ccstatusline: configuring the status bar

ccstatusline

Renders the two-line status bar under every Claude Code prompt

```
# opens the config TUI
npx ccstatusline@latest

# or install it
npm install -g ccstatusline

# render once, reading stdin
ccstatusline --config F < payload.json
```

One command in, up to three lines out

Claude Code runs your status-line command on every prompt it draws

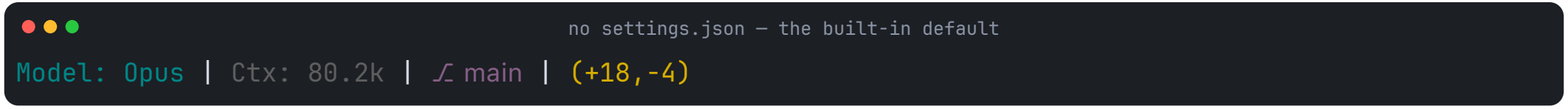
~/.claude/settings.json — who to run

```
{
  "statusLine": {
    "type": "command",
    "command": "ccstatusline"
  }
}
```

stdin — what it gets, per prompt

```
{
  "model": {"display_name": "Opus"},
  "workspace": {"current_dir": "..."},
  "cost": {"total_cost_usd": 1.20},
  "context_window": {
    "context_window_size": 200000,
    "current_usage": {...}
  }
}
```

With no settings.json of its own, ccstatusline still renders — this is the built-in default:



The TUI edits one committed JSON file

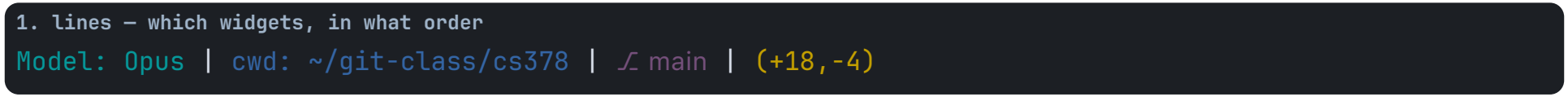
Edit it either way; only the file is worth committing

- npx ccstatusline@latest opens a terminal UI over the same file
  - Arrow keys add, reorder, and colour widgets, with a live preview
  - Use it to explore; read the file afterwards to learn what it wrote
- It lives at ~/.config/ccstatusline/settings.json
  - That path is what makes it a dotfile: symlink it out of your dotfiles repo and every machine gets the same bar
  - The --config flag overrides the location — which is how the pictures in this deck were made

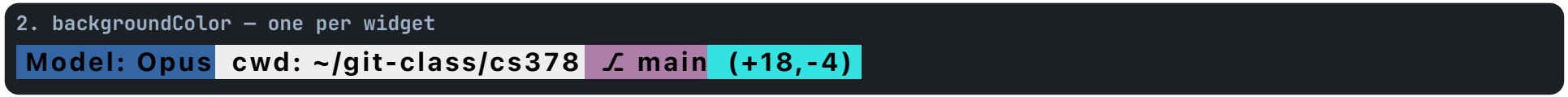
Five keys, five bars

Every strip below differs from the one above it by one key

1. lines — which widgets, in what order

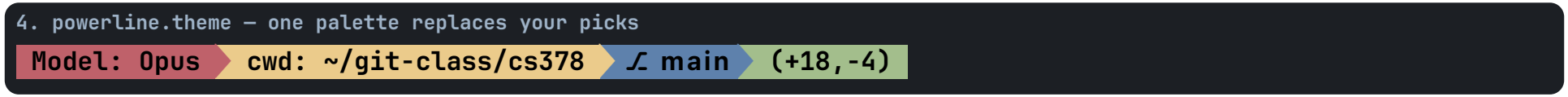


2. backgroundColor — one per widget



3. powerline.separators — segments interlock

4. powerline.theme — one palette replaces your picks



5. a second list in lines, plus powerline.autoAlign

The file that produced the last strip

Widgets are a list of lists: one inner list per rendered line

```
{
  "version": 3,
  "lines": [
    [ {"id": "1", "type": "model",
       "backgroundColor": "bgBlue"},
      {"id": "2", "type": "context-percentage-usable",
       "backgroundColor": "bgBrightRed"},
      {"id": "3", "type": "session-clock",
       "backgroundColor": "bgBrightBlack"} ],
    [ {"id": "4", "type": "current-working-dir",
       "backgroundColor": "bgBrightWhite"},
      {"id": "5", "type": "git-branch",
       "backgroundColor": "bgBrightMagenta"},
      {"id": "6", "type": "git-changes",
       "backgroundColor": "bgBrightCyan"} ]
  ],
  "overrideForegroundColor": "black",
  "globalBold": true,
  "defaultPadding": " ",
  "powerline": {
    "enabled": true, "separators": [""],
    "theme": "nord-aurora", "autoAlign": true
  }
}
```

- Order in the list is order on the bar
  - Moving a widget moves one list element; every id must be unique or the file is rejected
- A theme overrides your per-widget colours
  - Set colours first; then decide whether the theme wins
- separators holds a private-use glyph
  - U+E0B0 needs a patched Nerd Font; otherwise it is a box

The arrow is a font, not a character

U+E0B0 sits in Unicode's private-use area — only a patched font draws it

Install JetBrainsMono Nerd Font on the OS that draws the terminal

```
# macOS
brew install --cask font-jetbrains-mono-nerd-font
# Windows terminal host
winget install --id DEVCOM.JetBrainsMonoNerdFont
# Linux terminal host
NF=https://github.com/ryanoasis/nerd-fonts/releases/latest/download
mkdir -p ~/.local/share/fonts && cd ~/.local/share/fonts
curl -fLO $NF/JetBrainsMono.zip           # download the font archive
unzip -oq JetBrainsMono.zip && fc-cache -f  # unpack, refresh font cache
```

- Install on the terminal host, not merely inside the shell
  - A native Linux terminal uses Linux fonts; Windows Terminal, WSL, and MSYS2 use fonts installed on Windows

Install the font, then select it

Choose the Nerd Font in your terminal profile, then test the glyphs

One decisive check

```
# In bash or zsh after selecting the font in the terminal profile:
printf '\ue0b0 \uf07b\n'      # an arrow and a folder, not two boxes
```

- The menu may say JetBrainsMono Nerd Font or JetBrainsMono NF
  - Both are the patched font; its Mono and Propo siblings target editors and proportional layouts
- WSL and MSYS2 draw with Windows fonts
  - Install and select the font on Windows. Installing a font inside Ubuntu or MSYS2 does not change Windows Terminal
- A box means the terminal is still using an unpatched font
  - Ubuntu’s fonts-jetbrains-mono package is unpatched and does not contain U+E0B0

Every widget can drop its label

rawValue: true keeps the value and throws away the name

```
default — every widget names itself
Model: Opus | ⎇ main | Ctx(u) Used: 50.1%
```

```
the same three widgets with rawValue: true
Opus | main | 50.1%
```

Labels are for a bar you are still learning. Once you know the positions, the labels are just width you could spend on another widget.

87 widgets ship with it, in families

You are unlikely to want more than six at once

```
five of the git family
⎇ main | e91b745 | (+18,-4) | ?:2 | (no upstream)
```

```
four of the token and cost family
Ctx: 80.2k | Ctx(u) Used: 50.1% | Cache Hit: 77.7% | Cost: $1.20
```

Family	Examples	Reads from
Session	model, session-cost, context-percentage-usable	the JSON on stdin
Repository	git-branch, git-changes, git-sha	git, in the cwd
Account	session-usage, weekly-usage, reset-timer	your credentials
Custom	custom-text, custom-symbol, custom-command	you

One widget runs a program you wrote

custom-command is the extension point, and the sharp edge

Which machine am I on?

```
{
  "type": "custom-command",
  "commandPath": "short-hostname",
  "timeout": 1000
}
```

It runs on every prompt

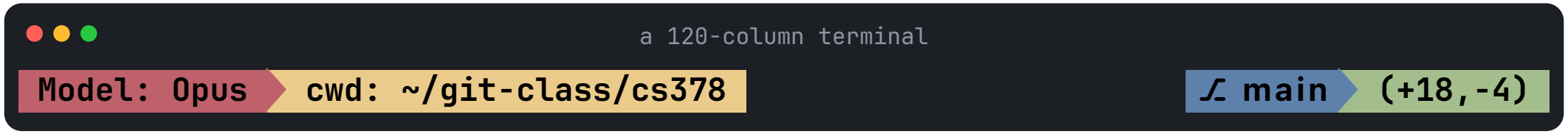
Hundreds of times an hour. A command that takes 300 ms costs you 300 ms of latency each time, and one that hangs freezes the bar.

Set timeout and mean it. Deck 928 is the story of what this widget did to a laptop on the day nothing bounded it.

The bar knows how wide your terminal is

flexMode decides how much of that width it is allowed to use

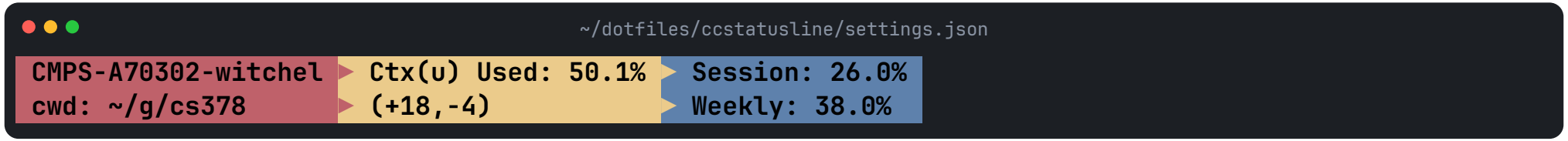
flexMode defaults to full-minus-40, which reserves 40 columns for Claude Code's own right-hand content. A flex-separator widget absorbs whatever is left over.



Same file, narrower window: 68 − 40 leaves 28 usable columns, so the git widgets are dropped rather than wrapped. Widgets you actually need belong early in the list.

What I actually run

~/dotfiles/ccstatusline/settings.json, symlinked into ~/.config



Line 1 — am I about to run out of something?

```
[ {"type": "custom-command",
    "commandPath": "short-hostname",
    "backgroundColor": "bgGreen"},
  {"type": "context-percentage-usable",
    "backgroundColor": "bgBrightRed"},
  {"type": "session-usage",
    "backgroundColor": "bgBrightBlack"} ]
```

Line 2 — where am I, and what have I changed?

```
[ {"type": "current-working-dir",
    "metadata": {"fishStyle": "true"},
    "backgroundColor": "bgBrightWhite"},
  {"type": "git-changes",
    "backgroundColor": "bgBrightMagenta"},
  {"type": "weekly-usage",
    "backgroundColor": "bgBrightCyan"} ]
```

Six widgets, one question each: which machine, how much context is left, what this session spent, where I am, what is uncommitted, and how much of the week’s budget is gone.

Build your own in five minutes

Then commit the file — it is the smallest dotfile you own

Step	Do this	Check
1	Install JetBrainsMono Nerd Font, select it in your terminal	printf ‘\ue0b0\n’ prints an arrow, not a box
2	Run npx ccstatusline@latest and add 4–6 widgets	The TUI preview updates as you move
3	Point Claude Code at it: statusLine in settings.json	The bar appears under your next prompt
4	Move the file into your dotfiles repo, symlink it back	ls -l on the config path shows the link
5	Give every custom-command widget a timeout	No widget can outlive one prompt

Every picture in this deck came from running the tool: see slides/slide-images/931_tool_ccstatusline/capture.py, which renders each config shown here against a fixed repository and session.