40 Projects, One Team

The Keys to Responsiveness in Production

Seb Ferrer

OVHcloud

About me

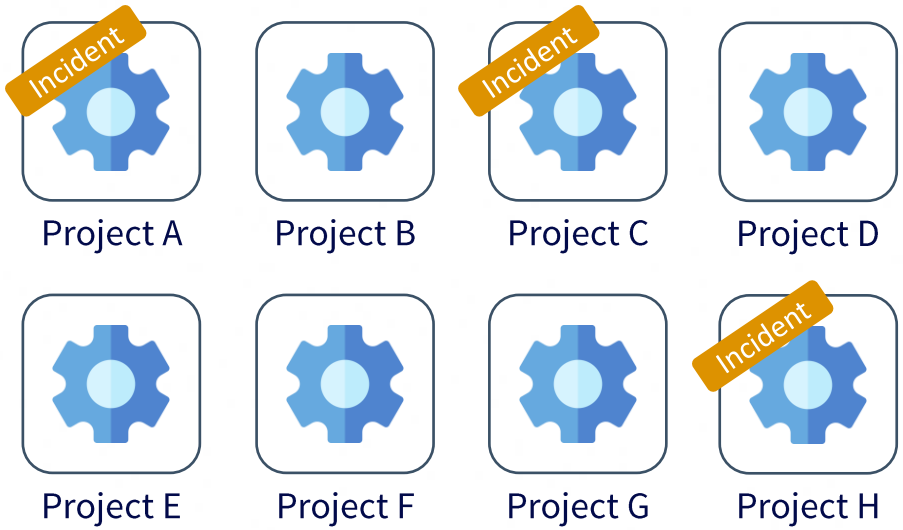
Lives in Nantes

Software engineer since 2012

- Web
- AI
- Proof-of-Concepts
- Cybersecurity / IAM
- Incident management

Me, before

Incidents


Incident
Project A
Project B
Incident
Project C
Project D
Project E
Project F
Project G
Incident
Project H

What to do?

Project A

→ Acknowledge

→ Understand the error → Context
Error message
Error type

→ Define the cause → Responsibility
Root cause analysis

→ Resolve → Contact the right team
Bug fix
Do an action

How?


TICKET
Incident ticket
Alert manager
Incident
Platform
Project A
Log system
Source code
Monitoring

Incident

Incident ticket

Affected Item

Project A

Assignment group

My Team

Assigned to

Seb Ferrer

Short description

```
[project-a] creating issue: ErrValidator: POST /rest/my-api: 403 ...
```

Description

...

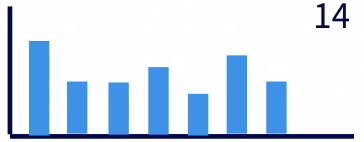
Notes

...

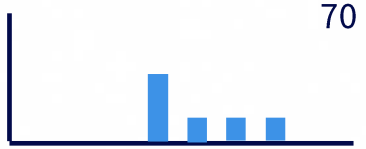
Acknowledged	02/02/2024 – 08:33:04
Created	02/02/2024 – 08:32:44

Alert manager

Last 24 hours



Last 30 days



Logs

Log / Monitoring

Log system

Search in the last **2** days

Project A

message: "creating issue" AND operation: ”createIssue"

timestamp	message
2024-02-02 08:34:12.548951 +02:00	Creating issue: ErrValidator: …
2024-02-02 08:34:01.548951 +02:00	creating issue: ErrValidator: …
2024-02-02 08:33:36.548951 +02:00	creating issue: ErrValidator: …
2024-02-02 08:33:24.548951 +02:00	creating issue: ErrValidator: …
2024-02-02 08:33:04.548951 +02:00	creating issue: ErrValidator: …
2024-02-02 08:32:44.548951 +02:00	creating ssue: ErrValidator: …

Monitoring

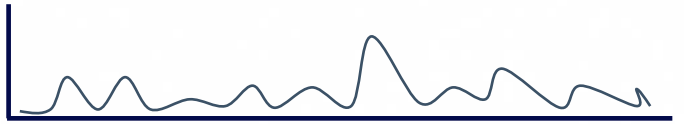
Last 24 hours

Restarts

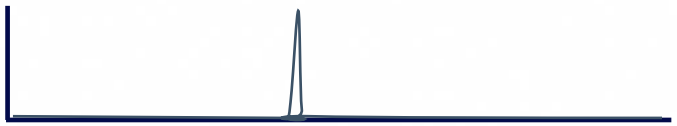
Uptime

28 days ago

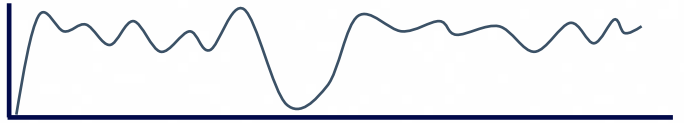
HTTP 4xx errors



HTTP 5xx errors



Nb issues created



Troubleshooting

history-based method

Fast
Someone has surely already met this error
Reducing learning curve

Limited to known issues
May be misleading
Less learning

Root-cause analysis (RCA)

Thorough
Helps identify the root cause
Better understanding of the project

Time-consuming
Risk of following a wrong path

Troubleshooting

The 5 whys

- ↑ Simple and easy to apply
- ↓ Can oversimplify complex issues

Fishbone diagram

- ↑ Visualizes multiple potential causes
- ↓ Can become complex with too many variables

Error elimination / Swapping

- ↑ Reduces possibilities step by step
  Helps isolate the problem through exclusion
- ↓ Can be slow, especially with many potential causes

Troubleshooting

Ask yourself the right question

What problem am I trying to solve?

Troubleshooting

Scope the issue

Who? What? When?

Reproduce

Gather information

Document

Troubleshooting – Case study

"operation X failed"

When ?

Acknowledgement

Watch alerts

Monitoring /
Assessing the extent
of the damage

Find logs

creating issue: operation X failed

creating issue: operation Y failed

creating issue: operation Z failed

Troubleshooting – Case study

Find logs

message	operation
terminal: creating issue: operation X failed	CreateIssue
terminal: creating issue: operation Y failed	CreateIssue
terminal: creating issue: operation Z failed	CreateIssue

Troubleshooting – Case study

Find logs

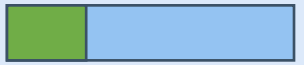
message	operation
creating issue: success	CreateIssue
terminal: creating issue: operation X failed	CreateIssue
creating issue: success	CreateIssue
creating issue: success	CreateIssue
terminal: creating issue: operation Y failed	CreateIssue
terminal: creating issue: operation Z failed	CreateIssue
consumer lag	

Troubleshooting – Case study

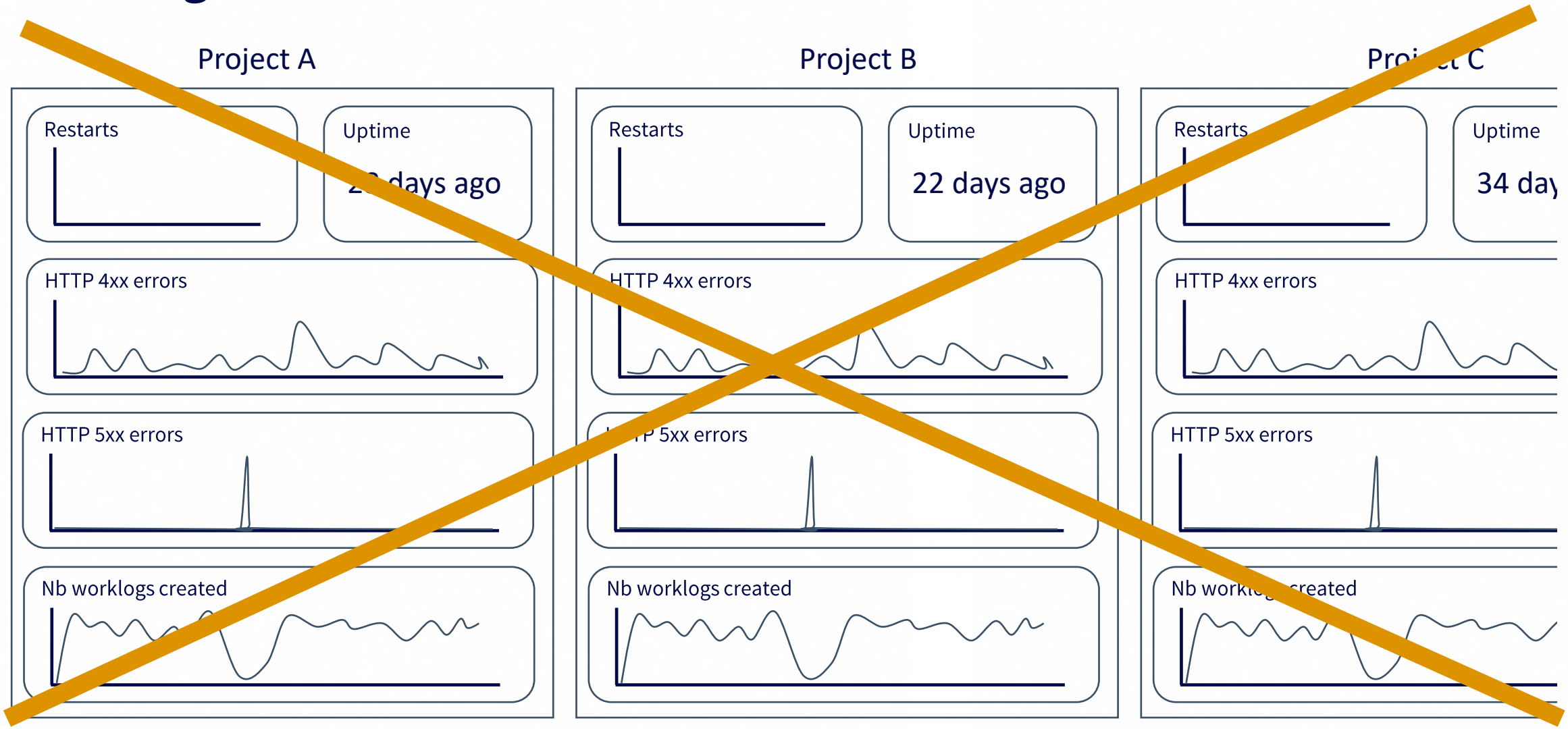
Find logs

message	pod_id
creating issue: success	app-54fb9-n4vkg
terminal: creating issue: operation X failed	app-54fb9-n4vkg
creating issue: success	app-54fb9-n4vkg
creating issue: success	app-54fb9-n4vkg
terminal: creating issue: operation Y failed	app-54fb9-n4vkg
terminal: creating issue: operation Z failed	app-54fb9-n4vkg
consumer lag	app-54fb9-n4vkg

RUN

Affected Item	Log	Date	Priority	Time to resolve
Project A	[project-a] 4xx error	2024-02-02 05:38:01	P5	
Project B	[…] 4xx error	2024-02-02 05:39:04	P4	
Project C	[…] 4xx error	2024-02-02 05:39:06	P5	
Project A	[…] 5xx error	2024-02-02 06:40:12	P5	
Project M	[…] 5xx error	2024-02-02 10:36:02	P5	
Project B	[…] 4xx error	2024-02-02 10:42:08	P3	
Project L	[…] 4xx error	2024-02-02 11:38:06	P5	
Project W	[…] 4xx error	2024-02-02 11:40:45	P5	
Project S	[…] 5xx error	2024-02-02 12:08:29	**P2**	

Morning check



Morning check

Project A

5xx errors	4xx errors
0	2

Success	Failures
58	2

Project B

5xx errors	4xx errors
1	0

Success	Failures
12	1

Project C

5xx errors	4xx errors
0	12

Success	Failures
85	12

Project D

5xx errors	4xx errors
0	0

Success	Failures
24	0

Project E

5xx errors	4xx errors
0	0

Success	Failures
42	0

Project F

5xx errors	4xx errors
15	1

Success	Failures
67	16


Develop your applications as if
you were going to maintain them.
Because you are.
~ a cool colleague

Logs

To avoid

Lack of data

Noise

Log level

INFO | WARN | ERROR | DEBUG | FATAL | TRACE

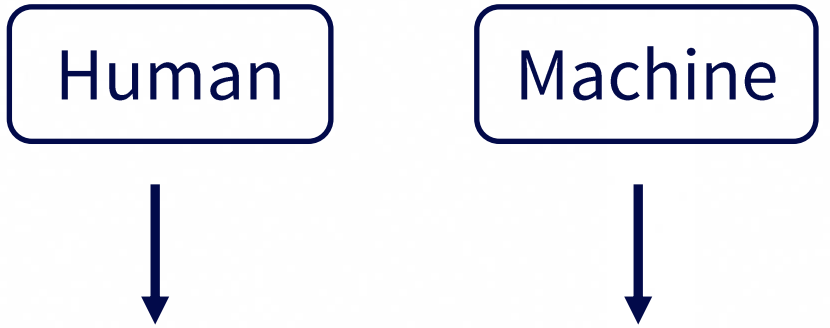
Write relevant logs

Request ID | URL Path | HTTP Method | HTTP Error Code | Body

User IDs | Stack trace | App Version | Operation

Logs

Define log format



Structure your logs

[timestamp] ERROR: error wrapping message: error message

[timestamp] INFO: upload start

Errors handling

Always check errors

seems evident, but it’s the most important

Use descriptive messages

provide explicit error info

Wrap errors

preserve context and the underlying error

Use sentinel errors

define errors for common cases (e.g., ErrNotFound)

Choose libraries wisely

prefer those with strong error handling

Monitoring / dashboards

Why?

- Performance tracking
- Issue detection
- Quick insights
- Reliability Assurance
- Productivity
- User experience
- Reactivity

What?

- **Application**
  - Error rates
- **Performance**
  - Database, request rates, response time, network latency, …
- **Availability**
  - Health check
- **Database**
  - DB queries performance
- **Resources**
  - CPU, RAM, bandwidth, …

…and everything that can lead to an alert

Monitoring / dashboards

Define the right color

Everything’s OK

Be careful

It’s broken

The rest in white

Communicate information quickly

Clear information

Chose the right type of charts

Organize the information

Make it customizable

Code architecture

Use an archetype

- Unified codebase
- Unified starter
- Microservice gateway communications
- Packages / dependencies
- Artifact repository / manager
- CI/CD setup

Use common packages

- logger
- alerter
- profiling
- http requests
- Error handling
- Health-check
- cmd
- metrics
- crypto
- reflect

Night troubleshooting / Runbooks

Useful links

Documentations
Request tracker
Metric dashboards
Logs
Etc…

Define the scopes

Concerned projects
Type of errors
Contact information

Explain each procedure

Anticipate any potential scenario
How to start, stop, restart the system

Keep it up to date!

Sources

Logs	https://betterstack.com/community/guides/logging/logging-best-practices/ https://www.dataset.com/blog/the-10-commandments-of-logging/
Error handling	https://www.digitalocean.com/community/tutorials/handling-errors-in-go https://www.jetbrains.com/guide/go/tutorials/handle_errors_in_go/best_practices/
Troubleshooting	https://medium.com/@dzanna.molly/devops-troubleshooting-strategy-2b5b38a5f3b7 https://www.youtube.com/watch?v=6pRBWM-J-c8
Monitoring	https://www.ibm.com/think/topics/application-monitoring-best-practices https://www.keyprod.com/en/knowledge/production-monitoring
Runbook	https://www.atlassian.com/software/confluence/templates/devops-runbook

Special thanks to the
OVHcloud SI&A team

Drawings by goupil.poil (instagram)

Icons from Flaticon

Freepik
Smashicons
WinWwin Winaldi
Good Ware
Vectors Market
Prosymbols Premium
dmitri13
Amethyst prime

Get the slides here

Find me online :

sebferrer

seb-ferrer

https://blog.kimi.ovh

OVHcloud