Vector Search in ClickHouse

Do you really need a Vector database?

Dale McDiarmid - Product@ClickHouse

May, 2023

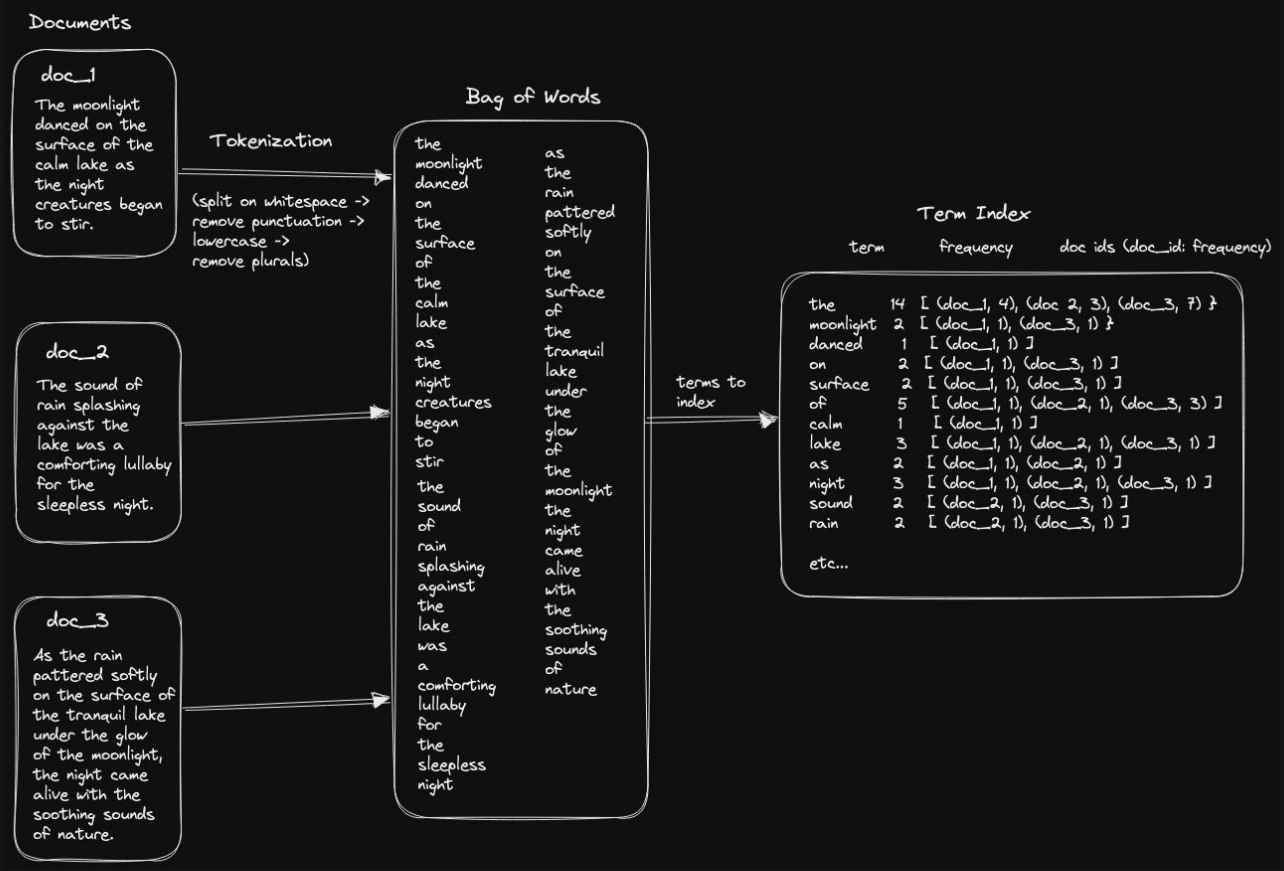
ClickHouse

A *very* brief history of search

(more here)

ClickHouse

Bag of Words + Inverted Indices


Documents
Tokenization
(split on whitespace -> remove punctuation -> lowercase -> remove plurals)
Bag of Words
terms to index
Term Index

doc_1

The moonlight danced on the surface of the calm lake as the night creatures began to stir.

doc_2

The sound of rain splashing against the lake was a comforting lullaby for the sleepless night.

doc_3

As the rain pattered softly on the surface of the tranquil lake under the glow of the moonlight, the night came alive with the soothing sounds of nature.

Bag of Words

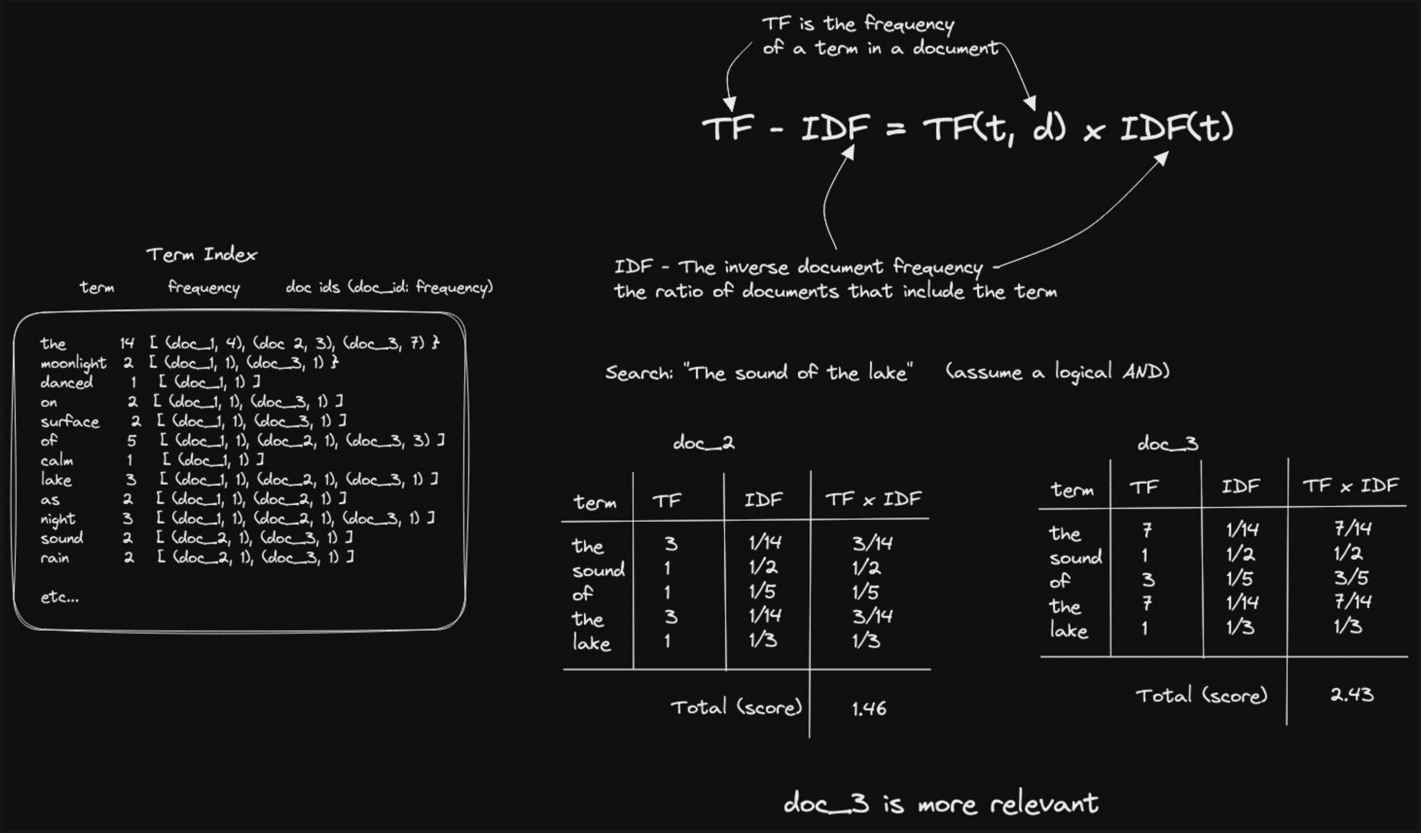
the
moonlight
danced
on
the
surface
of
the
calm
lake
as
the
night
creatures
began
to
stir
the
sound
of
rain
splashing
against
the
lake
was
a
comforting
lullaby
for
the
sleepless
night
as
the
rain
pattered
softly
on
the
surface
of
the
tranquil
lake
under
the
glow
of
the
moonlight
the
night
came
alive
with
the
soothing
sounds
of
nature

Term Index

term	frequency	doc ids (doc_id: frequency)
the	14	[(doc_1, 4), (doc 2, 3), (doc_3, 7) }
moonlight	2	[(doc_1, 1), (doc_3, 1) }
danced	1	[(doc_1, 1)]
on	2	[(doc_1, 1), (doc_3, 1)]
surface	2	[(doc_1, 1), (doc_3, 1)]
of	5	[(doc_1, 1), (doc_2, 1), (doc_3, 3)]
calm	1	[(doc_1, 1)]
lake	3	[(doc_1, 1), (doc_2, 1), (doc_3, 1)]
as	2	[(doc_1, 1), (doc_2, 1)]
night	3	[(doc_1, 1), (doc_2, 1), (doc_3, 1)]
sound	2	[(doc_2, 1), (doc_3, 1)]
rain	2	[(doc_2, 1), (doc_3, 1)]

etc...

Scoring - Measuring Relevancy - TF-IDF/BM25



TF is the frequency of a term in a document

$$TF - IDF = TF(t, d) \times IDF(t)$$

IDF - The inverse document frequency
the ratio of documents that include the term

Term Index

term	frequency	doc ids (doc_id: frequency)
the	14	[(doc_1, 4), (doc 2, 3), (doc_3, 7) }
moonlight	2	[(doc_1, 1), (doc_3, 1) }
danced	1	[(doc_1, 1)]
on	2	[(doc_1, 1), (doc_3, 1)]
surface	2	[(doc_1, 1), (doc_3, 1)]
of	5	[(doc_1, 1), (doc_2, 1), (doc_3, 3)]
calm	1	[(doc_1, 1)]
lake	3	[(doc_1, 1), (doc_2, 1), (doc_3, 1)]
as	2	[(doc_1, 1), (doc_2, 1)]
night	3	[(doc_1, 1), (doc_2, 1), (doc_3, 1)]
sound	2	[(doc_2, 1), (doc_3, 1)]
rain	2	[(doc_2, 1), (doc_3, 1)]
etc...		

Search: "The sound of the lake" (assume a logical AND)

doc_2

term	TF	IDF	TF x IDF
the	3	1/14	3/14
sound	1	1/2	1/2
of	1	1/5	1/5
the	3	1/14	3/14
lake	1	1/3	1/3
		Total (score)	1.46

doc_3

term	TF	IDF	TF x IDF
the	7	1/14	7/14
sound	1	1/2	1/2
of	3	1/5	3/5
the	7	1/14	7/14
lake	1	1/3	1/3
		Total (score)	2.43

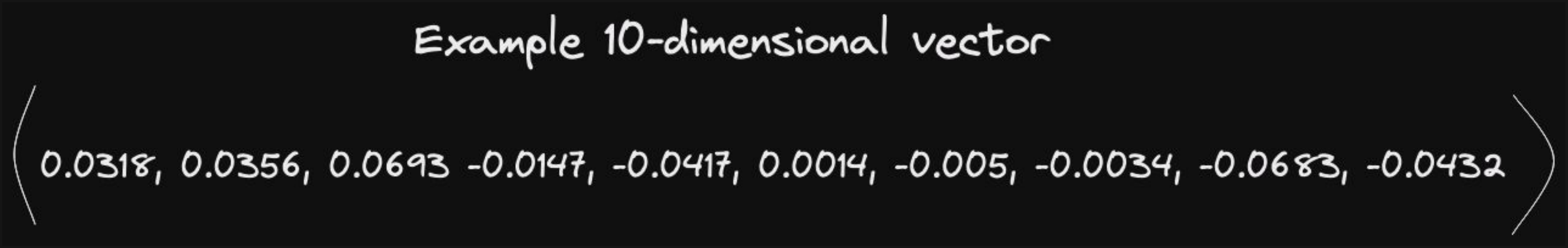
doc_3 is more relevant

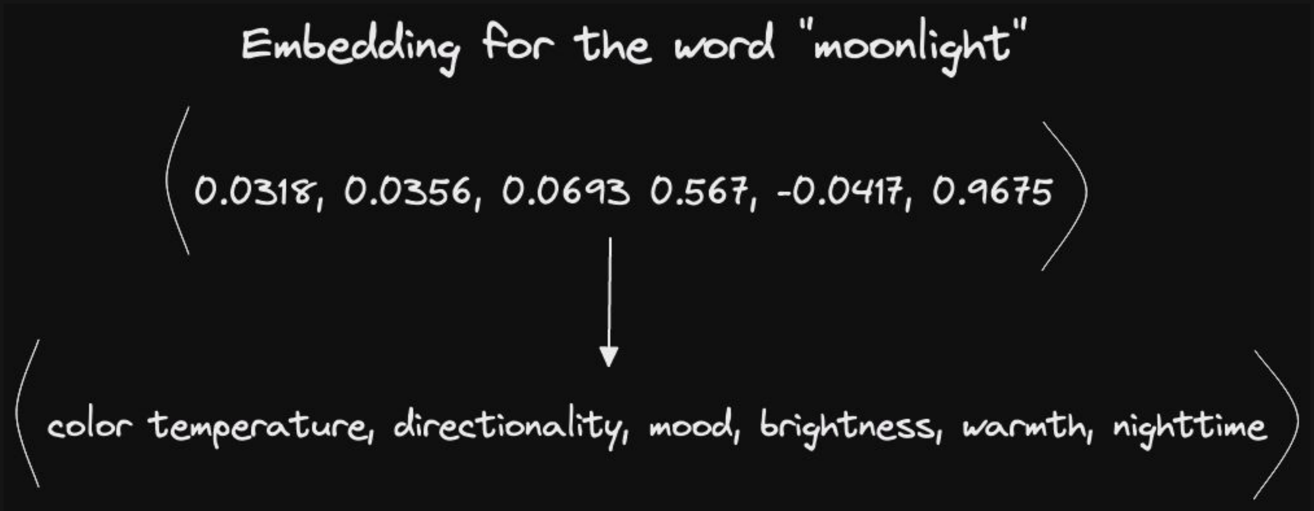
What is a vector/embedding?

(more than just text)

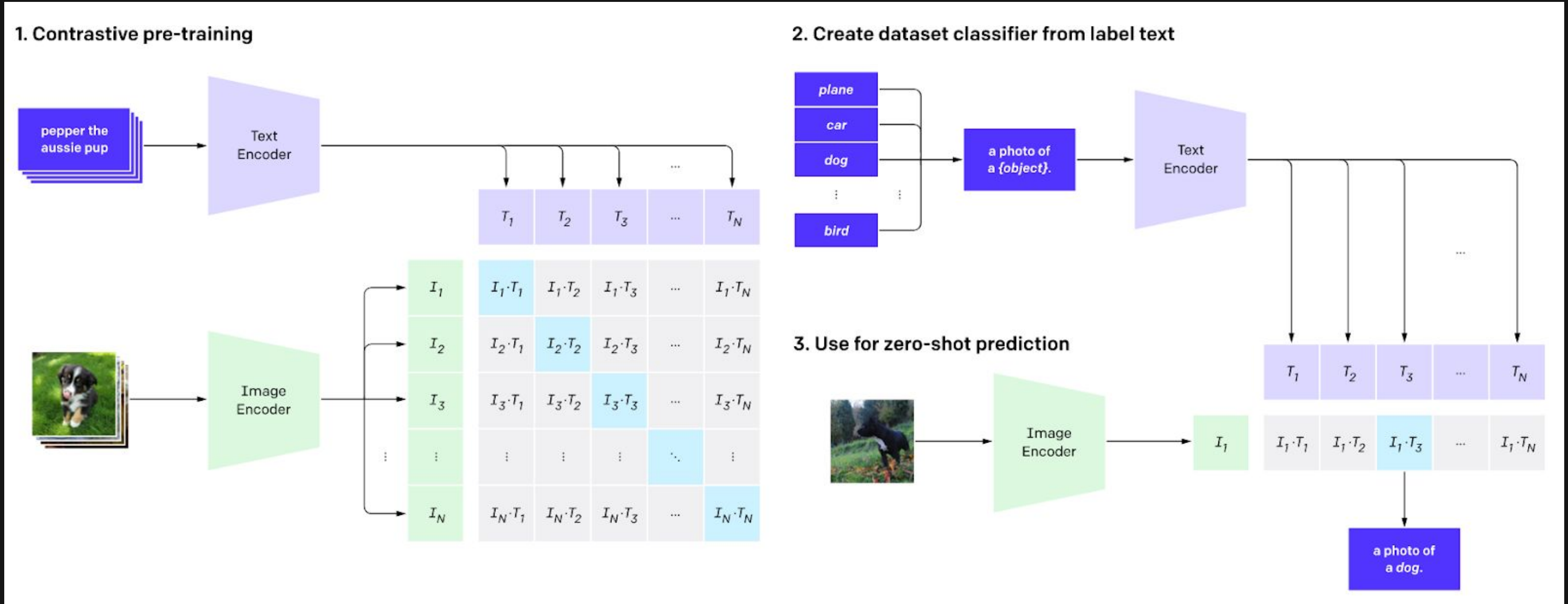
ClickHouse

Vector vs Embedding




Embedding for the word "moonlight"
0.0318, 0.0356, 0.0693 0.567, -0.0417, 0.9675
color temperature, directionality, mood, brightness, warmth, nighttime

Producing Embeddings - a lot more than text


1. Contrastive pre-training
pepper the aussie pup
Text Encoder
Image Encoder
T_1 T_2 T_3 ... T_N
I_1 I_2 I_3 I_N
$I_1 \cdot T_1$ $I_1 \cdot T_2$ $I_1 \cdot T_3$ ... $I_1 \cdot T_N$
$I_2 \cdot T_1$ $I_2 \cdot T_2$ $I_2 \cdot T_3$ ... $I_2 \cdot T_N$
$I_3 \cdot T_1$ $I_3 \cdot T_2$ $I_3 \cdot T_3$ ... $I_3 \cdot T_N$
$I_N \cdot T_1$ $I_N \cdot T_2$ $I_N \cdot T_3$ ... $I_N \cdot T_N$
2. Create dataset classifier from label text
plane
car
dog
bird
a photo of a {object}.
Text Encoder
3. Use for zero-shot prediction
Image Encoder
I_1
$I_1 \cdot T_1$ $I_1 \cdot T_2$ $I_1 \cdot T_3$ ... $I_1 \cdot T_N$
a photo of a dog.

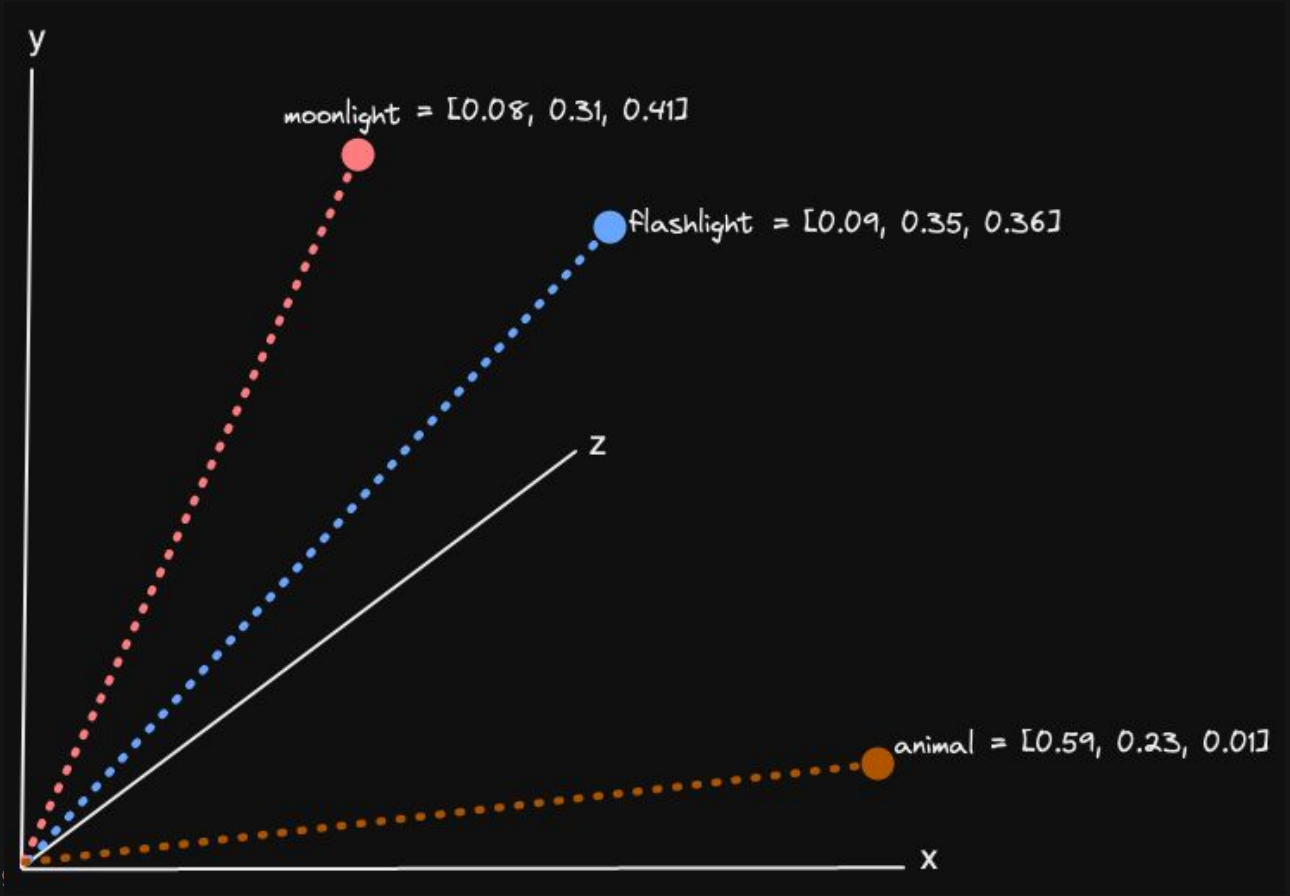
CLIP pre-trains an image encoder and a text encoder to predict which images were paired with which texts in our dataset. We then use this behavior to turn CLIP into a zero-shot classifier. We convert all of a dataset's classes into captions such as "a photo of a dog" and predict the class of the caption CLIP estimates best pairs with a given image.

Credit: CLIP - https://openai.com/research/clip

The importance of distance

ClickHouse

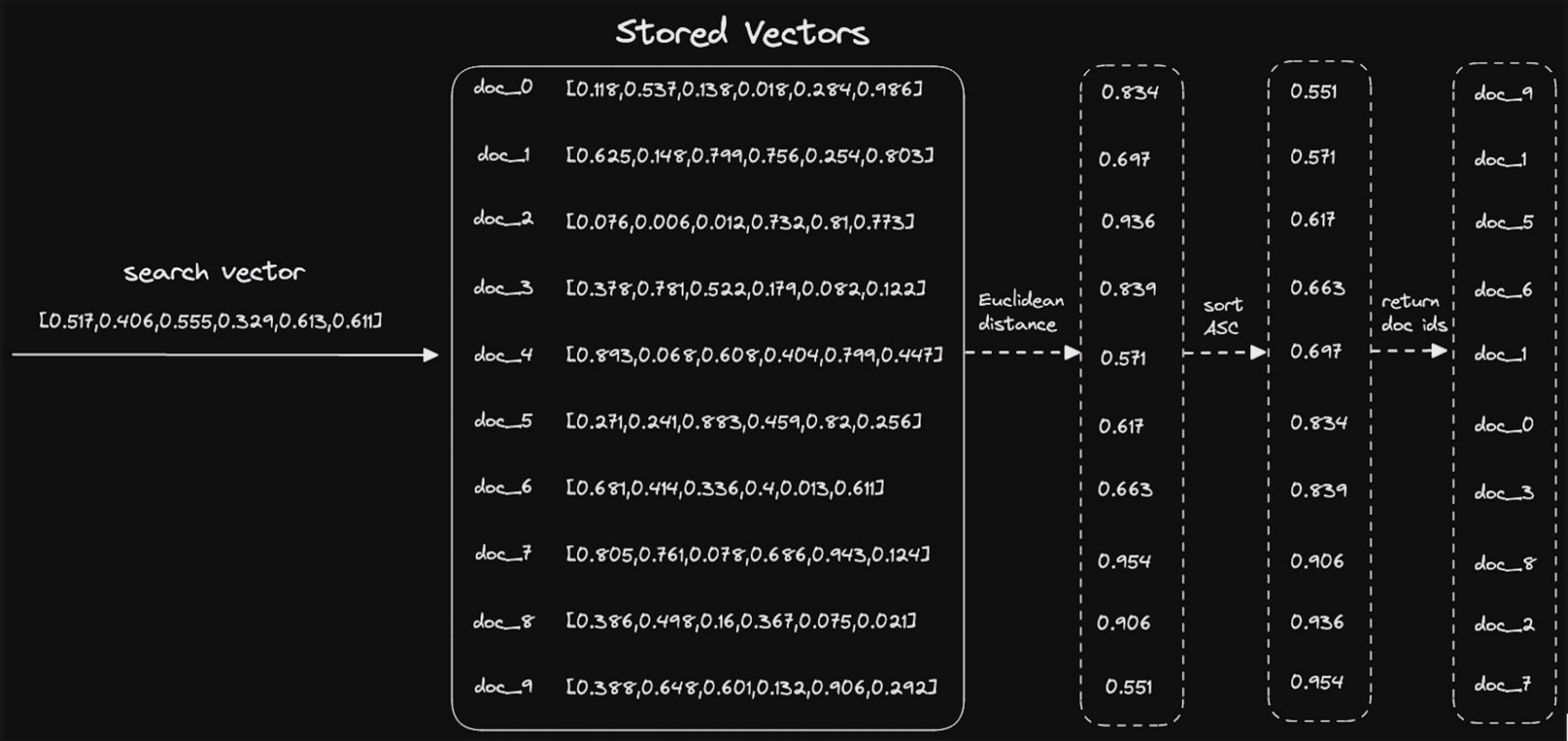
Similarity measured with distance

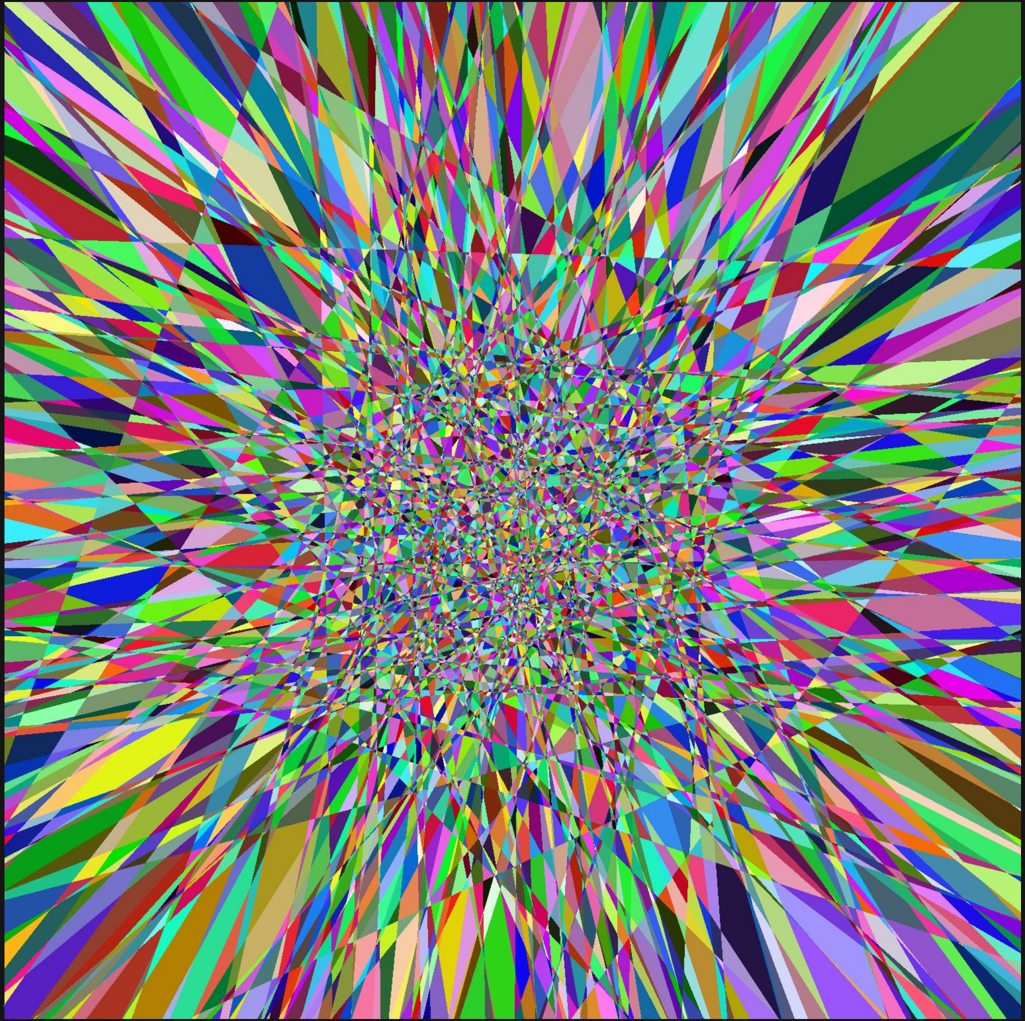

y
moonlight = [0.08, 0.31, 0.41]
flashlight = [0.09, 0.35, 0.36]
z
animal = [0.59, 0.23, 0.01]
x

Searching vectors

ClickHouse

Exact linear scan


Stored Vectors
search vector
[0.517,0.406,0.555,0.329,0.613,0.611]
doc_0 [0.118,0.537,0.138,0.018,0.284,0.986]
doc_1 [0.625,0.148,0.799,0.756,0.254,0.803]
doc_2 [0.076,0.006,0.012,0.732,0.81,0.773]
doc_3 [0.378,0.781,0.522,0.179,0.082,0.122]
doc_4 [0.893,0.068,0.608,0.404,0.799,0.447]
doc_5 [0.271,0.241,0.883,0.459,0.82,0.256]
doc_6 [0.681,0.414,0.336,0.4,0.013,0.611]
doc_7 [0.805,0.761,0.078,0.686,0.943,0.124]
doc_8 [0.386,0.498,0.16,0.367,0.075,0.021]
doc_9 [0.388,0.648,0.601,0.132,0.906,0.292]
Euclidean distance
0.834
0.697
0.936
0.839
0.571
0.617
0.663
0.954
0.906
0.551
sort ASC
0.551
0.571
0.617
0.663
0.697
0.834
0.839
0.906
0.936
0.954
return doc ids
doc_9
doc_1
doc_5
doc_6
doc_1
doc_0
doc_3
doc_8
doc_2
doc_7



Approximate Techniques

- Approximate Nearest Neighbour (ANN)
- Trade accuracy for performance
- Image is for Annoy algorithm
- Annoy builds a tree structure by partitioning the space into smaller and smaller hyperplanes for recursive evaluation

What problems can we solve?

ClickHouse

Possible Applications

01 **Recommendations**

Relevant to e-commerce websites. Beyond embedding text meaning into vectors, page views and past purchases can be encoded.

02 **Question Answering**

Equivalent meanings can, however, be encoded with vectors that are close, e.g., X and Y.

03 **Image and video search**

Search for images and videos based on text and vise versa

04 **Fraud Detection**

Find similar or dissimilar transactions by encoding users' behaviors or log-in patterns into vectors. These can be anomalous behaviors and prevent fraud.

05 **Providing Context**

Leveraging vector search to provide contextual content to chat applications powered by APIs like ChatGPT.

06 **Multilingual search**

Allow cross-language searching with the same concept in two languages encoded to the same vector.

Vector Search in ClickHouse

ClickHouse

- **Vectors are just Array(Float32) - 1.5x to 2x compression**
- **Matching is just a distance function - mainly Euclidean or Cosine but others supported**
- **Full SQL support and aggregations**
- **Approximate Nearest Neighbour supported through Annoy. Used to optimize distance functions.***
- **UDFs are useful for integrating embedding generation**

***Experimental.**

ClickHouse

```
 SELECT

     url,

     caption,

     L2Distance(image_embedding, [0.5736801028251648...0.2516217529773712]) AS score

 FROM laion WHERE similarity >= 0.2

 ORDER BY score ASC

 LIMIT 2

 FORMAT Vertical



10 rows in set. Elapsed: 1.603 sec. Processed 10.00 million rows, 32.70 GB (6.24 million rows/s.,
20.40 GB/s.)
```

When to use ClickHouse for Vector Search?

ClickHouse

ClickHouse for Vector Search

You have/need:

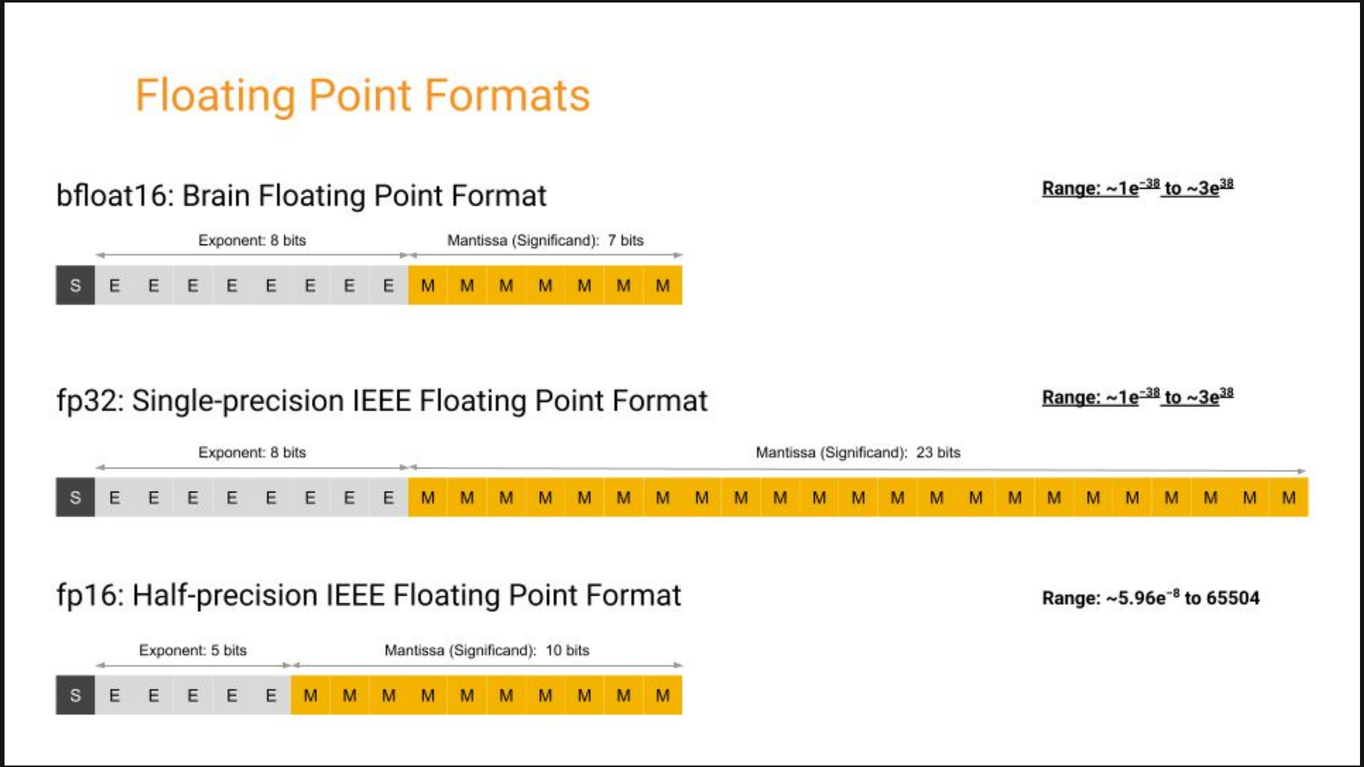
- High performance and scalable linear matching using many cores
- Metadata with your vectors for which you'll benefit from high compression and extremely fast querying/filtering
- Full SQL support in your querying inc. Joins and aggregations
- A very large dataset and do not wish to be memory bound or a small dataset where linear matching is fine (~millions)
- Data in ClickHouse and wish to complement with vectors
- An existing pipeline for generating vectors from a model and don't need tight coupling

Demo

LAION dataset

- 5 billion embeddings created with performance analysis in mind
- Created using a CLIP (Contrastive Language–Image Pre-training) multi-modal model. Allows images to be classified with a relevant category for previously unknown images not seen during the training process
- Trained on 400m image/caption pairs
- Each row has image metadata and a text + image embedding (768 dimensions)
- Allows searching for images with text and vise versa using Euclidean distance for matching
- We used a 2.2b english subset (4 billion vectors in total)
- We combined embeddings with metadata and made available in Parquet*

bfloat16


Floating Point Formats
bfloat16: Brain Floating Point Format
Range: ~1e-38 to ~3e38
Exponent: 8 bits
Mantissa (Significand): 7 bits
S E E E E E E E E M M M M M M M
fp32: Single-precision IEEE Floating Point Format
Range: ~1e-38 to ~3e38
Exponent: 8 bits
Mantissa (Significand): 23 bits
S E E E E E E E E M M M M M M M M M M M M M M M M M M M M M M M
fp16: Half-precision IEEE Floating Point Format
Range: ~5.96e-8 to 65504
Exponent: 5 bits
Mantissa (Significand): 10 bits
S E E E E E M M M M M M M M M M

```sql
INSERT INTO laion_bfloat16
SELECT
   arrayMap(x -> reinterpretAsFloat32(bitAnd(reinterpretAsUInt32(x),
4294901760)), image_embedding) AS image_embedding,
   arrayMap(x -> reinterpretAsFloat32(bitAnd(reinterpretAsUInt32(x),
4294901760)), text_embedding) AS text_embedding
FROM laion
```

END

ClickHouse

Speakers

Aaron Katz

CEO @ ClickHouse

Alexey Milovidov

CTO @ClickHouse

Yury Izrailevsky

President and
VP of Engineering

01

Section title with number

Author or subtitle

ClickHouse

Content heavy slide (2 columns)

Content slide - less text heavy

What is ClickHouse?

Open source

Developed since 2009

OSS 2016

28k+ Github stars

1k+ contributors

300+ releases

column-oriented

Best for aggregations

Files per column

Sorting and indexing

Background merges

distributed

Replication

Sharding

Multi-master

Cross-region

OLAP database

Analytics use cases

Aggregations

Visualizations

Mostly immutable data

What is ClickHouse? (highlighted)

Open source

- Developed since 2009
- OSS 2016
- 28k+ Github stars
- 1k+ contributors
- 300+ releases

column-oriented

- Best for aggregations
- Files per column
- Sorting and indexing
- Background merges

distributed

- Replication
- Sharding
- Multi-master
- Cross-region

OLAP database

- Analytics use cases
- Aggregations
- Visualizations
- Mostly immutable data

ClickHouse Journey

2009

The first ClickHouse prototype appeared to power internet scale web analytics

Jun 2016

On June 15th, ClickHouse was released under the Apache 2.0 licence

Aug 2021

ClcikHouse Inc incorporated in the US

Sep 2021

Closed first round of Series A financing ($50M USD) on Sept 14

Index Ventures BENCHMARK

Oct 2021

Closed follow-up Series B financing (250M USD)

ALTIMETER Lightspeed COATUE Redpoint.

May 2022

On June 15th, ClickHouse was released under the Apache 2.0 licence

Oct 2022

ClickHouse Cloud Public BETA

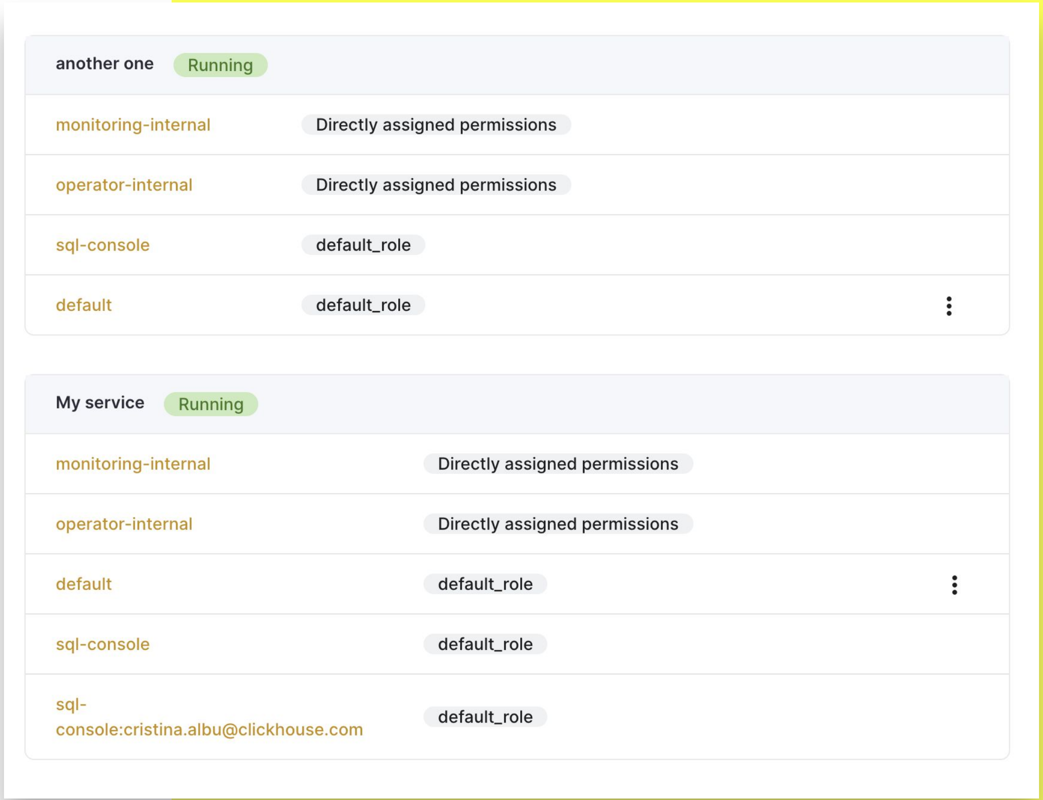
May 2022

ClickHouse cloud launches General Availability with 100+ customers and 1k+ users

ClickHouse

Feature highlight

- list item 1
- list item 2
- list item 3



another one Running

monitoring-internal	Directly assigned permissions
operator-internal	Directly assigned permissions
sql-console	default_role
default	default_role

My service Running

monitoring-internal	Directly assigned permissions
operator-internal	Directly assigned permissions
default	default_role
sql-console	default_role
sql-console:cristina.albu@clickhouse.com	default_role

Two cards highlight

Title here

Subtitle or a brief description

- Another bullet here
- Another bullet here
- Another bullet here
- Another bullet here

Title here

Subtitle or a brief description

- Another bullet here
- Another bullet here
- Another bullet here
- Another bullet here

Three cards highlight

Title here

Subtitle or a brief description

- Another bullet here
- Another bullet here
- Another bullet here
- Another bullet here

Title here

Subtitle or a brief description

- Another bullet here
- Another bullet here
- Another bullet here
- Another bullet here

Title here

Subtitle or a brief description

- Another bullet here
- Another bullet here
- Another bullet here
- Another bullet here

Use this slide for code (dark)

```
SELECT

    toStartOfMonth(upload_date) AS month,

    sum(view_count) AS `Youtube Views`,

    bar(sum(has_subtitles) / count(), 0.55, 0.7, 100) AS `% Subtitles`

FROM youtube

WHERE (month >= '2020-08-01') AND (month <= '2021-08-01')

GROUP BY month

ORDER BY month ASC


13 rows in set. Elapsed: 0.823 sec Processed 1.07 billion rows, 11.75 GB (1.30 billion rows/s., 14.27 GB/s.)
```

ClickHouse Cloud - from 0 to 1 in under a year

Private preview
May -July 2022

Serverless hosted ClickHouse for key design partners

- AWS with 3 regions initially
- Limited scalability within predefined limits
- Basic cloud console for ClickHouse service and user management
- Strong security and privacy with SOC 2 Type I compliance

Public Beta
Oct 2022

Serverless hosted ClickHouse for early adopters

- Ecosystem of first-party connectors to onboard and work with data
- Automatic scaling up and down for compute
- Integrated billing using pay as you go pricing model
- Enhanced security features such as Private Link, IP Filtering, Auditing

General Availability
Dec 2022

Serverless hosted ClickHouse for the broader market

- Enhanced cloud console for analytics and operational controls
- Support for more AWS regions and AWS marketplace billing
- Uptime SLA and additional operations tools
- Advanced security features with SOC 2 Type II compliance

"

Rokt has been an eager partner of ClickHouse as we modernize our analytics stack. By offloading operations to the experts our developers are focused on delivering the best experience possible while the business scales. We we are thrilled to see the path ClickHouse is forging.

ROKT

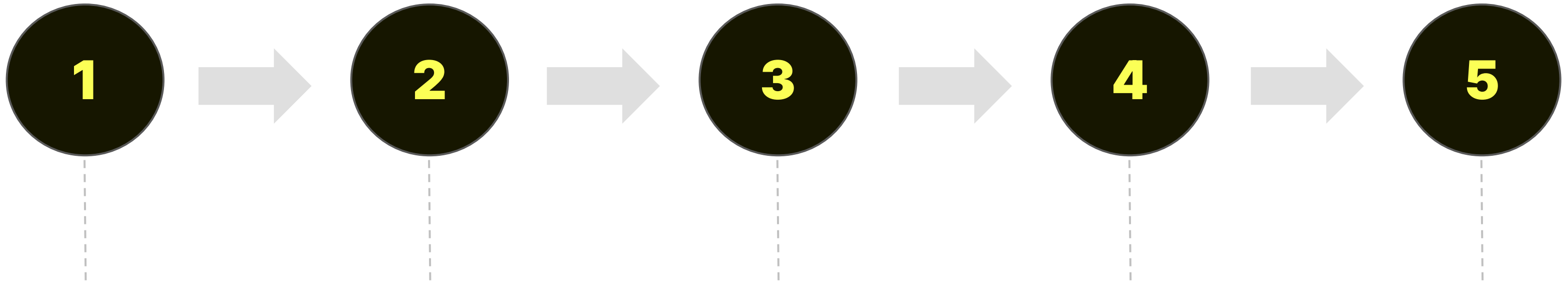
Attributor, role, company

Use this slide for code (light)

```
SELECT

    toStartOfMonth(upload_date) AS month,

    sum(view_count) AS `Youtube Views`,

    bar(sum(has_subtitles) / count(), 0.55, 0.7, 100) AS `% Subtitles`

FROM youtube

WHERE (month >= '2020-08-01') AND (month <= '2021-08-01')

GROUP BY month

ORDER BY month ASC


13 rows in set. Elapsed: 0.823 sec Processed 1.07 billion rows, 11.75 GB (1.30 billion rows/s., 14.27 GB/s.)
```

Process diagram, 5 ideas

Here you could describe the topic of the section, or not



Supporting text goes under the number

Supporting text goes under the number

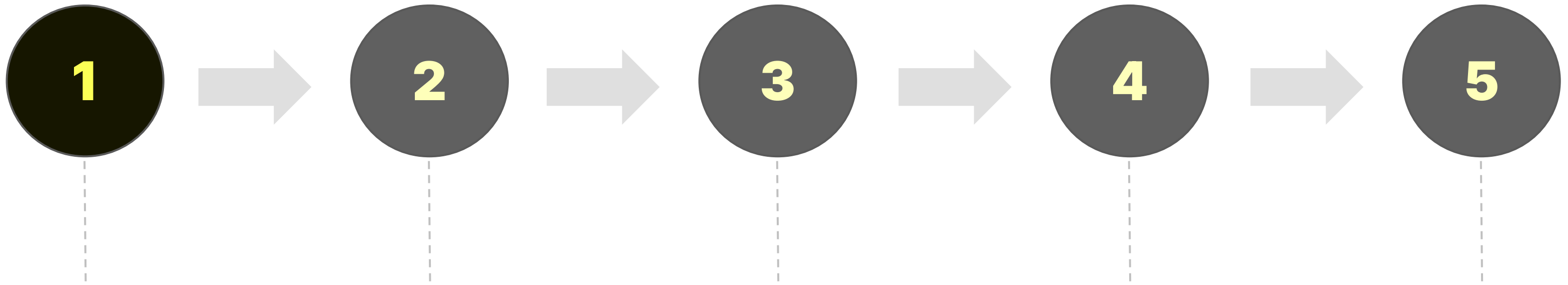
Supporting text goes under the number

Supporting text goes under the number

Supporting text goes under the number

Process diagram, 5 ideas

Highlighting one of the steps



Supporting text goes under the number

Supporting text goes under the number

Supporting text goes under the number

Supporting text goes under the number

Supporting text goes under the number

Big number treatment

2.3k

Header here

Supporting text goes here, under the header

1.2M

Header here

Supporting text goes here, under the header

45

Header here

Supporting text goes here, under the header

Features

Blazing fast

Uses all available hardware to its full potential to process each query as fast as possible. Peak processing performance for a single query stands at more than 2 terabytes per second.

Fault tolerant

Supports async replication and can be deployed across multiple datacenters. All nodes are equal, which allows avoiding having single points of failure.

Easy to use

ClickHouse is simple and works out-of-the-box. Simplifies data processing by instantly processing structured data using a user-friendly SQL dialect and eliminating non-standard API requirements.

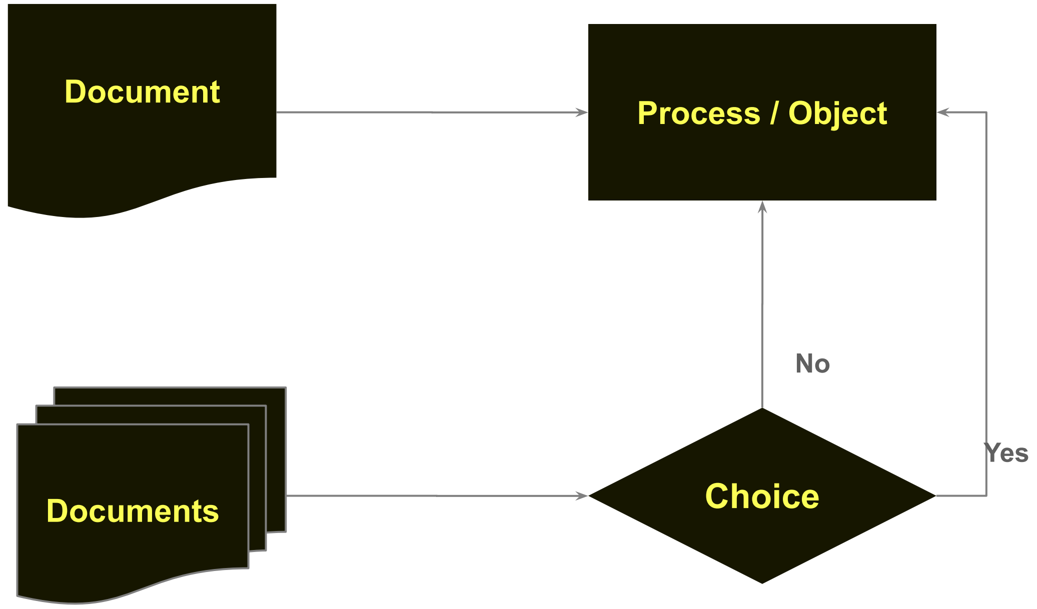
Highly reliable

Can be configured as a purely distributed system located on independent nodes, without any single points of failure. It also includes a lot of enterprise-grade security features and fail-safe mechanisms against human errors.

Table format

Header 1	Header 2	Header 3	Header 4	Header 5

For architecture/business process drawings



Icons for light background

Speakers (dark mode)

Aaron Katz

CEO @ ClickHouse

Alexey Milovidov

CTO @ClickHouse

Yury Izrailevsky

President and
VP of Engineering

Table of contents - many topics (dark mode)

Table of contents - fewer topics(dark mode)

Section title

Subtitle or description of the section.
Remove if not needed

ClickHouse

01 Section title

Subtitle or description of the section.
Remove if not needed

ClickHouse

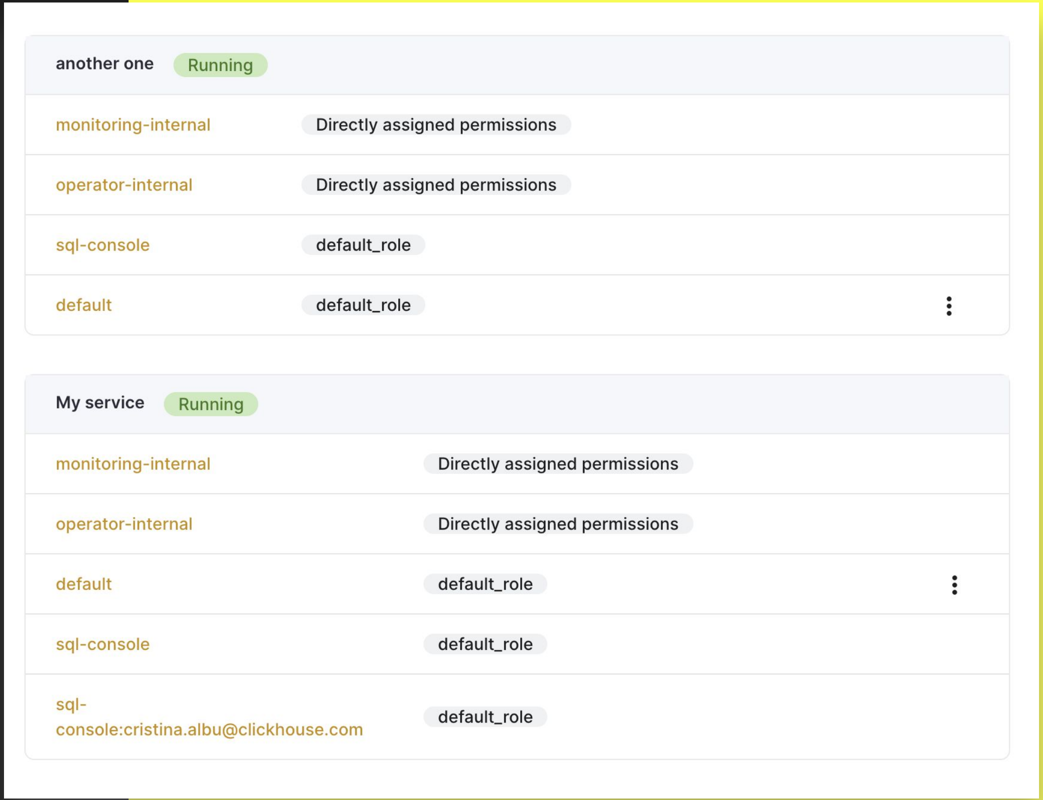
Content heavy slide

Content slide - less text heavy

ClickHouse

Feature highlight

- Item 1
- Item 2
- Item 3



another one Running

monitoring-internal	Directly assigned permissions
operator-internal	Directly assigned permissions
sql-console	default_role
default	default_role

My service Running

monitoring-internal	Directly assigned permissions
operator-internal	Directly assigned permissions
default	default_role
sql-console	default_role
sql-console:cristina.albu@clickhouse.com	default_role

Two cards highlight

Title here

Subtitle or a brief description

- Another bullet here
- Another bullet here
- Another bullet here
- Another bullet here

Title here

Subtitle or a brief description

- Another bullet here
- Another bullet here
- Another bullet here
- Another bullet here

Three cards highlight

Title here

Subtitle or a brief description

- Another bullet here
- Another bullet here
- Another bullet here
- Another bullet her

Title here

Subtitle or a brief description

- Another bullet here
- Another bullet here
- Another bullet here
- Another bullet here

Title here

Subtitle or a brief description

- Another bullet here
- Another bullet here
- Another bullet here
- Another bullet her

“

Rokt has been an eager partner of ClickHouse as we modernize our analytics stack. By offloading operations to the experts our developers are focused on delivering the best experience possible while the business scales. We we are thrilled to see the path ClickHouse is forging.

Attributor, role, company

Features

Blazing fast

Uses all available hardware to its full potential to process each query as fast as possible. Peak processing performance for a single query stands at more than 2 terabytes per second.

Fault tolerant

Supports async replication and can be deployed across multiple datacenters. All nodes are equal, which allows avoiding having single points of failure.

Easy to use

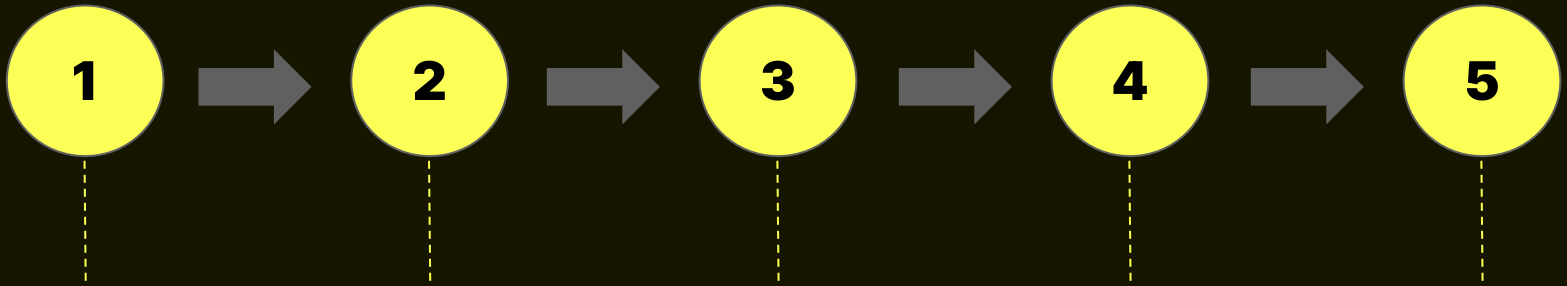
ClickHouse is simple and works out-of-the-box. Simplifies data processing by instantly processing structured data using a user-friendly SQL dialect and eliminating non-standard API requirements.

Highly reliable

Can be configured as a purely distributed system located on independent nodes, without any single points of failure. It also includes a lot of enterprise-grade security features and fail-safe mechanisms against human errors.

Process diagram, 5 ideas

Here you could describe the topic of the section, or not



Supporting text goes under the number

Supporting text goes under the number

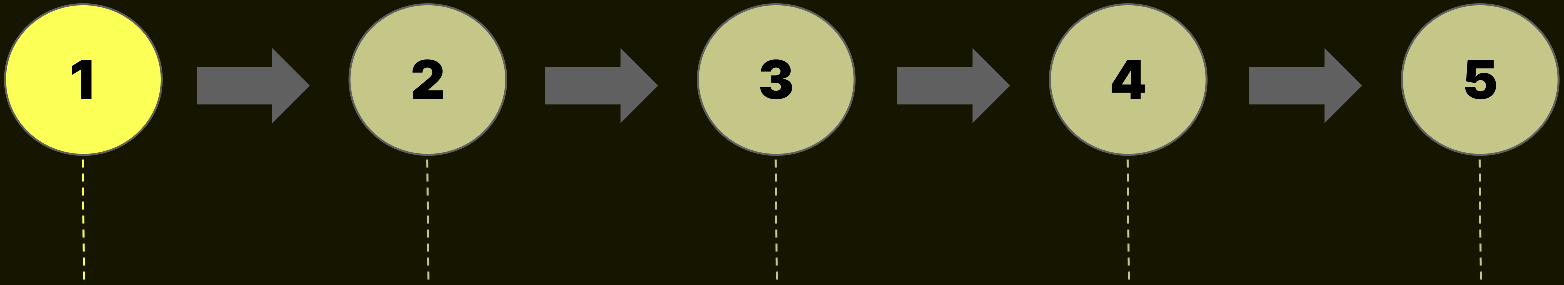
Supporting text goes under the number

Supporting text goes under the number

Supporting text goes under the number

Process diagram, 5 ideas

Highlighting one of the steps



Supporting text goes under the number

Supporting text goes under the number

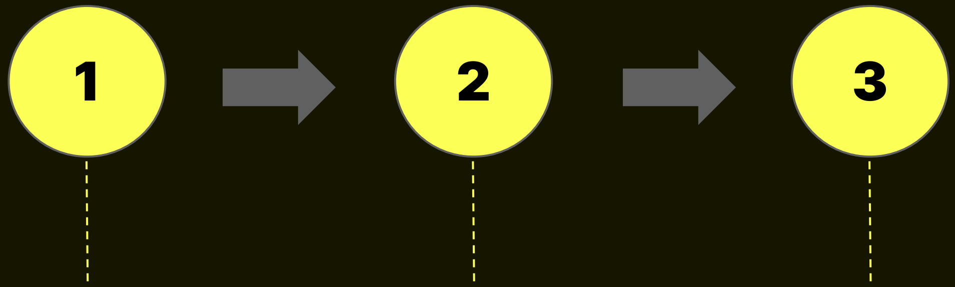
Supporting text goes under the number

Supporting text goes under the number

Supporting text goes under the number

Process diagram, 3 ideas

Here you could describe the topic of the section, or not



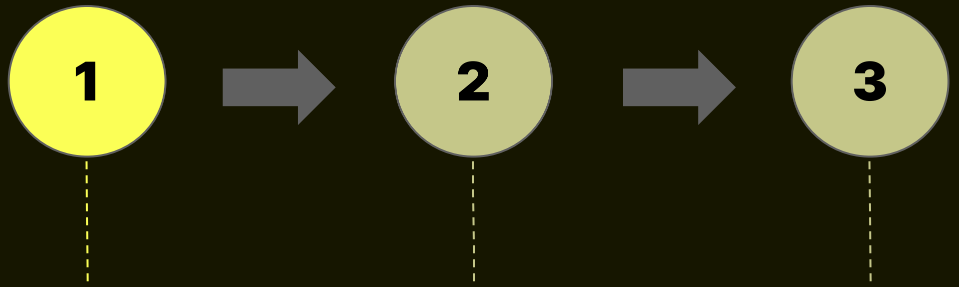
Supporting text goes under the number

Supporting text goes under the number

Supporting text goes under the number

Process diagram, 3 ideas

Highlighting one of the ideas



Supporting text goes under the number

Supporting text goes under the number

Supporting text goes under the number

Big number treatment

1.2k

Header here

Supporting text goes here, under the header

1M

Header here

Supporting text goes here, under the header

45

Header here

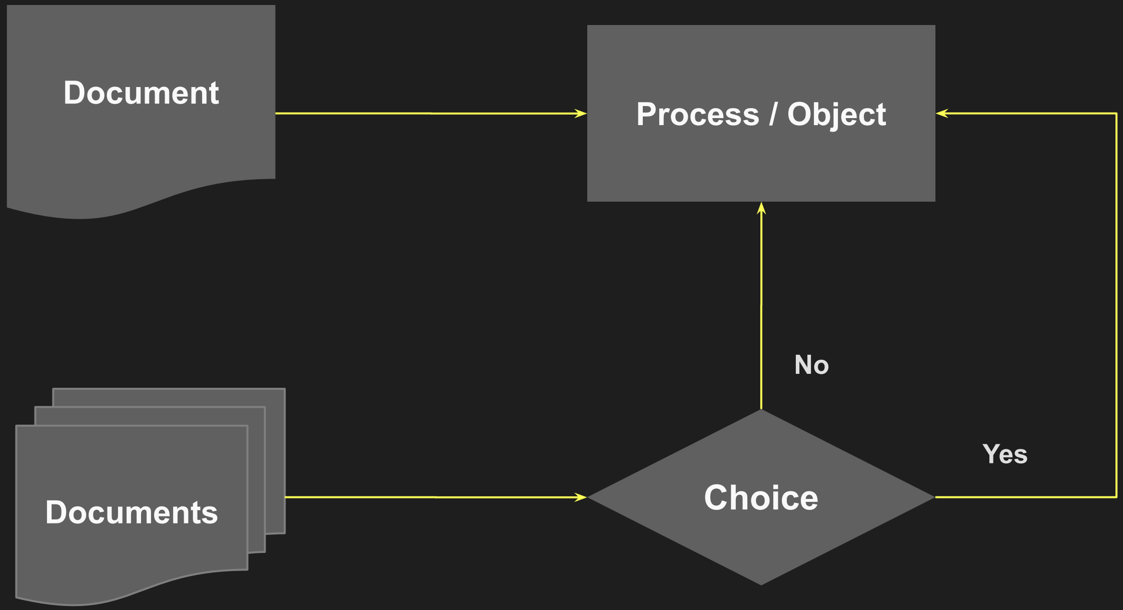
Supporting text goes here, under the header

Use this slide for code

Use template colors to highlight code. Feel free to remove this paragraph.

```
SELECT

    toStartOfMonth(upload_date) AS month,

    sum(view_count) AS `Youtube Views`,

    bar(sum(has_subtitles) / count(), 0.55, 0.7, 100) AS `% Subtitles`

FROM youtube

WHERE (month >= '2020-08-01') AND (month <= '2021-08-01')

GROUP BY month

ORDER BY month ASC


13 rows in set. Elapsed: 0.823 sec Processed 1.07 billion rows, 11.75 GB (1.30 billion rows/s., 14.27 GB/s.)
```

For architecture / business process drawings



Icons for dark background