S O L O P R E N E U R . S E

Agent-First
Dokumentation

Hur du strukturerar ditt företag så att AI
kan hjälpa dig på riktigt, utan att du behöver vara programmerare

Del 1 av 2

Daniel Borkeby

Mars 2026

Varför den här guiden?

Jag driver en koncern med fyra bolag. Bokföring, löner, fakturering, avtal, teknisk dokumentation: allt som hör till. För ett år sedan fanns det mesta i mitt huvud, i e-posttrådar och i utspridda dokument på olika ställen.

Jag använder **Claude**, Anthropics AI, som en medarbetare. Jag ger Claude tillgång till mitt repo, alla mina dokument, beslut och processer, och då kan den läsa, förstå sammanhang och hjälpa mig utföra arbete. Inte som en chatbot man ställer frågor till, utan som någon som faktiskt kan agera i min dokumentation.

Men det fungerar bara om dokumentationen är gjord för det. Den här guiden beskriver hur jag löste det. Inte med programmering eller dyra system, utan med en enkel struktur av textfiler och mappar.

Vad betyder agent-first?

Agent-first innebär att du strukturerar din information så att en AI kan *läsa och förstå* den, inte bara du. Det handlar om att organisera det du redan har på ett sätt som är tydligt och konsekvent.

> Tänk på det som att anställa en ny medarbetare. Om du förklarar allt muntligt varje gång blir det tungt. Men om du har en handbok, en filstruktur och tydliga mallar kan personen komma igång själv. AI fungerar likadant, fast den läser ännu snabbare.

I praktiken betyder det tre saker:

- **Strukturerat format:** dina dokument har en tydlig rubrik, kategori och taggar som en maskin kan filtrera på.
- **Konsekvent mönster:** alla dokument följer samma mall, oavsett om det är ett avtal, ett beslut eller en teknisk spec.
- **Centralt register:** det finns en fil som listar alla dokument med metadata, så att AI:n inte behöver ”söka runt”.

Två sätt att navigera, samma information

Det här är en nyckelinsikt: **människor och maskiner navigerar information på helt olika sätt**.

När jag letar efter något öppnar jag en mapp, tittar på filnamnen och klickar mig in. Jag föredrar en logisk mappstruktur där saker ligger där jag förväntar mig. Typ: avtal i en avtals-mapp, beslut i en besluts-mapp.

Claude gör inte så. Den öppnar istället registret, en enda fil som listar alla dokument, och filtrerar på typ, taggar eller system. Den behöver inte ens veta vilken mapp filen ligger i.

> Liknelse: Du hittar en bok på biblioteket genom att gå till rätt avdelning och hylla. En sökmotor hittar den genom att slå i katalogen. Båda behöver finnas: hyllan för dig, katalogen för maskinen.

Därför har jag både en **mappstruktur** (för människor) och en **registry.yml** (för AI:n). De innehåller samma information, men presenterar den på det sätt som respektive användare behöver.

Varför Markdown och YAML?

Jag använder **Markdown** (.md-filer) för allt innehåll och **YAML-frontmatter** för metadata. Det är det mest praktiska formatet för agent-first dokumentation:

- **Människa:** snabbt att skriva, läsa och jämföra ändringar
- **Maskin:** konsekvent struktur via frontmatter (metadata) + ren text
- **Versionering:** Git fungerar extremt bra med textfiler och varje ändring syns.
- **Portabelt:** låser dig inte till en leverantör. Du kan alltid flytta filerna

Så här byggde jag strukturen

Steg 1: Välj ett hem för dina dokument

Jag valde **GitHub** som plattform. Inte för att det är en utvecklarplattform, utan för att det ger tre saker:

- **Versionshantering:** varje ändring sparas med tidstämpel och vem som gjorde den.
- **Historik:** du kan alltid gå tillbaka och se hur ett dokument såg ut förra månaden.
- **Samarbete:** både du och Claude kan arbeta mot samma filer.

Du behöver inte kunna programmera för att använda GitHub. Tänk på det som en smart version av Dropbox där varje sparning loggas.

Du kan också rendera dina Markdown-filer som en snygg intern handbok via GitHub Pages, MkDocs eller Docusaurus.

Steg 2: Separera internt och externt

Jag skapade **två separata arkiv** (repos):

- **operations:** allt internt, som affärsbeslut, avtal, bokföring, tekniska specifikationer och drift.

- **public:** allt publikt, som hjälptexter, FAQ, guider och nyheter.

Varför? En supportagent som svarar kunder ska **bara** ha tillgång till det publika. Den ska aldrig kunna se intern prissättning, strategidokument eller avtalstexter. Genom att fysiskt separera filerna, istället för att bara sätta en flagga, blir det omöjligt att läcka fel information.

> Tips: Skriv guider och innehåll så att de i första hand kan vara externa. Om något måste vara internt: lägg det i det interna repot.

Steg 3: Ge varje dokument en identitet

Varje fil börjar med ett litet informationsblock, ett så kallat *frontmatter*. Det är några rader i början av filen som beskriver vad den handlar om:

```
id: DEC-2026-0014
type: decision
title: "Bokföringsdomän"
status: approved
tags: [bokslut, koncern, agent-first]
```

Det är **inte kod**. Det är etiketter. Som att sätta en lapp på en pärm. Claude läser lappen och vet omedelbart vad filen handlar om.

Några viktiga nyckelord:

- **status:** draft, approved, deprecated. Claude vet vad som gäller.
- **tags:** nyckelord för filtrering. Claude kan söka på [bokslut, 2025] och få alla relevanta filer.
- **system:** vilket bolag eller system dokumentet tillhör.

Steg 4: Skapa registret

Alla etiketter samlas i en central fil: **registry.yml**. Det är dokumentens telefonkatalog. Claude läser registret, filtrerar på typ och taggar, och får en lista med sökvägar till rätt filer. På millisekunder.

För dig som människa är registret mest en bonus. Men för AI:n är det *allt*. Utan registret måste den öppna varje fil. Med registret behöver den bara läsa en enda fil för att få en komplett bild.

Vad jag dokumenterar, och var

Här är de viktigaste kategorierna. Du behöver inte starta med alla:

Mapp	Innehåll	Exempel
decisions/	Affärsbeslut med kontext	Val av regelverk

policies/	Regler som gäller löpande	Lönemodell, prissättning
specs/	Tekniska specifikationer	Hur systemet fungerar
legal/	Avtal, villkor och regelverk	Kundavtal, integritetspolicy
handbook/	Handbok för dagligt arbete	Hur man hanterar en faktura
accounting/	Bokföring och bokslut	Bokslutsprocess, kontoplan

Praktiskt exempel: Bokslut

Låt mig visa hur det ser ut i verkligheten. Jag har byggt en bokslutsdomän för min koncern:

```
accounting/
  principer/                    ← Regler som gäller alla bolag
    redovisningsprinciper.md
    bokslutsprocess.md          ← Steg-för-steg-flöde
  drift-ab/                    ← Per bolag
    bolagskontext.md            ← Fast info om bolaget
    bokslut/
      2025/
        protokoll.md            ← Bokslutsprotokoll med checklista
        bedomningar.md          ← Professionella bedömningar
```

Bokslutsprocessen är dokumenterad som ett **steg-för-steg-flöde** där varje steg har: vad som behövs, vilka regler som gäller, och när jag som människa måste ta över.

> Varje år skapas en ny mapp (bokslut/2026/) medan fjolårets dokumentation bevaras intakt. Det ger Claude historik att jämföra med, precis som en ny revisor tittar på föregående års bokslut.

Vad Claude får göra, och inte

Det viktigaste i agent-first är inte vad AI:n *kan* göra, utan var gränsen går. Jag dokumenterar **eskaleringsregler** för varje process:

- **Bankavstämning:** Claude kan jämföra siffror automatiskt. Hög automation.
- **Avskrivningar:** helt regelbaserat. Hög automation.
- **Osäkra fordringar:** Claude tar fram underlag, jag bedömer. Eskalering.
- **Avsättningar:** alltid mänsklig bedömning. Claude påminner bara.

Dagligt arbete: så lite tid tar det

Om det här ska fungera över tid måste det bli en **naturlig del** av hur du driver bolaget. Inte ett ”extra dokumentationsprojekt”. Här är den viktiga principen:

> **Du dokumenterar inte allt.** Du dokumenterar: beslut som påverkar framtiden, regler som AI:n ska följa, återkommande processer, och undantag som annars sitter i ditt huvud. Allt annat är onödig information.

När du fattar ett beslut

Du ändrar en avgift, tillåter en ny kampanjtyp eller justerar en lönemodell. Direkt efteråt: be Claude skapa en MD-fil, fylla i mallen och lägga till i registret. Det tar en mening från dig och några sekunder för Claude.

När du gör något två gånger

Om du förklarar samma sak för support, rättar samma typ av fel, eller hanterar samma edge case igen: skapa en guide. Det är inte dokumentation för arkivet, det är operativ styrning.

Veckovis: governance-runda (15 min)

Gå igenom alla dokument med status ”draft”. Godkänn, arkivera eller ta bort. Kontrollera att registret är korrekt. Det här håller systemet levande.

Realistisk tidskostnad

20–40 minuter per vecka. Inte mer. Blir det mer gör du för mycket.

Befintlig dokumentation: vad gör du med den?

Det här är där många gör fel. De försöker återskapa hela historien. Det behöver du inte.

Skapa en formell startpunkt

Skapa ett dokument: **DEC-0001: Documentation Baseline**. Det säger: ”Från och med idag är det här repot source of truth. Historik före detta datum är ofullständig.” Det befriar dig från känslan att behöva återskapa allt.

Dokumentera bara det som påverkar framtiden

Ställ tre frågor för varje gammal sak:

1. Påverkar det fortfarande hur jag jobbar?
2. Kan det skapa juridisk risk om det inte är dokumenterat?
3. Behöver Claude förstå det?

Om svaret är nej på alla tre, låt det vara. Skriv om gällande regler *som de gäller idag*, inte som historiska referat. ”Gällande modell är X. Ursprungligt fattat ca 2024.” Det räcker.

> **Viktigt mentalskifte:** Du bygger inte ett arkiv. Du bygger en exekverbar kunskapsbas. Den ska vara aktuell, korrekt och styrande, inte komplett.

Avtal och juridiska dokument

Du behöver inte kopiera hela avtalet. Juridisk text är ofta överdetaljerad för operativ användning, och du vill inte att en supportagent råkar citera juridisk text. Det räcker med en **strukturerad sammanfattning** i en MD-fil plus en referens till PDF-originalet.

Dokumentera inte avslutade avtal eller engångsuppdrag retroaktivt. Fokusera på det som styr verksamheten *idag*.

Governance: håll ordning

Några enkla regler som gör stor skillnad:

- **reviewed-flaggan:** varje dokument har en reviewed-flagga i registret. Jag sätter den till true efter granskning. Claude får *aldrig* sätta reviewed: true själv. Det är min kvalitetsgaranti.
- **status-fältet:** draft → approved → deprecated. Claude filtrerar på ”approved” och får bara gällande dokument.
- **En källa:** granskningsstatus spåras centralt i registry.yml, inte i varje fils frontmatter.
- **Automatiska kontroller:** en CI-pipeline kan stoppa commits som saknar frontmatter eller har fel klassificering. Inte en mänsklig påminnelse, utan en automatisk spärr.

Så här enkelt ser det ut i praktiken

Det här är det viktigaste att förstå: **du behöver inte kunna skriva YAML eller Markdown själv**. Jag skriver vanlig svenska i en konversation med Claude, och den skapar filerna, placerar dem rätt och uppdaterar registret åt mig. Det är det som gör hela skillnaden: att Claude har tillgång till repot och kan se och läsa allt.

Här är ett riktigt exempel från när jag ville dokumentera min bokslutsstruktur:

Jag
Jag ska lägga till en bokföringsdomän i det interna repot. Jag har fyra bolag, alla med Fortnox. Ett bolag kör K3, resten K2. Skapa ett beslutsdokument, grundstruktur med mallar för bokslutsprotokoll per bolag, och uppdatera registret.

Claude
Klart. Jag har skapat: • DEC-2026-0014 (beslutsdokument) • 4 principfiler (redovisningsprinciper, kontoplan, bokslutsprocess, koncernelimineringar) • 4 bolagskontext-filer (en per bolag) • 8 bokslutsfiler (protokoll + bedömningar per bolag för 2025) • Alla 17 poster är tillagda i registry.yml Struktur verifierad, inga saknade filer eller felaktiga ID:n.

Det var allt. En mening från mig, och Claude skapade 17 filer med korrekt frontmatter, mappstruktur och registerposter. När jag sedan gör bokslut säger jag bara: *”Jag börjar med Drift AB, steg 2 bankavstämning. Saldot på banken är 234 500 kr, i Fortnox står det 234 500.”* Claude fyller i protokollet, markerar steget som klart och går vidare till nästa.

> Hela poängen: du pratar som vanligt. Claude sköter formateringen, placeringen och indexeringen. Men det fungerar bara för att strukturen finns på plats.

Var du börjar

Det är lätt att överkomplicera det här. Här är vad jag rekommenderar:

1. **Välj en domän,** inte hela företaget. Börja med det som tar mest tid.
2. **Skapa en mappstruktur.** Det behöver inte vara perfekt från början.
3. **Lägg till frontmatter:** ge varje fil ett ID, en typ och några taggar. 30 sekunder.
4. **Skapa ett register:** en registry.yml som listar alla filer med metadata.
5. **Ge Claude tillgång:** peka Claude mot ditt repo och be den hjälpa dig dokumentera medan du arbetar.

> Börja smått. En mapp, tio filer, ett register. Utöka när du ser att det fungerar. Bättre tio välstrukturerade dokument än hundra ostrukturerade.

Där jag är idag

Jag har byggt strukturen. Jag har 100+ dokument med frontmatter och register: principer, beslut, processer och handbok, allt i Markdown-filer på GitHub med versionshantering.

När jag skrev den första versionen av den här guiden hade jag inga autonoma agenter. Jag använde Claude som en interaktiv kollega: jag pratade, Claude läste mina dokument och hjälpte mig. Det var värdefullt i sig. Sedan dess har jag gått vidare. Idag har jag agenter som kör själva: de hämtar data, sammanställer rapporter, postar i Teams och föreslår åtgärder. Men det hade aldrig fungerat utan grunden som den här guiden beskriver. Det är **det** som är den stora skillnaden. Agenterna läser fortfarande samma Markdown-filer och navigerar via samma register.

Det jag lärde mig under implementationen var att dokumentationen behövde kompletteras med ett operativt lager: behörigheter, tillstånd, minne och godkännandeflöden. Det är ämnet för del 2.

Hur det känns efter tre månader

Jag tänker tydligare innan beslut. Jag slipper förklara samma sak igen. Jag får konsekvens och intern spårbarhet. Och jag kan bygga vidare på detta utan att börja om.

Vad kommer i del 2?

I **del 2: Från dokumentation till autonoma agenter** beskriver jag vad som hände när jag faktiskt byggde agenter på den här grunden. Bland annat:

- Varför dokumentation ensamt inte räcker: agenter behöver permissions, state och memory
- Skillnaden mellan att en agent läser en eskaleringsregel och att den följer en
- Hur du behåller kontrollen utan att vara flaskhals: godkännandeflöden i Teams
- Praktiska exempel från min marknadsföringsagent som kör själv varje vecka

Del 2 publiceras inom kort. Prenumerera på nyhetsbrevet så får du den direkt.
