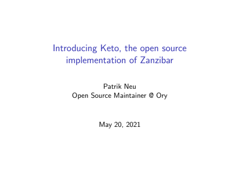
Introducing Keto, the open source implementation of Zanzibar

Patrik Neu
Open Source Maintainer @ Ory

May 20, 2021

2021-05-20

Introducing Keto, the open source implementation of Zanzibar



Introducing Keto, the open source implementation of Zanzibar

Patrik Neu
Open Source Maintainer @ Ory

May 20, 2021

1. breaking up hydra into separate services to not create yet another monolith trying to solve everything auth*
2. Keto first commit March 2018
3. Keto was build on open policy agent
4. from accumulating performance complains and our own experience we knew it was not a perfect fit
5. In 2019 at USENIX Google Research presented a paper about Google's internal authorization system, code-named Zanzibar.

Ory Design Philosophy

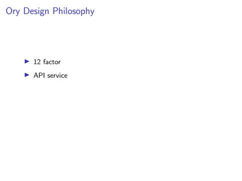
- 12 factor

Ory Design Philosophy

- 12 factor

Ory Design Philosophy

- 12 factor
- API service

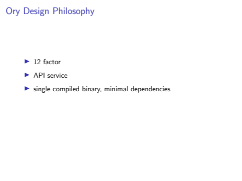


Ory Design Philosophy

- 12 factor
- API service

Ory Design Philosophy

- 12 factor
- API service
- single compiled binary, minimal dependencies

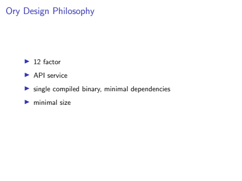


Ory Design Philosophy

- 12 factor
- API service
- single compiled binary, minimal dependencies

Ory Design Philosophy

- 12 factor
- API service
- single compiled binary, minimal dependencies
- minimal size

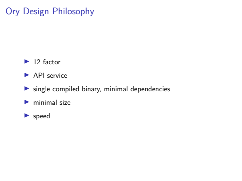


Ory Design Philosophy

- 12 factor
- API service
- single compiled binary, minimal dependencies
- minimal size

Ory Design Philosophy

- 12 factor
- API service
- single compiled binary, minimal dependencies
- minimal size
- speed


Ory Design Philosophy

- 12 factor
- API service
- single compiled binary, minimal dependencies
- minimal size
- speed

Requirements for Zanzibar

- flexible

Requirements for Zanzibar

- flexible

1. support all kinds of services, including Calendar, Cloud, Drive, Maps, Photos, and YouTube
2. different data models and permission requirements

Requirements for Zanzibar

- flexible
- fast
- always available

Requirements for Zanzibar

- flexible
- fast
- always available

1. authorization on critical path
2. required for each and every request
3. the best authorization system is never noticed by a regular user: don’t feel overhead, don’t experience errors
4. applications such as search require many authorization checks to serve one result

Requirements for Zanzibar

- flexible
- fast
- always available
- consistent

Requirements for Zanzibar

- flexible
- fast
- always available
- consistent

1. false positives: fatal, users do stuff they are not allowed
2. false negatives: at least annoying if time-bound, can be fatal if important tasks can not be done

Requirements for Zanzibar

- flexible
- fast
- always available
- consistent
- Google scale

Requirements for Zanzibar

- flexible
- fast
- always available
- consistent
- Google scale

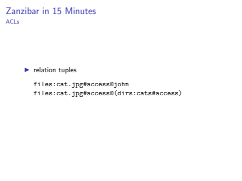
1. quote: "trillions of access control lists; millions of authorization requests per second"
2. distributed across the globe
3. cross-regional RTTs are already too high, handle locally

Zanzibar in 15 Minutes

ACLs

- relation tuples

```
files:cat.jpg#access@john
files:cat.jpg#access@(dirs:cats#access)
```



Zanzibar in 15 Minutes

ACLs

- relation tuples

files:cat.jpg#access@john
files:cat.jpg#access@(dirs:cats#access)

1. basic ACL structure
2. namespace:object#relation@subject
3. translates to "john has access on the cat.jpg file"
4. translates to "everyone who has access to the cats directory has access to the cat.jpg file"

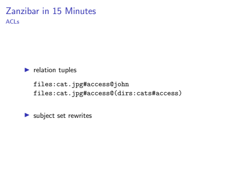
Zanzibar in 15 Minutes

ACLs

- relation tuples

  ```
  files:cat.jpg#access@john
  files:cat.jpg#access@(dirs:cats#access)
  ```

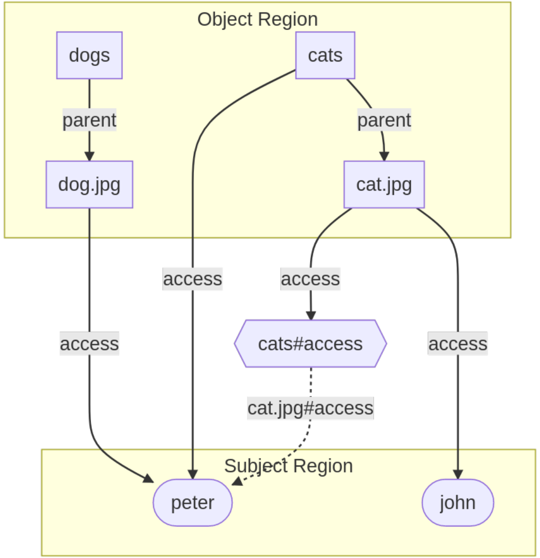
- subject set rewrites

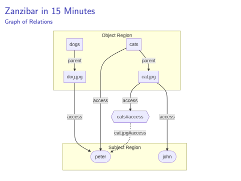

Zanzibar in 15 Minutes
ACLs
- relation tuples
files:cat.jpg#access@john
files:cat.jpg#access@(dirs:cats#access)
- subject set rewrites

1. defined globally in the namespace config
2. case 1: automatically add tuples; examples: read if you have write
3. case 2: compute effective set; examples: access child if access to parent, only access if you are admin AND got the explicit permission
4. not yet implemented in Keto, but the next big thing to work on as they are important

Zanzibar in 15 Minutes

Graph of Relations



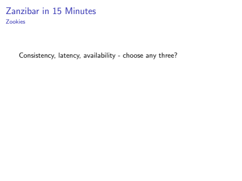


1. not only neat to look at
2. graph algorithms are well studied and common
3. ACL check ≡ reachability of node
4. Expand subject set by some graph traversal algorithm
5. relation tuples will usually result in a clustered/structured graph
6. relation tuples are directed edges

Zanzibar in 15 Minutes

Zookies

Consistency, latency, availability - choose any three?



Zanzibar in 15 Minutes
Zookies

Consistency, latency, availability - choose any three?

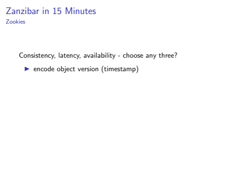
1. background in distributed systems
2. research and theorems show that requirements are in conflict
3. all can be meet at the same time once data are propagated
4. determine whether local data are recent enough

Zanzibar in 15 Minutes

Zookies

Consistency, latency, availability - choose any three?

- encode object version (timestamp)



Zanzibar in 15 Minutes
Zookies
Consistency, latency, availability - choose any three?
- encode object version (timestamp)

1. distributed systems: clock synchronisation is very hard
2. real-time easy for Google: GPS in datacenters to sync clocks

Zanzibar in 15 Minutes

Zookies

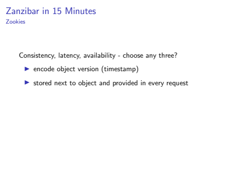
Consistency, latency, availability - choose any three?

- encode object version (timestamp)
- stored next to object and provided in every request

2021-05-20

Introducing Keto, the open source implementation of Zanzibar

└─Zanzibar in 15 Minutes



Zanzibar in 15 Minutes
Zookies

Consistency, latency, availability - choose any three?

- encode object version (timestamp)
- stored next to object and provided in every request

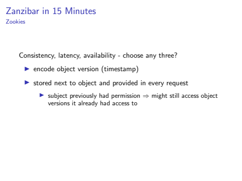
1. Local cache ACLs have to be at least as recent as object version

Zanzibar in 15 Minutes

Zookies

Consistency, latency, availability - choose any three?

- encode object version (timestamp)
- stored next to object and provided in every request
  - subject previously had permission ⇒ might still access object versions it already had access to



Zanzibar in 15 Minutes
Zookies

Consistency, latency, availability - choose any three?

- encode object version (timestamp)
- stored next to object and provided in every request
  - subject previously had permission ⇒ might still access object versions it already had access to

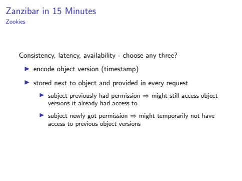
1. only during propagation time
2. new object versions will always be rejected

Zanzibar in 15 Minutes

Zookies

Consistency, latency, availability - choose any three?

- encode object version (timestamp)
- stored next to object and provided in every request
  - subject previously had permission ⇒ might still access object versions it already had access to
  - subject newly got permission ⇒ might temporarily not have access to previous object versions



Zanzibar in 15 Minutes
Zookies

Consistency, latency, availability - choose any three?

- encode object version (timestamp)
- stored next to object and provided in every request
  - subject previously had permission ⇒ might still access object versions it already had access to
  - subject newly got permission ⇒ might temporarily not have access to previous object versions

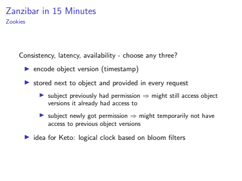
1. only during propagation time
2. new object versions will always be allowed

Zanzibar in 15 Minutes

Zookies

Consistency, latency, availability - choose any three?

- encode object version (timestamp)
- stored next to object and provided in every request
  - subject previously had permission ⇒ might still access object versions it already had access to
  - subject newly got permission ⇒ might temporarily not have access to previous object versions
- idea for Keto: logical clock based on bloom filters



Zanzibar in 15 Minutes
Zookies
Consistency, latency, availability - choose any three?
- encode object version (timestamp)
- stored next to object and provided in every request
  - subject previously had permission ⇒ might still access object versions it already had access to
  - subject newly got permission ⇒ might temporarily not have access to previous object versions
- idea for Keto: logical clock based on bloom filters

1. zookies not yet implemented (only single node operation)
2. bloom filter based to allow dynamic number of nodes
3. not settled, still searching for ideas

Current State of Keto

- single node operation mode (scaling horizontally possible)

Current State of Keto

- single node operation mode (scaling horizontally possible)

Current State of Keto

- single node operation mode (scaling horizontally possible)
- read, write, check, and expand APIs

Current State of Keto

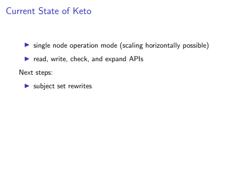
- single node operation mode (scaling horizontally possible)
- read, write, check, and expand APIs

Current State of Keto

- single node operation mode (scaling horizontally possible)
- read, write, check, and expand APIs

Next steps:

- subject set rewrites



Current State of Keto

- single node operation mode (scaling horizontally possible)
- read, write, check, and expand APIs

Next steps:

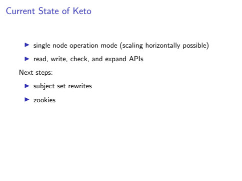
- subject set rewrites

Current State of Keto

- single node operation mode (scaling horizontally possible)
- read, write, check, and expand APIs

Next steps:

- subject set rewrites
- zookies



Current State of Keto

- single node operation mode (scaling horizontally possible)
- read, write, check, and expand APIs

Next steps:

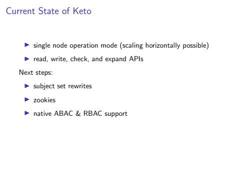
- subject set rewrites
- zookies

Current State of Keto

- single node operation mode (scaling horizontally possible)
- read, write, check, and expand APIs

Next steps:

- subject set rewrites
- zookies
- native ABAC & RBAC support



Current State of Keto

- single node operation mode (scaling horizontally possible)
- read, write, check, and expand APIs

Next steps:

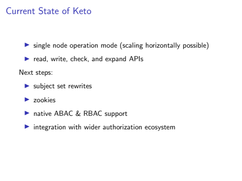
- subject set rewrites
- zookies
- native ABAC & RBAC support

Current State of Keto

- single node operation mode (scaling horizontally possible)
- read, write, check, and expand APIs

Next steps:

- subject set rewrites
- zookies
- native ABAC & RBAC support
- integration with wider authorization ecosystem



Current State of Keto

- single node operation mode (scaling horizontally possible)
- read, write, check, and expand APIs

Next steps:

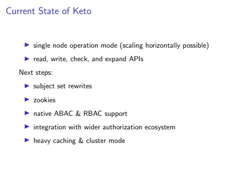
- subject set rewrites
- zookies
- native ABAC & RBAC support
- integration with wider authorization ecosystem

Current State of Keto

- single node operation mode (scaling horizontally possible)
- read, write, check, and expand APIs

Next steps:

- subject set rewrites
- zookies
- native ABAC & RBAC support
- integration with wider authorization ecosystem
- heavy caching & cluster mode



Current State of Keto

- single node operation mode (scaling horizontally possible)
- read, write, check, and expand APIs

Next steps:

- subject set rewrites
- zookies
- native ABAC & RBAC support
- integration with wider authorization ecosystem
- heavy caching & cluster mode

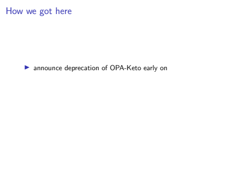
How we got here

- announce deprecation of OPA-Keto early on

2021-05-20

Introducing Keto, the open source implementation of Zanzibar

└How we got here



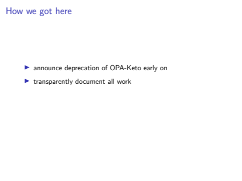
How we got here

- announce deprecation of OPA-Keto early on

1. multiple channels
2. no migration path yet

How we got here

- announce deprecation of OPA-Keto early on
- transparently document all work

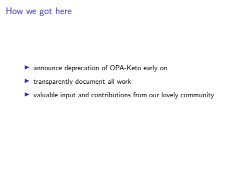

How we got here

- announce deprecation of OPA-Keto early on
- transparently document all work

instead of developing in the dark and suddenly pushing the new version

How we got here

- announce deprecation of OPA-Keto early on
- transparently document all work
- valuable input and contributions from our lovely community



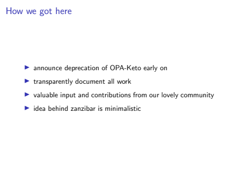
How we got here

- announce deprecation of OPA-Keto early on
- transparently document all work
- valuable input and contributions from our lovely community

1. although git shows that I did most work
2. ideas were always discussed with multiple people
3. community members followed me into the rabbit hole
4. jumped on calls to discuss details, ideas and findings

How we got here

- announce deprecation of OPA-Keto early on
- transparently document all work
- valuable input and contributions from our lovely community
- idea behind zanzibar is minimalistic



How we got here

- announce deprecation of OPA-Keto early on
- transparently document all work
- valuable input and contributions from our lovely community
- idea behind zanzibar is minimalistic

1. check engine currently 39 LoC

Open Source Foundation

- Go
- gRPC
- OpenAPI Spec
- gobuffalo/pop
- Cobra
- Docusaurus
- Docker

2021-05-20

Introducing Keto, the open source implementation of Zanzibar

└─Open Source Foundation

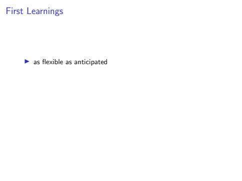
Open Source Foundation

- Go
- gRPC
- OpenAPI Spec
- gobuffalo/pop
- Cobra
- Docusaurus
- Docker

1. like our other open source projects
2.

First Learnings

- as flexible as anticipated



First Learnings

- as flexible as anticipated

1. from our own SaaS cloud production system
2. from community feedback

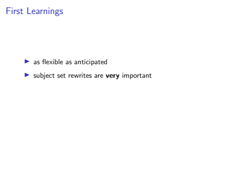
First Learnings

- as flexible as anticipated
- subject set rewrites are **very** important

2021-05-20

Introducing Keto, the open source implementation of Zanzibar

└First Learnings



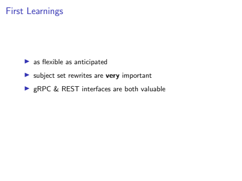
First Learnings

- as flexible as anticipated
- subject set rewrites are **very** important

1. from our own SaaS cloud production system
2. from community feedback

First Learnings

- as flexible as anticipated
- subject set rewrites are **very** important
- gRPC & REST interfaces are both valuable



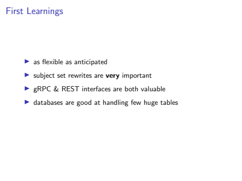
First Learnings

- as flexible as anticipated
- subject set rewrites are **very** important
- gRPC & REST interfaces are both valuable

1. from our own SaaS cloud production system
2. from community feedback

First Learnings

- as flexible as anticipated
- subject set rewrites are **very** important
- gRPC & REST interfaces are both valuable
- databases are good at handling few huge tables



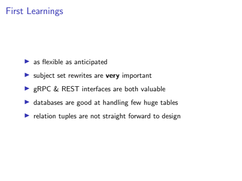
First Learnings

- as flexible as anticipated
- subject set rewrites are **very** important
- gRPC & REST interfaces are both valuable
- databases are good at handling few huge tables

1. from our own SaaS cloud production system
2. from community feedback

First Learnings

- as flexible as anticipated
- subject set rewrites are **very** important
- gRPC & REST interfaces are both valuable
- databases are good at handling few huge tables
- relation tuples are not straight forward to design



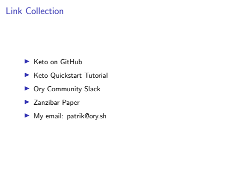
First Learnings

- as flexible as anticipated
- subject set rewrites are **very** important
- gRPC & REST interfaces are both valuable
- databases are good at handling few huge tables
- relation tuples are not straight forward to design

1. from our own SaaS cloud production system
2. from community feedback

Link Collection

- Keto on GitHub
- Keto Quickstart Tutorial
- Ory Community Slack
- Zanzibar Paper
- My email: patrik@ory.sh



Link Collection

- Keto on GitHub
- Keto Quickstart Tutorial
- Ory Community Slack
- Zanzibar Paper
- My email: patrik@ory.sh