

Don't manage data infrastructure yourself

Clickhouse Tips for Growing
Start-ups

Artem Popov @ ClickHouse Berlin Meetup on May 16 2023



Fingerprint

YOUR VISITOR ID ⓘ

YkW [REDACTED]

YOUR VISIT SUMMARY You visited 3 times

INCOGNITO ⓘ

1 session

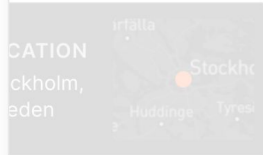
IP ADDRESS

2 IPs

GEOLOCATION ⓘ

2 locations

YOUR VISIT HISTORY



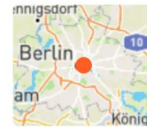
2023



IP 193 [REDACTED]

INCOGNITO No

LOCATION
Berlin,
Germany



CURRENT VISIT







Artem Popov

Identification Team Lead

- Area of expertise – intersection of High Load Web Engineering and Data Science/Analytics Engineering/Machine Learning.
- Before Fingerprint.com worked in Large IT corporations.
- Never done Data Engineering Myself before the Fingerprint.com
- <https://www.linkedin.com/in/system29a/>



State around a year ago

- Relatively small engineering team
- All services hosted on AWS
- Billions of raw events per year with hundreds of columns.



Data Infrastructure Story

I

Copper Age

1. Self-managed Clickhouse PoC
2. Self-managed Metabase

II

Bronze Age

1. Self-managed Clickhouse, Production ready
2. Self-managed Metabase

III

Silver Age

1. Self-managed Clickhouse
2. DBT
3. Prefect
4. Preset

IV

Golden Age

1. Clickhouse Cloud
2. DBT + Materialized Views
3. Prefect
4. Preset

PART I

Copper Age

Proof of concept





Amazon **Redshift**



SELF-MANAGED?

 Fingerprint +  ClickHouse



REFERENCE DEPLOYMENT

ClickHouse Cluster on AWS

An open-source, column-oriented database management system

[View deployment guide](#)

This solution deploys a [ClickHouse](#) cluster on the Amazon Web Services (AWS) Cloud. ClickHouse is an open-source, column-oriented database management system (DBMS), which can be used for online analytical processing (OLAP) of queries.

This deployment is for customers who want to process analytical queries using a DBMS, such as MySQL, PostgreSQL, and Oracle Database. During the deployment, customers can configure the AWS CloudFormation templates to define the desired cluster nodes and settings.

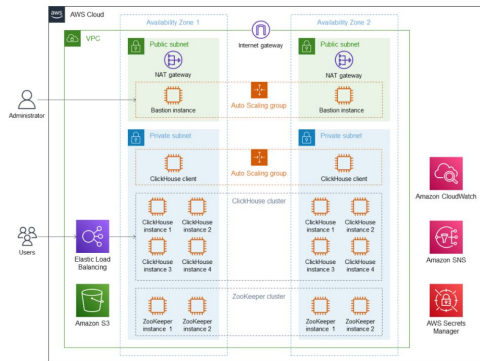


This solution was developed by AWS.

[What you'll build](#) | [How to deploy](#) | [Costs and licenses](#)

This solution sets up the following:

- A highly available architecture that spans two Availability Zones.*
- A virtual private cloud (VPC) configured with public and private subnets, according to AWS best practices, to provide you with your own virtual network on AWS.*
- An internet gateway to allow internet access for bastion hosts.*
- In the public subnets:
 - Managed network address translation (NAT) gateways to allow outbound internet access for resources in the private subnets.*
 - A Linux bastion host in an Auto Scaling group to allow inbound Secure Shell (SSH) access to Amazon Elastic Compute Cloud (Amazon EC2) instances in public and private subnets.*
- In the private subnets:
 - A ClickHouse client in an Auto Scaling group to allow administrators to connect to the ClickHouse cluster.
 - A ClickHouse database cluster that contains Amazon EC2 instances.
 - A ZooKeeper cluster that contains Amazon EC2 instances for storing metadata for ClickHouse replication. Each replica stores its state in ZooKeeper as the set of parts and its checksums.



[Enlarge image](#)

[View deployment guide for details](#)





The results are great!

- Data grows
- Use-cases and read usage grows

Cryptic Errors?

```
Code: 49. DB::Exception: Received from
localhost:9002. DB::Exception: Prefetch is
valid in readUntilPosition: (while reading
column column_name): (while reading from part
/var/lib/clickhouse/disks/s3/data/database/tabl
e/part/ from mark 225 with max_rows_to_read =
8192): While executing MergeTreeThread.
(LOGICAL_ERROR)
```

A decorative graphic on the left side of the slide, consisting of several concentric, curved lines that resemble a fingerprint or a stylized spiral, rendered in a dark gray color against a black background.

First problems

- Problem in the SSD (warm) + s3 (cold) storage scheme implemented in the recipe which is not working correctly
- Instead of product tasks, our engineer starts to dig into advanced Clickhouse configuration.



For self-managed solutions “Easy to start” ≠ and “Easy to support or modify. Every step into not a basic use case and you have to invest much more than you could imagine.

Insight #1



More Problems

- Everything starts to slow down, and timeout errors everywhere
- So we need to scale the system and our engineering practices to the higher demand both in terms of data scale and usage scale.

PART II

Bronze Age

From PoC to Production



From PoC to production?

- we use Clickhouse for more and more around the production use cases. It is a completely different SLA.
- What do we need for manageable data storage?



Monitorings and Alerts



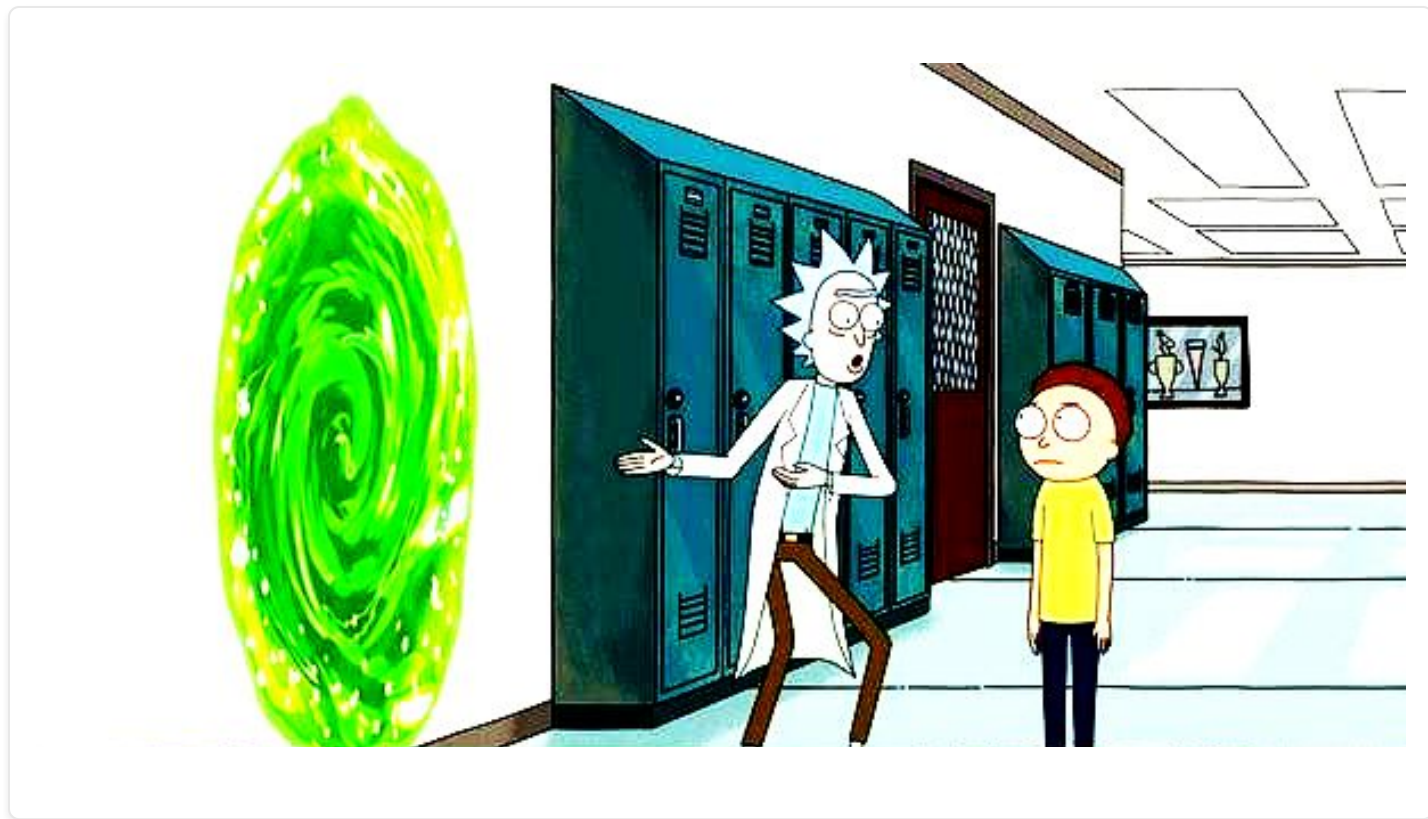


Backups





laC





Hardware costs optimisation





**We have a much more mature
self-managed clickhouse than
before!**

But what is the price?



If you start to use the self-managed infrastructure of the Open Source product, you will definitely put weeks of work into supporting it if it was successful and should be reliable and scaled. All this work will be combined from dozens of “small” improvements that have dozens of nuances.

Insight #2



Even after all these improvements...

- we can't use raw queries for everything already and have to consider other options

PART III

Silver Age

Dealing with the scaling



Problems

- SQL codebase grows, a lot of code duplication
- Running SQL queries over the raw data is ineffective on new scale.



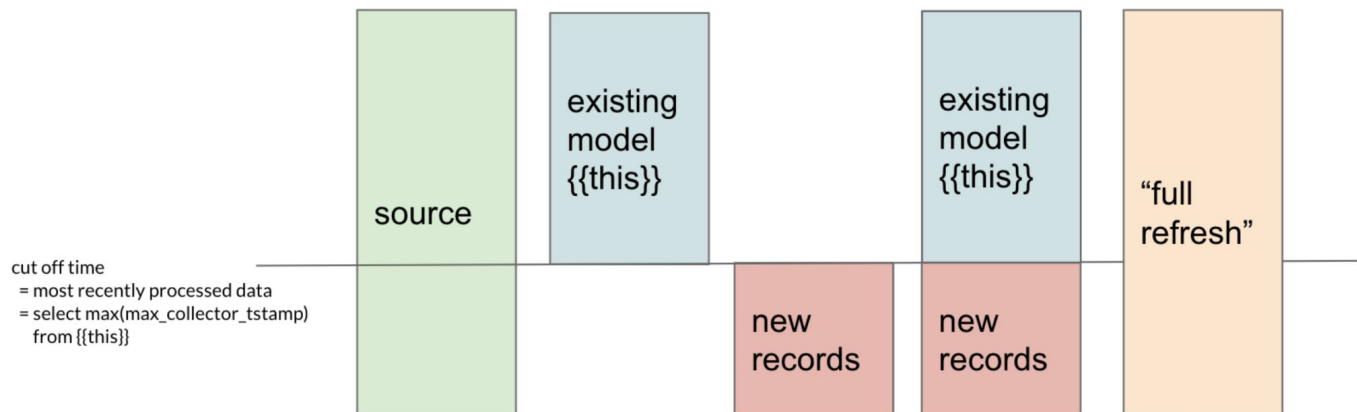


```
select
    {{ identification_events_aliases() }} ,
    *
from
    {{ source('default', 'identification_events') }}
```


```
version: 2
```

```
models:
```


- name: parsed_identification_events
 - description: parses and enriches identification_events data into human-readable fields
 - materialized: view
 - config:
 - tags: parsed_identification_events






 ClickHouse / **dbt-clickhouse** Public

[Code](#) [Issues 17](#) [Pull requests 4](#) [Actions](#) [Projects](#) [Security](#) [Insights](#)

 **Support for Distributed table engine** help wanted

#14 opened on Aug 27, 2021 by Volodin-DD 23

 **Standard `table` materialization does not work with ReplicateMergeTree engine**

#95 opened on Sep 8, 2022 by mharrisb1 6



Append Strategy (inserts-only mode)

To overcome the limitations of large datasets in incremental models, the plugin uses the dbt configuration parameter `incremental_strategy`. This can be set to the value `append`. When set, updated rows are inserted directly into the target table (a.k.a `imdb_dbt.actor_summary`) and no temporary table is created. Note: Append only mode requires your data to be immutable or for duplicates to be acceptable. If you want an incremental table model that supports altered rows don't use this mode!

To illustrate this mode, we will add another new actor and re-execute dbt run with `incremental_strategy='append'`.

1. Configure append only mode in actor_summary.sql:

```
{{ config(order_by='(updated_at, id, name)', engine='MergeTree()', materialized='incremental', unique_key='id', incremental_strategy='append') }}
```



Using DBT Incremental Materializations in Production

- Tasks Orchestration
- Errors/retries handling
- A lot of other nuances with distributed setup





PREFECT CLOUD



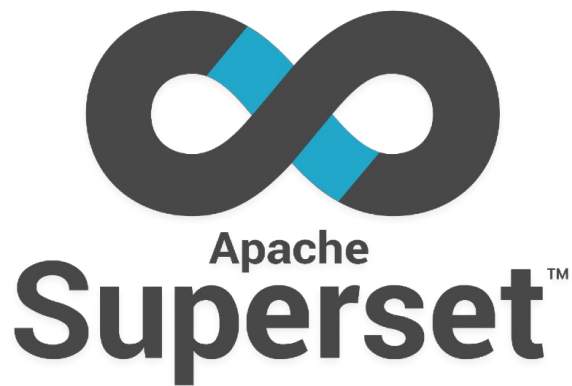
We still saw a great value in DBT

- looking for other options for data aggregation and materializations



The Idea of “Self-managed infrastructure could manifest itself in non-obvious places (like running CLI script every 5 minutes) and have same implications”

Insight #3



PART IV

Golden Age

#FailLearnGrow



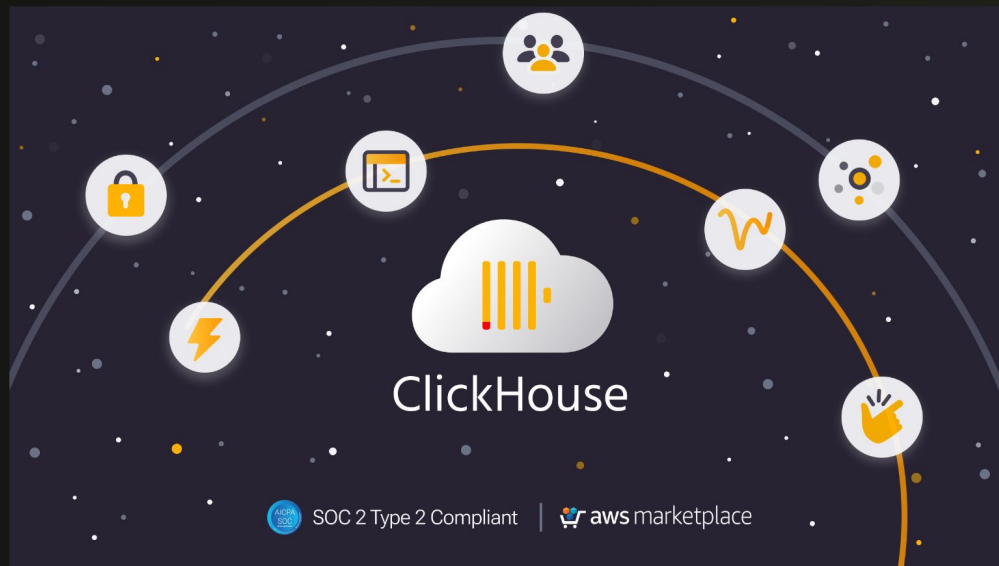
Blog / Product

ClickHouse Cloud is now Generally Available



Tanya Bragin

Dec 6, 2022





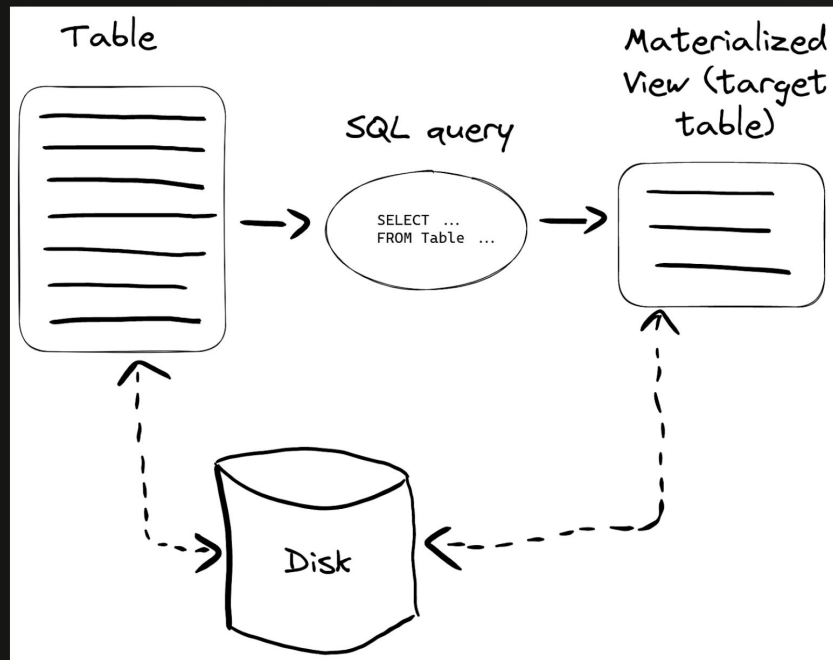
More value than expected!

- the Cloud solution was cheaper
- automatically scalable (less MemoryLimit/Timeout errors)
- Much easier database schema
- Support from the Clickhouse team
- We now can concentrate on working on our own product



What is a Materialized View?

A materialized view is a special trigger that stores the result of a `SELECT` query on data, as it is inserted, into a target table:

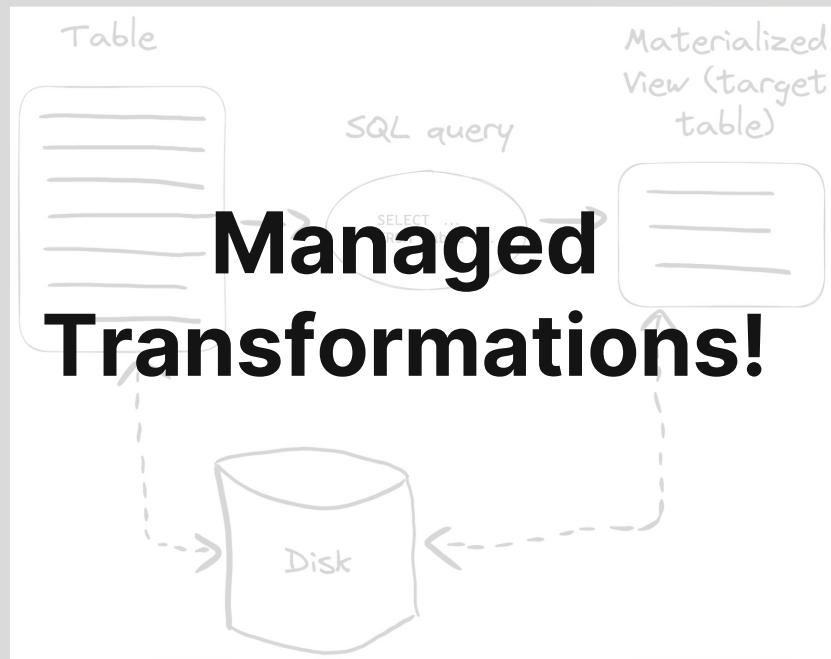


This can be useful in many cases, but let's take the most popular - making certain queries work faster.

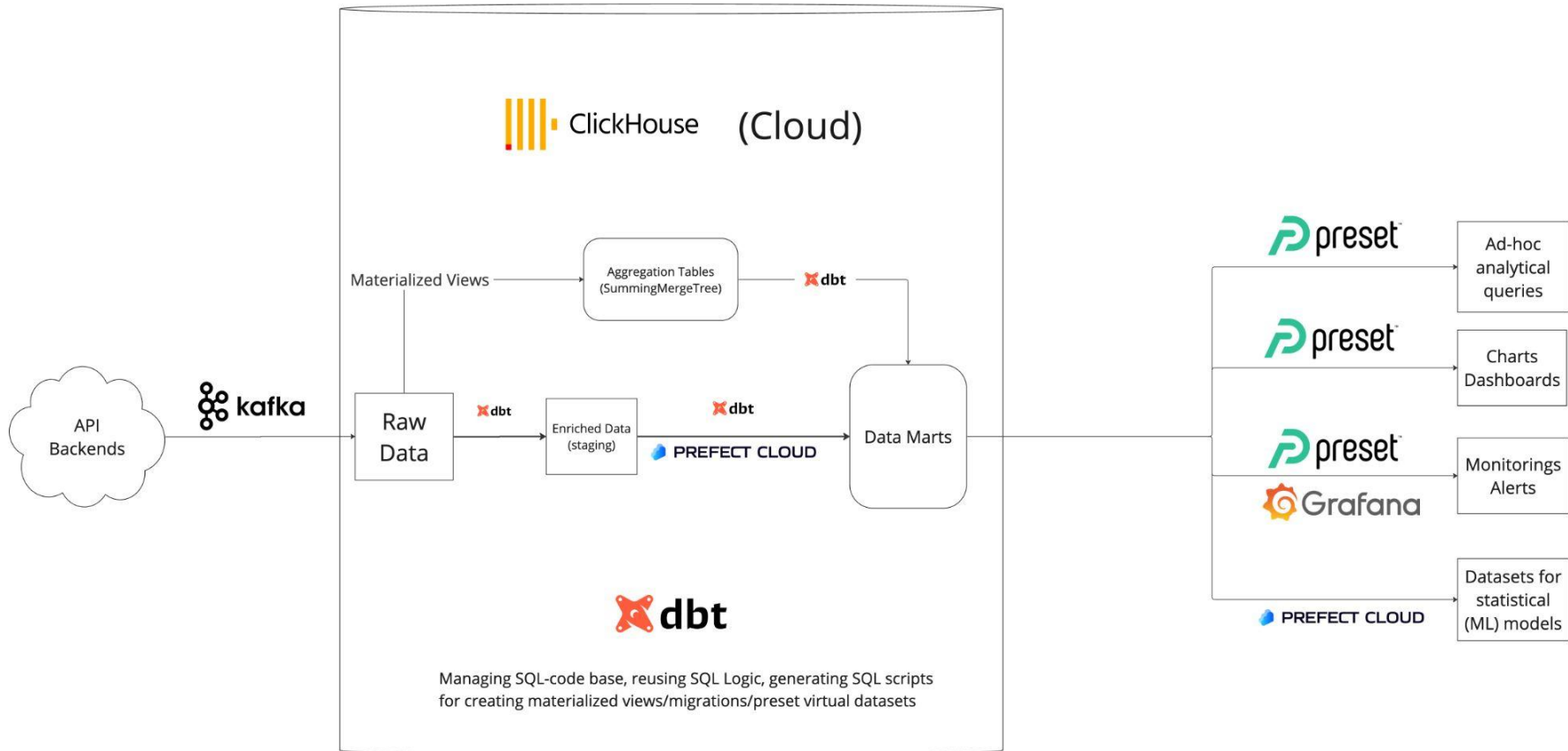


What is a Materialized View?

A materialized view is a special trigger that stores the result of a `SELECT` query on data, as it is inserted, into a target table:



This can be useful in many cases, but let's take the most popular - making certain queries work faster.



**Don't manage data
infrastructure yourself**



When considering the price of managed products:

- Include the price of your own infrastructure.
- Include potential load for support on your engineers: scaling, monitoring, dealing with your own nonoptimal solutions, upgrading, keeping the security well, backup, etc.
- **Include lost opportunities to work on unique tasks your team solves.**
- Include the absence of support when dealing with software issues.



Artem Popov

Team Lead | High Load Web | Data Science |
Analytics Engineering – Fingerprint



Thank you!

