From Dashboard Soup to Observability Lasagna

Building Better Layers

Martha Lambert

Product Engineer

incident.io

What we'll cover

1. A process to unsoup your dashboards
2. The importance of a layered stack
3. Technical tips for great o11y UX


555-3215

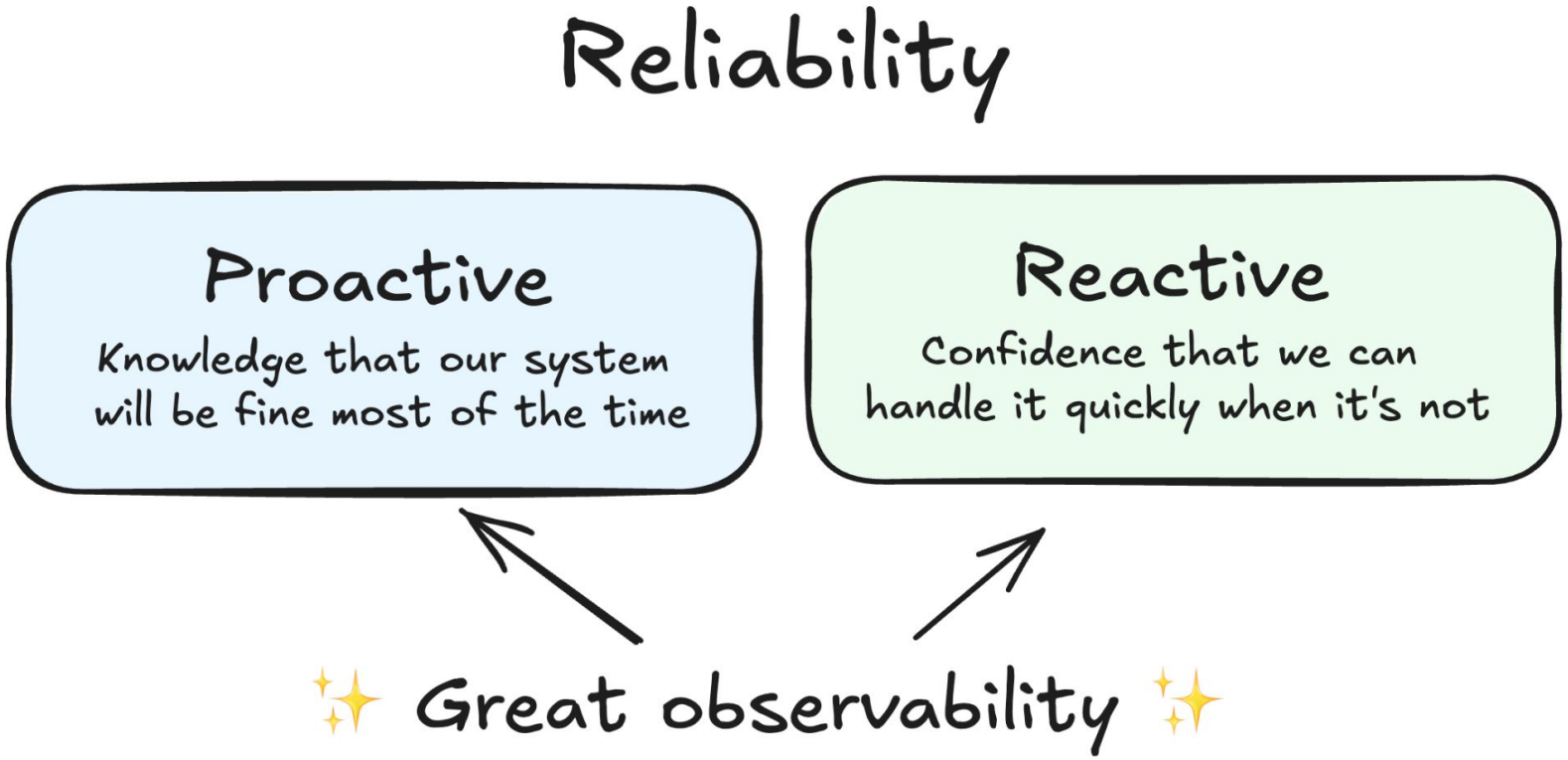
Reliability

Proactive

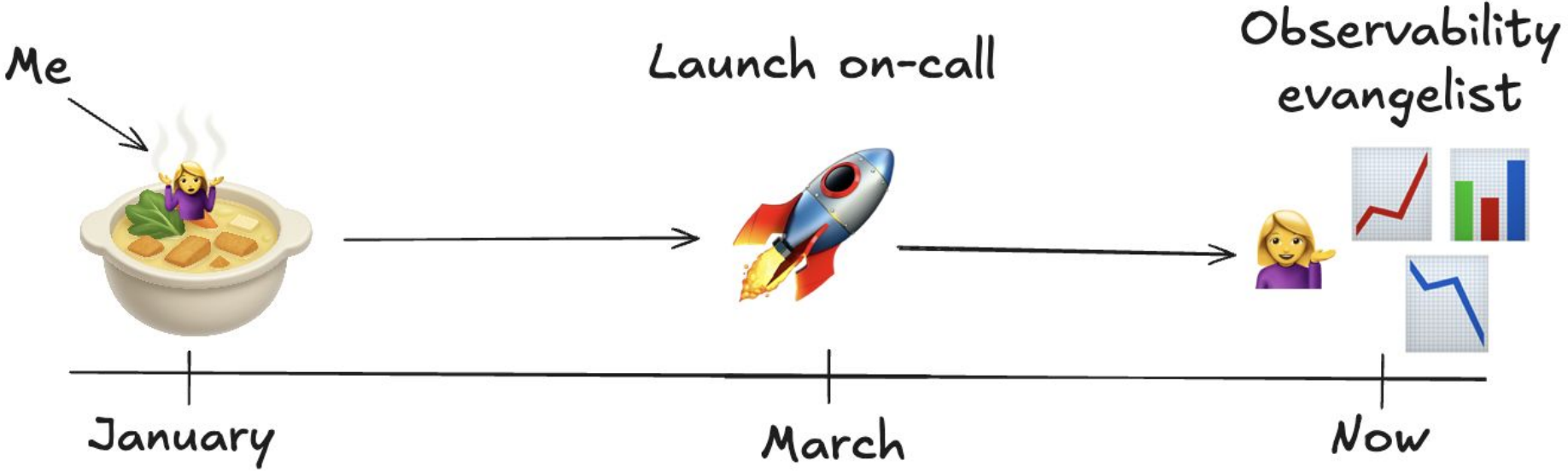
Knowledge that our system will be fine most of the time

Reactive

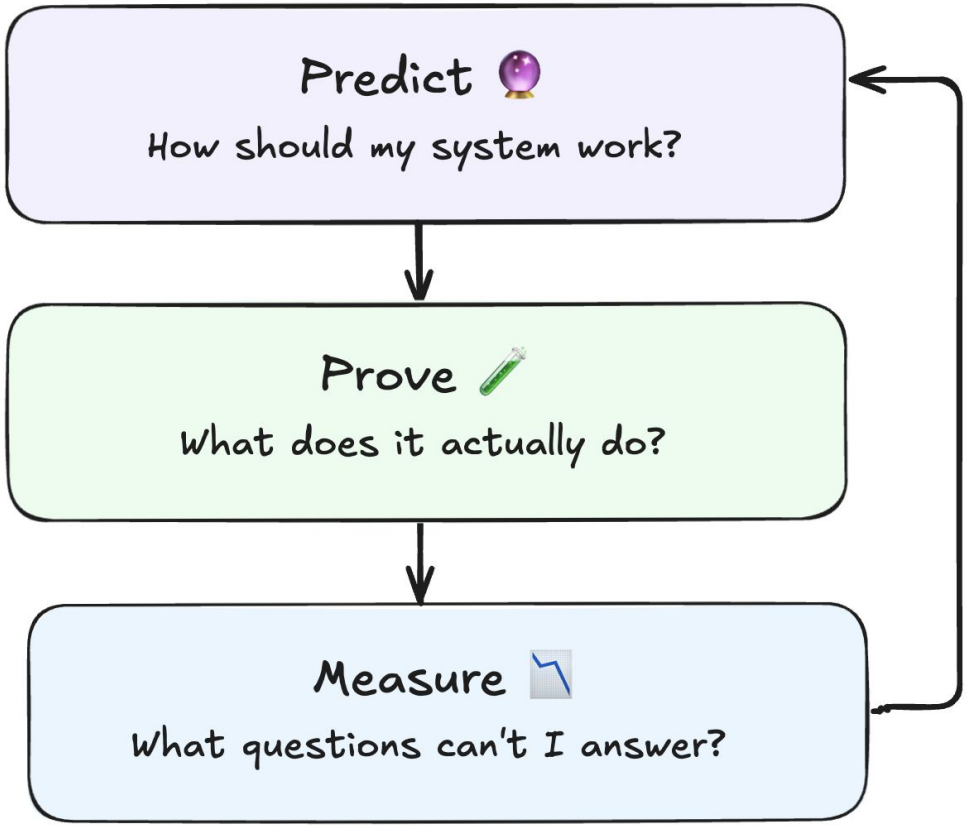
Confidence that we can handle it quickly when it's not


Reliability
Proactive
Knowledge that our system will be fine most of the time
Reactive
Confidence that we can handle it quickly when it's not
✨ Great observability ✨


Me

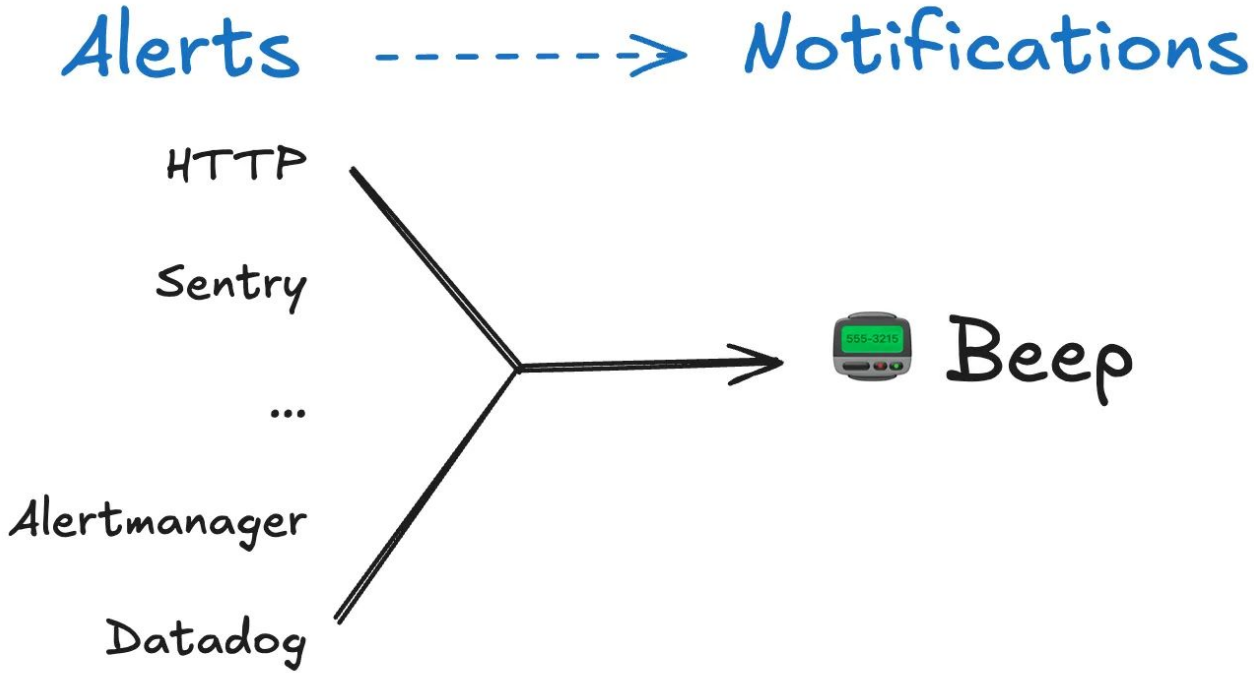

Me
Launch on-call
Observability evangelist
January
March
Now

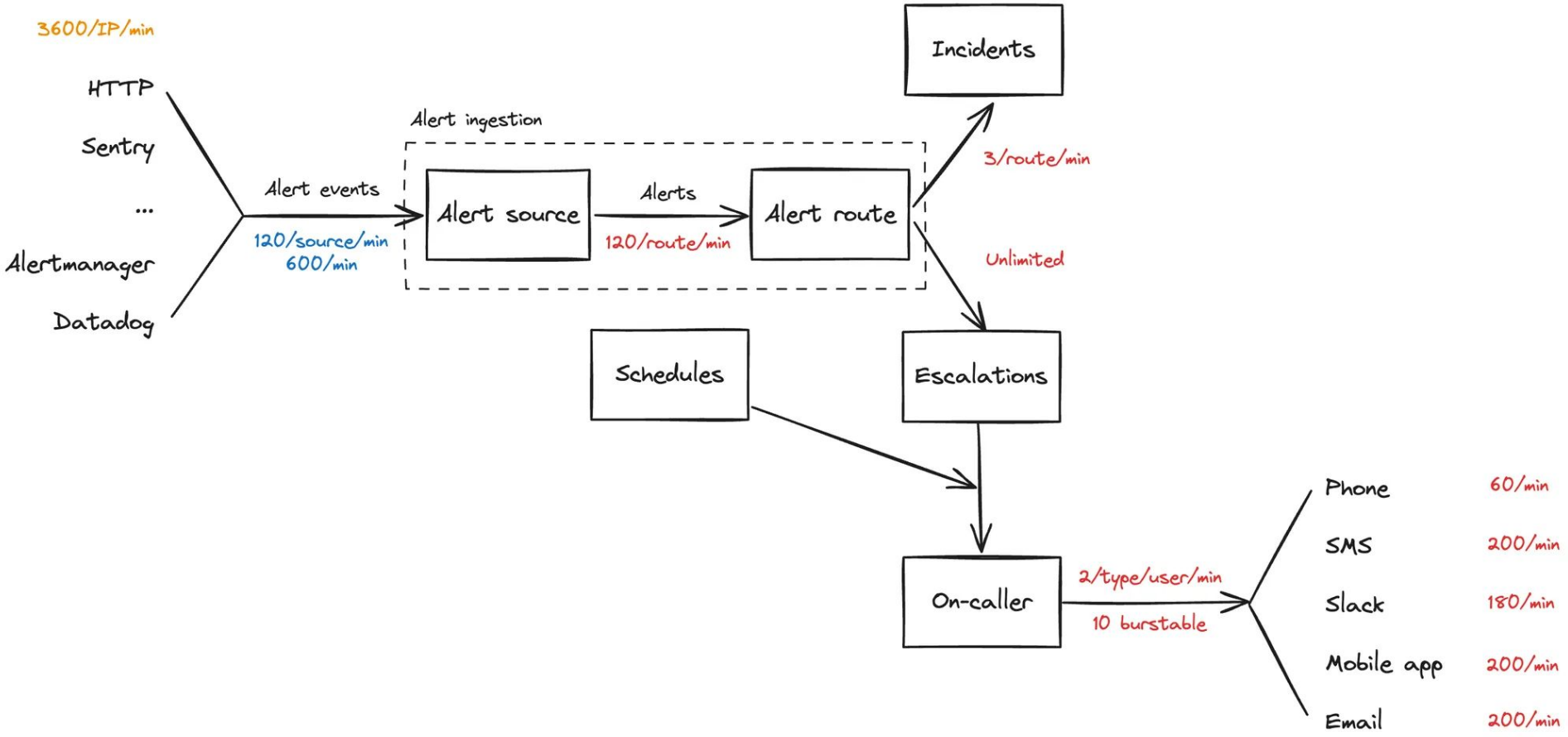
Unsouping our stack

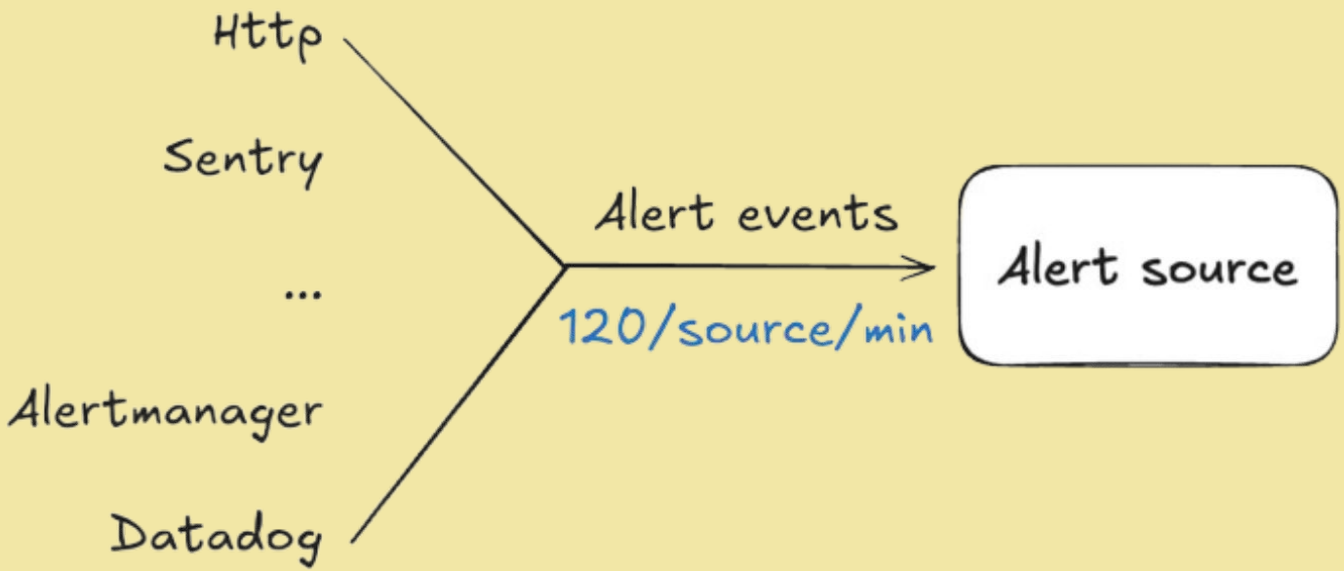


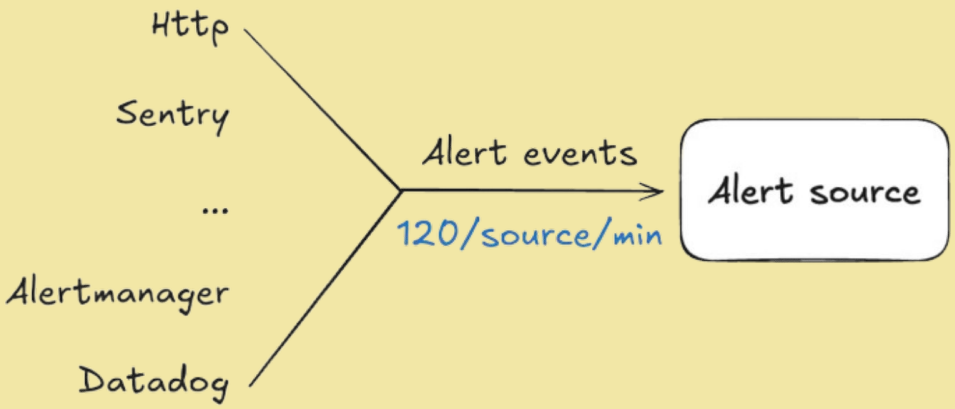
Predict

How should my system work?


Alerts
Notifications
HTTP
Sentry
...
Alertmanager
Datadog
555-3215
Beep


3600/IP/min
HTTP
Sentry
...
Alertmanager
Datadog
Alert events
120/source/min
600/min
Alert ingestion
Alert source
Alerts
120/route/min
Alert route
Incidents
3/route/min
Unlimited
Schedules
Escalations
On-caller
2/type/user/min
10 burstable
Phone
60/min
SMS
200/min
Slack
180/min
Mobile app
200/min
Email
200/min


Http
Sentry
...
Alertmanager
Datadog
Alert events
120/source/min
Alert source



Can we handle multiple alert storms at once?

Prove

What actually happens?

We can handle multiple alert storms at once

We can handle multiple alert storms at once

- We’re doing what our users expect
- The rest of our app is unaffected
- We could handle more load if we needed to

What questions can’t you answer?

- What are we rate limiting?
- What delays are our users experiencing?
- Where are our bottlenecks?

Measure

What actually happens?

But how does dashboard soup happen?

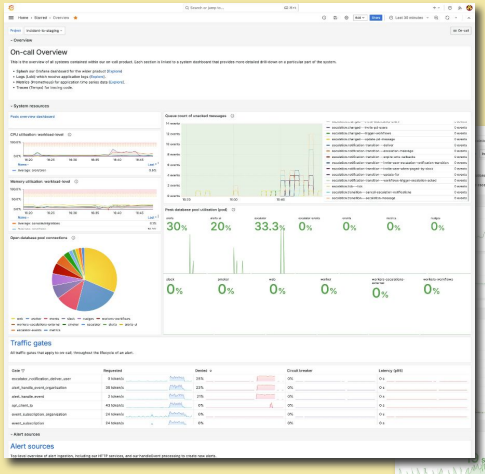
1. They answer overly-specific, now irrelevant questions

2. They are static and disconnected from the rest of your debugging stack

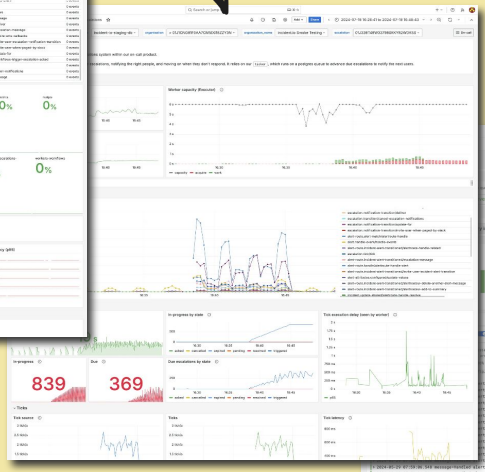
Your dashboards are a product

And your engineers are your customers

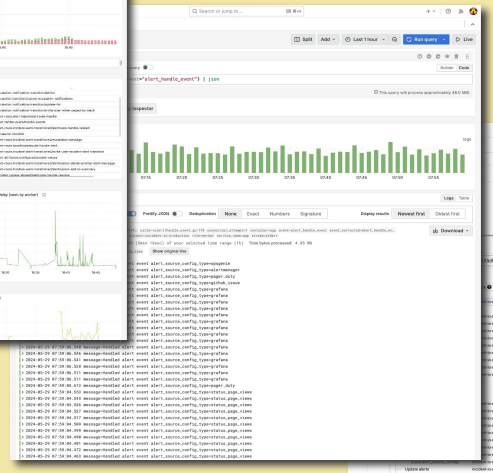
The Observability Lasagna



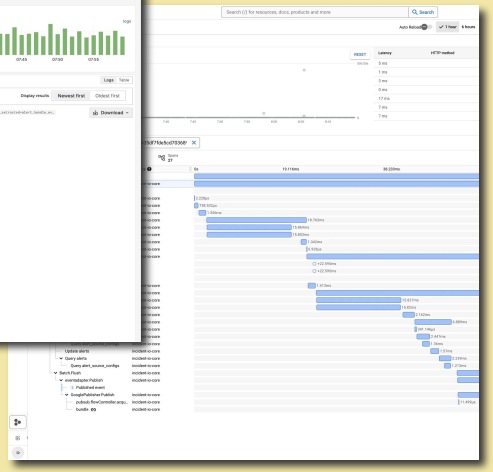
Overview Dashboard



System Dashboard

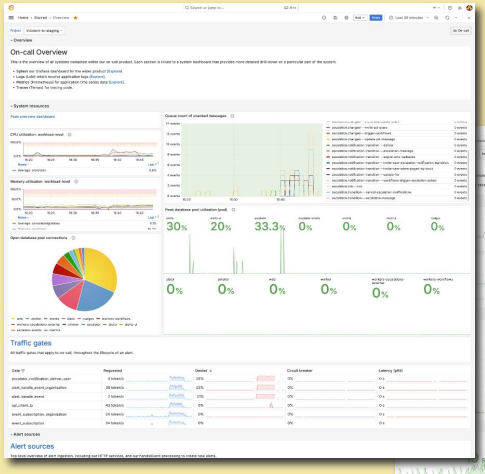


Logs

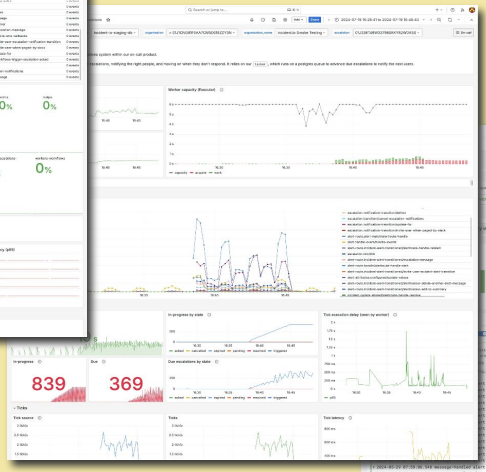


Traces

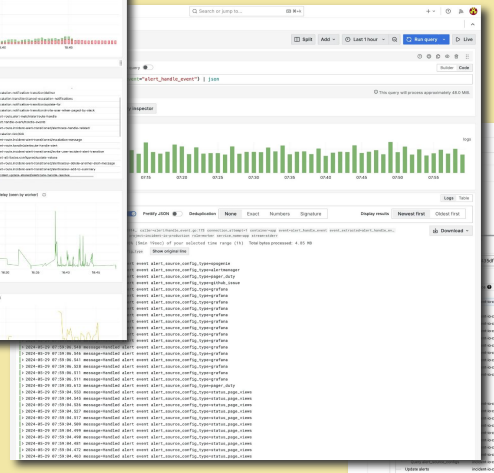
The Observability Lasagna



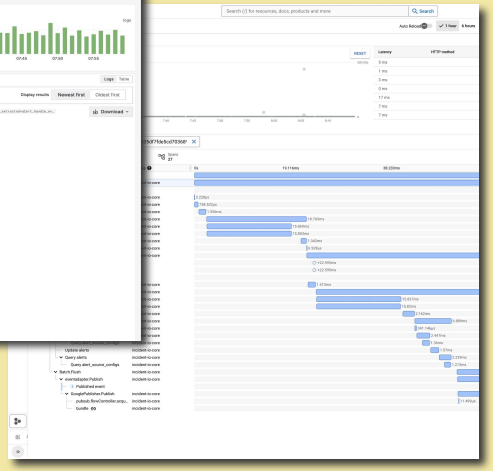
Overview Dashboard



System Dashboard



Logs

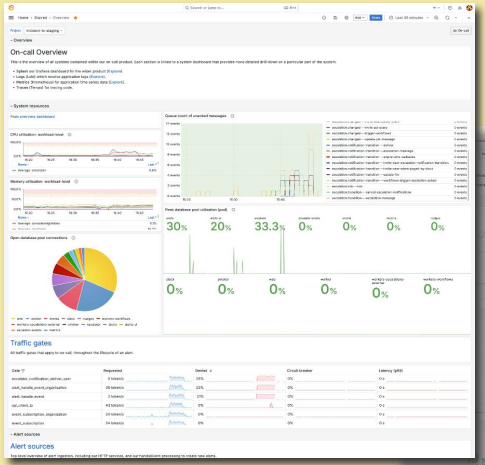


Traces

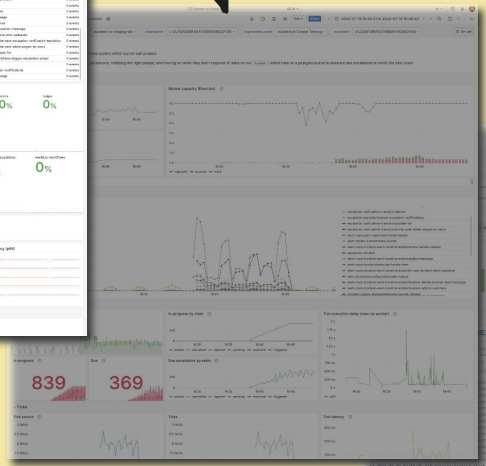
Connect your layers

Each layer of your stack should clearly point to the next level down

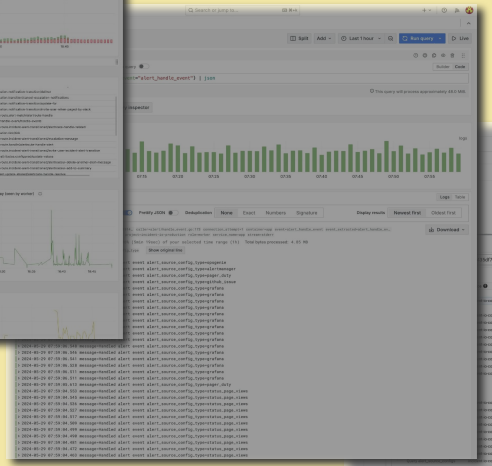
The Observability Lasagna



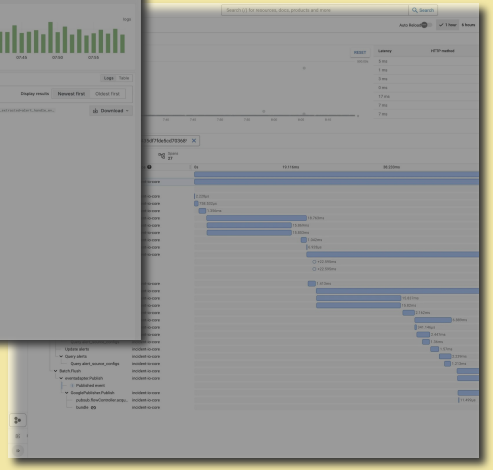
Overview Dashboard



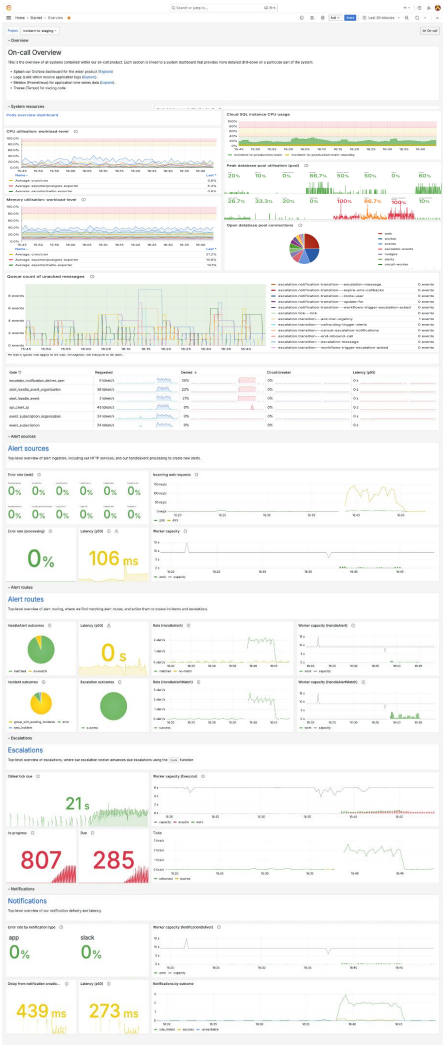
System Dashboard



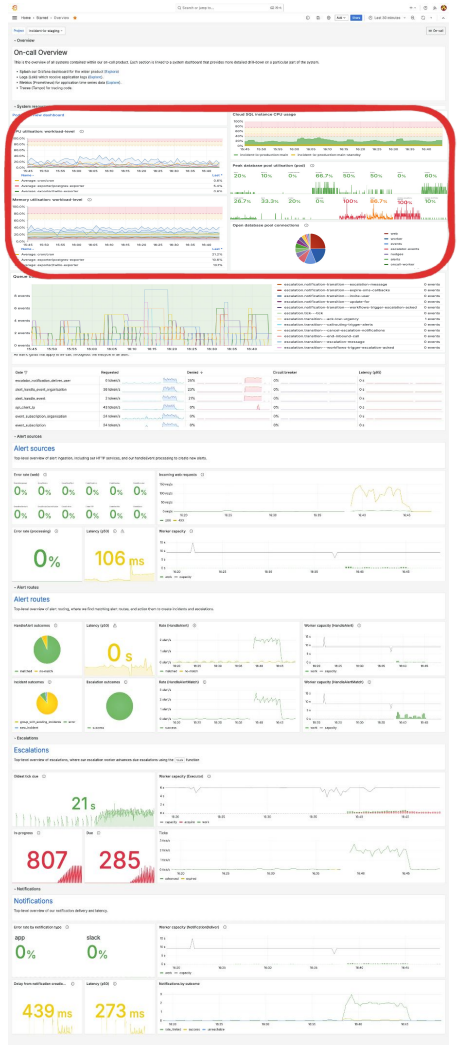
Logs



Traces



Overview dashboard



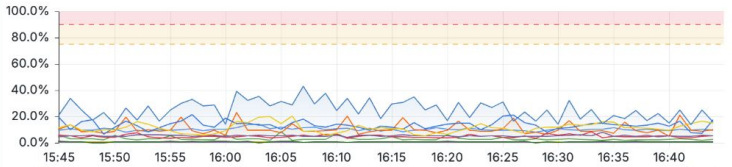
Infrastructure health

Overview dashboard

Infrastructure health

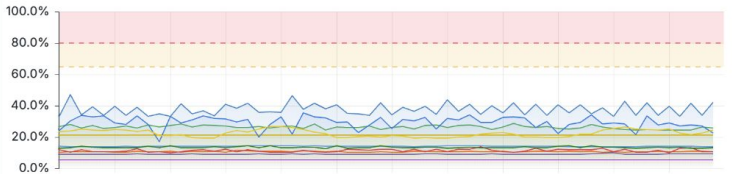
Pods overview dashboard

CPU utilisation: workload-level



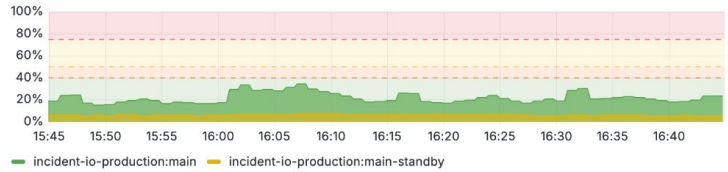
Name	Last *
Average: cron/cron	0.6%
Average: exporter/postgres-exporter	5.4%
Average: exporter/twilio-exporter	0.6%

Memory utilisation: workload-level

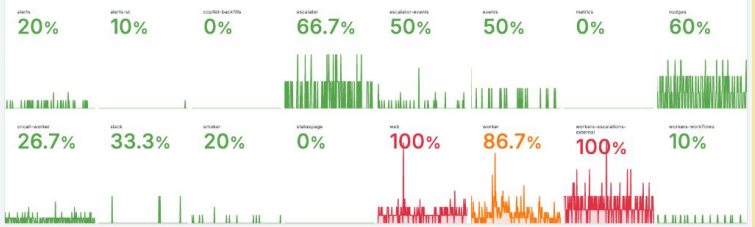


Name	Last *
Average: cron/cron	21.2%
Average: exporter/postgres-exporter	10.6%
Average: exporter/twilio-exporter	13.1%

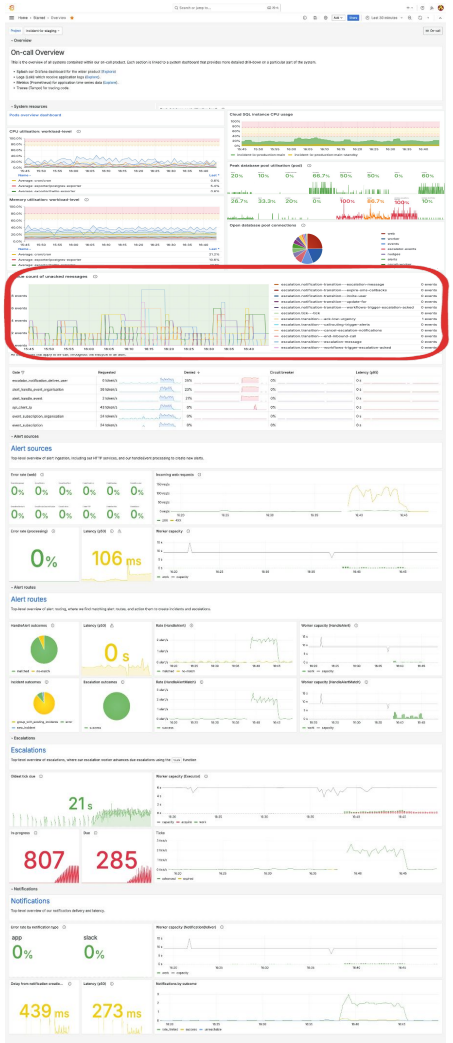
Cloud SQL instance CPU usage



Peak database pool utilisation (pod)



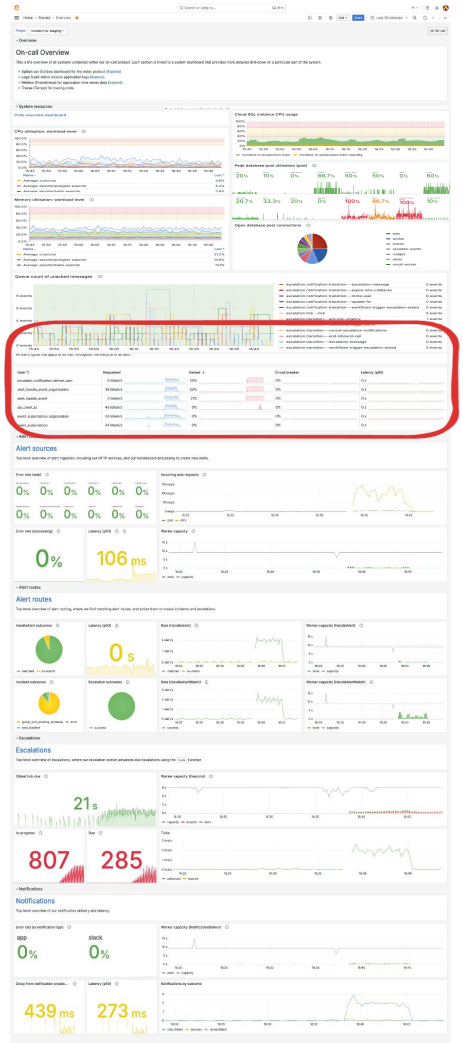
Open database pool connections



Infrastructure health

Queue health

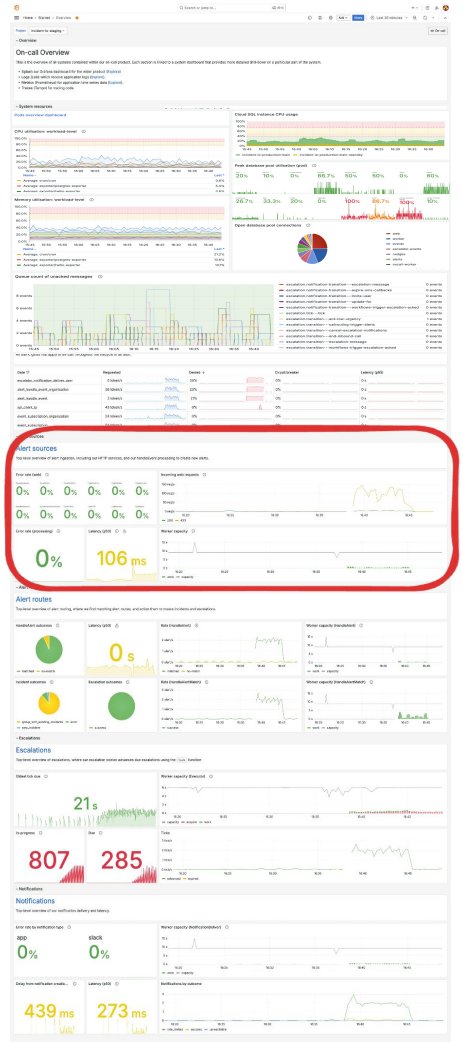
Overview dashboard



Infrastructure health

Queue health

Rate limits

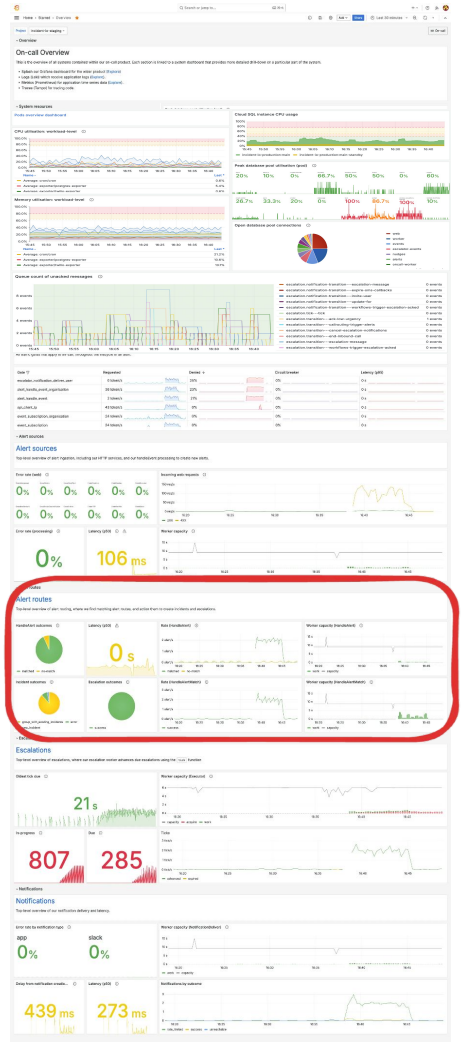


Infrastructure health

Queue health

Rate limits

Alert sources



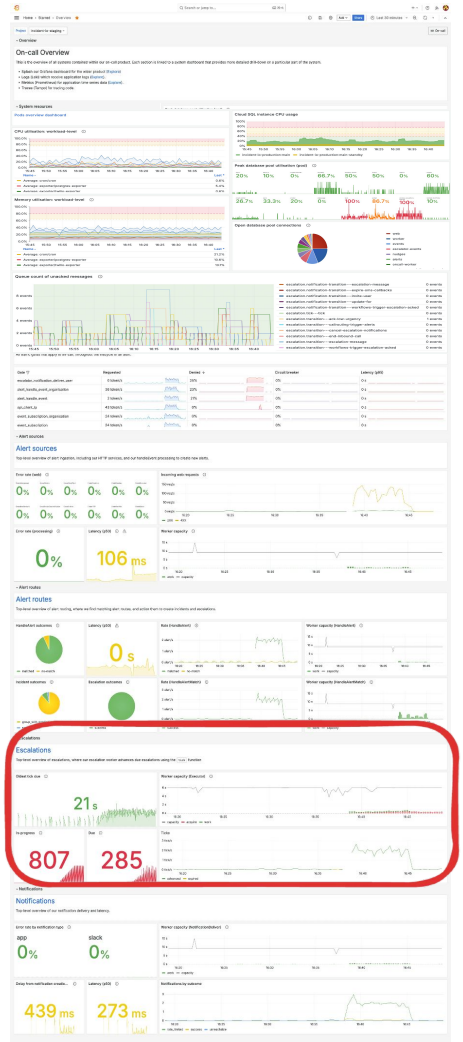
Infrastructure health

Queue health

Rate limits

Alert sources

Alert routes



Infrastructure health

Queue health

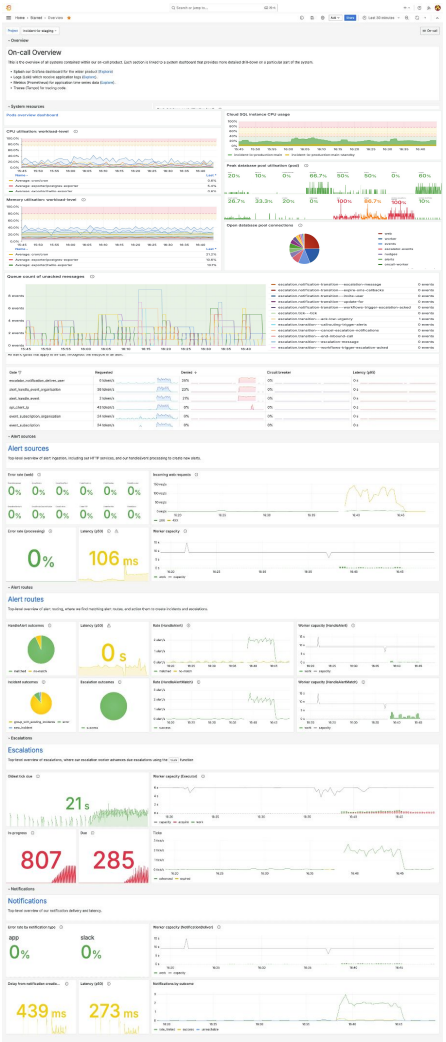
Rate limits

Alert sources

Alert routes

Escalations

Escalations



Infrastructure health

Queue health

Rate limits

Alert sources

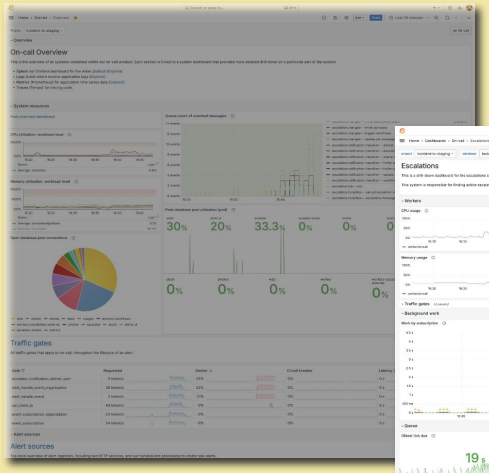
Alert routes

Escalations

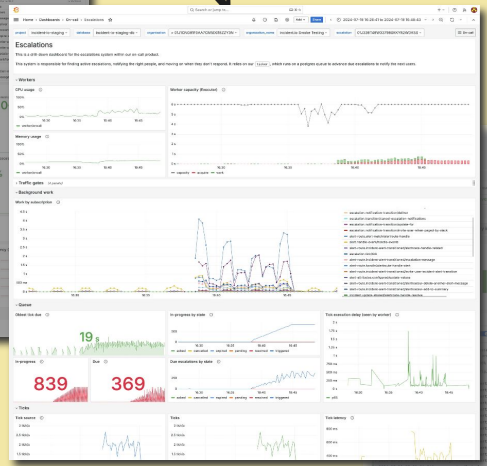
Notifications

Overview dashboard

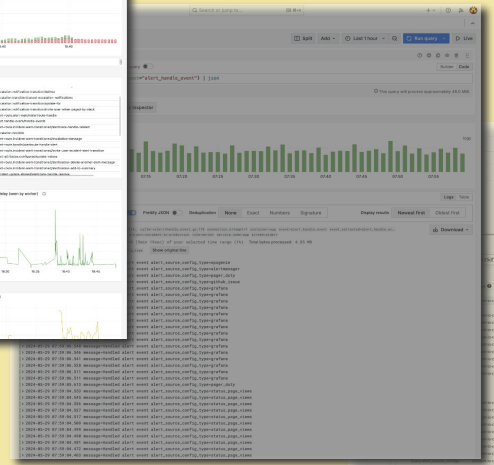
The Observability Lasagna



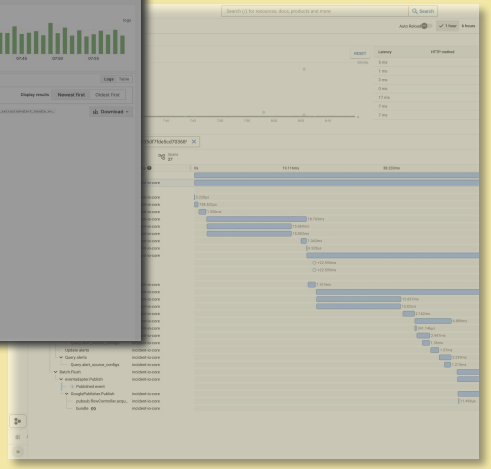
Overview Dashboard



System Dashboard

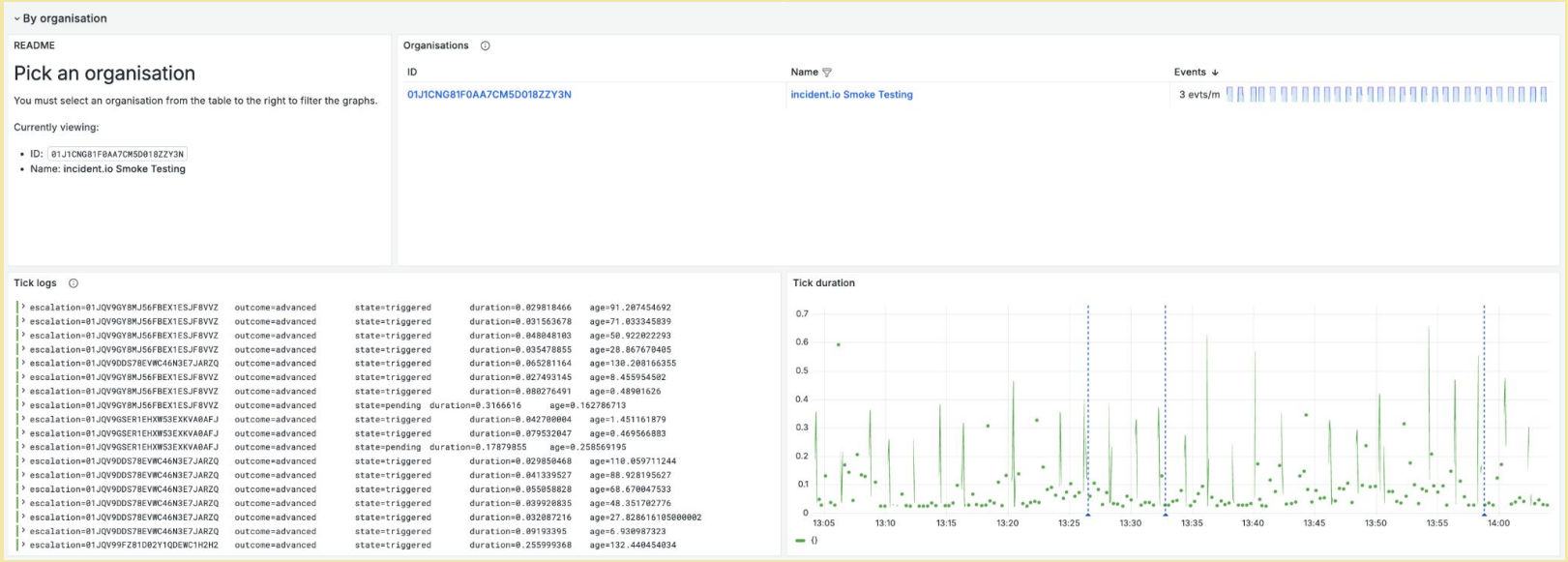


Logs

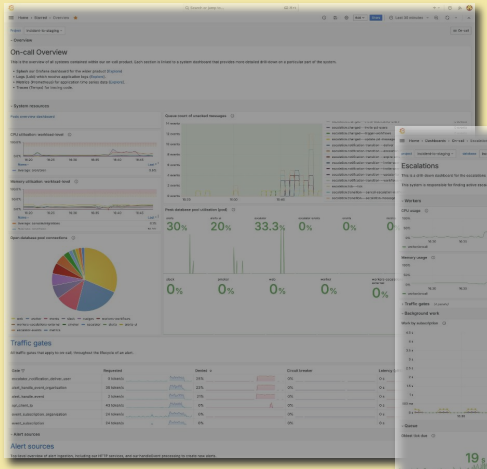


Traces

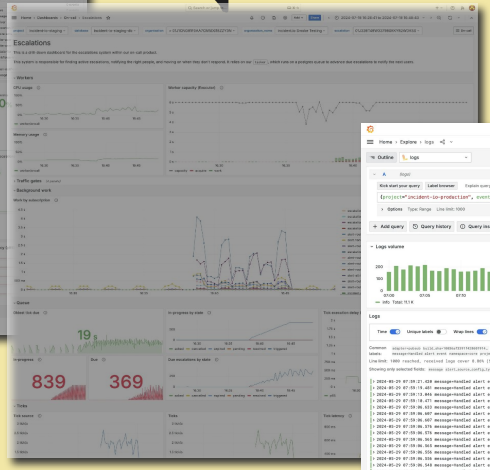
Logs


By organisation
README
Pick an organisation
You must select an organisation from the table to the right to filter the graphs.
Currently viewing:
ID: 01J1CNG81F0AA7CM5D018ZZY3N
Name: incident.io Smoke Testing
Organisations
ID
Name
Events
01J1CNG81F0AA7CM5D018ZZY3N
incident.io Smoke Testing
3 evts/m
Tick logs
escalation=01JQV9GY8MJ56FBEX1ESJF8VVZ outcome=advanced state=triggered duration=0.029818466 age=91.207454692
escalation=01JQV9GY8MJ56FBEX1ESJF8VVZ outcome=advanced state=triggered duration=0.031563678 age=71.033345839
escalation=01JQV9GY8MJ56FBEX1ESJF8VVZ outcome=advanced state=triggered duration=0.048048103 age=50.922022293
escalation=01JQV9GY8MJ56FBEX1ESJF8VVZ outcome=advanced state=triggered duration=0.035478855 age=28.867670405
escalation=01JQV9DDS78EVWC46N3E7JARZQ outcome=advanced state=triggered duration=0.065281164 age=130.208166355
escalation=01JQV9GY8MJ56FBEX1ESJF8VVZ outcome=advanced state=triggered duration=0.027493145 age=8.455954502
escalation=01JQV9GY8MJ56FBEX1ESJF8VVZ outcome=advanced state=triggered duration=0.080276491 age=0.48901626
escalation=01JQV9GY8MJ56FBEX1ESJF8VVZ outcome=advanced state=pending duration=0.3166616 age=0.162786713
escalation=01JQV9GSER1EHXW53EXKVA0AFJ outcome=advanced state=triggered duration=0.042700004 age=1.451161879
escalation=01JQV9GSER1EHXW53EXKVA0AFJ outcome=advanced state=triggered duration=0.079532047 age=0.469566883
escalation=01JQV9GSER1EHXW53EXKVA0AFJ outcome=advanced state=pending duration=0.17879855 age=0.258569195
escalation=01JQV9DDS78EVWC46N3E7JARZQ outcome=advanced state=triggered duration=0.029850468 age=110.059711244
escalation=01JQV9DDS78EVWC46N3E7JARZQ outcome=advanced state=triggered duration=0.041339527 age=88.928195627
escalation=01JQV9DDS78EVWC46N3E7JARZQ outcome=advanced state=triggered duration=0.055858828 age=68.670047533
escalation=01JQV9DDS78EVWC46N3E7JARZQ outcome=advanced state=triggered duration=0.039920835 age=48.351702776
escalation=01JQV9DDS78EVWC46N3E7JARZQ outcome=advanced state=triggered duration=0.032087216 age=27.828616105000002
escalation=01JQV9DDS78EVWC46N3E7JARZQ outcome=advanced state=triggered duration=0.09193395 age=6.930987323
escalation=01JQV99FZ81D02Y1QDEWC1H2HZ outcome=advanced state=triggered duration=0.255999368 age=132.440454034
Tick duration
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
13:05
13:10
13:15
13:20
13:25
13:30
13:35
13:40
13:45
13:50
13:55
14:00
{}

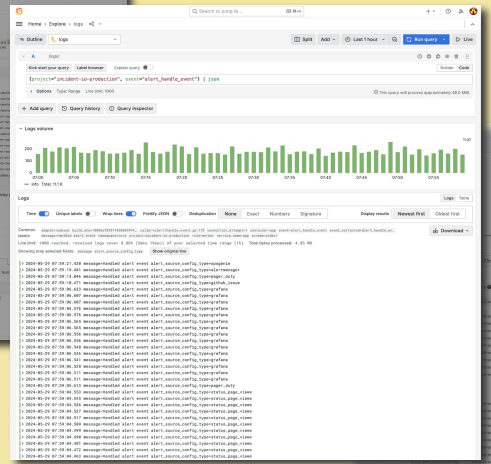
The Observability Lasagna



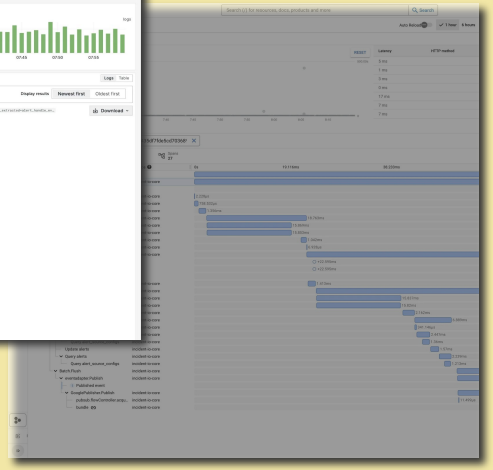
Overview Dashboard



System Dashboard

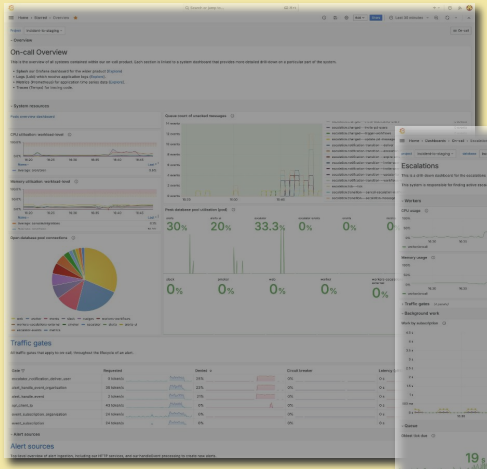


Logs

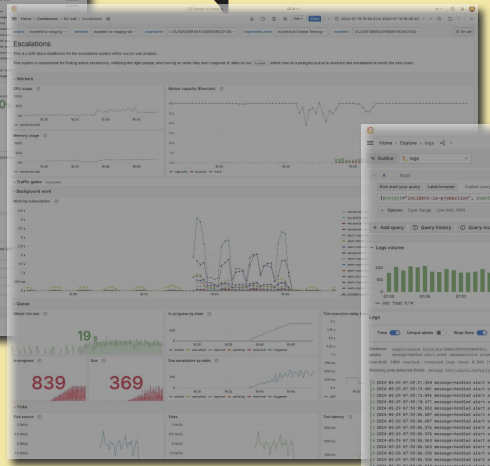


Traces

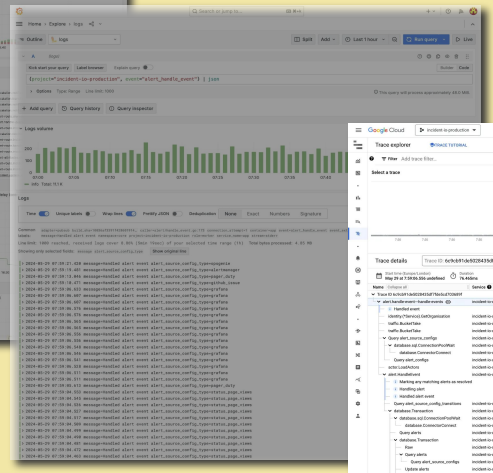
The Observability Lasagna



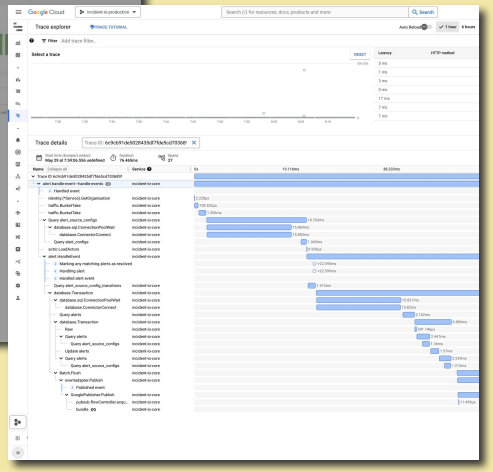
Overview Dashboard



System Dashboard

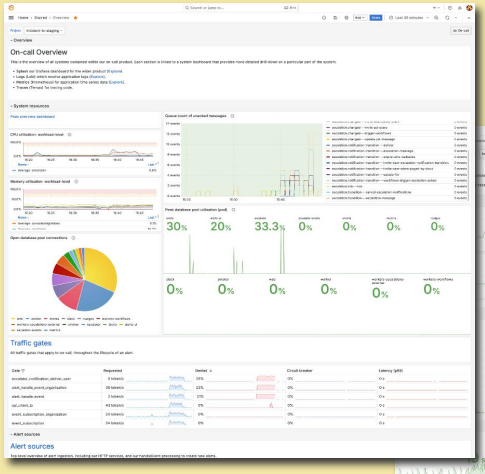


Logs

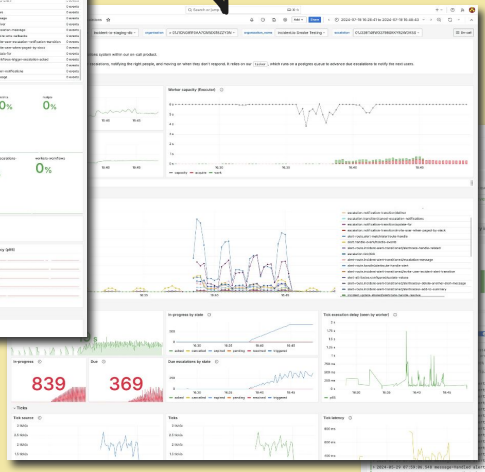


Traces

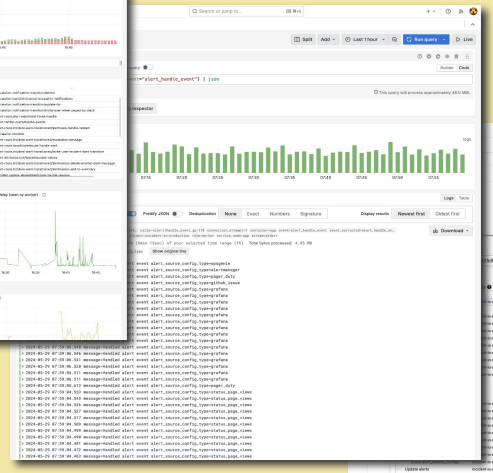
The Observability Lasagna



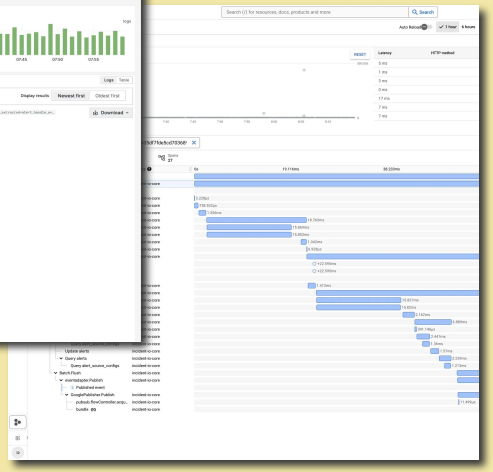
Overview Dashboard



System Dashboard



Logs



Traces

Practical tips

How do we actually implement this?

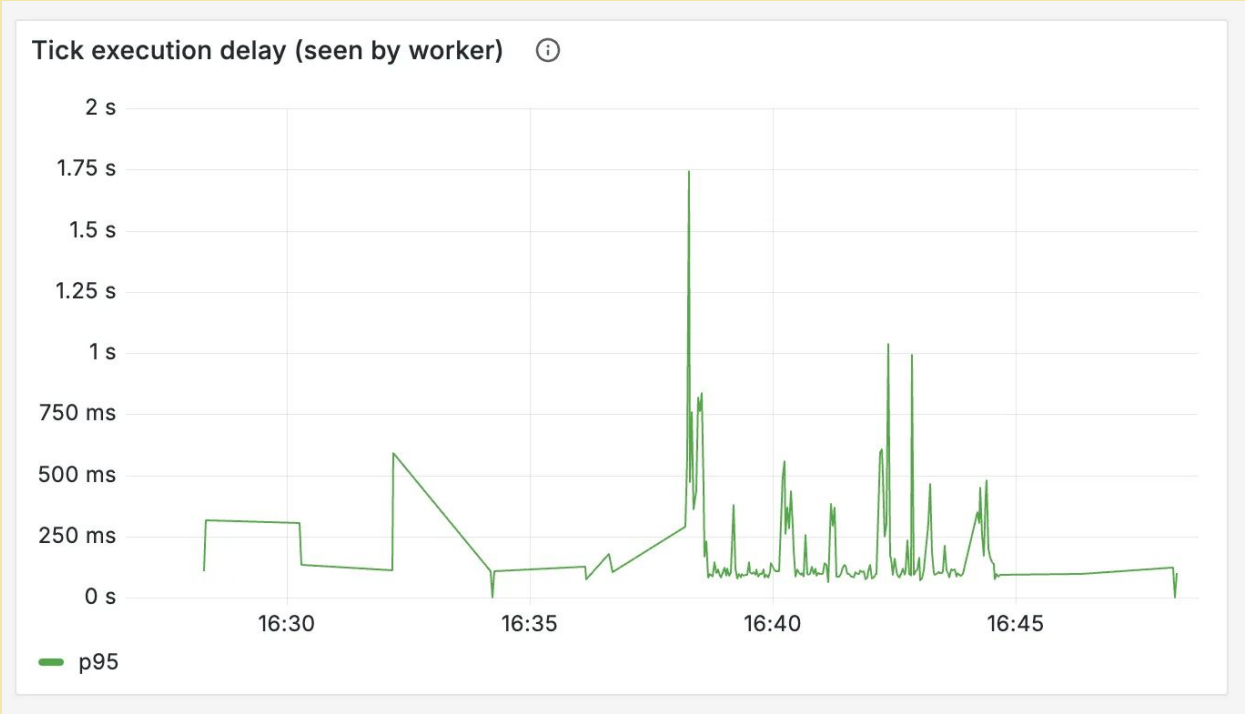
Make user impact your lens

Build observability that shows you what your end users are experiencing

Outcome field on metrics

```
type TickOutcome string

var (
    TickOutcomeArchived            TickOutcome = "archived"
    TickOutcomeExpired             TickOutcome = "expired"
    TickOutcomeGracePeriod         TickOutcome = "grace_period"
    TickOutcomeAdvanced            TickOutcome = "advanced"
    TickOutcomeAwaitScheduleCommit TickOutcome = "await_schedule_commit"
    TickOutcomeError               TickOutcome = "error"
)
```

Track user observed times



Connect metrics to logs

Always anchor metrics to a corresponding log with more detail

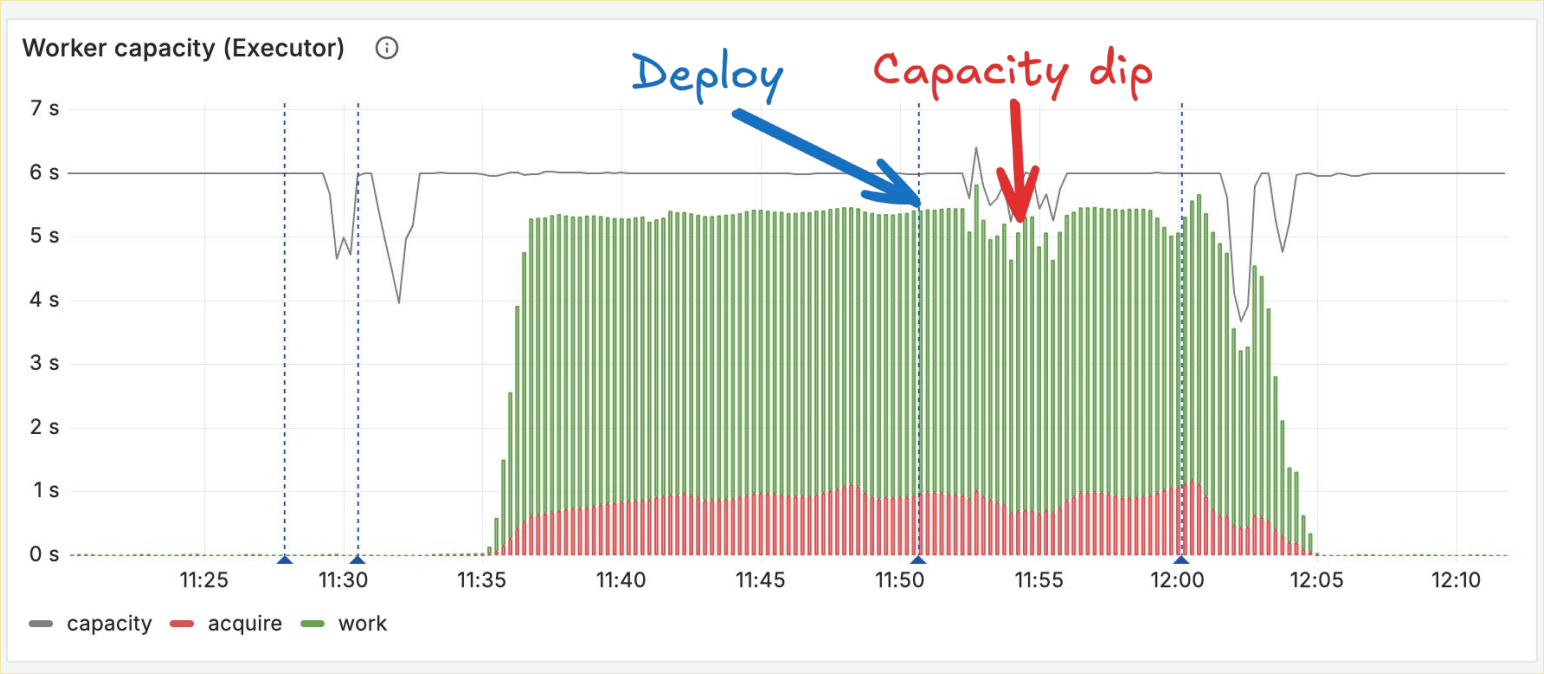
Event logs

```
log.Info(ctx, "Escalation ticked", o11y, map[string]any{
    // Event to search for quickly
    "event": "escalation_executor_tick",

    // Values tracked with our metrics
    "outcome":  outcome,
    "duration": time.Since(startAt).Seconds(),
    "escalation_initial_tick_delay_seconds": lo.Ternary(
        time.Since(escalation.TickDueAt).Seconds() > 0,
        time.Since(escalation.TickDueAt).Seconds(), 0,
    ),      You, 1 second ago • Uncommitted changes

    // High cardinality fields that we couldn't track with metrics
    "escalation":                    escalation.ID,
    "source":                        tickSource,
    "organisation_id":               escalation.OrganisationID,
    "escalation_idempotency_key":    escalation.IdempotencyKey,
    "escalation_start_at":           escalation.StartAt,
    "escalation_age_seconds":        time.Since(escalation.StartAt).Seconds(),
    "escalation_grace_period_seconds": escalation.GracePeriodSeconds,
    "escalation_initial_state":      escalation.CurrentTransition.State(),
    "escalation_initial_tick_due_at":  escalation.TickDueAt,
})
```

Visualise your limits

Know how much wiggle room you have

Capacity metrics



Practical tips

1. Make user impact your lens

2. Connect metrics to logs

3. Visualise your limits

Don’t do it alone

Build your observability stack with your team to get them bought in

Game days

- Quarterly drill of incident management scenario
- Closed book exercise
- See observability stack used in the wild

Observability Lasagna

Exercise your system

Connect your layers

Make user impact your lens

Don’t do it alone