Architecture Decision Records

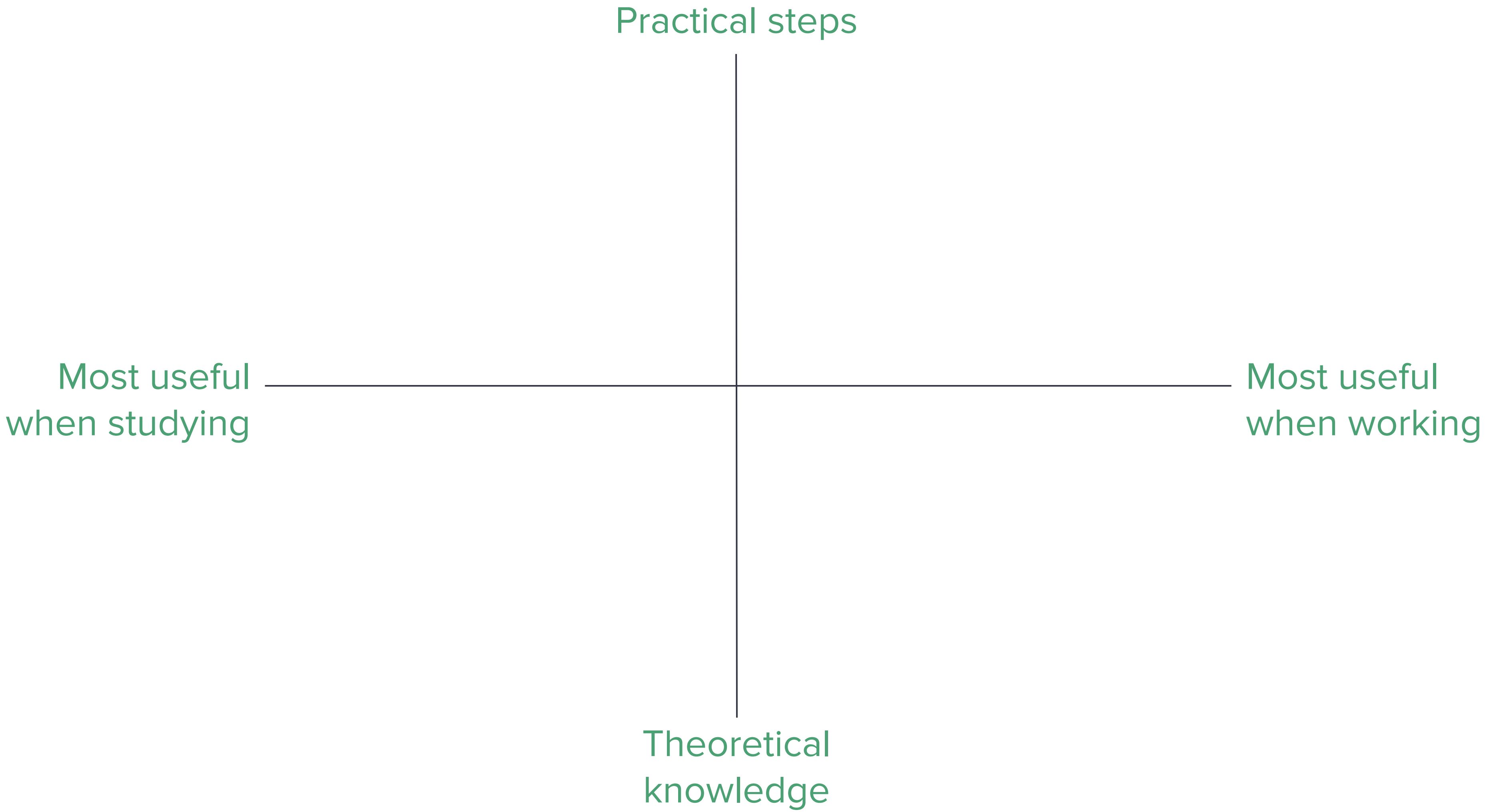
Arnaud Bos

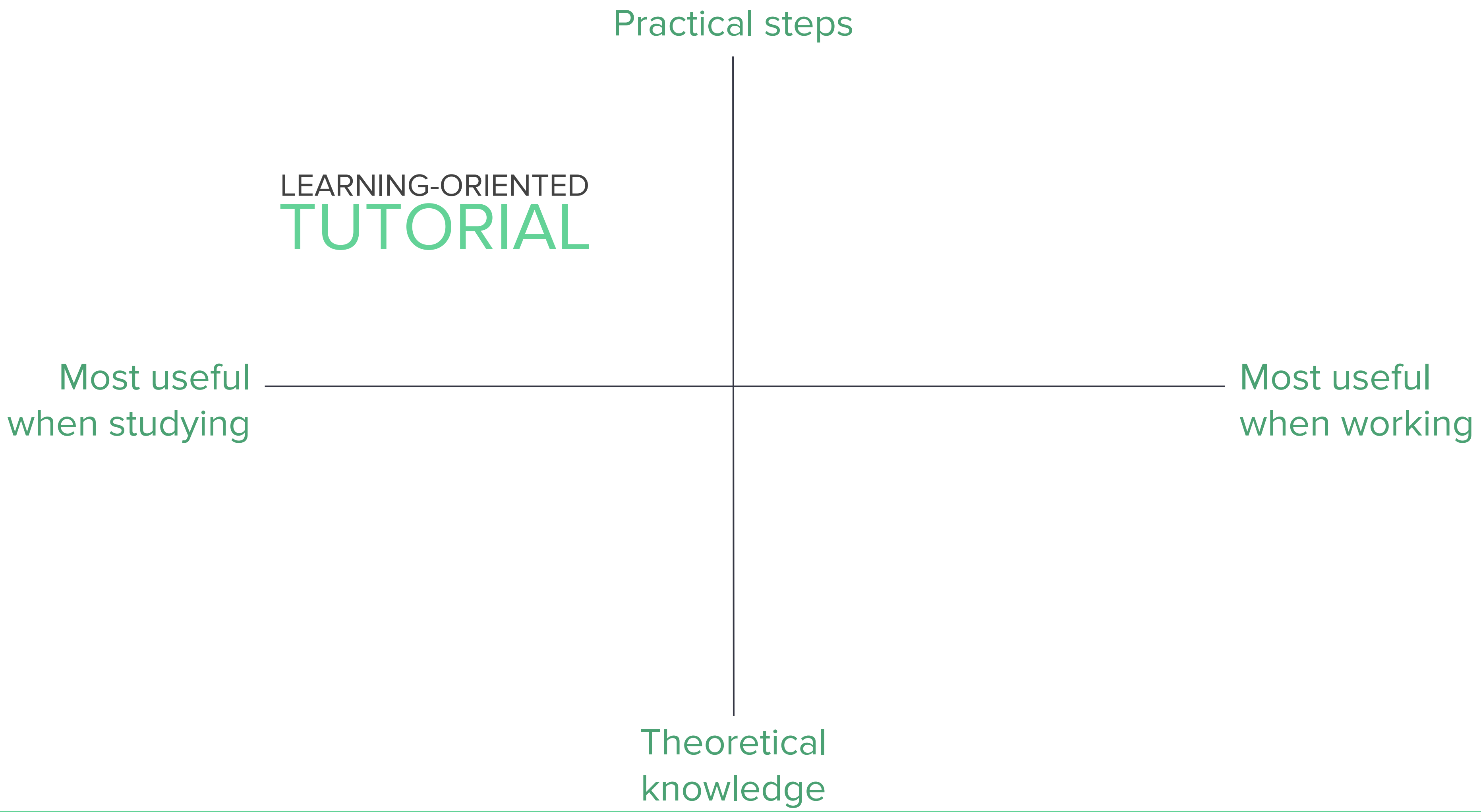
Toulouse DevFest 2018

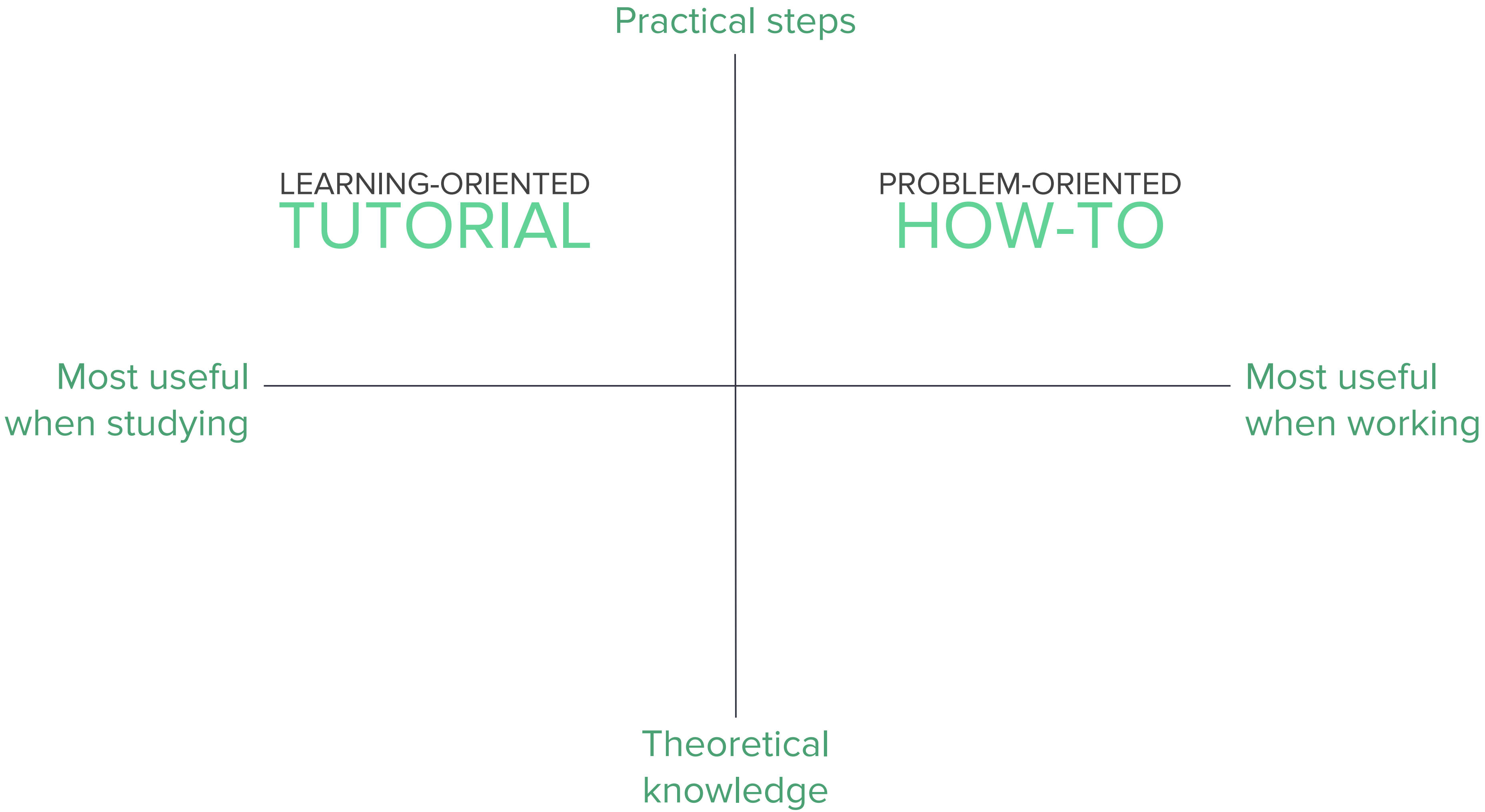
DIVIO BLOG

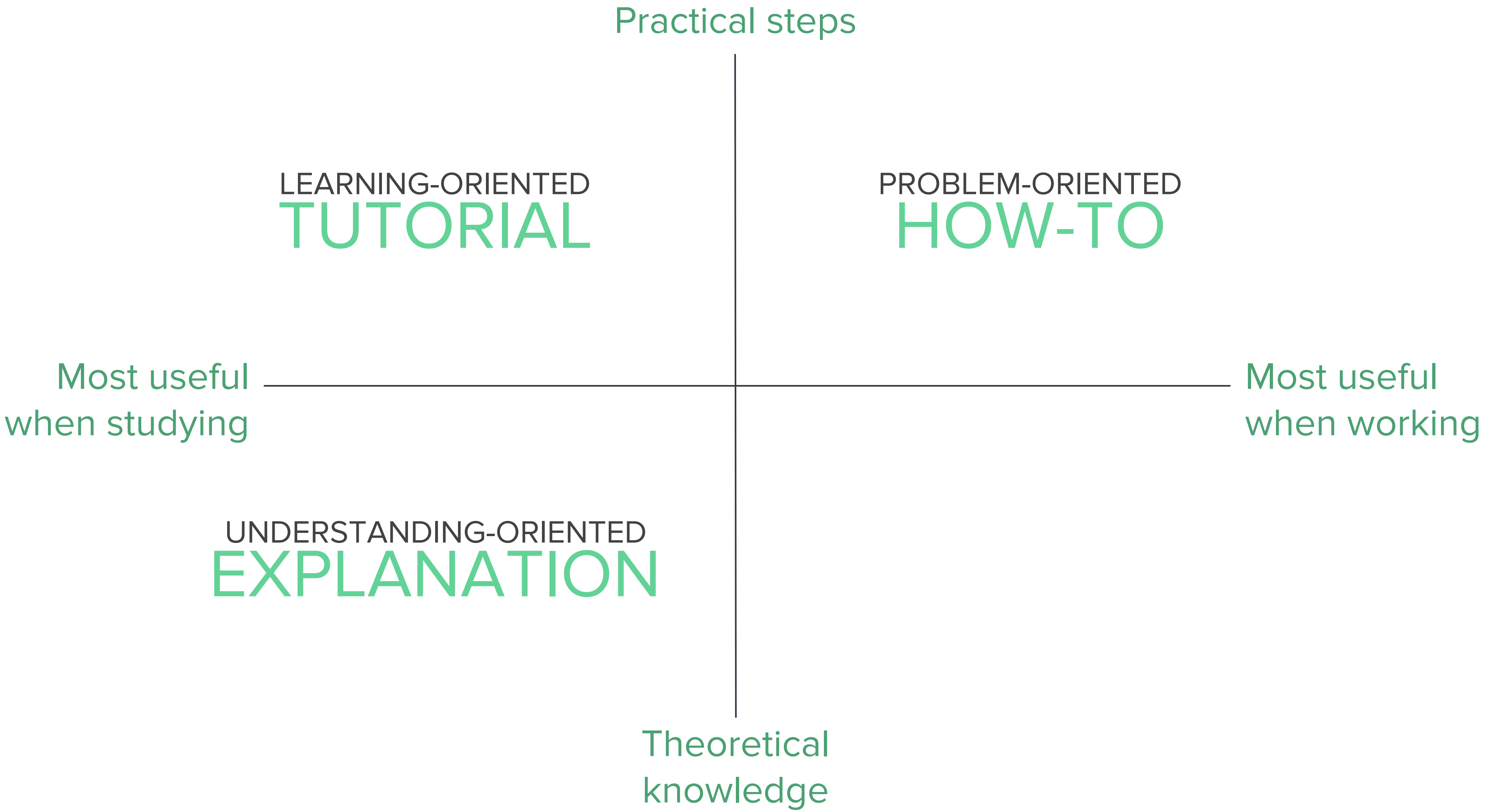
What nobody tells you about documentation

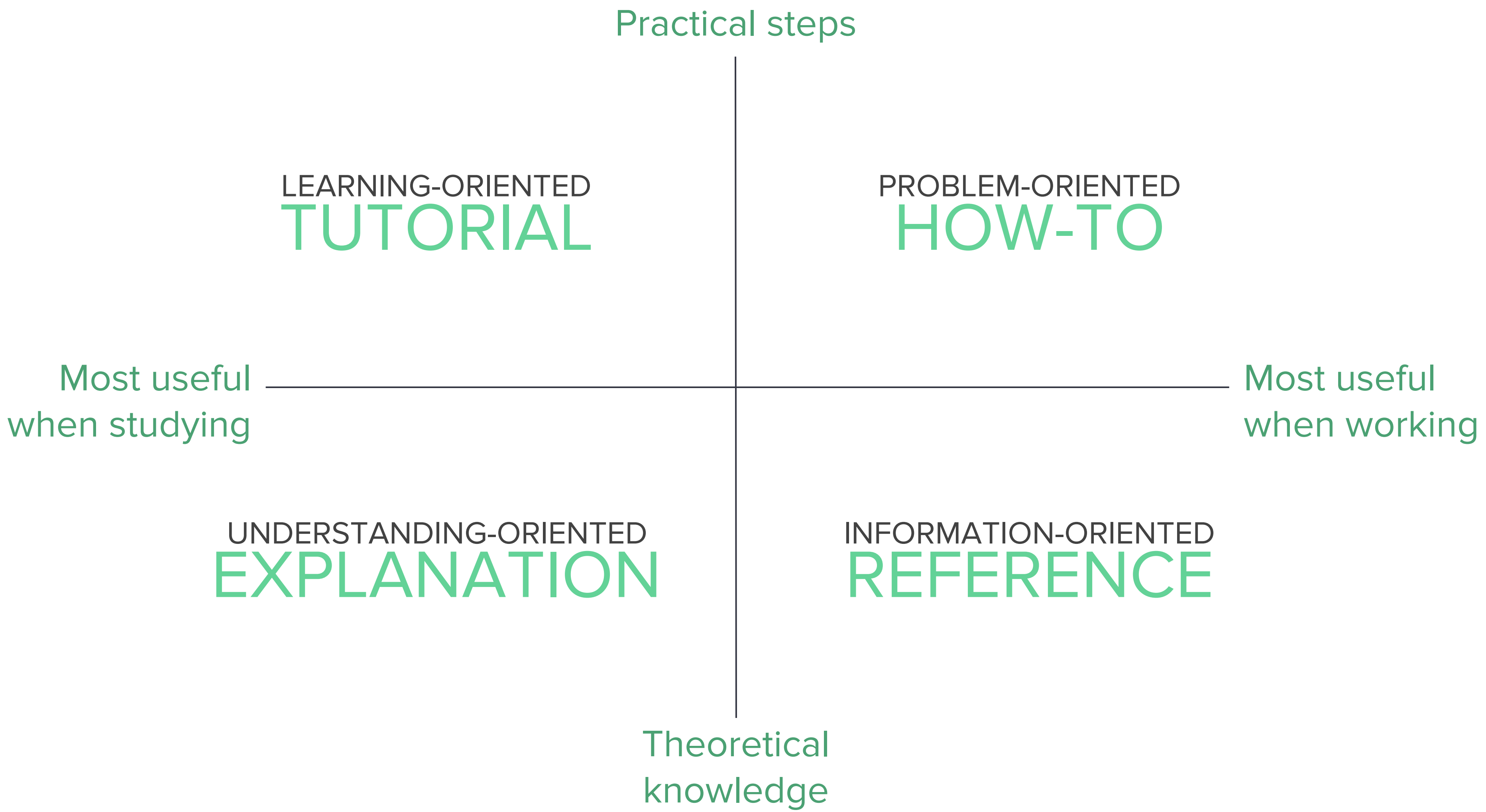
CODE

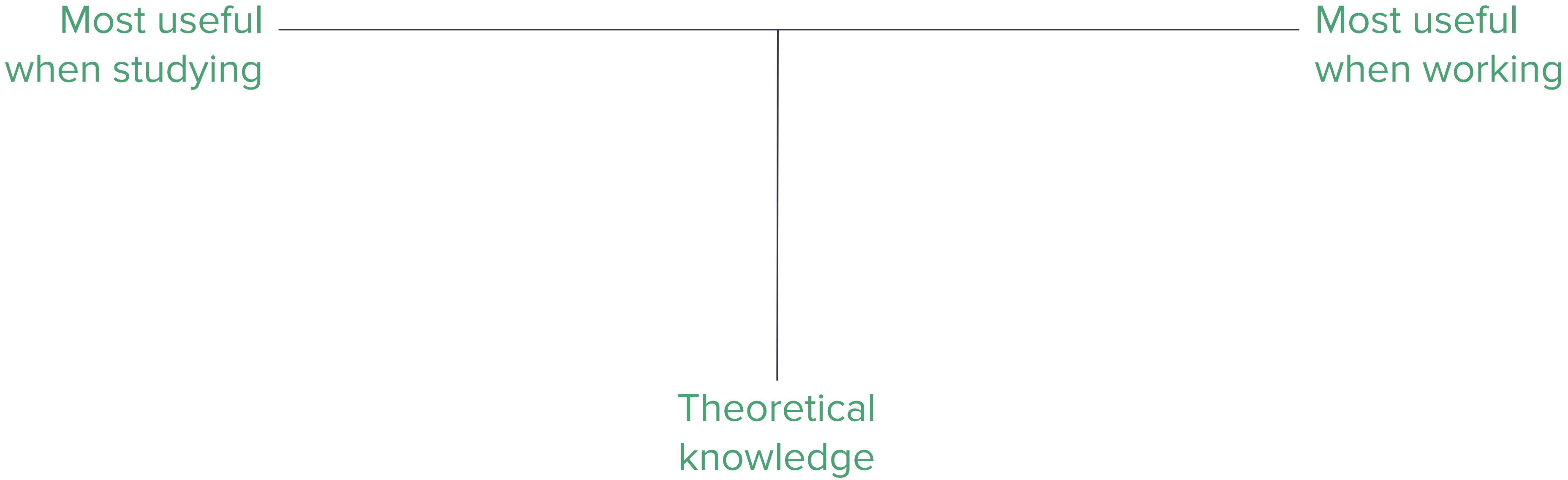

Practical steps
Most useful when studying
Most useful when working
Theoretical knowledge

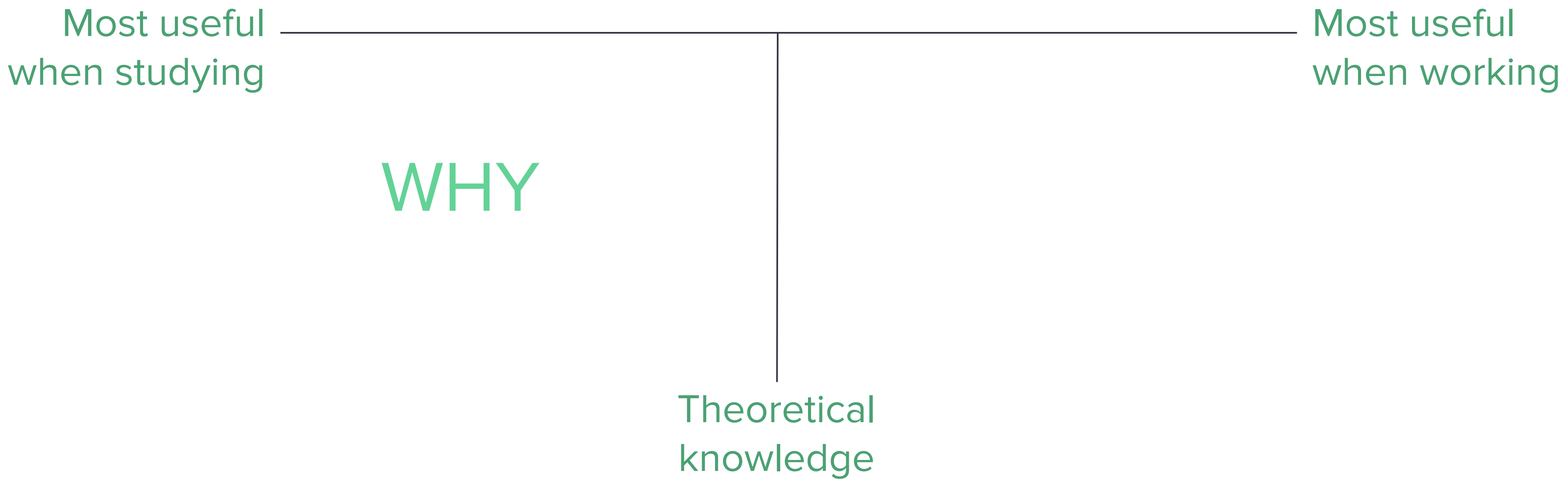

Practical steps
LEARNING-ORIENTED
TUTORIAL
Most useful when studying
Most useful when working
Theoretical knowledge

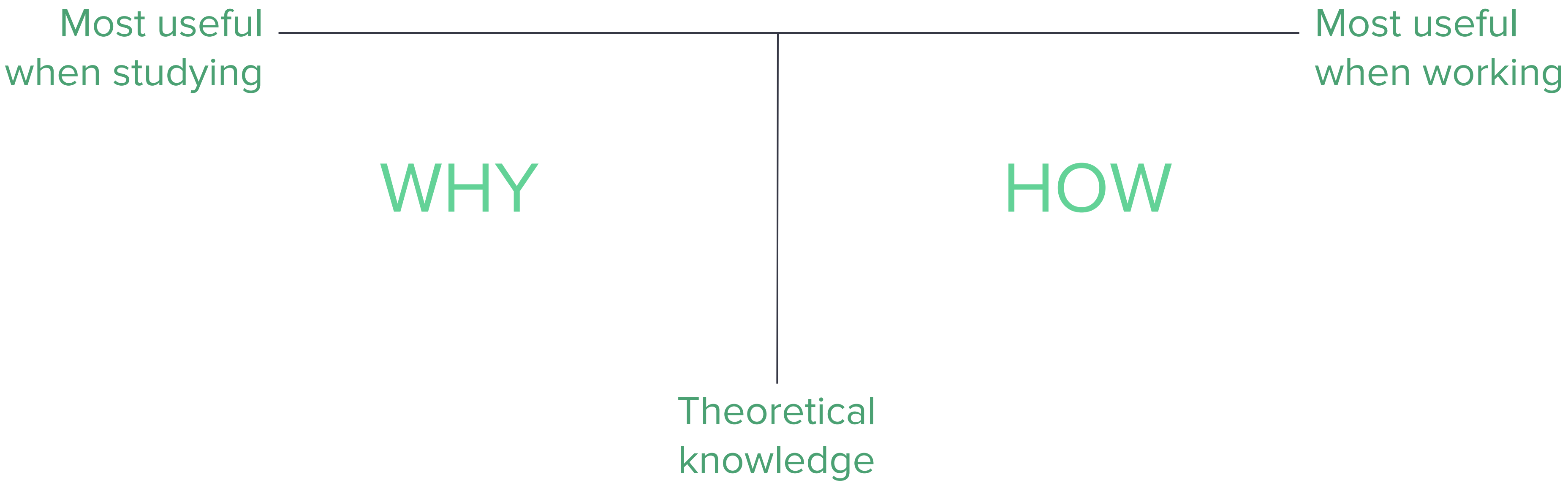

Practical steps
LEARNING-ORIENTED
TUTORIAL
PROBLEM-ORIENTED
HOW-TO
Most useful when studying
Most useful when working
Theoretical knowledge


Practical steps
LEARNING-ORIENTED
TUTORIAL
PROBLEM-ORIENTED
HOW-TO
Most useful when studying
Most useful when working
UNDERSTANDING-ORIENTED
EXPLANATION
Theoretical knowledge

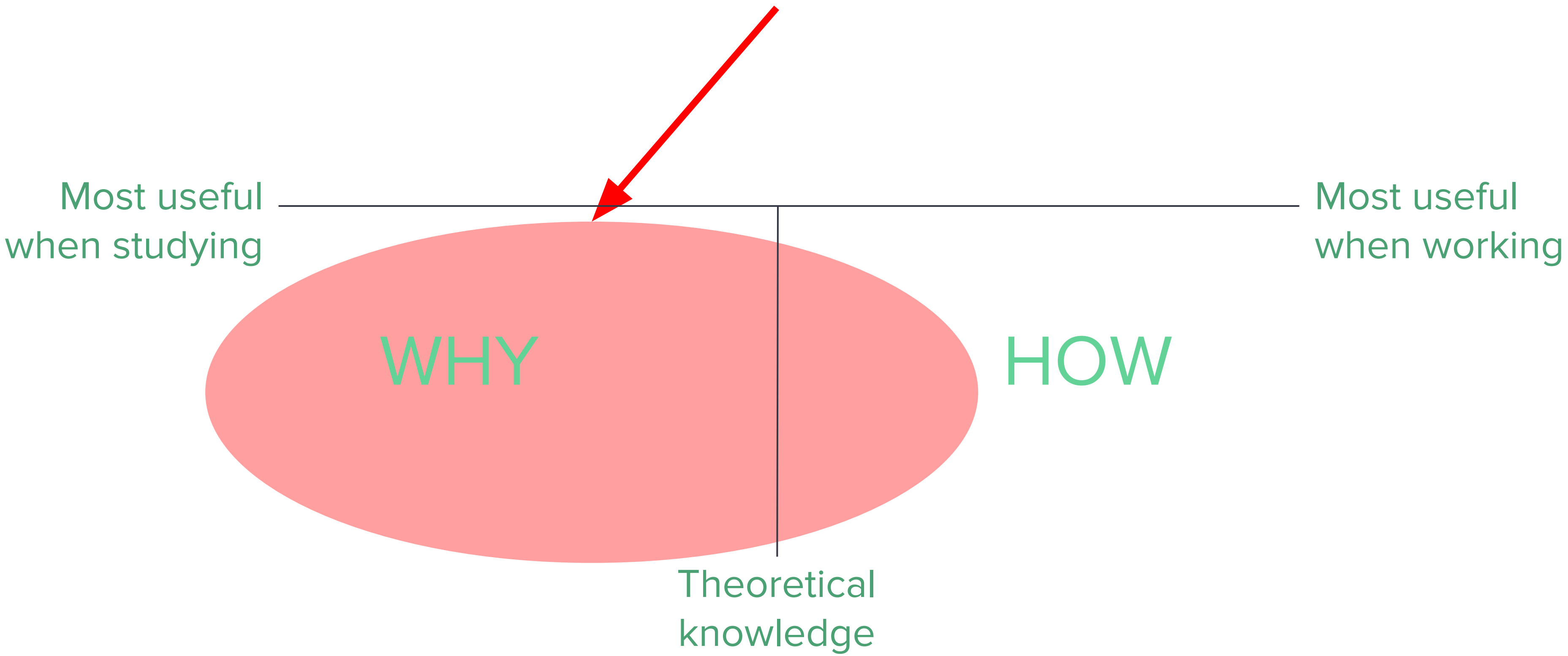

Practical steps
LEARNING-ORIENTED
TUTORIAL
PROBLEM-ORIENTED
HOW-TO
Most useful when studying
Most useful when working
UNDERSTANDING-ORIENTED
EXPLANATION
INFORMATION-ORIENTED
REFERENCE
Theoretical knowledge


Most useful when studying
Most useful when working
Theoretical knowledge

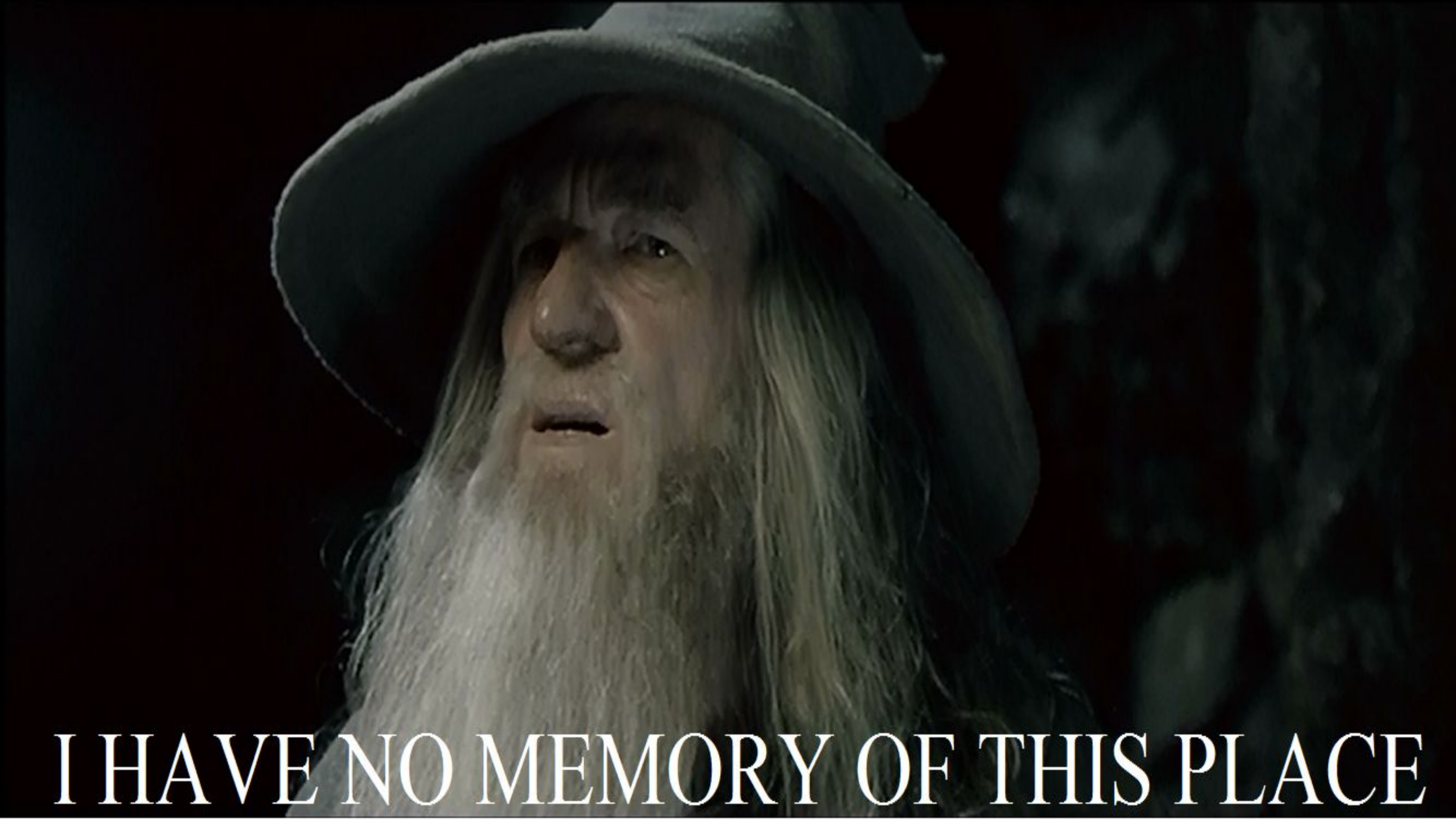

Most useful when studying
Most useful when working
WHY
Theoretical knowledge


Most useful when studying
Most useful when working
WHY
HOW
Theoretical knowledge

Lightweight Architecture Decision Records



Context: Why


I HAVE NO MEMORY OF THIS PLACE

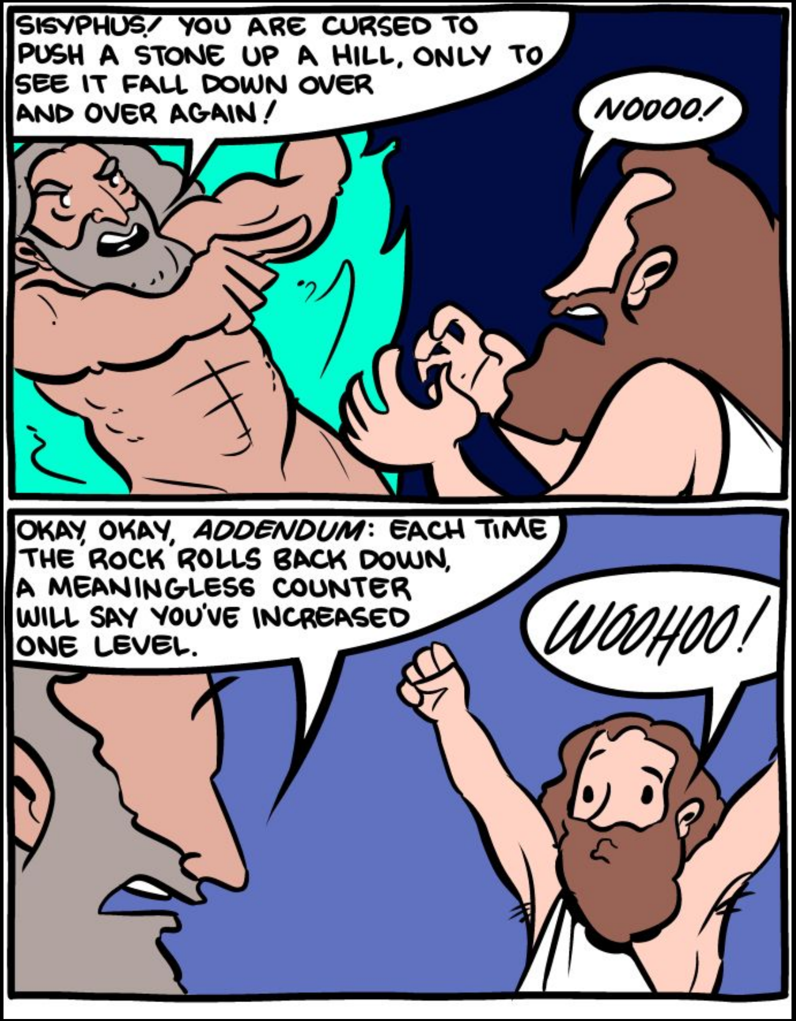
Context: why?

- New features
- Change existing features
- Refactor

BREAKING CHANGE

BREAKING CHANGE

STATUS QUO


SISYPHUS! YOU ARE CURSED TO PUSH A STONE UP A HILL, ONLY TO SEE IT FALL DOWN OVER AND OVER AGAIN!
NOOOO!
OKAY, OKAY, *ADDENDUM*: EACH TIME THE ROCK ROLLS BACK DOWN, A MEANINGLESS COUNTER WILL SAY YOU'VE INCREASED ONE LEVEL.
WOOHOO!

Context: recap

- Know thy history: keep track of events
- Take better decisions with greater insight
- Better ROI on documentation

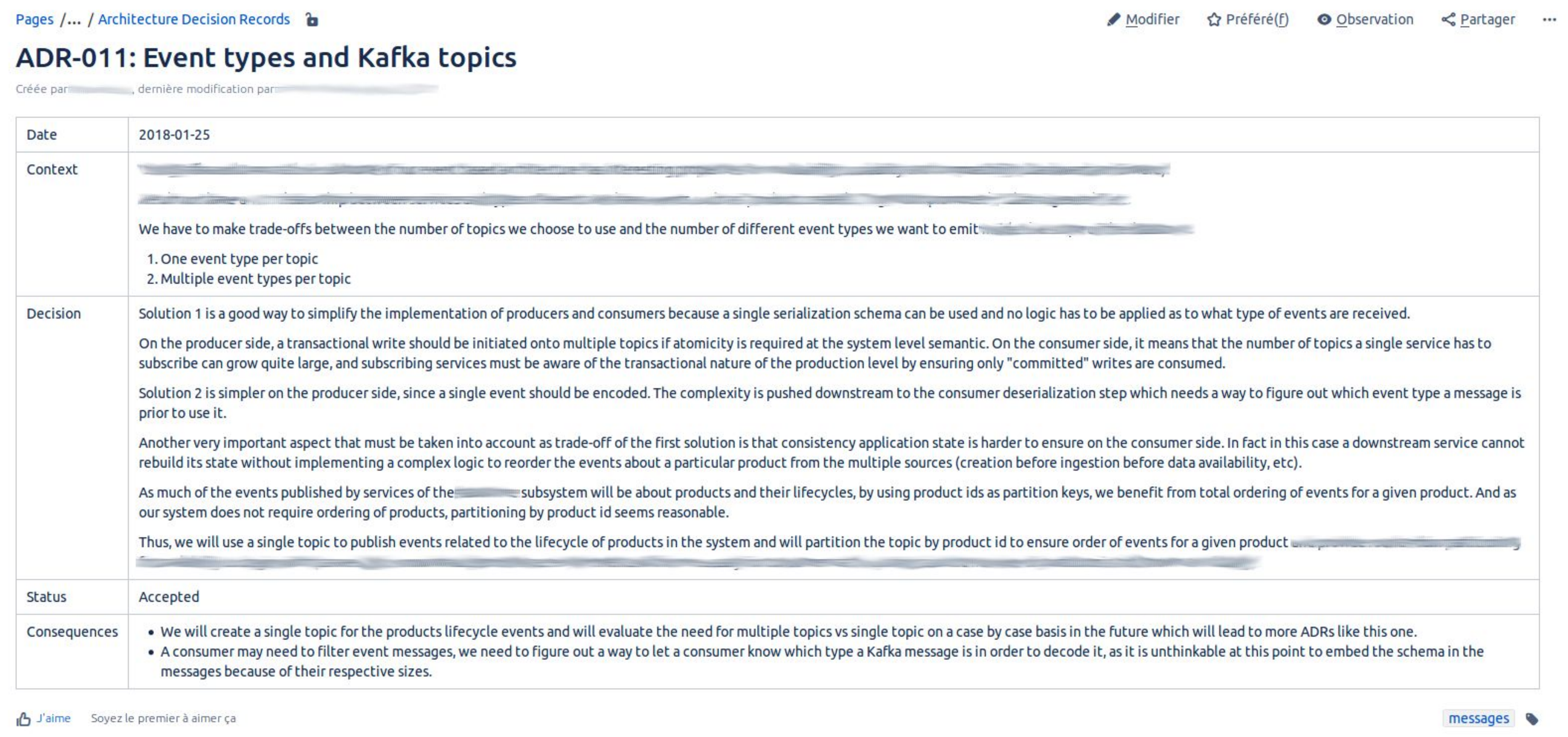
Decision: What/How

Title	ADR N°1: Use ADR
Date	
Context	
Decision	
Status	
Consequences	

Title	ADR N°1: Use ADR
Date	(local-date)
Context	
Decision	
Status	
Consequences	

Title	ADR N°1: Use ADR
Date	(local-date)
Context	Why? What’s the problem?
Decision	
Status	
Consequences	

Title	ADR N°1: Use ADR
Date	(local-date)
Context	Why? What’s the problem?
Decision	How? We will …
Status	
Consequences	



Pages /... / Architecture Decision Records

Modifier Préféré(f) Observation Partager ···

ADR-011: Event types and Kafka topics

Créée par [redacted], dernière modification par [redacted]

Date	2018-01-25
Context	[redacted] We have to make trade-offs between the number of topics we choose to use and the number of different event types we want to emit [redacted] 1. One event type per topic 2. Multiple event types per topic
Decision	Solution 1 is a good way to simplify the implementation of producers and consumers because a single serialization schema can be used and no logic has to be applied as to what type of events are received. On the producer side, a transactional write should be initiated onto multiple topics if atomicity is required at the system level semantic. On the consumer side, it means that the number of topics a single service has to subscribe can grow quite large, and subscribing services must be aware of the transactional nature of the production level by ensuring only "committed" writes are consumed. Solution 2 is simpler on the producer side, since a single event should be encoded. The complexity is pushed downstream to the consumer deserialization step which needs a way to figure out which event type a message is prior to use it. Another very important aspect that must be taken into account as trade-off of the first solution is that consistency application state is harder to ensure on the consumer side. In fact in this case a downstream service cannot rebuild its state without implementing a complex logic to reorder the events about a particular product from the multiple sources (creation before ingestion before data availability, etc). As much of the events published by services of the [redacted] subsystem will be about products and their lifecycles, by using product ids as partition keys, we benefit from total ordering of events for a given product. And as our system does not require ordering of products, partitioning by product id seems reasonable. Thus, we will use a single topic to publish events related to the lifecycle of products in the system and will partition the topic by product id to ensure order of events for a given product [redacted]
Status	Accepted
Consequences	• We will create a single topic for the products lifecycle events and will evaluate the need for multiple topics vs single topic on a case by case basis in the future which will lead to more ADRs like this one. • A consumer may need to filter event messages, we need to figure out a way to let a consumer know which type a Kafka message is in order to decode it, as it is unthinkable at this point to embed the schema in the messages because of their respective sizes.

J'aime Soyez le premier à aimer ça

messages

Decision

A timed PoC of the two approaches has been proposed but discarded because of timing constraints.

The second solution will be implemented for V1 and a PoC [redacted] will be postponed after V1 delivery to see if it fits.

Design:

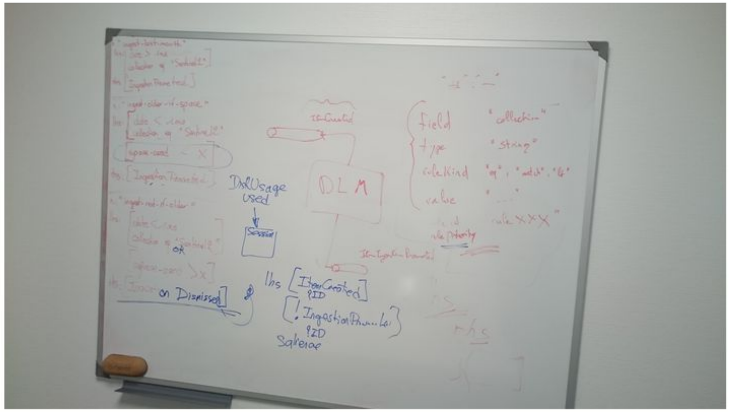
Rules stored in etcd.

Rules based on Strings or Dates. For string use "==" or "!=", for dates "<", ">" or "between".

Rules can be refreshed by a call to an HTTP endpoint of the DLM.

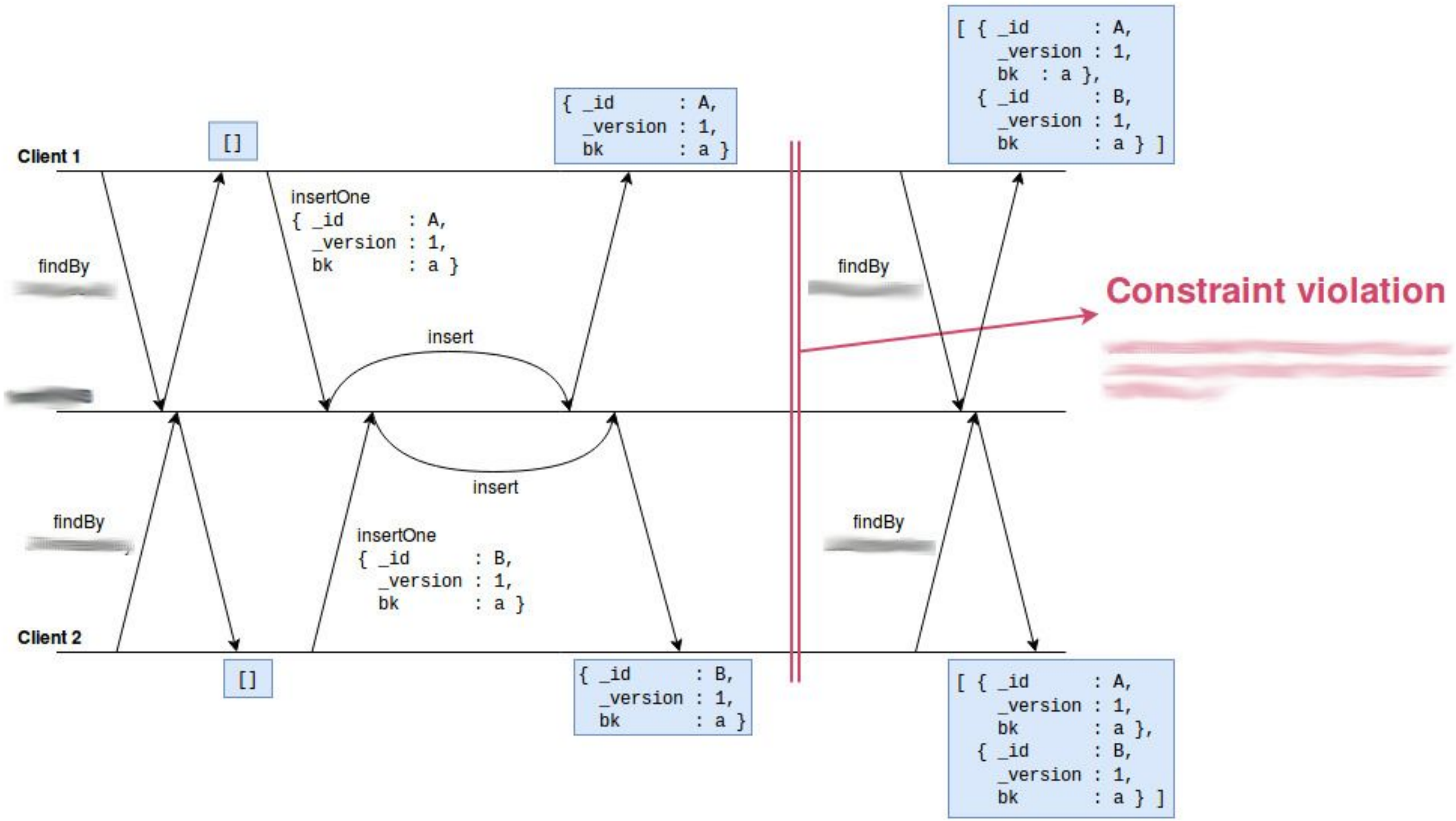
Rules have priorities. They are applied to incoming data in order of priority. When one rule matches, it is the one applied, following rules are not tested.

If no rule matches, the default rule of logging the decision to **not** ingest the product.



Sample:

```
{
    "id": "c26deab4-4278-4600-aff1-132e8694dde8",
    "name": "[redacted]",
    "doc": "[redacted]",
    "lhs": [{
        "field": "product.collection",
        "value": "[redacted]",
        "kind": "EQ",
        "type": "STRING"
    }, {
        "field": "pluginData([redacted])",
        "value": "[redacted]",
        "kind": "EQ",
        "type": "STRING"
    }, {
        "field": "pluginData([redacted])",
        "value": "1 MONTHS AGO",
        "kind": "GT",
        "type": "DATE"
    }],
    "pluginData": null
}
```



```
[]
```

```
insertOne
{ _id      : A,
  _version : 1,
  bk       : a }
```

```
{ _id      : A,
  _version : 1,
  bk       : a }
```

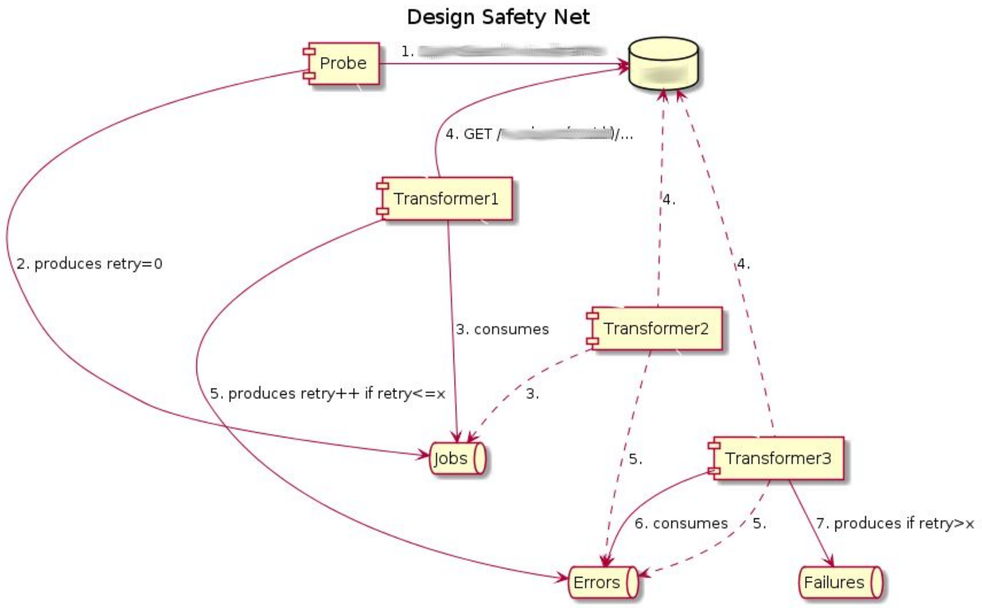
```
[ { _id      : A,
    _version : 1,
    bk   : a },
  { _id      : B,
    _version : 1,
    bk       : a } ]
```

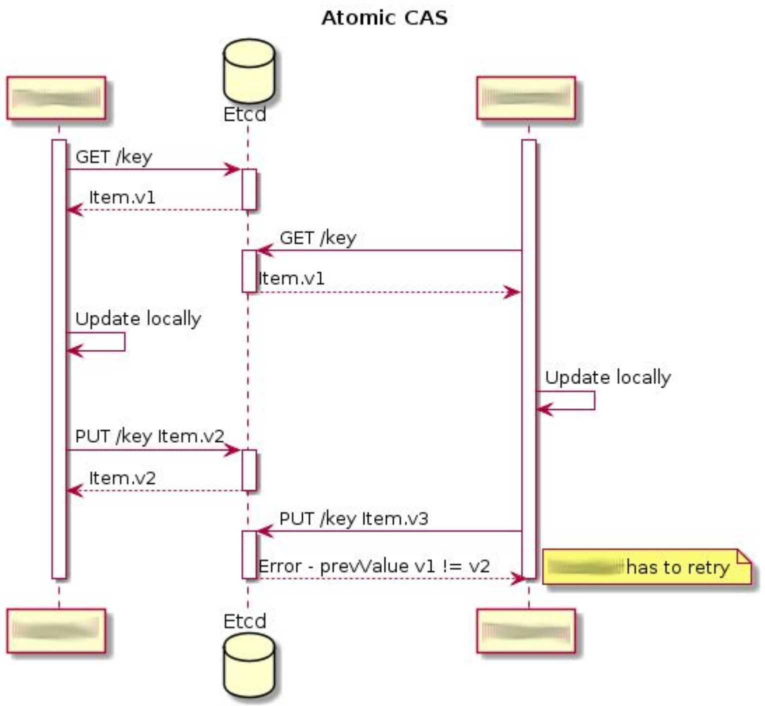
```
[]
```

```
insertOne
{ _id      : B,
  _version : 1,
  bk       : a }
```

```
{ _id      : B,
  _version : 1,
  bk       : a }
```

```
[ { _id      : A,
    _version : 1,
    bk       : a },
  { _id      : B,
    _version : 1,
    bk       : a } ]
```


Design Safety Net
Probe
1. [redacted]
Transformer1
4. GET /[redacted]/...
2. produces retry=0
3. consumes
Transformer2
4.
4.
3.
5. produces retry++ if retry<=x
Jobs
5.
Transformer3
6. consumes
5.
7. produces if retry>x
Errors
Failures


Atomic CAS
[redacted]
Etcd
[redacted]
GET /key
Item.v1
GET /key
Item.v1
Update locally
Update locally
PUT /key Item.v2
Item.v2
PUT /key Item.v3
Error - prevValue v1 != v2
[redacted] has to retry
[redacted]
Etcd
[redacted]

Title	ADR N°1: Use ADR
Date	(local-date)
Context	Why? What’s the problem?
Decision	How? We will …
Status	Proposed/Accepted/Superseded
Consequences	

Status: Proposed

Status: Accepted

Status: Superseded

Title	ADR N°1: Use ADR
Date	(local-date)
Context	Why? What's the problem?
Decision	How? We will ...
Status	Proposed/Accepted/Superseded
Consequences	+ / -

Consequences: Good/Bad/Ugly

Consequences

Consequences

- Rigor VS Flexibility: Find the right balance

Consequences

- Rigor VS Flexibility: Find the right balance
- Have someone who likes to write (a little), it helps

Consequences

- Rigor VS Flexibility: Find the right balance
- Have someone who likes to write, it helps
- Be agile, do “design tasks”

Consequences

- Rigor VS Flexibility: Find the right balance
- Have someone who likes to write, it helps
- Be agile, do “design tasks”
- Hammock driven development

Consequences

- Rigor VS Flexibility: Find the right balance
- Have someone who likes to write, it helps
- Be agile, do “design tasks”
- Hammock driven development
- It’s not about tooling

Status: Proposed

Superseeds previous ADR

N°0: “Outdated doc is better than no doc (lol)”

<MonkeyPatch/>

Arnaud Bos

arnaud@monkeypatch.io

Divio: https://www.divio.com/blog/documentation/

Relevance: http://thinkrelevance.com/blog/2011/11/15/documenting-architecture-decisions

Moar: https://adr.github.io/

Even moar: https://github.com/joelparkerhenderson/architecture_decision_record/blob/01cc3c801b1cc61f82391a0a08986e4145e21c56/README.md

Hammock-driven development https://www.youtube.com/watch?v=f84n5oFoZBc