AI Toys in the Attic: Our favorite AI tools and techniques for ClickHouse®

Boris Tyshkevich - AI Architect
Robert Hodges - CEO
3 June 2026

Run Open Source ClickHouse® Better

Altinity.Cloud Enterprise Support

ClickHouse® is a very popular real-time analytic database

Understands SQL

Runs on bare metal to cloud

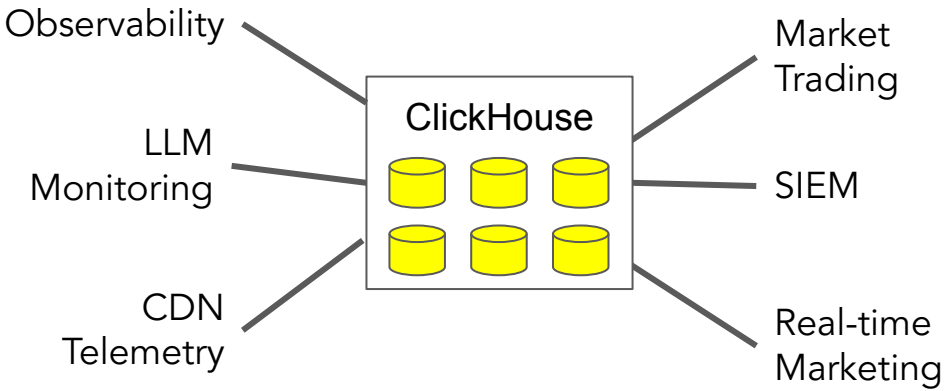
Shared nothing architecture

Stores data in columns

Parallel and vectorized execution

Scales to many petabytes

Is Open source (Apache 2.0)



47.8k GitHub Watchers
Can't Be Wrong!

Why combine ClickHouse and LLMs?

LLMs can be extraordinarily useful extensions to human engineering ability

Deep understanding of ClickHouse —> Ask questions about any aspect of ClickHouse data or applications

Fast iteration —> Speed up delivery by avoiding manual redesign of SQL, charts, or code

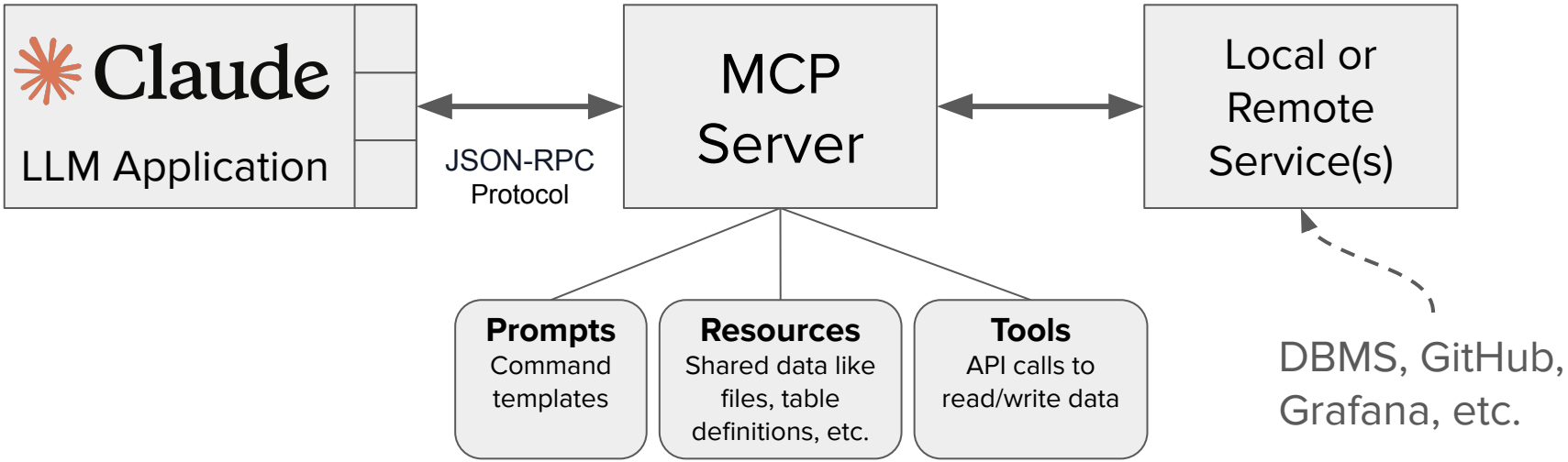
Design and Operation —> Evaluate design of new applications and health of existing systems

But...Using LLMs on production data is a work in progress

MCP Servers

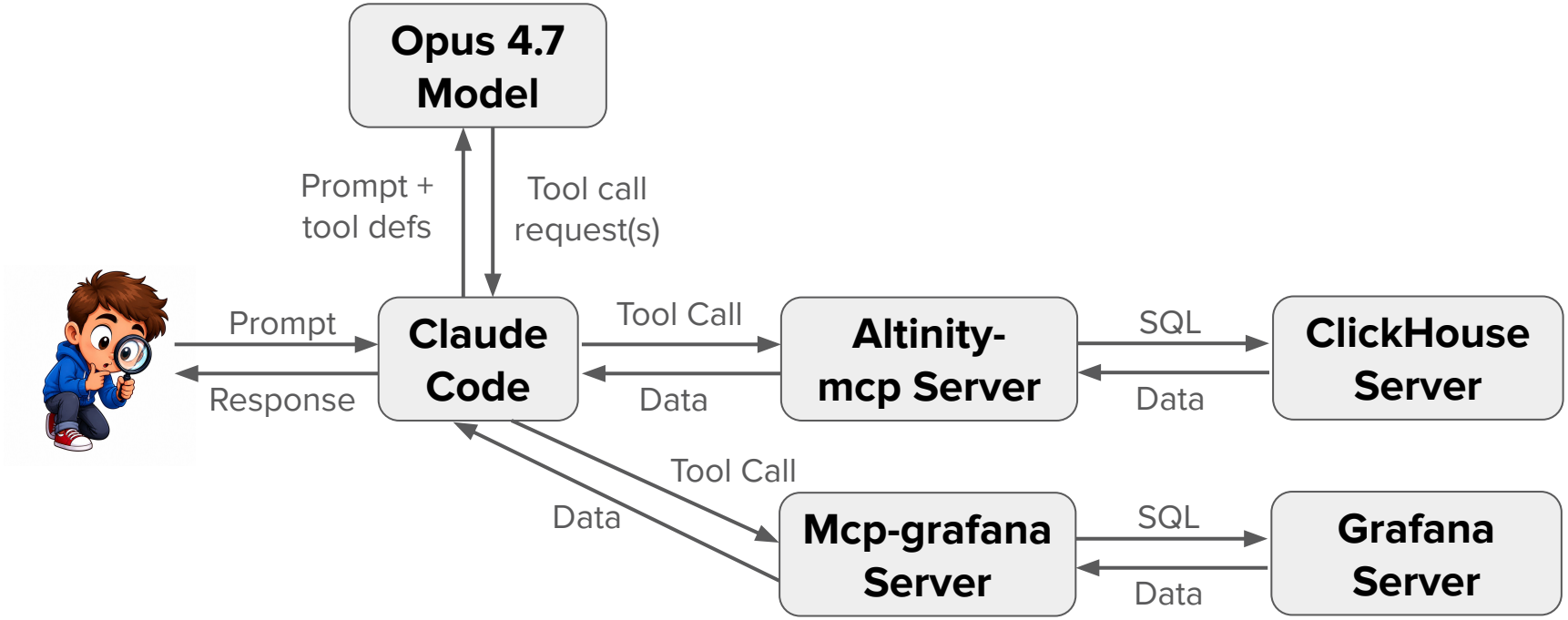
Using the (experimental) altinity-mcp and mcp-grafana servers to build Grafana dashboards for ClickHouse data

MCP* connects AI applications to external tools and data



* Model Context Protocol

AI workflows can use many MCP servers at once



Altinity MCP server

- Designed for ClickHouse®
- Easy-to-install Golang binary
- All MCP access methods (stdio, sse, http)
- execute_query tool as main instrument
- tools mapped to SQL code
- Password, JWE, OAuth2.1/OIDC authentication
- Additional OpenAPI (REST) endpoint
- Open source Apache 2.0 License

Experimental!

https://github.com/Altinity/altinity-mcp

How MCP servers are registered in Claude

```
.mcp.json:
{
  "mcpServers": {
    "grafana": {
      "command": "uvx", "args": ["mcp-grafana"],
      "env": {
        "GRAFANA_URL": "http://localhost:3000",
        "GRAFANA_SERVICE_ACCOUNT_TOKEN": "glsa…aa65"
      }
    },
    "clickhouse": {
      "command": "/usr/local/bin/altinity-mcp",
      "args": ["--config", "/home/user1/prezo/toys-in-the-attic/antalya.yaml"]
    }
  }
}
```

Configuration file for local altinity-mcp operation

antalya.yaml:

```
server:
  transport: "stdio"
clickhouse:
  read_only: true
  host: "antalya.demo.altinity.cloud"
  port: 9440
  protocol: "tcp"
  database: "default"
  username: "demo"
  password: "demo"
  limit: 5000
  tls:
    enabled: true
    insecure_skip_verify: false
```

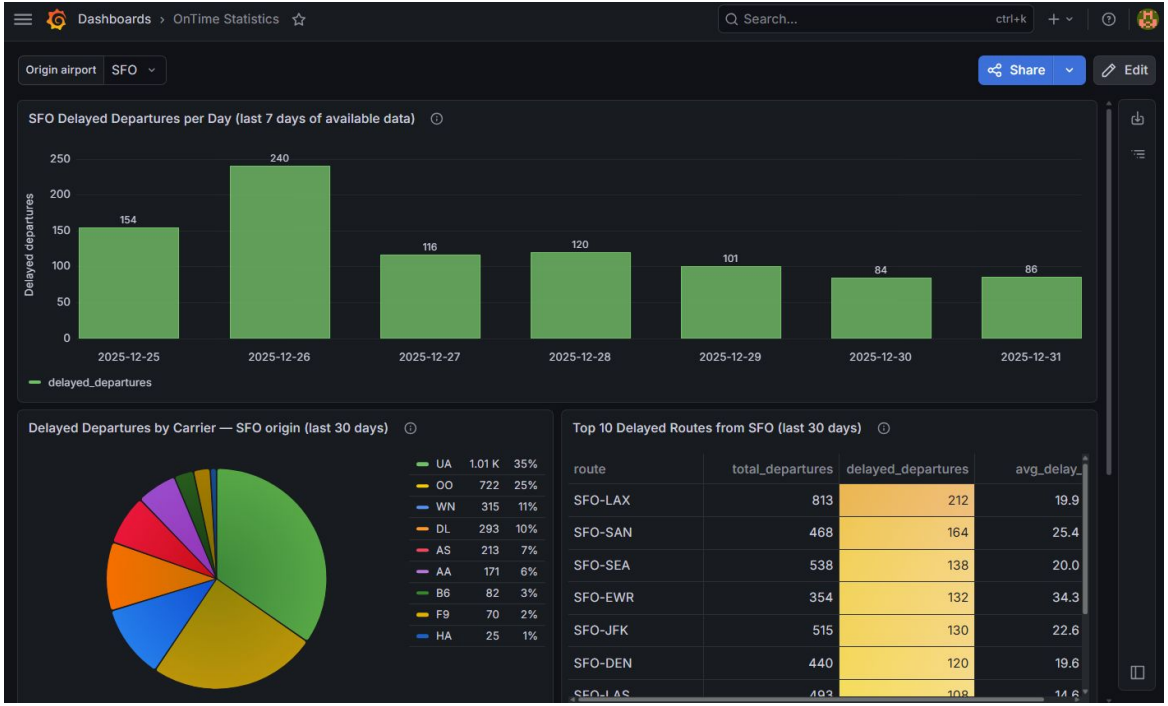
Demo time – Analyzing airline data

Please create a query that shows the number of delayed departures per day originating from SFO for the last 7 days.

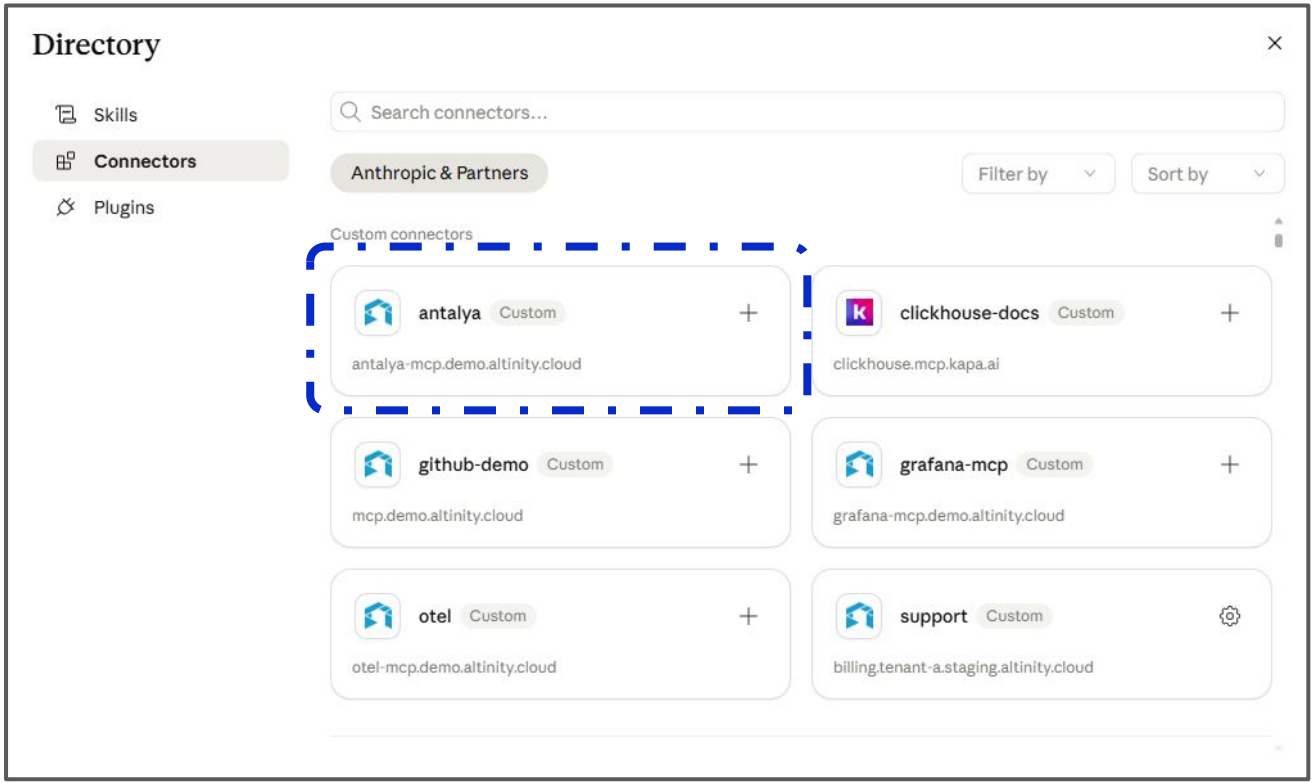
Demo time – Generating Grafana dashboards

Now create a new dashboard in Grafana for ontime statistics. Add a panel showing delayed departures as a bar chart.

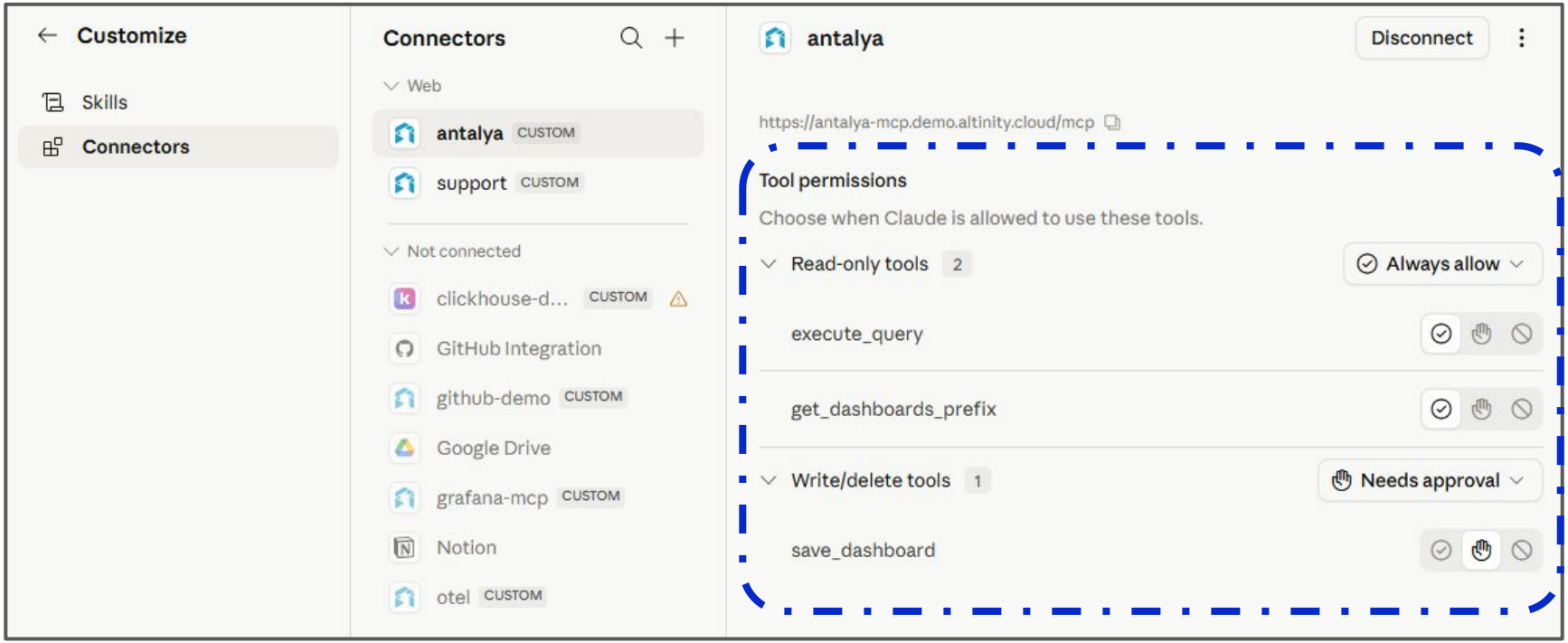
Resulting graph (after a small amount of work)



Installing MCP server in Claude



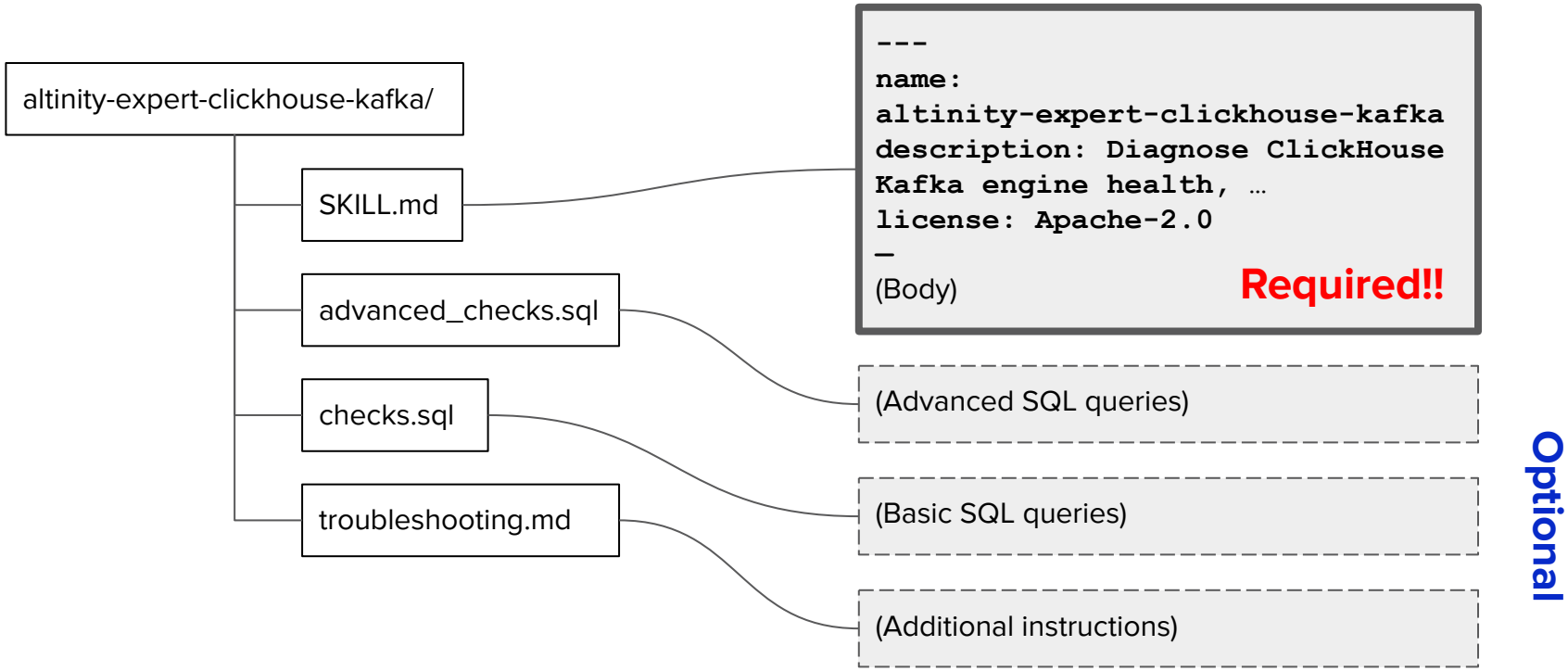
Finding available tools on the MCP server



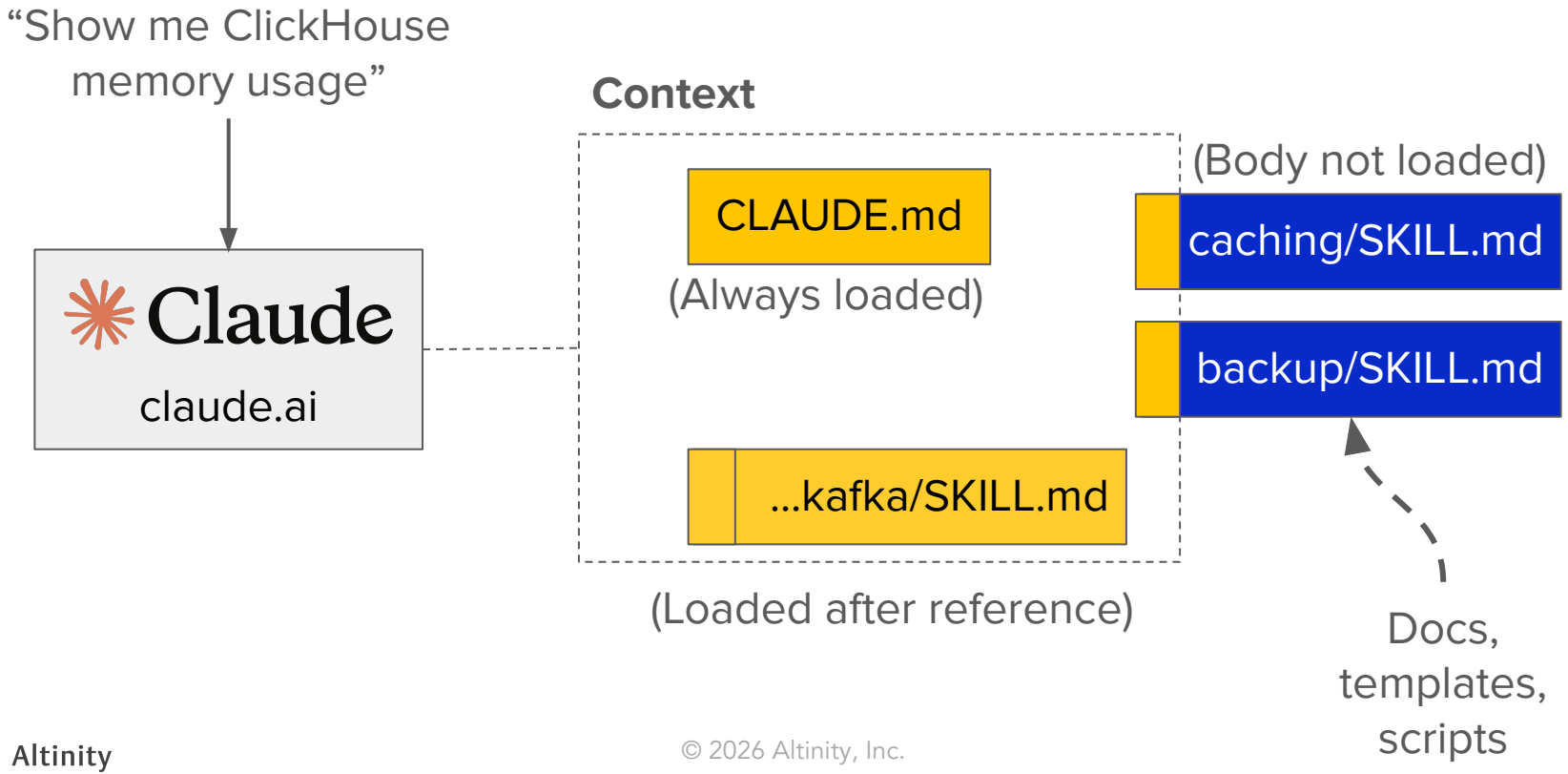
Skills

Using altinity-skills to design and debug real-time ClickHouse applications

Skills are packaged playbooks for agent tasks



Well-written skills minimize context utilization



Altinity-skills help you be a great ClickHouse DBA

- Provides context for analyzing ClickHouse design and operation
- Based on Altinity support team experience
- Usable on any build version
  - Upstream ClickHouse Official Builds
  - Altinity Stable Builds
  - Altinity Antalya Builds
  - Altinity FIPs Builds
- Open source Apache 2.0 License

https://github.com/Altinity/altinity-skills

Altinity-skills work on any agent that recognizes skills

Install directly from repo using npx (Vercel utility for skills)

```
npx skills add Altinity/altinity-skills/altinity-expert-clickhouse/
-a claude-code
```

Or clone the project and install/link directly.

```
git clone https://github.com/Altinity/altinity-skills
cd altinity-skills/altinity-expert-clickhouse/skills
for s in `ls -d $PWD/*`; do ln -s $s ~/.claude/skills/; done
```

Demo time – Assessing database cluster health

Use /altinity-clickhouse-expert-overview
to assess database health and generate
a nice HTML report

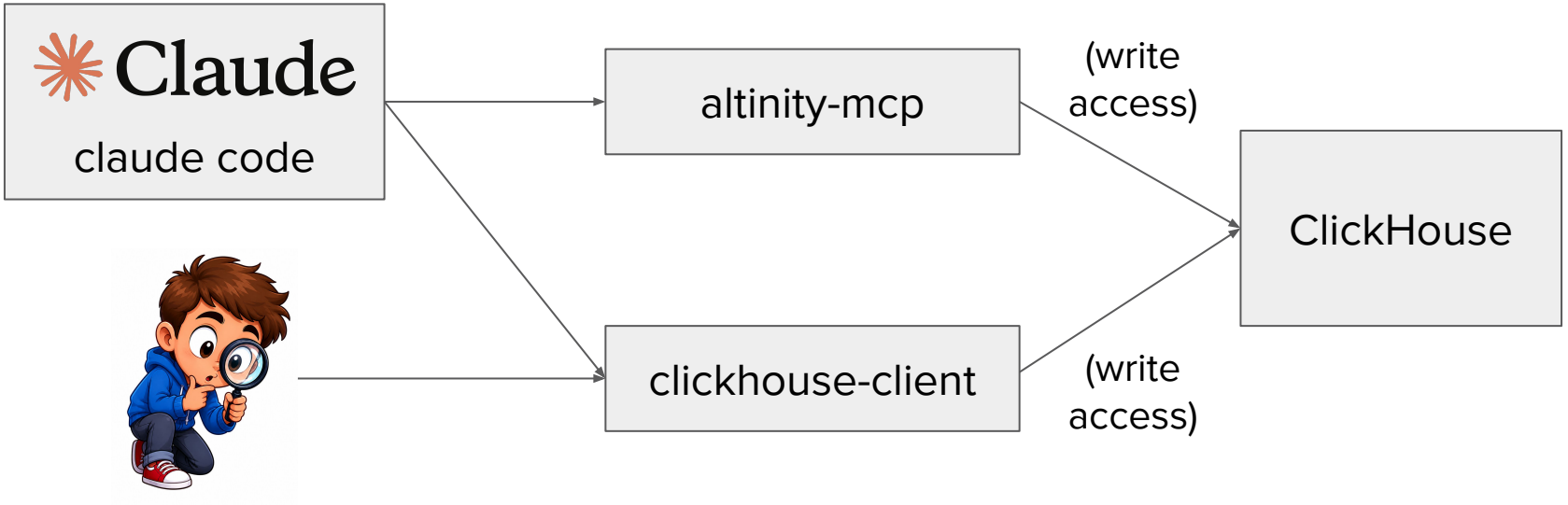
Guardrails for Databases

How to combine database roles, MCP, and authentication to protect data

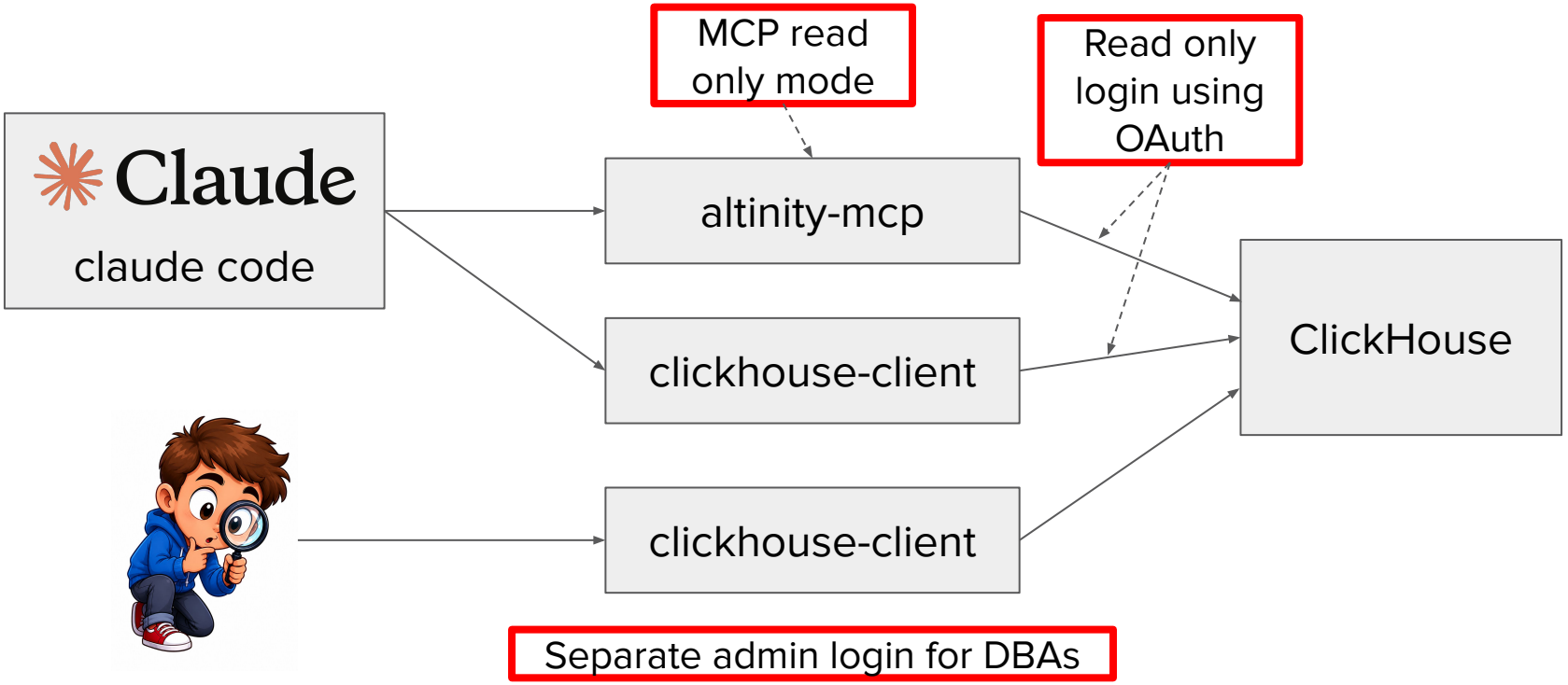
The real AI safety problem in one headline

https://x.com/lifeof_jer/status/2048103471019434248

Unsafe agent access to ClickHouse



Steps to lock down ClickHouse for agent access



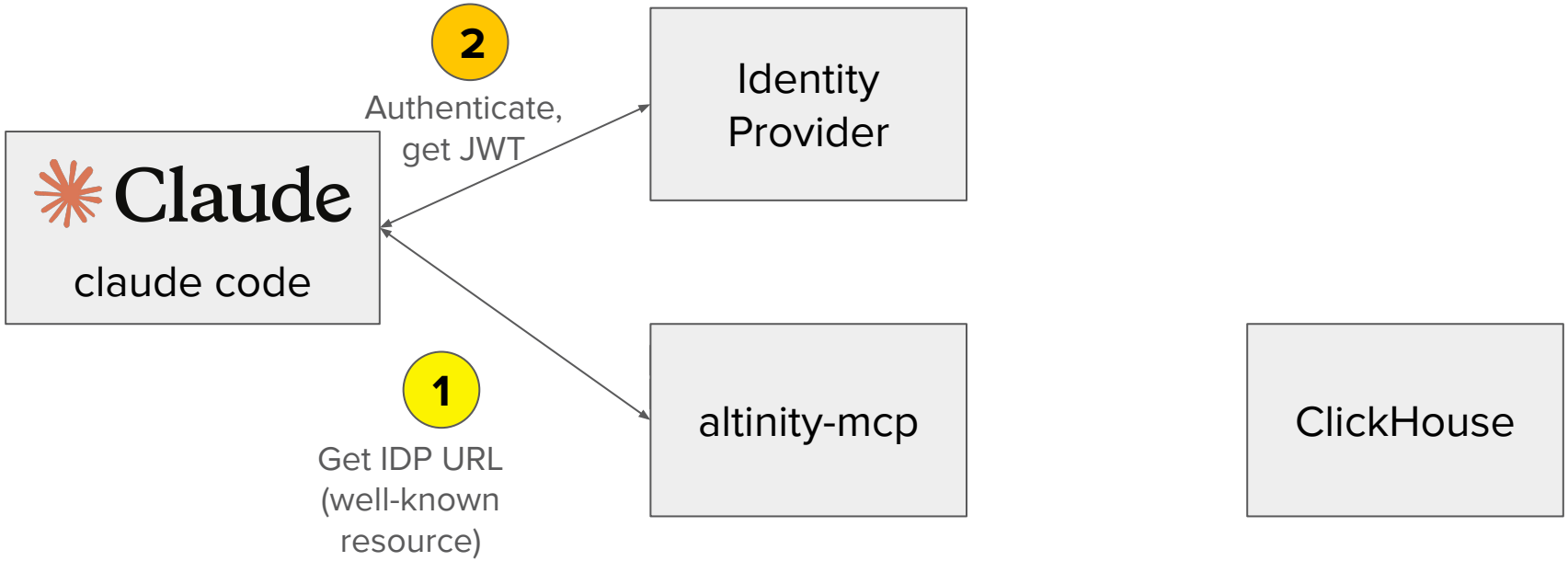
Step 1: Lock down MCP with read-only and OAuth

```
your-altinity-mcp.yaml:
server:
  transport: "stdio"
  oauth:
    enabled: true
    broker: false
    issuer: https://your-co.auth0.com/
    jwks_url: https://your-co.auth0.com/.well-known/jwks.json
    audience: https://antalya.demo.altinity.cloud/
    public_resource_url: https://antalya.demo.altinity.cloud/
    required_scopes: []
clickhouse:
  read_only: true
  host: "antalya.demo.altinity.cloud"
  . . .
```

Use OAuth but direct MCP clients to authenticate for themselves

Restrict clients to read-only SQL

Login: Claude authenticates directly to Identity Provider



Step 2: Configure OAuth token processor in Antalya server

/etc/clickhouse-server/config.d/google-oauth-antalya.xml

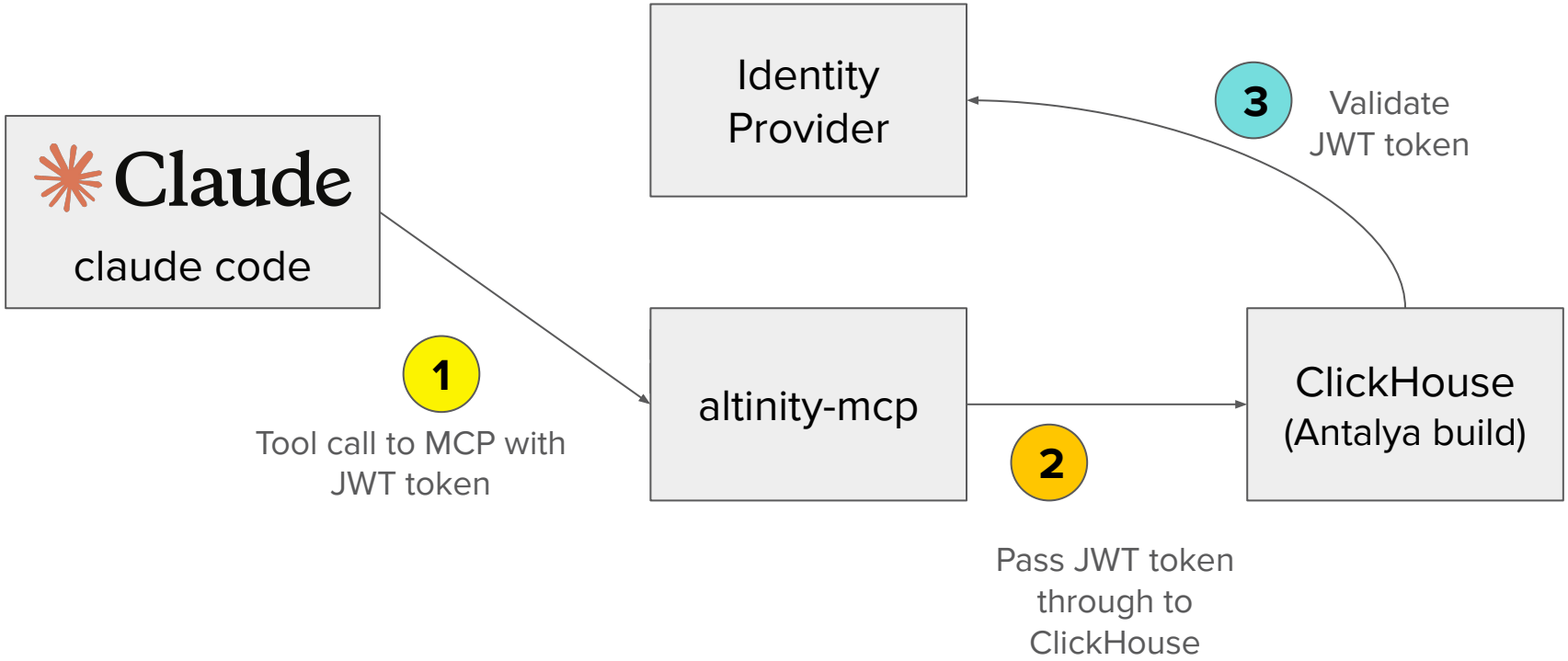
```
<clickhouse>
  <token_processors>
    <google_oauth>
      <type>openid</type>
      <userinfo_endpoint>https://openidconnect.googleapis.com/v1/userinfo
</userinfo_endpoint>
      <token_introspection_endpoint>https://openidconnect.googleapis.com/v1/userinfo
</token_introspection_endpoint>
      <jwks_uri>https://www.googleapis.com/oauth2/v3/certs</jwks_uri>
      <token_cache_lifetime>60</token_cache_lifetime>
      <username_claim>email</username_claim>
    </google_oauth>
  </token_processors>
</clickhouse>
```

Step 3: Configure OAuth user role mappings in Antalya

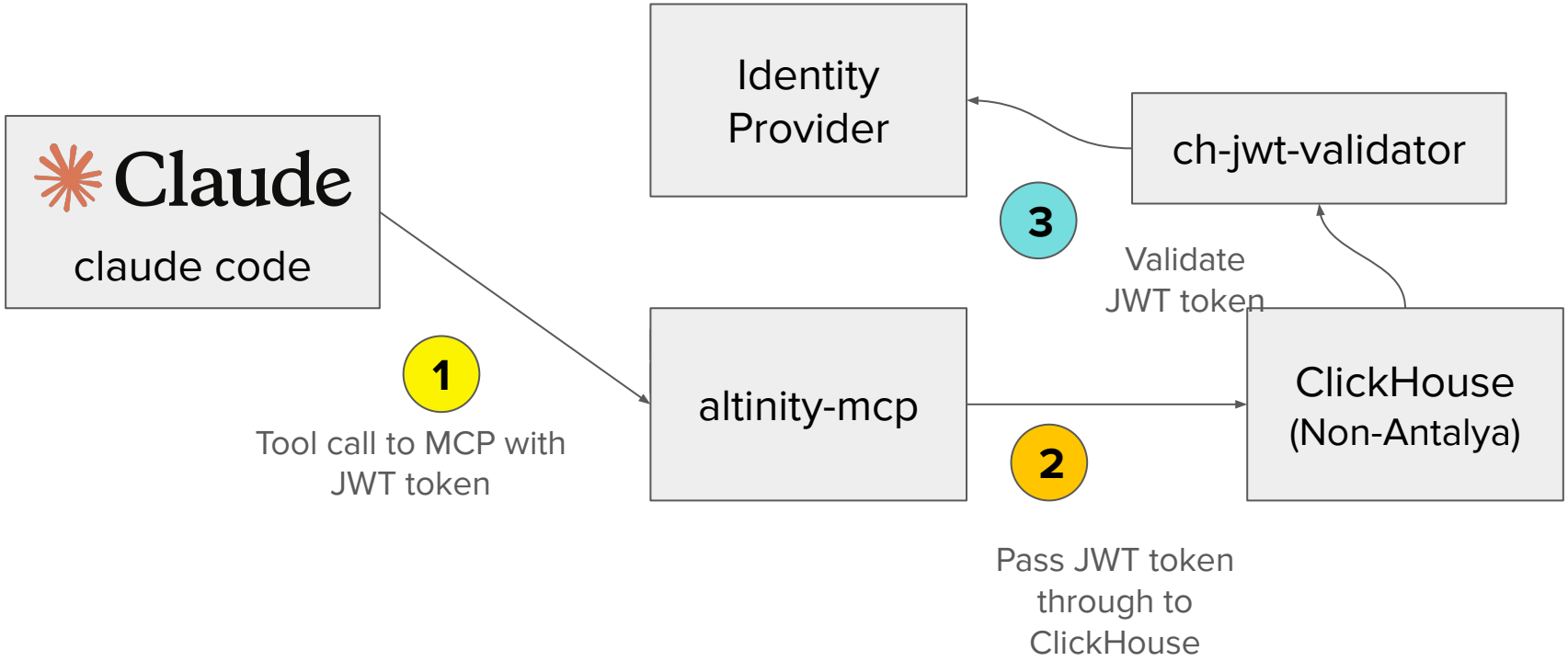
/etc/clickhouse-server/config.d/user_directories.xml:

```
<clickhouse>
  <user_directories replace="replace">
      <users_xml>
          <path>/etc/clickhouse-server/users.xml</path>
      </users_xml>
      <replicated>
          <zookeeper_path>/clickhouse/access/</zookeeper_path>
      </replicated>
      <token>
          <processor>google_oauth</processor>
          <common_roles>
              <oauth_demo_role />
          </common_roles>
      </token>
  </user_directories>
</clickhouse>
```

Tool call: Claude passes JWT token through to ClickHouse



What if your ClickHouse does not support OAuth?



Demo time – Using OAuth accounts with ChatGPT/Claude

Demonstrate how OAuth login enables read-only agent access to ClickHouse

Roadmap and Conclusion

What’s next?

Key learnings for applying AI to ClickHouse

Lower-cost models work well. Use Claude Sonnet rather than Opus for example.

Use altinity-skills project now to add bring tested DBA knowledge to your agents.

Once you have something interesting, reduce it to code, e.g., Grafana dashboard.

MCP is a good way to erect strong barriers between LLMs and production data

OAuth is very useful for granting broad agent access. It needs better documentation and tooling.

Altinity AI Roadmap

Enabling agent-centric application design and operation

- Improving altinity-skills – expand out from DBA focus
- Providing MCP access for any ClickHouse cluster in Altinity Cloud
- Simplifying security – broad, safe access with good docs and tools
- Answer key questions about agent use:
  - Optimization: cost, speed, and reliability
  - Security: Keeping data and schema out of hosted frontier models
- Introduce agents to automate “safe” operations fully
  - Tracking database metrics and handling alerts

Altinity Skills Project

Thank you!
Questions?

Contact us:

- Slack - https://altinity.com/slack
- Find out more - http://altinity.com

We're hiring!

ALTINITY

Traditional DBA

- Architecture & Schema design
- Performance tuning
- Backup, recovery, DR
- Security, compliance, access
- Capacity planning
- Incident response
- Business context translation (“what does the app really need?”)

AI DBA

- Query analysis & suggestions
- Index recommendations
- Anomaly detection
- Alert correlation
- Auto-tuning hints
- Log analysis
- Cost/performance optimization

Talk description - What we promised to discuss

AI Toys in the Attic: Our favorite AI tools and techniques for ClickHouse®

LLMs and agents are ushering in dramatic changes to data engineering. We're fascinated by them and working with AI constantly.

Join us to see our favorite tools, as well as how we're applying AI to build fast analytics on ClickHouse. We'll demonstrate MCP, skills, and agents while exploring popular datasets. We'll also show use of agents to design schema, load data, and generate queries.

Finally, we'll present an overview of our vision for applying AI safely and productively to real-time analytics. This is an exciting time to work on databases. Join the conversation with your questions and ideas!

Notes on talk - structure ideas

- What's altinity-mcp and how does it work?
  - What's MCP?
- Demo of searching ontime database
- What are altinity-skills and how do they work?
- Demo of using skills to check out schema
- How do build guardrails for data
  - Wiring separate MCP servers for each cluster
  - Users go directly to database, agents go through mcp
- Our roadmap for AI support
  - mcp
  - more skills!
  - custom agents?

Notes on talk - BI demos

> let's hammer that lifecycle, in the presentation.

How about something like that:

- https://claude.ai/share/bc05b1c6-1b09-49e9-b7c4-53ad40903329
- https://claude.ai/share/029bc932-4b0a-4bc6-a1a6-8ec2aa712a5b
- https://claude.ai/share/164df2d4-1008-45a7-9832-52246f56cefd

sonnet 4.6 work not too bad and fast enough, but we need to add more RAM and CPU power to the antalya cluster - it is much smaller than github.demo

Notes on talk: Current best practices

3:31

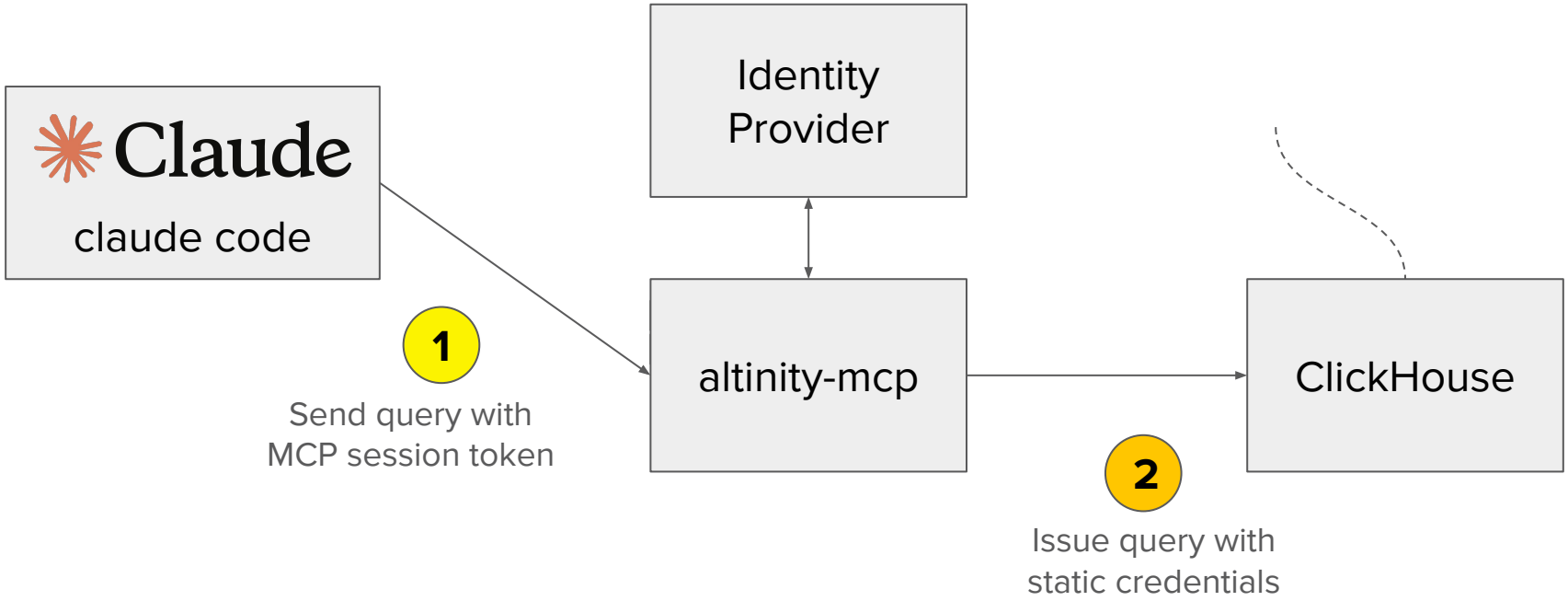
> it's far better to run that code afterwards because it's then deterministed and fast.

yes, that is my idea - BI analysts (and all people on the planet 🙂 should follow developers style of work - ask LLM to create a long-lasting artifact - dashboard/image/video/etc. The same as program. Not reinventing the wheel each time (edited)

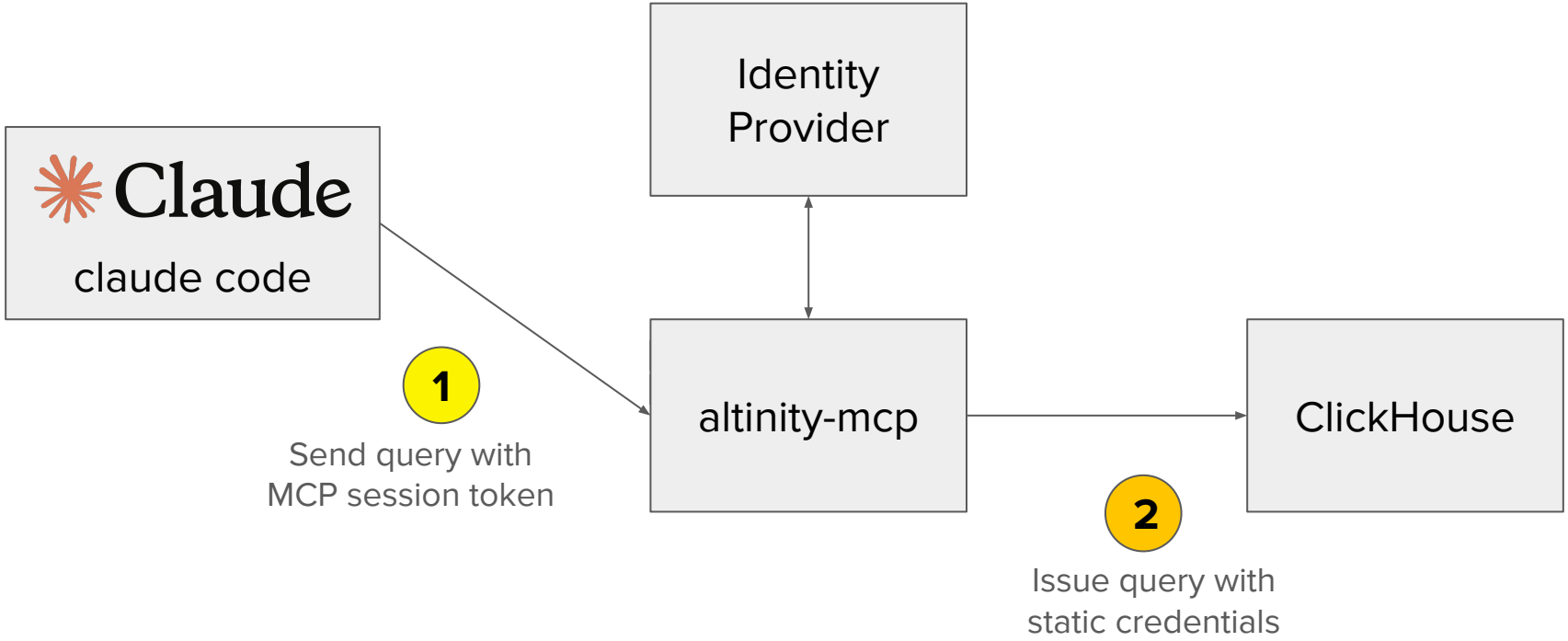
Robert Hodges (Altinity) Friday at 3:43 PM

Yep, let's hammer that lifecycle, in the presentation.

MCP operations on ClickHouse are simple in gated mode

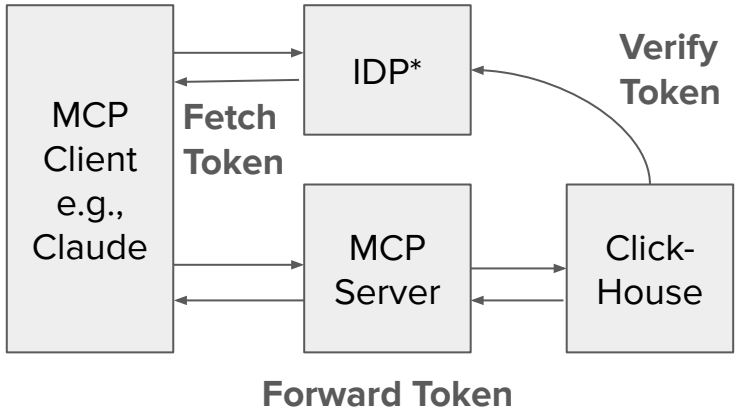


MCP operations on ClickHouse are simple in gated mode



altinity-mcp offers two different OAuth schemes

"Forward Mode"

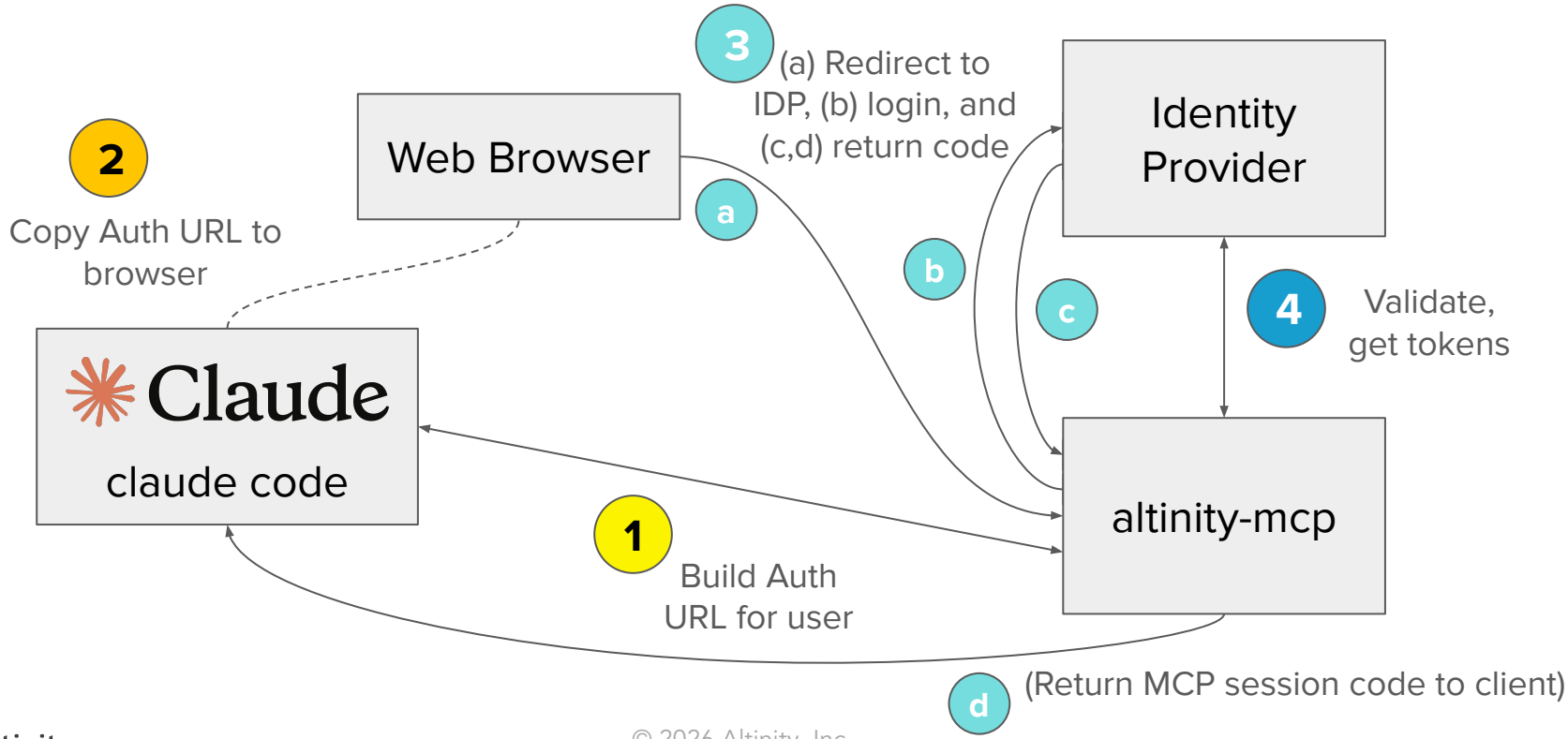


"Gating Mode"

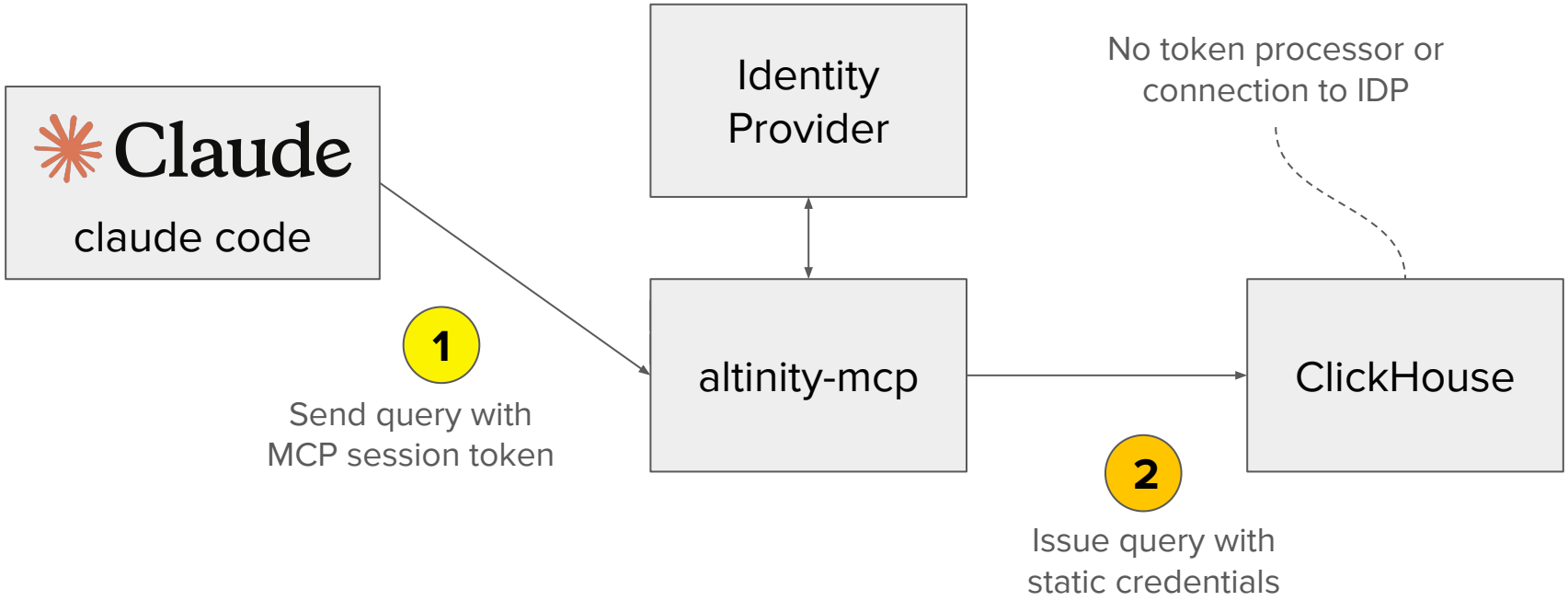
(Way too complicated to show here)

* Identity Provider

How browser-based login enables gating mode access



MCP operations on ClickHouse are simple in gated mode



Ensure your compute and storage match database needs

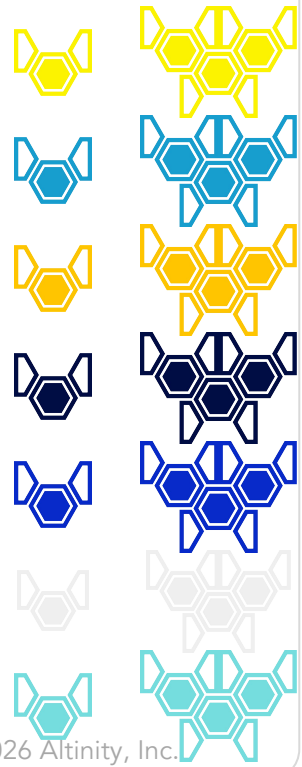
1. AWS Graviton instances are fast and cheap
2. Block storage enables vertical scaling and can failover when host dies
3. NVMe SSD gives the best raw performance but is tied to a single host and hard to configure in Kubernetes
4. Object storage is permission to play for large analytic databases

Icons- Transparent

Clickhouse (Native) Cluster

Director Cluster

Swarm Cluster



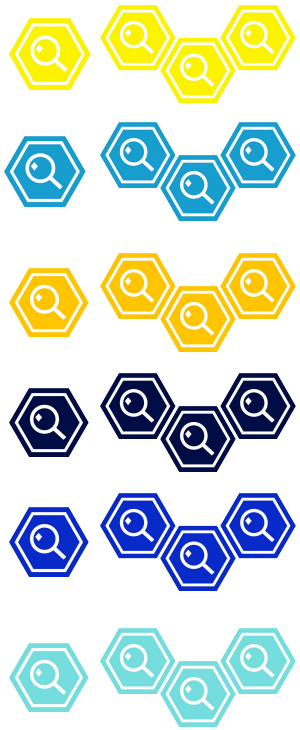
Keeper

Other

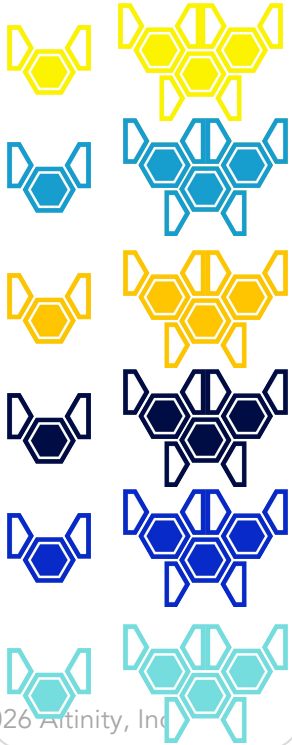
Icons-Stackable on White Backgrounds

Clickhouse (Native) Cluster

Director Cluster



Swarm Cluster



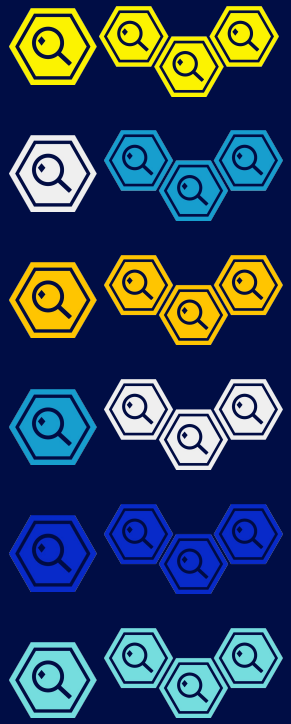
Keeper

Other

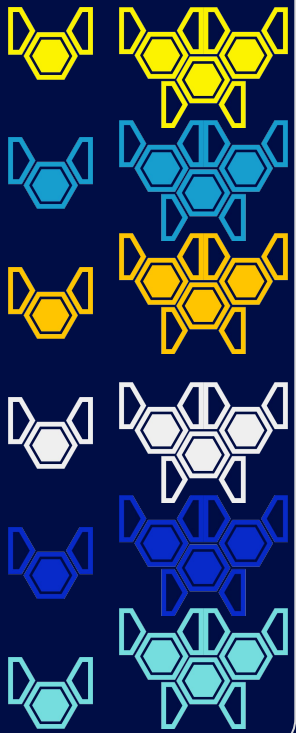
Icons-Stackable on Dark Backgrounds

Clickhouse (Native) Cluster

Director Cluster

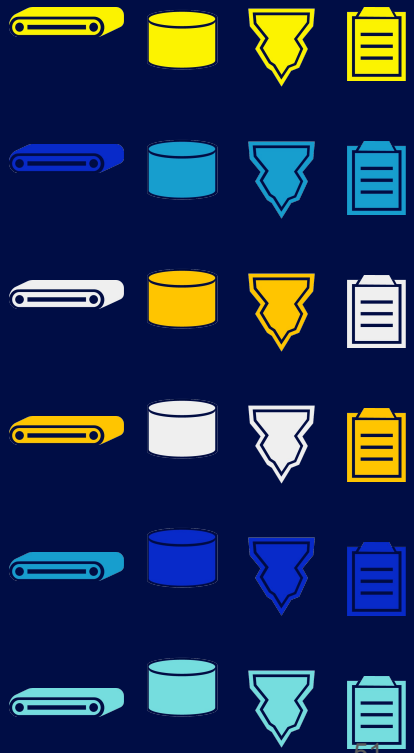


Swarm Cluster



Keeper

Other



Rescaling – adding more disks

```
templates:
  volumeClaimTemplates:
    - name: disk1
      spec:
        accessModes:
          - ReadWriteOnce
        resources:
          requests:
            storage: 100Gi
```

```
  settings:
    storage_configuration/disks/disk2/path: /var/lib/clickhouse2/
    storage_configuration/policies/default/volumes/default/disk: default
    storage_configuration/policies/default/volumes/disk2/disk: disk2
templates:
  volumeClaimTemplates:
    - name: disk1
      spec:
        accessModes:
          - ReadWriteOnce
        resources:
          requests:
            storage: 100Gi
    - name: disk2
      spec:
        accessModes:
          - ReadWriteOnce
        resources:
          requests:
            storage: 100Gi
  podTemplates:
    - name: default
      spec:
        containers:
          - name: clickhouse-pod
            volumeMounts:
              - name: disk1
                mountPath: /var/lib/clickhouse
              - name: disk2
                mountPath: /var/lib/clickhouse2
```

Restart rules

```
configurationRestartPolicy:
  rules:
    - version: "*"
      rules:
        - settings/*: "yes"
        # single values
        - settings/access_control_path: "no"
        - settings/dictionaries_config: "no"
        - settings/max_server_memory_*: "no"
        - settings/max_*_to_drop: "no"
        - settings/max_concurrent_queries: "no"
        - settings/models_config: "no"
        - settings/user_defined_executable_functions_config: "no"
        # structured XML
        - settings/logger/*: "no"
        - settings/macros/*: "no"
        - settings/remote_servers/*: "no"
        - settings/user_directories/*: "no"
        # these settings should not lead to pod restarts
        - settings/display_secrets_in_show_and_select: "no"
        - zookeeper/*: "no"
        - files/*.xml: "yes"
        - files/config.d/*.xml: "yes"
        - files/config.d/*dict*.xml: "no"
        - files/config.d/*no_restart*: "no"
        # exceptions in default profile
        - profiles/default/background_*_pool_size: "yes"
        - profiles/default/max_*_for_server: "yes"
```

Defined in operator configuration

Can be altered using ClickHouseOperatorConfiguration resource

Plan to utilize `system.server_settings.changeable_without_restart` eventually

Trick to avoid restarts for certain configuration files