Member-only story

A Comprehensive MongoDB Command Guide for Everyday Use

Muttineni Sai Rohith · Follow

Published in Dev Genius
6 min read · Mar 3, 2024

MongoDB is a popular and versatile NoSQL database system designed for efficient, scalable, and flexible data storage. It employs a document-oriented data model, using JSON-like documents (called BSON) to store data, making it easy to manage and query. MongoDB's key features include horizontal scalability through sharding, robust support for replication, and dynamic schema flexibility, making it well-suited for a wide range of applications and industries.

MongoDB

To understand in detail, let's compare between MongoDB and Relational Databases —

Aspect	MongoDB	SQL (Relational Databases)
Data Model	Document-oriented (JSON-like BSON documents)	Table-based (Rows and Columns)
Schema	Dynamic and flexible schema	Fixed and structured schema
Query Language	JSON-based queries using MongoDB Query Language	SQL (Structured Query Language)
Scaling	Horizontal scaling through sharding	Vertical scaling by adding more powerful servers
Transactions	Supports atomic operations within a document	Supports ACID transactions across multiple tables
Join Operations	Limited support for join operations	Strong support for complex join operations
Data Relationships	Embedded documents or references	Relationships established using foreign keys
Consistency	Eventual consistency	Strong consistency
Complex Transactions	Limited support for multi-document transactions	Supports complex transactions
Data Integrity	Less rigid integrity constraints	Enforces strict integrity constraints
Scalability	Horizontal scaling with ease	Vertical scaling, can be more complex
Example Terminology	Collection (like table)	Table
	Document (like a row)	Row
	Field (like a column)	Column (Attribute)

Comparison between MongoDB and Relational Databases

MongoDB Command sheet —

1. Connect to MongoDB:

```
mongo
```

2. Show Databases:

```
show databases
```

3. Switch to a Database:

```
use <database_name>
```

4. Show Collections in a Database:

```
show collections
```

5. Create collection:

In MongoDB, we don't explicitly create collections; they are created automatically when we insert data. However, we can use the `createCollection` command to pre-allocate space and create a collection.

```
db.createCollection("your_collection_name")
```

Additionally, you can insert a document into the collection, and MongoDB will create the collection if it doesn't exist.

```
db.your_collection_name.insert({ key: "value" })
```

6. Insert data into the collection:

To insert data into a collection in MongoDB, we can use the `insertOne` or `insertMany` method. Here's an example using the `insertOne` method:

```
db.your_collection_name.insertOne({
  key1: "value1",
  key2: "value2",
  // Add more key-value pairs as needed
```

```
})
```

If we want to insert multiple documents at once, we can use the `insertMany` method:

```
db.your_collection_name.insertMany([
  {
    key1: "value1",
    key2: "value2",
    // Add more key-value pairs as needed
  },
  {
    key1: "value3",
    key2: "value4",
    // Add more key-value pairs as needed
  },
  // Add more documents as needed
])
```

7. Query documents:

Find all the documents —

```
db.your_collection_name.find()
```

Find documents meeting a specific condition —

```
db.your_collection_name.find({ key: "value" })
```

Limit the Number of Documents Returned:

```
db.your_collection_name.find().limit(5)
```

Projection — Works similar to select —

```
db.your_collection_name.find({ key: "value" }, { _id: 0, key: 1 })
```

Sorting Documents (Use -1 for descending order):

```
db.your_collection_name.find().sort({ key: 1 })
```

Skipping Documents:

```
db.your_collection_name.find().skip(5)
```

8. Update documents —

To update documents in MongoDB, we can use the `updateOne` or `updateMany` method. Here are examples for both scenarios:

1. Update a Single Document:

```
db.your_collection_name.updateOne(
  { key: "value" },    // Query criteria
  { $set: { newKey: "newValue" } }  // Update operation
)
```

2. Update Multiple Documents:

```
db.your_collection_name.updateMany(
  { key: "value" },    // Query criteria
  { $set: { newKey: "newValue" } }  // Update operation
)
```

MongoDB provides various update operators that offer flexibility in updating documents. Additionally, the `upsert` option allows us to insert a new document if no matching document is found based on the query criteria —

1. Update with `$set` Operator:

```
db.your_collection_name.updateOne(
  { key: "value" },
  { $set: { newKey: "newValue" } }
)
```

2. Increment a Numeric Field with `$inc` Operator:

```
db.your_collection_name.updateOne(
  { key: "value" },
  { $inc: { numericField: 5 } }
)
```

Similarly, we have $mul, $min, and $max operators.

3. Add an Element to an Array with `$push` Operator:

```
b.your_collection_name.updateOne(
  { key: "value" },
  { $push: { arrayField: "newElement" } }
)
```

4. Remove an Element from an Array with `$pull` Operator:

```
db.your_collection_name.updateOne(
  { key: "value" },
  { $pull: { arrayField: "elementToRemove" } }
)
```

5. Pop the First or Last Element from an Array with `$pop` Operator:

```
db.your_collection_name.updateOne(
  { key: "value" },
  { $pop: { arrayField: 1 } }
)
```

6. Current Date with `$currentDate` Operator:

```
db.your_collection_name.updateOne(
  { key: "value" },
  { $currentDate: { lastModified: true, "timestampField": { $type: "date" } } }
)
```

7. Bitwise Operators with `$bit` Operator:

```
db.your_collection_name.updateOne(
  { key: "value" },
  { $bit: { flags: { and: 2, or: 4, xor: 1 } } }
)
```

This performs bitwise AND, OR, and XOR operations on the `flags` field.

8. Rename a Field with `$rename` Operator:

```
db.your_collection_name.updateOne(
  { key: "value" },
  { $rename: { oldKey: "newKey" } }
)
```

9. Upsert Option (Insert if Not Found):

```
db.your_collection_name.updateOne(
  { key: "nonexistentValue" },
  { $set: { newKey: "newValue" } },
  { upsert: true }
)
```

10. Set the Value Conditionally with `$set` and `$cond` Operator — (works as update and where condition in RDBMS)

```
db.your_collection_name.updateOne(
  { key: "value" },
  { $set: { newField: { $cond: { if: { $gt: ["$numericField", 0] }, then: "Posi
)
```

11. Set a Field's Value to the Result of an Expression with `$expr` Operator:

```
db.your_collection_name.updateOne(
  { key: "value" },
  { $set: { newField: { $expr: { $gt: ["$numericField", 0] } } } }
)
```

9. Delete Collections, Documents and Databases —

To delete documents in MongoDB, we can use the `deleteOne` or `deleteMany` method.

1. Delete a Single Document:

```
db.your_collection_name.deleteOne({ key: "value" })
```

2. Delete Multiple Documents:

```
db.your_collection_name.deleteMany({ key: "value" })
```

3. Delete a Collection:

```
db.your_collection_name.drop()
```

4. Delete a Database:

```
use your_database_name
db.dropDatabase()
```

10. Creating Indexes:

Indexes can significantly improve query performance. We can create an index using the below code —

```
db.your_collection_name.createIndex({ key: 1 })
```

This command creates an ascending index on the specified key in the collection named “your_collection_name”. The `1` indicates ascending order; use `-1` for descending order.

We can also create compound indexes by specifying multiple keys:

```
db.your_collection_name.createIndex({ key1: 1, key2: -1 })
```

Although indexes can significantly improve query performance, they come with trade-offs, such as increased storage and slower write operations. Therefore, we should consider the specific use cases and queries when creating indexes.

11. Aggregation:

To learn about Aggregation refer to the below article —

All about MongoDB Aggregation command and queries

MongoDB Aggregation Framework is a powerful set of tools for processing and transforming documents in a collection. It...

muttinenisairohith.medium.com

12. Shard a Collection:

```
sh.shardCollection("<database_name>.<collection_name>", { shard_key: 1 }
```

To learn more about sharding refer to below article —

MongoDB Sharding | Sharding vs Replication

As the size of the data increases, a single machine may not be sufficient to store the data nor provide an acceptable...

muttinenisairohith.medium.com

13. Check Sharding Status:

```
sh.status()
```

14. Enable Sharding for a Database:

```
sh.enableSharding("<database_name>"
```

15. Add Shard to Cluster:

```
sh.addShard("shard_address")
```

16. Check Replica Set Status:

```
rs.status()
```

To learn more about Replication refer to below article —

All About MongoDB Replication

Replication is pivotal for database systems due to its role in ensuring high availability and fault tolerance. By...

muttinenisairohith.medium.com

17. Initiate Replica Set:

```
rs.initiate()
```

18. Add Member to Replica Set:

```
rs.add("new_member_address")
```

19. Remove Member from Replica Set:

```
rs.remove("member_address")
```

20. Configure Write Concern:

```
db.getMongo().setWriteConcern("majority")
```

21. Backup Database:

```
mongodump --db <database_name> --out <backup_directory>
```

22. Restore Database:

```
mongorestore --db <database_name> <backup_directory>/<database_name>
```

23. MongoDB Regex:

```
db.your_collection_name.find({ key: { $regex: /pattern/ } })
```

This finds documents where the `key` field matches the specified regular expression pattern.

24. MongoDB Text Search:

Text search in MongoDB allows us to perform full-text searches on string content. MongoDB provides a powerful text search feature that includes language-specific stemming, stop words, and relevance scoring.

1. **Creating a Text Index:**
   Before we can perform text searches, we need to create a text index on one or more fields. For example:

```
db.your_collection_name.createIndex({ content: "text" })
```

This command creates a text index on the `content` field in the specified collection.

2. Performing a Text Search:
Once the text index is created, we can perform text searches using the `$text` operator. For example:

```
db.your_collection_name.find({ $text: { $search: "keyword" } })
```

This query finds documents where the `content` field contains the specified "keyword."

3. Relevance Scoring:
MongoDB assigns a relevance score to each document based on the frequency of the search term in the indexed fields. We can include the relevance score in the query result:

```
db.your_collection_name.find({ $text: { $search: "keyword" } }, { score: { $met
```

4. Language-Specific Stemming:
MongoDB's text search supports language-specific stemming, meaning it considers different grammatical forms of a word as the same. We can specify a language for stemming:

```
db.your_collection_name.find({ $text: { $search: "important", $language: "en" }
```

This query performs a text search with English language-specific stemming.

5. Stop Words:
MongoDB's text search excludes common stop words (e.g., "and," "the," "is") by default. We can customize the list of stop words if needed.

6. Case Insensitivity:
Text search in MongoDB is case-insensitive by default. We can perform a case-sensitive search by using the `$caseSensitive` option:

```
db.your_collection_name.find({ $text: { $search: "Keyword", $caseSensitive: tru
```

Text search in MongoDB is a powerful tool for finding relevant documents based on textual content. It's especially useful in scenarios where we need to search through large volumes of text data.

Conclusion: Acting as a handbook, this article covers most of the queries related to MongoDB. I hope this will be helpful.

Happy Learning...

Mongodb | Data Science | Data Engineering | Database | NoSQL

Follow

Published in Dev Genius

25K Followers · Last published 1 day ago

Coding, Tutorials, News, UX, UI and much more related to development

Follow

Written by Muttineni Sai Rohith

746 Followers · 326 Following

Senior Data Engineer with experience in Python, Pyspark and SQL! Reach me at sairohith.muttineni@gmail.com

No responses yet

What are your thoughts?

Respond

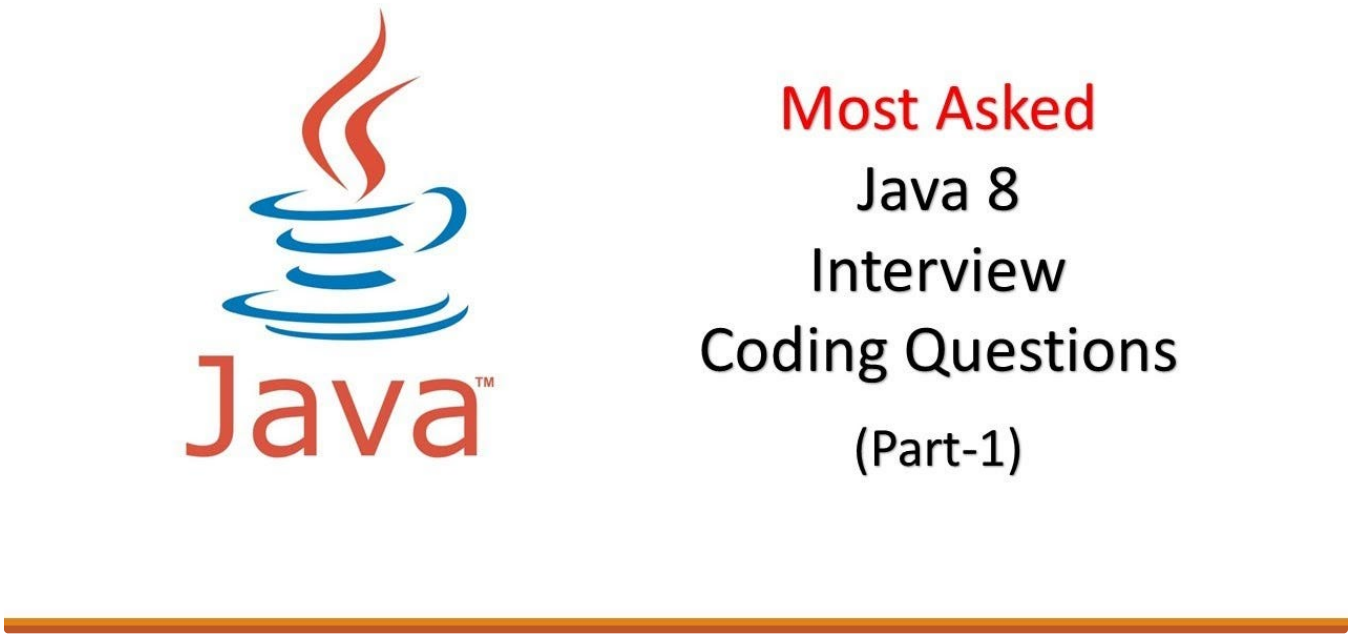
More from Muttineni Sai Rohith and Dev Genius

In Dev Genius by Muttineni Sai Rohith

Understanding Data Partitioning in Pyspark

In the world of big data processing, efficiency is king. When dealing with terabytes or even petabytes of data, even small inefficiencies...

Jan 2 11



In Dev Genius by Anusha SP

Java 8 Coding and Programming Interview Questions and Answers

It has been 8 years since Java 8 was released. I have already shared the Java 8 Interview Questions and Answers and also Java 8 Stream API...

Jan 31, 2023 1.4K 19

In Dev Genius by Aleksei Aleinikov

10 Ways to Work with Large Files in Python: Effortlessly Handle Gigabytes of Data!

Handling large text files in Python can feel overwhelming. When files grow into gigabytes, attempting to load them into memory all at once...

Dec 1, 2024 713 7

In CodeX by Muttineni Sai Rohith

Start Using Salting Technique in Pyspark

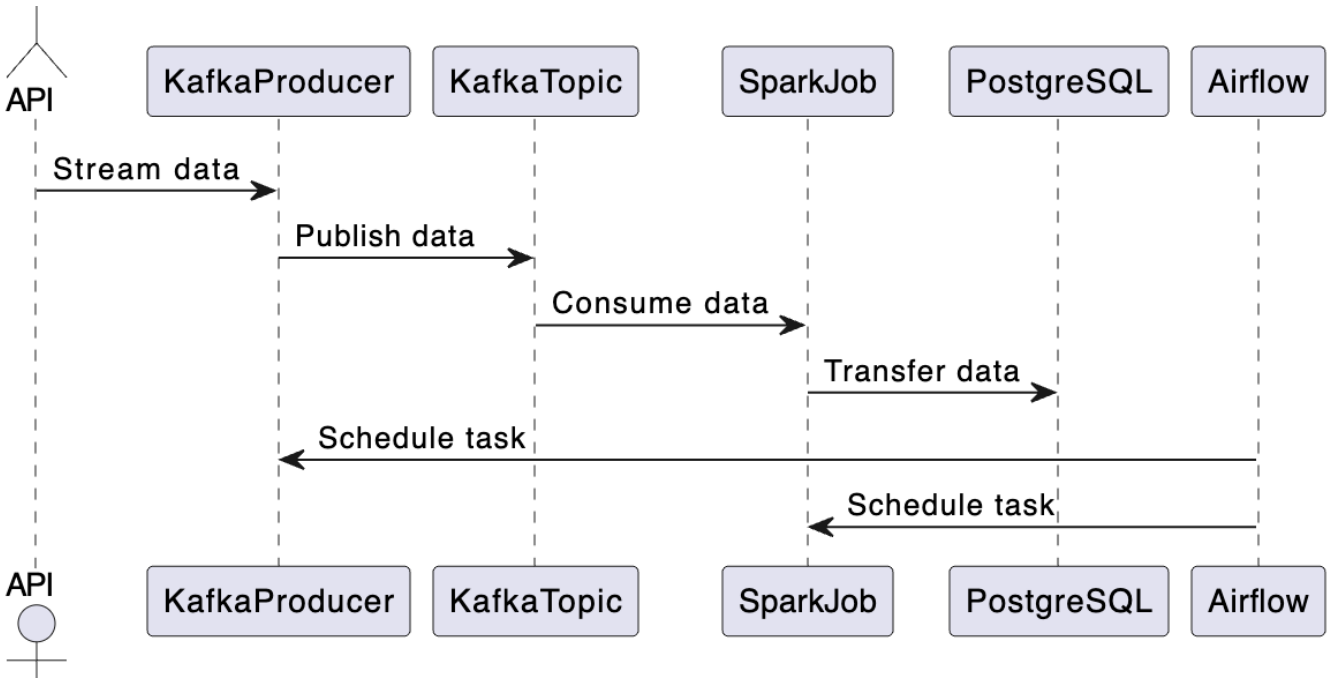
In the world of distributed computing, performance bottlenecks are a common challenge. A particularly tricky issue in PySpark (or any...

Jan 3 6 1

See all from Muttineni Sai Rohith

See all from Dev Genius

Recommended from Medium



tds In Towards Data Science by Hamza Gharbi

End-to-End Data Engineering System on Real Data with Kafka, Spark, Airflow, Postgres, and Docker

Building a Practical Data Pipeline with Kafka, Spark, Airflow, Postgres, and Docker

Jan 19, 2024 1.6K 14

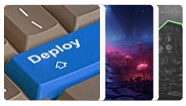
In Level Up Coding by Anna Geller

2025 Data Engineering & AI Trends

How GenAI, new data regulations, Postgres, DuckDB, and open table formats affect data engineering in 2025 and beyond

Jan 24 250 3

Lists



Predictive Modeling w/ Python

20 stories · 1811 saves

Practical Guides to Machine Learning

10 stories · 2185 saves

Coding & Development

11 stories · 991 saves

data science and AI

40 stories · 324 saves

Archana Goyal

Case Study: How PayPal Scaled to a Billion Daily Transactions with Just 8 Virtual Machines

My articles are open to everyone; non-member readers can read the full article by clicking this link.

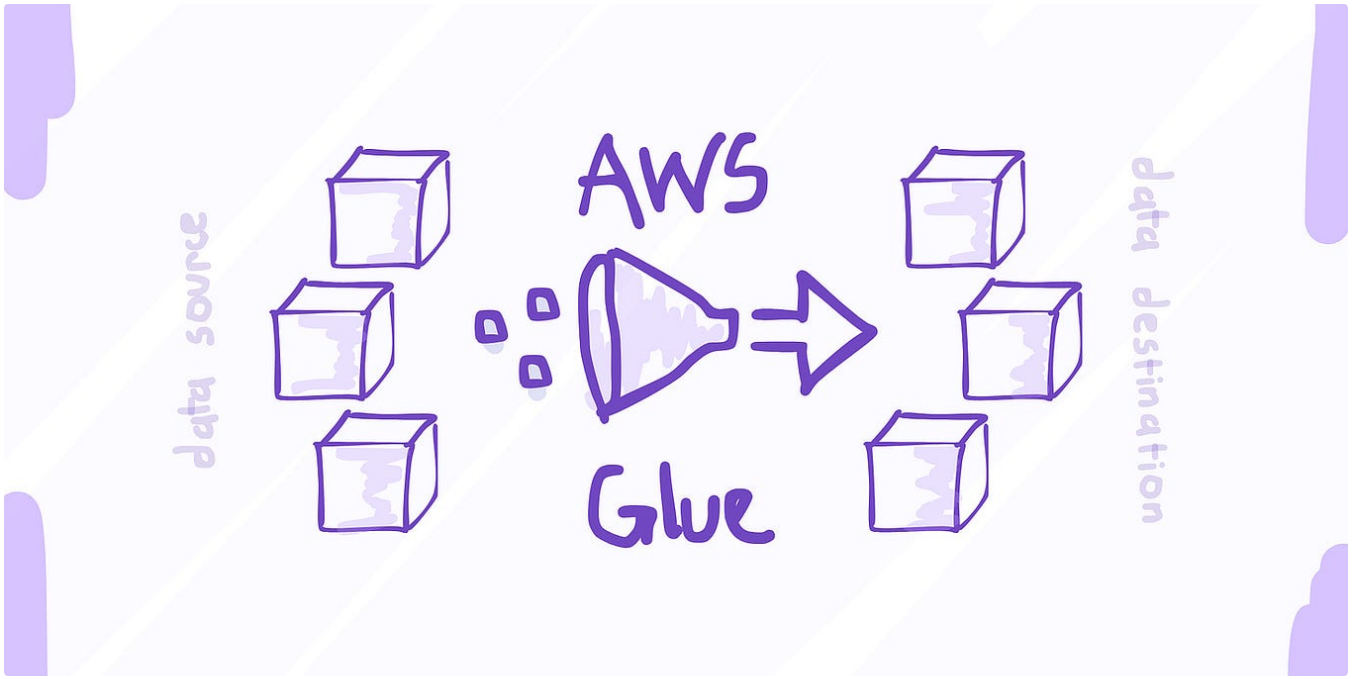
Nov 15, 2024 102 4

In Generative AI by Jim Clyde Monge

How To Install And Use DeepSeek R-1 In Your Local PC

Here's a step-by-step guide on how you can run DeepSeek R-1 on your local machine even without internet connection.

Jan 24 2.5K 54

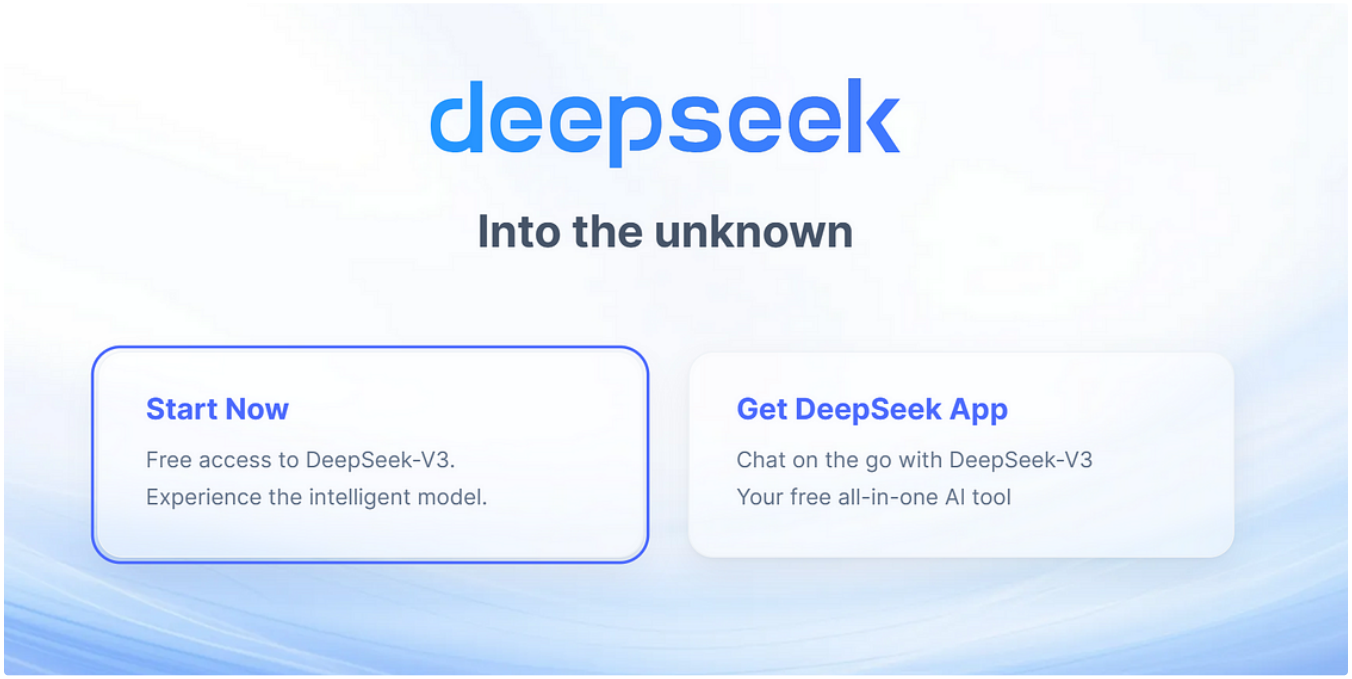


In Data Engineer Things by Vu Trinh

I spent 6 hours learning AWS Glue. Here is what I found

The cloud-native and robust data integration tool.

Jan 23 90



In Data Science in your pocket by Mehul Gupta

DeepSeek is highly biased, don’t use it

DeepSeek has taken the Generative AI arena by storm. Both their models, be it DeepSeek-v3 or DeepSeek-R1 have outperformed SOTA models by a...

Jan 25 3.2K 316

See more recommendations