Green Checkmarks, Red Flags: What CI/CD Can't Catch

Tilda Udufo

March 29, 2024: Unusual CPU usage detected

```
$ ssh debian-sid
# CPU spikes unexpectedly
```

Tests Passing

CI Green

Reviewed

About Me

TILDA UDUFO

Software Engineer. Program Organizer @Outreachy

What this Talk is About

What

CI/CD can show green pipelines while fragile assumptions, risky changes, or subtle regressions still slip into production.

Why

Automation enforces rules we already know — but it can't reason about intent, context, social dynamics, or overloaded reviewers, especially as systems and teams scale.

How

Pair automation with human safeguards: structured reviews, risk checklists, staged rollouts, and a culture that rewards slowing down when something feels uncertain.

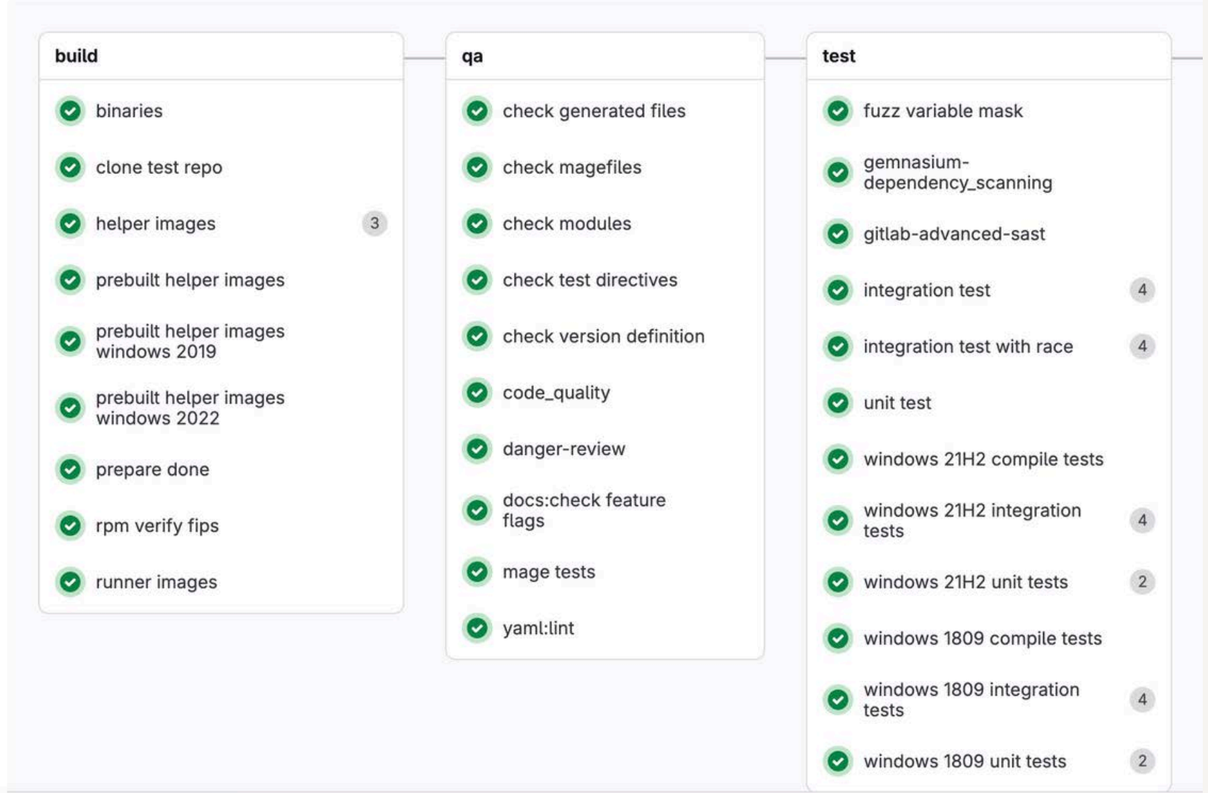
A Familiar Scenario

Green tests

Successful Builds

Successful Deployments

Users still have issues



The XZ Utils Backdoor

Timeline

Late 2021: Contributor appears

Late 2022-2023: Influnce grows, becomes a maintainer and releases version

Feb 2024: Backdoor introduced into release

Mar 29, 2024: Backdoor found and disclosed

What Was Bypassed?

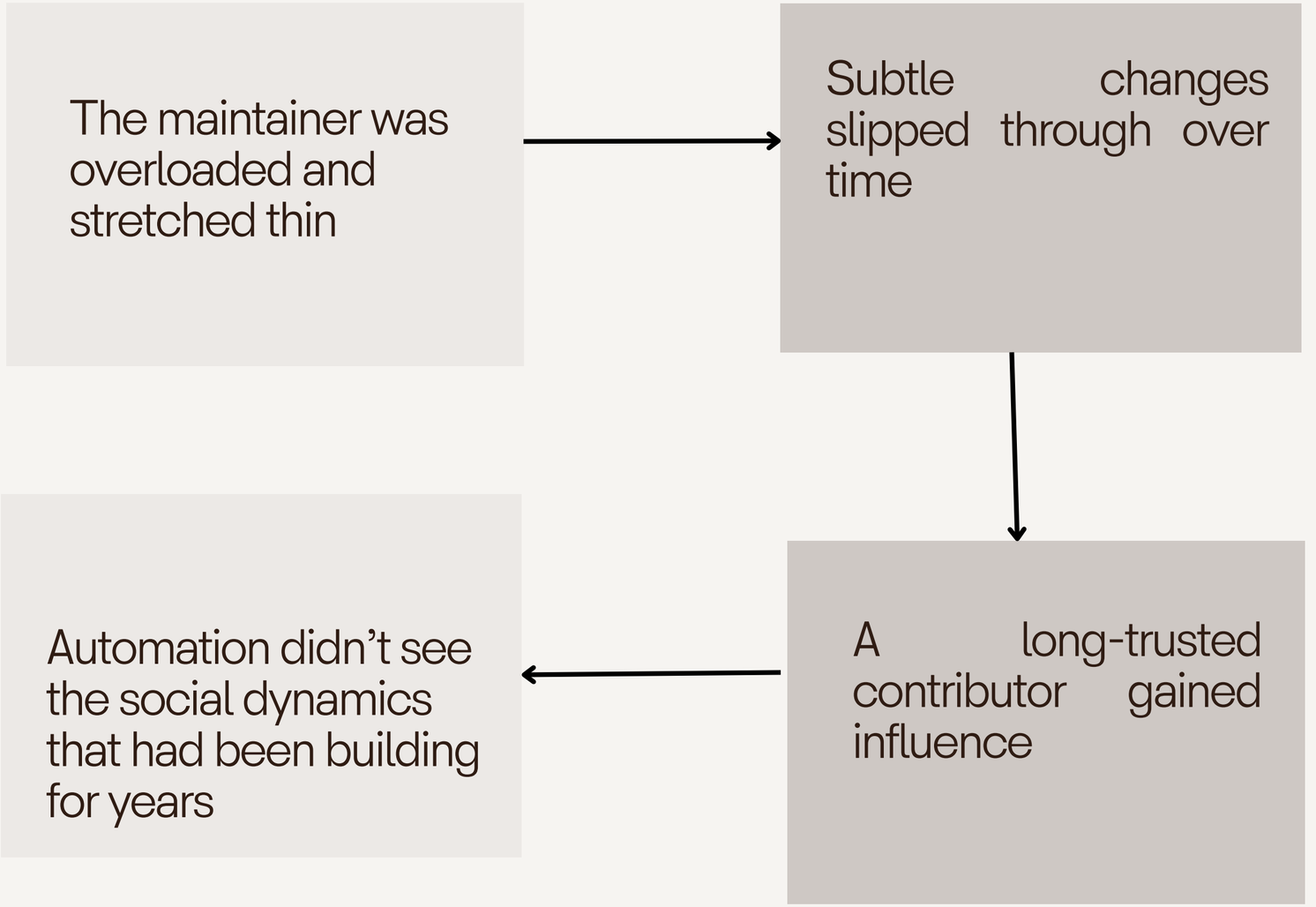
Tests and CI passed

Changes looked reviewable

Release artifacts contained hidden behaviour

Approved and shipped

The XZ Utils Lesson



Why We Over-Trust CI/CD

Green checks feel like certainty

Teams scale faster than review capacity

Deadlines push speed over caution

“If CI passed, it must be fine… right?”

CI/CD Is Powerful — But Limited

CI/CD answers:

“Does this code meet the rules we defined?”

It does not answer:

“Is this change safe in the real world?”

What CI/CD Is Good At

1. *Repeatable, reliable builds*
2. *Consistent checks and linting*
3. *Automated testing*
4. *Faster deployments*
5. *Reduced human error*

What CI/CD Can't Catch (By Design)

1. *Social engineering*
2. *Subtle performance regressions*
3. *Integration behavior across systems*
4. *UX and workflow mismatches*
5. *Risky dependency changes*
6. *Misconfigurations that look harmless*

Trust Breaks at Scale

Personal trust works

Semi-known

Trust Broken

Why These Risks Escalate

1. *More contributors = more assumptions*
2. *Automation hides fragile processes*
3. *Review becomes rubber-stamping*
4. *Trust grows faster than oversight*

Systemic Pressures

1. *Review queues pile up*
2. *Approvals happen under time pressure*
3. *“Just ship it” becomes the norm*
4. *Maintenance becomes reactive*

Fragile Processes

1. *Single reviewers for critical code*
2. *Unwritten “tribal knowledge”*
3. *Informal rules about approvals*
4. *No ownership for risky areas*

Designing Safeguards Alongside CI/CD

Stronger Reviewer Practises

1. *Second reviewer for high-risk areas*
2. *Rotate reviewers to avoid power concentration*
3. *Use risk checklists before merging*
4. *Separate fast lane vs slow lane PRs*

Cultural Safeguards

1. *Normalize: “I don’t understand — explain?”*
2. *Reward caution, not just speed*
3. *Document decisions openly*
4. *Pair review tricky changes*

Before You Merge: Risk Reality Check

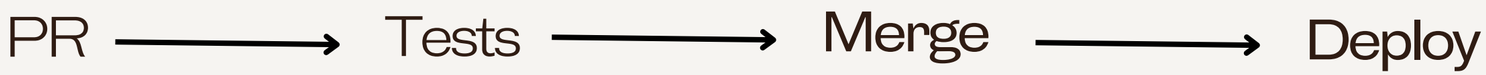
- [x] Does this touch auth or infrastructure?
- [x] Any new dependencies added?
- [x] Hard-to-test code pattıs changed?
- [x] Is rollback simple?
- [x] Is the reviewer new to this area?
- [x] Dees this rely on "tribal knowiedge"?
- [x] Are we merging under time pressure?
- [x] Whot could go wrong if we're wrong?

Strengthening CI Beyond Tests

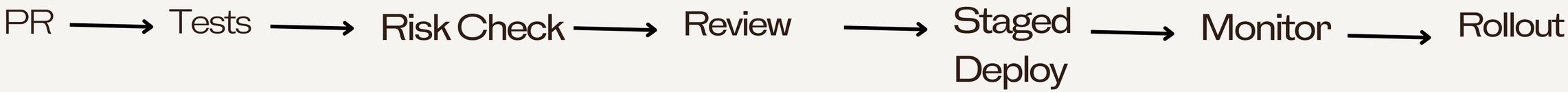
1. *Canary deployments*
2. *Post-deploy smoke checks*
3. *Monitoring tied to releases*
4. *Dependency anomaly alerts*
5. *Error budgets and rollback triggers*

Putting It Together

Before:



After:



Key Takeaways

1. *CI/CD is necessary — but not sufficient.*
2. *Risk grows quietly as projects scale.*
3. *People and process fill the gaps automation can’t.*
4. *Safeguards don’t need to slow delivery*

Thank You!

LINKEDIN & X

Tilda Udufo