Trino & OPA @ Stackable

Sönke Liebau
& Sebastian Bernauer

Stackable

Agenda

1. What is Stackable?
2. Open Policy Agent (OPA) authorization plugin
   - History
   - Recent development
   - Compatibility layer to Trino’s File-based access control
   - Quick demo on row filtering and column masking
3. Auto-scale Trino clusters using trino-lb
   - Differences between trino-gateway and trino-lb
4. Lessons learned running Trino on Kubernetes
   - What our trino-operator is doing
   - Potential next steps

About us

Sönke Liebau

CPO Stackable

Co-Founder of Stackable, working with Big Data Open Source Software since 2012, speaker, contributor, husband & father…

Sebastian Bernauer

Software developer

Working with Big Data Open Source Software since 2019
Big Open Source and Trino fan :)

Stackable

Stackable in a Nutshell

Founded

2020

OpenCore

b.telligent

IONOS

Stackable Data Platform

- Open Source
- Infrastructure as Code
- Cloud-native (Kubernetes)
- On-Premises, Cloud, Hybrid

Our Team: 20 People

International
in Germany & Europe

Our Services

- Product Support
- Big Data Consulting
- Trainings

Network - Collaborations

OSB Open Source Business ALLIANCE

KI BUNDESVERBAND

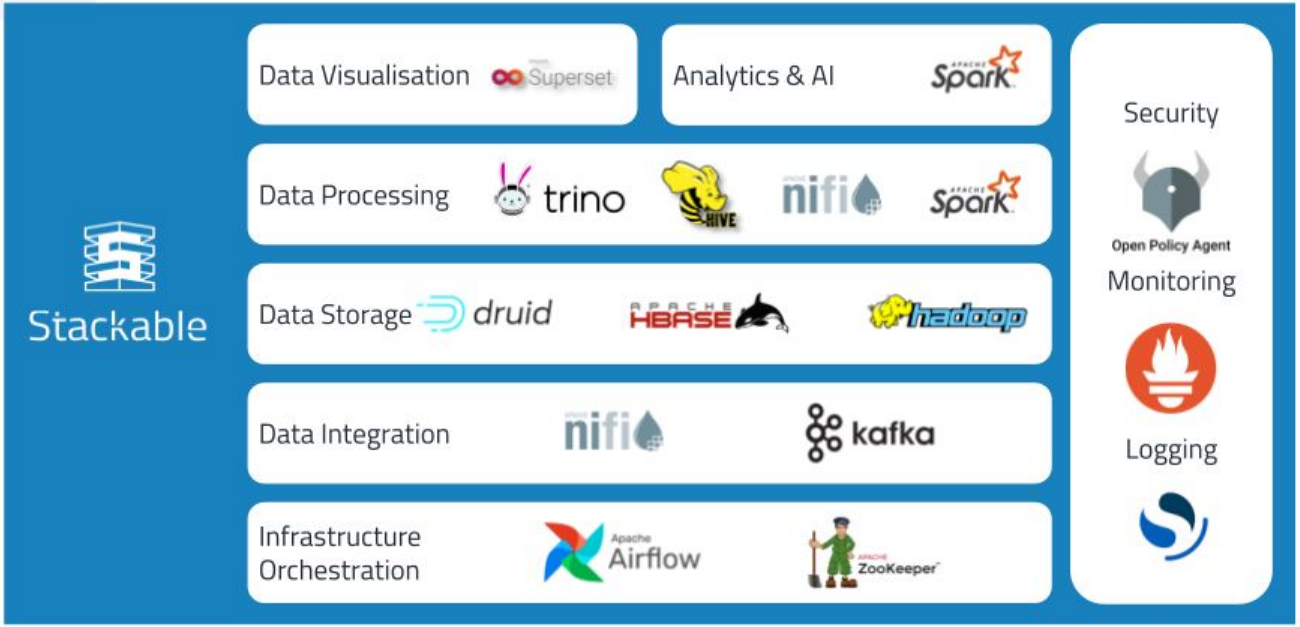
gaia-x

bitkom

eco

Stackable

Popular Data Apps. Kubernetes-native. Easy to deploy and operate.



Stackable

Data Visualisation

Analytics & AI

Data Processing

Data Storage

Data Integration

Infrastructure Orchestration

Security

Monitoring

Logging

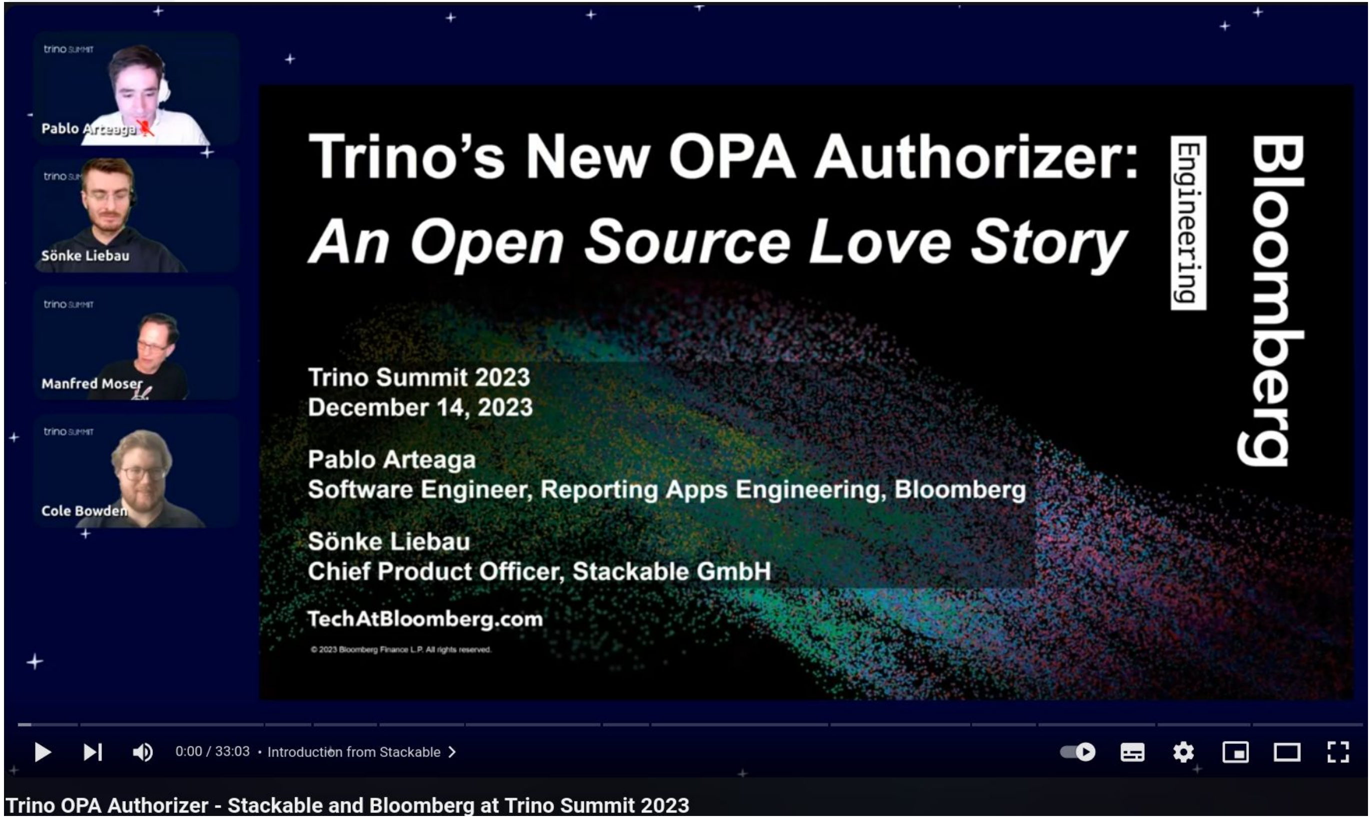
100% Open Source

modular and flexible

In every Cloud and in your own Data Center

as Managed Service in IONOS Cloud

OPA plugin - History



Trino's New OPA Authorizer:

An Open Source Love Story

Trino Summit 2023
December 14, 2023

Pablo Arteaga
Software Engineer, Reporting Apps Engineering, Bloomberg

Sönke Liebau
Chief Product Officer, Stackable GmbH

TechAtBloomberg.com

© 2023 Bloomberg Finance L.P. All rights reserved.

Trino OPA Authorizer - Stackable and Bloomberg at Trino Summit 2023

https://www.youtube.com/watch?v=fbqqapQbAv0

Stackable

OPA plugin - History

History

1. 2021/10: Stackable creates the stackabletech/trino-opa-authorize repo
2. 2023/02: After Bloomberg reached out license was changed to ASL2
3. 2023/05: Bloomberg created Trino PR upstream with much improved version
4. 2024/01: OPA plugin was merged into Trino and released with version 438 🚀

Recent development

5. 2024/07: Bloomberg improve the performance of column masks by batching requests send to OPA, released in 453 🚀
   https://github.com/trinodb/trino/pull/21997

Compatibility layer to Trino’s File-based access control

- Trino already offers a great and flexible access control

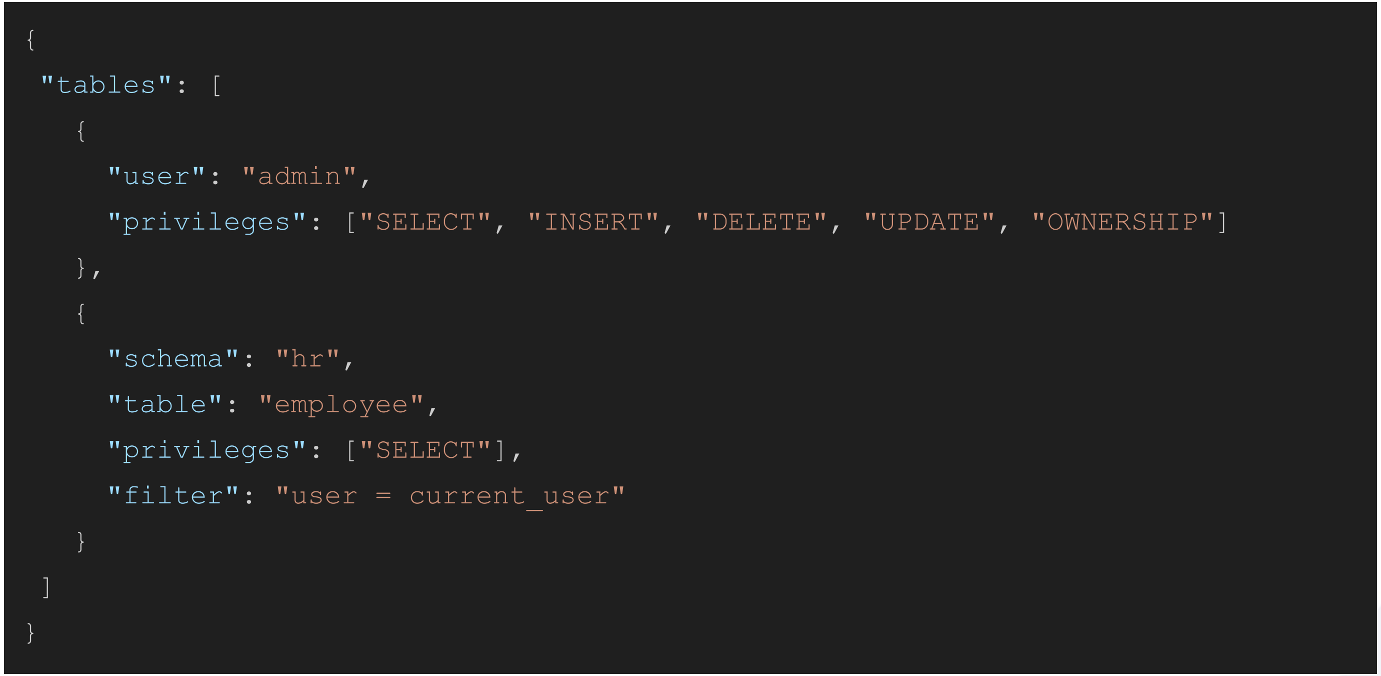
```
access-control.name=file
security.config-file=etc/rules.json
```

- We want users to be able to migrate to OPA as easy as possible

→ Compatibility layer written in rego, which takes the same JSON definition as input and emulates the Trino behaviour
→ Can server as a starting point

https://github.com/stackabletech/trino-operator/tree/main/tests/templates/kuttl/opa-authorization/trino_rules

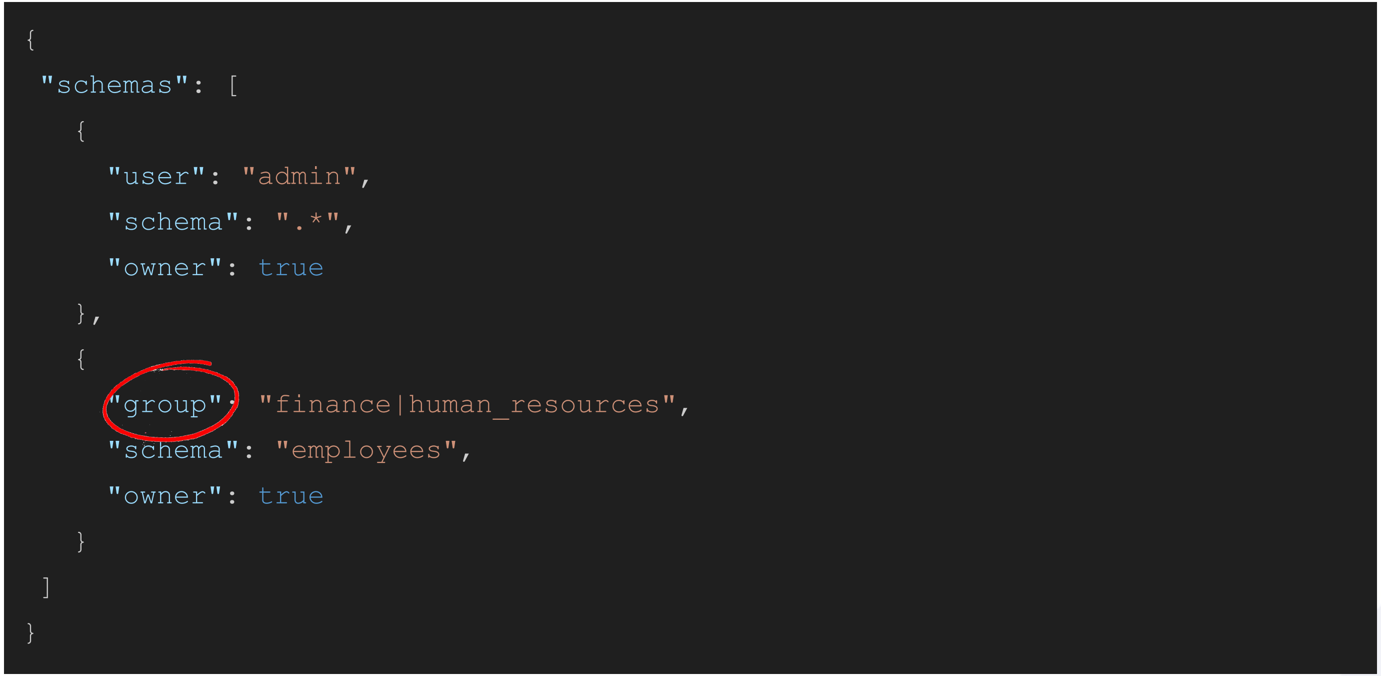
Compatibility layer



```
{
 "tables": [
   {
     "user": "admin",
     "privileges": ["SELECT", "INSERT", "DELETE", "UPDATE", "OWNERSHIP"]
   },
   {
     "schema": "hr",
     "table": "employee",
     "privileges": ["SELECT"],
     "filter": "user = current_user"
   }
 ]
}
```

https://trino.io/docs/current/security/file-system-access-control.html

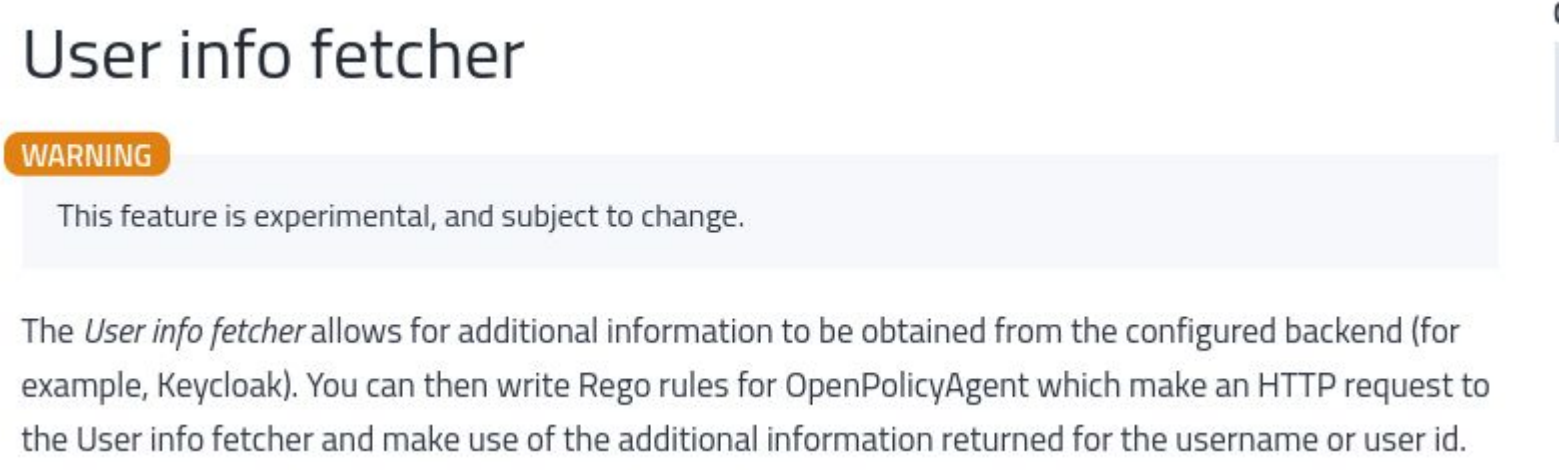
Compatibility layer



```
{
 "schemas": [
   {
     "user": "admin",
     "schema": ".*",
     "owner": true
   },
   {
     "group": "finance|human_resources",
     "schema": "employees",
     "owner": true
   }
 ]
}
```

https://trino.io/docs/current/security/file-system-access-control.html

Userinfo Fetcher



User info fetcher

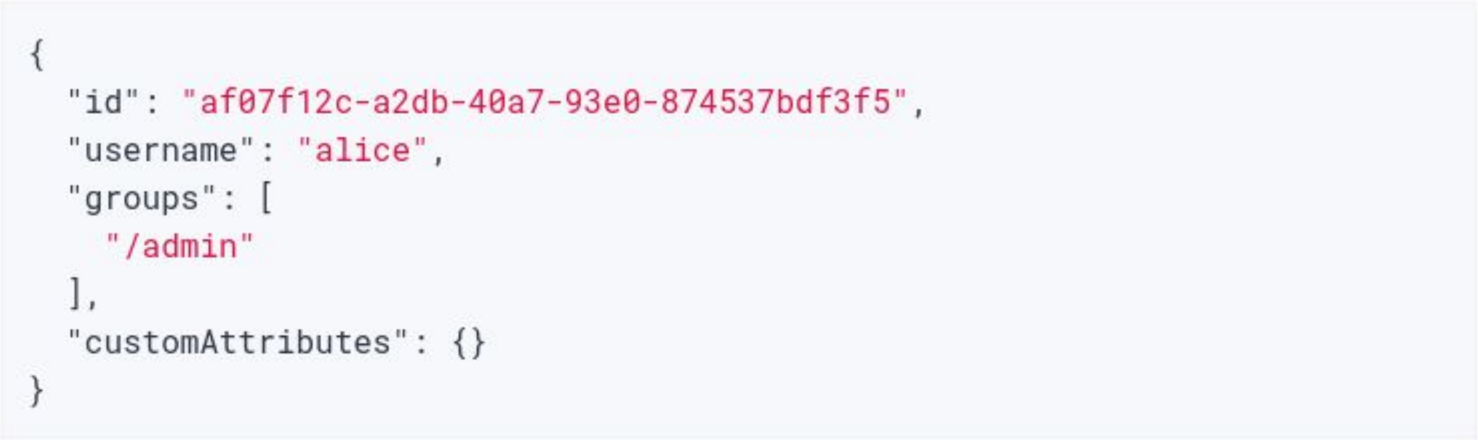
WARNING

This feature is experimental, and subject to change.

The *User info fetcher* allows for additional information to be obtained from the configured backend (for example, Keycloak). You can then write Rego rules for OpenPolicyAgent which make an HTTP request to the User info fetcher and make use of the additional information returned for the username or user id.

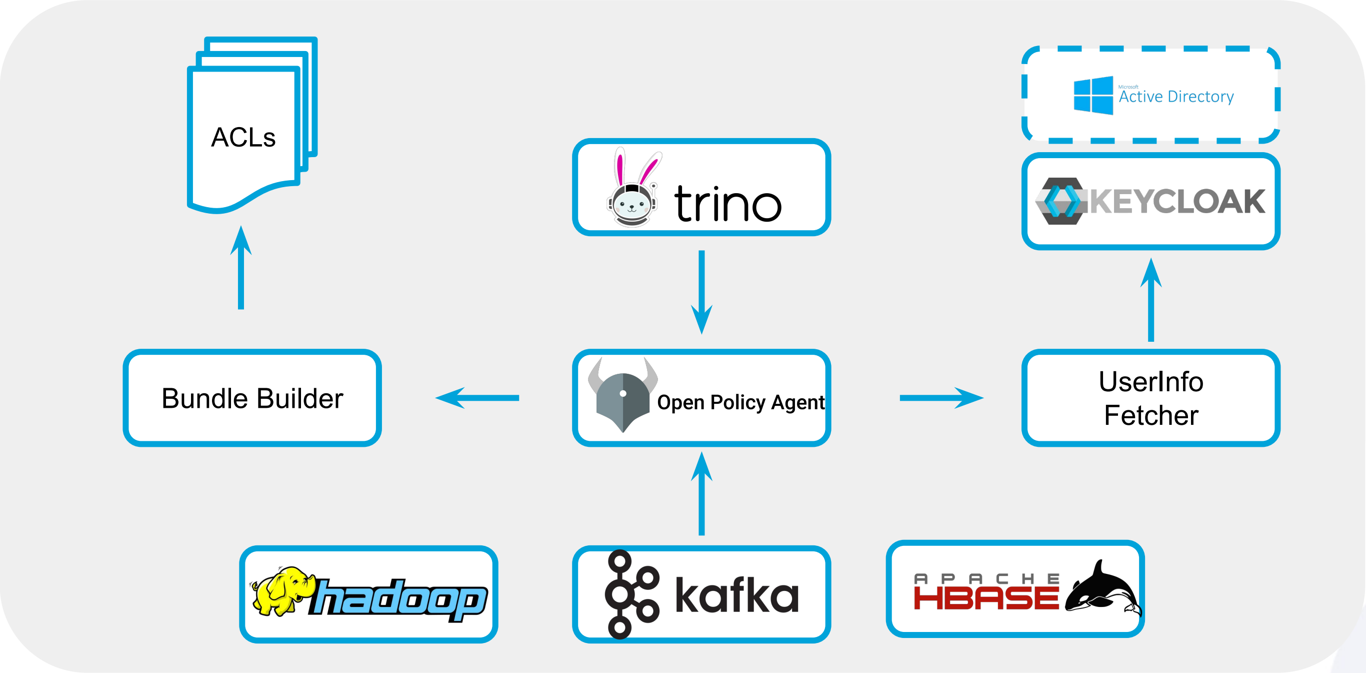
Stackable

Userinfo Fetcher



```
{
  "id": "af07f12c-a2db-40a7-93e0-874537bdf3f5",
  "username": "alice",
  "groups": [
    "/admin"
  ],
  "customAttributes": {}
}
```

The Big Picture



Stackable

Quick demo on row & column level security

Live-Demo: May 15th, 12 p.m. – 1 p.m.

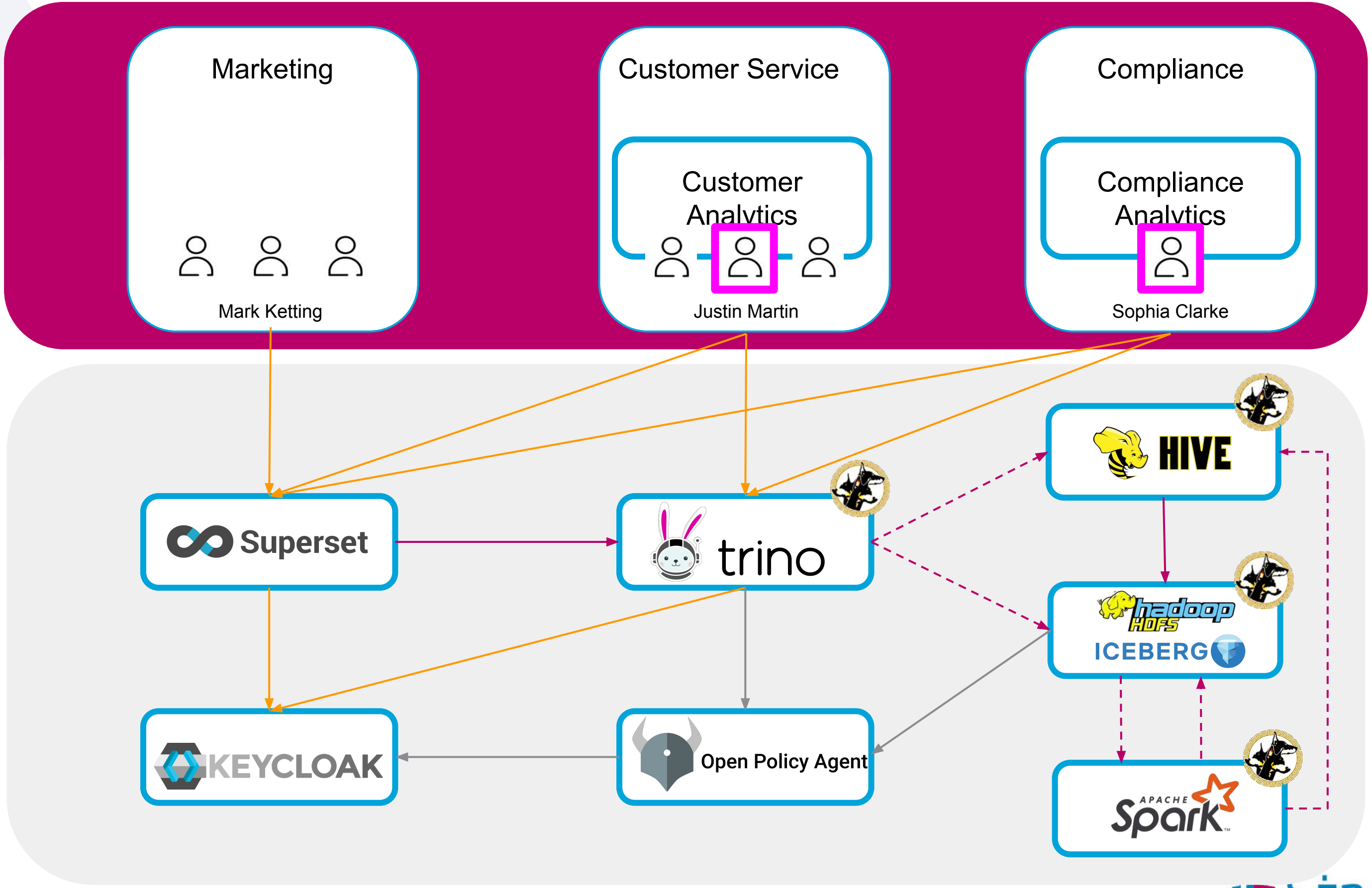
Stackable TechTalk

Mastering Data Platform Security

Stackable TechTalk - Mastering Data Platform Security

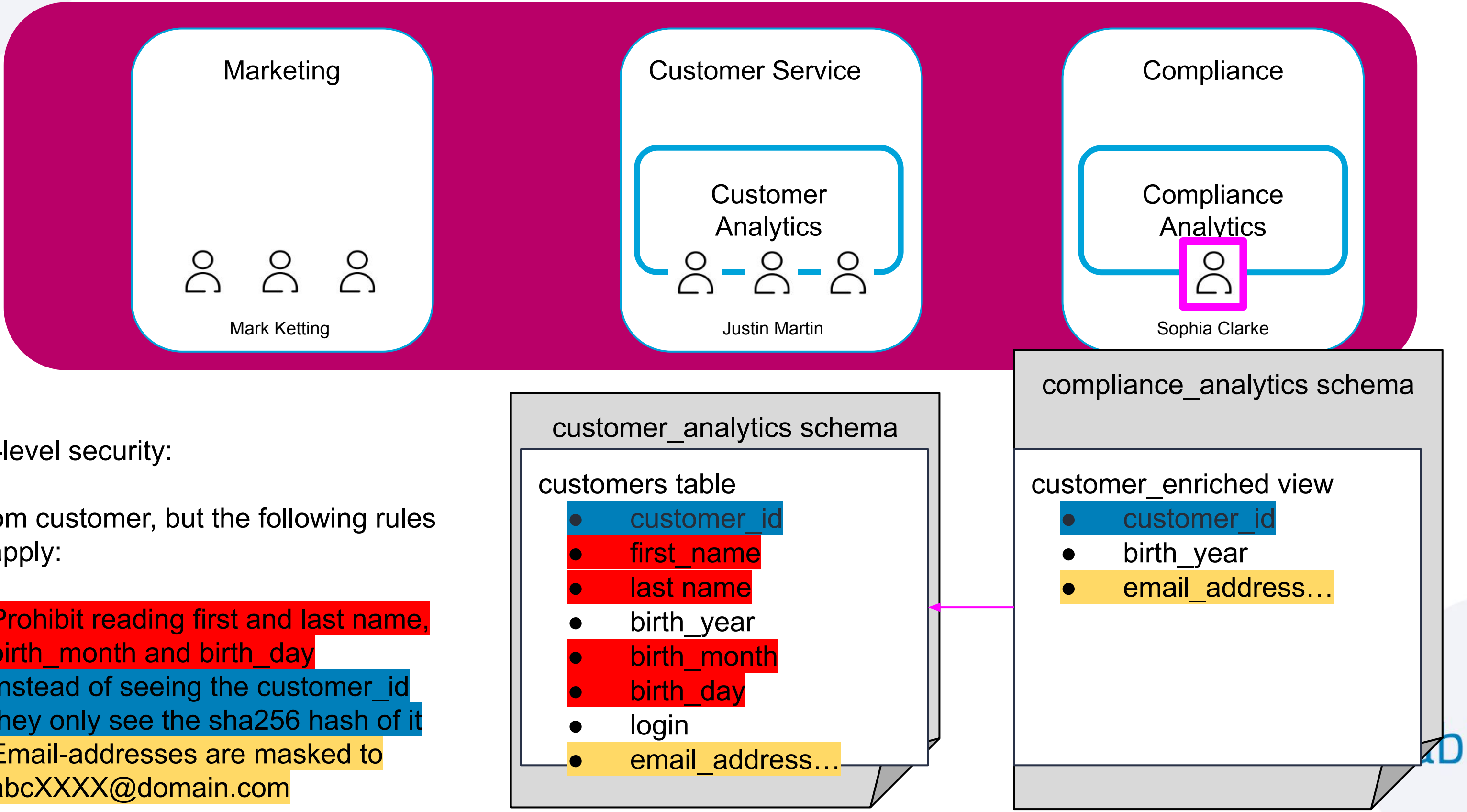
https://www.youtube.com/watch?v=ATIq_I3WNiA

Stackable



Kerberos secured → OpenID connect → Authorization → Impersonation - - ► No impersonation

Column Level Security



Marketing

Mark Ketting

Customer Service

Customer Analytics

Justin Martin

Compliance

Compliance Analytics

Sophia Clarke

customer_analytics schema

customers table

- customer_id
- first_name
- last name
- birth_year
- birth_month
- birth_day
- login
- email_address…

compliance_analytics schema

customer_enriched view

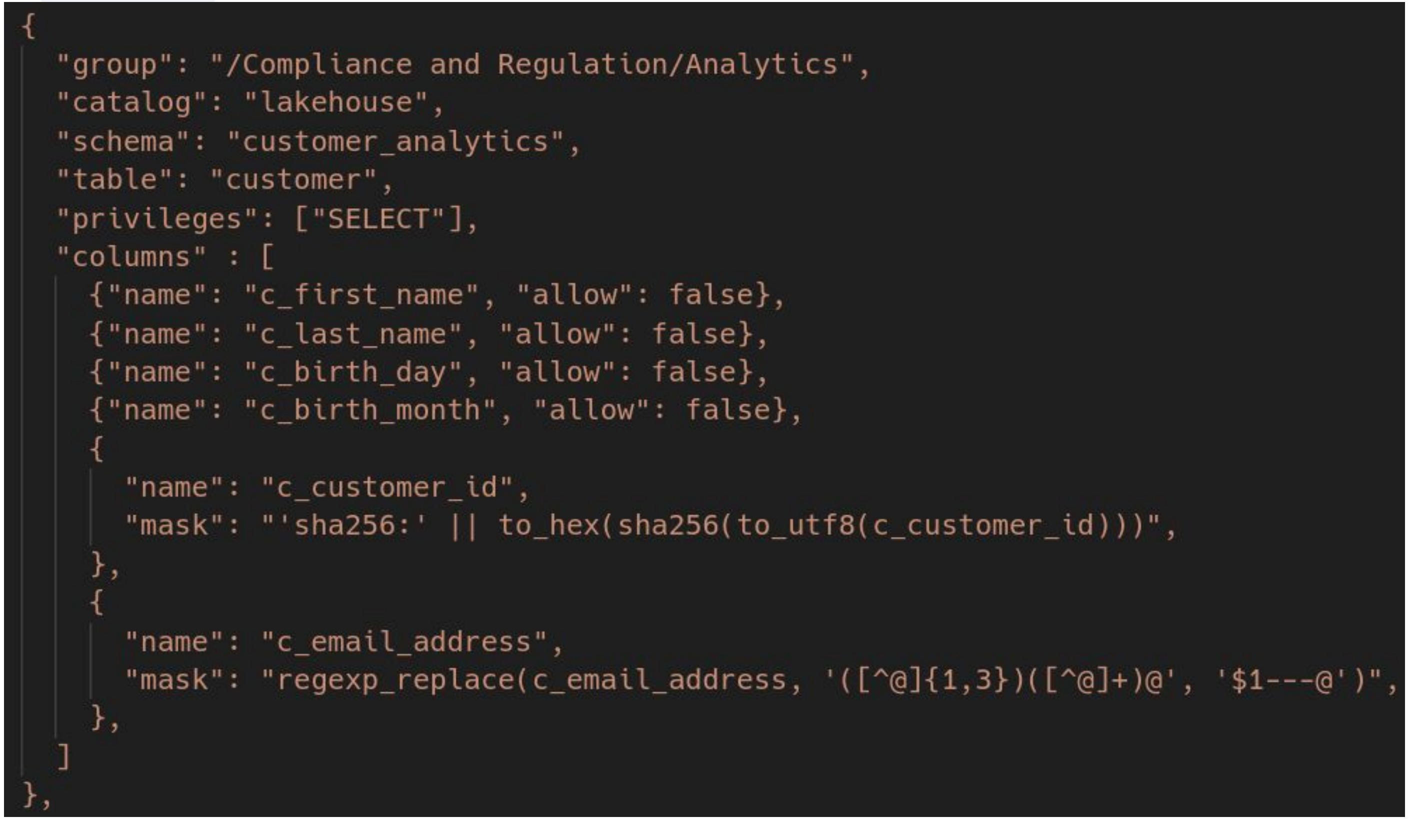
- customer_id
- birth_year
- email_address…

Column-level security:

Read from customer, but the following rules should apply:

1. Prohibit reading first and last name, birth_month and birth_day
2. Instead of seeing the customer_id they only see the sha256 hash of it
3. Email-addresses are masked to abcXXXX@domain.com

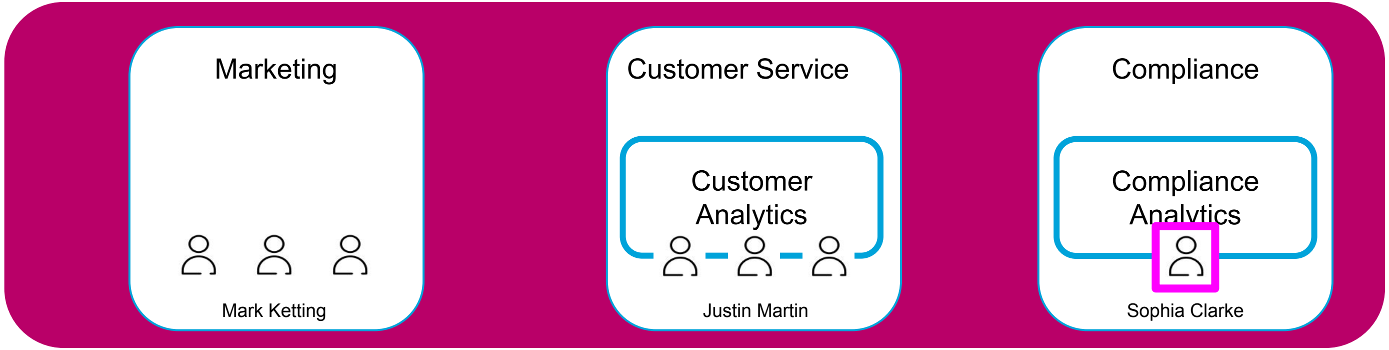
How does it look in code?



```
{
  "group": "/Compliance and Regulation/Analytics",
  "catalog": "lakehouse",
  "schema": "customer_analytics",
  "table": "customer",
  "privileges": ["SELECT"],
  "columns" : [
    {"name": "c_first_name", "allow": false},
    {"name": "c_last_name", "allow": false},
    {"name": "c_birth_day", "allow": false},
    {"name": "c_birth_month", "allow": false},
    {
      "name": "c_customer_id",
      "mask": "'sha256:' || to_hex(sha256(to_utf8(c_customer_id)))",
    },
    {
      "name": "c_email_address",
      "mask": "regexp_replace(c_email_address, '([^@]{1,3})([^@]+)@', '$1---@')",
    },
  ]
},
```

Stackable

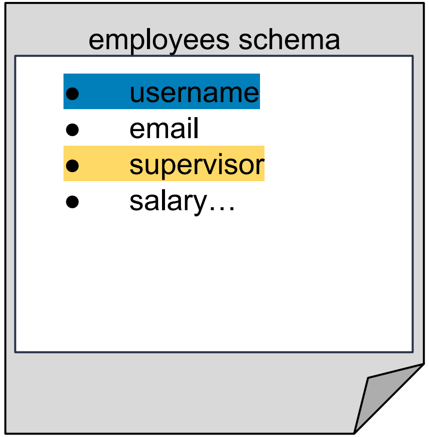
Row Level Security



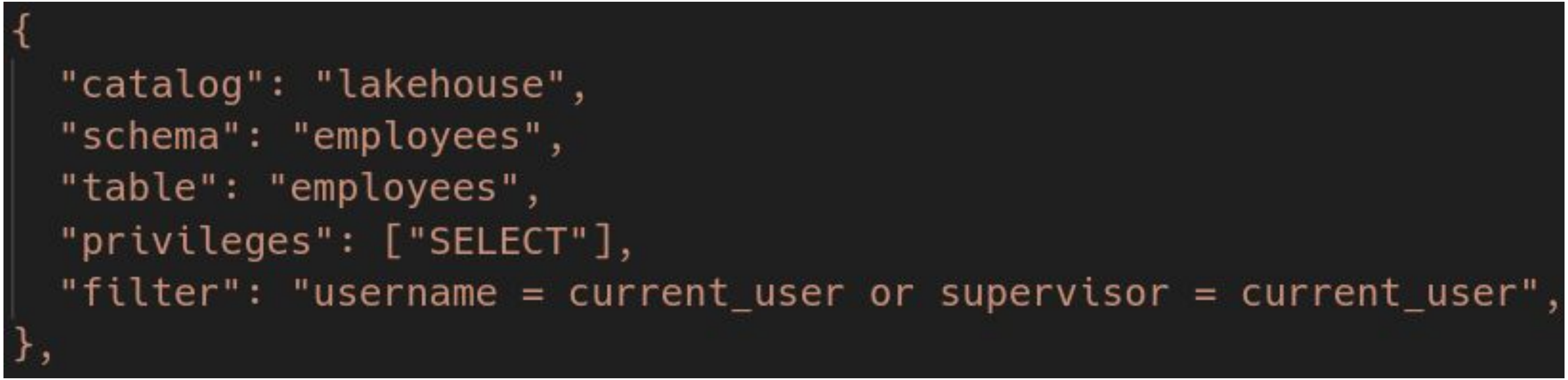
Row-level security:

Read from employees, but the following rules should apply:

1. Everyone can only see themselves
2. Supervisor additionally see their reports



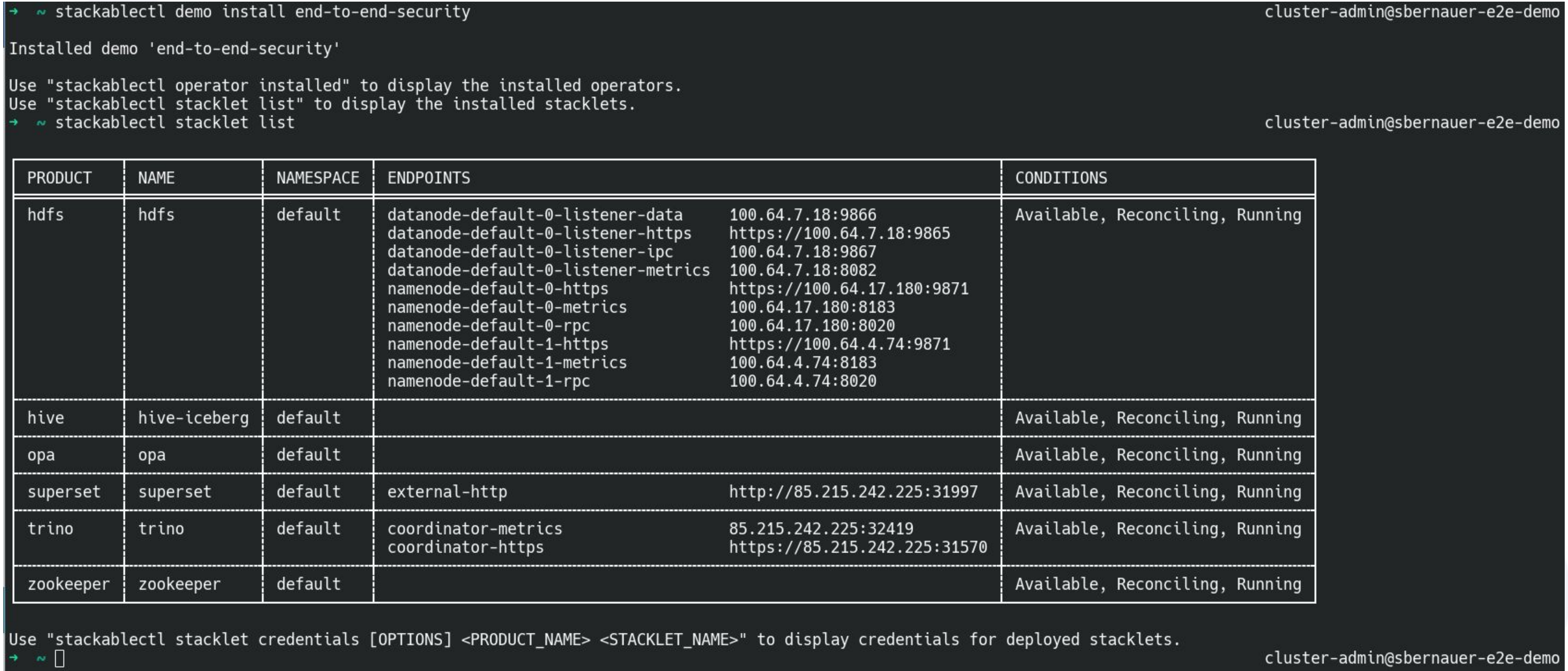
How does it look in code?



```
{
  "catalog": "lakehouse",
  "schema": "employees",
  "table": "employees",
  "privileges": ["SELECT"],
  "filter": "username = current_user or supervisor = current_user",
},
```

Demo time

Quick demo on row & column level security



```
➜ ~ stackablectl demo install end-to-end-security                                                     cluster-admin@sbernauer-e2e-demo

Installed demo 'end-to-end-security'

Use "stackablectl operator installed" to display the installed operators.
Use "stackablectl stacklet list" to display the installed stacklets.
➜ ~ stackablectl stacklet list                                                                        cluster-admin@sbernauer-e2e-demo

┌───────────┬─────────────┬───────────┬─────────────────────────────────────────────────────────────────┬──────────────────────────────────┐
│ PRODUCT   ┆ NAME        ┆ NAMESPACE ┆ ENDPOINTS                                                       ┆ CONDITIONS                       │
╞═══════════╪═════════════╪═══════════╪═════════════════════════════════════════════════════════════════╪══════════════════════════════════╡
│ hdfs      ┆ hdfs        ┆ default   ┆ datanode-default-0-listener-data     100.64.7.18:9866           ┆ Available, Reconciling, Running  │
│           ┆             ┆           ┆ datanode-default-0-listener-https    https://100.64.7.18:9865   ┆                                  │
│           ┆             ┆           ┆ datanode-default-0-listener-ipc      100.64.7.18:9867           ┆                                  │
│           ┆             ┆           ┆ datanode-default-0-listener-metrics  100.64.7.18:8082           ┆                                  │
│           ┆             ┆           ┆ namenode-default-0-https             https://100.64.17.180:9871 ┆                                  │
│           ┆             ┆           ┆ namenode-default-0-metrics           100.64.17.180:8183         ┆                                  │
│           ┆             ┆           ┆ namenode-default-0-rpc               100.64.17.180:8020         ┆                                  │
│           ┆             ┆           ┆ namenode-default-1-https             https://100.64.4.74:9871   ┆                                  │
│           ┆             ┆           ┆ namenode-default-1-metrics           100.64.4.74:8183           ┆                                  │
│           ┆             ┆           ┆ namenode-default-1-rpc               100.64.4.74:8020           ┆                                  │
├╌╌╌╌╌╌╌╌╌╌╌┼╌╌╌╌╌╌╌╌╌╌╌╌╌┼╌╌╌╌╌╌╌╌╌╌╌┼╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌┼╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌┤
│ hive      ┆ hive-iceberg┆ default   ┆                                                                 ┆ Available, Reconciling, Running  │
├╌╌╌╌╌╌╌╌╌╌╌┼╌╌╌╌╌╌╌╌╌╌╌╌╌┼╌╌╌╌╌╌╌╌╌╌╌┼╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌┼╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌┤
│ opa       ┆ opa         ┆ default   ┆                                                                 ┆ Available, Reconciling, Running  │
├╌╌╌╌╌╌╌╌╌╌╌┼╌╌╌╌╌╌╌╌╌╌╌╌╌┼╌╌╌╌╌╌╌╌╌╌╌┼╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌┼╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌┤
│ superset  ┆ superset    ┆ default   ┆ external-http                        http://85.215.242.225:31997┆ Available, Reconciling, Running  │
├╌╌╌╌╌╌╌╌╌╌╌┼╌╌╌╌╌╌╌╌╌╌╌╌╌┼╌╌╌╌╌╌╌╌╌╌╌┼╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌┼╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌┤
│ trino     ┆ trino       ┆ default   ┆ coordinator-metrics                  85.215.242.225:32419       ┆ Available, Reconciling, Running  │
│           ┆             ┆           ┆ coordinator-https                    https://85.215.242.225:31570┆                                 │
├╌╌╌╌╌╌╌╌╌╌╌┼╌╌╌╌╌╌╌╌╌╌╌╌╌┼╌╌╌╌╌╌╌╌╌╌╌┼╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌┼╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌┤
│ zookeeper ┆ zookeeper   ┆ default   ┆                                                                 ┆ Available, Reconciling, Running  │
└───────────┴─────────────┴───────────┴─────────────────────────────────────────────────────────────────┴──────────────────────────────────┘

Use "stackablectl stacklet credentials [OPTIONS] <PRODUCT_NAME> <STACKLET_NAME>" to display credentials for deployed stacklets.
➜ ~ ▯                                                                                                 cluster-admin@sbernauer-e2e-demo
```

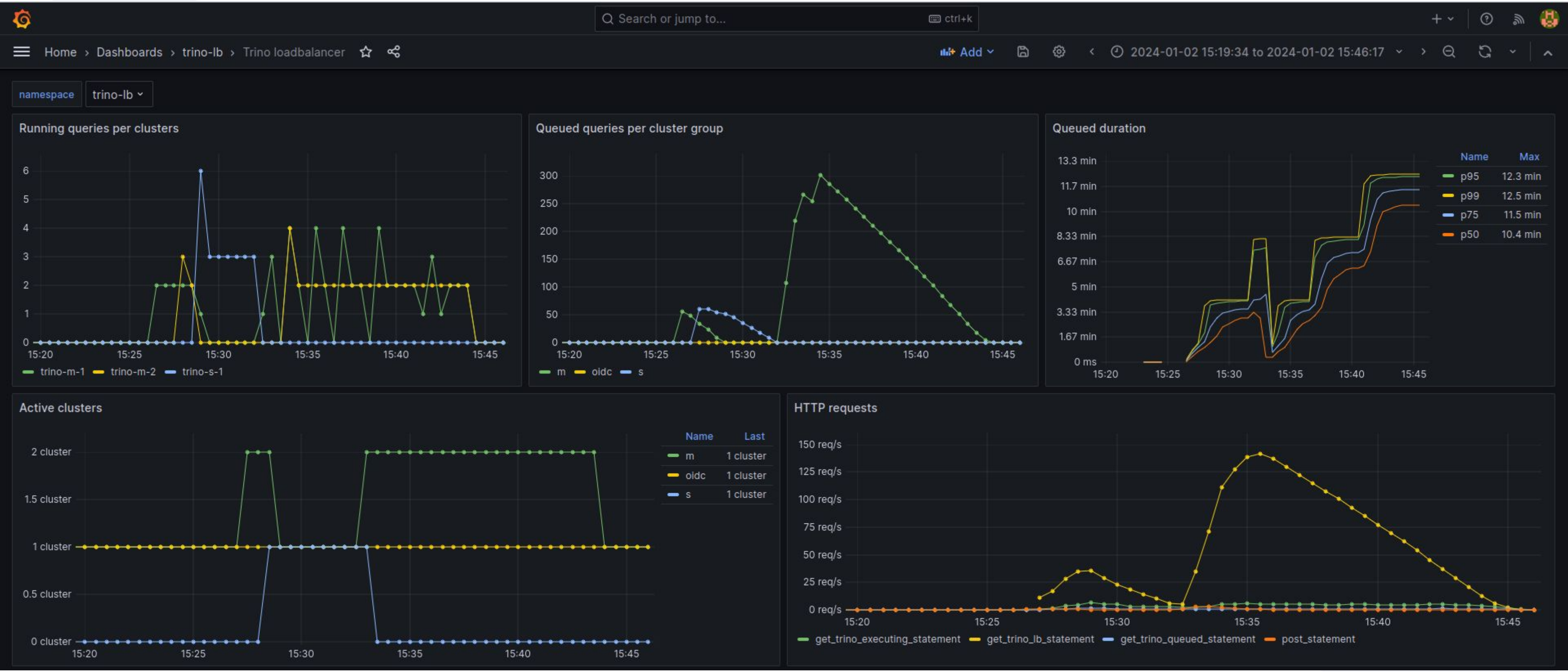
Auto-scale Trino clusters using trino-lb

- trino-lb development started around 2023/10, just before trino-gateway was first released
- The primary goals are
    - Queuing of queries in case all available Trino clusters are already full
    - Auto-scaling of entire Trino clusters (load and time based)
    - Performance (trino-lb is horizontally scalable)
    - High availability (trino-lb is stateless)
    - Very flexible routing strategies (e.g. Python script)
    - Modularity to supported different persistence, routing and scaling implementations

https://github.com/stackabletech/trino-lb

Stackable

Auto-scale Trino clusters using trino-lb



Search or jump to... ctrl+k

Home › Dashboards › trino-lb › Trino loadbalancer

Add

2024-01-02 15:19:34 to 2024-01-02 15:46:17

namespace: trino-lb

Running queries per clusters

Queued queries per cluster group

Queued duration

Name	Max
p95	12.3 min
p99	12.5 min
p75	11.5 min
p50	10.4 min

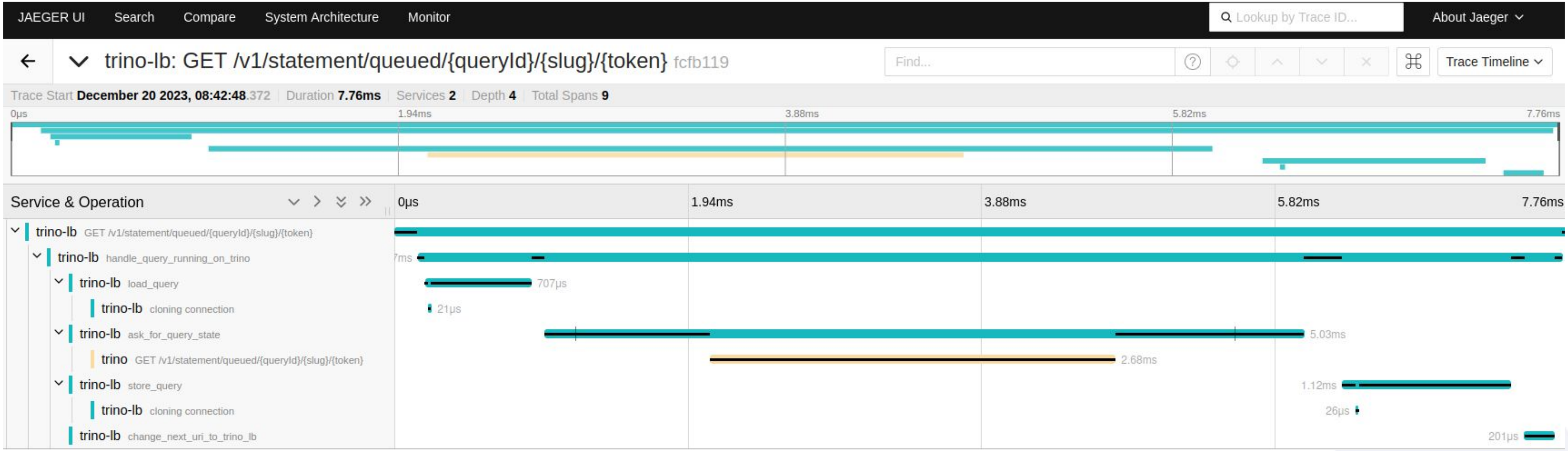
Active clusters

Name	Last
m	1 cluster
oidc	1 cluster
s	1 cluster

HTTP requests

Auto-scale Trino clusters using trino-lb

- OpenTelemetry tracing
  - Trace propagation to Trino



trino-lb: GET /v1/statement/queued/{queryId}/{slug}/{token} fcfb119

Trace Start **December 20 2023, 08:42:48**.372 Duration **7.76ms** Services **2** Depth **4** Total Spans **9**

Service	Operation
trino-lb	GET /v1/statement/queued/{queryId}/{slug}/{token}
trino-lb	handle_query_running_on_trino
trino-lb	load_query
trino-lb	cloning connection
trino-lb	ask_for_query_state
trino	GET /v1/statement/queued/{queryId}/{slug}/{token}
trino-lb	store_query
trino-lb	cloning connection
trino-lb	change_next_uri_to_trino_lb

Stackable

Lessons learned running Trino on Kubernetes

- First off: We don’t run any production Trino on Kubernetes

- But our customers do :)

- We have written an operator to manage Trino on Kubernetes: https://github.com/stackabletech/trino-operator
- Documentation: https://docs.stackable.tech/home/stable/trino/

Stackable

Try to avoid coordinators restarts

- A coordinator restart kills all running queries
- Mitigation:
  - We have a flag: Don’t touch this cluster at all costs!
- Potential future work:
  - Trino HA [#391]? :)
  - Maintenance windows
  - Graceful shutdown of coordinator
    i. Remove coordinator from trino-lb/trino-gateway
    ii. Wait till all queries finished
    iii. Restart
    iv. Add coordinator to trino-lb/trino-gateway again
    v. Requires Kubernetes nodes to wait long enough while draining!

Graceful shutdown of workers

- A worker restart kills all running queries (without fault tolerant execution)
- Mitigation:
  - Graceful shutdown of workers
    1. Requires Kubernetes nodes to wait long enough while draining!
    2. We also set query.max-execution-time
  - Fault tolerant execution :)

Pod placement

- Avoid too many workers being down at the same time
- Mitigation:
  - By default we spread all workers across as many nodes as possible
    1. Can be customized by customer based on their topology
    2. Avoid impact of node/rack/room/datacenter failures
    3. Assumption: Big worker nodes to reduce internal Trino traffic
  - PodDisruptionBudets: Only X nodes can be down simultaneously