COMPLETE SETUP GUIDE · MAC & WINDOWS

The Always-On Setup

One machine that never sleeps. Your files on every device. Claude in your pocket.

- → Tailscale — a private network between your own machines
- → Syncthing — your files, everywhere, automatically
- → Remote Control — drive live sessions from your iPhone

Works on any combination of Mac and Windows. Hand this document to Claude and it will do the work.

START HERE

How to use this document

You read the first eight pages. Claude does everything after that.

The build itself is three blocks of text. You copy a block, paste it into Claude on the machine it names, and answer the questions it asks you. You do not need to understand a single command in this document — but the first eight pages explain what you are building and why, because people who understand it stop getting stuck.

> **IF YOU ONLY READ ONE THING**
>
> Use the **same Claude account** and the **same Tailscale account** on every machine and on your phone. Two different accounts is the single most common reason none of this works, and it looks like a broken install rather than the sign-in mistake it is.

01

What you end up with

You have a computer that sits in one place and stays plugged in. The desktop at the office. An old laptop on a shelf. A Mac mini behind the monitor.

When you finish this document, that machine will:

- **Never sleep.** Not when the screen goes dark, not when the lid closes, not after a restart.
- **Hold a live copy of your working files** that updates itself within seconds of you changing anything, anywhere.
- **Be reachable from your laptop** — from home, from a hotel, from a phone hotspot — over a private, encrypted link.
- **Run Claude sessions you can watch and steer from your phone.** Not a summary of the session. The session.

What it does not do

Nothing here opens your computer to the internet. Tailscale builds a private network between devices you own and sign in to; a stranger cannot find it, because there is nothing public to find. Syncthing sends your files from one of your machines to another with nothing in between — no company account, no server holding a copy.

Neither tool costs anything at this scale. Both are free for personal use, and Syncthing is free forever for everyone.

The shape of it

THE PIECE	WHAT IT IS
The always-on machine	The one that never sleeps. Long jobs run here. Your phone talks to this one. Usually the desktop that never leaves the building.
Your laptop	The one you carry. It holds the original of your files and pushes them to the always-on machine.
Tailscale	The private road between the two. Follows them anywhere.
Syncthing	The truck that drives your files down that road, continuously, without being asked.
Remote Control	What puts the live session on your phone.

You can add a third or fourth machine later using the same steps. Nothing about this is limited to two.

02

Why two tools instead of one

People try to do this with one tool and it half-works. Here is why it takes two, and what each one is actually for.

Tailscale

The road

Tailscale makes your machines able to reach each other. It builds a small private network — only your devices, only ones signed in to your account — and keeps it working no matter where those devices go. Your laptop on hotel wifi and your desktop at the office behave as though they are sitting on the same desk.

It moves no files. Tailscale is the road. An empty road delivers nothing.

Syncthing

The truck

Syncthing watches a folder and copies every change to your other machines, continuously, in the background. No cloud account. No company server. The files go from your machine straight to your other machine, encrypted, and nobody else ever holds a copy.

It needs a path. On the same wifi it finds one easily. Across the internet it has to negotiate through routers and firewalls, and when it cannot, it falls back to a public relay run by volunteers — which is slow, and which is somebody else's computer in the middle.

Together

Put Tailscale underneath Syncthing and the negotiation problem disappears. Syncthing always has a clean, direct, encrypted path to your other machine, because Tailscale already built one. The relay never gets used. Sync goes from *"works at home, sometimes, slowly"* to *"works everywhere, instantly."*

That is the whole argument. Tailscale without Syncthing gives you machines that can talk but files that are stuck in one place. Syncthing without Tailscale gives you files that move on your home wifi and crawl everywhere else.

Why not Dropbox, iCloud, or OneDrive?

REASON	WHAT IT MEANS
Your files sit on their server	Client work, business records, anything you would not email to a stranger — it all lives on a company's hardware under their terms.
They are bad at what you have	A code repository or a notes vault is thousands of small files changing constantly. That is precisely the workload cloud sync handles worst — and where it produces the most conflict files.
The trip is twice as long	Cloud sync goes up to a data centre and back down. Syncthing goes machine to machine. It is faster, and it keeps working on a day when the cloud provider is having problems.

Why not just remote into the machine?

You can, and Tailscale makes that easy. But remote desktop means the files only exist over there. The moment your connection drops or you board a plane, you have nothing. Syncthing means the files exist in **both** places, all the time, so a dead connection costs you the remote session and not the work.

> **THE SHORT VERSION**
>
> Tailscale is how the machines find each other. Syncthing is how the files get there. You want both, and the setup for each is about ten minutes.

03

How they work together — one ordinary day

Abstractions do not stick. Here is the same setup described as a Tuesday.

8:40am — kitchen table

You open your laptop and work on a proposal. Every time you save, Syncthing notices within a second or two and pushes the change to the office machine. You do not open Syncthing. You do not think about Syncthing. It is a background process with a small icon.

9:15am — in the car

Laptop closed, tethered to your phone. Tailscale does not care that your network changed; the private link between your two machines follows you. Anything you saved before you closed the lid has already landed.

10:30am — a job too big for a laptop

You need Claude to work through something long — rebuild a site, process a pile of documents, run a full analysis. On your laptop you type `ssh office-pc`, you are on the always-on machine, you start the session there, and you walk away. The laptop can now go to sleep in your bag. The job is running on a machine that cannot.

12:05pm — back of a cab

You open the Claude app on your iPhone. The session is in the list. You tap it and you are looking at the real thing — the commands running, the files being edited, live. You type a correction. It lands. You put the phone away.

2:40pm — a client asks for something, and you are nowhere near a desk

You open the app and start a *brand new* session on the office machine from the phone. You never touched the computer. That is the `ccrs` command, and Section 04 explains it.

6:00pm — back at the laptop

Everything the office machine produced today is already sitting in your folder. There was no download step. There was never a moment where you had to remember to move a file.

> **THE POINT**
>
> Notice what is missing from that day: no uploading, no downloading, no emailing yourself a file, no "which version is the current one," and no being stuck at a desk because a task has forty minutes left to run.

04

What the iPhone adds

This is the piece that did not exist before, and it is worth being precise about what changed.

What it used to be

A Claude Code session lived inside a terminal window on one computer. Close the laptop and the session died with it. If a task had thirty minutes left, you had thirty minutes left at your desk. That was the deal.

What it is now

Remote Control publishes a live session to the Claude app on your phone and to `claude.ai/code` in any browser. What arrives on the phone is not a summary and not a transcript. It is the session — every command, every file edit, as it happens — and you can type into it.

The machine keeps doing the work. Your phone is a window onto it.

The two commands

COMMAND	WHAT IT DOES	WHEN TO USE IT
`ccr`	Shares the session in whatever folder you are standing in.	One task, and you are about to step out.
`ccrs`	Runs a *spawner*. Your phone can open brand-new sessions on that machine on demand.	You want the machine available all day without deciding in advance what for.

If you are not sure which, use `ccr`.

Why `ccrs` is the one that changes how you work

`ccr` shares a session you already started. Useful, but you still had to be at the machine to start it.

`ccrs` hands your phone the ability to *open* sessions on that machine. You start it once in the morning and leave the window open. For the rest of the day, from anywhere, you open the app and begin new work on a machine you are nowhere near. Standing in a parking lot, you can start a session, point it at a project, and set it running.

Name your sessions if you plan to run several, or you will not be able to tell them apart on a phone screen:

```
claude remote-control --name "Client Site" --permission-mode bypassPermissions
```

The flag everybody skips, and why it matters most

> **READ THIS TWICE**
>
> `--permission-mode bypassPermissions` turns off the approval prompts.
>
> **Your phone cannot click "Allow."** Without this flag, the first moment Claude wants to run a command, the session stops and waits for a click that will never come — and you are looking at a frozen session from the back of a cab with no way to unfreeze it.
>
> The commands in this guide include the flag. If you ever write your own, include it.

What holds it all up

None of this survives the machine going to sleep. Keep-awake is not a nice extra — it is the foundation, and it only holds on **AC power**, on both Mac and Windows. Plug the machine in before you close the lid. Step 1 of the build sets this permanently.

05

Mac, Windows, or both — answered

The question everybody asks: "My laptop is a Mac and the office machine is Windows. Is that a different setup?" No. Here is the whole matrix.

FROM ↓ TO →	A MAC	A WINDOWS PC
A Mac	**Identical**	**Identical**
A Windows PC	**Identical**	**Identical**

Not "similar." Identical. There is no such thing as a Mac-to-Windows procedure in this setup, and three concrete reasons why.

1. Tailscale is one product with one account

There is no Mac network and no Windows network. There is your account, and every machine signed in to it appears in the same list with the same kind of name and the same kind of address. A Mac and a Windows PC see each other exactly the way two Macs do.

2. Syncthing has no per-platform protocol

You exchange Device IDs — a long code with dashes. A Windows Device ID and a Mac Device ID look the same and behave the same. The folder path differs (`C:\Users\you\vault` against `/Users/you/vault`) and Syncthing does not care; it matches folders by the **Folder ID** you give them, which is why every block in this guide uses the same one: `vault`.

3. `ssh` is `ssh`

The command to reach the other machine is the same sentence typed in a Mac terminal or in Windows PowerShell:

```
ssh yourname@office-pc
```

Windows has shipped an OpenSSH client for years. You do not install anything to connect *out* of a Windows machine.

So what does differ?

Per **machine**, never per pair. Work out what each of your machines is, run that machine's block, and stop thinking about the combination.

DIFFERS	ON A MAC	ON WINDOWS
Installing the apps	`brew install` / the Mac app	`winget install`
Letting others connect in	Remote Login, in Sharing settings	The OpenSSH Server feature
Stopping it sleeping	`pmset` plus a `caffeinate` login item	`powercfg` plus a scheduled task
Where the phone commands live	`~/.zshrc` (or `~/.bashrc`)	Your PowerShell `$PROFILE`

Three cross-platform gotchas that are real

WORTH KNOWING BEFORE THEY BITE

Forbidden characters. Windows will not accept a filename containing `: * ? " < > |`. If a Mac creates one, that single file will not land on the Windows side. Syncthing reports it clearly rather than failing silently — rename the file and it goes through.

Upper and lower case. Mac and Windows both treat `Report.md` and `report.md` as the same file. Linux does not. This only matters if a Linux machine ever joins the group.

Landing in the wrong shell. When you `ssh` *into* a Windows machine you arrive in the old `cmd` prompt, not PowerShell, unless someone changed it. The Windows block in Step 1 changes it.

06

Before you start

Five things. Settle them now and the build takes half an hour. Skip them and you will spend an afternoon chasing a problem that was never technical.

1. One Claude account, everywhere

The same account on the always-on machine, on your laptop, and in the app on your phone. If the phone shows an empty session list, this is almost always the reason.

2. One Tailscale account, everywhere

Same rule, and the same consequence. Two machines signed in to two different Tailscale accounts cannot see each other and never will. It presents as a broken install. It is a sign-in mistake.

3. Decide which machine holds the originals

One machine is the **source of truth** — the one that already has your files. Usually your laptop. The other starts empty and receives.

Say it out loud before you begin, and tell Claude which one it is when the block asks. Getting this backwards is the only step in this entire document that can lose you work: an empty folder declared the source will happily empty the other one.

4. The always-on machine lives on AC power

Keep-awake does not hold on battery, on either operating system. If your always-on machine is a laptop, it must stay plugged in. Closing the lid is fine once Step 1 is done — unplugging is not.

5. Claude Code must already run on each machine

Check it with `claude --version` in a fresh terminal. If it is missing, each block below installs it.

> **WINDOWS ONLY — THE QUIET ONE**
>
> Claude Code on Windows needs Git for Windows installed, and the setting `CLAUDE_CODE_GIT_BASH_PATH` must point at **`bash.exe`** — not `git.exe`.
>
> Pointed at the wrong file it breaks quietly, with symptoms that look like anything but a path setting. It is the most common Windows problem there is. Every Windows block below checks it first.

If your company's IT blocks something

An administrator password you do not have. An install that is refused. A PowerShell execution policy that will not change. Those are permissions policy, not something you did wrong and not something you can retry your way past.

Stop, and send it to whoever handles IT. Sending that request *early* is the single biggest time-saver in this whole process — the approval, not the installing, is what takes the week.

What the build looks like

STEP	WHERE YOU PASTE IT	ROUGHLY
Step 1	Claude, on the machine that stays put	20 minutes
Step 2	Claude, on your laptop	10 minutes, plus the first sync
Step 3	Claude, on the machine you want to reach from your phone	5 minutes

Do them in order. Step 2 needs a code that Step 1 produces.

THE BUILD — STEP 1 OF 3

07

The machine that stays put

This is the desktop that never leaves, or the laptop that lives on a shelf plugged in. Open Claude on *that machine* and paste the block for its operating system.

When it finishes, Claude will hand you a **Syncthing Device ID** — a long code with dashes. Copy it somewhere you can get at it. Step 2 needs it.

> ONE THING SOFTWARE CANNOT DO
>
> **Restarting automatically after a power cut** is a BIOS or firmware setting on a Windows PC. No command can set it. If the building loses power, the machine comes back only if you have turned on something like "Restore on AC Power Loss" or "After Power Failure: Power On" in the BIOS.
>
> A Mac can do this in software, and Block 1A sets it. A Windows PC cannot. Ask Claude to tell you where to look in your BIOS, and set it yourself next time you restart.

BLOCK 1A — PASTE INTO CLAUDE MACOS

I'm setting this Mac up as my always-on AI machine. It should never sleep, it should be reachable from my other machines and my phone, and it should share one synced folder with my laptop. Put yourself in goal mode and don't come back to me until all six parts are done or you're genuinely blocked. Explain anything you need from me like I'm in fifth grade, and tell me plainly when you need my password rather than failing quietly.

1. Install Homebrew if it isn't here. Then make sure the Claude Code CLI is installed using the native installer — `curl -fsSL https://claude.ai/install.sh | bash` — not npm and not brew. Confirm `claude --version` works in a fresh terminal.
2. Install Syncthing with `brew install syncthing` and set it to start automatically and stay running across reboots (`brew services start syncthing`). Open its page at `http://127.0.0.1:8384` and give this machine a clear name — something like "Office Mac" — so I can recognise it later. Then print this machine's **Device ID** in large text. It's the long code with dashes. I need to copy it to my laptop, so make it impossible to miss.
3. Create the folder that will hold my synced files and share it in Syncthing under the Folder ID `vault` . Use exactly that ID. Leave the folder empty — my laptop is the source of truth and will fill it in. Tell me the full path you used.
4. Install Tailscale, walk me through signing in with the account I'll use on every machine, and confirm this Mac appears in my Tailscale device list. Turn on MagicDNS if it isn't already, so my machines can be reached by name instead of by number. Then turn on Remote Login under System Settings → General → Sharing so my other machines can connect in. Tell me this Mac's Tailscale name, its Tailscale IP, and the exact username I should use when connecting.
5. Make this Mac never sleep, permanently and across restarts:
   - Set the power settings so the computer never sleeps, the disks never sleep, and it does not sleep when the display turns off
   - Set it to restart automatically after a power failure
   - Install a login item that runs `caffeinate -dimsu` continuously, so keep-awake survives a reboot instead of being a one-off command I have to remember
   - If this Mac is a laptop, tell me clearly that all of this only holds while it's plugged in

   Then verify it actually worked and show me the proof — the real output, not a list of the commands you ran.
6. Give me one block I can copy, containing: this Mac's Syncthing Device ID, its Tailscale name, its Tailscale IP, my username on it, and the path to the synced folder. Under that, a short plain-English list of what is now true about this machine.

BLOCK 1B — PASTE INTO CLAUDE WINDOWS

I'm setting this Windows PC up as my always-on AI machine. It should never sleep, it should be reachable from my other machines and my phone, and it should share one synced folder with my laptop. Put yourself in goal mode and don't come back to me until all six parts are done or you're genuinely blocked. Explain anything you need from me like I'm in fifth grade, and tell me when you need an Administrator PowerShell rather than silently skipping a step.

1. Confirm Claude Code works here — run `claude --version` in a fresh terminal. If it's missing, install it with `irm https://claude.ai/install.ps1 | iex` and confirm again. Then check that `CLAUDE_CODE_GIT_BASH_PATH` in my Claude settings points at **bash.exe** and not git.exe. That one breaks quietly and it's the most common problem on Windows — check it even if everything looks fine.
2. Install Syncthing so it starts by itself and stays running. SyncTrayzor is the easiest way on Windows — `winget install SyncTrayzor.SyncTrayzor`. Set it to start with Windows and to start minimised. Open its page at `http://127.0.0.1:8384`, give this machine a clear name like "Office PC", and then print this machine's **Device ID** in large text. It's the long code with dashes. I need to copy it to my laptop, so make it impossible to miss.
3. Create the folder that will hold my synced files and share it in Syncthing under the Folder ID `vault`. Use exactly that ID. Leave it empty — my laptop is the source of truth and will fill it in. Tell me the full path you used.
4. Install Tailscale with `winget install tailscale.tailscale`, walk me through signing in with the account I'll use on every machine, and confirm this PC appears in my Tailscale device list. Turn on MagicDNS if it isn't already. Then turn on the OpenSSH Server so my other machines can connect in: add the Windows capability, set the `sshd` service to start automatically, start it, and confirm the firewall allows it. Also set PowerShell as the default SSH shell, so that when I connect in I don't land in the old cmd prompt. Tell me this PC's Tailscale name, its Tailscale IP, and the exact username I should use when connecting.
5. Make this PC never sleep, permanently and across restarts:
   - `powercfg /change standby-timeout-ac 0` and `powercfg /change hibernate-timeout-ac 0`
   - If this is a laptop, set it to do nothing when the lid closes on AC power
   - Turn off Fast Startup, which interferes with waking and with scheduled tasks
   - Create a Scheduled Task that runs at startup whether or not I'm logged in, so the machine is useful before anyone touches it
   - If this is a laptop, tell me clearly that all of this only holds while it's plugged in

   Then verify it actually worked and show me the proof — the real output of `powercfg /a` and the timeout settings, not a list of the commands you ran.
6. Give me one block I can copy, containing: this PC's Syncthing Device ID, its Tailscale name, its Tailscale IP, my username on it, and the path to the synced folder. Under that, a short plain-English list of what is now true about this machine.

THE BUILD — STEP 2 OF 3

08

Your laptop

Do this only after Step 1 has finished and handed you a Device ID. Open Claude on your laptop, paste the block for its operating system, and paste that Device ID into the line at the bottom.

> **SAY THIS OUT LOUD FIRST**
>
> Your laptop is the **source of truth** — it has the real files, and the other machine receives them. Both blocks below say so, and both tell Claude to confirm the direction with you before it starts. If your setup is the other way round, change that sentence before you paste.

BLOCK 2A — PASTE INTO CLAUDE MACOS

I want to sync my working files to my always-on machine and be able to reach that machine from here. Goal mode — work through all five parts, and explain anything you need from me simply.

1. Make sure Syncthing is installed and running on this Mac and set to start automatically (`brew install syncthing` and `brew services start syncthing` if it isn't). Open `http://127.0.0.1:8384` and give this machine a clear name like "My Laptop".
2. Add my always-on machine as a remote device using the Device ID at the bottom of this message. Then share my working folder with it using the Folder ID `vault` — exactly that ID, because that's what the other machine is expecting.
   This laptop is the source of truth. The other machine should receive my files, not overwrite them. Confirm with me that you have the direction the right way round, and tell me in one sentence which way you set it, before you start the first sync.
3. Watch it until the first full sync finishes, then confirm both sides say "Up to Date." If the other machine needs me to accept something at its end, tell me exactly what to click over there. Warn me that the first sync moves everything and can take a while — and that this is normal and only happens once.
4. Install Tailscale here and sign me in with the same account I used on the always-on machine. Confirm both machines can see each other in the device list, then actually test that I can reach the other machine from here — don't just tell me it should work.
5. Tell me, in one line I can write on a sticky note, exactly what I type when I want to work on the always-on machine from this laptop.

Always-on machine Device ID: [PASTE IT HERE]

BLOCK 2B — PASTE INTO CLAUDE WINDOWS

I want to sync my working files to my always-on machine and be able to reach that machine from here. Goal mode — work through all five parts, and explain anything you need from me simply. Tell me when you need an Administrator PowerShell rather than silently skipping a step.

1. Make sure Syncthing is installed and running on this PC and set to start automatically — `winget install SyncTrayzor.SyncTrayzor` if it isn't, set to start with Windows and start minimised. Open `http://127.0.0.1:8384` and give this machine a clear name like "My Laptop".
2. Add my always-on machine as a remote device using the Device ID at the bottom of this message. Then share my working folder with it using the Folder ID `vault` — exactly that ID, because that's what the other machine is expecting.
   This laptop is the source of truth. The other machine should receive my files, not overwrite them. Confirm with me that you have the direction the right way round, and tell me in one sentence which way you set it, before you start the first sync.
3. Watch it until the first full sync finishes, then confirm both sides say "Up to Date." If the other machine needs me to accept something at its end, tell me exactly what to click over there. Warn me that the first sync moves everything and can take a while — and that this is normal and only happens once.
4. Install Tailscale here with `winget install tailscale.tailscale` and sign me in with the same account I used on the always-on machine. Confirm both machines can see each other in the device list, then actually test that I can reach the other machine from here using `ssh` — don't just tell me it should work. Windows already has an SSH client built in, so nothing extra should be needed to connect out.
5. Tell me, in one line I can write on a sticky note, exactly what I type when I want to work on the always-on machine from this laptop.

Always-on machine Device ID: [PASTE IT HERE]

ADDING A THIRD MACHINE LATER

Run Block 2A or 2B on it, with the always-on machine's Device ID, exactly as written. Syncthing is not limited to pairs — every machine that knows the always-on machine ends up with the same folder. Nothing about Step 1 needs repeating.

THE BUILD — STEP 3 OF 3

09

Your phone

Do this on whichever machine you want to reach from your phone. Almost always that is the always-on machine — it is the one that cannot fall asleep and take your session with it.

BLOCK 3A — PASTE INTO CLAUDE MACOS

Set up Claude Code Remote Control on this Mac so I can drive sessions from my iPhone, including starting brand-new sessions without touching this computer. Goal mode — run the whole sequence without stopping between steps, and explain anything you need from me like I'm in fifth grade. Give me the Step 6 report as plain sentences, not a wall of checkmarks.

1. Confirm Claude Code is here — run `claude --version`. If it's missing, install it with `curl -fsSL https://claude.ai/install.sh | bash` and confirm again. If it still fails, stop and tell me the error rather than carrying on and writing shortcuts that point at nothing.
2. Check the version is new enough. Run `claude --help` and `claude remote-control --help` and confirm all three of these exist: `--remote-control`, `--permission-mode` accepting `bypassPermissions`, and `--spawn`. If any is missing, run `claude update` and check again. Never write a shortcut around a flag you haven't confirmed exists.
3. Find my shell config — run `echo $SHELL` and use `~/.zshrc` for zsh or `~/.bashrc` for bash. Don't assume zsh. Create the file if it doesn't exist. Check whether `ccr` or `ccrs` already exist in it; if either does, show me the current line and ask before overwriting it.
4. Append these two lines exactly:

   ```
   alias ccr='caffeinate -s claude --remote-control --permission-mode bypassPermissions'
   alias ccrs='caffeinate -s claude remote-control --permission-mode bypassPermissions --spawn same-dir'
   ```

   Then verify with `grep -n "alias ccr"` on that file and confirm both are present and correct.
5. Confirm this machine won't sleep. If I've already done Step 1 of this guide it's set — check it rather than setting it twice. If this is a laptop, remind me it has to stay plugged in, because keep-awake only holds on AC power.
6. Do NOT run `ccr` yourself — it takes over the terminal and I'll start it when I'm ready. Instead tell me, in plain sentences: which file you edited, that I need to open a new terminal window or run `source` on that file before the commands work, what `ccr` and `ccrs` each do in one line, that the machine must be plugged in before I close the lid, and that bypass mode means Claude runs commands without asking me first.

BLOCK 3B — PASTE INTO CLAUDE WINDOWS

Set up Claude Code Remote Control on this Windows PC so I can drive sessions from my iPhone, including starting brand-new sessions without touching this computer. Goal mode — run the whole sequence without stopping between steps, and explain anything you need from me like I'm in fifth grade. Give me the Step 6 report as plain sentences, not a wall of checkmarks.

1. Confirm Claude Code is here — run `claude --version`. If it's missing, install it with `irm https://claude.ai/install.ps1 | iex` and confirm again. If it still fails, stop and tell me the error rather than carrying on and writing shortcuts that point at nothing.
2. Check the version is new enough. Run `claude --help` and `claude remote-control --help` and confirm all three of these exist: `--remote-control`, `--permission-mode` accepting `bypassPermissions`, and `--spawn`. If any is missing, run `claude update` and check again. Never write a shortcut around a flag you haven't confirmed exists.
3. Find my profile with `$PROFILE`. If the file doesn't exist, create it with `New-Item -ItemType File -Path $PROFILE -Force`. Check whether `ccr` or `ccrs` already exist in it; if either does, show me the current line and ask before overwriting it.
4. Check the execution policy first with `Get-ExecutionPolicy -Scope CurrentUser`. If it's `Restricted` or `Undefined`, my profile will silently refuse to load — which is the single most common reason this whole thing appears not to work. Fix it with `Set-ExecutionPolicy -Scope CurrentUser RemoteSigned`, then tell me you changed it and what it means. If company policy blocks this, stop and tell me — that's one for our IT people, not something to work around.

   Then append these two functions to the profile. **PowerShell aliases cannot carry flags** — `Set-Alias` produces a broken command — so they have to be functions:

   ```
   function ccr  { claude --remote-control --permission-mode bypassPermissions @args }
   function ccrs { claude remote-control --permission-mode bypassPermissions --spawn
   same-dir @args }
   ```

   Verify with `Select-String -Path $PROFILE -Pattern "ccr"` and confirm both are present.
5. Confirm this machine won't sleep. If I've already done Step 1 of this guide it's set — check it with `powercfg /a` and the standby timeout rather than setting it twice. If this is a laptop, remind me it has to stay plugged in. If it keeps sleeping anyway, check whether it uses Modern Standby, which ignores the lid setting.
6. Do NOT run `ccr` yourself — it takes over the terminal and I'll start it when I'm ready. Instead tell me, in plain sentences: which file you edited, anything else you changed, that I need to open a new terminal window or run `. $PROFILE` before the commands work, what `ccr` and `ccrs` each do in one line, that the machine must be plugged in before I close the lid, and that bypass mode means Claude runs commands without asking me first.

10

The daily reference card

Everything above happens once. This is what you actually do afterwards.

Start a session you can reach from your phone

```
cd your-project-folder
ccr
```

Leave the terminal window open. Plug the machine in. Close the lid. Open the Claude app on your iPhone and the session is in the list — tap it.

Make the machine available all day for new sessions

```
ccrs
```

Leave it running. From then on you can start brand-new sessions from the phone without going near the computer.

Work on the always-on machine from your laptop

```
ssh yourname@office-pc
```

Same command on a Mac or in PowerShell. `office-pc` is the Tailscale name Claude gave you in Step 1. If names do not work, use the Tailscale IP instead — it starts `100.` and it always works.

Run several remote sessions at once

```
claude remote-control --name "Client Site" --permission-mode bypassPermissions
```

Without names they are indistinguishable on a phone screen.

Check on the pieces

YOU WANT TO KNOW	WHERE TO LOOK
Are my files synced?	`http://127.0.0.1:8384` in a browser on either machine. Both sides should say **Up to Date**.
Can my machines see each other?	The Tailscale menu-bar or tray icon. Every machine should be listed and not greyed out.
Is the always-on machine actually awake?	Try to `ssh` into it. That is the only test that matters.
Is my session still alive?	The terminal window on the machine is still open. Closing it ends the session.

> **THE ONE HABIT WORTH KEEPING**
>
> Before you walk away from the always-on machine: **plugged in, window open, sync says Up to Date.** Three seconds, and it prevents almost every problem in the next section.

11

When something breaks

Organised by what you see, not by what caused it — because when it breaks you only know the symptom.

Nothing shows up in the phone app

Three things, in this order. Is it the **same Claude account** on the phone and on the computer? Is the **terminal window still open** on the computer — closing it closes the session? Did you actually run `ccr`, or only set it up?

The session vanished while I was out

The machine went to sleep. Keep-awake only holds on AC power — check it is plugged in before you check anything else. On Windows, some laptops use Modern Standby and ignore the lid setting entirely; run `powercfg /a` if it keeps happening.

It is asking me to approve something and I am not at my desk

The bypass flag did not make it into the command. Check the alias or function, then start the session again. There is no way to answer the prompt from the phone.

ccr: command not found / ccr is not recognized

The terminal was not reloaded after the setup wrote the file. Run `source ~/.zshrc` on a Mac or `. $PROFILE` on Windows — or just open a new window, which is easier.

Windows: my profile changes do not stick

The execution policy is blocking the profile from loading. Run `Get-ExecutionPolicy -Scope CurrentUser`. If it says `Restricted`, set it to `RemoteSigned`. If company policy refuses, that is an IT question rather than a technical one.

The two machines cannot see each other

Almost always **two different Tailscale accounts**. Open Tailscale on both and check the signed-in email is the same. After that, check whether one machine is simply asleep. Only after both of those should you suspect anything else.

Syncing looks stuck, or is crawling

The first sync moves everything and is slow by nature; after that it is near-instant. If it is stuck rather than slow: does the other machine still need to **accept** the shared folder? Syncthing waits indefinitely for that click, and it looks identical to being broken. Open `http://127.0.0.1:8384` on the other machine and look for a pending request at the top.

One file will not sync and the rest are fine

Its name almost certainly contains a character Windows will not accept: `: * ? " < > |`. Syncthing names the file. Rename it and it goes through.

I have files ending in .sync-conflict-2026...

The same file was changed on two machines before they could talk to each other. Syncthing keeps both rather than picking a winner. Open the two, keep the one you want, delete the other. To make it rarer: let a machine finish syncing before you close the lid on it.

I connected into the Windows machine and the commands do not work

You landed in the old `cmd` prompt rather than PowerShell. Type `powershell` and press Return. To fix it permanently, ask Claude to set PowerShell as the default SSH shell on that machine.

The machine did not come back after a power cut

On Windows that is a BIOS setting and nothing on the computer can set it — see the warning at the start of Step 1. On a Mac, Block 1A sets it in software; if it did not come back, run that block again and ask Claude to show you the proof.

Claude Code on Windows behaves strangely in ways nothing explains

Check `CLAUDE_CODE_GIT_BASH_PATH`. If it points at `git.exe` rather than `bash.exe`, that is your answer. It is the most common Windows fault and it never announces itself.

> **STILL STUCK AFTER TEN MINUTES?**
>
> Stop and ask. Paste the exact error into Claude on that machine and tell it which step of this document you were on — that context is usually enough for it to fix the problem itself. Most of these take a minute once somebody has seen the actual message.

12

The honest limits

Four things this setup does not do, said plainly, so none of them surprises you later.

Battery ends everything

Keep-awake holds on AC power and nowhere else, on either operating system. An always-on machine on battery is an ordinary machine that will sleep and take your session with it. If your always-on machine is a laptop, it stays plugged in. That is the whole rule.

A power cut is a BIOS question on Windows

No command can make a Windows PC restart itself after the power goes out. It is a firmware setting, usually called something like "Restore on AC Power Loss." You have to set it yourself, in the BIOS, and the only time to do that is on a restart you planned. A Mac can be set in software and Block 1A does it.

Closing the terminal window closes the session

Remote Control shares a running session. It does not make the session survive on its own. The window stays open, or the session ends — no matter how awake the machine is.

Bypass mode means Claude does not ask

> **THE ONE WORTH THINKING ABOUT**
>
> `bypassPermissions` turns off the approval prompts. That is the entire point — your phone cannot answer them — but be clear-eyed about what you have agreed to. Use it on work you would have said yes to anyway, do not point it at anything you would hate to lose, and keep your projects in version control so there is always a way back. Those three rules are the whole of the risk management, and they are enough.

What you have when this is finished

A machine that does not sleep, holding a copy of everything you work on, reachable from anywhere, running sessions you can watch from your pocket. No file was uploaded to anyone, no monthly fee was added, and the setup took under an hour, once.

The change is not that any one task got faster. It is that the length of a task stopped deciding where you have to be.

> **ADDING MACHINES LATER**
>
> A new laptop, a machine at home, a second office PC — run **Block 2A or 2B** on it with the always-on machine's Device ID, then **Block 3A or 3B** if you want it on your phone too. Step 1 is done once, forever.