Engineer

Nedetid

tidsrommet der en tjeneste ikke svarer noen

Hva det er

Når en tjeneste står stille for dem som er avhengige av den, kalles tiden det varer, nedetid. Andre navn på det samme er driftsavbrudd og utilgjengelighet. Det motsatte er oppetid, og andelen av tiden en tjeneste er oppe, er tilgjengeligheten dens, som regel oppgitt i prosent over en måned eller et år.

Det rene tilfellet er totalt: nettstedet laster ikke, API-et avviser tilkoblinger, databasen er nede. Det vanlige tilfellet er rotete. En betaling som går ut på tid mens katalogen virker, en innloggingsside som laster, men avviser hvert passord, et API som svarer på én forespørsel av tre: hver av dem er nedetid for den som trenger akkurat den veien, selv om en helsesjekk rettet mot forsiden melder grønt. En brukbar definisjon teller det brukerne kan gjøre, ikke om en prosess kjører.

Nedetid finnes i to slag. Uplanlagt nedetid er driftsavbruddet ingen valgte: en disk, en mislykket produksjonssetting, et utløpt sertifikat, en avhengighet som gikk ned først. Planlagt nedetid velges på forhånd, som regel innenfor et vedlikeholdsvindu, for å oppgradere, migrere eller flytte noe som ikke kan endres mens det betjener trafikk. Planlagt nedetid er lettere å leve med, siden brukerne kan varsles, men den teller likevel mot tilgjengeligheten, med mindre avtalen uttrykkelig holder den utenfor.

HTTP har et ordforråd for selve øyeblikket. RFC 9110 definerer 5xx-klassen av statuskoder for en server som vet at den har gjort en feil, eller som ikke kan utføre det den ble bedt om. Tre av dem betyr forskjellige ting under et driftsavbrudd. 502 Bad Gateway sier at en proxy fikk et ugyldig svar fra serveren bak seg. 504 Gateway Timeout sier at proxyen ventet for lenge på et svar. Og 503

Service Unavailable er den du sender med vilje: spesifikasjonen knytter den til midlertidig overbelastning eller planlagt vedlikehold og lar serveren si når klienten kan komme tilbake:

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

Det forteller nettlesere, API-klienter og søkeroboter at tilstanden er midlertidig, og foreslår å vente to minutter. Det er en helt annen beskjed enn en ødelagt side servert med en statuskode for suksess.

Bruk begrepet i en beslutning

Bruk «Hva er nedetid?» i en konkret beslutning. Vi kan knytte begrepet til systemet dere bygger eller kjøper. Overvåking og varsler som sier fra mens en feil fortsatt er liten, og tallene som viser hvor ofte den holder seg liten.

Se tjenestene våre Ta kontakt →

Undersøk antakelsene bak ordet

Bruk forklaringen og det tilknyttede arbeidet til å formulere spørsmål for prosjektet. Vi kan undersøke antakelsene sammen før et verktøy velges. Bygde selskapets første observerbarhetsdashbord — selskapets første innsyn i egne systemer

Bygde selskapets første observerbarhetsdashbord Ta kontakt →

Hvorfor det er viktig

Mål for tilgjengelighet skrives i niere, og niene kan regnes om til tid. Over et år på 365 dager, som er 8 760 timer, gir hvert mål rom for så mye nedetid:

- 99 %: 87,6 timer i året, omtrent tre og en halv dag;
- 99,9 %: 8,76 timer i året, eller 43,2 minutter i en måned på 30 dager;
- 99,95 %: 4,38 timer i året, eller 21,6 minutter i en måned på 30 dager;
- 99,99 %: 52,56 minutter i året;
- 99,999 %: omtrent 5,26 minutter i året.

Hver ekstra nier deler rammen på ti, og prisen for å nå den stiger bratt. Ved 99,9 % kan en person vekkes, logge inn, finne problemet og rette det innenfor månedens ramme. Ved 99,99 % gir hele året mindre enn en time, og det gir plass til kanskje én driftshendelse, så gjenopprettingen må skje automatisk. Ved 99,999 % må selv en rutinemessig omstart skje uten at brukerne merker det.

Avhengigheter ganges med hverandre. En forespørsel som trenger en lastbalanserer, en applikasjon og en database, hver tilgjengelig 99,9 % av tiden og med feil som er uavhengige av hverandre, blir bare betjent når alle tre er oppe: 0,999³, eller omtrent 99,7 %, som er rundt 26 timer nedetid i året i stedet for 8,76. En tjeneste kan ikke være mer tilgjengelig enn kjeden av ting den trenger, med mindre den fjerner ledd fra kjeden eller dobler dem.

Planleggingen begynner på forretningssiden. NISTs veiledning for beredskapsplanlegging i føderale systemer, SP 800-34, nevner tre tall som er verdt å avklare før noe svikter. Maksimal akseptabel nedetid (MTD, maximum tolerable downtime) er det samlede avbruddet eieren av systemet vil godta. Målet for gjenopprettingstid (RTO, recovery time objective) er hvor lenge et system kan være nede før konsekvensene blir uakseptable, og den må være kortere enn MTD. Målet for gjenopprettingspunkt (RPO, recovery point objective) sier hvor langt tilbake i tid dataene kan gjenopprettes fra siste sikkerhetskopi, så det måler hvor mye datatap som kan tåles, ikke hvor lenge noe er nede. Med de tre tallene skrevet ned blir en plan for sikkerhetskopiering og et design for failover svar på et spørsmål, ikke gjetninger.

Slik ser det ut i praksis

På en IPTV/OTT-strømmeplattform med kunder i Kina, USA og Tyskland var hver time beste sendetid for noen, så en rolig time å gjøre vedlikehold i fantes aldri. Støtten gikk døgnet rundt for ca. 1 000 servere, pluss en god del kundeeide systemer. Mye av arbeidet handlet om å merke at noe var i ferd med å skli ut før det ble et driftsavbrudd, for på en plattform for direktestrømming er et driftsavbrudd at noens TV stopper. Plattformen holdt seg kontinuerlig tilgjengelig for et globalt publikum.

I et mellomstort teknologiselskap holdt ti databaseservere med PostgreSQL og SQL Server 99,9 % oppetid, som etter regnestykket over gir en ramme på 8,76 timer i året for hver av dem. Bak tallet lå automatiske sikkerhetskopier med `pg_dump` og SQL Server Agent-jobber, med regler for oppbevaring og ekstern lagring, overvåking i Prometheus og Grafana, og dokumenterte prosedyrer for katastrofegjenoppretting, bl.a. gjenoppretting til et bestemt tidspunkt og failover.

Og på en flåte av tretti Ubuntu-servere der oppsettet hadde vokst organisk, ble katastrofegjenopprettingen bygd opp på nytt med sikkerhetskopier og gjenopprettingsprosedyrer som faktisk var testet. Nettverkskonfigurasjonen ble

én herdet grunnlinje på hver eneste instans. Risikoen for nedetid gikk ned, med veier tilbake som var prøvd før noen trengte dem.

Hvor det går galt

Overvåkingen sitter inne i driftsavbruddet. En helsesjekk som kjører på den samme serveren, eller i det samme datasenteret, fortsetter å melde grønt mens brukerne møter et utløpt TLS-sertifikat, en ødelagt DNS-post eller en lastbalanserer som peker på ingenting. Sjekk utenfra, fra mer enn ett sted, langs den veien en faktisk bruker går, og la driftsvarselet bygge på det den sjekken ser.

Feilsiden sier 200. Et rammeverk som viser en vennlig feilside med en statuskode for suksess, skjuler driftsavbruddet for overvåkingen, for klienter som ellers ville prøvd igjen, og for søkeroboter, som kan indeksere feilsiden i stedet for den ekte siden. Feil skal bære en feilstatus, og planlagt vedlikehold skal svare med 503 og en Retry-After-header.

Sikkerhetskopien har aldri vært gjenopprettet. En sikkerhetskopi beviser bare at bytes ble kopiert. Før en gjenoppretting er kjørt, tatt tiden på og sjekket mot dataene, er gjenopprettingstiden en gjetning, og ofte langt lengre enn RTO-en på papiret. Gjenopprett etter en fast plan, inn i et eget miljø, og skriv ned hvor lang tid det tok. Det samme gjelder failover: en reserve som aldri har tatt over, er et håp, og øving på failover under rolige forhold hjelper med å avdekke problemer før en hendelse.

Utgivelser skaper driftsavbruddene de skulle forhindre. En produksjonssetting som stopper den gamle versjonen før den nye er klar, etterlater et hull ved hver utgivelse. En endring i databaseskjemaet som skriver om en stor tabell under en eksklusiv lås, blokkerer hver spørring mot den til den er ferdig. Rull ut nye instanser bak en sjekk av om de er klare før de gamle stopper, del skjemaendringer i et utvidende steg og et senere innsnevrende steg (expand og contract) slik at både gammel og ny kode virker i mellomtiden, og sett et PodDisruptionBudget på Kubernetes slik at tømming av en node ikke kan ta ned alle replikaene på en gang.

Les mer

- RFC 9110: HTTP Semantics, section 15.6 — statuskodene i 5xx-klassen, blant dem 503 og hintet Retry-After.

- NIST SP 800-34 Rev. 1: Contingency Planning Guide for Federal Information Systems — der maksimal akseptabel nedetid, RTO og RPO er definert.
- PostgreSQL: High Availability, Load Balancing, and Replication — reserveservere og failover, slik databasens egen manual beskriver dem.
- PostgreSQL: Continuous Archiving and Point-in-Time Recovery — basissikkerhetskopien og WAL-arkivet som gjenoppretting til et bestemt tidspunkt bygger på.
- Kubernetes: Disruptions — frivillige avbrudd og de PodDisruptionBudgets som begrenser dem.

Ordliste

Velg ut fra deres faktiske rammer

Ta med utrullingsløpet, driftsrisikoene og dem som har vakt. Vi kan avgrense en endring med kontroller, en vei tilbake og tydelig ansvar.

Se hva det koster Ta kontakt →

Gjør begrepene praktiske

Spør hvordan «Hva er nedetid?» gjelder i organisasjonen deres. Et konkret eksempel fra dagens system er nok til å starte samtalen.

Se tjenestene våre

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

Ta med deg denne siden

Visittkort

Laget med kjærlighet av D.

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

Opphavsrett © 2025 – nå · Engineer ApS · Alle rettigheter forbeholdt.

Skrevet ut fra https://engineer.company/nb/glossary/downtime/