DEN FULLSTÄNDIGA GUIDEN

GitHub för marknadsförare 2027

Marknadsplanen blir en .md-fil. Så jobbar ett AI-native marknadsteam tillsammans i GitHub, och så klär du din Claude med smak, skills och processer.

Axel Tuvemo · 26 november 2026

Originalguiden i sin helhet. Den kortare artikeln finns på axeltuvemo.com.

Varför den här guiden finns

De flesta marknadsteam jobbar fortfarande som 2015. Planen ligger i en slidedeck. Tonalitetsguiden är en PDF ingen hittar. Bästa prompten bor i någons privata ChatGPT-historik. Varje gång du ska be en AI om hjälp börjar du om från noll och klistrar in samma bakgrund igen.

Det fungerar inte när AI gör en allt större del av jobbet. AI behöver kontext. Teamet behöver en gemensam sanning. Båda problemen har samma lösning, och utvecklare har använt den i snart tjugo år: textfiler i ett delat repo, med versionshistorik och granskning.

Min tes är enkel. 2027 är marknadsplanen en .md-fil. Alla i teamet, och alla AI-agenter teamet använder, läser från samma fil. Ändringar föreslås, granskas och godkänns. Ingen jobbar längre från en gammal version.

Den här guiden förklarar vad GitHub är, hur det fungerar och exakt hur ett marknadsteam använder det. Du behöver inte kunna koda.

Del 1. Vad är GitHub?

Git och GitHub är inte samma sak

Git är ett versionshanteringssystem. Det håller koll på varje ändring i en mapp med filer: vem ändrade, vad, när och varför. Du kan alltid gå tillbaka till en tidigare version.

GitHub är en webbtjänst (ägd av Microsoft) där Git-mappar ligger online. Ovanpå Git lägger GitHub allt som gör samarbete möjligt: granskning, kommentarer, uppgifter, projekttavlor, behörigheter, automatisering och AI.

En bra liknelse: Git är "spåra ändringar" i Word, fast för en hel mapp. GitHub är Google Drive byggt runt den funktionen, med inbyggd godkännandeprocess.

Hur stort är det?

GitHubs egen årsrapport Octoverse 2025 visar att plattformen har över 180 miljoner utvecklare, att mer än 36 miljoner gick med på ett år och att det finns 630 miljoner projekt och 4,3 miljoner AI-projekt på plattformen. Nästan 80 procent av nya utvecklare använder Copilot redan första veckan.

På GitHub Universe 2026 var budskapet tydligt: GitHub vill bli plattformen där även icke-utvecklare skapar, via agenter, gränssnitt, kommandorad och mobil. Det är alltså inte vi marknadsförare som tränger oss in. Dörren öppnas inifrån.

Varför textfiler vinner

Allt i den här guiden bygger på en enda insikt: AI läser text bäst. En Markdown-fil (.md) är ren text med enkel formatering. Den är lätt för människor att läsa och perfekt för en språkmodell att förstå. En slidedeck eller en PDF är det motsatta.

Markdown på 30 sekunder: en rad som börjar med # är en rubrik och ## en underrubrik. Text mellan två dubbla stjärnor blir fet. En rad som börjar med ett bindestreck blir en punkt i en lista. En länk skrivs som [länktext](https://exempel.se).

Det är i princip allt du behöver kunna.

Del 2. Grundbegreppen, översatta till marknadsspråk

Repository (repo): en projektmapp med full historik. Marknadsmotsvarighet: teamets gemensamma marknadsmapp.

Fil (.md): ett textdokument. Marknadsmotsvarighet: marknadsplanen, en brief, en persona.

Commit: en sparad ändring med kommentar. Marknadsmotsvarighet: "Uppdaterade Q1-budget efter ledningsmöte".

Branch (gren): en egen kopia där du jobbar ostört. Marknadsmotsvarighet: ditt utkast innan det blir officiellt.

Pull request (PR): en begäran om att få in din ändring. Marknadsmotsvarighet: "Kan någon granska min nya kampanjbrief?"

Review: granskning och kommentarer på en PR. Marknadsmotsvarighet: copy-granskning, juridisk check, chefsgodkännande.

Merge: när ändringen godkänns och blir officiell. Marknadsmotsvarighet: nu gäller den nya versionen för alla.

Issue: en uppgift, idé eller ett problem. Marknadsmotsvarighet: ett ärende på kampanjtavlan.

Projects: kanban- eller tabellvy över issues. Marknadsmotsvarighet: content-kalender, kampanjplanering.

README.md: startsidan i ett repo. Marknadsmotsvarighet: "Läs detta först" för nya i teamet.

Actions: automatiska flöden som körs vid händelser. Marknadsmotsvarighet: "När briefen godkänns, skapa uppgifter".

Pages: gratis hosting av en webbsida från ett repo. Marknadsmotsvarighet: kampanjsida, intern wiki, landningssida.

Fork: en egen kopia av någon annans repo. Marknadsmotsvarighet: ta en färdig skill-samling och anpassa den.

Det viktigaste flödet att förstå är branch → commit → pull request → review → merge. Det är samma sak som ett bra godkännandeflöde i ett marknadsteam, bara spårbart och automatiserat.

Del 3. Varför GitHub blir marknadsteamets hem 2027

Problemet: kontexten är utspridd

I ett vanligt team bor kunskapen på tio ställen: Drive, Notion, Slack-trådar, någons huvud, någons ChatGPT. När AI ska hjälpa till måste någon samla ihop kontexten manuellt varje gång. Resultatet blir generiskt.

Lösningen: kontext som filer

När ICP, positionering, tonalitet och plan ligger som .md-filer i ett repo kan AI-verktyg som Claude Code och OpenAI Codex koppla direkt mot repot och läsa allt själva. Du slutar klistra in bakgrund. Varje agent utgår från samma sanning.

CXL beskriver exakt den här modellen: prompter, briefer, ICP-profiler och tonalitetsdokument lagras i ett repo så att AI kan läsa kontexten direkt i stället för att få den förklarad i varje chatt. Poängen är att det ackumuleras. En skill blir bättre för varje körning, uppdaterad kontext skärper allt nedströms och vem som helst i teamet kan läsa, köra och förbättra systemet.

Tre skäl som spelar roll för en marknadschef

1. En sanning. Planen finns i en version. Alla, människor och agenter, läser samma fil.

2. Spårbarhet. Varje beslut har en commit med vem, när och varför. Du ser exakt när budgeten ändrades och av vem.

3. Hastighet utan kaos. Fem personer kan jobba parallellt i egna branches. Granskning sker i PR:en, inte i en mejltråd med "final_v3_RIKTIG".

Del 4. Marknadsplanen som .md-fil

Det här är kärnan. Marknadsplanen slutar vara en deck som presenteras en gång och glöms. Den blir ett levande dokument i repots rot, som alla utgår från.

Ett exempel på hur marknadsplan.md kan vara uppbyggd, rubrik för rubrik:

Överst: rubriken Marknadsplan 2027, med raderna Ägare: Marknadschef och Senast godkänd: 2027-01-15.

Mål: 1 200 MQL under året och 30 % av pipeline från marknadskanaler.

Målgrupp: Se /context/icp.md. Positionering: Se /context/positionering.md.

Prioriterade initiativ, med kvartal, initiativ, ägare och budget: Q1, lansering produkt X, Anna, 400 tkr. Q2, mässa Y, Erik, 250 tkr.

Kanaler och fördelning. Mätning: KPI:er och var de följs upp.

Beslutslogg: 2027-02-03: Flyttade 100 tkr från LinkedIn till sök. Skäl: CPL.

Notera tre saker.

Planen länkar till kontextfiler i stället för att upprepa dem. Ändrar du ICP:n ändras den överallt.

Det finns en beslutslogg. Git sparar ändå historiken, men en mänsklig sammanfattning gör det läsbart.

Den är skriven så att en AI-agent kan läsa den och direkt veta vad som prioriteras. Be Claude "skriv tre annonsvarianter för Q1-initiativet" och den hittar mål, målgrupp och tonalitet själv.

Ändringar i planen görs alltid via pull request och godkänns av planens ägare. Då vet alla att det som ligger i main är det som gäller.

Del 5. Så bygger du teamets repo

Ett förslag på mappstruktur, inspirerat av hur AI-native team bygger idag. Allt ligger i en mapp som heter marknad/.

I roten: README.md (läs detta först: vad repot är och hur vi jobbar), CLAUDE.md (instruktioner till AI: vilka vi är, regler, var saker ligger) och marknadsplan.md (den gemensamma sanningen).

context/: icp.md, personas/, positionering.md, tonalitet.md, budskapsramverk.md och konkurrenter.md.

skills/: annonsvarianter/SKILL.md, seo-brief/SKILL.md och nyhetsbrev/SKILL.md.

agents/: kampanjlansering.md.

kampanjer/: en mapp per kampanj, till exempel 2027-q1-lansering/ med brief.md, budskap.md och resultat.md.

content/: till exempel blogg/. outputs/: där agenterna skriver sina resultat.

CLAUDE.md (eller motsvarande AGENTS.md) är filen AI:n läser först. Den förklarar repots upplägg, vad varje mapp innehåller och vilka konventioner som gäller.

Skills är fristående instruktioner för en uppgift, till exempel att skriva annonsvarianter eller granska en landningssida. Varje skill har en egen mapp med en SKILL.md som anger indata och utdata.

Agents är filer som kedjar ihop skills i en ordning. En agent innehåller inga egna prompter, så samma skill kan återanvändas av flera agenter.

Ett konkret exempel från CXL: en demand gen-marknadsförare lägger ICP, budskapsramverk och en bedömningsmall för annonser i /context. En agent kör tre skills i ordning. Den första tar fram vinklar från ICP:n, den andra gör annonsvarianter och den tredje poängsätter dem mot mallen. Resultatet hamnar i /outputs och marknadsföraren granskar.

Del 6. När använder marknadsföraren GitHub?

Allt ska inte in i GitHub. Här är min tumregel.

Använd GitHub för: marknadsplan och strategi. ICP, personas, positionering och tonalitet. Kampanjbriefer och budskap. Prompter, skills och agenter. Blogginlägg och webbinnehåll på sajter byggda i till exempel Astro eller Next.js. Content-kalender och kampanjuppgifter (Issues och Projects). Beslutsloggar och retros. Mallar för mejl och landningssidor.

Använd något annat för: snabba chattar (Slack eller Teams). Bilder, video och tunga designfiler (DAM, Figma, Canva). Rådata från annonsplattformar (BI-verktyg). Kunddata och personuppgifter (CRM). Kontrakt och avtal. Allt som innehåller lösenord eller API-nycklar.

Growth Method listar fler ställen där marknadsförare redan rör GitHub utan att tänka på det: webbplatser byggda i Astro, Next.js eller Hugo lagrar sidor och inlägg som filer i repon, hjälpcenter i GitBook eller Mintlify synkar mot GitHub, och designsystem ligger ofta i ett repo.

Del 7. Ett vanligt arbetsflöde, steg för steg

Scenario: du ska uppdatera tonalitetsguiden efter en ny varumärkesplattform.

1. Öppna repot på github.com. 2. Öppna filen context/tonalitet.md. 3. Klicka på pennan för att redigera. 4. Gör dina ändringar.

5. Välj "Create a new branch" när du sparar. Ge branchen ett tydligt namn, till exempel tonalitet-ny-plattform. 6. Skriv en kort commit-kommentar. Förklara varför du ändrar.

7. Klicka "Propose changes". GitHub skapar en pull request. 8. Lägg till en granskare, till exempel varumärkesansvarig. 9. Granskaren läser ändringarna. Granskaren kommenterar eller godkänner.

10. Klicka "Merge". Nu gäller den nya tonaliteten för alla, och för alla AI-agenter som läser repot.

Samma flöde med AI: be Claude Code (kopplad till repot) "uppdatera tonalitetsguiden enligt den nya plattformen i bifogad fil och öppna en pull request". Agenten gör ändringen. Du granskar. Människan godkänner alltid.

Issues och Projects som kampanjtavla

1. Skapa en issue per uppgift: "Skriv LinkedIn-annonser för Q1-lanseringen". 2. Tilldela en ägare och sätt etiketter som kanal:linkedin och q1.

3. Lägg alla issues i ett Project med kolumnerna Idé, Pågår, Granskning, Klart. 4. Koppla pull requests till issues. När PR:en mergas stängs issuen automatiskt.

Då hänger uppgift, leverans och godkännande ihop på ett ställe.

Del 8. Regler som gör det hållbart

Utan regler blir ett repo lika rörigt som en delad Drive. Fem regler räcker långt.

1. Main är heligt. Ingen ändrar direkt i main. Allt går via pull request. Slå på "branch protection" i inställningarna.

2. Varje viktig fil har en ägare. Använd en CODEOWNERS-fil så att rätt person automatiskt blir granskare. Marknadschefen äger planen. Varumärkesansvarig äger tonaliteten.

3. Commit-kommentarer förklarar varför. "Uppdaterade" säger ingenting. "Sänkte CPL-mål efter Q1-utfall" säger allt.

4. En fil, en sak. Hellre tio små filer som länkar till varandra än ett monster.

5. Inga hemligheter i repot. Inga lösenord, API-nycklar eller kunddata. Använd GitHub Secrets för nycklar som automatiseringar behöver.

Del 9. Verktygen

Du behöver inte terminalen för att börja. Välj nivå efter mognad.

Nybörjare: github.com i webbläsaren, för att läsa, redigera filer, granska PR:er och hantera issues. GitHub Desktop, för att jobba med filer lokalt utan kommandorad.

Mellan: Obsidian eller VS Code med Git, för att skriva och läsa .md-filer bekvämt.

AI-native: Claude Code, OpenAI Codex eller Cursor, för att låta AI läsa repot, köra skills och öppna PR:er. GitHub MCP-servern, för att koppla AI-verktyg direkt mot issues, PR:er och projekt.

Avancerad: GitHub Actions, för att automatisera flöden, till exempel en morgonrapport eller kvalitetskontroll.

GitHub MCP-servern är GitHubs officiella koppling för AI-verktyg som Claude Desktop, Cursor och VS Code. Via den kan en agent läsa filer, hantera issues, skapa pull requests och följa projekt. Uppdateringen i januari 2026 samlade projektverktygen och halverade ungefär hur mycket kontext de tar.

Färdiga repon att börja från

Du behöver inte bygga allt själv. Forka en befintlig samling och anpassa den till din kontext.

Corey Haines Marketing Skills: CRO, copy, SEO, analys och growth. Över 26 000 stjärnor.

Michael Sitarzewskis agency agents: skills och agenter för marknad, projektledning och design. Över 90 000 stjärnor.

Forka, byt ut kontexten mot din egen och testa en skill i taget.

Del 10. GitHub klär din Claude med smak, skills och processer

En Claude utan kontext är en skicklig frilansare på första dagen. Den kan allt i teorin men vet ingenting om er. Den skriver generiskt, väljer säkra formuleringar och gissar vad ni tycker är bra. Problemet är inte modellen. Problemet är att den inte har på sig era kläder.

GitHub är garderoben. Där ligger det som gör Claude till er Claude, och det består av tre lager.

Lager 1. Smak

Smak är det svåraste att förklara och det lättaste att tappa. Det är skillnaden mellan en text som är korrekt och en text som låter som ni.

Smak skrivs ner i filer som tonalitet.md, ordlista.md och exempel/bra.md och exempel/dåligt.md. Den sista filen är underskattad. Visa Claude tio texter ni är stolta över och tio ni skäms över, med en rad om varför. Det lär modellen mer än tre sidor adjektiv.

Smak är också vad ni inte gör. Ord ni aldrig använder. Påståenden ni aldrig gör. Tonlägen som inte är ni. En kort lista med förbud är ofta det som lyfter texten mest.

Lager 2. Skills

En skill är en mapp med en SKILL.md: en återanvändbar instruktion för en uppgift, med indata, steg och hur ett bra resultat ser ut. Formatet är öppet och fungerar i flera AI-verktyg, inte bara Claude.

Här blir GitHub avgörande. Claude Code behandlar ett GitHub-repo som en plugin-marknadsplats. Teamet lägger till repot en gång med /plugin marketplace add organisation/repo och installerar sedan era skills. När någon förbättrar en skill i repot får alla den nya versionen.

Det löser ett problem som nästan alla team har idag: att den bästa prompten skickas runt i Slack och finns i fem olika versioner. Med ett repo finns en version, och den blir bättre för varje pull request.

Tumregeln är enkel. En skill du bara testar själv kan ligga lokalt. En skill som teamet ska använda hör hemma i repot, versionerad och granskad.

Du behöver inte börja från noll. Det finns redan publika marknadsrepon med många skills för till exempel SEO, content, outbound och analys, och Anthropic har ett officiellt skill-repo. Men läs alltid igenom en skill innan du installerar den. Kontrollera vad den gör med filer, nätverk och inloggningar.

Lager 3. Processer

Smak och skills gör enskilda svar bättre. Processer gör teamet bättre. Det är här GitHubs egna funktioner kommer in.

Pull requests gör att ingen ändring i tonalitet, plan eller skill blir officiell utan granskning. CODEOWNERS gör att rätt person granskar automatiskt.

Issues och Projects gör att Claude kan läsa vad som ska göras och rapportera vad som är klart. Actions gör att saker händer utan att någon minns det, till exempel att en skill testas varje gång den ändras.

Historiken gör att ni kan se varför en skill ändrades och gå tillbaka om den blev sämre.

Därför är det en konkurrensfördel

Alla har tillgång till samma modeller. Det som skiljer ett team från ett annat 2027 är vad de har lärt sin AI. Den kunskapen ackumuleras bara om den sparas, delas och förbättras på ett ställe. Ett marknadsteam utan repo börjar om varje måndag. Ett team med repo blir bättre varje vecka.

Gör det här konkret i morgon: 1. Skapa mappen skills/ i teamets repo. 2. Välj den prompt som teamet använder mest. 3. Skriv om prompten till en SKILL.md med syfte, indata, steg och exempel. 4. Lägg till filen via en pull request. 5. Be en kollega köra skillen och föreslå en förbättring.

Del 11. Vad praktikerna säger (Reddit, Hacker News och bloggar)

Jag har gått igenom diskussioner på Reddit, Hacker News och dev.to, plus marknadsförare som skriver om sina egna upplägg. Bilden är tydlig och mer nyanserad än i de entusiastiska bloggarna.

Det som fungerar. I r/ClaudeCode och r/DigitalMarketing beskriver content- och marknadsplanerare att Claude Code med skills plötsligt känns logiskt, men att det tar tid att vänja sig vid Markdown. I r/SideProject sprids repon med dussintals Markdown-filer fulla av marknadsramverk som installeras i Claude Code. Och i r/ClaudeWorkflows beskrivs upplägg där Git, strukturerad Markdown och validering tillsammans blir ett stabilt minne för Claude.

Det som skaver. Den vanligaste klagan gäller Git självt. I r/technicalwriting beskriver personer som är nya på "docs as code" att de hatar det, och att det svåraste var rebase. I r/learnprogramming och r/unpopularopinion återkommer samma sak: Git är svårt att lära sig för den som inte är tekniskt, även om det fungerar bra när man väl kan det. Recensioner på Capterra säger samma sak: att få med icke-utvecklare i flödet är en av GitHubs nackdelar.

Från Hacker News. Utvecklarna är överlag positiva till att samla dokument i repot, och en del menar att det varit praxis i åratal. Ett team berättar att de flyttade in dokumentationen i samma repo som koden och lade en rad i sin CLAUDE.md om att alltid uppdatera dokumenten. Nu kan de be Claude hitta allt som är inaktuellt. Men samma trådar visar baksidan. Flera beskriver att Git plus Markdown fungerar dåligt när icke-tekniska kollegor ska bidra, och att en marknadschef som måste ha Git-åtkomst för att ändra en fil helt enkelt skriver mindre. En annan påpekar att stora företag har dokumenten i Confluence eller SharePoint, eftersom budget och godkännanden

kräver att icke-tekniska personer kan kommentera. Och det finns en varning om "markdown-djungel": filer som ingen städar blir brus för både människor och AI.

Från marknadsförare som gör det. Travis Wilson har allt sitt marknadsmaterial i en mapp i repot: tonalitet, daglig checklista, konkurrentanalys och loggar per kanal. En egen Claude-skill läser in allt när sessionen startar och ger honom en dagordning. Hans poäng är att det finns en enda källa och att en ny Claude-session är uppdaterad på några sekunder. Jonathan Martinez, som jobbade i Postmates tidiga growth-team, beskriver hur informationen då låg utspridd i Docs, Sheets, Tableau, Amplitude och Dropbox. Med ett repo och Claude Code kan teamet ställa frågor över allt material, till exempel jämföra säljsamtal med budskapen på webben. Han medger själv att upplägget kan kännas tekniskt.

Vad jag drar för slutsats.

Låt inte marknadsförare lära sig Git från terminalen. Börja i webbläsaren eller i GitHub Desktop.

Undvik avancerade Git-moment helt. Ett marknadsteam behöver branch, pull request och merge. Inte rebase.

Låt AI ta det tekniska. Claude Code kan skapa branch, commit och pull request åt dig. Människan läser och godkänner.

Räkna med en inlärningströskel på några veckor. Planera för den i stället för att låtsas att den inte finns.

Del 12. Kostnad, säkerhet och alternativ

Vad kostar det?

GitHub har tre huvudnivåer: Free, Team för 4 dollar per användare och månad, och Enterprise från 21 dollar per användare och månad. Ett marknadsteam på tio personer på Team-planen kostar alltså runt 40 dollar i månaden.

Välj Team om ni vill ha reglerna i den här guiden på riktigt i privata repon. Team-planen innehåller bland annat skyddade branches, obligatoriska granskare och CODEOWNERS. Observera att Copilot inte ingår i Team-planen.

Säkerhet

Håll repot privat. Marknadsplaner och budgetar är affärskritiska.

Lägg aldrig in kunddata, personuppgifter, lösenord eller API-nycklar. Det är både en säkerhetsrisk och en GDPR-fråga.

Använd organisationskonton, inte privata konton. Då kan ni ta bort behörighet när någon slutar. Kräv tvåstegsverifiering för alla i organisationen.

Granska tredjeparts-skills innan ni installerar dem. En skill kan innehålla skript.

Prata med IT om var datan lagras och vilken plan som krävs för era krav. Enterprise ger mer kontroll om ni behöver det.

Varför inte bara Notion eller Google Drive?

Notion och Drive är bättre för att skriva tillsammans i realtid och för kollegor som aldrig vill se en fil. GitHub är bättre när tre saker är viktiga: att AI ska läsa kontexten direkt, att varje ändring ska granskas och spåras, och att skills ska delas och versioneras.

Mitt råd är att inte välja det ena. Låt det levande skrivandet och mötesanteckningarna bo där de bor. Flytta in det som är kontext och regler i GitHub: plan, ICP, positionering, tonalitet och skills. Det är det som AI och teamet ska utgå ifrån.

Svaga punkter att vara ärlig om

Bilder, video och PDF:er passar dåligt i Git. Länka till dem i stället.

Två personer som ändrar samma rad samtidigt får en mergekonflikt. Håll filerna små så händer det sällan.

Det kräver disciplin. Ett repo som ingen äger blir lika rörigt som en delad Drive.

Del 13. Kom igång på 30 dagar

Vecka 1. Grunden

Skapa en GitHub-organisation för teamet och ett privat repo, marknad. Skriv README.md: vad repot är, hur vi jobbar, vem som äger vad. Flytta in tre filer: ICP, positionering, tonalitet.

Vecka 2. Planen

Skriv om marknadsplanen till marknadsplan.md. Slå på branch protection och sätt upp CODEOWNERS. Låt varje person i teamet göra sin första pull request, även om den bara rättar ett stavfel.

Vecka 3. AI

Skriv CLAUDE.md. Gör om teamets mest använda prompt till en SKILL.md. Koppla Claude Code eller Codex mot repot och kör skillen.

Vecka 4. Flödet

Flytta nästa kampanj till Issues och Projects. Bygg en första agent som kedjar två eller tre skills. Håll en retro: vad sparade tid, vad var krångligt?

Del 14. Fallgropar

Allt ska in direkt. Börja med kontexten och planen. Resten följer.

Bara en person kan systemet. Då har du bytt en flaskhals mot en annan. Alla ska kunna göra en PR.

AI mergar själv. Låt agenter föreslå. Låt människor godkänna.

Filerna ruttnar. Utse ägare och lägg in en kvartalsvis genomgång av context-mappen.

GitHub blir ännu ett verktyg. Det ersätter delar av Drive och Notion för strategi och kontext. Det ersätter inte Slack, Figma eller CRM.

Ordlista

Branch: en parallell version där du jobbar utan att påverka huvudversionen. Commit: en sparad ändring med en kommentar. CODEOWNERS: en fil som anger vem som äger och granskar vilka filer. Fork: en egen kopia av någon annans repo.

Issue: en uppgift, idé eller ett problem. Main: huvudversionen, det som gäller. Markdown (.md): enkel textformatering som både människor och AI läser bra. MCP: Model Context Protocol, ett standardsätt att koppla AI-verktyg till andra system.

Merge: när en ändring godkänns och blir en del av main. Pull request (PR): en begäran om att få in en ändring, med granskning. Repo: en mapp med filer och hela deras historik. Skill: en återanvändbar instruktion till AI för en specifik uppgift.

Vanliga frågor

Vad är skillnaden på Git och GitHub?

Git är ett versionshanteringssystem som håller koll på varje ändring i en mapp med filer. GitHub är en webbtjänst där Git-mappar ligger online, med granskning, uppgifter, behörigheter, automatisering och AI ovanpå.

Behöver marknadsförare kunna koda för att använda GitHub?

Nej. Du kan börja i webbläsaren eller i GitHub Desktop. Ett marknadsteam behöver branch, pull request och merge, och AI-verktyg som Claude Code kan sköta det tekniska medan människan granskar och godkänner.

Vad kostar GitHub?

GitHub har tre huvudnivåer: Free, Team för 4 dollar per användare och månad, och Enterprise från 21 dollar per användare och månad. Team-planen ger skyddade branches, obligatoriska granskare och CODEOWNERS i privata repon.

Ska vi byta ut Notion mot GitHub?

Nej, välj inte det ena. Låt levande skrivande och mötesanteckningar bo kvar i Notion eller Drive. Flytta in kontext och regler i GitHub: plan, ICP, positionering, tonalitet och skills.

Vad är en skill?

En skill är en mapp med en SKILL.md: en återanvändbar instruktion till AI för en specifik uppgift, med indata, steg och hur ett bra resultat ser ut. Läggs den i teamets repo blir den versionerad, granskad och delad.

Källor och vidare läsning

Octoverse 2025

https://github.blog/news-insights/octoverse/octoverse-a-new-developer-joins-github-every-second-as-ai-leads-typescript-to-1/

GitHub Universe 2026: lowering the floor and raising the ceiling

https://www.forrester.com/blogs/github-universe-2026-lowering-the-floor-and-raising-the-ceiling/

GitHub for marketing

https://cxl.com/blog/github-for-marketing-ai-workflows/

GitHub MCP Server

https://growthmethod.com/github-mcp-server/

The Marketer's Guide to Working With GitHub

https://getmarketingwithai.substack.com/p/the-marketers-guide-to-github

GitHub-priser 2026

https://capitalandcompute.net/blog/github-pricing-plans-cost/

Claude Code skills på GitHub

https://allthings.how/claude-code-skills-on-github-how-to-find-install-and-build-them/

När en skill blir ett plugin

https://implicator.ai/the-best-github-tools-for-building-claude-code-skills/

Officiella skills

https://github.com/anthropics/skills

r/ClaudeCode

https://www.reddit.com/r/ClaudeCode/comments/1ou5570/as_a_digital_marketing_and_content_planner_claude/

r/ClaudeWorkflows

https://www.reddit.com/r/ClaudeWorkflows/comments/1v7pys4/workflow_claude_code_marketing_workflow_gitbacked/

r/technicalwriting

https://www.reddit.com/r/technicalwriting/comments/1nwb06w/new_to_docs_as_code_and_hating_it/

It's time to move your docs in the repo

https://news.ycombinator.com/item?id=47380231

How do you organize documentation at work

https://news.ycombinator.com/item?id=39370226

Marketing as Code

https://dev.to/traviticus/marketing-as-code-why-my-brand-docs-live-in-git-4efd

Future marketing teams will live in GitHub

https://jon4growth.substack.com/p/future-marketing-teams-will-live

GitHub-recensioner

https://www.capterra.com/p/129067/GitHub/