

CLAUDE COMMUNITY EVENT · BERLIN

Build agents that debug and review with Claude Code

Part 1: an AI SRE that finds the root cause.

Part 2: a software factory that ships a feature.

Thu 22 Oct 2026 · 18:00-21:30 · P.19 Studio, Prinzessinnenstr. 19

workshop-berlin.glassflow.ai



TONIGHT

Two parts, both on Claude Code and Tares

18:40

Part 1: Build an AI SRE

Two agents watch a live service and find the root cause when you break it.

About 45 minutes

19:45

Part 2: Software factory

Set up a crew of Claude Code sessions that turns a plan into merged pull requests.

About 30 minutes

21:00

Show and tell

What your agents found, and what your crew shipped.

20:15 short break · 21:30 networking

BEFORE YOU START

Pick local or cloud, then open the guide

Local

Tares runs on your laptop. You did the setup page at home: Tares, Claude Code, and Docker for Part 1.

Part 1 needs your own Anthropic API key.

Cloud

A Tares workspace we host, with \$10 of model credit. Join with the link we hand out tonight.

No Docker, no API key.



Docker or the Wi-Fi giving you trouble? Switch to cloud. Stuck on setup? Come to the help desk.

Setup: `workshop-berlin.glassflow.ai/setup`

PART 1 · 18:40

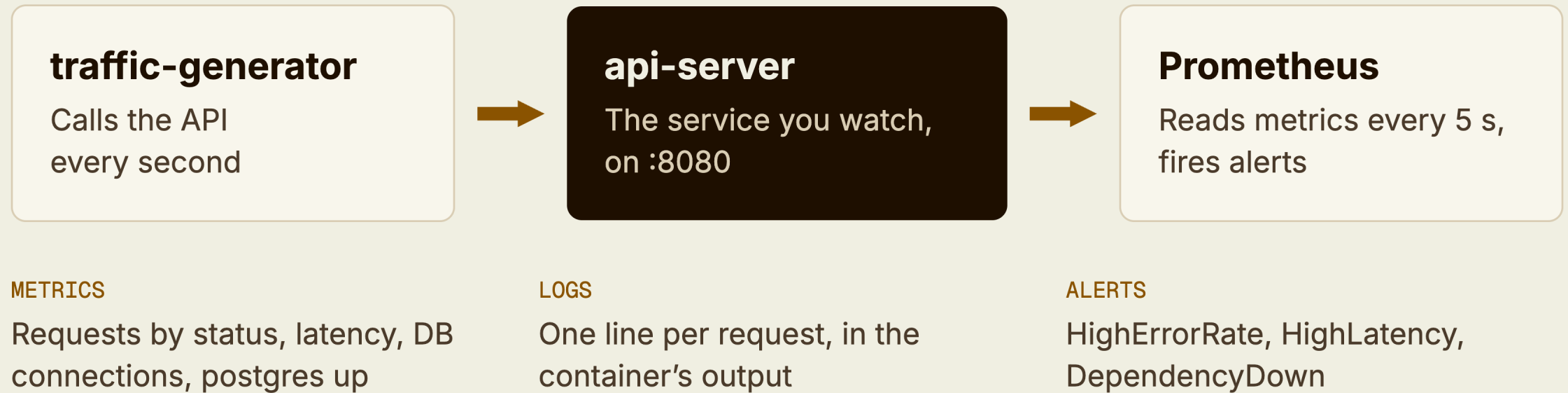
Build an AI SRE with a handover agent

Two agents watch a live service. You break it.
They find out why.

About 45 minutes · workshop-berlin.glassflow.ai/ai-sre



A small service that is always busy



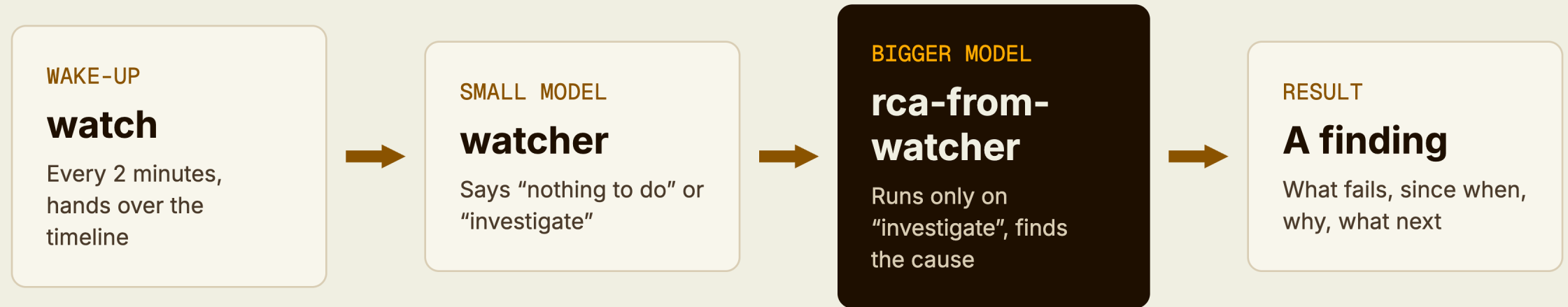
```
./inject.sh error_spike · latency · dependency_outage · clear
```

Three sources, one timeline



The AI SRE demo template creates all three sources for you

A watcher that hands off to a root-cause agent



You make all three in the Tares console: the root-cause agent first, then the wake-up, then the watcher with its handoff.

Cheap model always on. Big model only when it matters.

2–4 min

from breaking the service to the root-cause finding

\$0.07

in model calls for our whole walk of this guide

Most runs

the watcher says “nothing to do” and the big model never starts

13 steps in four stages

STEPS 1-3

Start

- The demo service
- Tares
- The MCP endpoint

STEPS 4-7

Set up

- Add your API key
- Start the demo project
- Turn off its own agent
- Connect Claude Code

STEPS 8-10

Build

- The root-cause agent
- The 2-minute wake-up
- The watcher, with its handoff

STEPS 11-13

Break it

- Wait for a quiet run
- `./inject.sh error_spike`
- Compare with Claude Code

Every step says what you should see. If you don't see it, check "If something is off" at the end of the guide.

Swap the sources, keep the agents

1. Your sources

Prometheus, Docker logs or OpenTelemetry, your alerts. One label per service.

2. Your deploys

One webhook call from your pipeline. Most incidents start with a change.

3. The wake-up

Point it at your sources. Tell the watcher what normal looks like.

4. Where findings go

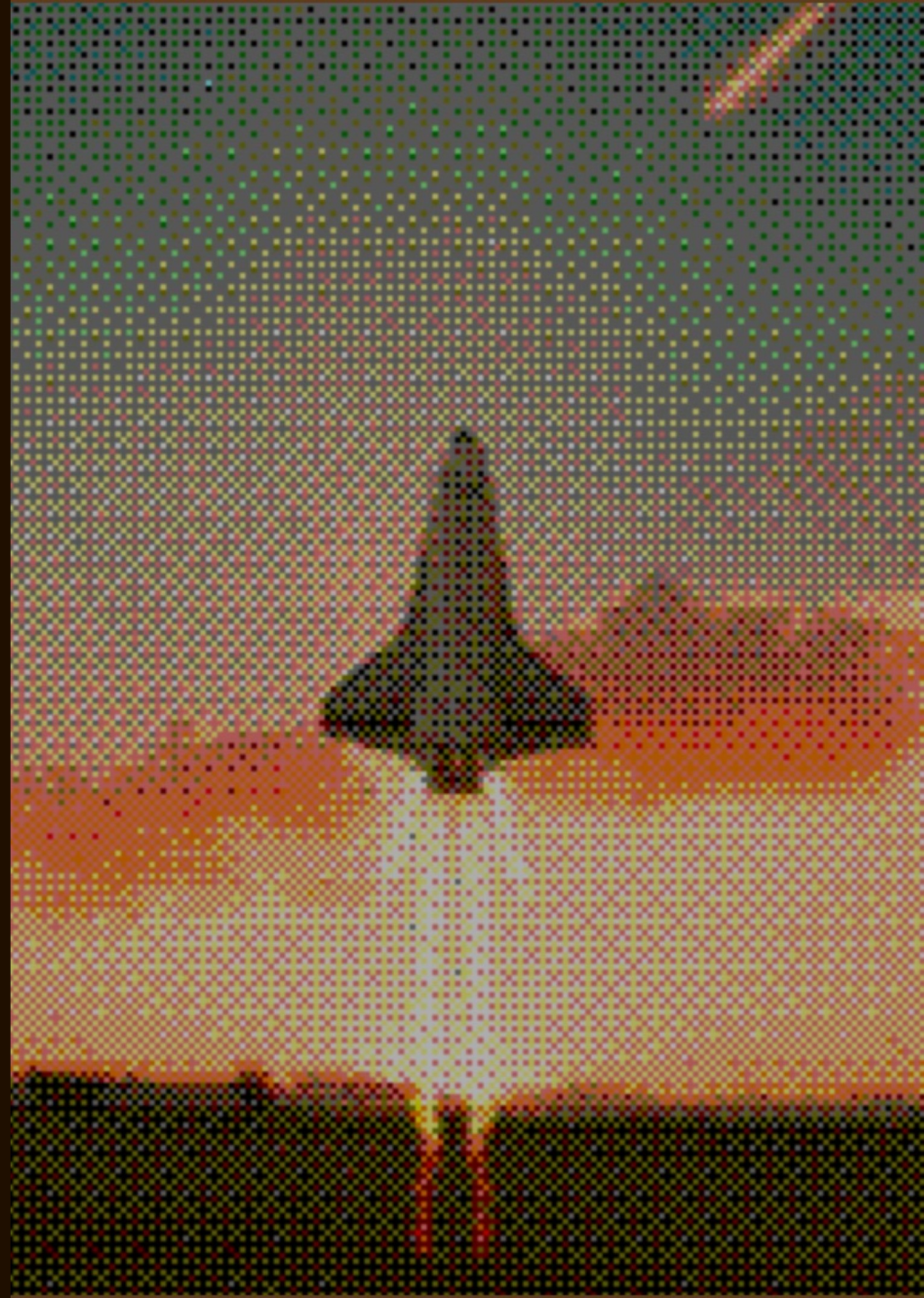
The project page, or a Slack channel your team already reads.

PART 2 · 19:45

Software factory

Set up a crew of Claude Code sessions that plans, builds, reviews and merges. Then give it a first job.

About 30 minutes · workshop-berlin.glassflow.ai/software-factory



A software factory on your laptop

THE FACTORY

A crew of Claude Code sessions that turns a plan into merged pull requests

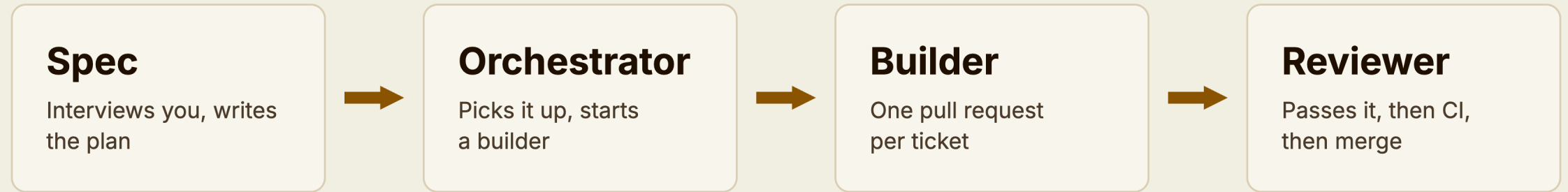
- An orchestrator, builders and a reviewer
- Tares holds the plan, tickets, reviews and your answers
- Runs on your own Claude Code plan, on any repo

ITS FIRST JOB: A TEST

A toy link shortener, and a click counter to add: two tickets.

Small on purpose, so you see the whole loop, from plan to merge, in 30 minutes.

Four Claude Code sessions, one job each



Tares keeps the plan, the tickets, the reviews and your answers. Every session reads and writes there.

You answer three times. The crew does the rest.

1

The interview

Four questions, each with a suggested answer. Add “two tickets at most”.

2

The hand-over

“Hand this to the orchestrator now?” Say yes, then close the spec session.

3

“Is it done?”

After both merges, answer the orchestrator.

Start the crew	Plan	First merge	Second merge
2 min	7 min	+7 min	+5 min

Times from our test run · short on time? Use the ready-made spec

13 steps in four stages

STEPS 1-2

Your repo

- Copy the starter
- Try the app as it is

STEPS 3-6

Tares

- tares up
- The MCP endpoint
- tares factory doctor
- A GitHub token

STEPS 7-9

The crew

- Start it, 1 builder
- Plan the feature
- Hand it over

STEPS 10-13

Watch it ship

- Console, GitHub, sessions
- Say it is done
- See what Tares kept
- Try the feature

Pick 1 builder, not the suggested 2: it is enough for two tickets and saves your plan's limits.

The same crew works on your own repos

Finished early?

Plan the README's next idea, from step 8: link expiry or tags. The crew is still up.

Your own repo

Run tares factory spec in it, with a real idea. One crew handles all your projects.

Bigger projects

Several milestones, Linear tickets, releases. See a project followed end to end in the docs.

Stop it

tares factory crew down. Tares keeps the project; crew up brings it back.

21:00 · SHOW AND TELL

What did your agents find? What did your crew ship?

Two minutes each: show the finding or the merged pull requests.

workshop-berlin.glassflow.ai

docs.glassflow.ai/tares

github.com/glassflow/tares