte kan ändras medan det betjänar trafik. Planerade driftstopp är lättare att leva med, eftersom användarna kan förvarnas, men de räknas ändå mot tillgängligheten om inte avtalet uttryckligen undantar dem.

HTTP har ett ordförråd för själva ögonblicket. RFC 9110 definierar klassen 5xx av statuskoder för en server som vet att den har gjort fel eller inte kan utföra det som efterfrågades. Tre av dem betyder olika saker under ett avbrott. 502 Bad Gateway säger att en proxy fick ett ogiltigt svar från servern bakom sig. 504 Gateway Timeout säger att proxyn väntade för länge på ett svar. Och 503 Service

Unavailable är den som ska skickas med avsikt: specifikationen knyter den till en tillfällig överbelastning eller ett schemalagt underhåll, och låter servern säga när klienten kan komma tillbaka:

```
HTTP/1.1 503 Service Unavailable
Retry-After: 120
```

Det talar om för webbläsare, API-klienter och sökmotorernas sökrobotar att tillståndet är tillfälligt och föreslår att de väntar två minuter, vilket är ett helt annat budskap än en trasig sida som serveras med en statuskod för framgång.

Använd begreppet i ett beslut

Använd ”Vad är ett driftstopp?” i ett konkret beslut. Vi kan knyta begreppet till systemet ni bygger eller köper. Övervakning och larm som säger till medan ett fel fortfarande är litet, och siffrorna som visar hur ofta det förblir så.

Se våra tjänster Hör av er →

Granska antagandena bakom ordet

Använd förklaringen och det länkade arbetet för att formulera frågor till projektet. Vi kan undersöka antagandena tillsammans innan ett verktyg väljs. Byggde företagets första observerbarhetsdashboard — företagets första överblick över de egna systemen

Byggde företagets första observerbarhetsdashboard Hör av er →

Varför det spelar roll

Mål för tillgänglighet skrivs i nior, och niorna går att översätta till tid. Under ett år på 365 dagar, alltså 8 760 timmar, tillåter varje mål så här mycket driftstopp:

- 99 %: 87,6 timmar per år, ungefär tre och ett halvt dygn;
- 99,9 %: 8,76 timmar per år, eller 43,2 minuter under en månad på 30 dagar;
- 99,95 %: 4,38 timmar per år, eller 21,6 minuter under en månad på 30 dagar;
- 99,99 %: 52,56 minuter per år;
- 99,999 %: ungefär 5,26 minuter per år.

Varje extra nia delar utrymmet med tio, och kostnaden för att förtjäna den stiger brant. Vid 99,9 % hinner en person bli väckt, logga in, hitta problemet och åtgärda det inom månadens budget. Vid 99,99 % tillåter hela året mindre än en timme, vilket kanske ger plats för en enda incident, så återställningen måste ske automatiskt. Vid 99,999 % måste till och med en rutinmässig omstart ske utan att användarna märker något.

Beroenden multipliceras. Ett anrop som behöver en lastbalanserare, en applikation och en databas, var och en tillgänglig 99,9 % av tiden och med fel som uppstår oberoende av varandra, betjänas bara när alla tre är uppe: $0{,}999^3$, eller ungefär 99,7 %, vilket blir omkring 26 timmars driftstopp per år i stället för 8,76. En tjänst kan aldrig bli mer tillgänglig än kedjan av det den behöver, såvida den inte tar bort länkar ur kedjan eller dubblerar dem.

Planeringen börjar på verksamhetssidan. Vägledningen från NIST om kontinuitetsplanering för federala informationssystem, SP 800-34, namnger tre tal som är värda att bestämma innan något går sönder. Maximal acceptabel avbrottstid, på engelska maximum tolerable downtime (MTD), är det totala avbrott som systemets ägare kan godta. Målet för återställningstid, recovery time objective (RTO), är hur länge ett system kan vara nere innan konsekvenserna blir oacceptabla, och det måste vara kortare än MTD. Målet för återställningspunkt, recovery point objective (RPO), anger hur långt tillbaka i tiden data får återställas från den senaste säkerhetskopian, så det mäter hur mycket data som får gå förlorad snarare än hur länge något är nere. När de tre talen är nedskrivna blir ett schema för säkerhetskopior och en design för failover svar på en fråga i stället för gissningar.

Så ser det ut i praktiken

På en strömningsplattform för IPTV och OTT med kunder i Kina, USA och Tyskland var varje timme bästa sändningstid för någon, så en lugn timme för underhåll fanns helt enkelt inte. Supporten pågick dygnet runt för ungefär tusen servrar plus ett stort antal kundägda system. En stor del av arbetet bestod i att märka när något började glida innan det blev ett avbrott, eftersom ett avbrott på en direktsänd ström är någons tv som stannar. Plattformen förblev oavbrutet tillgänglig för sin globala publik.

På ett medelstort techföretag gick tio databasservrar med PostgreSQL och SQL Server med 99,9 % upptid, vilket enligt räkningen ovan är ett utrymme på 8,76 timmar per år för var och en. Bakom siffran fanns automatiserade säkerhetskopior med `pg_dump` och jobb i SQL Server Agent, med regler för lagringstid och kopior på en annan plats, övervakning i Prometheus och Grafana, och dokumenterade rutiner för katastrofåterställning med återställning till en given tidpunkt och failover.

Och på en park av trettio servrar med Ubuntu vars uppsättning hade vuxit av en slump byggdes katastrofåterställningen upp på nytt, med säkerhetskopior och återställningsrutiner som faktiskt testades. Nätverkskonfigurationen blev en och samma härdade baskonfiguration på varje instans. Risken för driftstopp minskade, med återställningsvägar som hade prövats innan någon behövde dem.

Var det går fel

Övervakningen sitter inuti avbrottet. En hälsokontroll som körs på samma server, eller i samma datacenter, fortsätter att visa grönt medan användarna möter ett utgånget TLS-certifikat, en trasig DNS-post eller en lastbalanserare som pekar på ingenting. Kontrollera utifrån, från mer än en plats, längs samma väg som en verklig användare tar, och låt larmet utgå från det den kontrollen ser.

Statuskoden på felsidan är 200. Ett ramverk som renderar en vänlig felsida med en statuskod för framgång döljer avbrottet för övervakningen, för klienter som annars skulle försöka igen och för sökrobotar, som kan indexera felsidan i stället för den riktiga sidan. Svara med en felstatus när något har gått sönder, och låt planerat underhåll svara 503 med en header för Retry-After.

Återställningen har bara provats på papperet. En säkerhetskopia bevisar bara att byte har kopierats. Tills en återställning har körts, tidmätts och kontrollerats mot datan är återställningstiden en gissning, och den är ofta betydligt längre än det RTO som står på papperet. Återställ enligt ett schema, till en separat miljö, och skriv ned hur lång tid det tog. Detsamma gäller failover: en reservserver som aldrig har tagit över är en förhoppning, och övningar i failover under lugna förhållanden hjälper till att upptäcka problem före en incident.

Releaserna orsakar de avbrott de skulle förhindra. En driftsättning som stoppar den gamla versionen innan den nya är redo lämnar en lucka vid varje release. En ändring i databasschemat som skriver om en stor tabell under ett exklusivt lås blockerar varje databasfråga mot den tills den är klar. Rulla ut nya instanser bakom en kontroll av att de är redo innan de gamla stoppas, dela upp ändringar i schemat i ett steg som utökar och ett senare steg som drar ihop, så att både gammal och ny kod fungerar däremellan, och sätt en PodDisruptionBudget i Kubernetes så att en tömning av en nod aldrig kan ta ner alla repliker på en gång.

Läs mer

- RFC 9110: HTTP Semantics, section 15.6 — statuskoderna i klassen 5xx, bl.a. 503 och dess ledtråd i Retry-After.
- NIST SP 800-34 Rev. 1: Contingency Planning Guide for Federal Information Systems — där maximal acceptabel avbrottstid, RTO och RPO definieras.
- PostgreSQL: High Availability, Load Balancing, and Replication — reservservrar och failover, så som databasens egen manual beskriver dem.
- PostgreSQL: Continuous Archiving and Point-in-Time Recovery — grundkopian och WAL-arkivet som en återställning till en given tidpunkt bygger på.
- Kubernetes: Disruptions — frivilliga störningar och de PodDisruptionBudgets som begränsar dem.

Ordlista

Välj utifrån era faktiska förutsättningar

Ta med driftsättningsvägen, driftsriskerna och dem som har jour. Vi kan avgränsa en förändring med kontroller, en väg tillbaka och tydligt ansvar.

Se vad det kostar Hör av er →

Gör begreppen praktiska

Fråga hur ”Vad är ett driftstopp?” gäller i er organisation. Ett konkret exempel från ert nuvarande system räcker för att börja samtalet.

Se våra tjänster

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

Ta med dig den här sidan

Visitkort

Omsorgsfullt skapad med kärlek av D.

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

Upphovsrätt © 2025 – nu · Engineer ApS · Alla rättigheter förbehållna.

Utskrivet från https://engineer.company/sv/glossary/downtime/