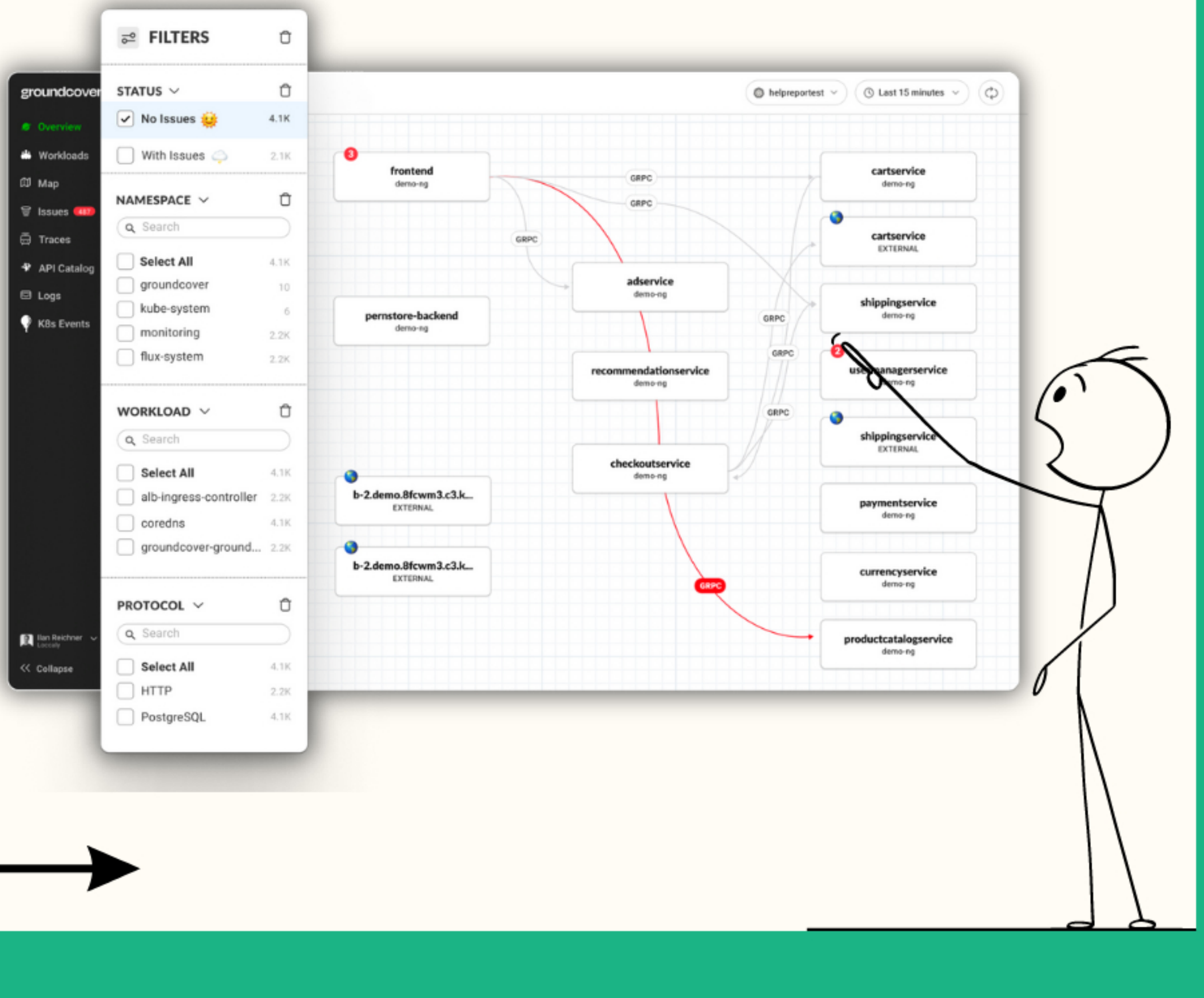
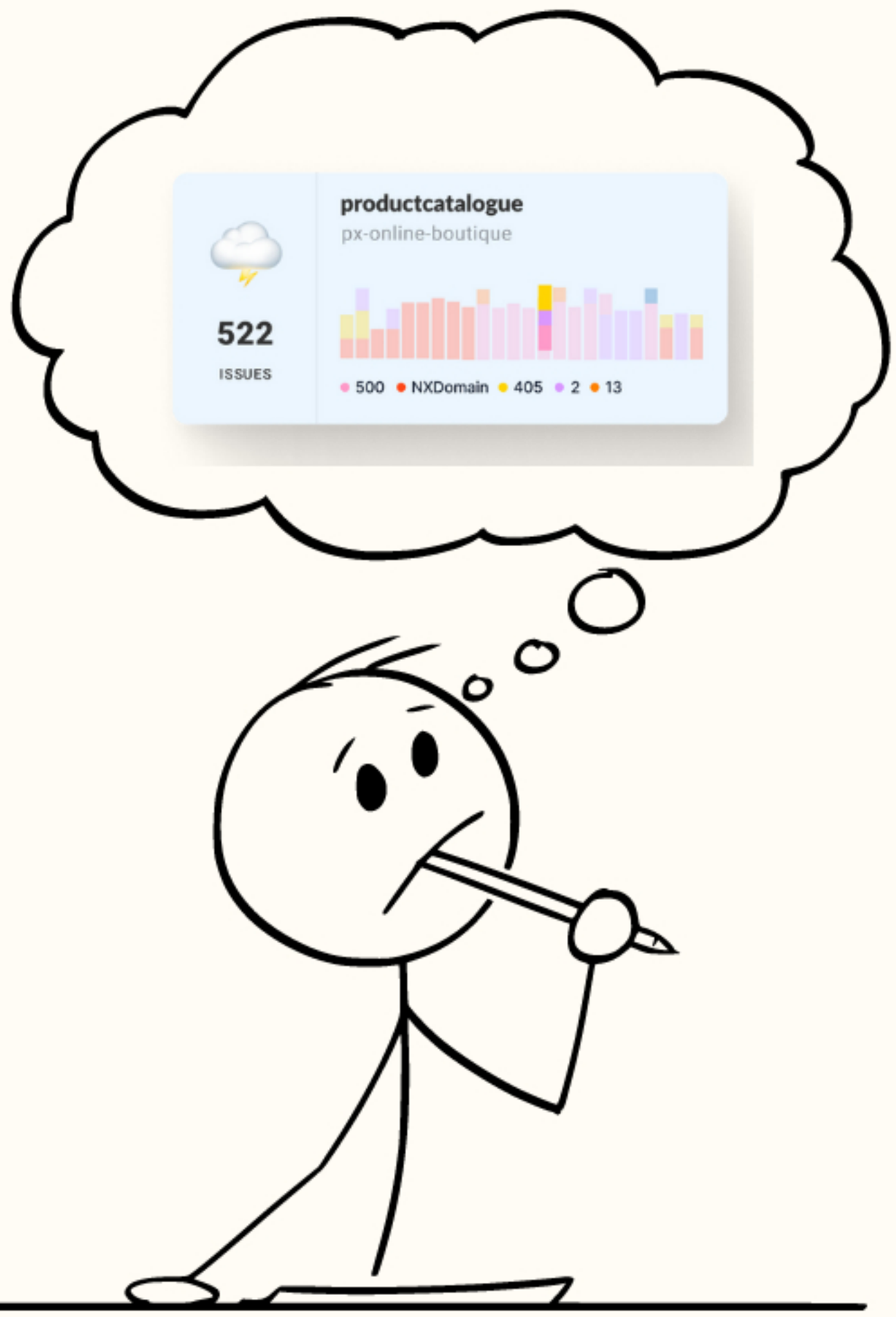
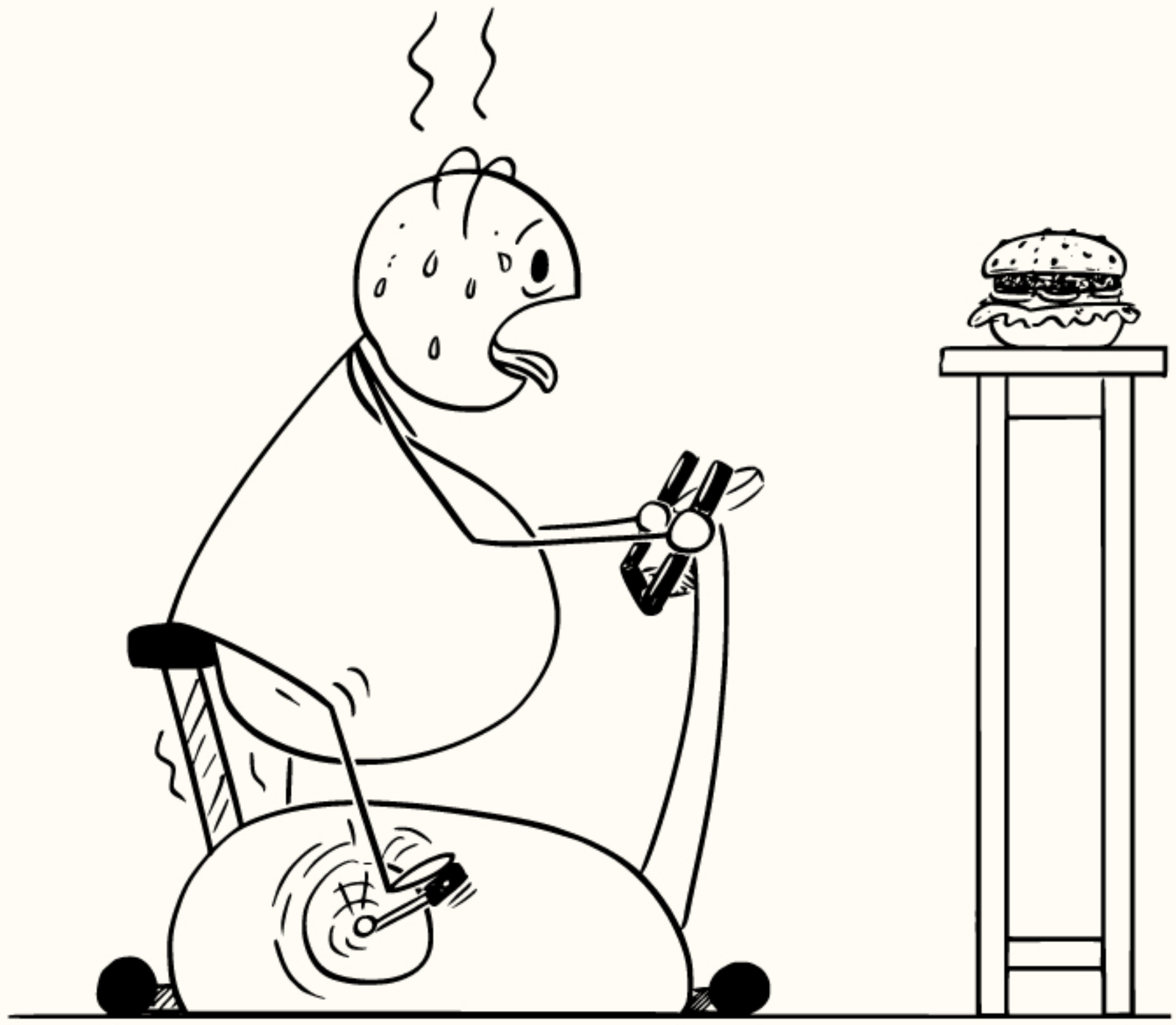
What to Know about Microservices Observability



The main reason why observing **microservices** can be hard...



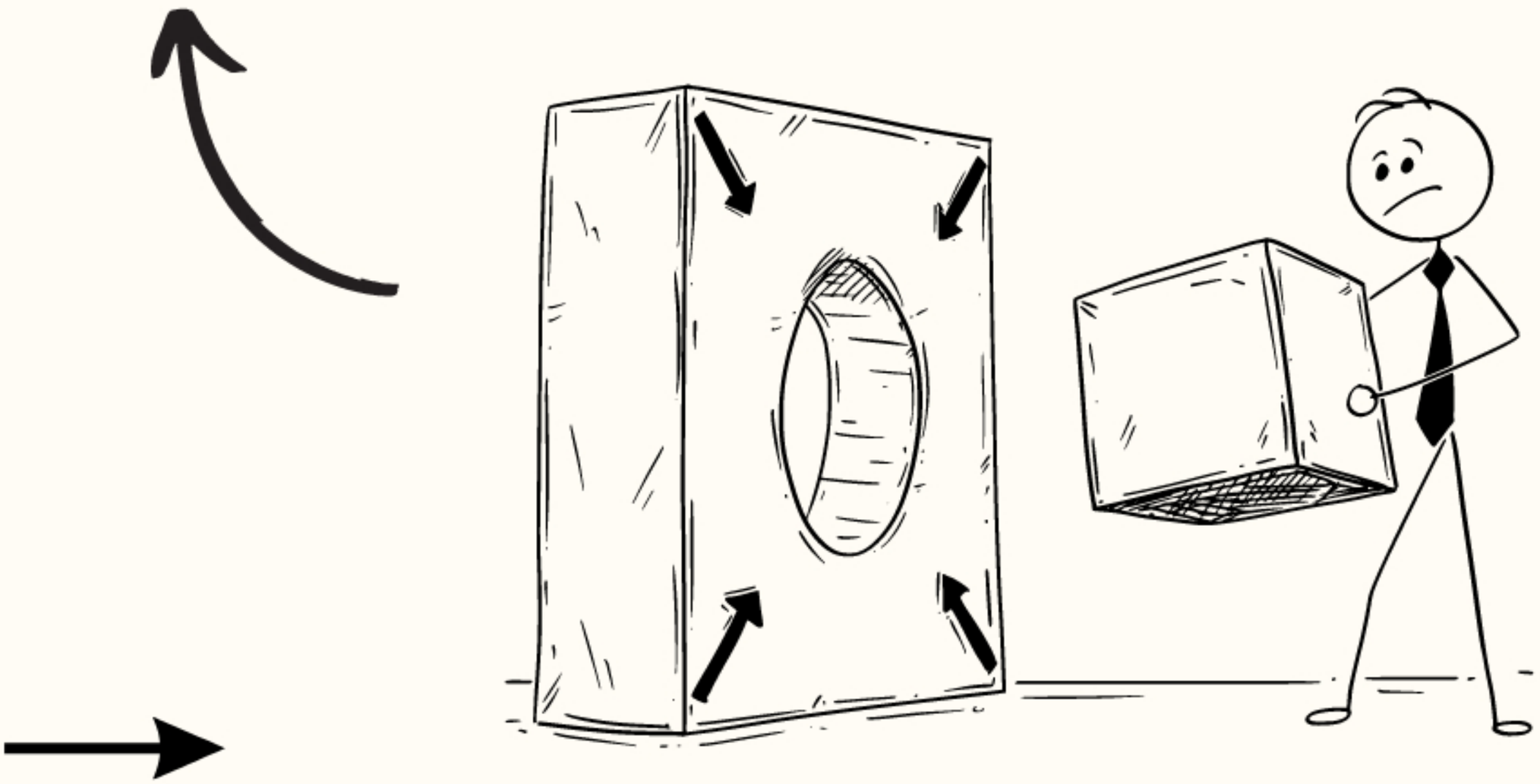
Is that this **“fatter technology stack”** demands much more attention and a more complex integration for R&D teams.



Moreover, it introduces the following issues:

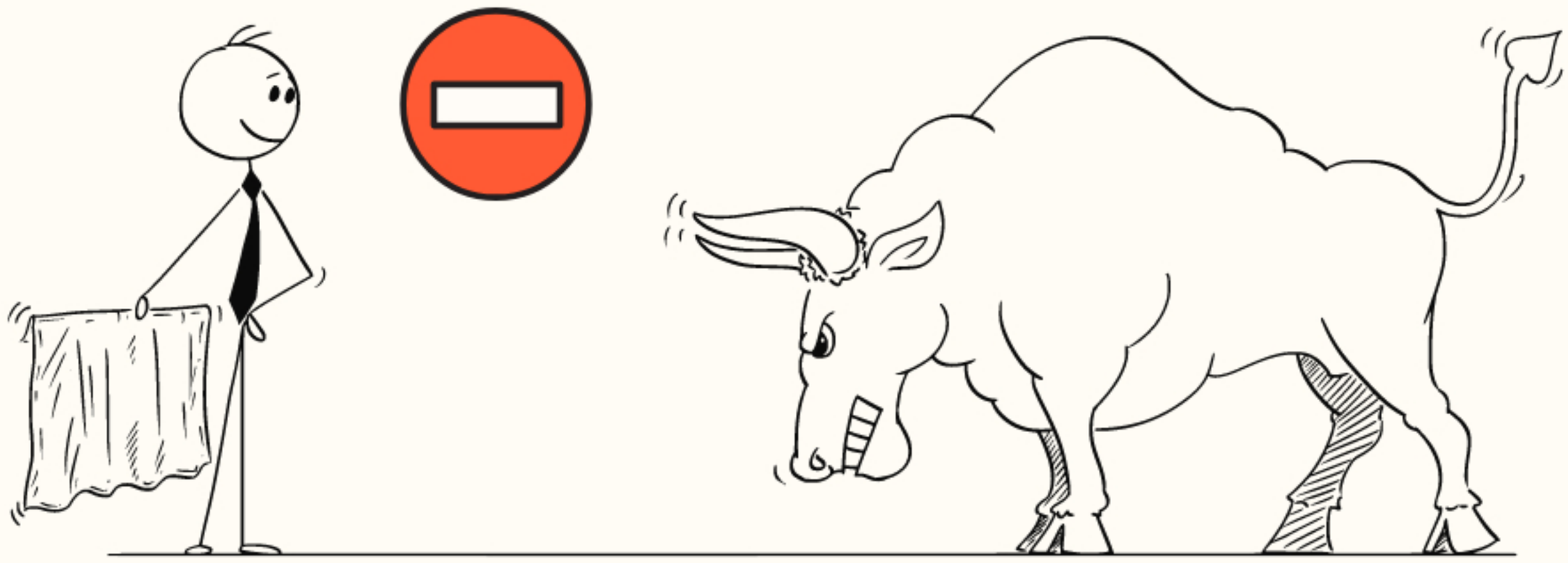
1) Root causes are not always obvious within distributed microservices environments.

2) Each microservice typically generates its own logs and metrics, so there is more data to collect and analyze in order to observe application state.



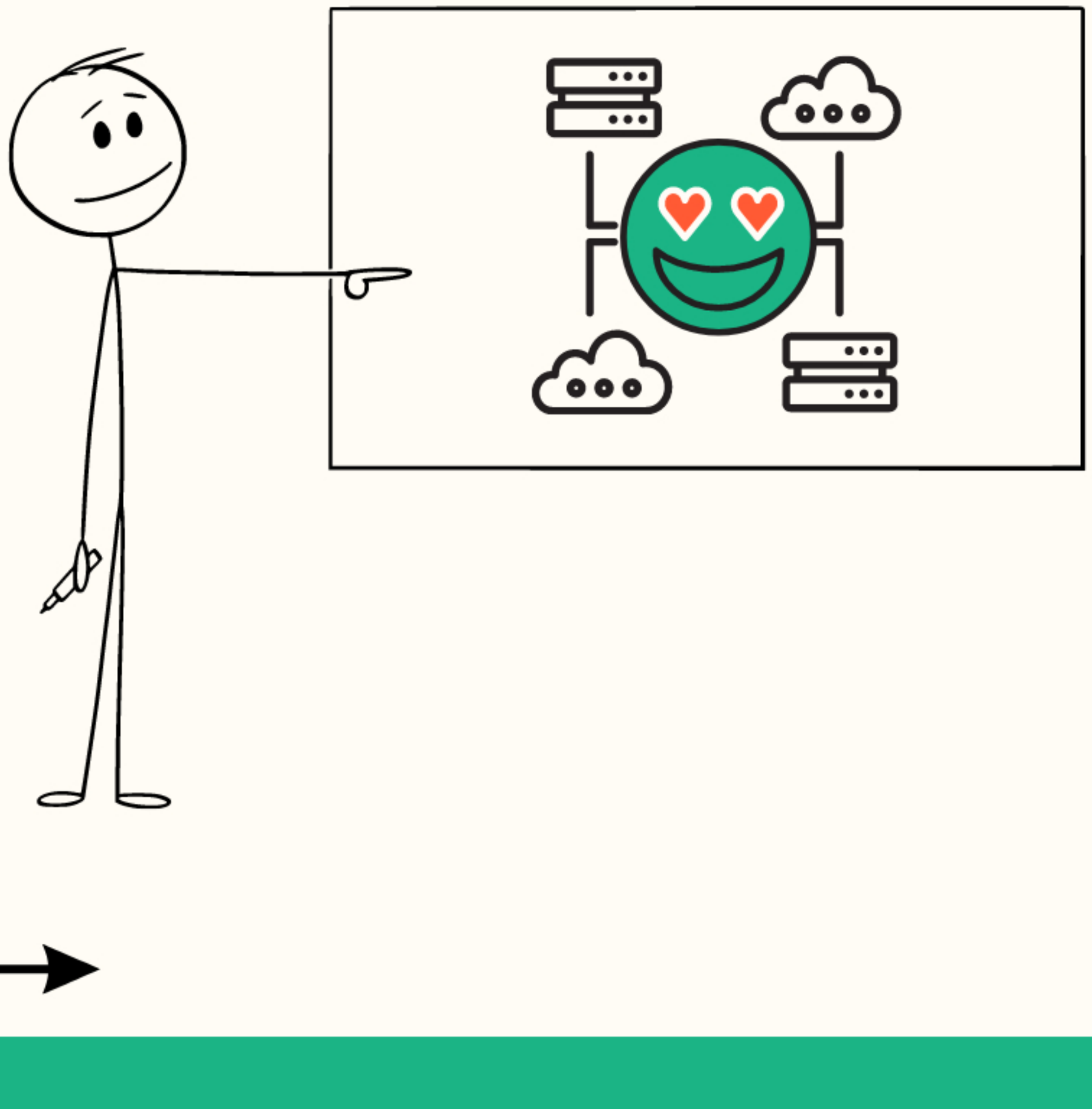
That said, one way to simplify microservices observability is to take advantage of the fact that microservices architectures are API-driven by observing API performance in addition to relying on metrics, logs and traces from your microservices themselves.

Because API transactions involve exchanges between microservices, tracing API calls is a great way not just to identify performance issues...

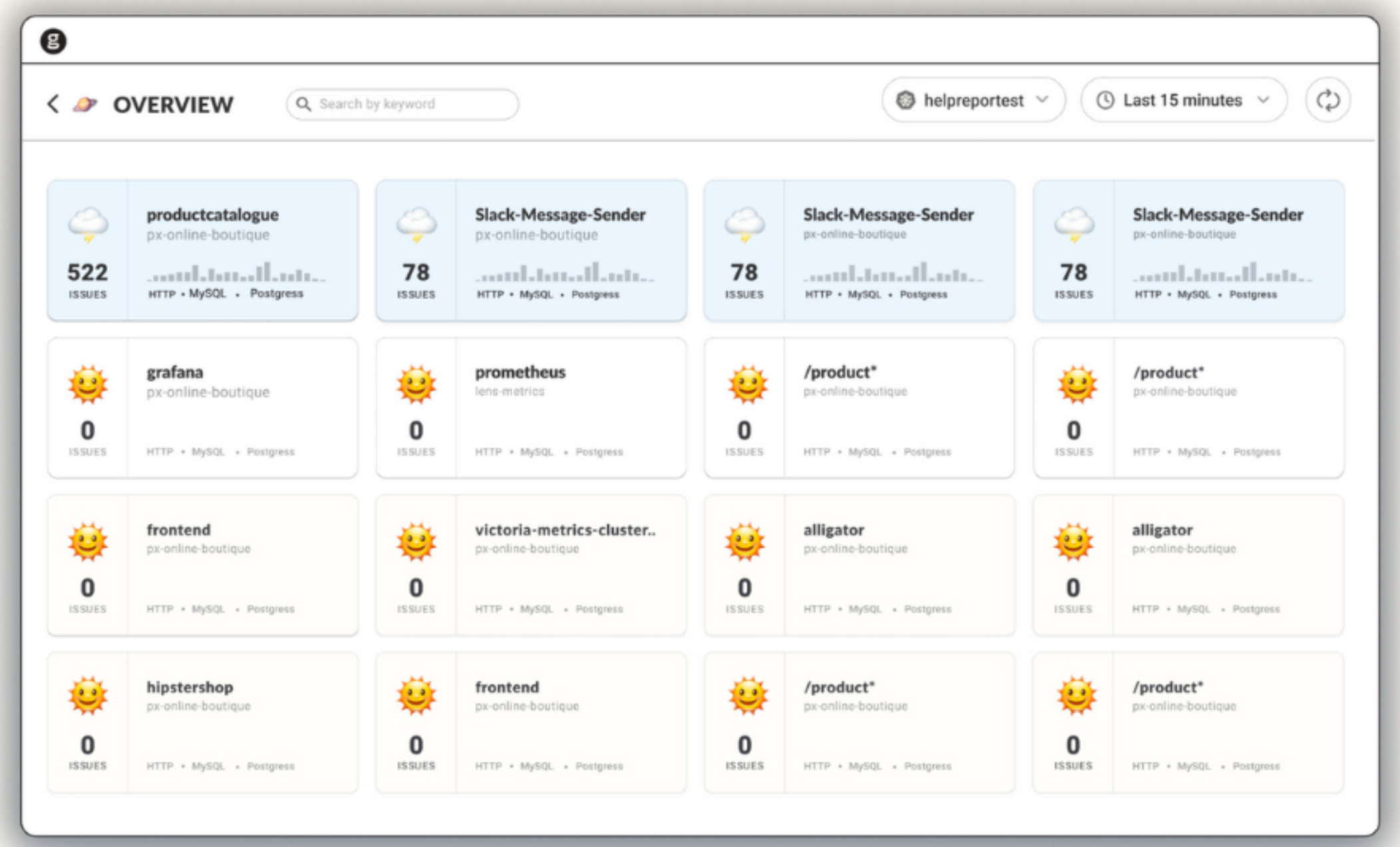


but also to determine which microservices did what during a problematic transaction – and which ones are therefore likely to be the root cause of problems.

You can collect most of the data you need to observe your app from API transactions, rather than requiring each microservice to expose metrics and logs directly.



*Thanks to tools like **groundcover** ,there are virtually no limits on the amount of information we can collect about microservices' state and performance.*



groundcover

groundcover opens up new ways to observe an API-centric architecture which means that you can observe any app – whether it's a monolith or a set of microservices – with zero instrumentation and without compromising on the depth or granularity of data.