SRE NEXT 2022

Sensible Incident Management for Software Startups

Takayuki Watanabe
@Launchable, Inc.

Who?

```
Name: Takayuki Watanabe
Affiliation: Launchable, Inc.
Role: Software Engineer
Sns:
  Blog: blog.takanabe.tokyo
  GitHub: takanabe
  Twitter: @takanabe_w
Interests:
  - Developer Productivity
  - Site Reliability Engineering
  - Sustainability Engineering
```

Your takeaways

You can understand:

- Incident management has a life cycle.
- Incident response roles and structures exist to embody 3T mental models.
- Choosing strategies and tools makes incident managements at startups sensible.

Out of scope

- Fundamental SRE terminology (e.g. SLO, SLI, Error budget, Postmortem)

Disclaimer

- This session refers a lot of existing incident management and SRE practices.
- But contains a lot of opinionated ideas and philosophy as well.
- So, the ideas might contradict to some people's.
  - Let's discuss on Twitter using #srenext with @takanabe_w

Today's agenda

- About Launchable
- Does a startup need incident management?
- Dissect incident management practices.
- 3T mental models and life cycles
- How can we improve incident management?
- Choosing right strategies and tools

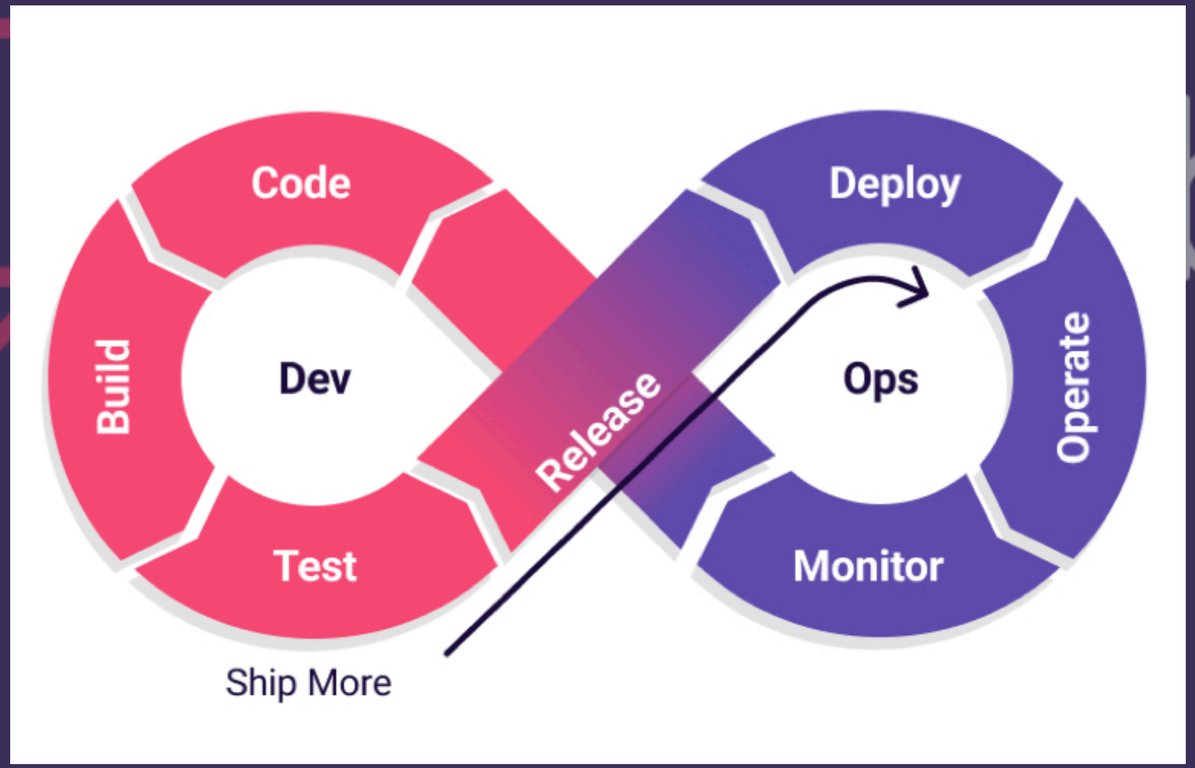
Chapter 1:

About Launchable

Launchable

What is Launchable?

A SaaS accelerating software development cycles.



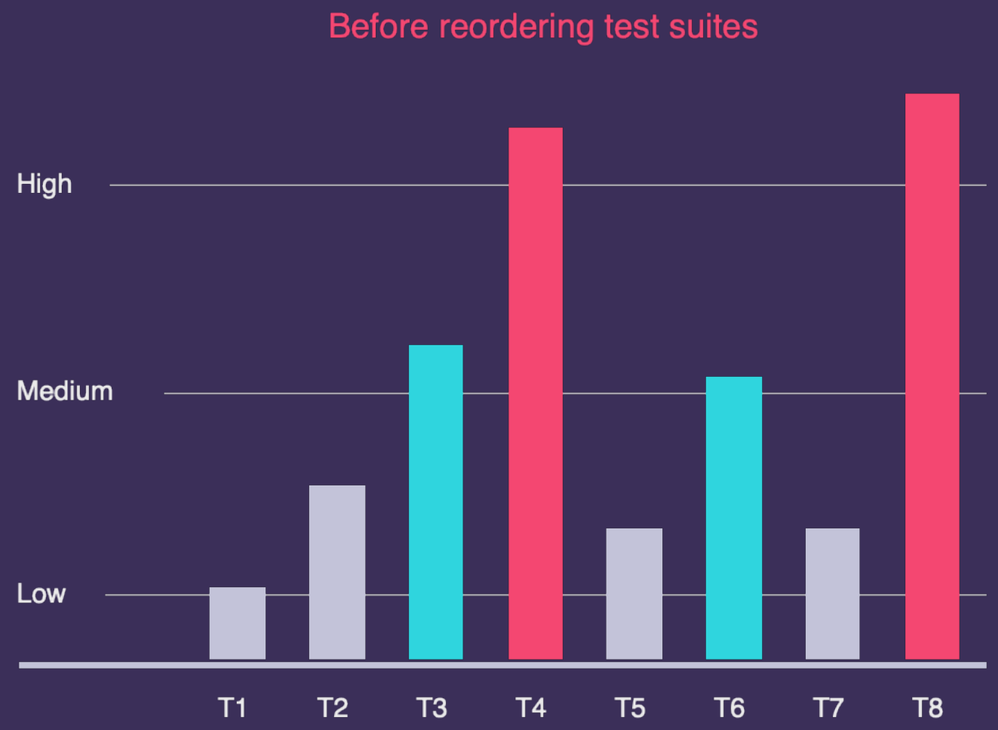
What is Launchable?

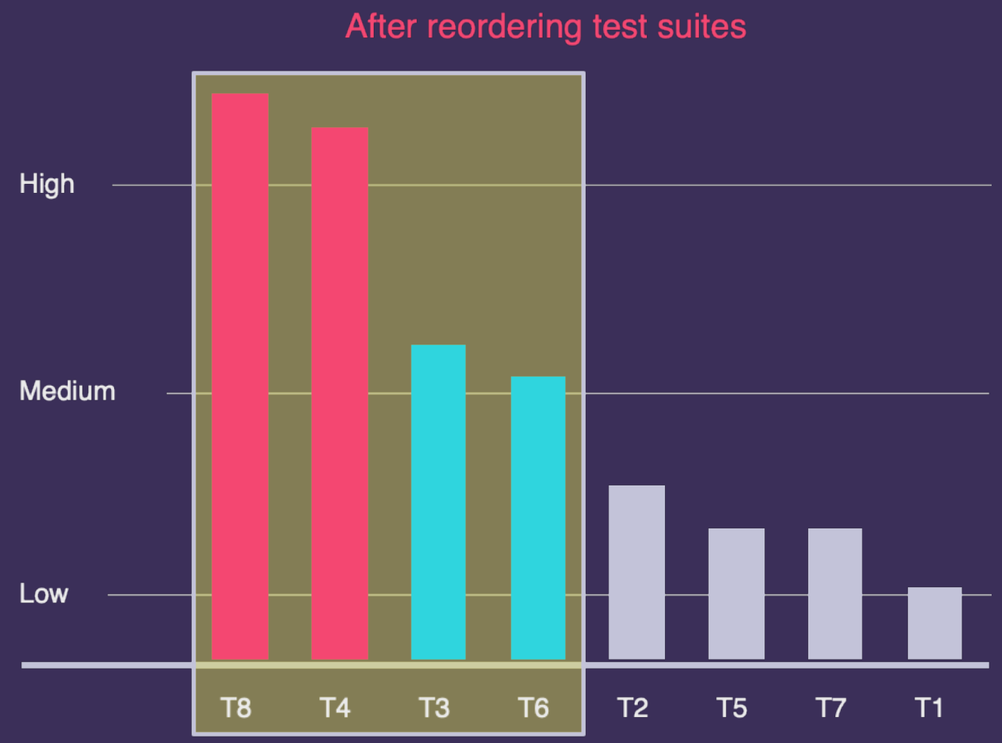
Current focus is **machine learning based test selections** by:

- Predicting a meaningful subset of tests.
- Identifying flaky tests.
- Visualizing test trends with metrics.

What is Launchable?

e.g. Reordering tests based on likelihood of failures.





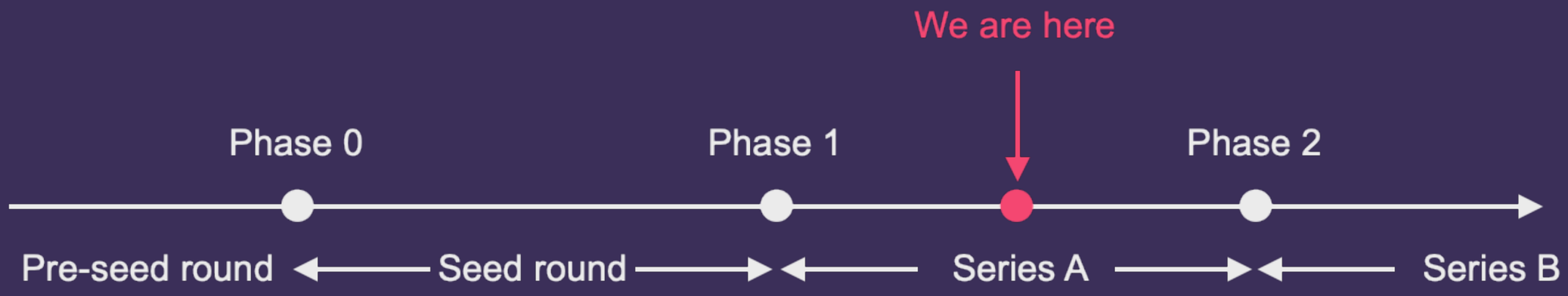
Our team size

- Launchable is a startup
- **2 CEOs + 15 employees**
  - Software engineer (7 people)
  - Product manager
  - Marketing
  - Sales
  - etc...

Phases and the number of software engineers

Note: the numbers are estimated by the presenter based on previous experiences.

- Phase 0: Founding ~ 4 software engineers
- Phase 1: 5 ~ 10 software engineers
- Phase 2: 11 ~ software engineers



My SRE NEXT 2022 is about ...

Incident management at **software startups**

Does a startup need incident management?

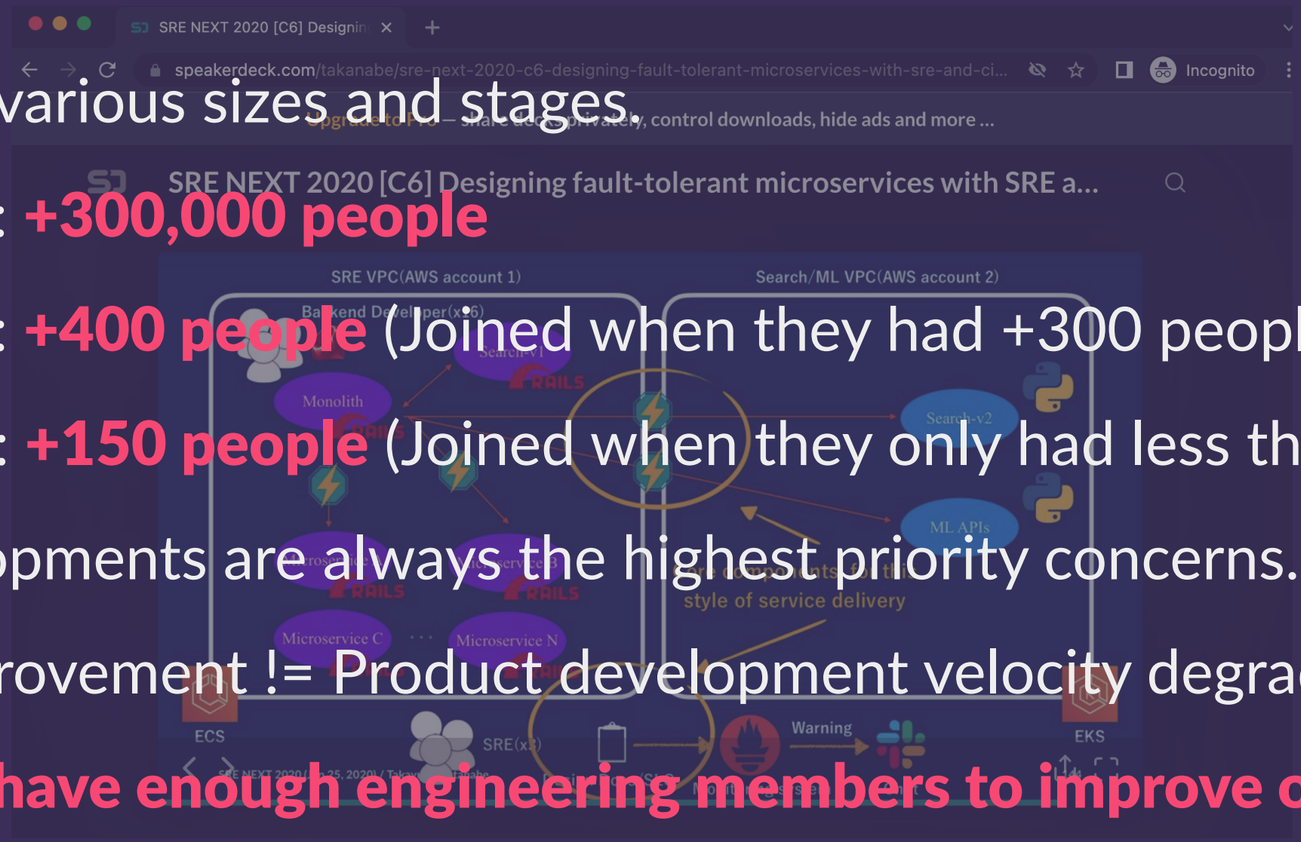
Yes, it's obvious if products have customers.

Do you have enough engineering members?

No! but...

Learning from previous careers[1]

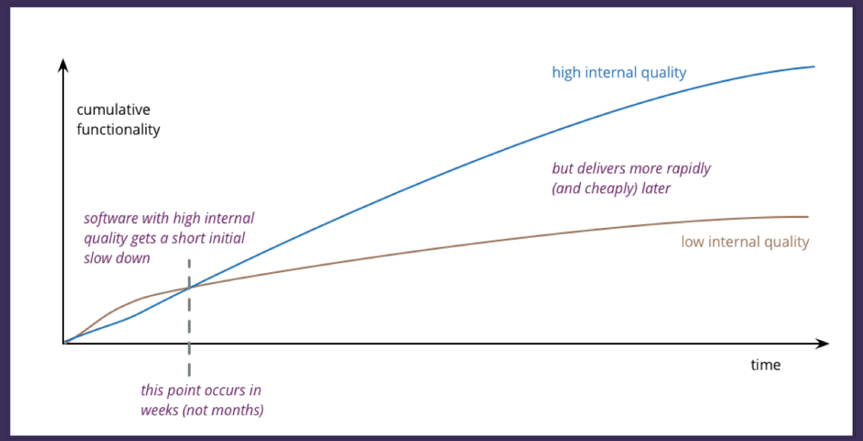
- I've worked at various sizes and stages.
  - Company A: **+300,000 people**
  - Company B: **+400 people** (Joined when they had +300 people)
  - Company C: **+150 people** (Joined when they only had less than 10 people)
- Product developments are always the highest priority concerns.
- Operation improvement != Product development velocity degradation.
- **We will never have enough engineering members to improve operations. Never**.



[1] SRE NEXT 2020: Designing fault-tolerant microservices with SRE and circuit breaker centric architecture

Are speed and quality trade-off?

- I personally don't think so [2] [3].
- I believe sensible incident management accelerates our development velocity.



[2] martinFowler.com: Is High Quality Software Worth the Cost?

[3] A Philosophy of Software Deisgn, Chapter 3: Working Code Isnt' Enough, pp. 13 - 18.

Can we reframe the original question?

- We want to reframe "Does a startup need incident management?" to:
  - Which incident management processes won't change even for rapid developments?
  - Which processes should we improve?
- Let's dissect incident management practices in the industry.

Chapter 2:

Dissect incident management practices

What is incident management?

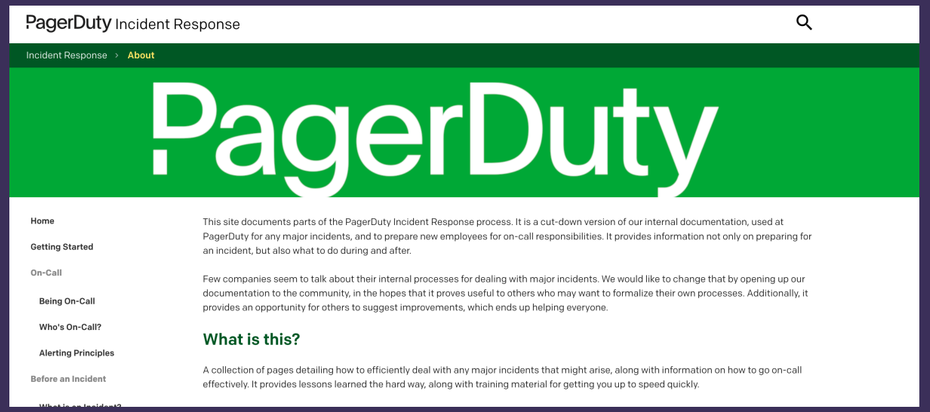
Incident management

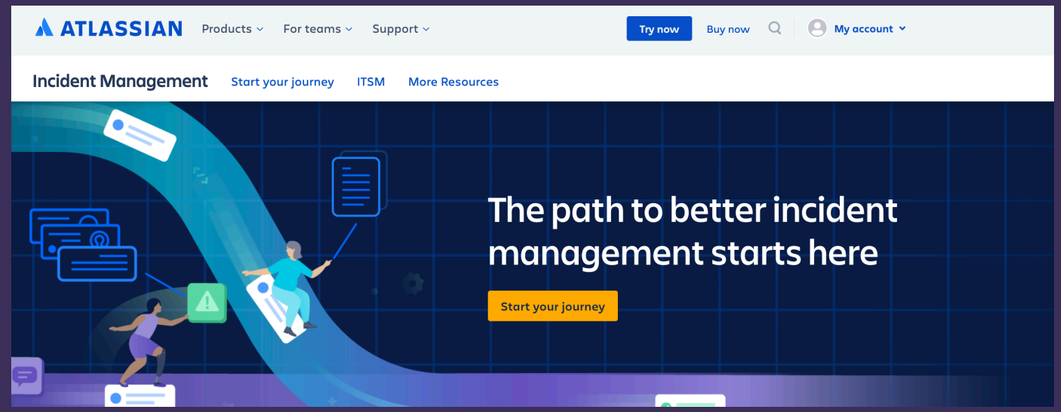
- High level and overall process for handling incidents in an organization.

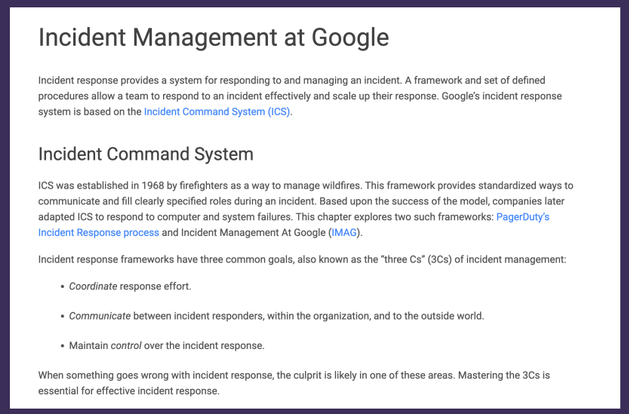
Incident response

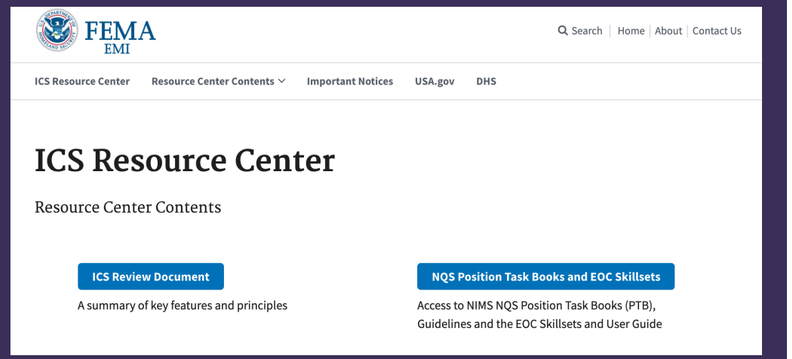
- Part of incident management for actual technical steps including detection, reporting, mitigation, and recovery during incidents.

Existing practices









e.g. Terminology

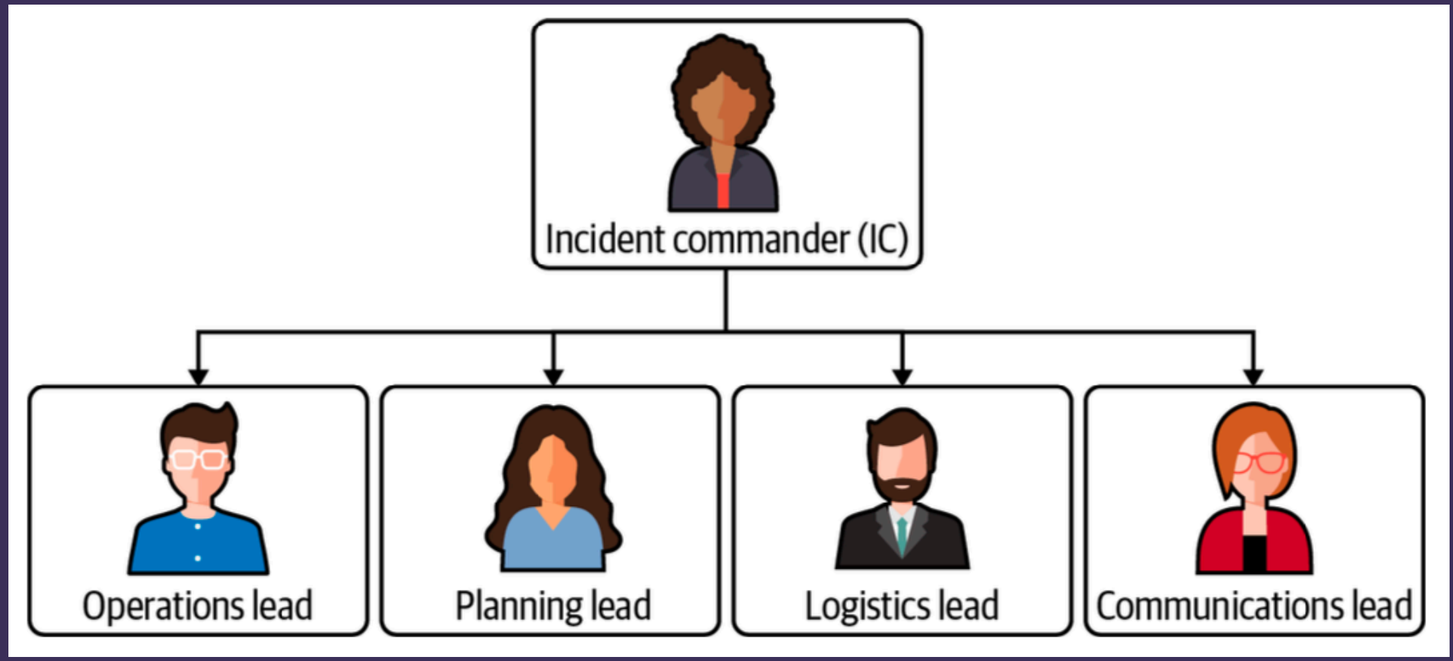
Examples of terminology [4]

- CAN Reports
- Deputy
- Executive Swoop
- Grenade Thrower
- Incident Commander (IC)
- Resolver
- Severity
- Scribe
- Subject Matter Expert (SME)

[4] https://response.pagerduty.com/training/glossary/

e.g. Roles

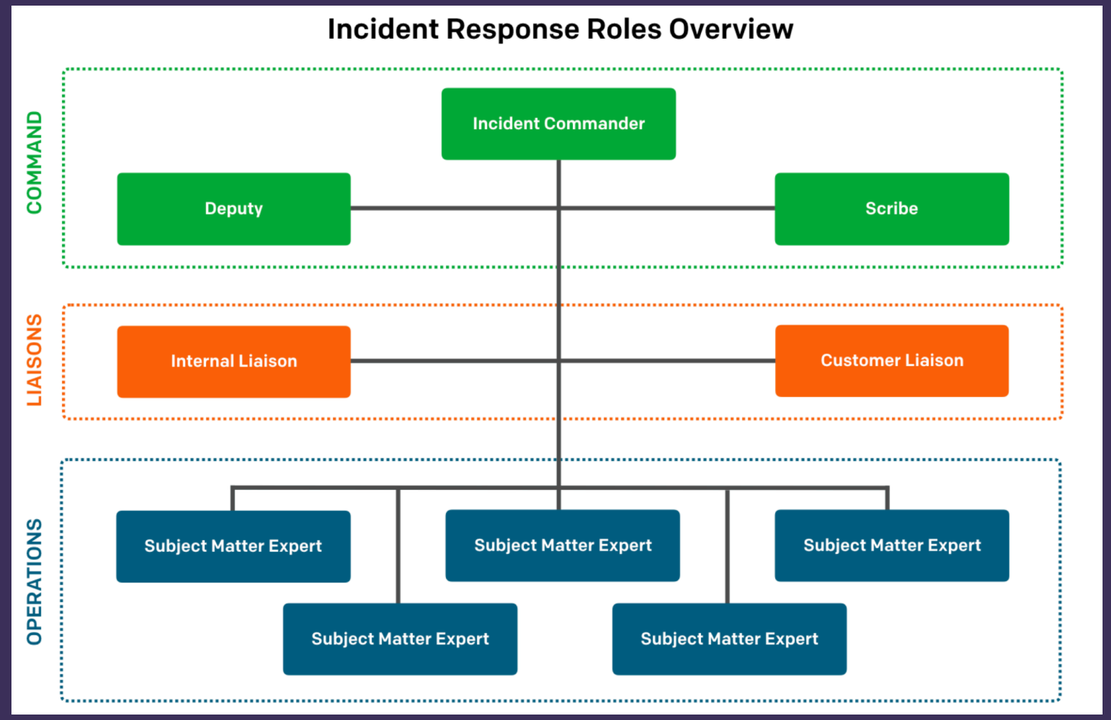
Examples of roles at Google [5] [6]



[5] Google SRE Workbook, Chapter 9: Incident Response

[6] Anatomy of an Incident Google's Approach to Incident Management for Production Services, Chapter 4: Mitigation and Recovery, pp. 31-32.

Examples of roles at PagerDuty [7] [8]



[7] PagerDuty Incident Response Documentation, Different Roles -

[8] Google SRE Workbook, Chapter 9: Incident Response

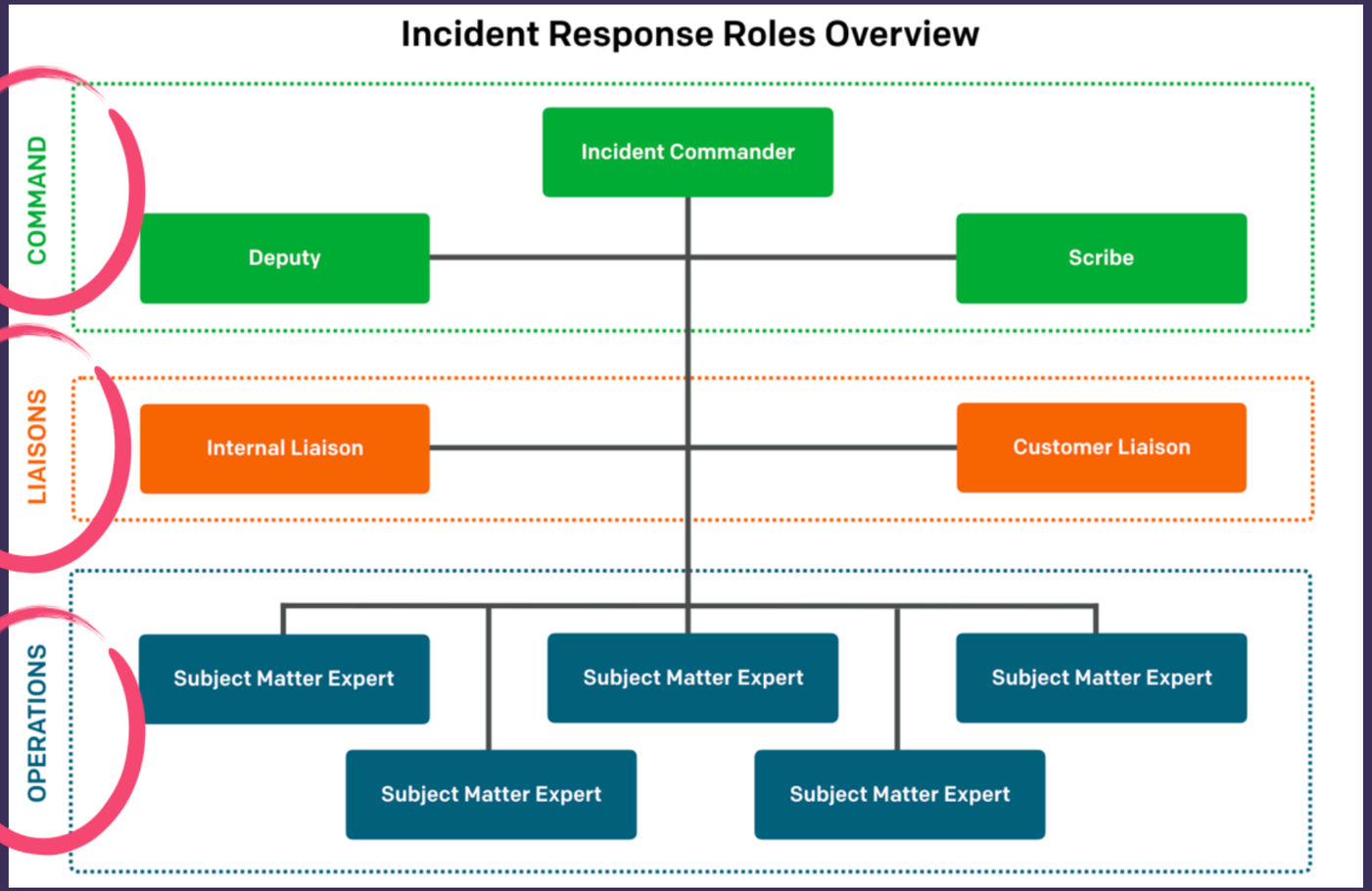
Too much!

Can we translate these practices into more higher level concepts?

Chapter 3:

3T mental models and life cycles

Examples of roles at PagerDuty



Command role

- Responsibility is managing incident responses to align in organizations.
  - Understand ongoing operations
  - Understand who is doing what
  - Delegate sub-commander responsibility to others if necessary.
- Make incident response **tangible**.

Liason role

- Responsibility is smooth reporting and communications.
  - For both internally and externally.
- Make incident response **transparent**.

Operation role

- Responsibility is actual technical activities to solve issues.
  - Focus on triage, analysis, mitigation and recovery.
  - Communication with rest of organizations is not a primary concern.
- In many cases, operators produce root causes of incidents but don't blame them.
  - Nobody wants to cause incidents.
- All participants focus on assigned roles based on chain of **trust**.

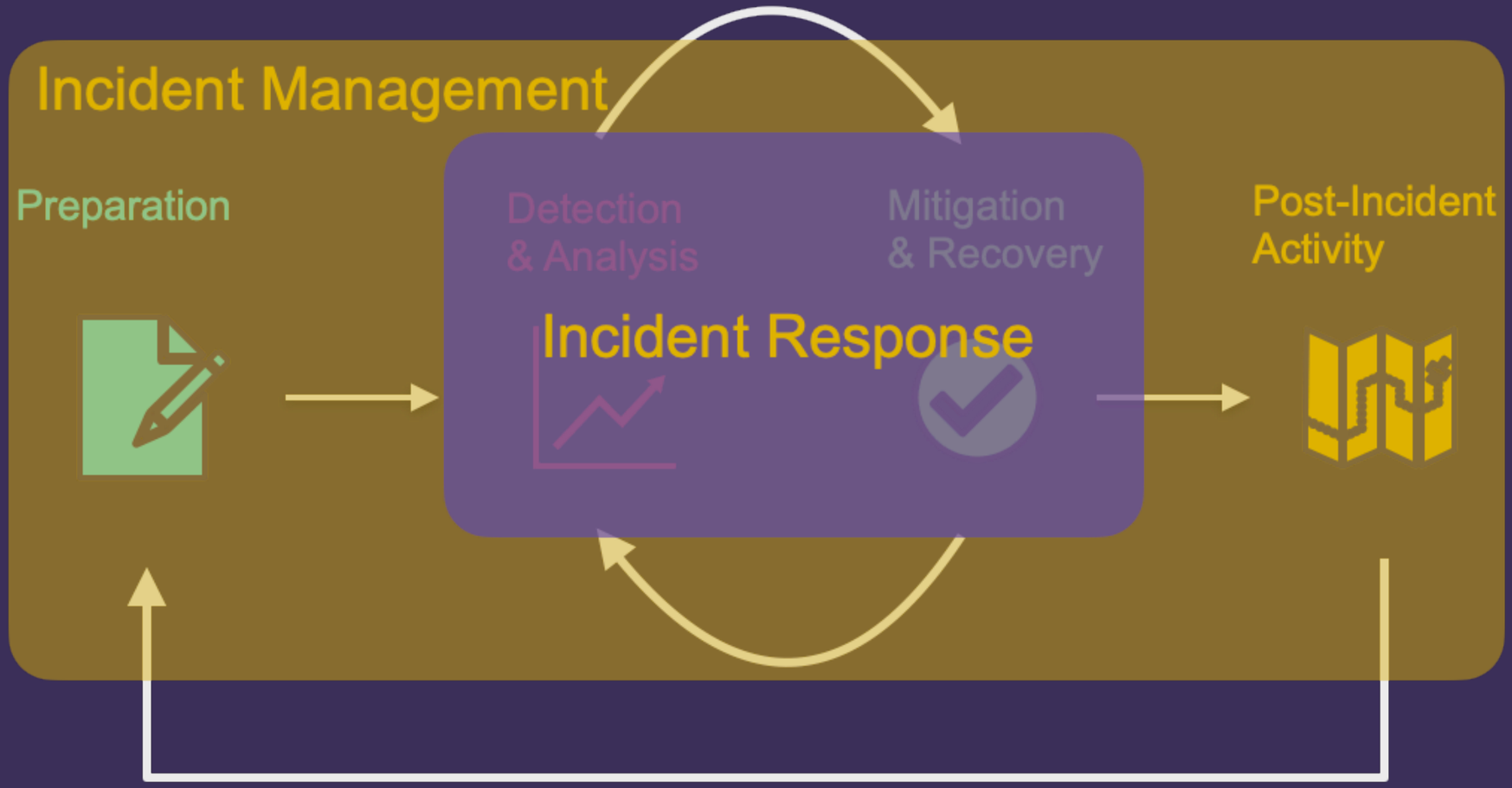
3T mental models for incident response

The incident response roles embody **3T mental models**.

- **Transparency**
  - Keep information of incident responses reachable for everybody.
- **Tangibility**
  - Manage status of incidents.
  - Manage who handles what.
- **Trust**
  - Believe everybody makes best efforts during incidents.
  - Don't blame anybody because nobody wants to cause incidents.

High level view of incident management cycles

High level view of incident management cycles



High level view of incident management cycles

Examples:

- Incident management policy
- Documentation
- Reporting mechanism
- Observability
- Alerting policy
- Incident response training

High level view of incident management cycles

Examples:

- Alerting
- Triage
- Root-cause analysis
- Escalations
- Opening war rooms

High level view of incident management cycles

Examples:

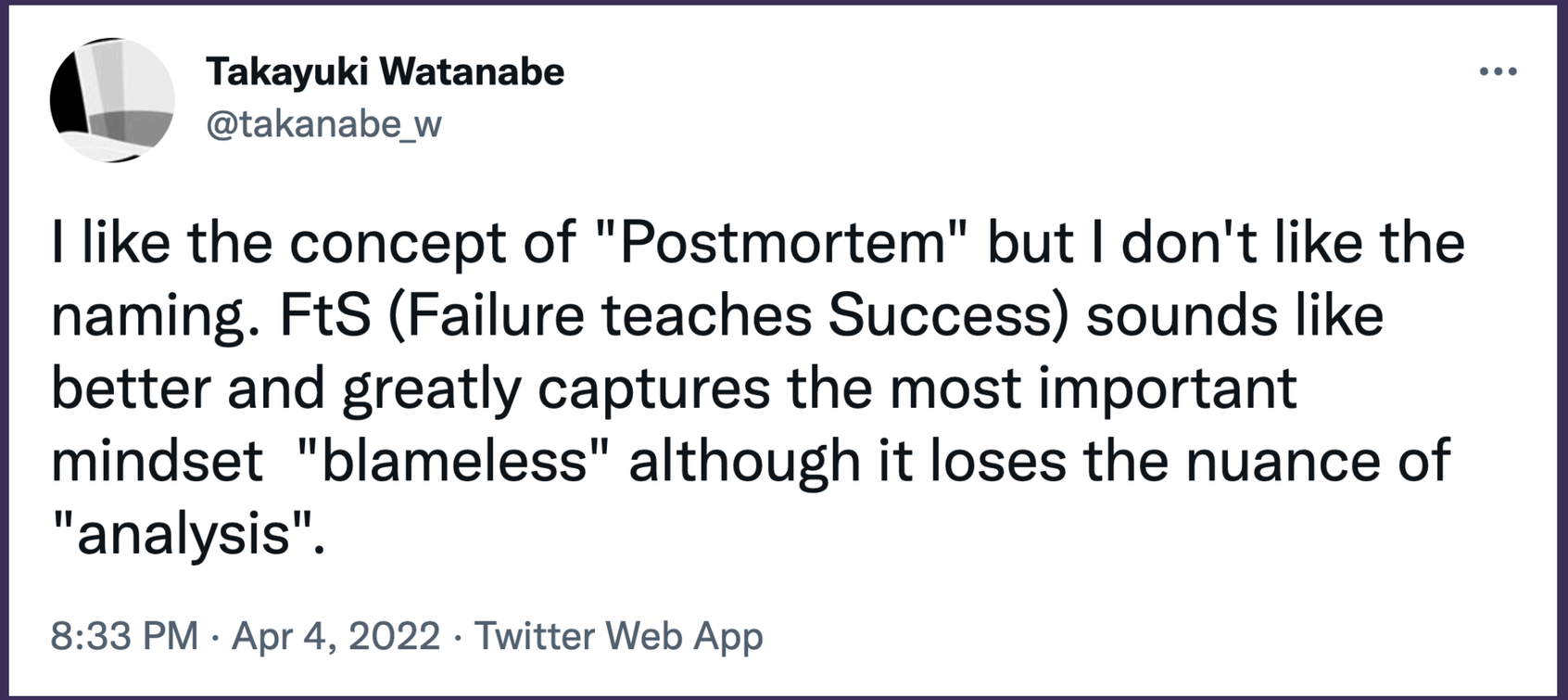
- Rollback deployment (mitigation)
- Kill slow queries (mitigation)
- Fix bug (recovery)
- Add index to tables (recovery)

High level view of incident management cycles

Examples:

- Additional triage
- Preparation for postmortems
- Postmortems
- Handle action items raised at postmortems

Postmortem vs FtS



https://twitter.com/takanabe_w/status/1510943694467186699

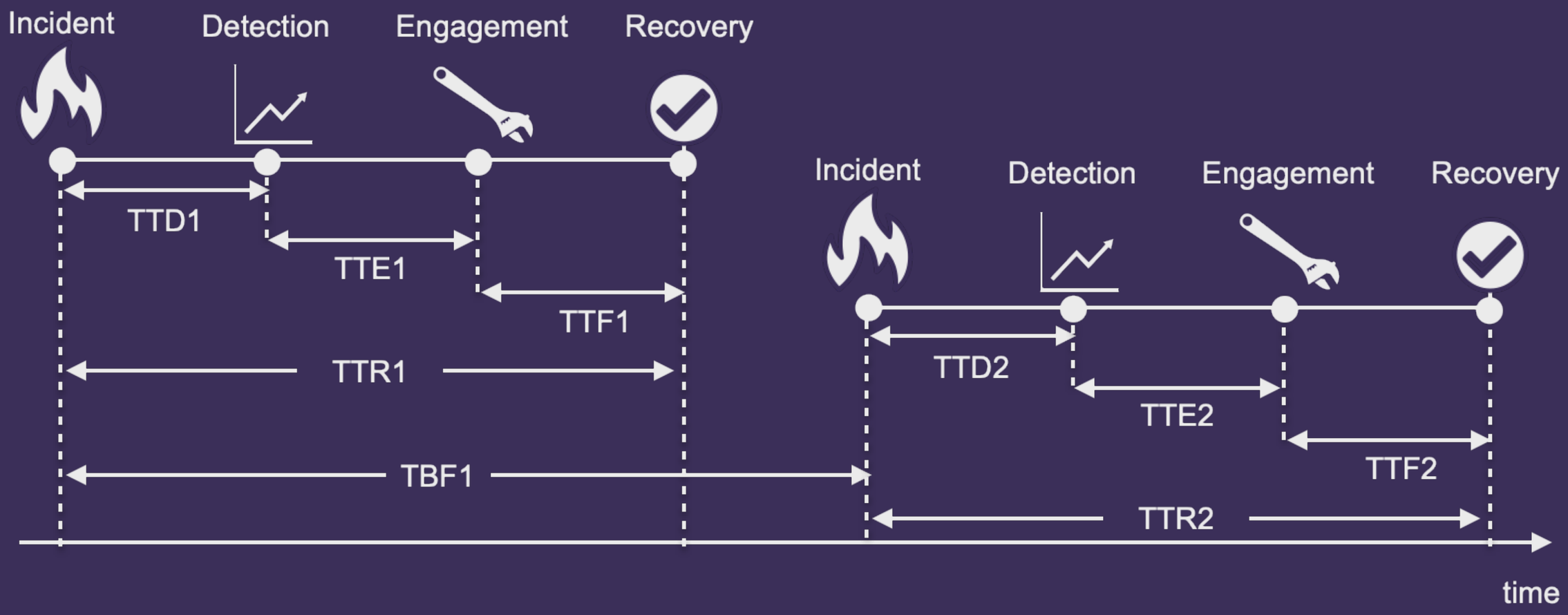
Chapter 4:

How can we improve incident management?

Where should we invest our time?

Where should we invest our time?

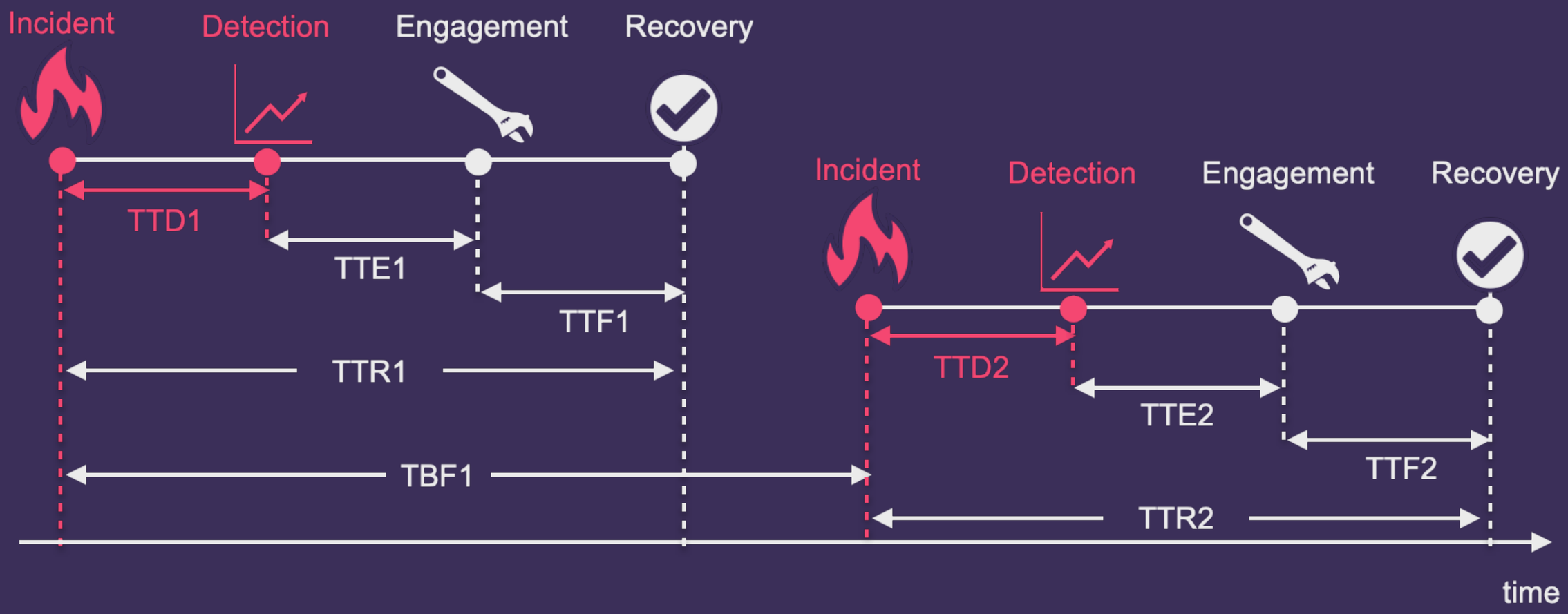
Key times of incident response



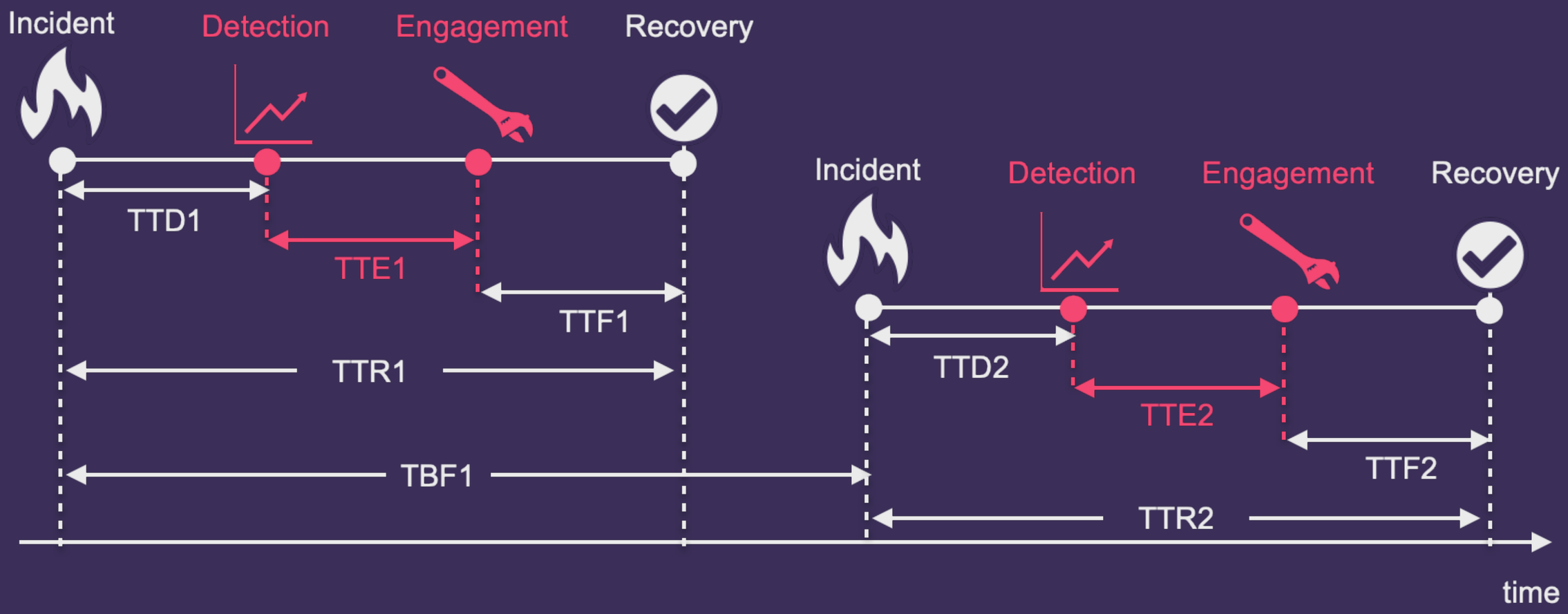
Key times of incident response

- Time **to detect (TTD)**
- Time **to engagement (TTE)**
- Time **to fix (TTF)**
- Time **to repair/recovery (TTR)**
- Time **between failures (TBF)**

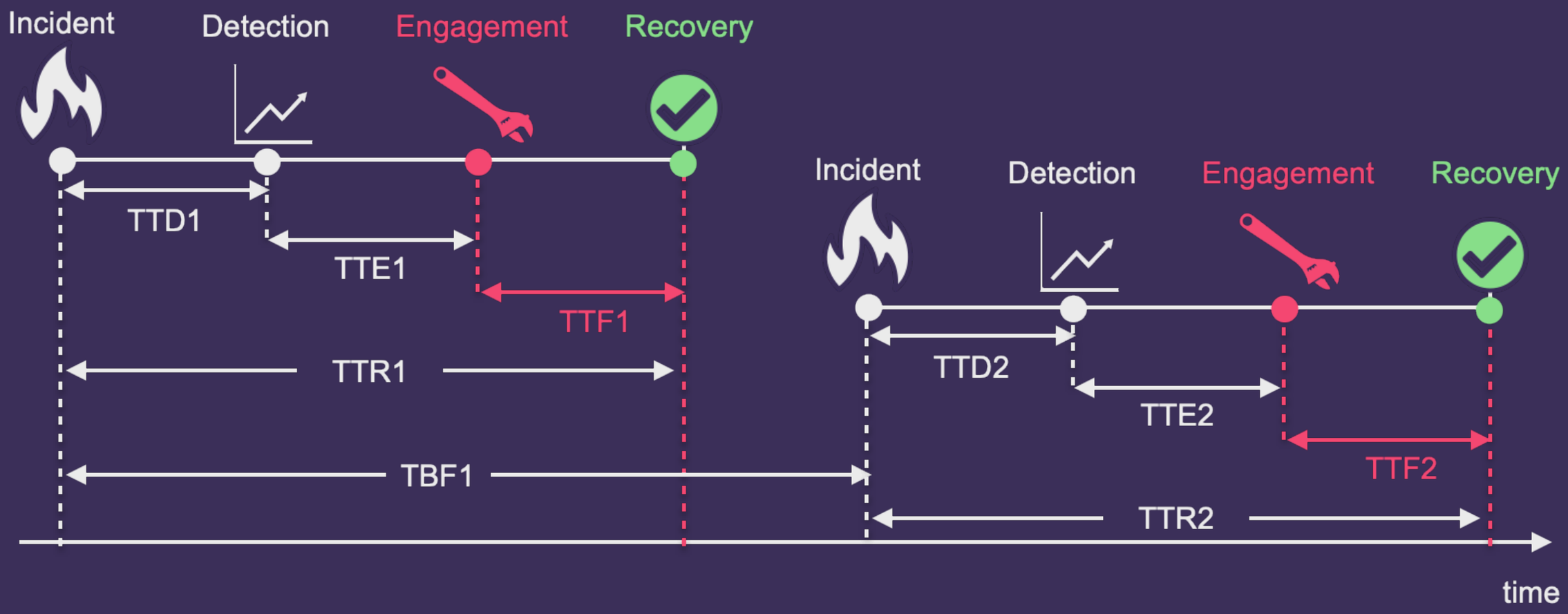
Time to detect (TTD)



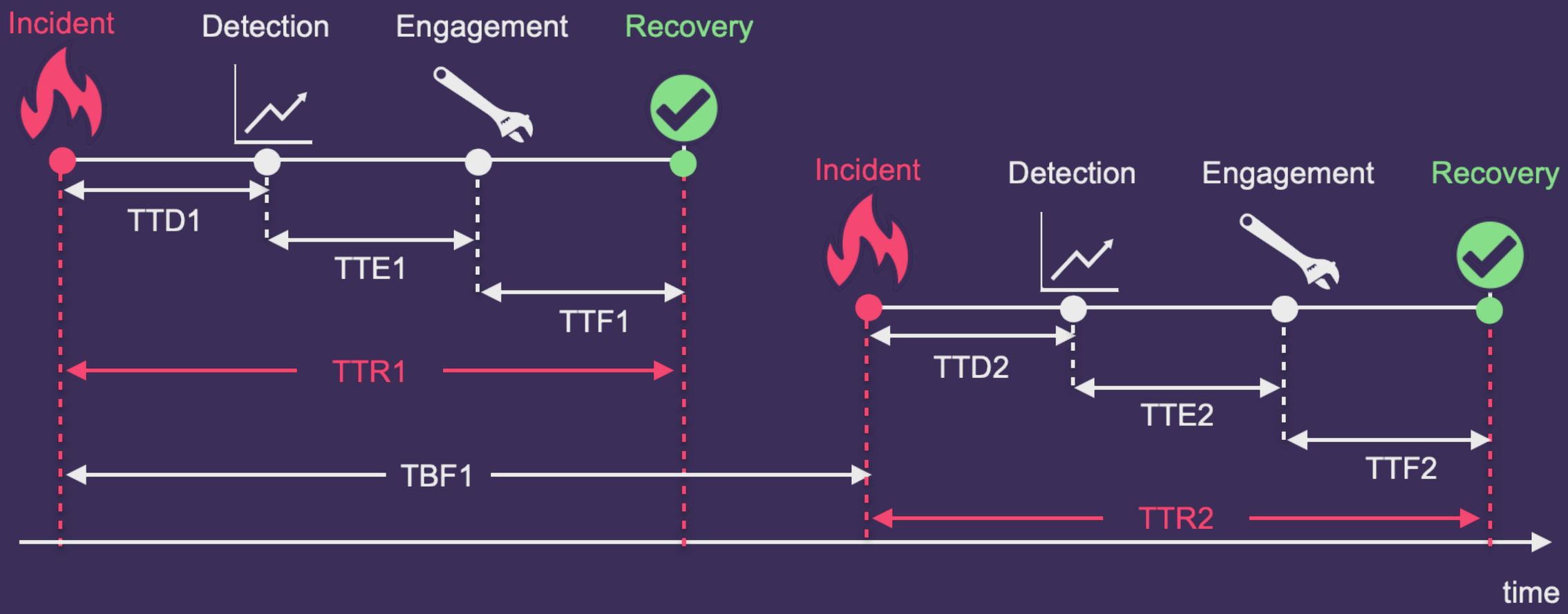
Time to engagement (TTE)



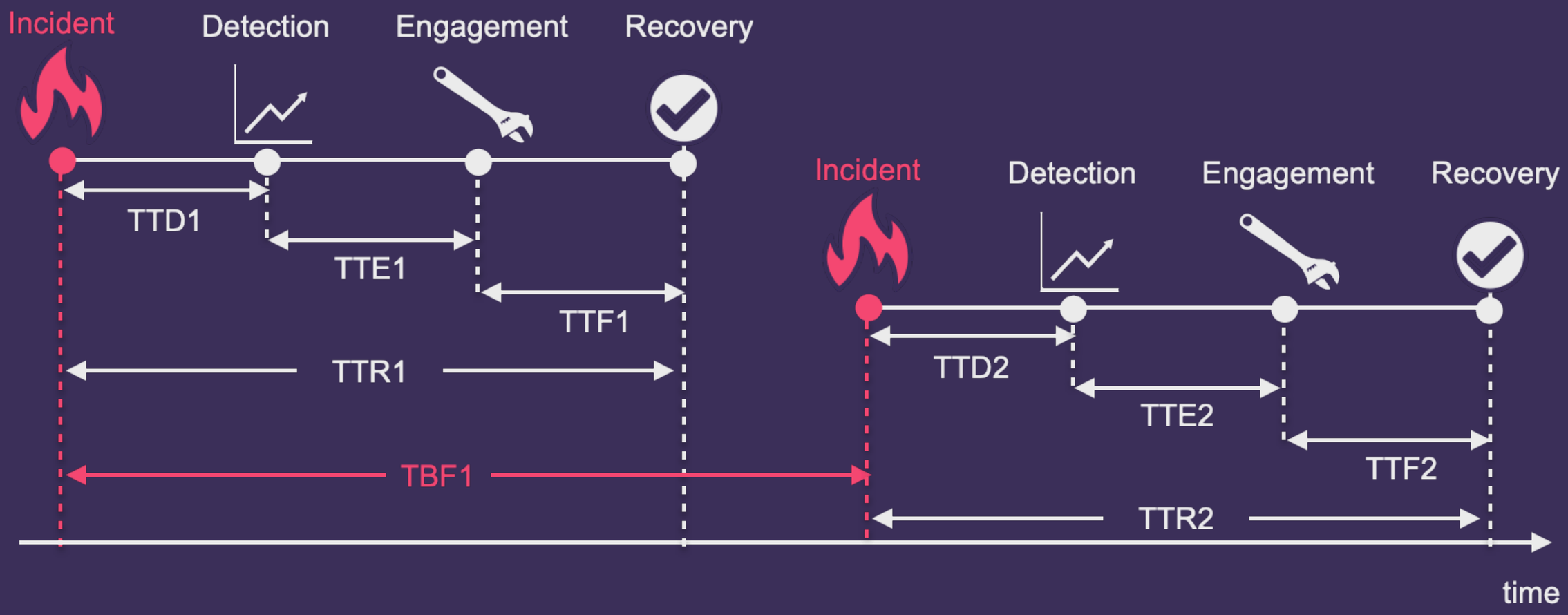
Time to fix (TTF)



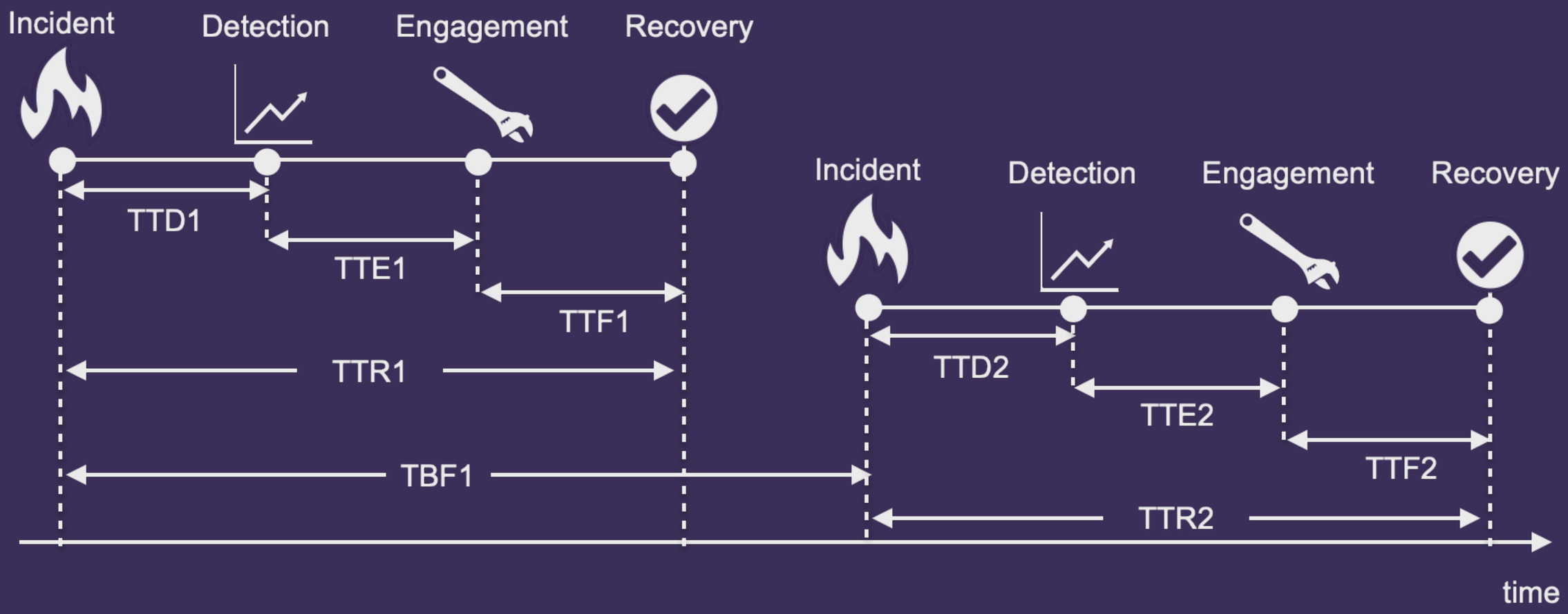
Time to recovery (TTR)



Time between failures (TBF)



Which time do we want to improve?



TTD at Launchable

Current status

- We've already had several detection mechanisms using Datadog and Sentry.

Solution

- Introduction of SLO and Error Budget makes our alerting criteria more clear.
  - But don't forget "Law of diminishing returns" to make decisions.

TTE at Launchable

Current status

- Easy enough to notice during office hours at Slack channels.
- We don't have on-call rotations ATM, which makes TTE uncontrollable.

Solution

- Apply follow-the-sun strategy to cover wide-range hours.
- Introducing on-call rotations and pager.
  - But we don't feel it's necessary now.

TTF at Launchable

Current status

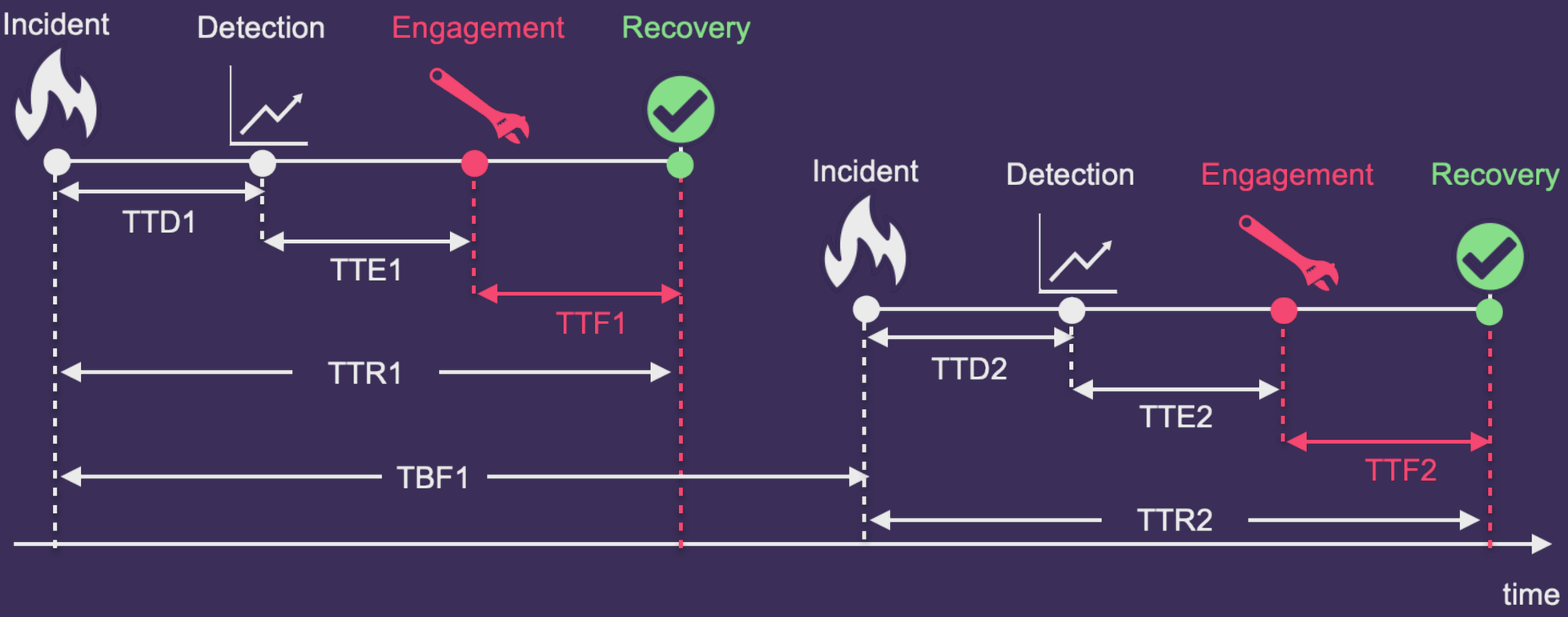
- We don't have enough observability mechanisms
- Depending on each developer's debug skill
- During this window, **developers cannot spend time on product developments**.

Solution

- Introducing more team-shared observability dashboards.
- Introducing more observability mechanism to drill down root causes.

Which time do we want to improve?

TTF improvement brings us high returns with small efforts.

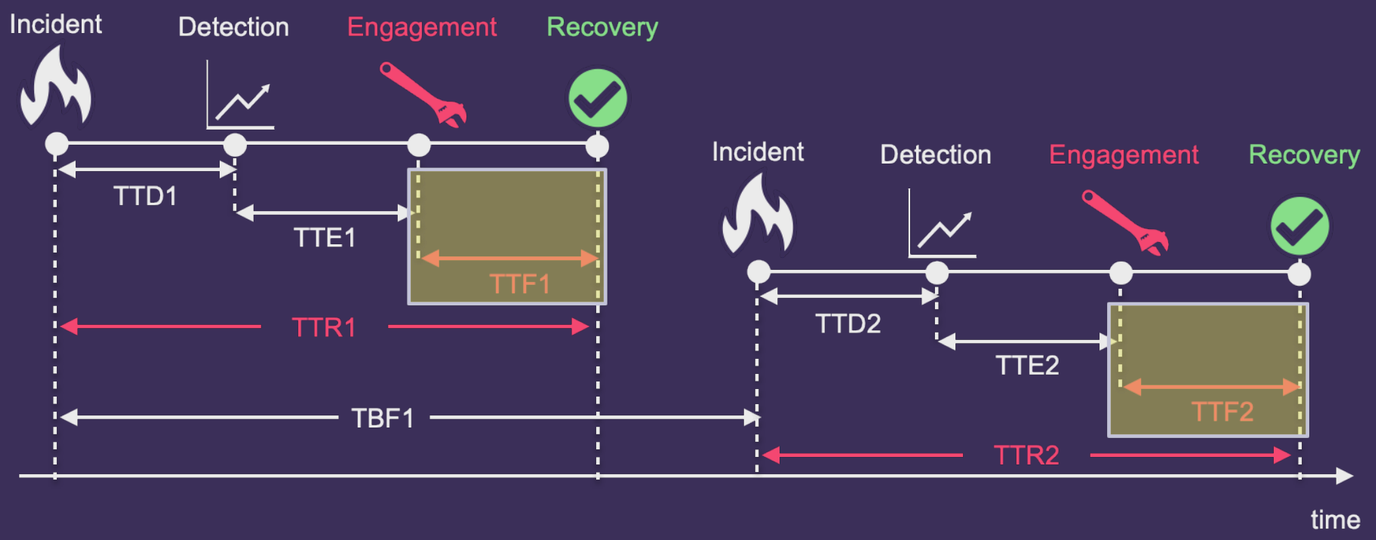


Which do we optimize? MTTR vs MTBF

- Short MTTR and long MTBF are the best
- **Short MTTR but short MTBF**
  = **Incidents frequently occur but are recovered quickly**.
- Long MTTR but long MTBF
  = Incidents don't occur frequently but once occur, they aren't recovered soon.

Startups should focus on MTTR improvement

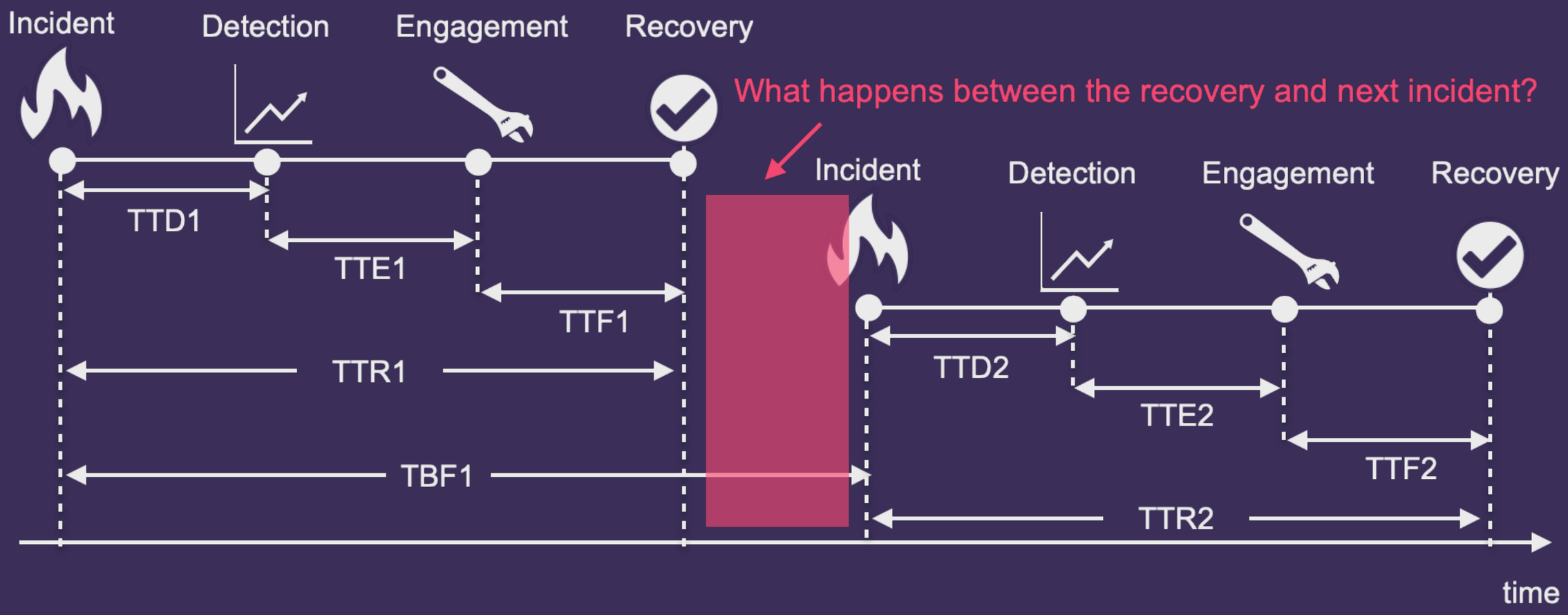
- There is no evolution without high cadence iterations at startups.
- TTD and TTE are difficult to improve for us.
- **Reducing TTF results in reducing MTTR**.



Do we have other times

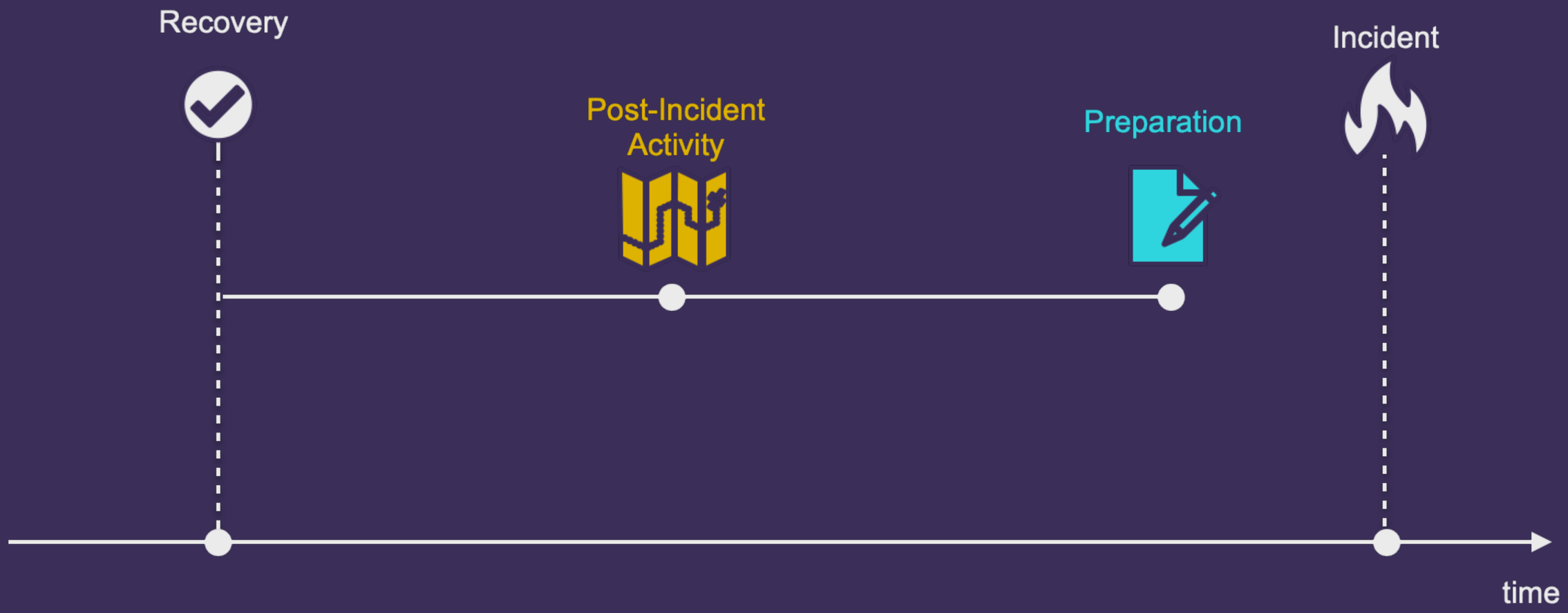
we haven't articulated?

Hidden key times of incident life cycles

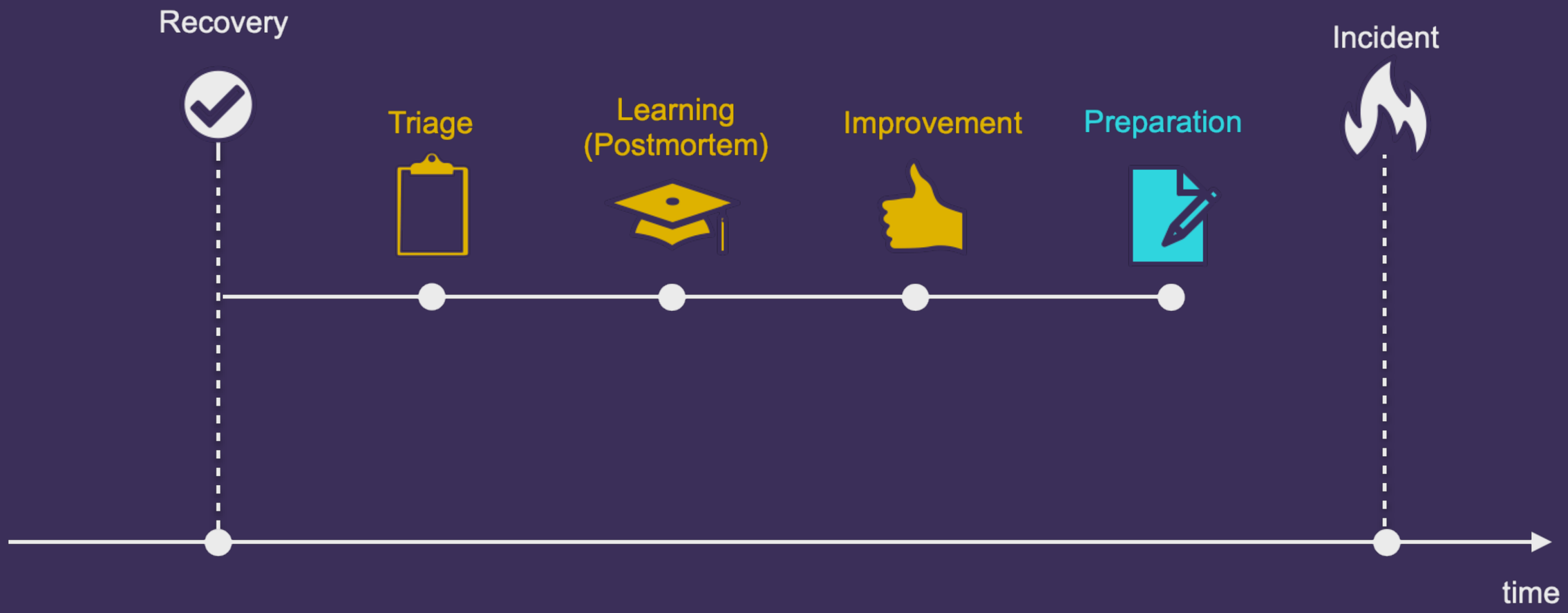


Hidden key times of incident life cycles

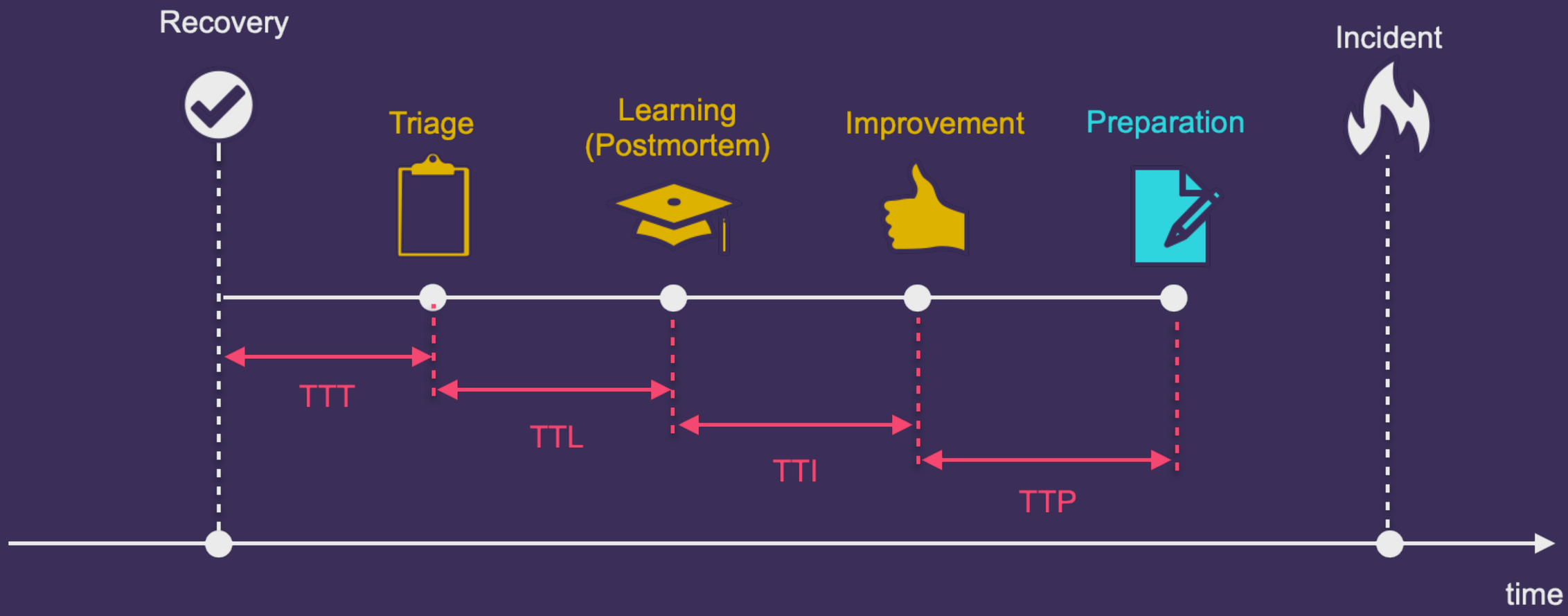
Hidden key times of incident life cycles



Hidden key times of incident life cycles



Hidden key times of incident life cycles



Hidden key times of incident life cycles

Don't underestimate the times we spend as post-incident activities.

- Time to (additional)triage (TTT)
- Time to learn (TTL)
- Time to improvement (TTI)
- Time to preparation (TTP)

Power question:

Which process do you hate?

Which process do you hate?

I personally don't want to spend time on the following processes.

- **Additional triages** to dig root causes.
- **Preparation for learning** (!= I don't like joining postmortem sessions).
- **Maintainance of incident management processes**.

Why additional triages?

- Startups don't have enough observability mechanisms.
  - We sometimes cannot find root causes (this is acceptable).
  - We tend to spend a lot of time here in that situation.

Why preparation for learning?

- Preparation for team-wise learning sessions take time.
  - Documenting for Postmortems.
  - Copy & paste dances to create timeline scattered various places.
  - Timeline needs to consider time-zones.
- There is a gravity which prevent people from announcing incident casually.
  - For starups, the most important activities are learning as a team.
  - If TTL is long, people cannot announce incidents casually.
  - As a result, postmortems ruin short MTTR with high cadence learning iterations.

Why maintenance of incident management processes?

- **Maintainance of incident management processes** contains:
  - Updating incident management policy.
  - Improving incident management structures.
  - Updating documents.
  - Training people to align with the updates.
- Characteristically, incidents don't occur frequently,
  - Too tough to memorize incident response processes for everybody.
  - In urgent situation, people don't read documents.

Can we reduce TTI?

- It's depending on action items coming from postmortems.
- No teams can handle all action items we discussed during postmortems.
- Common anti-pattern: people create too many action items and assign without priorities.

Approach for unbalanced action items

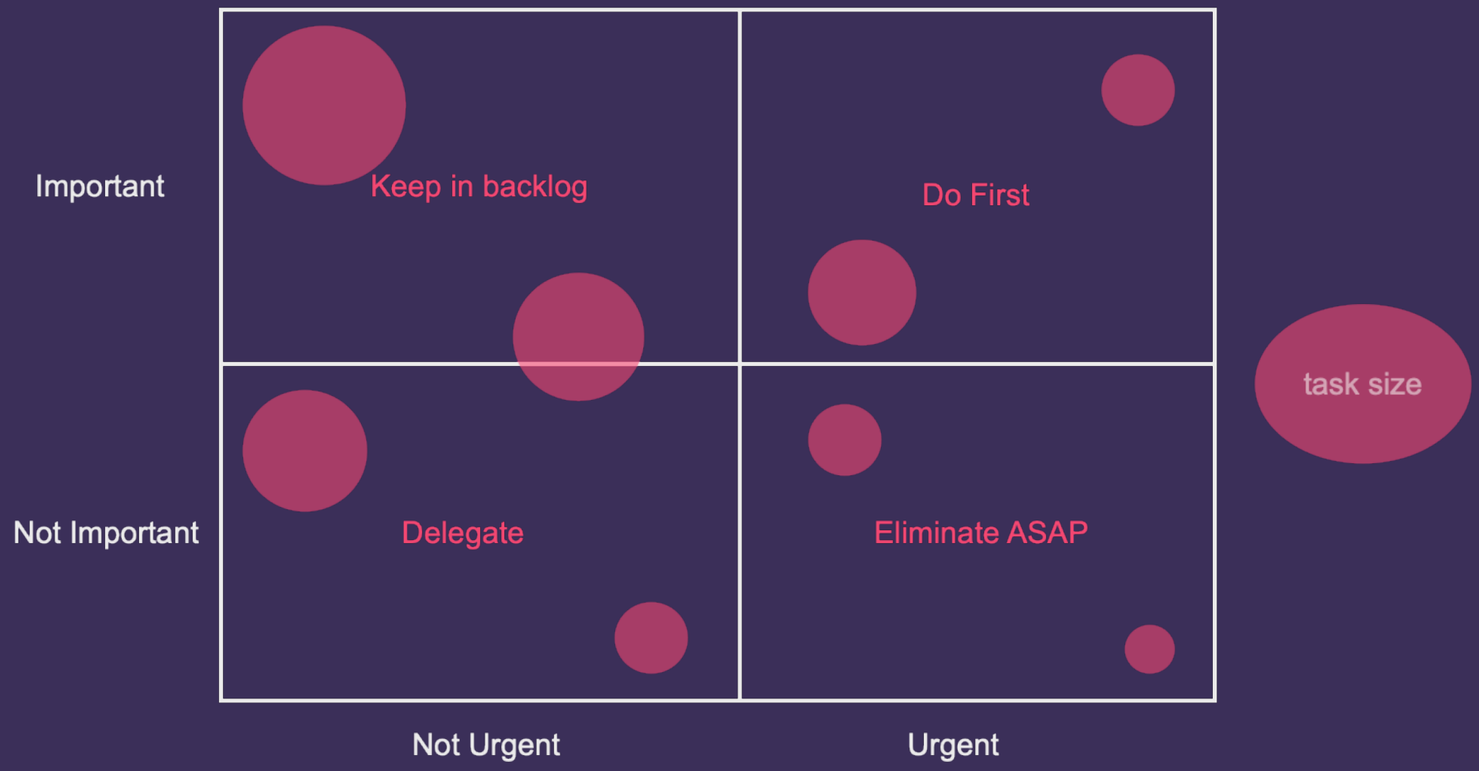
- Think of engineering members' capacity
- Prioritize and classify the work[9] [10]

[9] Postmortem Action Items: Plan the Work and Work the Plan, USENIX SRECon 2017

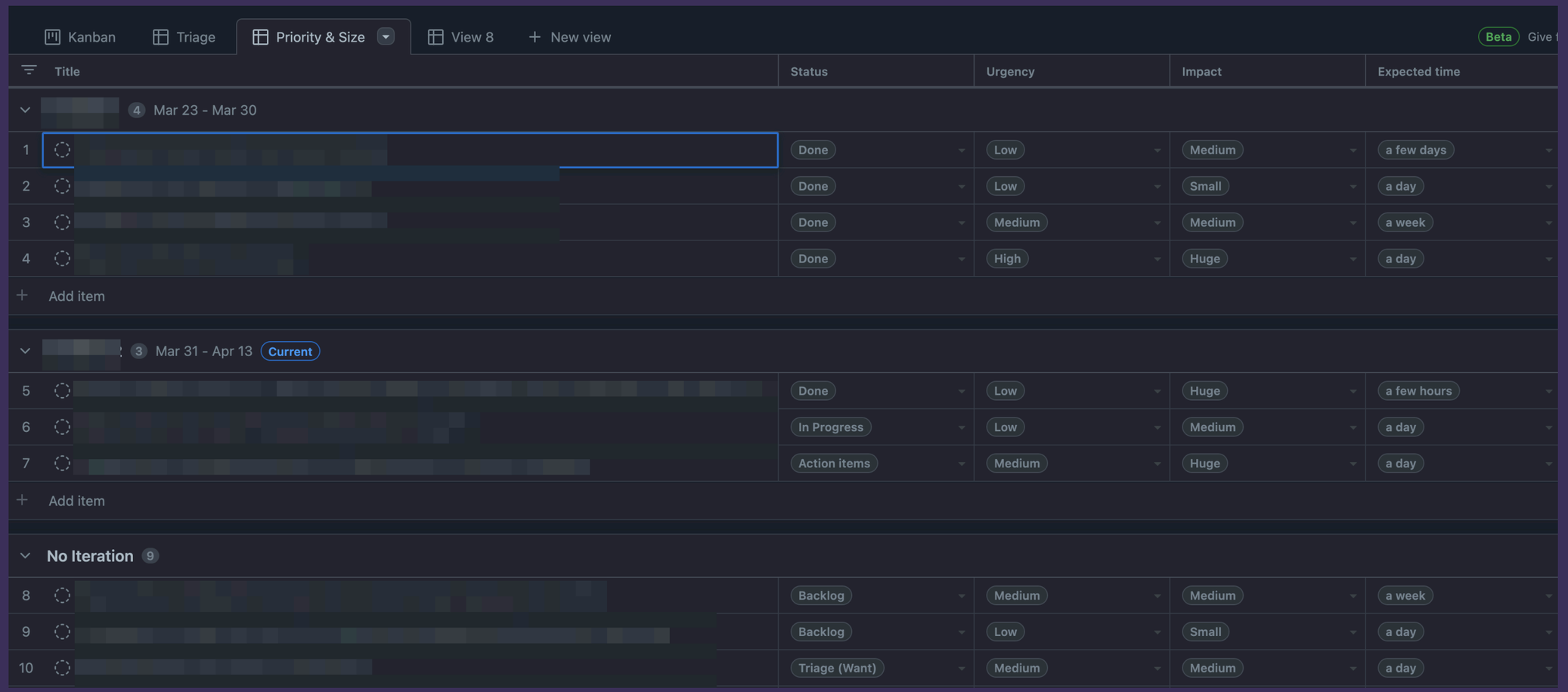
[10] Anatomy of an incident management, Chapter 5

Importance / Size / Urgency (ISU) Matrix

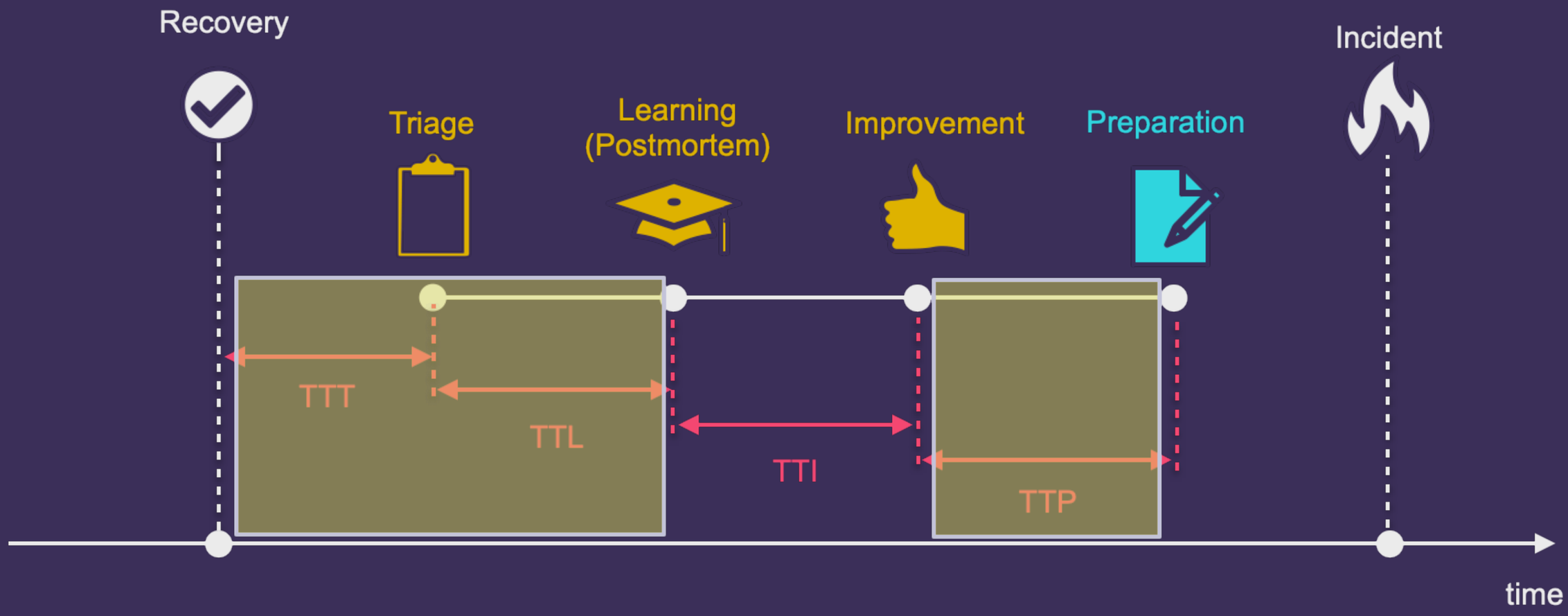
- Assignee's confidence is also valuable to declare.



ISU Matrix on GitHub Projects



My focus is reduction of TTT, TTL, and TTP



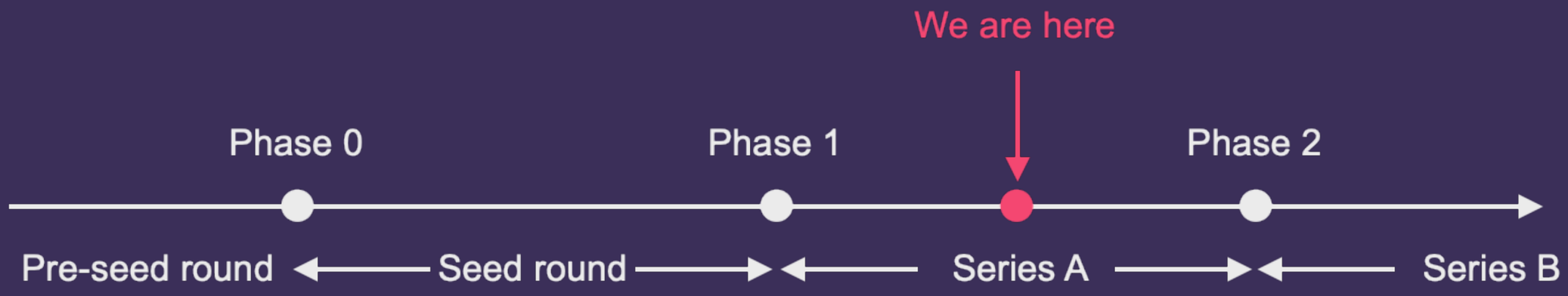
Chapter 5:

Choosing right strategies and tools

Phases and the number of software engineers

Note: the numbers are estimated by the presenter based on previous experiences.

- Phase 0: Founding ~ 4 software engineers
- Phase 1: 5 ~ 10 software engineers
- Phase 2: 11 ~ software engineers



Let's reframe the original question again

- Reframe "Does a startup need incident management?"
- At startups, **how can we**:
  - **Build an incident management structure enforcing the 3T mental models**?
  - **Improve the "times" of the incident management life cycle**?

Evolution of incident management at Launchable

Improvement target	Actions from phase 0 to 1	Actions from phase 1 to 2
Transparency	- Encourage push communication	- Encourage pull communication - Create war rooms - Share status pages
Tangibility	- Automate parts of incident response flow	- Automate entire incident response flow - Introduce incident lead role
Trust	- Introduce blameless culture	- Split lead and operation roles for complex incidents
Time to Engagement (TTE)	- Automate incident announcements	- Automate entire incident response flow - Introduce on-call rotations - Expand follow-the-sun coverages
Time to Fix (TTF)	- Introduce observability	- Improve observability
Time to Triage (TTT)	- Introduce observability	- Improve observability
Time to Learn (TTL)	- Introduce postmortem template	- Generate postmortem
Time to Preparation (TTP)	- Create incident management policies	- Enforce incident management policies - Self-service incident response trainings

Phase 0: Founding ~ 4 software engineers

No strategy

- Product does not have customers.
- We don't need incident responses.
- Build incident management structure based on product growth.
- All members do everything if necessary.

Incident management system

Phase 1: 5 ~ 10 software engineers

Environmental changes from phase 0 to 1

- When products have customers, we need an incident management.
- The more software engineers join, the more incidents happen.

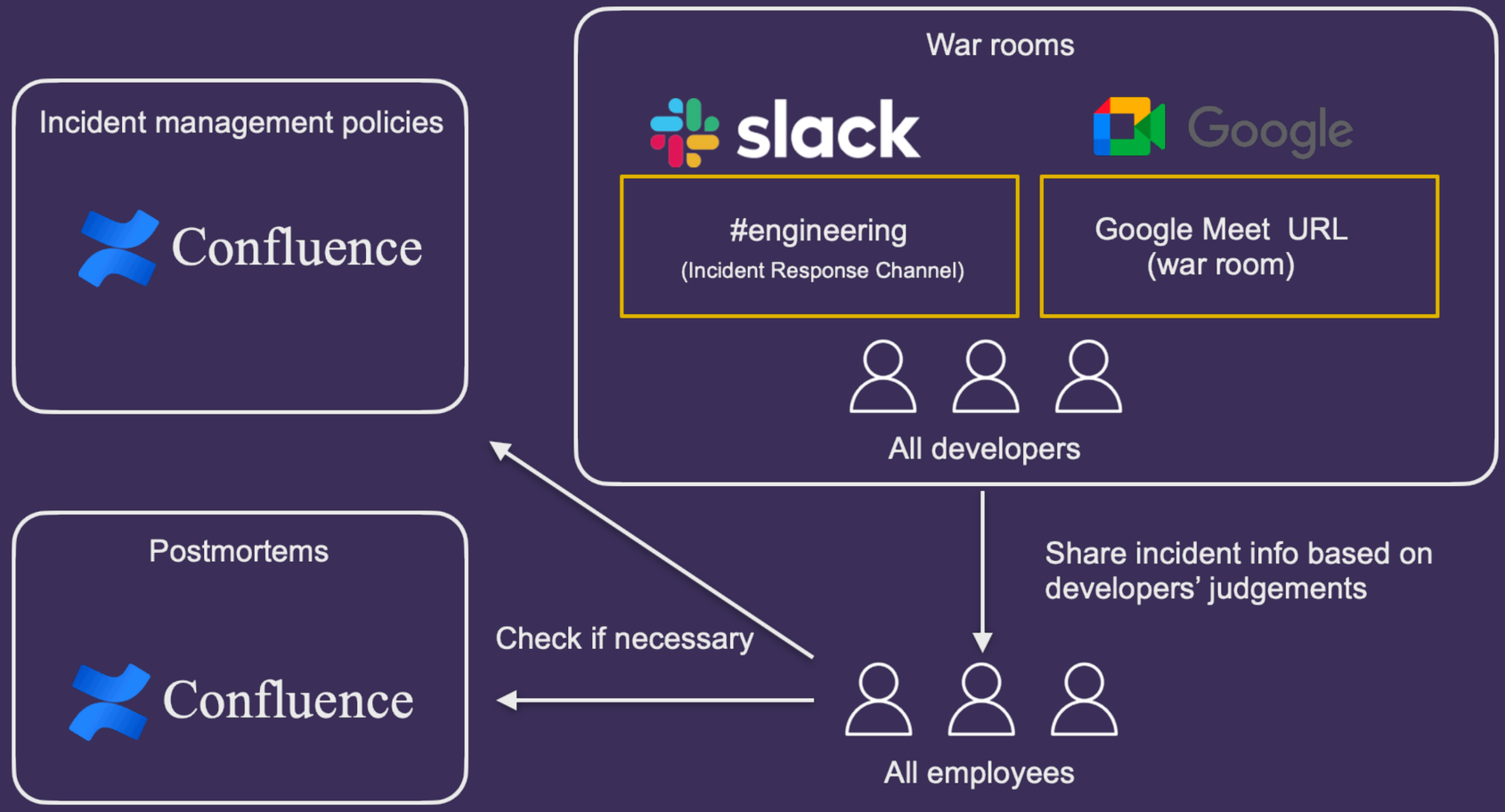
Strategy

Make everything **simple** and **easy** to follow

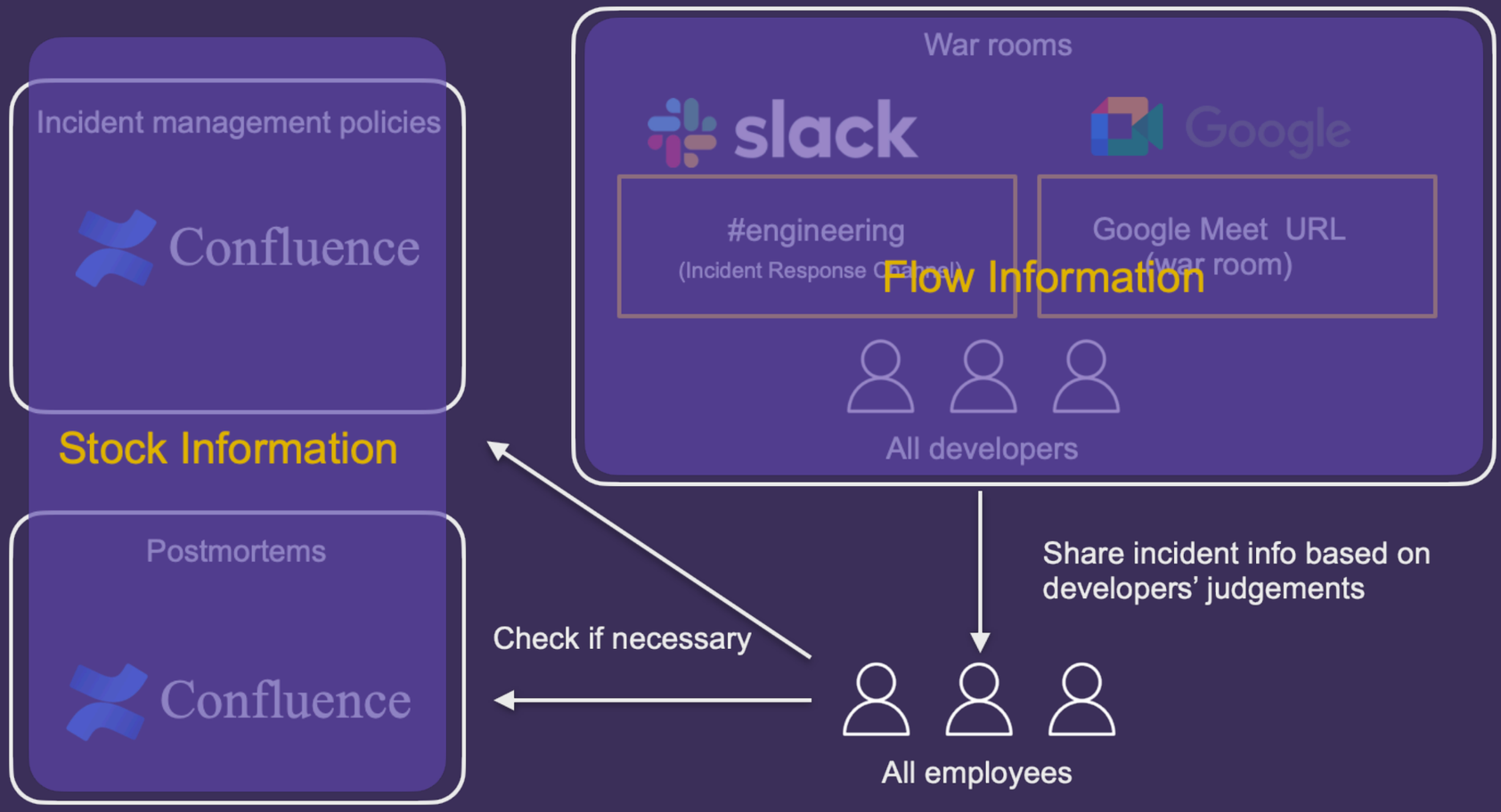
Incident management changes from phase 0 to 1

Improvement target	Actions from phase 0 to 1	Actions from phase 1 to 2
Transparency	- **Encourage push communication**	- Encourage pull communication - Create war rooms - Share status pages
Tangibility	- **Automate parts of incident response flow**	- Automate entire incident response flow - Introduce incident lead role
Trust	- **Introduce blameless culture**	- Split lead and operation roles for complex incidents
Time to Engagement (TTE)	- **Automate incident announcements**	- Automate entire incident response flow - Introduce on-call rotations - Expand follow-the-sun coverages
Time to Fix (TTF)	- Introduce observability	- Improve observability
Time to Triage (TTT)	- Introduce observability	- Improve observability
Time to Learn (TTL)	- **Introduce postmortem template**	- Generate postmortem
Time to Preparation (TTP)	- **Create incident management policies**	- Enforce incident management policies - Self-service incident response trainings

Incident management system (phase 0 to 1)

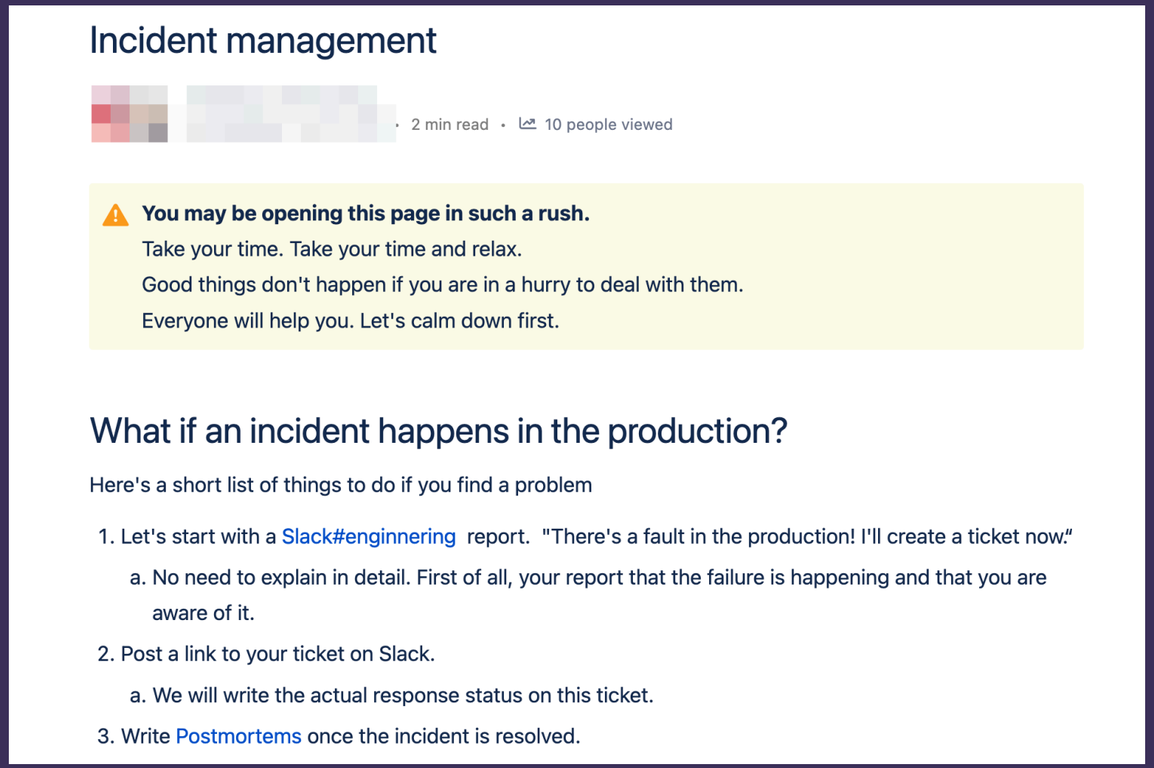


Incident management system (phase 0 to 1)



Foundation of incident management policies

- We maintain policies on Confluence.



Incident management

[redacted] · 2 min read · 10 people viewed

You may be opening this page in such a rush.
Take your time. Take your time and relax.
Good things don't happen if you are in a hurry to deal with them.
Everyone will help you. Let's calm down first.

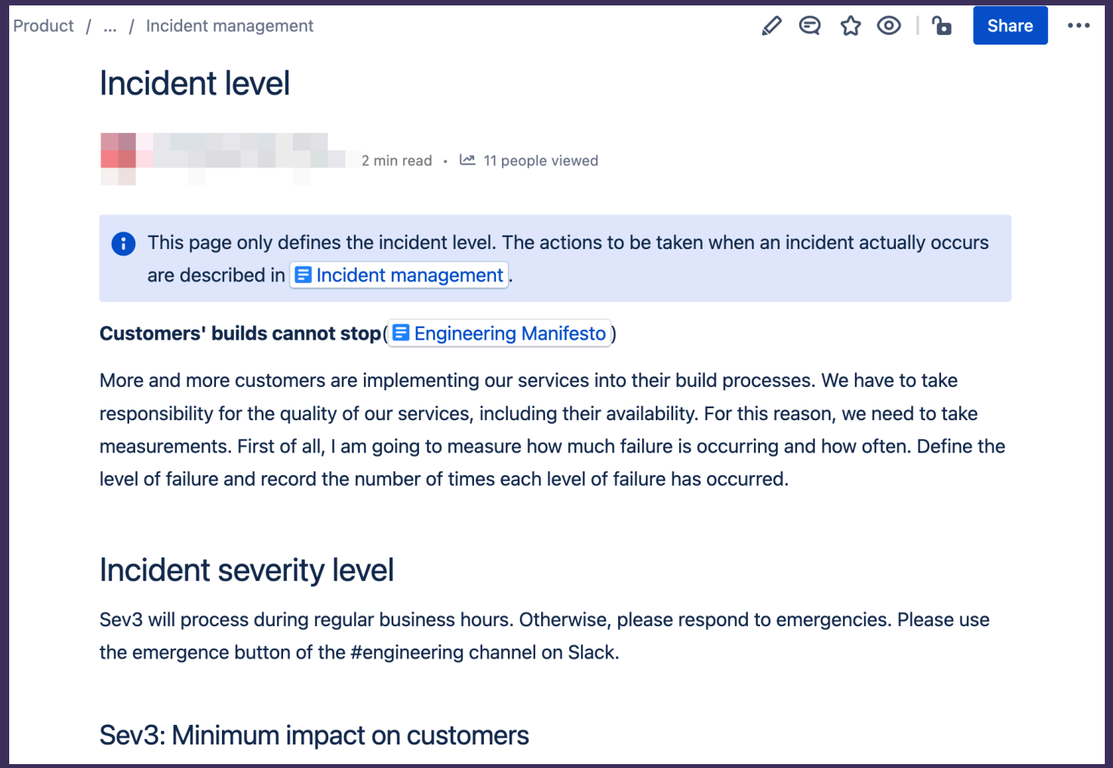
What if an incident happens in the production?

Here's a short list of things to do if you find a problem

1. Let's start with a Slack#enginnering report. "There's a fault in the production! I'll create a ticket now."
    a. No need to explain in detail. First of all, your report that the failure is happening and that you are aware of it.
2. Post a link to your ticket on Slack.
    a. We will write the actual response status on this ticket.
3. Write Postmortems once the incident is resolved.

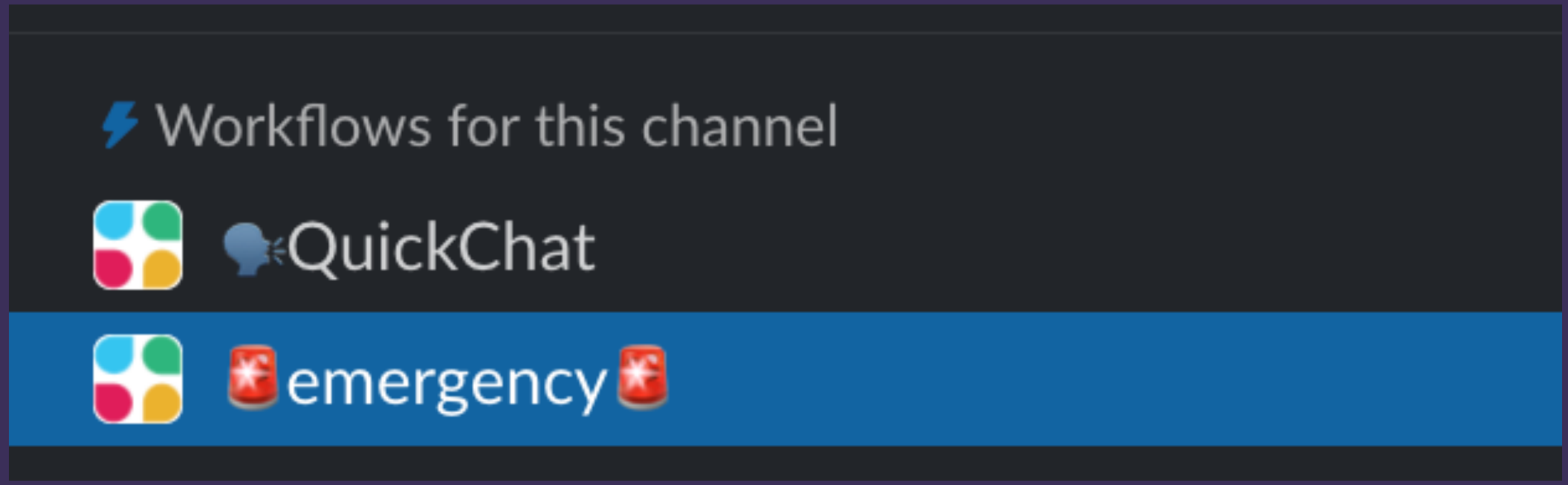
Foundation of incident management policies

- We maintain policies on Confluence.



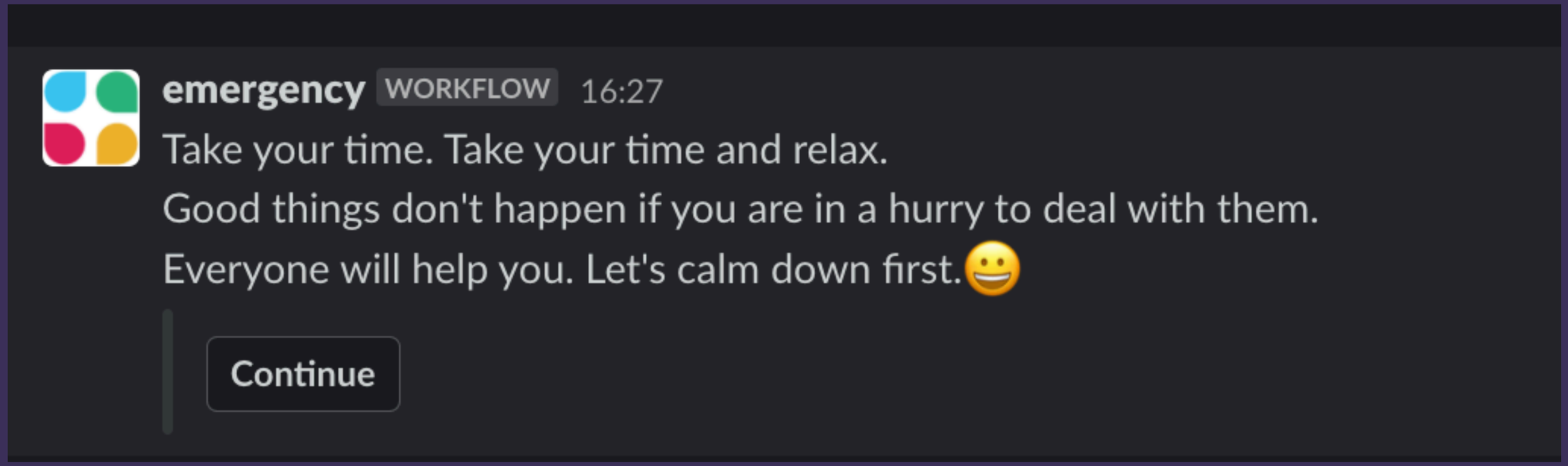
Automation of incident escalations

- We escalate incidents using Slack Workflow.
- We handle incidents in Slack channel and Google Meet.



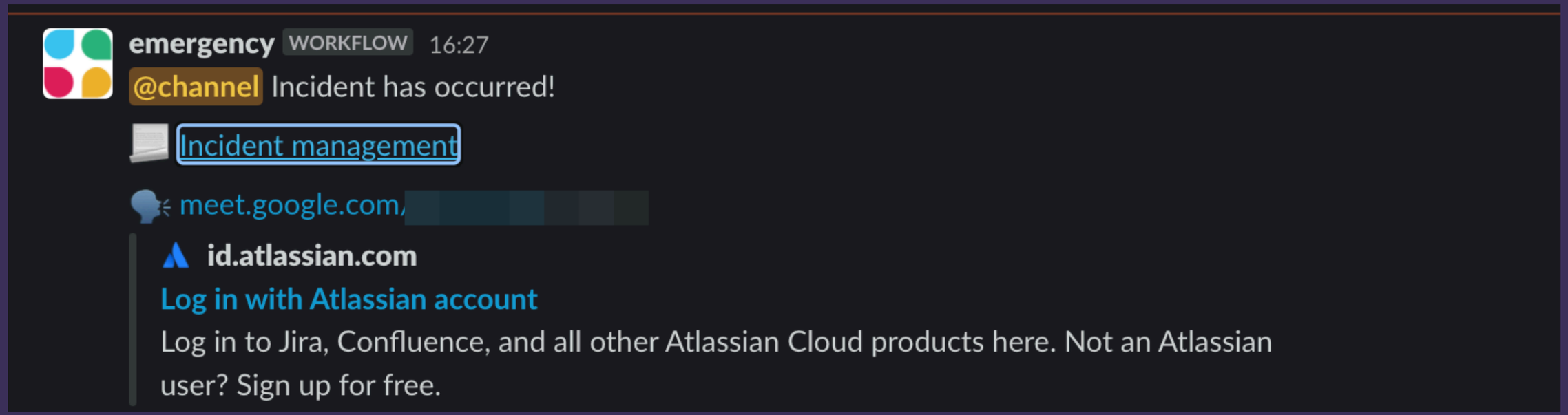
Automation of incident escalations

- We escalate incidents using Slack Workflow.
- We handle incidents in Slack channel and Google Meet.



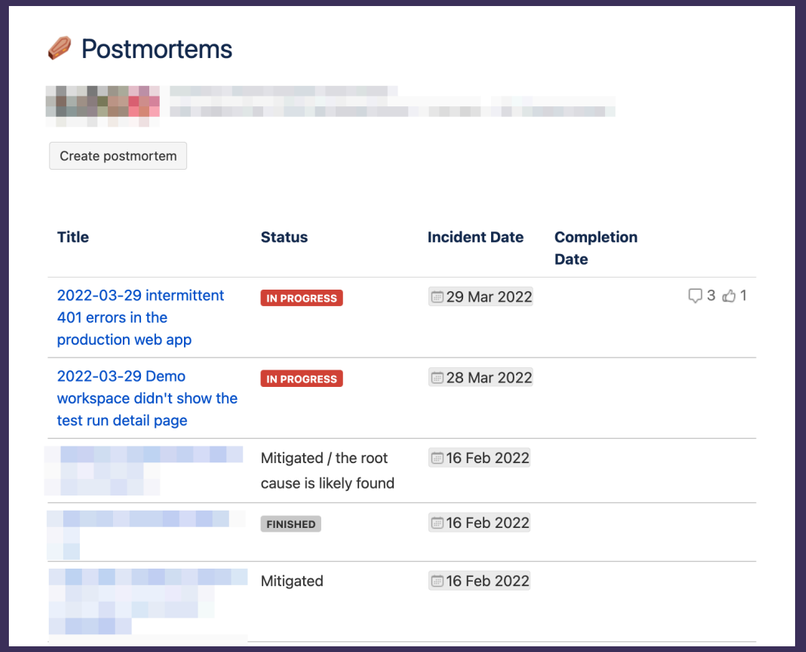
Automation of incident escalations

- We escalate incidents using Slack Workflow.
- We handle incidents in Slack channel and Google Meet.



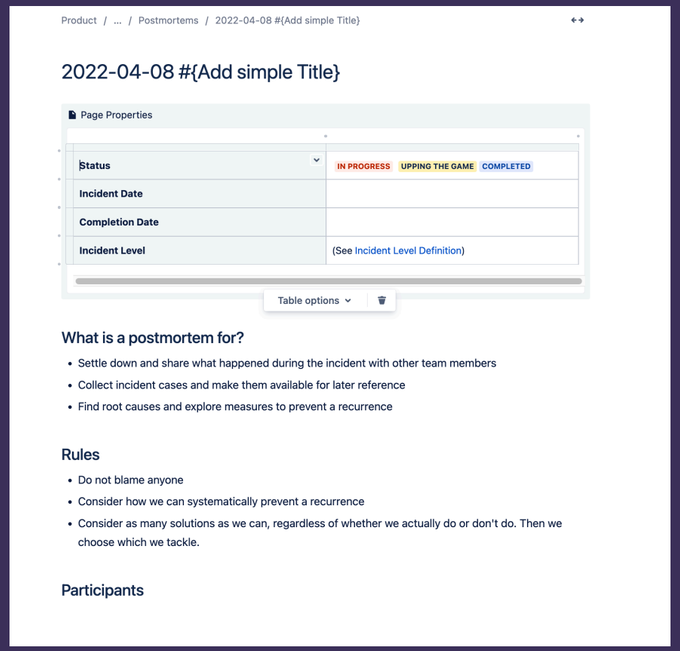
Introduction of postmortem

- We keep all postmortems on Confluence.
- We create a new postmortem page using a Confluence template feature.



Introduction of postmortem

- We keep all postmortems on Confluence.
- We create a new postmortem page using a Confluence template feature.



Very simple and easy to follow

Postmortems as strong fact-based data

- Even we cannot solve root causes, you can use the postmortems as data.

Separate [illegible] from [illegible] for maintainability and extensibility

Created by [illegible]
Last updated: Mar 24, 2022 • 4 min read • 6 people viewed

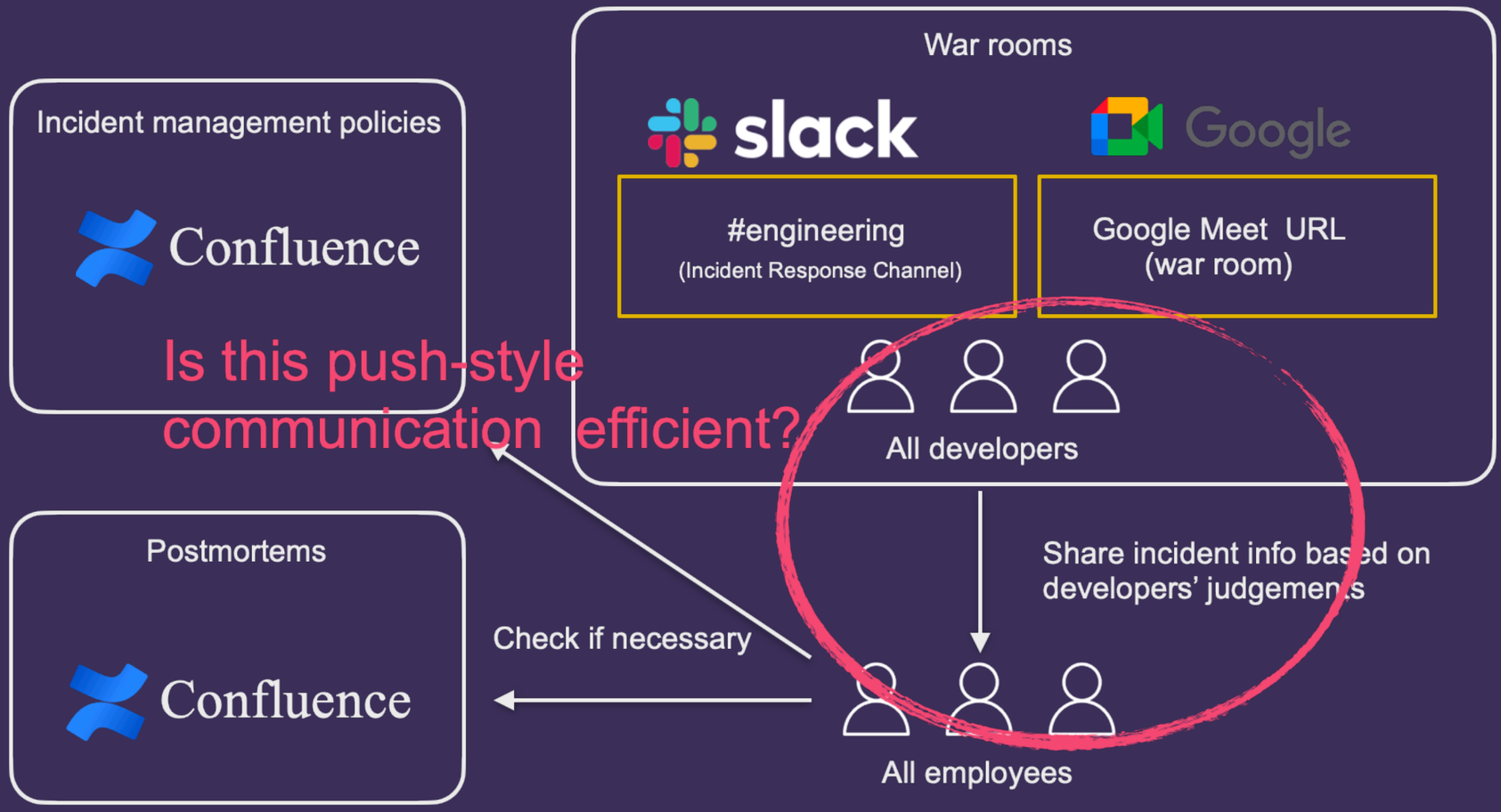
Status	PROPOSED
Driver	@[illegible]
Date	Mar 22, 2022
Team Central	[illegible]
Expected audience	engineers

Why

- We have high ratio of [illegible]-related incident compared to API server portion of [illegible] Out of 19 incidents occurred in 2021 and 2022, 14 of them are caused by [illegible], which we are experiencing one [illegible]-related incident per month. Doubtless to say high availability of the [illegible] system would improve service level as well as engineer resource focused to contribute on important product.

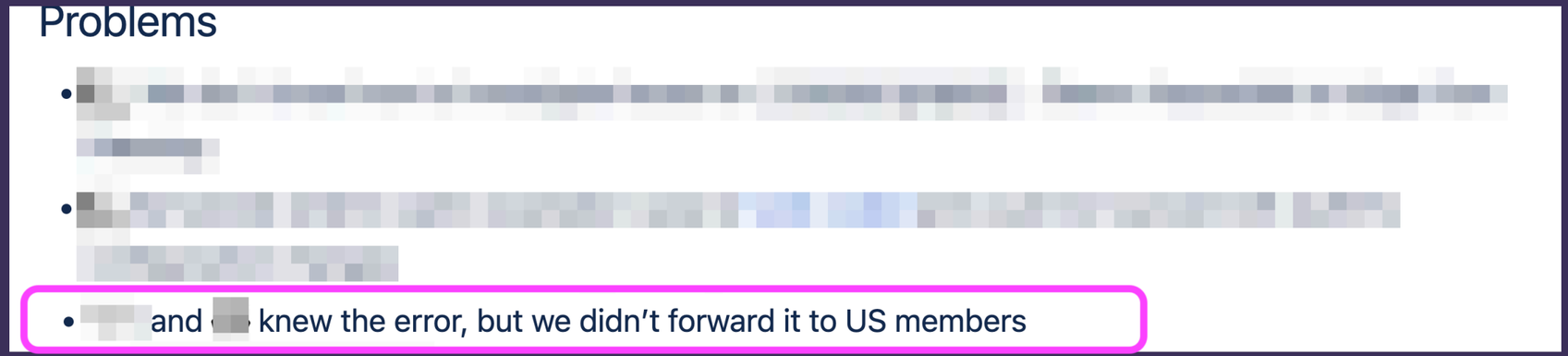
Can we improve the incident management?

e.g. Does this what human should take care?

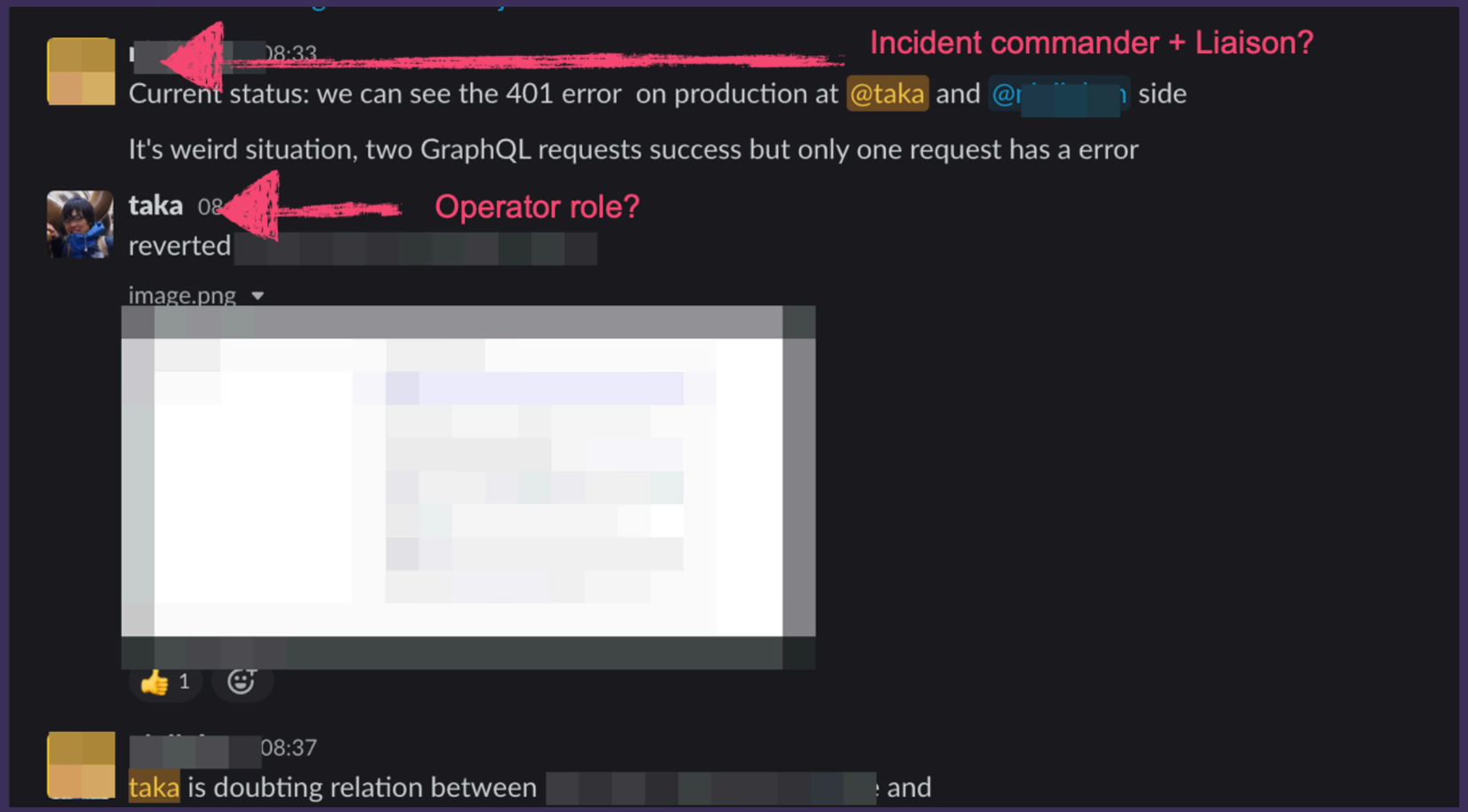


e.g. Does this what human should take care?

- We don't have solid policy but policy does not scale.
- Employees are living in Japan and US.
- Sharing all information on Slack is easy to miss.



e.g. Do we need roles?



Phase 2: 11 ~ ?? software engineers

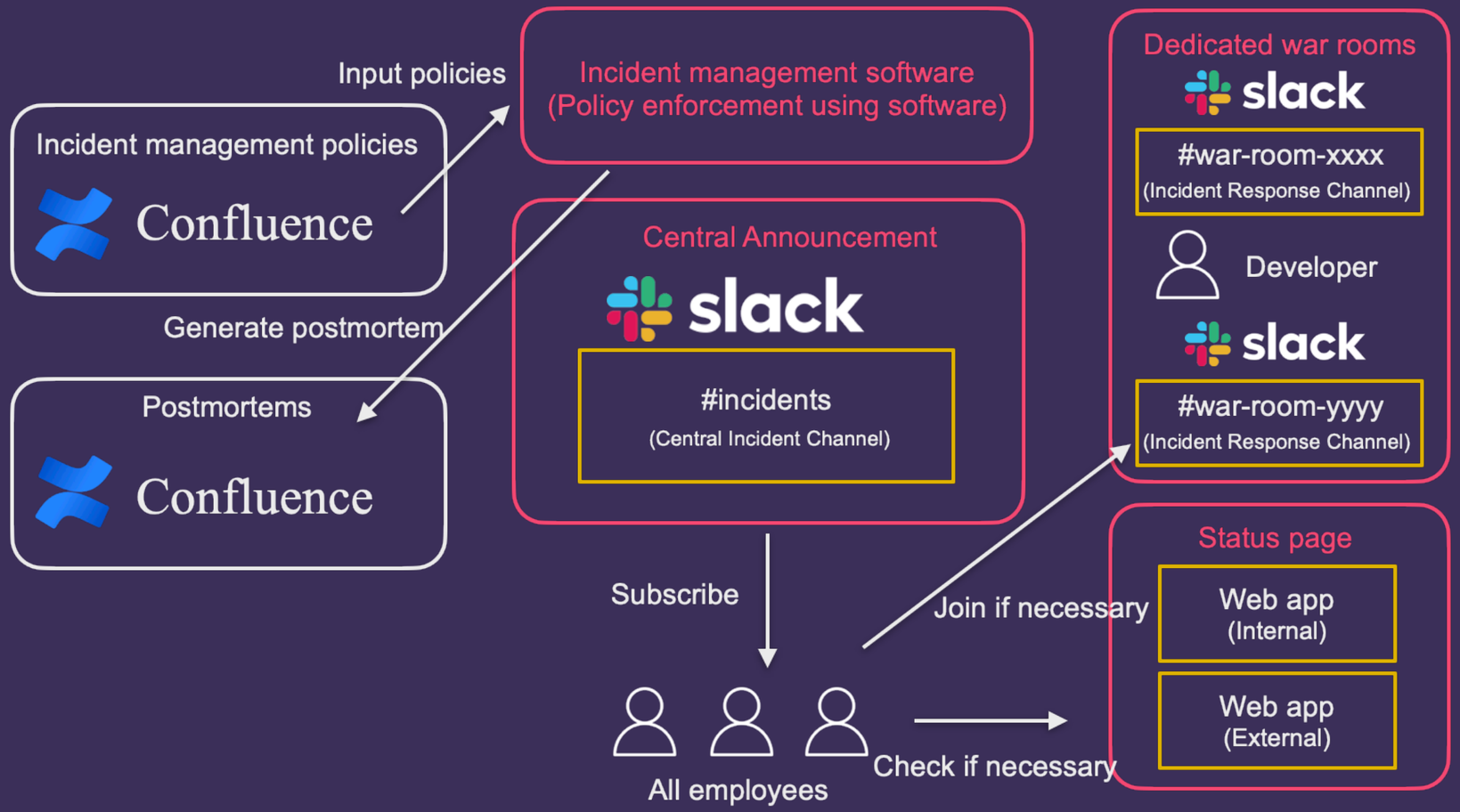
Environmental changes from Phase 1 to 2

- Our products have more customers.
- The more software engineers join, the more incidents happen.
- Increase of employees and time-zone gaps make sync and push-style communications tough.
  - In the first place, Launchable encourages async and written communications.

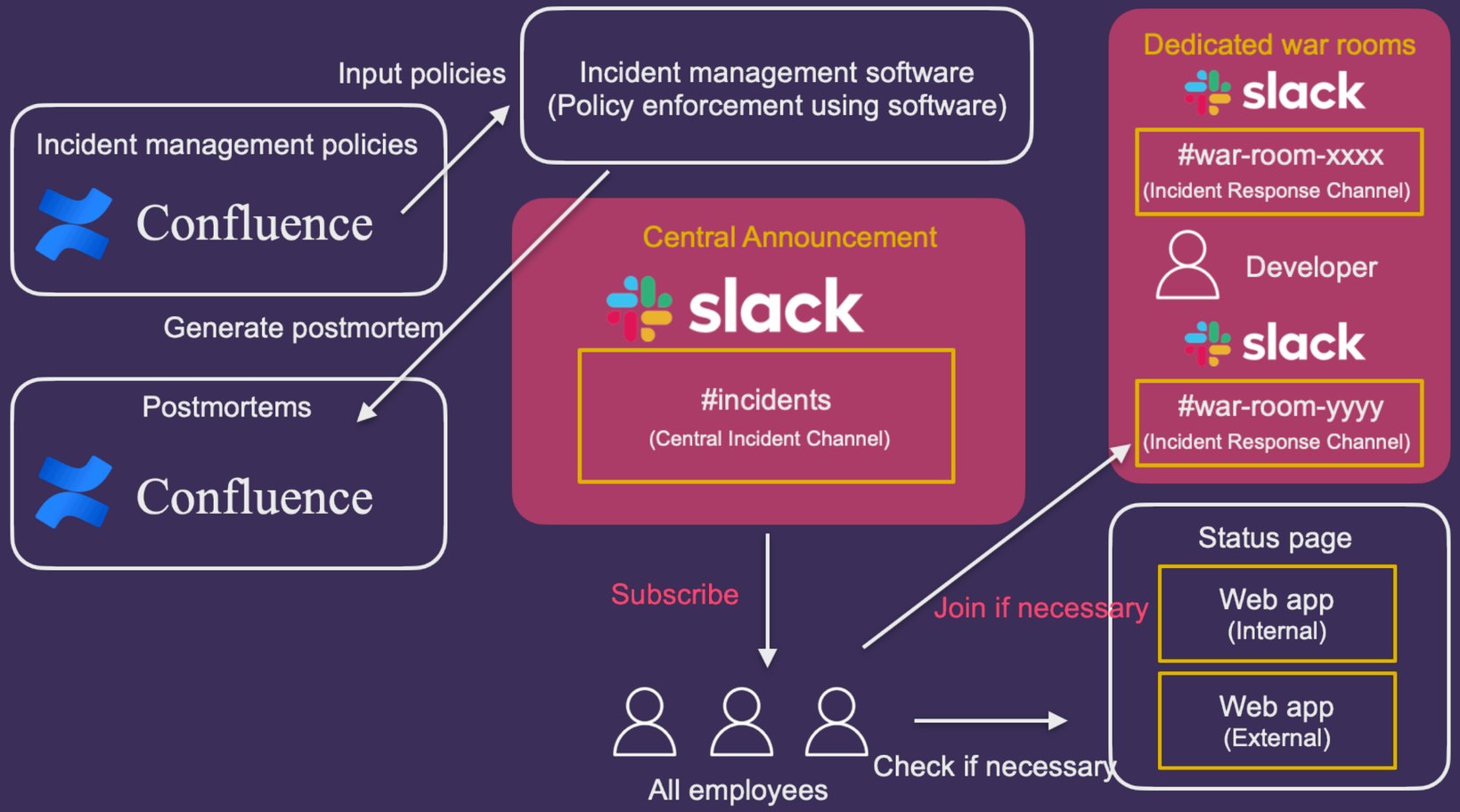
Strategy

- **Enforce incident management policies by software** not by documents.
- Involve appropriate people based on **pull-style communications**.
- **Use the current tool chains** in the company.
  - Too many new tools degrade teams' performance.
  - Use Slack as interactive communication places to keep flow info.
  - Use Confluence to keep stock info (non-urgent communications).

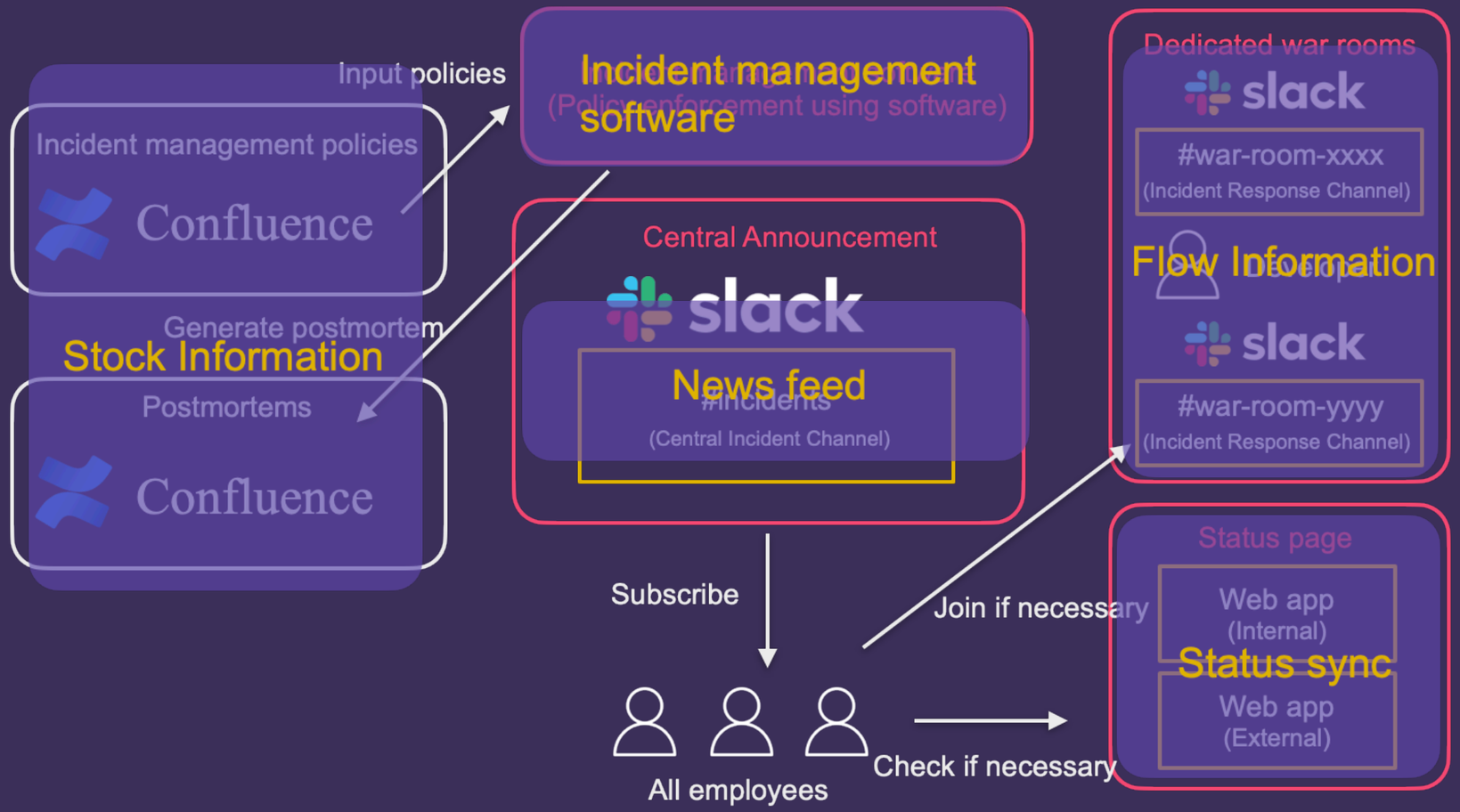
Incident management system (phase 1 to 2)



Incident management system (phase 1 to 2)

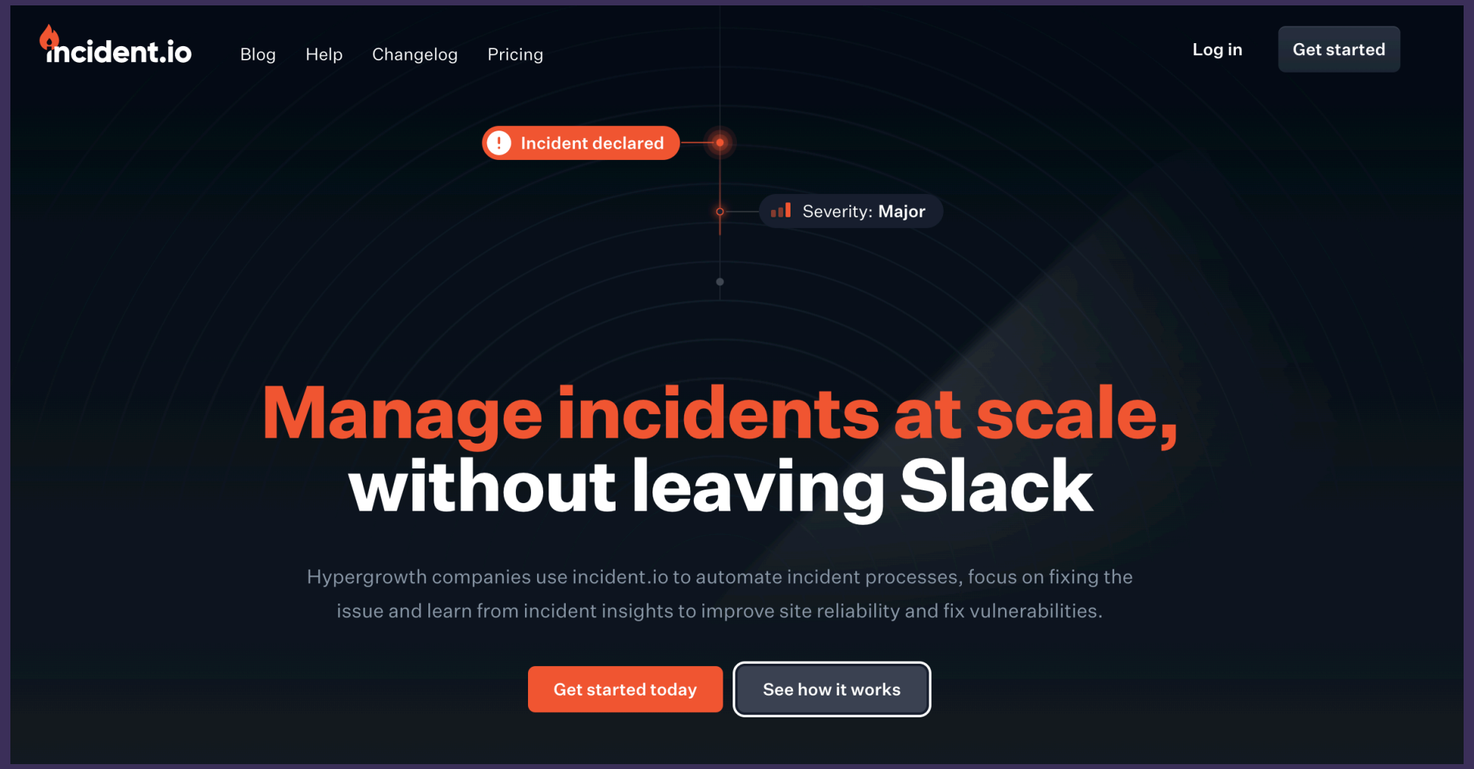


Incident management system (phase 1 to 2)



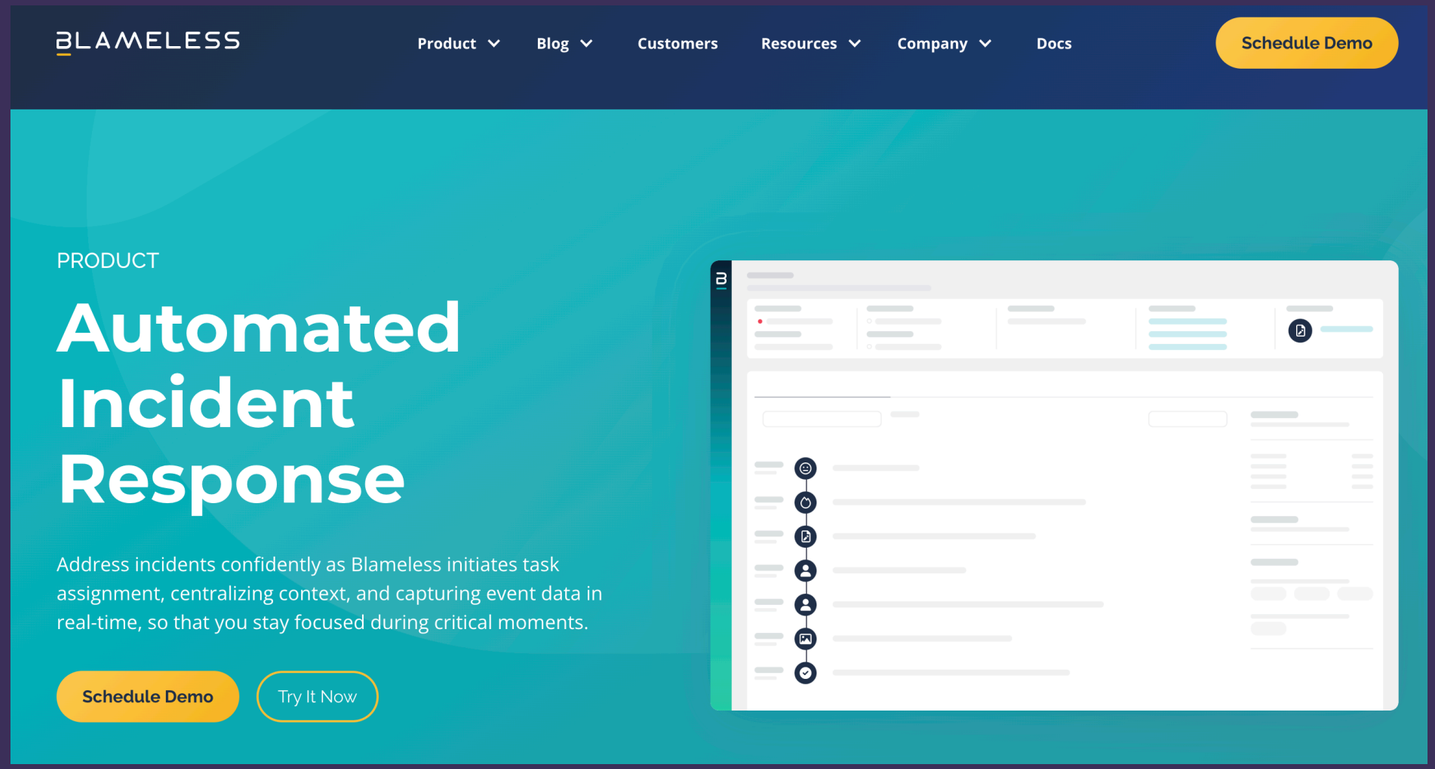
SaaS: incident.io

- https://incident.io/



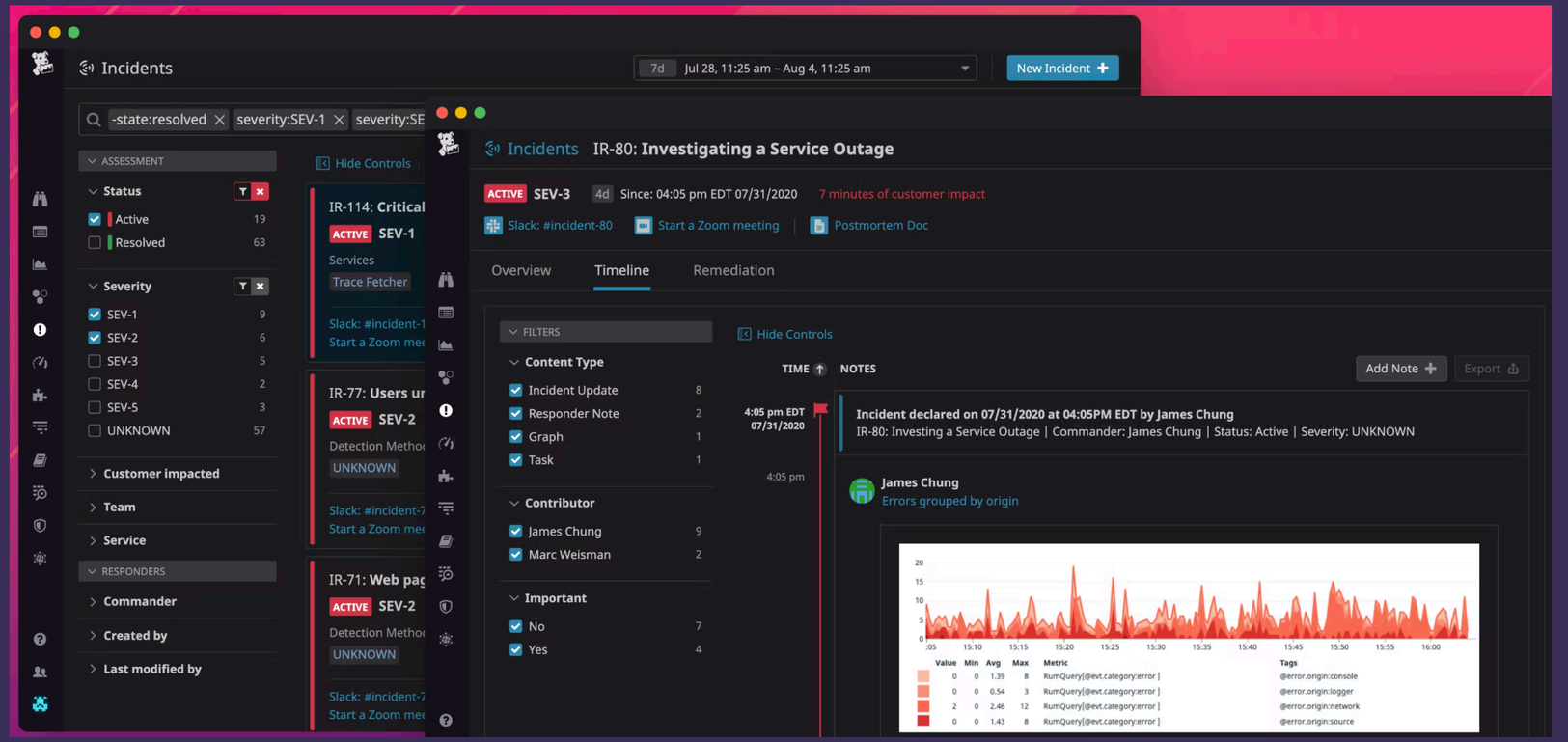
SaaS: Blameless

- https://www.blameless.com/product/incident-resolution



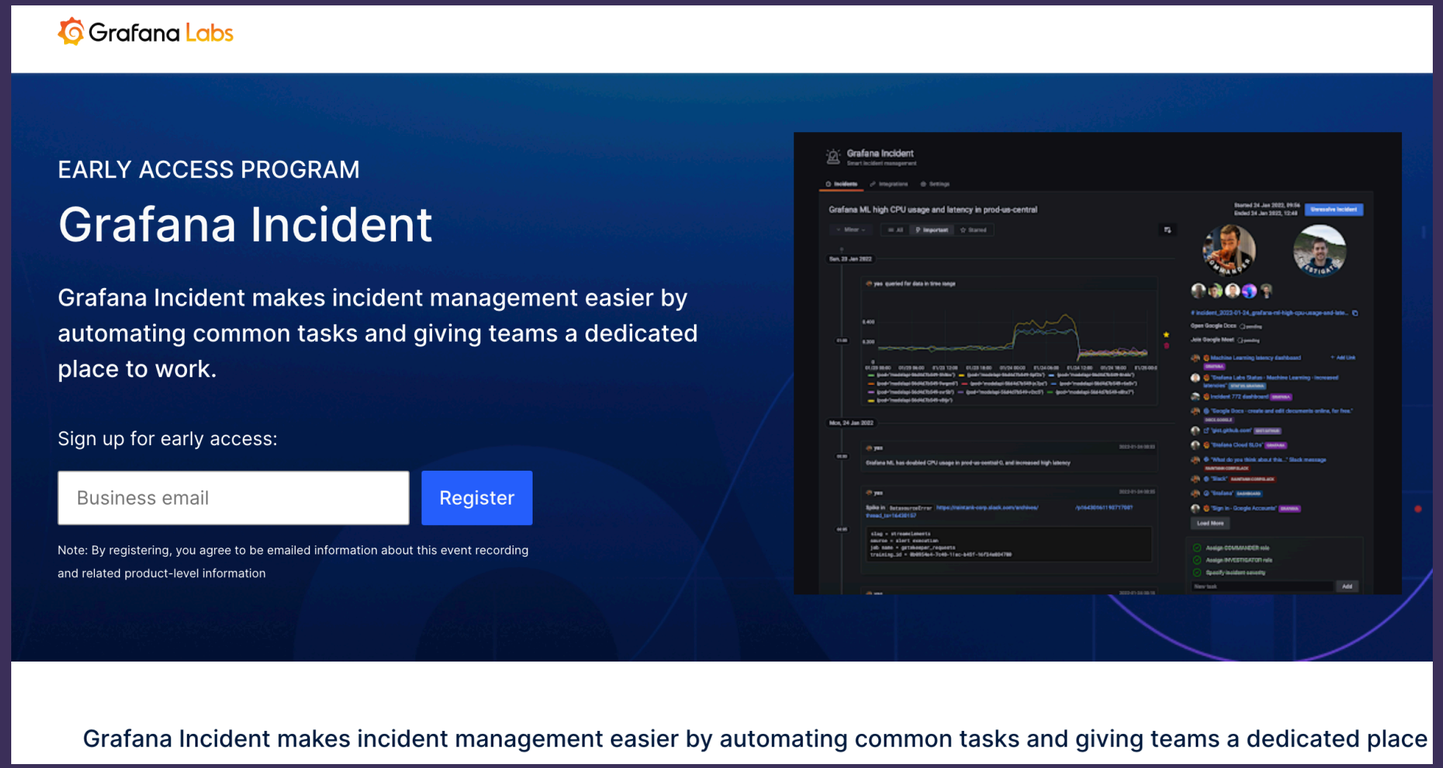
SaaS: Datadog Incident

- https://www.datadoghq.com/blog/incident-response-with-datadog/



SaaS: Grafana Incident

- https://go2.grafana.com/incident-beta-interest.html



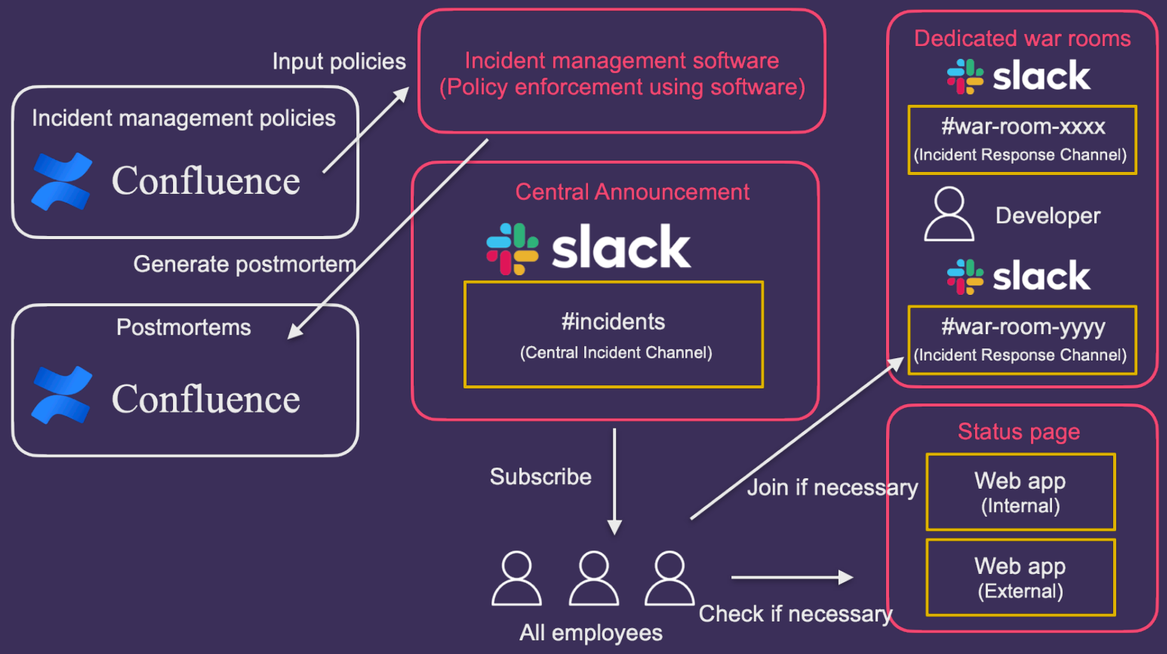
OSS: monzo/response

OSS version of incident.io

- https://github.com/monzo/response
- https://monzo.com/blog/2019/07/08/how-we-respond-to-incidents/

in-house tool: Slack App + Web App

- It's not so difficult to implement Slack App and Web App for this purpose.
- But... I want to use my time for other stuff.



SaaS vs OSS vs in-house tool

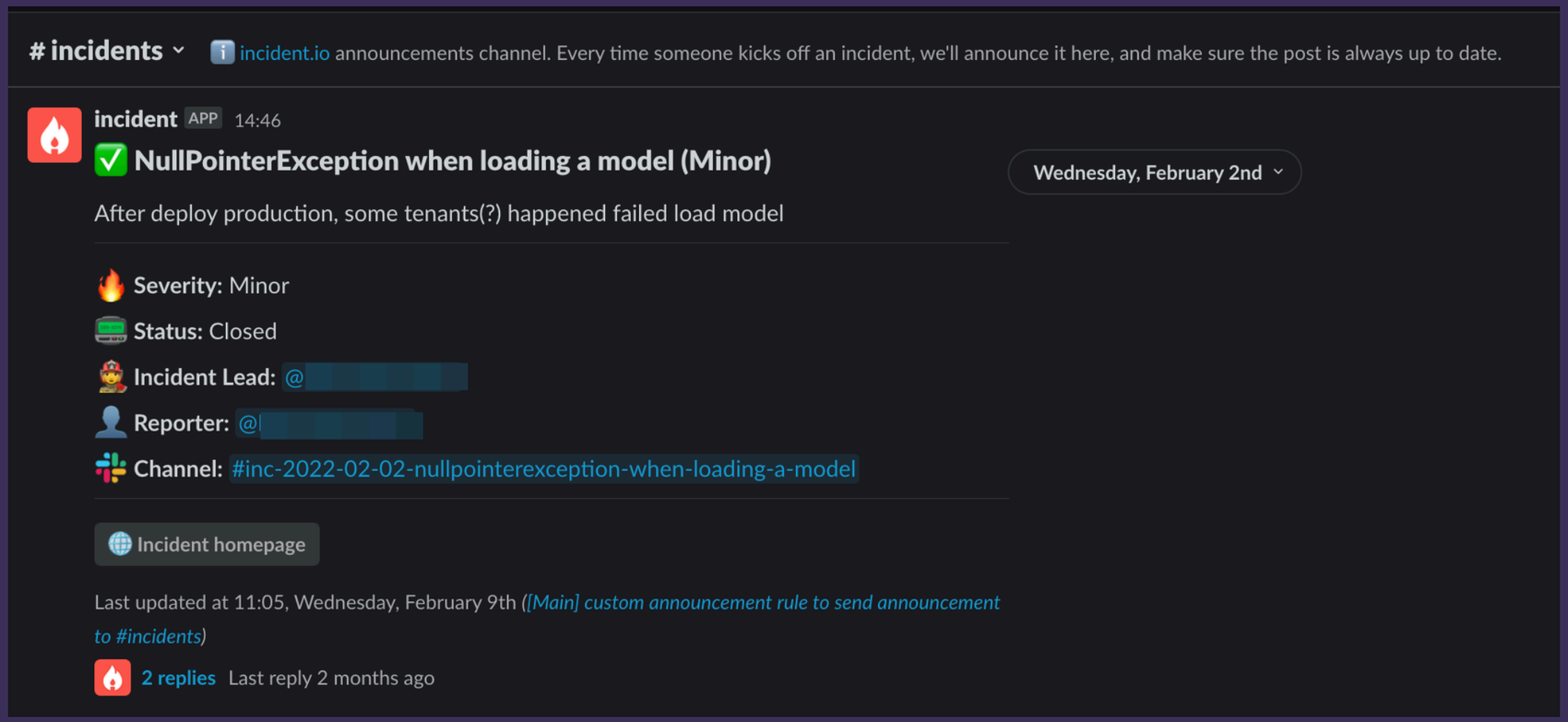
- We want to maximize developers' disposal time for product developments.
- We don't want to increase cognitive loads.
  - OSS and in-house tool needs code and document maintenance.
  - OSS and in-house tool needs evangelical activities for this type of tools.
- **Use SaaS if money allows (Buy, Not Build)**
  - Salaries for software engineers are way more expensive than SaaS cost.
  - SaaS improves their features as their business.
  - SaaS maintains documents as product features.

incident.io covers wide improvement targets

Improvement target	Actions from phase 0 to 1	Actions from phase 1 to 2
Transparency	- Encourage push communication	- **Encourage pull communication** - **Create war rooms** - **Share status pages**
Tangibility	- Automate parts of incident response flow	- **Automate entire incident response flow** - **Introduce incident lead role**
Trust	- Introduce blameless culture	- **Split lead and operation roles for complex incidents**
Time to Engagement (TTE)	- Automate incident announcements	- **Automate entire incident response flow** - Introduce on-call rotations - Expand follow-the-sun coverages
Time to Fix (TTF)	- Introduce observability	- Improve observability
Time to Triage (TTT)	- Introduce observability	- Improve observability
Time to Learn (TTL)	- Introduce postmortem template	- **Generate postmortem**
Time to Preparation (TTP)	- Create incident management policies	- **Enforce incident management policies** - **Self-service incident response trainings**

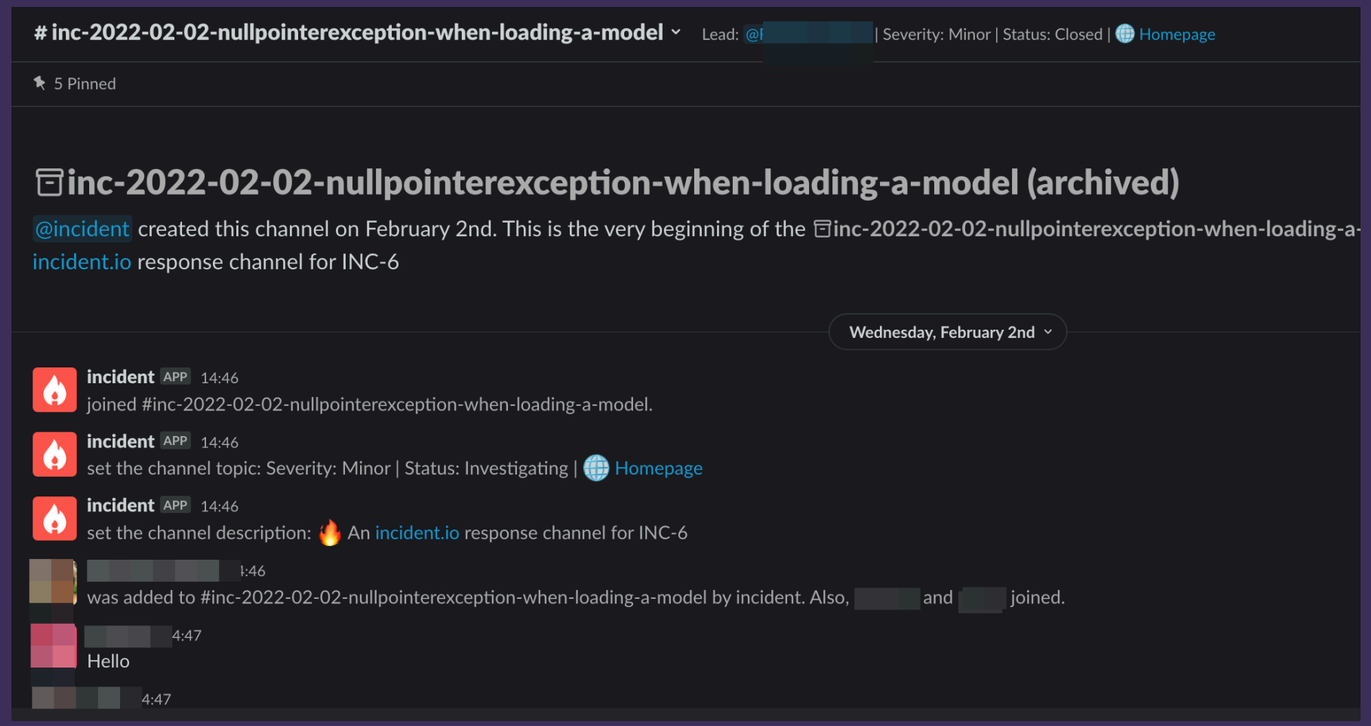
Central channel for all incidents

incident.io can share all incidents in the specified Slack channel.



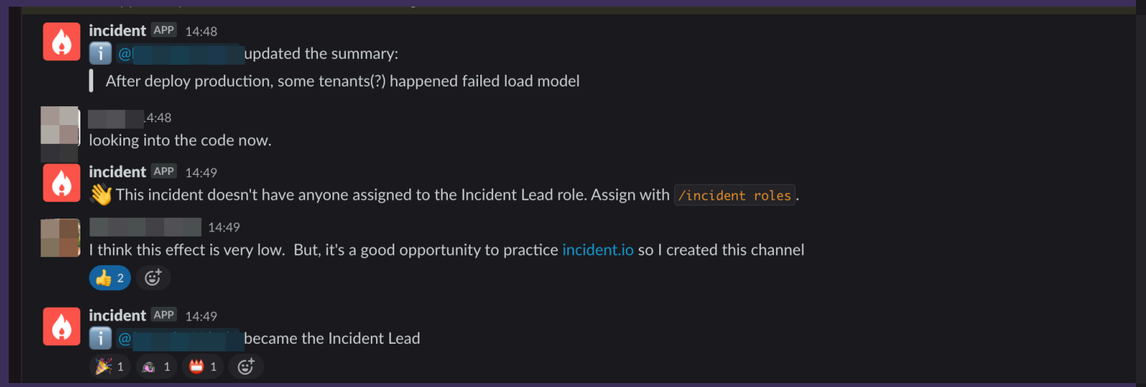
Dedicated war rooms (Slack channel)

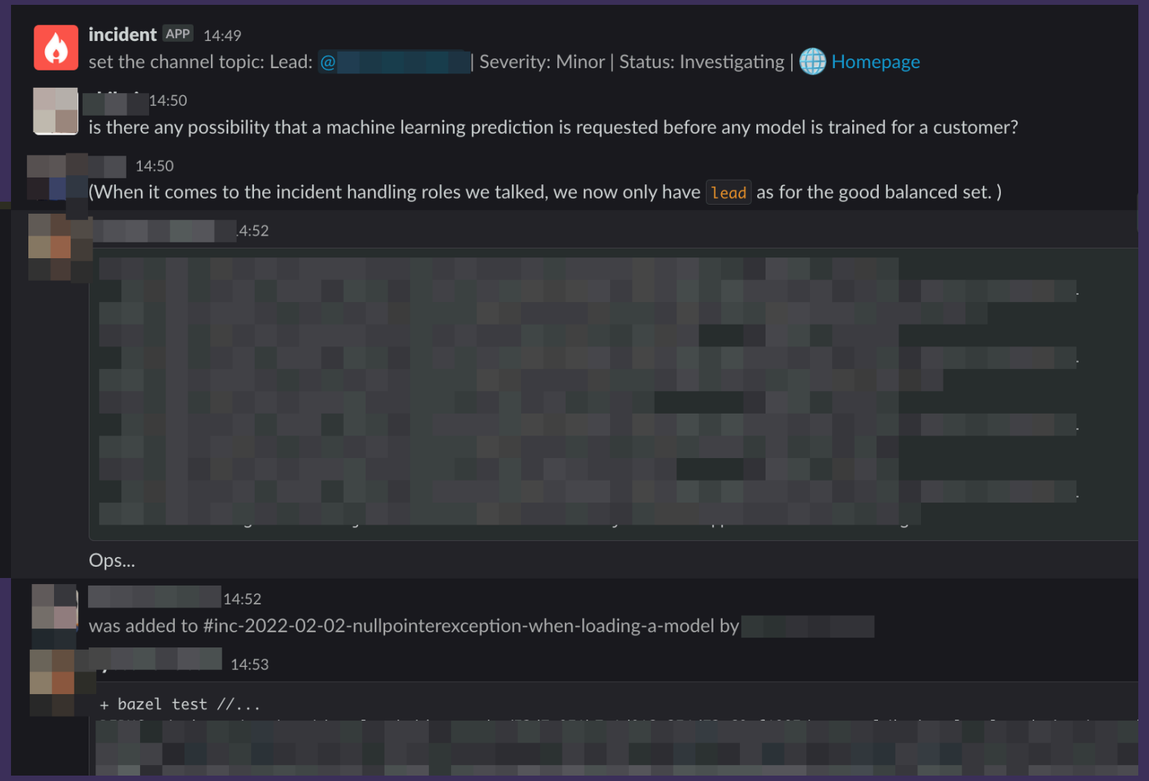
incident.io handles all tasks we want to complete for incident response initializations.



Dedicated war rooms (Slack channel)

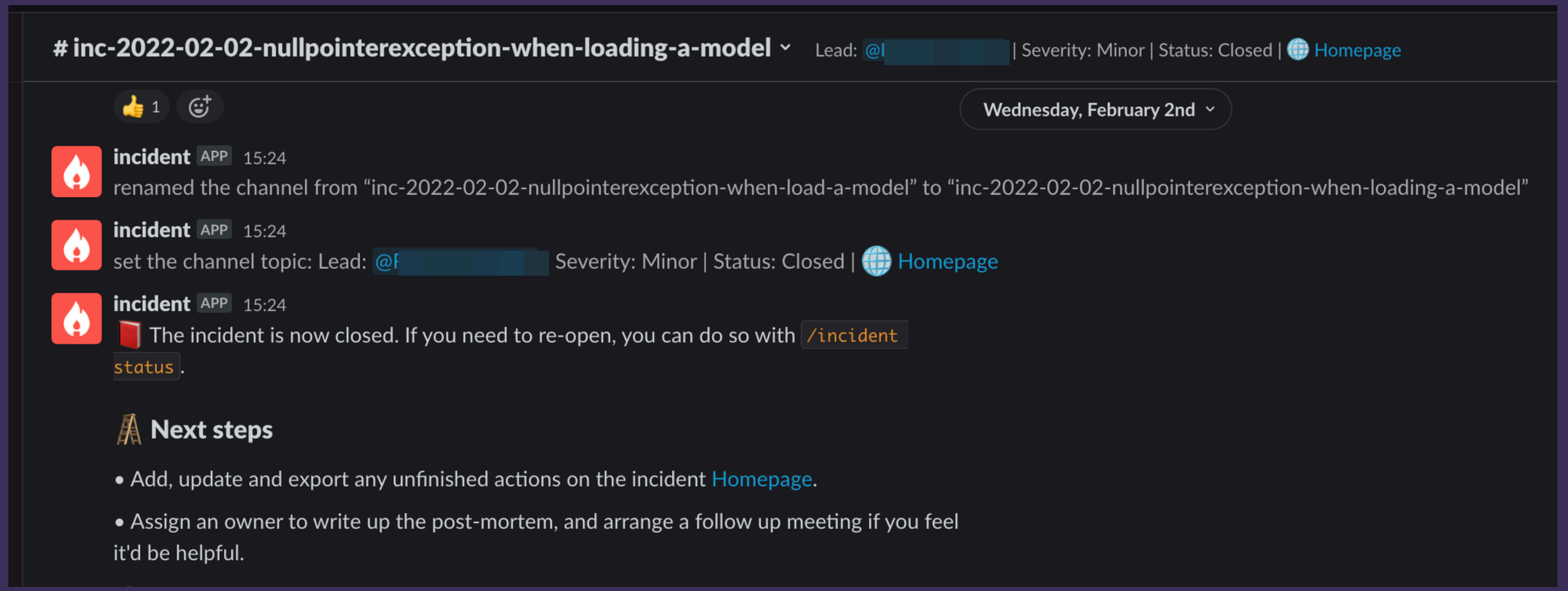
incident.io can assist incident responses.





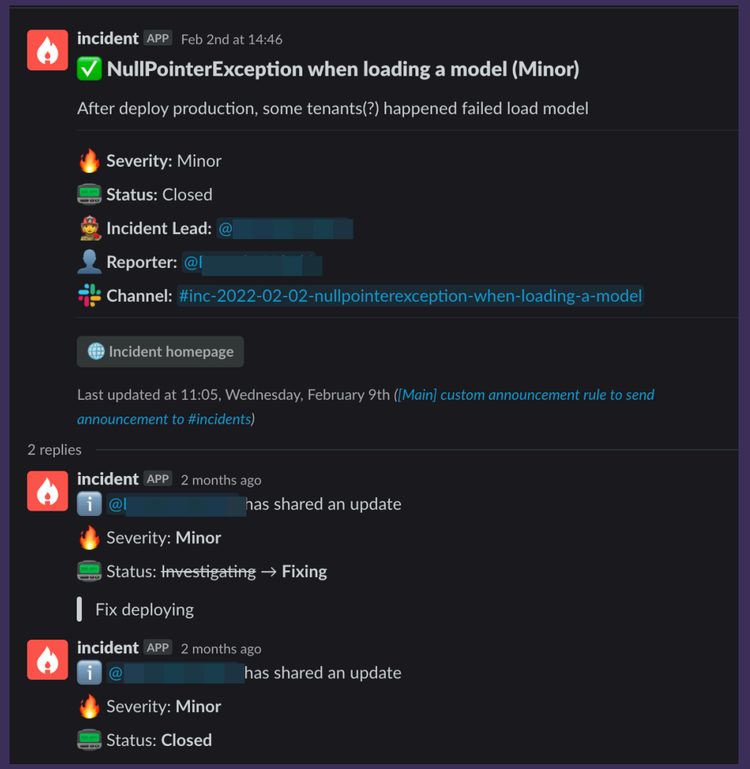
Dedicated Slack channel (closing incident)

At the end of incident responses, incident.io tells us what we need to be done next.



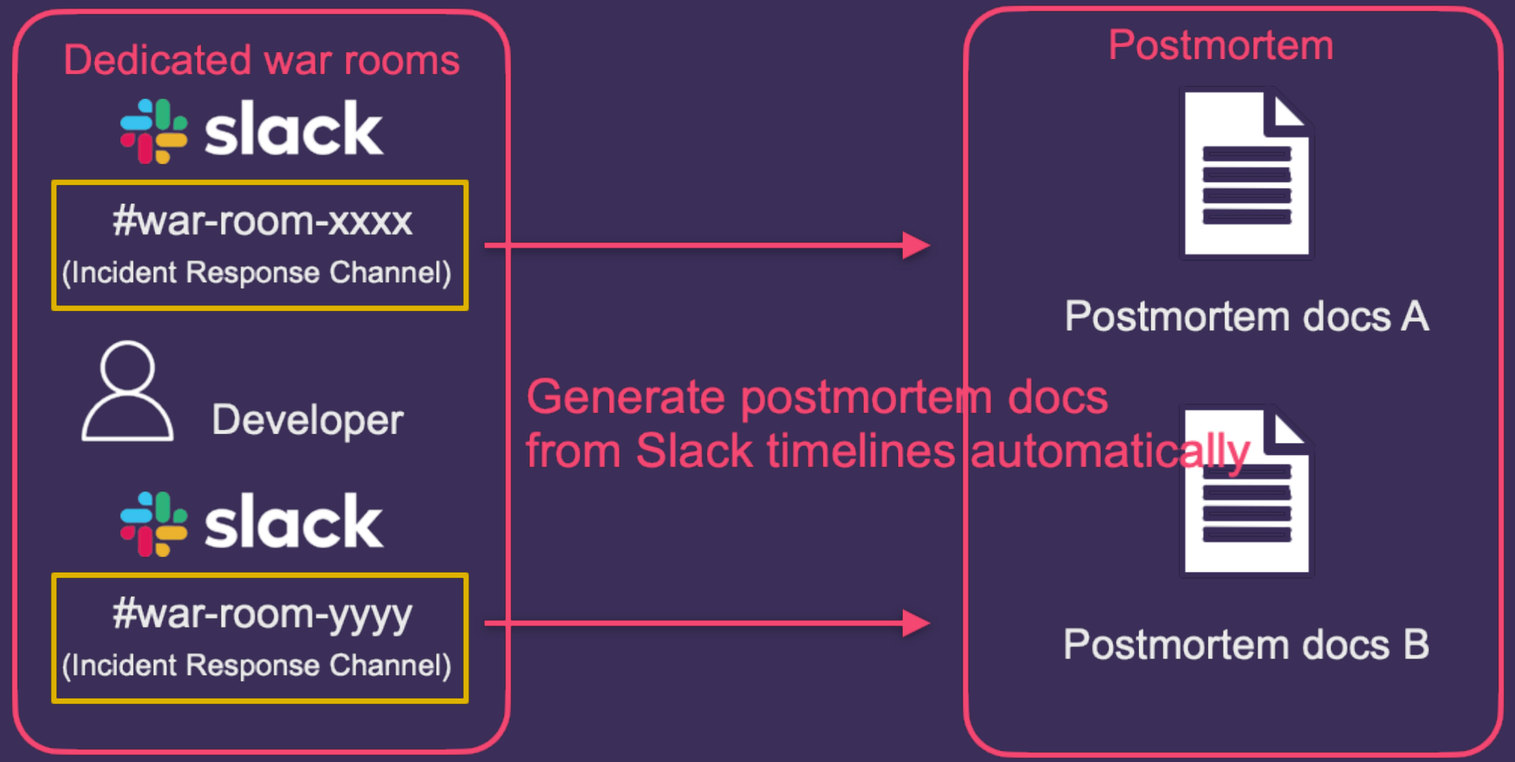
Status updates at central channel

incident.io automatically syncs the latest status of incidents at the central channel.



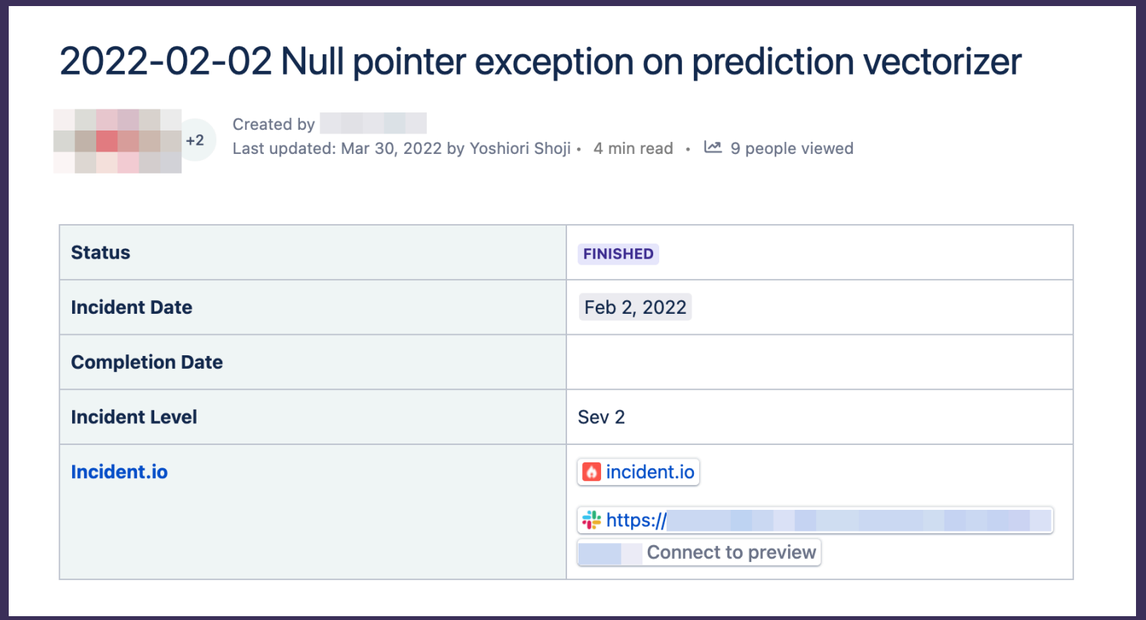
Postmortem generation

incident.io can collect timelines from war rooms and generates postmortems.



Postmortem generation

We can generate a postmortem documents using incident.io.

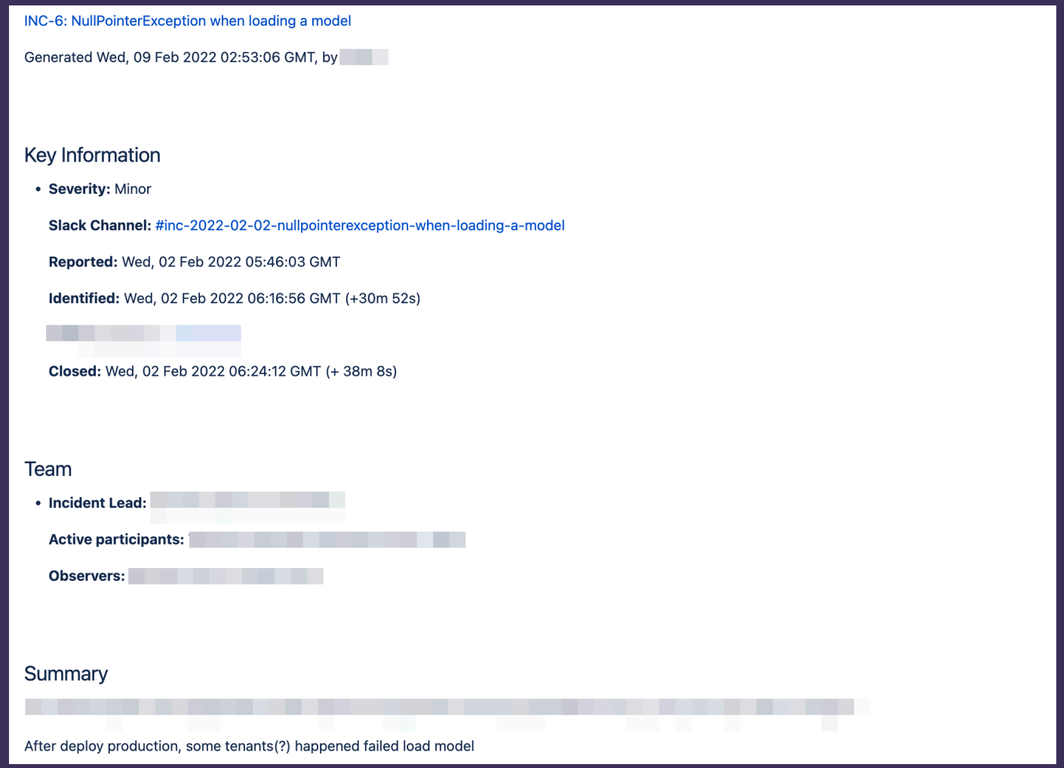


2022-02-02 Null pointer exception on prediction vectorizer

+2 Created by

Last updated: Mar 30, 2022 by Yoshiori Shoji • 4 min read • 9 people viewed

Status	FINISHED
Incident Date	Feb 2, 2022
Completion Date	
Incident Level	Sev 2
Incident.io	incident.io https:// Connect to preview



INC-6: NullPointerException when loading a model

Generated Wed, 09 Feb 2022 02:53:06 GMT, by

Key Information

- **Severity:** Minor

Slack Channel: #inc-2022-02-02-nullpointerexception-when-loading-a-model

Reported: Wed, 02 Feb 2022 05:46:03 GMT

Identified: Wed, 02 Feb 2022 06:16:56 GMT (+30m 52s)

Closed: Wed, 02 Feb 2022 06:24:12 GMT (+ 38m 8s)

Team

- **Incident Lead:**

Active participants:

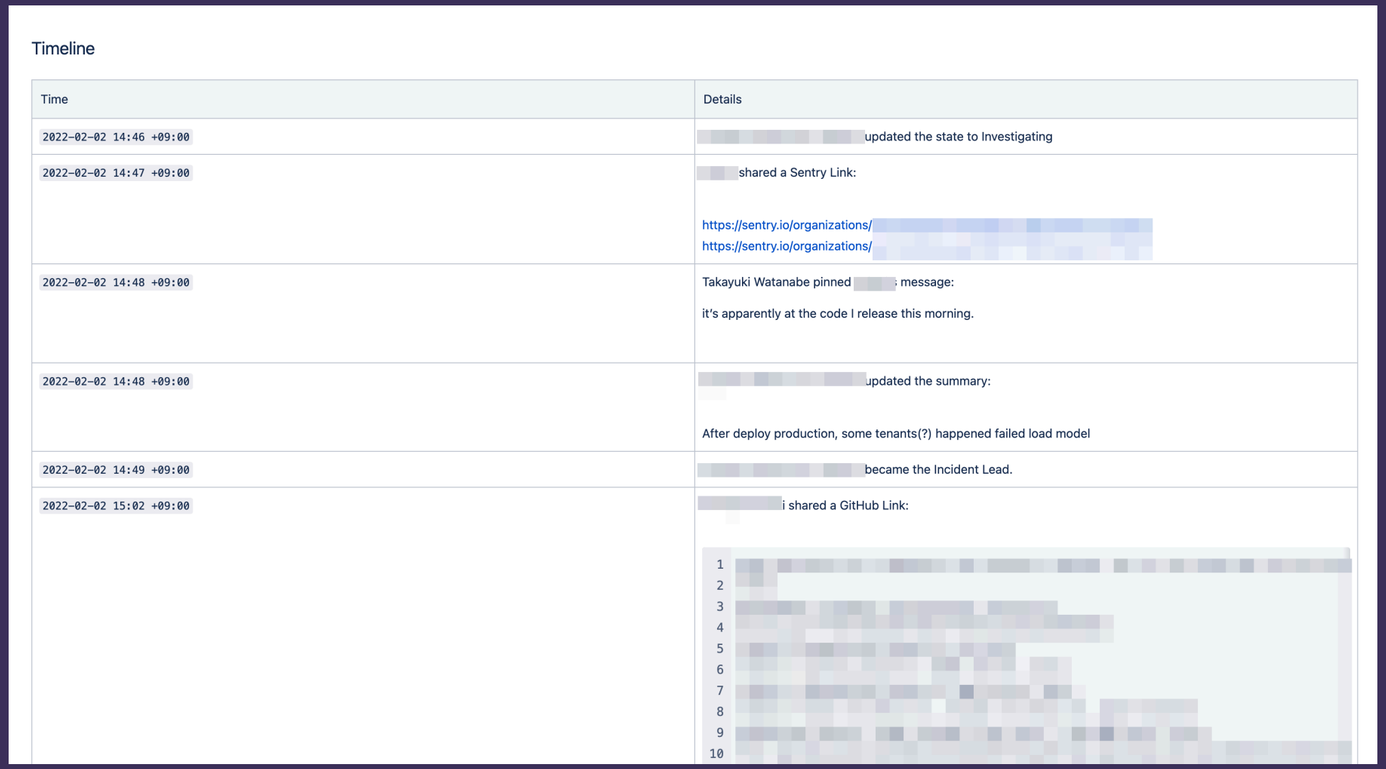
Observers:

Summary

After deploy production, some tenants(?) happened failed load model

Postmortem generation

We can collect timelines from dedicated Slack channels.



Timeline

Time	Details
2022-02-02 14:46 +09:00	updated the state to Investigating
2022-02-02 14:47 +09:00	shared a Sentry Link: https://sentry.io/organizations/ https://sentry.io/organizations/
2022-02-02 14:48 +09:00	Takayuki Watanabe pinned : message: it's apparently at the code I release this morning.
2022-02-02 14:48 +09:00	updated the summary: After deploy production, some tenants(?) happened failed load model
2022-02-02 14:49 +09:00	became the Incident Lead.
2022-02-02 15:02 +09:00	i shared a GitHub Link:

Postmortem generation

Contributors

Mitigators

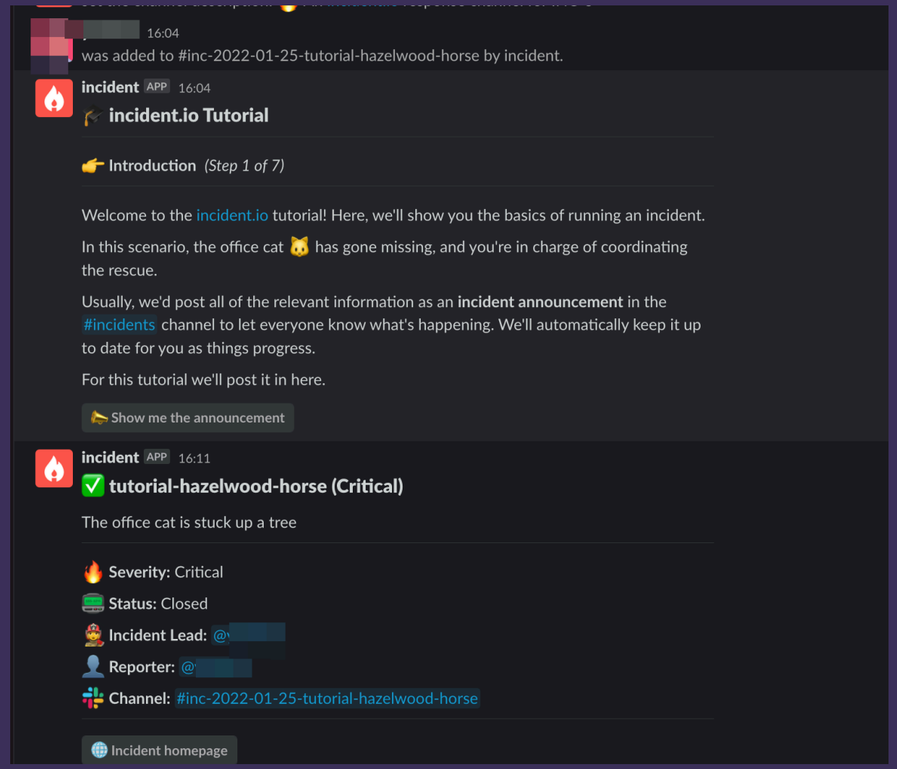
Learnings

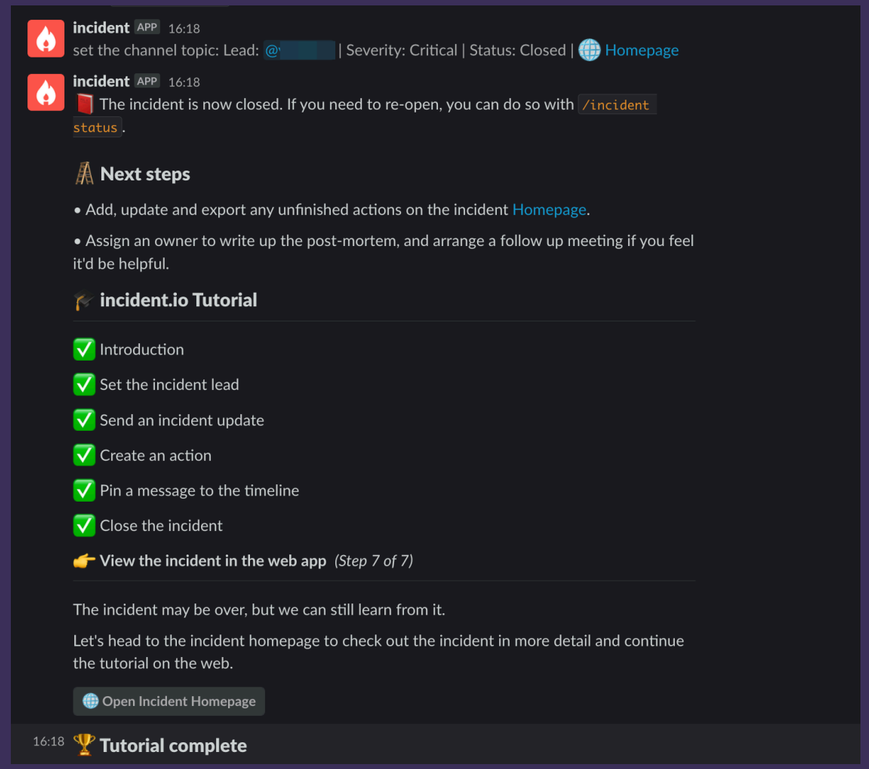
Actions

Owner	Action	Status	Link
		Outstanding	

Self-training mode

incident.io has a mode to walk though dummy incident responses on Slack.





Introduction of lead role

- We need communication leads when incidents are complex
- However, for most of incident, a single person can be responsible for operations and communications.
- So, adding a lead role only is prudent so we don't make incident managements overly complex.

We have more rooms to improve!

Recap

- Incident management has a life cycle.
  - Preparation -> Detection -> Recovery -> Post-incident actions -> Preparation
- Incident response roles and structures exist to embody 3T.
  - Transparency
  - Tangibility
  - Trust
- Choosing strategy and tools makes incident managements at startups sensible.

Thanks

References

Incident management

1. Atlassian: Understanding incident response roles and responsibilities, https://www.atlassian.com/incident-management/incident-response/roles-responsibilities

2. PagerDuty Incident Response Training, https://response.pagerduty.com/training/overview/.

3. Anatomy of an Incident, Ayelet Sachto, Adrienne Walcer, and Jessie Yang, 2022.

4. US Federal Emergency Management Agency, Emergency Management Institute ICS Resource Center, https://training.fema.gov/emiweb/is/icsresource/.

5. The National Institute of Standards and Technology SP 800-61, Computer Security Incident Handling Guide, http://dx.doi.org/10.6028/NIST.SP.800-61r2.

6. Introduction: Incident Response overview, Gov UK National Cyber Security Centre, https://www.ncsc.gov.uk/collection/incident-management/incident-response

7. Incident Review and Postmortem Best Practices, https://newsletter.pragmaticengineer.com/p/incident-review-best-practices

8. Incident Review Practices [The Pragmatic Engineer Newsletter], https://docs.google.com/spreadsheets/d/1GPINipdf-l2H05iKOUbpkrqwlZ61ZCJDnwY5iE8LtRM/edit#gid=0

SRE

1. Google SRE book Chapter 14 - Managing Incidents, https://sre.google/sre-book/managing-incidents/
2. Postmortem Action Items: Plan the Work and Work the Plan, Sue Lueder and Betsy Beyer (Google), USENIX SRECon 2017, https://www.usenix.org/conference/srecon17americas/program/presentation/lueder.
3. Google SRE book Chapter 15 - Postmortem Culture: Learning from Failure, https://sre.google/sre-book/postmortem-culture/.
4. Postmortem Metadata Index, https://postmortems.app/.
5. The Art of SLOs, Google Site Reliability Engineering, https://sre.google/resources/practices-and-processes/art-of-slos/
6. danluu/post-mortems: A collection of postmortems, https://github.com/danluu/post-mortems.
7. Great Incident Review Examples, The Pragmatic Engineer, https://blog.pragmaticengineer.com/postmortem-best-practices/#great-incident-review-examples

DevOps performance metrics

1. Accelerate: The Science of Lean Software and DevOps: Building and Scaling High Performing Technology Organizations, 2018.
2. GoogleCloudPlatform/fourkeys, https://github.com/GoogleCloudPlatform/fourkeys
3. Are you an Elite DevOps performer? Find out with the Four Keys Project, Google Cloud, https://cloud.google.com/blog/products/devops-sre/using-the-four-keys-to-measure-your-devops-performance
4. DORA DevOps Quick Check., https://www.devops-research.com/quickcheck.html

SaaS and OSS

1. Datadog, https://www.datadoghq.com/blog/incident-response-with-datadog/
2. incident.io, https://incident.io/
3. jeli, https://www.jeli.io/
4. monzo/response, https://monzo.com/blog/2019/07/08/how-we-respond-to-incidents
5. Etsy/morgue, https://github.com/etsy/morgue