

iTerm2 Shell Integration — van de hoed en de rand

Volledige analyse van de officiële iTerm2-documentatie. Bronnen die ik echt geopend heb: de pagina Shell Integration, plus de sub-pagina's Utilities, Badges, Triggers, Automatic Profile Switching, Status Bar, Proprietary Escape Codes, tmux Integration en Claude Code Integration (allemaal op iterm2.com).
Alles wat hieronder staat komt daaruit, of uit een meting op jouw eigen MacBook.

In het kort (de 7 dingen die tellen)

1. **Shell Integration is een klein scriptje in je zsh-startbestand.** Meer is het niet.
Het stuurt onzichtbare codes naar iTerm2 met: waar je prompt begint, waar je commando eindigt, of het commando gelukt is, wie je bent, op welke computer je zit en in welke map.
2. **iTerm2 kan daardoor je terminal "lezen" in plaats van alleen tekst tonen.**
Elk commando wordt een blokje met een begin, een eind en een uitslag.
3. **Blauw driehoekje links = een mark.** Springen met Cmd-Shift-Omhoog en Omlaag.
Mislukt commando kleurt het driehoekje rood. Rechtsklikken toont de foutcode.
4. **Je krijgt gratis:** commando-geschiedenis per server, veelgebruikte mappen, bestanden downloaden met een rechtermuisklik, uploaden door te slepen, en een alarm als een lang commando klaar is.
5. **Voor jou is het interessantste deel niet Shell Integration zelf, maar wat eromheen hangt:** de **Session Status**-lijst in de toolbelt, die per tab laat zien of Claude **bezig** is, op jou wacht of klaar is.
6. **Op jouw MacBook staat Shell Integration nu NIET aan.** Ik heb het gemeten:
er is geen `~/ .iterm2_shell_integration.zsh` en je `~/ .zshrc` noemt iTerm2 nergens.
Je iTerm2 is versie **3.7.0**, dus alles uit dit verslag kan gewoon.
7. **Aan- en uitzetten is klein en omkeerbaar.** Eén regel in `~/ .zshrc` erbij, of die regel weer weghalen. Geen installatie diep in je systeem.

1. Wat is het precies

iTerm2 is normaal gesproken dom. Het toont letters die de computer stuurt. Het weet niet wat een prompt is, wat een commando is en wat uitvoer is. Voor iTerm2 is dat allemaal dezelfde soep van tekens.

Shell Integration lost dat op. Je shell (bij jou: **zsh**) gaat zelf melden wat er gebeurt.

De documentatie noemt dit letterlijk zo. Shell Integration werkt door je shell op elke computer waar je inlogt zo in te stellen dat hij speciale codes stuurt met deze informatie:

- waar de prompt begint en eindigt
- waar het commando eindigt en de uitvoer begint
- de foutcode van het laatste commando
- je gebruikersnaam
- de naam van de computer
- de huidige map

Het werkt met **bash**, **fish** (2.3 en nieuwer), **tcsh**, **xonsh** (iTerm2 3.6.10 en nieuwer) en **zsh**.

2. Hoe het technisch werkt

De codes zelf

De basis komt van een oude terminal die **FinalTerm** heette. Die is verdwenen, maar zijn codes leven voort. iTerm2 gebruikt ze nog steeds. De documentatie noemt ze **FTCS**-codes.

CODE	WAT HET BETEKENT
OSC 133 ; A ST	vlak voor het begin van de prompt
OSC 133 ; B ST	vlak na het eind van de prompt, dus waar jij begint te typen
OSC 133 ; C ST	vlak voor het begin van de uitvoer van het commando
OSC 133 ; D ; [code] ST	het commando is klaar, met de foutcode erbij (0 tot 255)

Bij die laatste geldt: **0 is gelukt, alles wat niet 0 is, is mislukt**. De terminal mag dat zichtbaar maken. Dat is precies waarom je mark rood kleurt bij een fout.

Daarbovenop heeft iTerm2 zijn eigen codes. Die beginnen met OSC 1337 .

CODE	WAT HET DOET
OSC 1337 ; RemoteHost=gebruiker@computer ST	meldt wie je bent en waar je zit
OSC 1337 ; CurrentDir=[map] ST	meldt in welke map je staat
OSC 1337 ; SetMark ST	zet een mark, hetzelfde als Cmd-Shift-M
OSC 1337 ; SetUserVar=[naam]=[base64] ST	zet je eigen variabele die iTerm2 kan tonen
OSC 1337 ; ShellIntegrationVersion=[nummer] ; [shell] ST	meldt welke versie van het script draait
OSC 1337 ; StealFocus ST	haalt iTerm2 naar de voorgrond
OSC 1337 ; RequestAttention=[waarde] ST	laat het dock-icoon springen
OSC 9 ; [tekst] ST	stuurt een macOS-melding
OSC 1337 ; SetBadgeFormat=[base64] ST	zet de badge (grote tekst rechtsboven)

ST is gewoon het teken dat de code afsluit. Je ziet deze codes nooit op je scherm. iTerm2 vangt ze af en toont ze niet.

Over ShellIntegrationVersion : iTerm2 weet zelf welke versie "de huidige" is. Ziet hij een lager nummer, dan krijg je de vraag om bij te werken.

Wat het script in je shell precies doet

Het script hangt zich aan je prompt. Bij zsh en bash betekent dat: het bemoeit zich met je PS1 (bash) of je prompt-functie. Daarom staat er in de handleiding uitdrukkelijk dat je de source -regel **onderaan** je startbestand moet zetten. Andere scripts kunnen anders overschrijven wat Shell Integration nodig heeft.

Heb je een prompt over meerdere regels en wil je de mark op een andere plek?
Dan zet je dit zelf in je prompt:

- zsh: `%{$(iterm2_prompt_mark)%}`
- bash: `\[ $(iterm2_prompt_mark) \]`
- fish: `iterm2_prompt_mark` ergens in je `fish_prompt` -functie

- xonsh: `$PROMPT = '{iterm2_prompt_start}' + $PROMPT + '{iterm2_prompt_end}'`
- tcsh: **niet mogelijk**, deze functie bestaat daar niet

Belangrijk voor jou als je ooit een zsh-thema gebruikt dat je prompt zelf regelt (Powerlevel10k, Oh My Zsh en dat soort dingen): dan mag Shell Integration niet aan je prompt zitten. Zet dan `export ITERM2_SQUELCH_MARK=1` voor de source-regel, en plaats de mark zelf via je thema.

3. Wat je er concreet aan hebt

Marks: terugspringen in je scherm

- Bij elke prompt zet iTerm2 automatisch een **mark**: een klein blauw driehoekje in de linkermarge.
- Navigeren doe je met **Cmd-Shift-Omhoog** en **Cmd-Shift-Omlaag**.
- Heb je Auto Composer aan, dan is de mark geen driehoekje maar een streep boven de regel.

Status van je commando's

- De mark wordt rood als een commando mislukt.
- Rechtsklik op de mark toont de foutcode.
- Via het menu kun je de **uitvoer van het laatste commando in één keer selecteren**.
- Je kunt per commando uit het verleden opvragen: foutcode, in welke map het draaide, hoe lang het duurde en meer.

Alarm als een lang commando klaar is

Dit is de functie die voor jou het meest voor de hand ligt.

Je start iets dat lang duurt. Dan kies je **Edit > Marks and Annotations > Alert on next mark (Cmd-Opt-A)**. Je gaat iets anders doen. Zodra de prompt terugkomt, krijg je een venster in beeld dat je erop wijst dat het klaar is.

Commando-geschiedenis

- iTerm2 onthoudt je commando's, **apart per gebruiker plus computer**.
- Opzoeken en zoeken: **Session > Open Command History** (Shift-Cmd-;).
- Ze komen ook in Autocomplete (Cmd-;).

- Zet je in de instellingen aan dat de geschiedenis bewaard blijft, dan overleeft die een herstart van iTerm2, tot **200 commando's per gebruiker en computer**.
- Er is een aparte toolbelt-tegel: **Toolbelt > Command History**. Vet gedrukte regels komen uit je huidige sessie. Eén klik scrollt ernaartoe. Dubbelklik typt het commando voor je. Option-dubbelklik typt een `cd` naar de map waar je toen zat.

Veelgebruikte mappen

- iTerm2 onthoudt welke mappen je gebruikt, ook weer per gebruiker plus computer.
- Gesorteerd op wat de documentatie "**frecency**" noemt: hoe vaak én hoe recent.
- Opzoeken: **Session > Open Recent Directories** (Cmd-Opt-/), of via **Toolbelt > Recent Directories**.
- Dubbelklik typt het pad. Option-dubbelklik maakt er een `cd` van.
- Rechtsklik zet er een ster bij. Mappen met een ster staan altijd onderaan de lijst, dus makkelijk terug te vinden.

Bestanden ophalen en wegzetten

- **Downloaden**: rechtsklik op een bestandsnaam (bijvoorbeeld in de uitvoer van `ls`) en kies **Download with scp from [computer]**. Er komt een **Downloads**-menu bij waar je de voortgang ziet.
- **Uploaden**: houd **Option** ingedrukt en sleep een bestand vanuit Finder je terminal in. Het gaat via scp naar de map waar je op dat moment zat op de regel waar je losliet. Er komt een **Uploads**-menu bij.

Badge en offscreen command line

- De **badge** is grote tekst rechtsboven in je sessie, bijvoorbeeld je computernaam of je git-tak. Je stelt hem in bij **Settings > Profiles > General > Badge**, of je zet hem live met een code. Een werkend voorbeeld uit de documentatie:

```
printf "\e]1337;SetBadgeFormat=%s\a" \ $(echo -n "\ (session.name) \
(user.gitBranch)" | base64)
```

- **Offscreen Command Line**: scrollt de uitvoer van een commando boven je scherm uit, dan blijft de bijbehorende prompt bovenaan in beeld staan. Je ziet dus altijd bij welk commando je zit te kijken.

Command Selection

Klik op een commando om het te selecteren. Daarna werken **Find**, **Filter** en **Select All** alleen nog binnen de uitvoer van dat ene commando. Handig als je scherm

vol staat.

Nog een klein maar prettig detail

Shell Integration zorgt dat je prompt **altijd links begint**, ook als het vorige commando niet netjes met een nieuwe regel eindigde.

4. De utilities (het extra gereedschap)

Bij de installatie krijg je de vraag of je ook het **Utilities Package** wilt. Doe dat.

Dat zijn losse commando's die je gewoon kunt aanroepen. De meeste werken zelfs **zonder** Shell Integration.

COMMANDO	WAT HET DOET
<code>imgcat</code>	toont een afbeelding gewoon in je terminal, ook animaties
<code>imgls</code>	een <code>ls</code> met kleine voorbeeldjes van de afbeeldingen
<code>it2</code>	bestuurt iTerm2 vanaf de opdrachtregel: vensters, tabs, splitsingen, tekst sturen, status van een sessie zetten , gebeurtenissen volgen
<code>it2attention</code>	vraagt aandacht: <code>start</code> laat het dock-icoon springen, <code>stop</code> stopt dat, <code>fireworks</code> geeft een explosie bij je cursor
<code>it2check</code>	controleert of je wel echt in iTerm2 zit
<code>it2copy</code>	kopieert tekst naar je klembord, ook via ssh
<code>it2dl</code>	downloadt een bestand van een computer waar je op ingelogd bent
<code>it2getvar</code>	haalt een sessie-variabele op, bijvoorbeeld <code>it2getvar session.name</code>
<code>it2setcolor</code>	verandert de kleuren, ook de tabkleur
<code>it2setkeylabel</code>	labels op de Touch Bar
<code>it2ul</code>	uploadt bestanden
<code>it2universion</code>	zet de unicode-versie, belangrijk voor emoji-breedte

Twee dingen die er voor jou uitspringen:

- **it2 set-status** zet de status van een sessie: het bolletje, de ondertitel en de kleur van de ondertitel op de tab.
- **it2setcolor tab [kleur]** kleurt de tab. Terug naar standaard met `it2setcolor tab default`.

De scripts komen in `$HOME/.iterm2/` te staan. Onderaan je `~/.iterm2_shell_integration.zsh` komen verwijzingen ernaartoe.

Ik heb het op jouw MacBook nagekeken: **it2** en **it2attention** staan al op je **PATH**, want iTerm2 3.7 levert ze mee in de app zelf

(`/Applications/iTerm.app/Contents/Resources/utilities/`).

`imgcat` heb je al staan in `~/.local/bin/`.

Nog iets: `it2` werkt ook **over ssh**. Draai je het op een computer waar je via de SSH-integratie van iTerm2 naartoe bent gegaan, dan komt het verzoek terug bij je eigen iTerm2. De eerste keer wordt om toestemming gevraagd, per sessie.

5. Zonder installatie: triggers

Soms kun je het script niet installeren. Op een server van iemand anders bijvoorbeeld. Of als `root`, wat volgens de documentatie meestal een slecht idee is.

Dan gebruik je **triggers**. Een trigger kijkt naar de tekst op je scherm en doet iets als er een patroon in staat. Drie triggers doen samen bijna hetzelfde als Shell Integration:

- **Report User & Host:** vertelt iTerm2 wie je bent en waar je zit
- **Report Directory:** vertelt iTerm2 in welke map je staat
- **Prompt Detected:** vertelt iTerm2 waar je prompt begint, en zet daar de mark

Het voorbeeld uit de documentatie gaat uit van een prompt als

`george@example.com:/home/george%`. Voor gebruiker en computer:

```
^(\w+)(@([\w. ]+)):.+%
```

met als parameter `\1@\2`. En voor de map:

```
^\w+@([\w. ]+):([^\%]+)%
```

met als parameter `\1`.

Zet bij beide het vinkje "Instant" aan. Anders werken ze pas als er een nieuwe regel binnenkomt, en dan ben je te laat.

Je mag ook alleen een gebruikersnaam opgeven (gebruiker@) of alleen een computernaam (@voorbeeld.nl). Wat je weglaat blijft staan zoals het was.

Triggers kunnen veel meer

De trigger-pagina noemt een hele lijst acties. Voor jou zijn deze interessant:

- **Set Tab Status:** zet het bolletje, de ondertitel en de kleur van de ondertitel op de tab
- **Set Title:** zet de titel van de sessie
- **Post Notification:** stuurt een macOS-melding
- **Bounce Dock Icon:** laat het dock-icoon springen tot je het venster aanklikt
- **Show Alert:** toont een venster met je eigen tekst
- **Run Command / Run Silent Coprocess:** draait iets voor je
- **Set Named Mark:** geeft een mark een naam, zodat je er gericht naartoe kunt springen
- **Highlight Line** en **Change Style:** kleurt de regel of de tekst
- **Enter Workgroup** en **Exit Workgroup:** bouwt of sluit een groepje samenwerkende sessies

En er zijn **gebeurtenis-triggers**: die kijken niet naar tekst maar naar wat er gebeurt.

Deze zijn voor jou relevant:

GEBEURTENIS	GAAT AF ALS	FILTER
Command Finished	een commando klaar is (heeft Shell Integration nodig)	foutcode: alles, 0, niet-0, of een precies getal
Long-Running Command	een commando langer draait dan je grens	grens in seconden (standaard 60) plus eventueel een patroon
Idle (Silence)	er even niets meer binnenkomt	aantal seconden (standaard 30)
Activity After Idle	er weer iets binnenkomt na stilte	aantal seconden dat "stil" betekent
Job Started / Job Ended	een programma start of stopt op de voorgrond	de naam van het programma, bijvoorbeeld <code>c</code> laude
Prompt Detected	Shell Integration ziet een nieuwe prompt	geen
Directory Changed / Host Changed / User Changed	je map, computer of gebruiker verandert	patroon

GEBEURTENIS	GAAT AF ALS	FILTER
Session Ended	de sessie stopt	geen
Variable Changed	een sessie-variabele verandert	naam van de variabele plus eventueel een patroon
Notification Posted	er komt een melding binnen via OSC 9	patroon op de tekst

Elke trigger heeft ook een los veld **Job**. Zet je daar `claude` in, dan werkt die trigger alleen zolang `claude` op de voorgrond draait. De documentatie noemt Claude Code hier zelf als voorbeeld.

Triggers kun je ook **exporteren naar een `.it2triggers` -bestand** of kopiëren als link. Dus je kunt een goede set één keer maken en over al je profielen uitrollen.

6. Automatisch van profiel wisselen

Met Shell Integration weet iTerm2 wie je bent, op welke computer je zit en in welke map. Daarmee kan hij automatisch een ander profiel opzetten. Bijvoorbeeld: een rode achtergrond zodra je op de VPS zit.

De regels zet je in **Settings > Profiles > Advanced**. Een regel bestaat uit maximaal drie delen: gebruikersnaam, computernaam en pad. Minstens één deel moet erin staan.

- Gebruikersnaam eindigt op `@`, bijvoorbeeld `root@`
- Pad begint met `/`
- Computer en pad worden gescheiden door `:`, bijvoorbeeld `iterm2.com:/users/george`
- `*` is een jokerteken
- Een programmaam begint met `&`, bijvoorbeeld `&vim`
- Een regel die met `!` begint is **sticky**: het profiel blijft staan als de regel niet meer klopt, zolang geen andere regel wel klopt

Passen er meerdere regels tegelijk? Dan wint de regel met de hoogste score:

ONDERDEEL	PUNTEN
computernaam exact	16

ONDERDEEL	PUNTEN
computernaam met jokerteken	8
programmanaam	4
gebruikersnaam	2
pad exact	1
pad met jokerteken	0 (telt wel mee als match)

Klopt geen enkele regel meer, dan gaat de sessie terug naar het profiel waar hij mee begon.

7. De statusbalk (dit is jouw hoek)

De statusbalk staat los van Shell Integration, maar hij wordt er veel beter van.

Aanzetten: **Settings > Profiles > Session > Status bar enabled**, dan **Configure Status Bar**. Je sleept onderdelen naar boven.

Onderdelen die je kunt neerzetten:

- **Current Directory, Host Name, User Name:** de documentatie zegt er zelf bij dat je Shell Integration nodig hebt om deze kloppend te houden als je via ssh werkt
- **Job Name:** welk programma nu op de voorgrond draait. Klikken toont de hele boom van processen in die sessie
- **git state:** je tak, of er wijzigingen zijn, en hoeveel je voor of achter loopt op origin
- **Battery, CPU, Memory, Network:** grafiekjes over tijd
- **Clock, Composer, Search Tool, Filter, Snippet, Triggers**
- **Progress Indicator:** een voortgangsbalk
- **Interpolated String:** je eigen tekst met variabelen erin. **Dit is de flexibele.**
- **Call Script Function:** laat het resultaat van je eigen Python-functie zien

Handig detail: elk onderdeel heeft een **Priority** (standaard 5). Is er te weinig ruimte, dan verdwijnen eerst de onderdelen met de laagste prioriteit. Zet het belangrijkste dus hoger.

De git-tegel is niet alleen kijken. Klikken geeft je knoppen voor commit, add en commit, stash, log, pull, push en wisselen naar recente takken. Het git-commando draait in een afgeschermd hokje dat alleen naar `/tmp` en `/var` mag schrijven, zodat er niets stuk gaat.

8. Claude Code in iTerm2 (het stuk waar jij het meest aan hebt)

iTerm2 3.7 heeft een **eigen Claude Code-integratie**. Jij hebt versie 3.7.0. Ik heb het nagekeken.

Installeren: **iTerm2 > Install Claude Code Integration**. Er opent een venster dat eerst laat zien wat het gaat veranderen, en dat je dan stap voor stap meeneemt. Elke stap is optioneel, behalve de Python-API aanzetten.

Wat de installatie doet:

1. **Zet de Python-API aan.** Nodig, want daarmee geeft Claude zijn status door aan iTerm2. macOS vraagt je om dit te bevestigen.
2. **Zet een hook in je `~/.claude/settings.json`.** Die hook meldt wat Claude aan het doen is. Hij wijst naar een vaste plek in de configuratiemap van iTerm2, dus een update van iTerm2 breekt hem niet.
3. **Zet de toolbelt aan met de tegel Session Status.**
4. **Zet een Claude Code Workgroup in je instellingen.**
5. **Zet Enter- en Exit-triggers op de profielen die jij aanwijst.**

Session Status: welke tab wacht op jou

Dit is de kern. De tegel **Session Status** in de toolbelt toont al je sessies met een gekleurd bolletje en een label:

STATUS	BETEKENIS
working	Claude draait een opdracht of een gereedschap
waiting	Claude heeft jou nodig, bijvoorbeeld om iets goed te keuren
idle	Claude is klaar en wacht op je volgende bericht

Twee dingen die het echt bruikbaar maken voor jouw manier van werken:

- **Achtergrond-agents tellen mee.** Met een recente versie van Claude Code blijft een sessie op **working** staan zolang achtergrond-agents of achtergrondtaken nog bezig zijn, ook als het hoofdgesprek even stil is. Zo gaat eenwachter niet onterecht af tussen twee fases van subagents door.

- **De lijst sorteert op urgentie.** Sessies die jou nodig hebben komen bovenaan. Eén klik op een regel springt naar die sessie.

Er kan ook een extra regeltje bij staan: voor welk gereedschap toestemming gevraagd wordt, of een stukje van Claudes laatste bericht.

Toolbelt aan- en uitzetten kan altijd zelf: **View > Toolbelt > Show Toolbelt** of **Cmd-Shift-B**.

De Workgroup

Een **Workgroup** bouwt uit één sessie een groepje bij elkaar horende sessies. De Claude Code-Workgroup maakt er drie:

- **Chat:** je gewone Claude-gesprek
- **Diff:** je wijzigingen naast elkaar
- **Code Review:** een aparte sessie om de wijzigingen na te lopen

Je wisselt met één klik via de werkbalk van de Workgroup. De hele indeling kun je aanpassen in **Settings > Arrangements > Workgroups**. Het is dus geen dichtgetimmerd ding, maar gewoon instellingen die je mag veranderen.

Heb je de triggers geïnstalleerd, dan gaat de Workgroup vanzelf open als `claude` start in een profiel dat je hebt aangewezen, en weer dicht als hij stopt. Die triggers kijken naar de `claude -job`, dus ze gaan niet af bij iets anders. En de Exit-trigger geldt alleen voor de leider: stop je Claude in de Diff- of Code Review-sessie, dan blijft de rest gewoon staan.

Eerst uitproberen

Draait Claude Code in een sessie en heb je niets ingesteld? Dan biedt iTerm2 één keer aan: *"Claude Code is running. Want to try the Claude Code integration?"* Met **Try It Now** ga je tijdelijk de Workgroup in, tot Claude stopt. Met **Don't Show Again** is de vraag voorgoed weg.

Als hij stukgaat

De documentatie is hier eerlijk over: **Claude Code herschrijft af en toe je `~/ .claude/settings.json` en kan de `cc-status-hook` eruit gooien.** Gebeurt dat, dan merkt iTerm2 het de volgende keer dat Claude draait en biedt aan de hook opnieuw te zetten. Handmatig kan ook: **iTerm2 > Reinstall Claude Code Integration**.

Weghalen: **iTerm2 > Uninstall Claude Code Integration**. Dat haalt de hook, de Workgroup en de triggers weer weg. De Python-API blijft aan als je dat wilt, want andere scripts kunnen die inmiddels gebruiken.

9. Installeren

Er zijn **vier** manieren. In volgorde van makkelijk naar precies.

A. Automatisch laden (iTerm2 3.5 en nieuwer)

Zet **Load shell integration automatically** aan in **Settings > Profiles > General**, in het blok **Command**.

Dit kan als je **Command** op **SSH** staat, of op **Login Shell** met een ondersteunde shell.

Dit is de netste manier: er verandert niets aan je `~/.zshrc` .

B. Via het menu

iTerm2 > Install Shell Integration. Je krijgt eerst de vraag of je ook de Utilities wilt.

Daarna kies je:

- **downloaden en uitvoeren:** iTerm2 typt dit commando voor je
`curl -L https://iterm2.com/shell_integration/install_shell_integration.sh | bash`
- **Internet-Free Install:** iTerm2 bepaalt zelf je shell, zet de bestanden neer en past je startbestanden aan, zonder iets te downloaden

C. Met de hand (voor zsh, jouw shell)

```
curl -L https://iterm2.com/shell_integration/zsh -o ~/.iterm2_shell_integration.zsh
```

En dan **onderaan** je `~/.zshrc` :

```
source ~/.iterm2_shell_integration.zsh
```

Voor de volledigheid, de andere shells uit de documentatie:

SHELL	BESTAND	WAAR DE SOURCE-REGEL HEEN MOET
bash	<code>~/.iterm2_shell_integration.bash</code>	onderaan <code>~/.bash_profile</code> of <code>~/.profile</code>
zsh	<code>~/.iterm2_shell_integration.zsh</code>	onderaan <code>~/.zshrc</code>
fish	<code>~/.iterm2_shell_integration.fish</code>	onderaan <code>~/.config/fish/config.fish</code>
tcsh	<code>~/.iterm2_shell_integration.tcsh</code>	onderaan <code>~/.login</code>
xonsh	<code>~/.config/xonsh/rc.d/iterm2.xsh</code>	niets nodig, xonsh laadt die map zelf

D. Via triggers

Zie hoofdstuk 5. Voor computers waar je niets mag installeren.

En weer uitzetten

- Manier A: het vinkje weer uit.
- Manier B en C: haal de `source` -regel uit je `~/.zshrc` weg en open een nieuwe tab. Het bestand mag blijven staan, dat doet niets zolang niemand het laadt.
- Manier D: zet de triggers uit.
- Claude Code-integratie: **iTerm2 > Uninstall Claude Code Integration.**

Verder is er nog het geval **root**. Wil je Shell Integration ook als root, dan zegt de documentatie: word root met `sudo -s` (die laadt je `.bashrc` als root) en zet dit in je `.bashrc` :

```
test $(whoami) == root && source "${HOME}/.iterm2_shell_integration.bash"
```

Het alternatief is opnieuw: triggers.

10. Valkuilen en beperkingen

Dit staat allemaal letterlijk in de documentatie.

tmux en screen

- Standaard werkt Shell Integration **niet** met tmux of screen.
- Wil je het toch met tmux, zet dan dit in je startbestand **voordat** Shell Integration geladen wordt:

```
export ITERM_ENABLE_SHELL_INTEGRATION_WITH_TMUX=1
```
- Het werkt **goed** met de tmux-integratie van iTerm2 zelf (`tmux -CC`), maar **niet** met de gewone tmux-schermindeling.
- Ter info: bij `tmux -CC` opent iTerm2 een venster dat aanvoelt als een gewoon venster, maar tmux blijft aan de andere kant doordraaien. Valt je verbinding weg, dan pak je met `tmux -CC attach` alles terug zoals het was.

Prompt-thema's

Gebruik je een zsh-thema dat je prompt zelf regelt, dan botst dat. Los het op met `ITERM2_SQUELCH_MARK=1` en zet de mark handmatig. Zie hoofdstuk 2.

Volgorde in je startbestand

De source-regel hoort **onderaan**. De documentatie zegt het er expliciet bij: andere scripts kunnen overschrijven wat Shell Integration nodig heeft, bijvoorbeeld `PROMPT_COMMAND` .

tcsh

Daar kun je de mark niet zelf verplaatsen. Die functie bestaat niet in tcsh.

scp: hoe uploads en downloads echt werken

Dit is belangrijk als je met sleutels en ssh-instellingen werkt, zoals bij jouw VPS.

- iTerm2 gebruikt **libssh2** in de app zelf. Het roept dus **niet** het `scp` -commando aan.
- Het houdt zich aan `/etc/known_hosts` en `~/.ssh/known_hosts` , en werkt die laatste bij.
- Vingerafdrukken van computers worden gecontroleerd.
- Wachtwoord, keyboard-interactive en publieke sleutels worden ondersteund.
- Standaard zoekt hij je sleutel in `~/.ssh/id_rsa` , `id_dsa` , `id_ed25519` of `id_ecdsa` . Een wachtwoordzin mag.
- Van je `ssh_config` begrijpt hij maar een **deel**: `Host` , `HostName` , `User` , `Port` en `IdentityFile` . Meer niet.
- Wat uit `ssh_config` komt **overschrijft** de computernaam en gebruikersnaam die Shell Integration doorgaf.
- De bestanden worden in deze volgorde gelezen:
 1. `~/Library/Application Support/iTerm/ssh_config`
 2. `~/.ssh/config`
 3. `/etc/ssh_config`
- Gebruik je de SSH-integratie van iTerm2, dan gaan bestanden **niet** via scp maar via het eigen protocol van die integratie. Dat is betrouwbaarder, want er hoeft geen nieuwe verbinding open.

Nieuw in 3.7

- De SSH-integratie draait het ssh-commando via je login-shell. Je startbestanden draaien dus eerst en ssh ziet je eigen omgeving. Voordeel: een eigen `SSH_AUTH_SOCK` naar een agent met je sleutels wordt nu wel gebruikt.
- Bij een download vraagt iTerm2 of het bestand naar je Downloads-map moet of ergens anders heen. Die vraag kun je uitzetten.

Snelheid

De documentatie geeft geen snelheidscijfers over Shell Integration zelf. Wat wel genoemd wordt over prestaties:

- Triggers kijken **per regel**, niet over meerdere regels heen. Bij een hele lange regel worden alleen de laatste drie schermregels bekeken. Dat is expliciet **om prestatieredenen**. Je kunt die grens verhogen in Advanced Settings.
- De git-tegel in de statusbalk draait echte git-commando's. Je kunt zelf instellen hoe vaak. Zet je dat te vaak, dan werkt het meer dan nodig.

Eerlijk: ik kan niet met bewijs zeggen hoeveel milliseconden Shell Integration per prompt kost. Dat staat er niet. Wat er wel staat, is dat het script zich aan je prompt hangt, dus het draait bij elke prompt mee. Als je het merkt, is het meestal je thema en niet Shell Integration.

11. Wat betekent dit voor Jamal

Jij werkt de hele dag in iTerm2 met een rij tabs vol Claude Code-agents: Conductor, Appel, gastvrouw, boekhouder, SEO, VCA, webdesigner. Je grootste last is niet dat een agent iets niet kan. Je grootste last is **weten welke tab jou nodig heeft**.

Daar is dit precies goed voor.

Wat ik zou aanzetten, in deze volgorde

1. De Claude Code-integratie (grootste winst, minste werk)

- iTerm2 > Install Claude Code Integration
- Je krijgt de tegel **Session Status** met per tab een bolletje: **working**, **waiting**, **idle**.
- Sessies die op jou wachten komen **bovenaan**. Eén klik en je bent er.
- Achtergrond-agents tellen mee, dus een tab die met subagents bezig is springt niet ten onrechte op "klaar".
- Dat lost het probleem op dat in je eigen regels staat: *"een bot die die vraag beantwoordt draait in een verse sessie en ziet zijn eigen parallele processen niet"*. iTerm2 meet dit **van buitenaf**. Dat is precies wat jouw regel voorschrijft.
- **Let op:** de documentatie waarschuwt dat Claude Code je `~/ .claude/settings.json` soms herschrijft en de hook eruit gooit. Jouw settings.json is groot en heeft veel hooks (PreToolUse, UserPromptSubmit, SessionStart, PreCompact, PostToolUse, Stop, SessionEnd). Dus: **maak eerst een backup van dat bestand**, en houd er rekening mee dat je hem soms opnieuw moet zetten. iTerm2 biedt dat zelf aan.

2. Shell Integration zelf

Bij jou staat hij nu **uit**. Gemeten: geen `~/ .iterm2_shell_integration.zsh` , geen regel in `~/ .zshrc` . Ik zou hem aanzetten via het vinkje

Settings > Profiles > General > Load shell integration automatically. Dan blijft je `~/ .zshrc` schoon en is uitzetten net zo makkelijk.

Wat je er direct aan hebt:

- **Marks:** springen tussen commando's met Cmd-Shift-Omhoog en Omlaag. In een tab waar een agent honderden regels heeft uitgespuugd is dat goud. Je springt terug naar het commando in plaats van te scrollen.
- **Rode mark bij een fout.** Je ziet in één blik waar het misging.
- **Cmd-Opt-A** voor een alarm zodra een lang commando klaar is.
- **Uitvoer van het laatste commando in één keer selecteren.**
- **Recente mappen per computer.** Jij springt tussen `~/vca-work` , `~/financieel-advies` , `~/salon-coach` , `~/seo` en de VPS. Dit onthoudt dat voor je.

3. Een statusbalk per tab

Zet aan: **Settings > Profiles > Session > Status bar enabled.**

Wat ik erin zou zetten:

ONDERDEEL	WAAROM VOOR JOU
Job Name	zie meteen of <code>claude</code> , <code>python3</code> of <code>ssh</code> draait in die tab
Current Directory	welk project is dit, <code>~/vca-work</code> of <code>~/seo</code>
git state	tak, wijzigingen, voor of achter op origin. Jij werkt met veel repos
Interpolated String	hier zet je je eigen agent-label in
Clock	handig bij je 18:00-besluitenlijst

Je hebt al agent-labels per tab (`~/ .claude/agent-labels.json`). Die kun je hiermee ook in de statusbalk zelf zetten, niet alleen in de tabtitel.

4. Badges (grote tekst rechtsboven)

Een badge is groter en beter zichtbaar dan een tabtitel. Zet daar de agentnaam in. Dan zie je in een gesplitst venster meteen wie waar zit. Instelling:

Settings > Profiles > General > Badge.

5. Automatisch van profiel wisselen

Regel: zodra je op `149.210.171.174` zit, ander profiel met een andere achtergrondkleur. Dan kun je nooit meer per ongeluk denken dat je lokaal zit terwijl je op de VPS bent. Dat sluit direct aan op je eigen regel over `machine-lookup.py` : eerst weten op welke machine je zit.

Voorbeeld van zo'n regel in **Settings > Profiles > Advanced**:

```
jamalhaarvisie@149.210.171.174
```

6. Triggers voor je vloot

Een paar die ik zou maken:

- **Job Started** met filter `claude` → **Set Tab Status** → tab krijgt een bolletje
- **Long-Running Command**, grens bijvoorbeeld 300 seconden → **Post Notification**
- **Command Finished** met filter "niet-0" → **Highlight Line** rood, zodat een mislukt commando eruit springt
- **Idle (Silence)** van bijvoorbeeld 120 seconden → **Set Tab Status**, zodat je ziet welke tab stilstaat

En bedenk: het veld **Job** zorgt dat een trigger alleen werkt terwijl `claude` draait. Zo gaat hij niet af als jij zelf iets aan het typen bent.

Triggers kun je **exporteren** en in één keer in al je profielen zetten. Dus je maakt de set één keer goed.

7. `it2` in je eigen scripts

Dit is de mooiste. `it2` staat al op je PATH. Daarmee kan een script van jou **zelf** de tab kleuren of de status zetten als het klaar is.

Dat past precies bij je eigen regel *"beloof nooit ik wacht op de melding, laat de klus zichzelf melden"*. Je hangt gewoon `it2 set-status` of `it2attention fireworks` achter je achtergrondklus. Dan meldt de tab zichzelf, ook als de sessie al weg is.

En het werkt **over ssh**: draai je iets op de VPS via de SSH-integratie, dan komt de melding terug bij je eigen iTerm2 op de MacBook.

Waar je op moet letten

- **Alles hierboven is omkeerbaar.** Vinkje uit, regel weg, of Uninstall in het menu.
- **Backup je `~/.claude/settings.json`** voordat je de Claude Code-integratie installeert. Dat bestand is bij jou 14 KB groot en zit vol met jouw eigen hooks.
- **Gebruik je ergens tmux zonder `-CC` ?** Daar werkt Shell Integration niet, tenzij je die omgevingsvariabele zet.
- **De statusbalk en badges kosten geen geld en geen tokens.** Dit is puur je terminal.

Wat ik NIET heb gedaan

- Ik heb **niets geïnstalleerd of aangezet** op je MacBook. Dit verslag is alleen onderzoek plus een meting van wat er nu staat.
- Ik heb de Claude Code-integratie **niet uitgeprobeerd**, want dat verandert je `~/ .claude/settings.json` . Dat wil ik eerst met jou afspreken.
- **Eerlijk over onzekerheid:** de documentatie geeft geen cijfers over hoeveel Shell Integration kost aan snelheid. Ik heb dat niet gemeten en beweer er dus niets over.

Bronnen

Allemaal echt geopend op 9 september 2026:

- <https://iterm2.com/documentation-shell-integration.html>
- <https://iterm2.com/documentation-utilities.html>
- <https://iterm2.com/documentation-badges.html>
- <https://iterm2.com/documentation-triggers.html>
- <https://iterm2.com/documentation-automatic-profile-switching.html>
- <https://iterm2.com/documentation-status-bar.html>
- <https://iterm2.com/documentation-escape-codes.html>
- <https://iterm2.com/documentation-tmux-integration.html>
- <https://iterm2.com/claude-code-integration.html>