“Disorganizing” your SRE organization

Leonid Belkind
CTO and co-founder, StackPulse

“Disorganization can scarcely fail to result in efficiency”

Dwight D. Eisenhower
What I’m going to talk about today

1. The impacts of WFH on the teams responsible for reliability
2. How we ‘disorganized’ the SRE team to address
3. Lessons learned along the way
4. Suggested ‘Action Items’
What were our challenges?

- The agility and innovation in how we build and ship software has increased velocity and fragmented knowledge.

- Roles like SRE, collaboration tools like Slack, Confluence/Notion, and “shift-left” tooling that allows developers to build, test, deploy and monitor software services are widely adopted.

- When WFH became the new normal - this status quo was no longer enough.
62% of IT and DevOps practitioners are spending 10+ additional hours a week on incidents since COVID

Source: PagerDuty
Over 80% of SREs spend over half their time on operations - not engineering.

Source: Catchpoint
The state of the world

Our Journey: Biggest Challenges

→ Training / Onboarding team members in a growing team

→ Dealing with Increased Noise in a service with growing adoption

→ Limited informal communication
What does “disorganizing” the SRE organization mean?
1. Democratize responsibility to all engineers
2. Empower autonomous but consistent action
Democratizing responsibility

- Treat reliability like a feature and build for it in every task

- Train developer teams on monitoring/alerting, observability, error budgets, SLOs and incident response metrics like MTTD/MTTR

- Make every developer part of on-call work – leadership too
Empowering autonomy and consistency - why

→ People handling incidents should feel **empowered** to have all the relevant data and to take relevant remediation steps

→ It is much easier to ask for help when you are in a room with people, not so easy to reach out remotely

→ When Slacking / Zooming with people, it is harder to understand their underlying intentions / mood

→ “Remote” collaboration should be about tasks / facts / findings
Empowering autonomy and consistency - how

- Build playbooks for every workflow – never do the same thing manually twice
- Turn on-call / incident response into deterministic code – make available as “modules” to developers
- Common language and format for all playbooks – no exceptions
- Educate and train team with playbook artifacts vs. wiki articles
What does “turn into code” look like for us?

- **Declarative** playbooks/workflows
- **Encapsulated** process steps
- Four parts to each:
  - **Enrich** – append environment and application context, assess customer impact and assign severity
  - **Triage** – rule out possible causes, focus on suspicious signals
  - **Communicate** – open war rooms, create/update incidents, communicate with on-callers / stakeholders
  - **Remediate** – bring the service environment back to operating state
Our Journey: Getting Started

- **1 Month:** On-call and playbook writing spread across developers

- **3 Months:** Weekly review of incidents and their resolution metrics, outlining missing pieces and scheduling their development
Our biggest lessons learned were about the **human part of the process**
1. Accept the new normal
2. Build to the individual
3. Explicitly build culture
4. Terminate loops locally
Accept the new normal

➔ Trying to ‘keep everyone in the room’ with Slack, Zoom, Discord doesn’t work.

➔ Increased fatigue, poor responsiveness, low morale
Lessons Learned

Build to individual need

→ People have different preferences for interruptions, privacy, communication style

→ Work with individuals to find what’s right for them

→ Balance critical need with personal preference
Explictly build a new culture

→ Directly share that you’re working on culture as a project

→ Define changes in day-to-day responsibilities

→ Build opportunities for informal interaction that use different formats
Terminate loops locally

→ Re-divide responsibilities

→ Empower with playbooks as documentation

→ “4 eyes” verification only for critical issues

→ Measure performance, share with the team
Our Journey: 6 Months In

➔ Enrichment, RCA over 60% automated
➔ MTTR reduced by 35%
➔ Playbooks used to manage incidents from create to post-mortem.
➔ Over half the team has led incident response
Our Journey: 1+ Year In

➔ Playbooks as code allow uniform processes and don’t fail to deliver; ~85% of the steps are automated

➔ Examples:
  - Open Support Incident
  - Scale Services Up/Down
  - Run End-to-End Tests
  - Collect Logs / Extended Context
  - Rollback to Stable State
**Thoughts on how to get started**

1. Be open with your teams. **Explicitly explain that the organization is embarking on a journey** (to change its culture)

2. **Identify individuals that are passionate about it** and involve them in leading the efforts

3. Let the teams drive choices of automation tools. **Technologists enjoy solving problems with tools much more than they do with manual processes.** Tools do matter

4. Don’t assume that people will tell you how they feel or how confident they are. **Constantly monitor the “soft” metrics**
Thank You!

Questions?
leonid@stackpulse.com