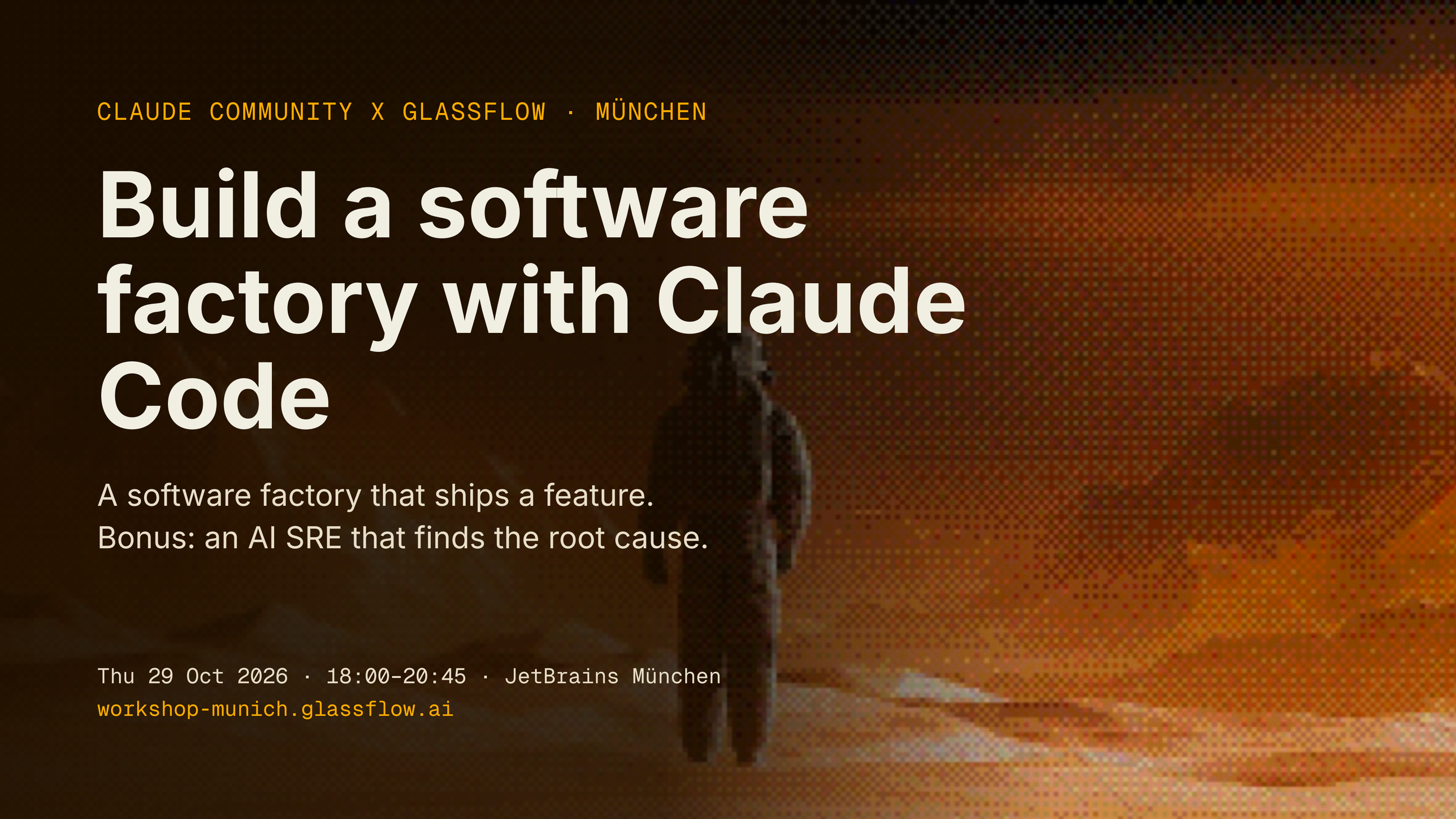

CLAUDE COMMUNITY X GLASSFLOW · MÜNCHEN
Build a software factory with Claude Code
A software factory that ships a feature.
Bonus: an AI SRE that finds the root cause.
Thu 29 Oct 2026 · 18:00–20:45 · JetBrains München
workshop-munich.glassflow.ai

YOUR HOSTS TONIGHT

# Ask us anything, tonight or after



## Armend Avdijaj

Founder & CEO, GlassFlow

armend@glassflow.dev
linkedin.com/in/armend-avdijaj
x.com/arm1end



## Kiran Nambiar

Senior Backend Engineer, GlassFlow

kiran@glassflow.dev
linkedin.com/in/kiran-nambiar

glassflow.ai

GLASSFLOW

# Data infrastructure for AI agents

Rius

## Fix agent mistakes before your users see them

glassflow.ai/rius

Tares

## Build, run and orchestrate agents with the latest context

github.com/glassflow/tares

RIUS · THE PROBLEM

# Agents fail silently, and say “done”

**11%**

of about 10k runs ended with the agent confidently claiming success. A manual check showed it was wrong.

Advani, “From Confident Closing to Silent Failure”, arXiv 2606.09863, June 2026

### FALSE SUCCESS

“Your ticket has been created.” No ticket exists.

### HALLUCINATION

“Refunds are allowed within 90 days.” No such policy.

### ENDLESS LOOP

“Still working on it.” Same call for hours.

### CUT OFF

“Here is the summary.” Half the document was never read.

### CONTEXT POISONING

One bad tool result, and every step after builds on it.

### QUIET DRIFT

Same output, slowly worse. Nothing errored.

RIUS · WHAT IT DOES

# Catch failures before your users do

Observability tools show that a run happened, how long it took and what it cost. Rius checks whether what the agent said actually happened.

01

## Collect

Collects the traces of every agent run.

02

## Detect

Detects failures, the loud ones and the silent ones.

03

## Investigate

Investigates and tells you the root cause.

04

## Hand off

Shares it with your coding agent to act on it.

TARES · HOW IT WORKS

# One platform, ready-made templates

Tares is the platform. A template is a ready-made set of agents that runs on it.

TEMPLATE · PART 1

## Software Factory

A crew of Claude Code sessions with fixed roles that ships a feature.

TEMPLATE · BONUS

## AI SRE

Two agents that watch a service and find the root cause.

PLATFORM · TARES

Build, run and orchestrate agents · memory that outlives any session · a place to coordinate · the latest context, over MCP

github.com/glassflow/tares · docs.glassflow.ai/tares

TONIGHT

# Two builds, both on Claude Code and Tares

01

## Set up the Software Factory

The Tares template: a crew of Claude Code sessions that turns a plan into merged pull requests.

02

## Investigate failures

Find where your crew went wrong, why, and what the reviewer caught.

03

## Bonus: the AI SRE template

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

Zoom build session

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 the bonus build.

**The bonus build 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? Ask in Slack or ask one of us.

**Join the Slack**

Questions and help, tonight and after

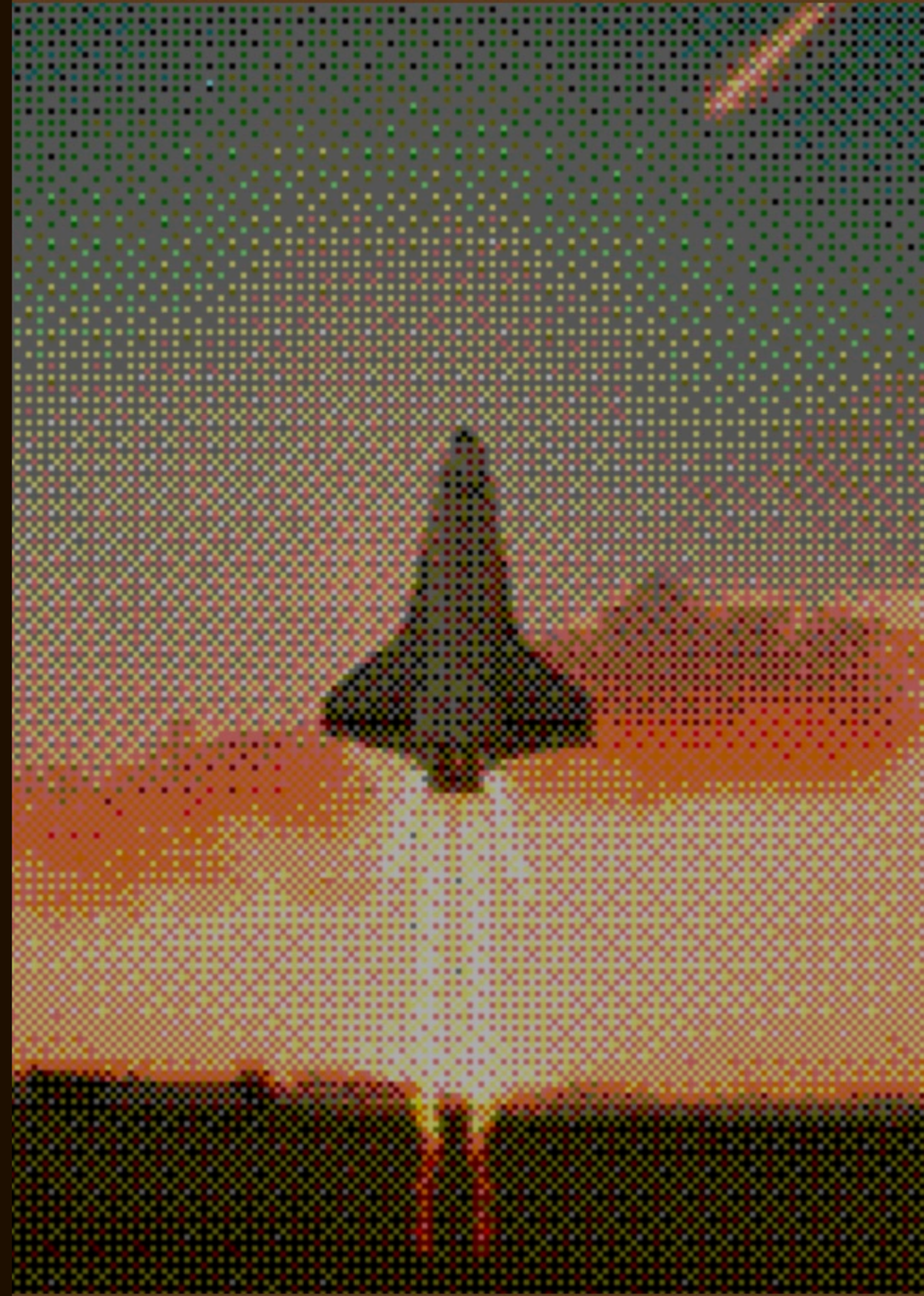
Setup: workshop-munich.glassflow.ai/setup · All links: workshop-munich.glassflow.ai/links

PART 1

# Software factory

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

workshop-munich.glassflow.ai/software-factory



PART 1 · WHAT YOU SET UP

# The Software Factory template, on your laptop

A TARES TEMPLATE

## A crew of Claude Code sessions with fixed roles, turning a plan into merged pull requests

- An orchestrator, builders and a reviewer, on your machine
- 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.

The test job: github.com/glassflow/factory-workshop-starter

PART 1 · YOUR PART

# You answer three times. The crew does the rest.

1 · THE INTERVIEW · 4 QUESTIONS

**The goal and the project name?**
Accept the suggestion.

**What does the stats endpoint return?**
Accept, and add: `keep it small: two tickets at most, one milestone`

**Who builds it?**
Pick the tares-factory crew.

**Where do the tickets live: Linear or Tares?**
Pick Tares.

2 · THE HAND-OVER

## “Hand this to the orchestrator now?”

Type yes, then close the spec session.

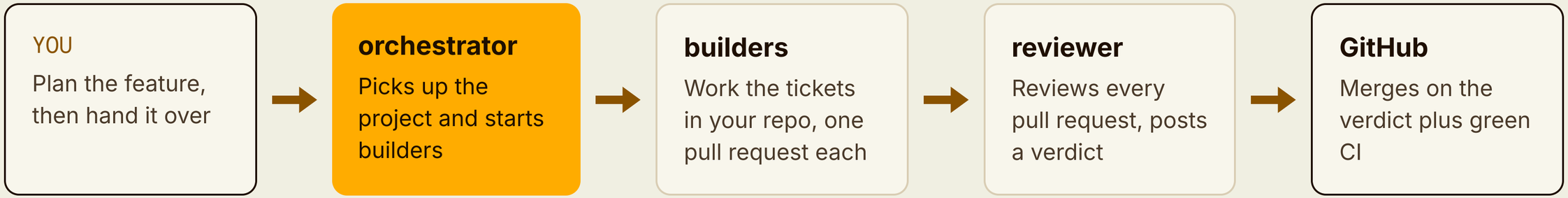
3 · THE CHECK

## “Is it done?”

After both merges, type A. If it asks to keep an assumption, A again.

PART 1 · ARCHITECTURE

# The orchestrator runs the crew. Tares keeps the record.




↑ every session reads and writes Tares over MCP ↓

TARES · THE RECORD UNDER THE CREW

Holds the spec, tickets, reviews and your answers · sessions pick up where they stopped · one console at /crew

PART 1 · WHY A TEMPLATE

# Five things break when you run Claude Code by hand

01

## It forgets its work

A restart or a full context wipes what it knew.

TARES

The work lives in Tares. The session picks up where it stopped.

02

## It stops, and nobody notices

A session that waits too long gets shut down. Messages to it are lost.

TARES

Restarts it under the same name.

03

## Two sessions do the same job

Both review or merge the same pull request.

TARES

Each action has exactly one owner.

04

## It merges without review

Or the pull request sits and waits on you.

TARES

The reviewer’s verdict plus green CI decides.

05

## It waits on you for hours

One permission prompt blocks the whole session.

TARES

The request lands on your desk. One click approves it for good.

PART 2 · INVESTIGATE FAILURES (CLOUD)

# Now imagine you go to prod

Many things can happen. Loud failures that throw an error, and silent ones that don’t. When it happens, you want three answers.

01 · WHAT

**What failure happened?**

02 · WHY

**Why did it happen?**

03 · IMPACT

**What did it hit, and how badly?**

PART 2 · THE SILENT FAILURE WE SHOW

# Bloated context

01

## You add an MCP

It looks useful. Its tools return far more than the agent needs.

02

## One call floods the context

The context window jumps. No error, and the agent keeps going.

03

## Answers quietly get worse

Less room to think, more noise to read. Nobody notices.

BONUS

# 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-munich.glassflow.ai/ai-sre



BONUS · NOT DONE YET?

# No worries: we build it together on Zoom

Didn’t finish tonight, or didn’t get to the bonus? That’s fine.

THE ZOOM BUILD SESSION

**We send everyone the Zoom invite in Slack, and build the AI SRE together, step by step.**



## Join the Slack

The Zoom invite lands here

BONUS · THE SCENARIO

# A busy service, and you break it on purpose

**traffic-generator**

`the user`

Calls the API every second

→

**api-server**

`the backend`

Orders, stats and users, on :8080

→

**Prometheus**

Reads its metrics, checks three alert rules

THE FAULT SWITCH · `./inject.sh`

**`error_spike`**

40% of requests return 500

**`latency`**

Every request takes 1.5 s

**`dependency_outage`**

Postgres is down, every request returns 503

`./inject.sh clear` puts it back

BONUS · WHAT TARES DOES

# From fault to finding in 2 to 4 minutes

01

## Collect

Metrics and error logs every 5 s, alerts every 10 s, on one timeline for api-server.

02

## Wake

Every 2 minutes, a trigger hands the timeline to the watcher.

03 · HAIKU

## Decide

Firing alert, p99 over 50 ms or postgres down? “Investigate.” Else “nothing to do”.

04 · SONNET

## Explain

The root-cause agent writes what fails, since when, the likely cause and a next step.

WHERE IT LANDS

On the project page, as a “Needs action” finding

HOW YOU CHECK IT

Ask Claude Code. It reads the same timeline over MCP.

LIVE WALKTHROUGH

# What it looks like when it all works

01 · TARES

## A finished run, and what else Tares can do

Merged PRs, plus the plan, tickets and reviews in the console.

02 · RIUS

## Investigating the bloated MCP

The context jumps, nothing errors. We trace it back to the call.

github.com/glassflow/tares · glassflow.ai/rius

WHAT'S NEXT

# The software factory, for teams and enterprises

Tonight you ran a crew on your laptop. Next, we take it to whole teams and enterprise use.

github.com/glassflow/tares

Q&A

# Questions? Ask us anything



## Give us feedback

Two minutes, and it shapes the next one

BUILDING SOMETHING MORE COMPLEX?

## Reach out, we'll help you build it

**Slack** · the GlassFlow community

**LinkedIn** · linkedin.com/in/armend-avdijaj

**Email** · armend@glassflow.dev

**Or right now** · come talk to us

All links: workshop-munich.glassflow.ai/links