Best Practice for On-Call/”Pager” Rotations in the Dev-Ops Model

- By Joe Huffer
- CSD380
- 12/1/2024

Intro – Pager Rotation and It’s Importance

- Pager rotation, also referred to as “on-call”, is a way to ensure that incidents are continuously monitored and resolved promptly by a designated team or team member. It’s integral to ensuring service reliability and operational efficiency in a DevOps model.
- Why is it important?
  - It reduces downtime by ensuring timely incident responses.
  - It distributes workloads evenly to prevent burnout among engineers.
  - Enhances overall service reliability and custom satisfaction.

Benefits of a Strong Pager Rotation System

Employees are Hopefully Satisfied!

- A well-maintained on-call schedule helps handle after-hour needs of clients and can improve retention of clients and users.

Improves Trust and Reliability of Technical Services Provided

- Faster and constant response to incidents – which then minimizes downtime.

Transparency and Accountability

- Clear schedules and roles prevent miscommunication during incidents.
- The intent is to build a culture of ownership and responsibility, fairly.

Best Practice 1 – Implementing DevOps Principles

What this means in terms of Pager Rotation:

Developers will own the services they build, driving them to create resilient and long-standing systems.

Breaks down silos between development and operations for smoother collaboration and faster processing times.

Examples of Implementation:

Cross-functional teams with consistent ticket closing times (SLAs where I work)

Developers receive training to support their own services in production.

The Outcome, Optimistically, is a stronger, more unified team and improved product reliability, as well as improved confidence that issues will be maintained at all times.

Best Practice 2 – Scheduling and Automating Rotations

Tips for Scheduling I’ve Found:

- User tools like Jira, Confluence, Email Systems (we use Outlook/M365), Slack, and many other tools.
- It’s important to map out that structure and time where each individual is “on-call”, but it’s also important to hear your team and navigate/listen to any anxieties they may have.

Bonus Info: Automation

- Notifications via email/text can streamline and verify alerts are received.
- Predefined rotations can help reduce human error and ensure engineer engagement.

Best Practices 3 – Set Clear Escalation Policies

Why Escalation Matters:

- So many reasons, one being that it alleviates confusion and sets a clear chain of command during incidents.
- It also reduces response time by ensuring the right people are contact initially, and that the right people are able to be engaged eventually.

How to Implement:

- Define specific roles and responsibilities for each escalation tier.
- Set time-bound escalation triggers to avoid delays or breaching of SLA’s.

Challenges and Solutions

Some Challenges Teams Have and Might Face Could Include:

- Frequent after-hours pages, causing potential burnout.
- Coordinating responses across multiple teams, could cause delays.

Potential Solutions

- Monitor non-business hour alerts and adjust schedules accordingly.
- Automate Escalations and try to integrate tools that help notify/monitor to create a network of seamless collaboration.

Conclusion

- To wrap up, a strong pager – or “on-call” – system is critical to maintain reliability and for development teams to operate efficiently. Following best practices ensures timely responses, reduces burnout, and fosters a more collaborative work environment!

Source Cited:

- AlertOps. "Best Practices for Managing On-Call Rotation (in 2023)." *AlertOps*, 2023, https://alertops.com/on-call-rotation/.
- Cortex. "Best Practices for On-Call Rotations." *Cortex*, 13 Nov. 2024, https://www.cortex.io/post/best-practices-for-on-call-rotations.
- *Confluence*. Atlassian, https://www.atlassian.com/software/confluence.
- *Jira*. Atlassian, https://www.atlassian.com/software/jira.
- *Microsoft 365*. Microsoft, https://www.microsoft.com/en-us/microsoft-365.