

GOTO **COPENHAGEN** 2025

September 29 - October 3 • TAP1, Copenhagen • gotocph.com

DOWNLOAD & USE **THE GOTO GUIDE APP**

DOWNLOAD THE APP
CREATE YOUR SCHEDULE
ASK QUESTIONS
RATE SESSIONS



A Commune in the Ivory Tower?

Andrew Harmel-Law
(they/them)

Tech Principal @Thoughtworks
`@andrewhl.bsky.social`



A Commune in the Ivory Tower?

Andrew Harmel-Law
(they/them)

Tech Principal @Thoughtworks
@andrewhl.bsky.social



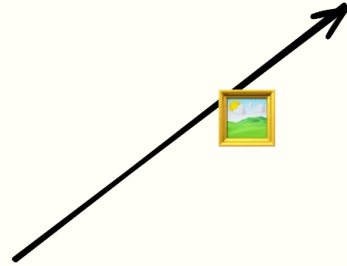
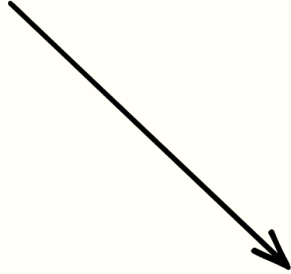


Image The Nightmare, Henry Fuseli, Public domain, via Wikimedia Commons



Image: The Nightmare, Henry Fuseli, Public domain, via Wikimedia Commons

Decision needed



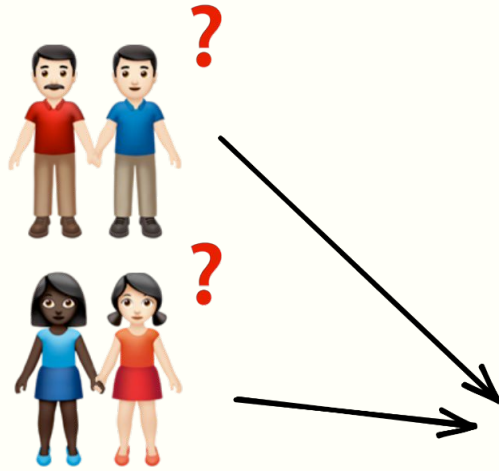
Decision received



Decision implemented

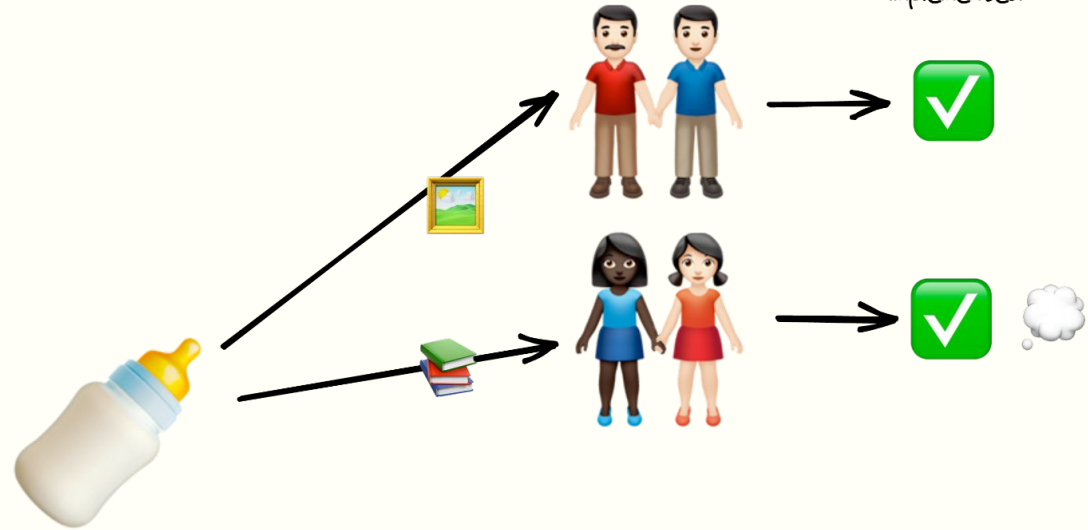


Decision needed

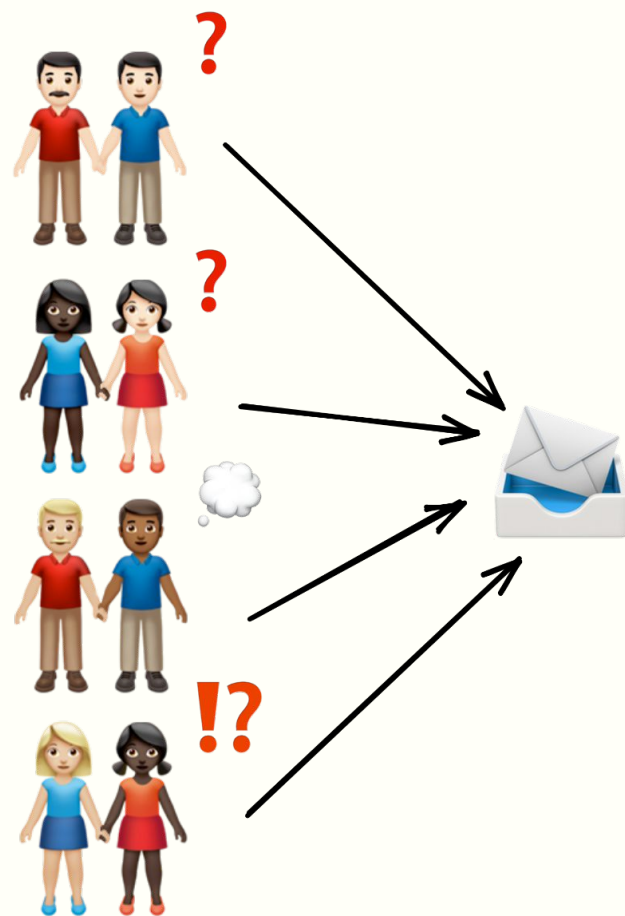


Decision received

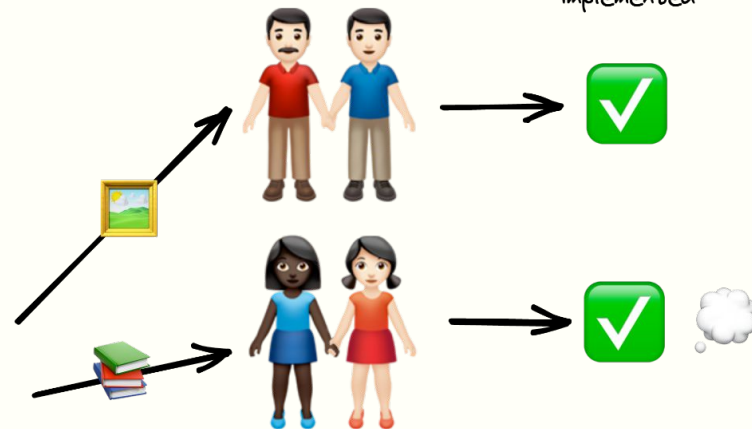
Decision implemented



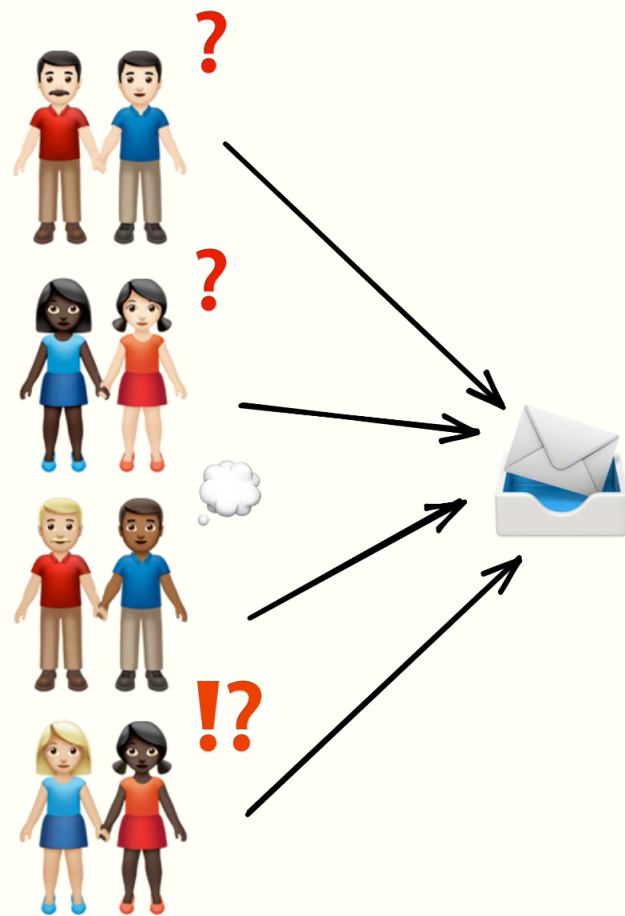
Decision needed



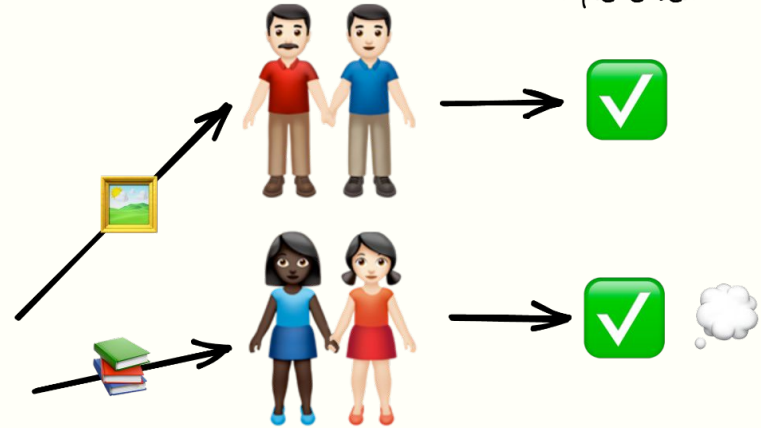
Decision received



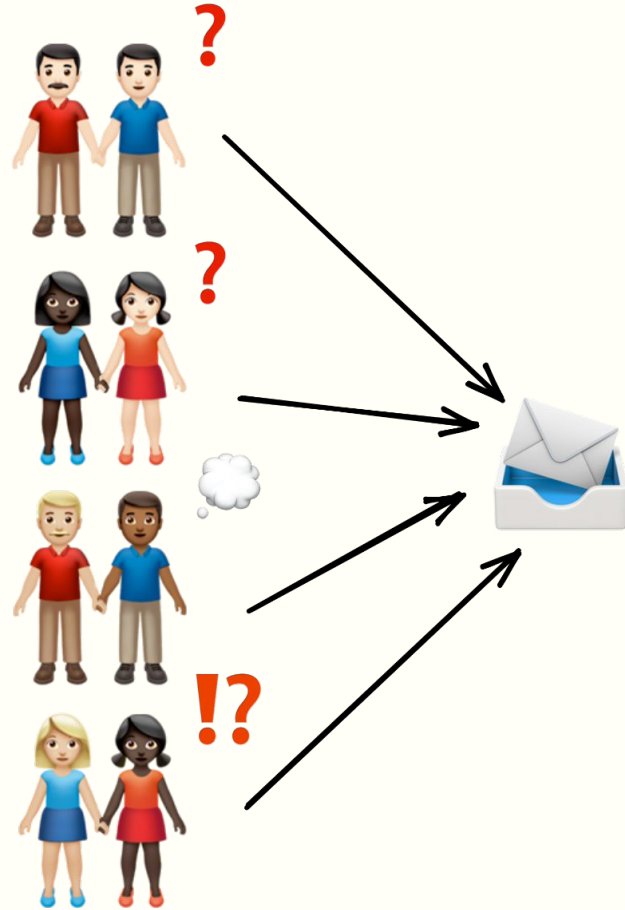
Decision needed



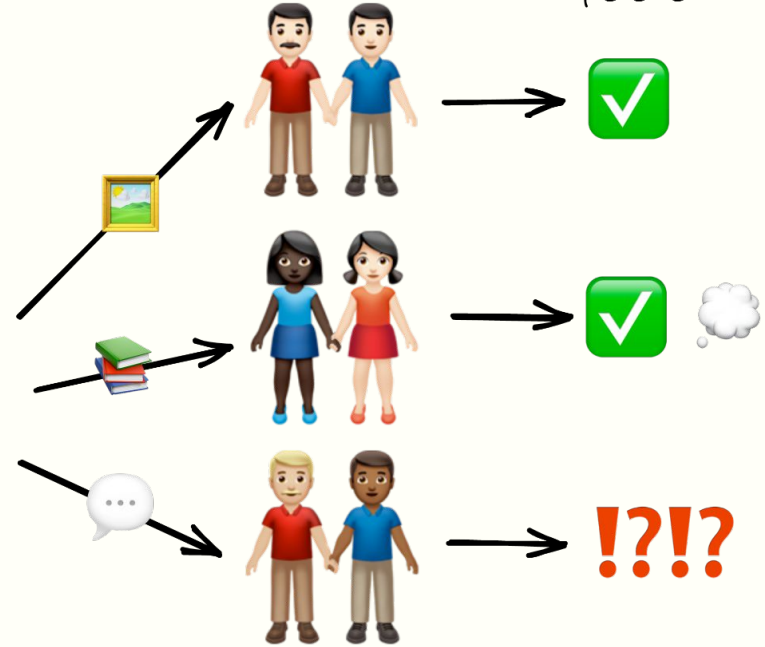
Decision received



Decision needed

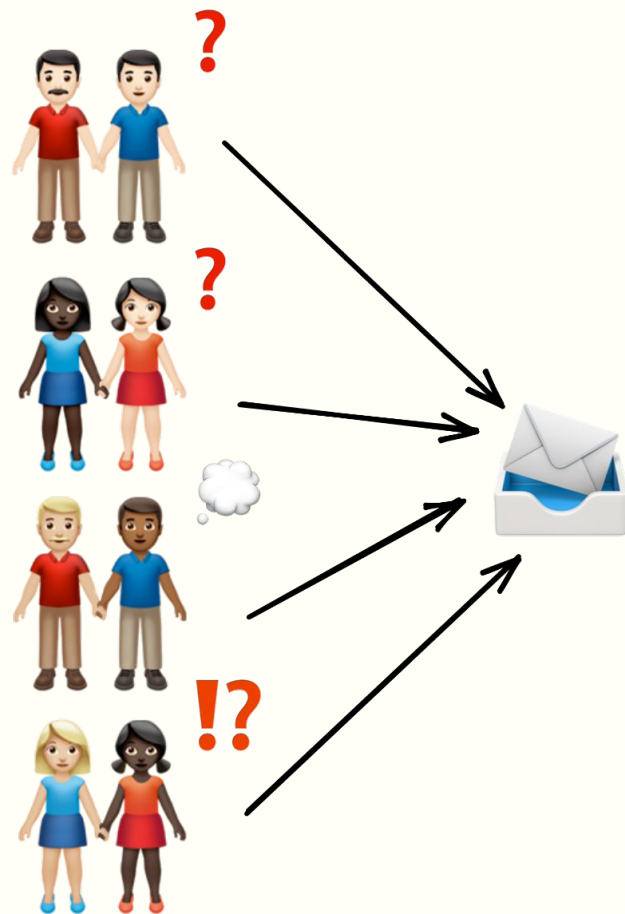


Decision received



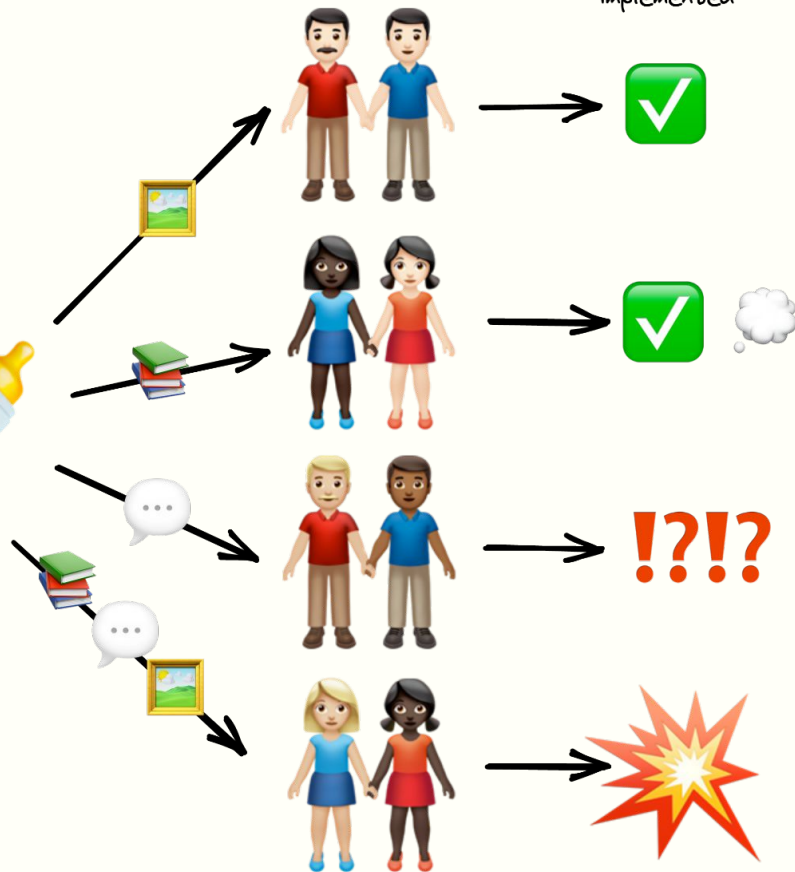
Decision implemented

Decision needed



Decision received

Decision implemented

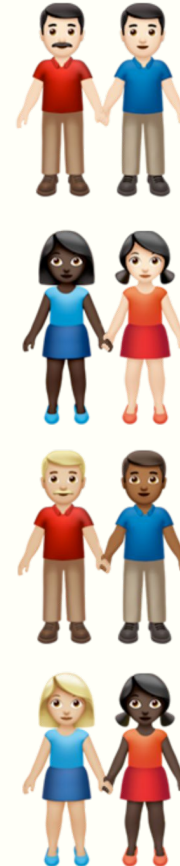


Decision needed



Decision being made

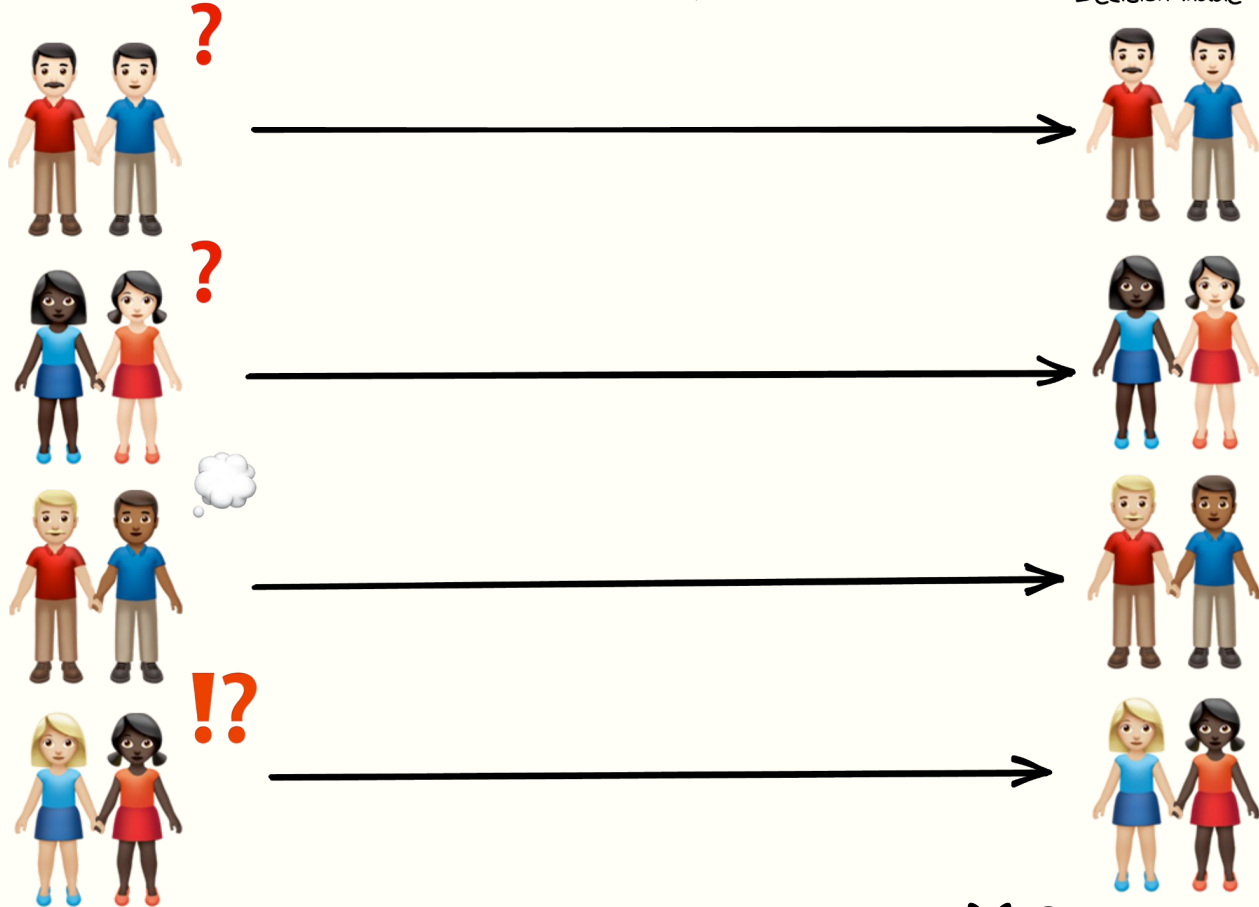
Decision made



Decision needed

Decision being made

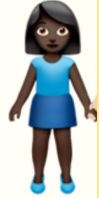
Decision made



Decision needed

Decision being made

Decision made



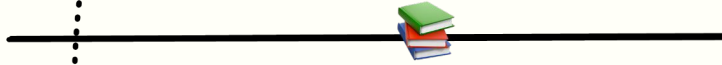
[] a decentralised way to
make decisions(?)



Decision needed

Decision being made

[] a decentralised way to make decisions



The Advice Process:

Anyone can make any decision

The Advice Process:

Anyone can make any decision

But before doing so they *must*:

The Advice Process:

Anyone can make any decision

But before doing so they *must*:

seek advice from all affected

parties

The Advice Process:

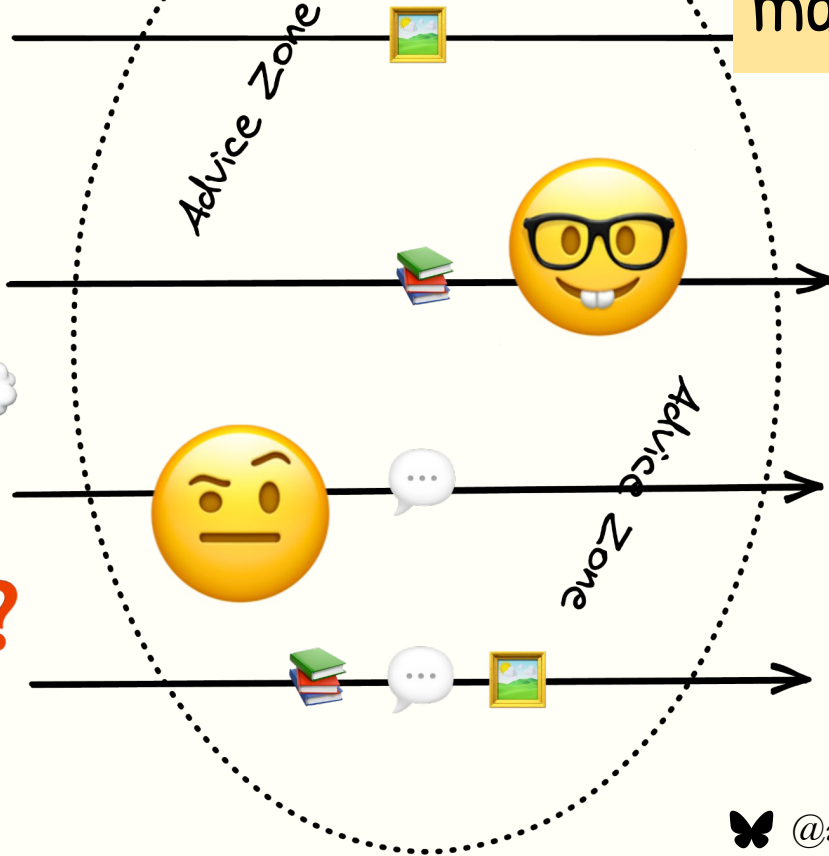
Anyone can make any decision
But before doing so they *must*:
seek advice from all affected
parties
& people with
expertise in the matter.



Decision needed

Decision being made

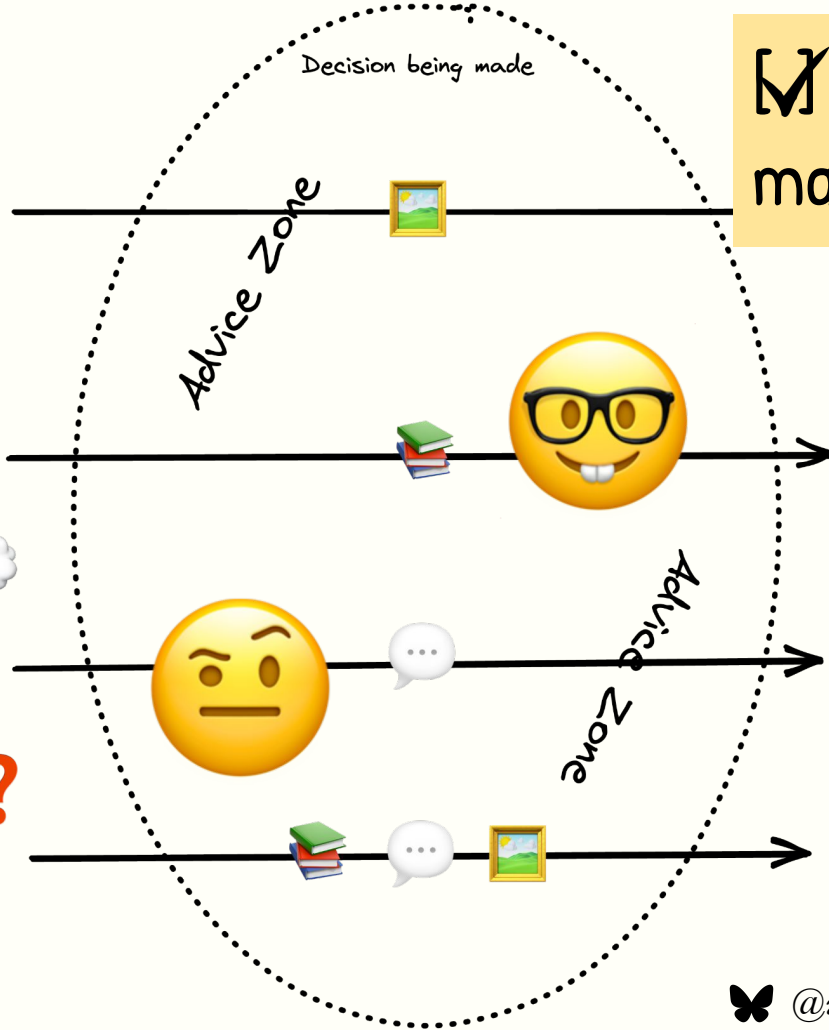
[] a decentralised way to make decisions



Decision needed

Decision being made

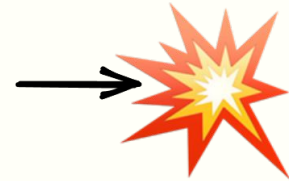
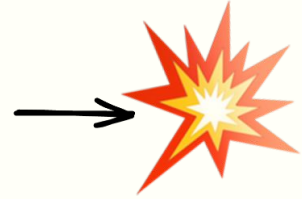
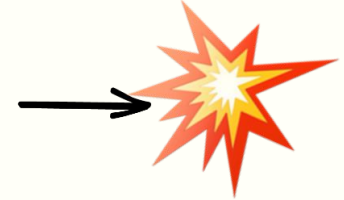
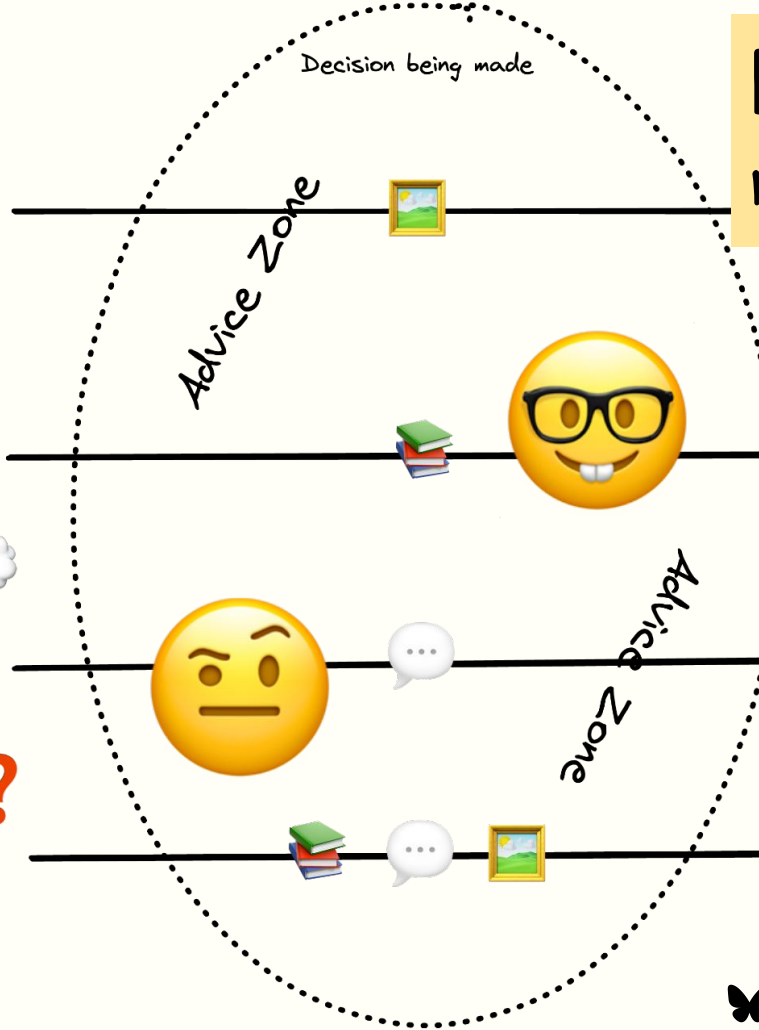
✓ a decentralised way to make decisions



Decision needed

Decision being made

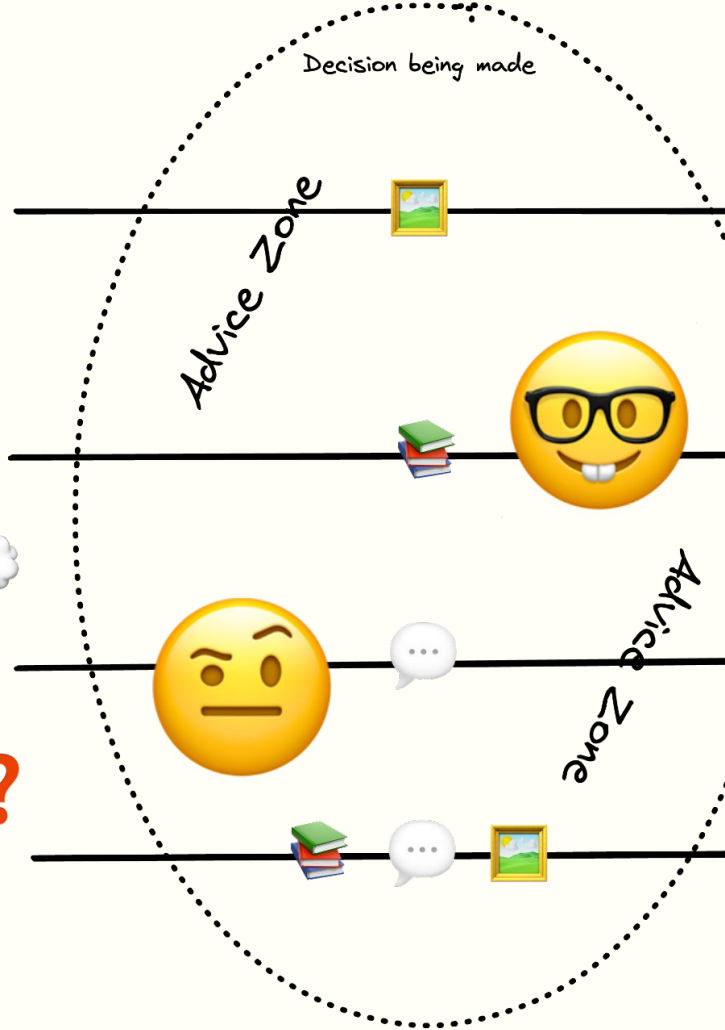
✓ a decentralised way to make decisions



@andrewhl.bsky.social #GOTOCph2025

Decision needed

Decision being made



✓ a decentralised way to make decisions
[] support less experienced in deciding

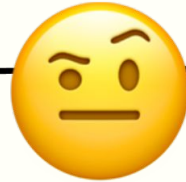
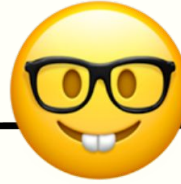


Decision needed

Decision being made



Advice Zone



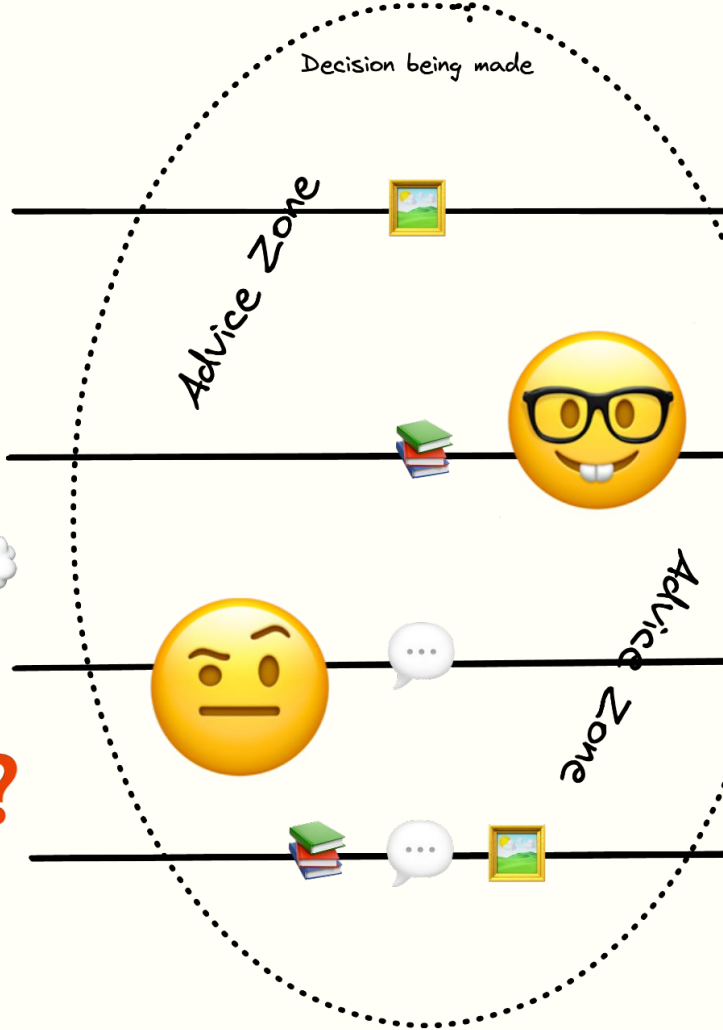
Advice Zone

☒ a decentralised way to make decisions
☐ support less experienced in deciding
☐ know who was consulted & their advice



Decision needed

Decision being made



☒ a decentralised way to make decisions
☐ support less experienced in deciding
☐ know who was consulted & their advice
☐ know about all decisions



Lightweight ADRs^{*} for decision support

^{*} Architecture Decision Record

☒ a decentralised way to
make decisions
☐ support less
experienced in deciding
☐ know who was
consulted & their advice
☐ know about all
decisions

ADR-01 Short Inventory Ids

Date: 2020-03-25
Driver: Wisen Tanasa

Status

Accepted - 2020-03-25

Decision

- We will increase the ID length when we're scaling up
- We're going with option 1, and will use [nanoid](#)
- with the following configuration:
 - Building ID:
 - ID Length: 6

Context

Our building

22cddcb6-c0e5-4baa-a701-78584fc2a9e

We would like to make our building id shorter to make the seekers' UX better:

- They would be able to type the ID in the browser if they are printing the building page
- They would be able to share the URL without URL shortener

These are the decision criteria:

- Short
- Low collision probability (<https://zslark.github.io/nano-id-cc/>)

- Unambiguity, 0 (zero) and O (capital o) must not be confused
- Cost of implementation

Based on the above consideration, we have the following options:

1. AaVh4QFG: Random generated letters and numbers
2. 123456: Automatically generated sequence ID
3. canvas: Manually generated ID
4. 80882221: Pretty generated letters and numbers
5. canvas-AaVh: Combination of building name and generated id

On a discussion and research:

- We ruled out option 2, as it will be too costly to implement. It requires new infrastructure to be provisioned as we're adopting serverless architecture. Generated sequence is a concept from RDBMS, if we randomly generate numbers, the collision probability is also quite high and will need a longer id.
- We ruled out option 3, as it will involve too much human intervention
- We ruled out option 4, as we couldn't find a library that will easily support this
- We ruled out option 5, as we can use a slug if we're concerned about the URL

Consequences

- We may produce profanity words, but we're not bothered as we might get more traffic from the joke
- We don't have to provision any infrastructure, as the ID can be generated with nanoid tiny library
- If we generate 1 building ID per hour, we have 1% probability of 1 collision. We think this is quite high, and we don't think that we'll have 1 building ID per hour.

Advice

- Have we thought about profanity in the auto-generated text? (A considerate Product Manager)
- What's the possibility of id collisions? (The DBA)
- Have we looked at generating this in the database? (Also the DBA)
- Does making it human-meaningful matter, or just human-readable? (Someone from UX)
- Can we list out all the places where we'll need to use this id? (Concerned Tech Lead from another team)
- Will the ids be exposed publicly? Have we thought about leakage of internal data models? (The Infosec Professional)



@andrewhl.bsky.social #GOTOCph2025

ADR-01 Short Inventory Ids

Date: 2020-03-25
Driver: Wisen Tanasa

Status

Accepted - 2020-03-25

Decision

- We will increase the ID length when we're scaling up
- We're going with option 1, and will use [nanoid](#)
- with the following configuration:
 - Building ID:
 - ID Length: 6
 - Characters: 23456789ABCDEFHJKMNPQRSTUVWXYZ
 - Space ID:
 - ID Length: 8
 - Characters: 0123456789ABCDEFHJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz
 - Provider ID:
 - ID Length: 5
 - Characters: 0123456789ABCDEFHJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz

Context

Our building id is currently generated from a UUID
22cadc06-c0e5-4baa-a701-7858541

We would like to make our building id shorter to

- They would be able to type the ID in the s
- They would be able to share the URL with

These are the decision criteria:

- Short
- Low collision probability (<https://zelen.gi>)

- Unambiguity, 0 (zero) and O (capital o) m
- Cost of implementation

Based on the above consideration, we have the following options:

1. AaVvH4OfG: Random generated letters and numbers
2. 123456: Automatically generated sequence ID
3. canvas: Manually generated ID
4. 80882221: Pretty generated letters and numbers
5. canvas-AaVvH: Combination of building name and generated id

On a discussion and research:

- We ruled out option 2, as it will be too costly to implement. It requires new infrastructure to be provisioned as we're adopting serverless architecture. Generated sequence is a concept from RDBMS, if we randomly generate numbers, the collision probability is also quite high and will need a longer id.
- We ruled out option 3, as it will involve too much human intervention
- We ruled out option 4, as we couldn't find a library that will easily support this
- We ruled out option 5, as we can use a slug if we're concerned about the URL

Consequences

- We may produce profanity words, but we're not bothered as we might get more traffic from the joke
- We don't have to provision any infrastructure, as the ID can be generated with nanoid try library
- If we generate 1 building ID per hour, we have 1% probability of 1 collision. We think this is quite high, and we don't think that we'll have 1 building ID per hour.

Advice

- Have we thought about profanity in the auto-generated text? (A considerate Product Manager)
- What's the possibility of id collisions? (The DBA)
- Have we looked at generating this in the database? (Also the DBA)
- Does making it human-meaningful matter, or just human-readable? (Someone from UX)
- Can we list out all the places where we'll need to use this id? (Concerned Tech Lead from another team)
- Will the ids be exposed publicly? Have we thought about leakage of internal data models? (The Infosec Professional)

Decision

- We will increase the ID length when we're scaling up
- We're going with random generated letters and numbers (option 1) and will use [nanoid](#) for this
 - with the following configuration:

- Building ID:



@andrewhl.bsky.social #GOTOCph2025

ADR-01 Short Inventory Ids

Date: 2020-03-25
Driver: Wisen Tanasa

Status

Accepted - 2020-03-25

Decision

- We will increase the ID length when we're scaling up
- We're going with option 1, and will use [nanoid](#)
- with the following configuration:
 - Building ID:
 - ID Length: 6
 - Characters: 23456789ABCDEFHGHIJKLMNOPQRSTUVWXYZ
 - Space ID:
 - ID Length: 8
 - Characters: 0123456789ABCDEFHGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz
 - Provider ID:
 - ID Length: 5
 - Characters: 0123456789ABCDEFHGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz

Context

Our building id is currently generated from a UUID, and looks like:

22cadcb6-00e5-4baa-a701-785854fc2a9e

We would like to make our building id shorter to make the seekers' UX better:

- They would be able to type the ID in the browser
- They would be able to share the URL with others

These are the decision criteria:

- Short
- Low collision probability (<https://zslens.gty>)

Context

Our building id is currently generated from a UUID, and looks like:

22cadcb6-00e5-4baa-a701-785854fc2a9e.

We would like to make our building id shorter to make the seekers' UX better:

- They would be able to type the ID in the browser if they are printing the building page

- Unambiguity, 0 (zero) and O (capital o) must not be used
- Cost of implementation

Based on the above consideration, we have the following options:

1. A4VH4CFG: Random generated letters and numbers
2. 123456: Automatically generated sequence
3. canvas: Manually generated ID
4. 80882221: Pretty generated letters and numbers
5. canvas-A4VH: Combination of building name and random letters

On a discussion and research:

- We ruled out option 2, as it will be too complex to be provisioned as we're adopting server concept from RDBMS, if we randomly generate quite high and will need a longer id.
- We ruled out option 3, as it will involve too much human intervention
- We ruled out option 4, as we couldn't find a library that will easily support this
- We ruled out option 5, as we can use a slug if we're concerned about the URL

Consequences

- We may produce profanity words, but we're not bothered as we might get more traffic from the joke
- We don't have to provision any infrastructure, as the ID can be generated with nanoid tiny library
- If we generate 1 building ID per hour, we have 1% probability of 1 collision. We think this is quite high, and we don't think that we'll have 1 building ID per hour.

Advice

- Have we thought about profanity in the auto-generated text? (A considerate Product Manager)
- What's the possibility of id collisions? (The DBA)
- Have we looked at generating this in the database? (Also the DBA)
- Does making it human-meaningful matter, or just human-readable? (Someone from UX)
- Can we list out all the places where we'll need to use this id? (Concerned Tech Lead from another team)
- Will the ids be exposed publicly? Have we thought about leakage of internal data models? (The Infosec Professional)



@andrewhl.bsky.social #GOTOCph2025

ADR-01 Short Inventory Ids

Date: 2020-03-25
Driver: Wisen Tanasa

Status

Accepted - 2020-03-25

Decision

- We will increase the ID length when we're scaling up
- We're going with option 1, and will use `nanoid`
- with the following configuration:
 - Building ID:
 - ID Length: 6
 - Characters: 23456789ABCDEFHJKMNPQRSTUVWXYZ
 - Space ID:
 - ID Length: 8
 - Characters: 0123456789ABCDEFHJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz
 - Provider ID:
 - ID Length: 5
 - Characters: 0123456789ABCDEFHJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz

Context

Our building id is currently generated from a UUID, and looks like:

22cadc06-c0e5-4baa-a701-785854fc2a9e.

We would like to make our building id shorter to make the seekers' UX better:

- They would be able to type the ID in the browser if they are printing the building page
- They would be able to share the URL without URL shortener

These are the decision criteria:

- Short
- Low collision probability (<https://zelenk.com>)

Options Considered

1. A4VHAOFG: Random generated letters and numbers
2. 123456: Automatically generated sequence ID:
3. canvas: Manually generated ID
4. BBBB2221: Pretty generated letters and numbers
5. canvas-A4VH: Combination of building name and generated id

- Unambiguity, 0 (zero) and O (capital o) must be avoided
- Cost of implementation

Based on the above consideration, we have the following options:

1. A4VHAOFG: Random generated letters and numbers
2. 123456: Automatically generated sequence ID
3. canvas: Manually generated ID
4. BBBB2221: Pretty generated letters and numbers
5. canvas-A4VH: Combination of building name and generated id

On a discussion and research:

- We ruled out option 2, as it will be too costly to be provisioned as we're adopting serverless concept from RDBMS, if we randomly generate quite high and will need a longer id.
- We ruled out option 3, as it will involve too much manual work.
- We ruled out option 4, as we couldn't find a library that we easily support and maintain.
- We ruled out option 5, as we can use a slug if we're concerned about the URL.

Consequences

- We may produce profanity words, but we're not bothered as we might get more traffic from the joke
- We don't have to provision any infrastructure, as the ID can be generated with nanoid tiny library
- If we generate 1 building ID per hour, we have 1% probability of 1 collision. We think this is quite high, and we don't think that we'll have 1 building ID per hour.

Advice

- Have we thought about profanity in the auto-generated text? (A considerate Product Manager)
- What's the possibility of id collisions? (The DBA)
- Have we looked at generating this in the database? (Also the DBA)
- Does making it human-meaningful matter, or just human-readable? (Someone from UX)
- Can we list out all the places where we'll need to use this id? (Concerned Tech Lead from another team)
- Will the ids be exposed publicly? Have we thought about leakage of internal data models? (The Infosec Professional)



@andrewhl.bsky.social #GOTOcph2025

ADR-01 Short Inventory Ids

Date: 2020-03-25
Driver: Wisen Tanasa

Status

Accepted - 2020-03-25

Decision

- We will increase the ID length when we're scaling up
- We're going with option 1, and will use [nanoid](#)
- with the following configuration:
 - Building ID:
 - ID Length: 6
 - Characters: 23456789ABCDEF GHIJ KLMN PQRS TUVWXY Z
 - Space ID:
 - ID Length: 8
 - Characters: 0123456789ABCDEF GHIJ KLMN PQRS TUVWXY Z abcdefghijklmnopqrstuvwxyz
 - Provider ID:
 - ID Length: 5
 - Characters: 0123456789ABCDEF GHIJ KLMN PQRS TUVWXY Z abcdefghijklmnopqrstuvwxyz

Context

Our building id is currently generated from a UUID, and looks like:

```
22cadc06-c0e5-4baa-a701-785854fc2a9e.
```

We would like to make our building id shorter to make the seekers' UX better:

- They would be able to type the ID in the browser if they are printing the building page
- They would be able to share the URL without URL shortener

These are the decision criteria:

- Short
- Low collision probability (<https://zelenk.com>)

Consequences

- We don't have to provision any infrastructure, as the ID can be generated with nanoid tiny library
- If we generate 1 building ID per hour, we have 1% probability of 1 collision. We think this is quite high, and we don't think that we'll have 1 building ID per hour
- We may produce profanity words, but the likelihood of this seems incredibly low, and ids will clearly have been randomly generated to external consumers

Consequences

- We may produce profanity words, but we're not bothered as we might get more traffic from the joke
- We don't have to provision any infrastructure, as the ID can be generated with nanoid tiny library
- If we generate 1 building ID per hour, we have 1% probability of 1 collision. We think this is quite high, and we don't think that we'll have 1 building ID per hour.

Advice

- Have we thought about profanity in the auto-generated text? (A considerate Product Manager)
- What's the possibility of id collisions? (The DBA)
- Have we looked at generating this in the database? (Also the DBA)
- Does making it human-meaningful matter, or just human-readable? (Someone from UX)
- Can we list out all the places where we'll need to use this id? (Concerned Tech Lead from another team)
- Will the ids be exposed publicly? Have we thought about leakage of internal data models? (The Infosec Professional)



@andrewhl.bsky.social #GOTOCph2025

ADR-01 Short Inventory Ids

Date: 2020-03-25
Driver: Wisen Tanasa

Status

Accepted - 2020-03-25

Decision

- We will increase the ID length when we're scaling up
- We're going with option 1, and will use `nanoid`
- with the following configuration:
 - Building ID:
 - ID Length: 8
 - Characters: 23456789ABCDEF GHIJ KLMN PQ RSTUVWXY Z
 - Space ID:
 - ID Length: 8
 - Characters: 0123456789ABCDEF GHIJ KLMN PQ RSTUVWXY Z abcdefghijklmnopqr stuvw xyz
 - Provider ID:
 - ID Length: 5
 - Characters: 0123456789ABCDEF GHIJ KLMN PQ RSTUVWXY Z abcdefghijklmnopqr stuvw xyz

Context

Our building id is currently generated from a UUID, and looks like:

22cadcb6-c0e5-4baa-a701-785854fc2a9e.

We would like to make our building id shorter to make the seekers' UX better:

- They would be able to type the ID in the browser if they are printing the building page
- They would be able to share the URL without URL shortener

These are the decision criteria:

- Short
- Low collision probability (<https://zslark.github.io/nano-id-cc/>)

Advice

- Unambiguity, 0 (zero) and O (capital o) must be avoided
- Cost of implementation

Based on the above consideration, we have the following options:

1. AUVHACFG: Random generated letters and numbers
2. 123456: Automatically generated sequence
3. canvas: Manually generated ID
4. 80882221: Pretty generated letters and numbers
5. canvas-AUVH: Combination of building name and random letters

On a discussion and research:

- We ruled out option 2, as it will be too complex to be provisioned as we're adopting server concept from RDBMS. If we randomly generate quite high and will need a longer ID.
- We ruled out option 3, as it will involve too much manual work.
- We ruled out option 4, as we couldn't find a better way to generate a unique ID.
- We ruled out option 5, as we can use a standard ID.

Consequences

- We may produce profanity words, but we'll filter them from the joke
- We don't have to provision any infrastructure or any library
- If we generate 1 building ID per hour, we'll have 1 ID per hour. It's quite high, and we don't think that we'll have a running server per hour.

Advice

- Have we thought about profanity in the auto-generated text? (A considerate Product Manager)
- What's the possibility of id collisions? (The DBA)
- Have we looked at generating this in the database? (Also the DBA)
- Does making it human-meaningful matter, or just human-readable? (Someone from UX)
- Can we list out all the places where we'll need to use this id? (Concerned Tech Lead from another team)
- Will the ids be exposed publicly? Have we thought about leakage of internal data models? (The Infosec Professional)

- Have we thought about profanity in the auto-generated text? (A considerate Product Manager)
- What's the possibility of id collisions? (The DBA)
- Have we looked at generating this in the database? (Also the DBA)
- Does making it human-meaningful matter, or just human-readable? (Someone from UX)
- Can we list out all the places where we'll need to use this id? (Concerned Tech Lead from another team)
- Will the ids be exposed publicly? Have we thought about leakage of internal data models? (The Infosec Professional)



@andrewhl.bsky.social #GOTOCph2025

Lightweight ADRs for decision support

- ☒ a decentralised way to make decisions
- ☐ support less experienced in deciding
- ☐ know who was consulted & their advice
- ☐ know about all decisions

Lightweight ADRs for decision support

- ☒ a decentralised way to make decisions
- ☒ support less experienced in deciding
- ☐ know who was consulted & their advice
- ☐ know about all decisions

Lightweight ADRs for decision support

- ✓ a decentralised way to make decisions
- ✓ support less experienced in deciding
- ✓ know who was consulted & their advice
- [] know about all decisions

Lightweight ADRs for decision support

- ✓ a decentralised way to make decisions
- ✓ support less experienced in deciding
- ✓ know who was consulted & their advice
- ✓ know about all decisions

Lightweight ADRs for decision support

- ✓ a decentralised way to make decisions
- ✓ support less experienced in deciding
- ✓ know who was consulted & their advice
- ✓ know about all decisions
- [] refactor to optimise

1. Advice Process
2. Lightweight ADRS

- ✓ a decentralised way to make decisions
- ✓ support less experienced in deciding
- ✓ know who was consulted & their advice
- ✓ know about all decisions
- [] refactor to optimise

1. Advice Process
2. Lightweight ADRS
3. Architectural Advice Forum

☒ a decentralised way to make decisions

☒ support less experienced in deciding

☒ know who was consulted & their advice

☒ know about all decisions

☐ refactor to optimise

Conversations at an Architectural Advice Forum

- ☒ a decentralised way to make decisions
- ☒ support less experienced in deciding
- ☒ know who was consulted & their advice
- ☒ know about all decisions
- ☐ refactor to optimise

Conversations at an Architectural Advice Forum

- ✓ a decentralised way to make decisions
- ✓ support less experienced in deciding
- ✓ know who was consulted & their advice
- ✓ know about all decisions
- ✓ refactor to optimise

“[In order for an architecture to be successful]
it is very much about ensuring that conversations
that are needed to be happening are happening -

-- Ruth Malan

“[In order for an architecture to be successful]
it is very much about ensuring that conversations
that are needed to be happening are happening -
not always initiating them,

-- Ruth Malan

“[In order for an architecture to be successful]
it is very much about ensuring that conversations
that are needed to be happening are happening -
not always initiating them,
nor always helping to focus or navigate them,

-- Ruth Malan

“[In order for an architecture to be successful]
it is very much about ensuring that conversations
that are needed to be happening are happening -
not always initiating them,
nor always helping to focus or navigate them,
but ensuring they do happen [...]

-- Ruth Malan

“[In order for an architecture to be successful]
it is very much about ensuring that conversations
that are needed to be happening are happening -
not always initiating them,
nor always helping to focus or navigate them,
but ensuring they do happen [...] & guiding when needed.”

-- Ruth Malan

1. Advice Process
2. Lightweight ADRS
3. Architectural Advice Forum

- ✓ a decentralised way to make decisions
- ✓ support less experienced in deciding
- ✓ know who was consulted & their advice
- ✓ know about all decisions
- ✓ refactor to optimise

How to fail.

How to fail #1:

“Bad”

Decisions

How to fail #2:

Old guard ==

New guard.

How to fail #3:

Off-the-grid
decisions.

How to fail #4:

No trust.

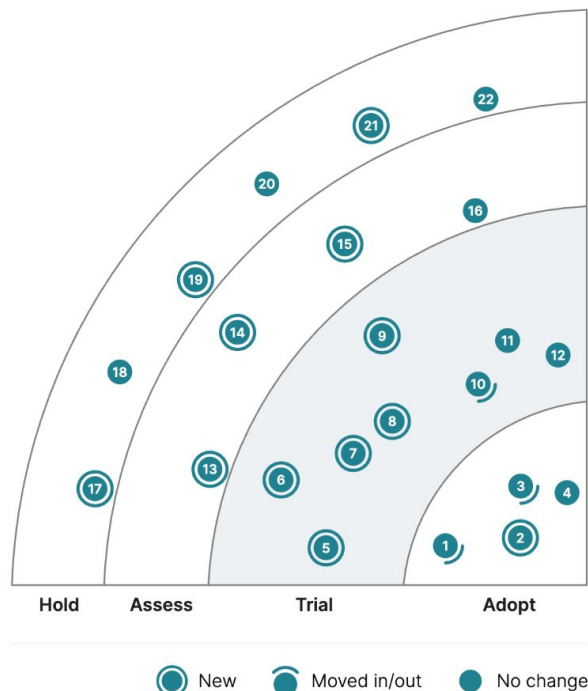
How to fail:

1. “Bad” decisions
2. Old guard == New guard
3. Off-the-grid decisions
4. No trust

It's an “anybody”
approach to
architecture.



Techniques



6. Architecture advice process

One of the persistent challenges in large software teams is determining who makes the architectural decisions that shape the evolution of systems. The [State of DevOps report](#) reveals that the traditional approach of Architecture Review Boards is counterproductive, often hindering workflow and correlating with low organizational performance. A compelling alternative is an **architectural advice process** — a decentralized approach where anyone can make any architectural decision, provided they seek advice from those affected and those with relevant expertise. This method enables teams to optimize for flow without compromising architectural quality, both at small and large scales. At first glance, this approach may seem controversial, but practices such as [Architecture Decision Records](#) and advisory forums ensure that decisions remain informed, while empowering those closest to the work to make decisions. We've seen this model succeed at scale in an increasing number of organizations, including those in highly regulated industries.

[View blip history](#) >

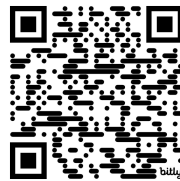
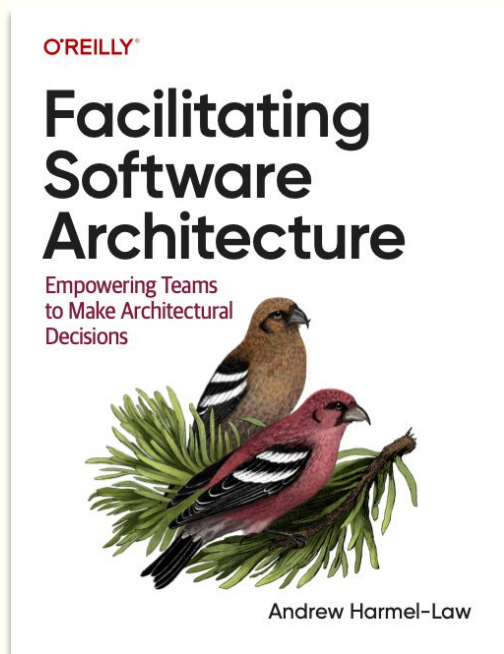
Related blips

Lightweight Architecture Decision Records
Adopt

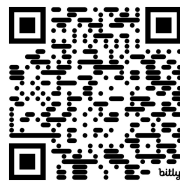
7. GraphRAG

In our last update of RAG, we introduced **GraphRAG**, originally described in [Microsoft's article](#) as a two-step approach: (1) Chunking documents and using an LLM-based analysis of the chunks to create a knowledge graph; (2) retrieving relevant chunks at query time via embeddings while following edges in the knowledge graph t... [Read more](#)

Thank you.



facilitatingsoftwarearchitecture.com



Free pass to O'Reilly.com
bit.ly/orly2025



Article on MartinFowler.com:
bit.ly/scaling-arch-mf



[@andrewhl.bsky.social](https://bsky.social/@andrewhl) #GOTOCph2025

RATE THIS SESSION

YES, YOU'LL GET A REWARD