LabAssignment3-Task1

Observability

The Three pillars of observability

Tracing, Causal ordering of recorded events.

Logging, Records events that have happened.

Metrics, Data combined of events we track.

Blackbox vs. Whitebox monitoring

Blackbox - Monitorering av själva systemet/servern, t.ex minne, cpu och disk användning, utan att ta hänsyn vad som körs på servern.

Whitebox - Monitorering av en applikation/process.

Alerting

Används ofta för att systemet ska skicka ut ett meddelande om något är fel, det kan vara något kritiskt som behöver ses på direkt, eller något mindre kritiskt men som ändå behöver uppmärksamhet. Meddelandet kan skickas genom olika kanaler, t.ex email eller slack.

Monitoring Signals

De fyra gyllene signalerna enligt Google.

Latency - Tiden det tar att betjäna en begäran, dock behöver man också skilja på misslyckade förfrågningar och lyckade.

Traffic - Hur mycket trafik/förfrågningar som tjänsten belastas med.

Errors - Andelen förfrågningar som misslyckas.

Saturation - Ett mått på systemanvändning och de resurser som är mest begränsade, t.ex minne eller cpu.

Why do we need logs?

För att lättare kunna felsöka och hitta vad som kan gått fel i systemet/applikationen/servern.

Man bör också tänka på vad man ska logga, så man inte loggar allt utan att man har någon användning av det.

What is the difference between logs and metrics?

Loggar beskriver olika händelser i applikationen/servern.

Metrics är en mätning vid en tidpunkt för systemet/applikationen. Hämtas oftast i tidsintervaller, som varje timme eller var 10e minut.

What is Tracing ?

Följer en request/förfrågan genom systemet för att se vilken väg den tar och för att kunna undersöka “bottlenecks” där förfrågan kan ta längre tid på sig eller fastna.

Service Mesh

What is a Service Mesh?

Ett infrastruktur lager för att underlätta kommunikation mellan tjänster.

How does enable it’s data plane?

Med hjälp av sidecars som placeras på poddar eller containrar. Som sen används för kommunikation mellan tjänsterna.

mTLS

Säkerställer att autentisering för trafiken är säker i båda riktningarna mellan en klient och server.

Circuit Breaking

Upptäcker fel och stoppar försök/requests att utföras och skickar istället ett felmeddelande. Öppnar igen tills en timeout som satts går ut, börjar då släppa fram requests igen.

Traffic splitting/steering

Regler om hur trafik ska gå igenom nätverket. Splitting, styr t.ex 5% av trafiken till en tjänst och 95% av trafiken till en annan. Steering, styr förfrågan beroende på något, t.ex typ av webbläsare(Chrome,Safari,Firefox) eller om förfrågan kommer från en mobilapp eller webbläsare.

Fault Injection

Simulera olika fel, används ofta inom chaos engineering**.**

Rolling Update

Uppdatera utan någon nertid, genom att stegvis uppdatera poddar.

Serverless

What are some benefits and drawbacks of adopting serverless architecture?

Fördelar

Skalbarhet, hanterar 1 användare lika bra som 1000 användare pga av att den automatiskt skalar upp och ner beroende på behov.

Snabbare deployments och uppdateringar.

Mindre driftkostnad (inte alltid sant) då applikationen inte använder några resurser när den inte används. Har du då din applikation på AWS eller Google så betalar du bara för resurserna som applikationen använder.

Nackdelar

Svårt att omvandla sin applikation till serverless om man har en monolitisk arkitektur, då man ofta behöver gå över till en microservice arkitektur först, vilket kan ta tid.

Prestanda kan bli sämre, om applikationen inte körts på ett tag och en request kommer in så behöver den göra en “cold start”.

Risken att bli låst till en leverantör då koden ofta skrivs/anpassas till just den leverantören.