

[Open in app](#)

★ Member-only story

Elastic Search — Day 3: Searching and Querying!!



Navya Cloudops · Following

8 min read · Feb 9, 2024



Listen



Share



More

Welcome back to Day 3 of our 10-day DevOps journey into Elasticsearch! Today, we delve into the intricate world of Elasticsearch Searching and Querying. In the previous days, we laid the groundwork, understanding Elasticsearch's architecture and setup. Today, we unlock the power of Elasticsearch's search capabilities.



Basic search concepts:

Basic search concepts in Elasticsearch form the foundation of its search functionality, enabling users to efficiently retrieve relevant data from their indices. Here's a detailed overview:

1. Indexing:

Indexing is the process of storing your data in Elasticsearch in a structured format. Elasticsearch organizes data into indices, which are logical collections of documents that share similar characteristics. Each document represents a single data entity and is stored in JSON format.

- **Index:** An index is like a database in traditional relational databases. It is a collection of documents that have somewhat similar characteristics.
- **Document:** A document is a basic unit of information that can be indexed. It is represented in JSON format and contains data fields with corresponding values.

For example, if you're indexing documents related to books, each document might contain fields like title, author, publication year, and description.

2. Querying:

Once your data is indexed, you can perform searches to retrieve specific documents or aggregate data based on certain criteria. Elasticsearch provides a powerful Query DSL (Domain Specific Language) for constructing queries.

- **Query DSL:** Elasticsearch's Query DSL is a JSON-based language used to define queries. It offers a wide range of query types and options for filtering, aggregating, and sorting data.

3. Searching:

Searching in Elasticsearch involves retrieving documents that match specific criteria or conditions. Elasticsearch supports various types of searches, including:

- **Match Query:** Searches for documents that contain a specified term or terms.
- **Term Query:** Searches for documents that contain an exact term in a specified field.
- **Range Query:** Searches for documents with fields that contain values within a specified range.
- **Bool Query:** Allows combining multiple queries using boolean logic (AND, OR, NOT).

4. Aggregations:

Aggregations in Elasticsearch enable you to perform analysis on your data and

derive insights. Aggregations can be used to calculate metrics, generate histograms, and more.

- **Terms Aggregation:** Groups documents based on the values of a specified field.
- **Histogram Aggregation:** Divides numeric data into fixed or variable-sized intervals (buckets) and provides counts for each interval.
- **Date Histogram Aggregation:** Similar to histogram aggregation but used specifically for date fields.

5. Relevance Scoring and Boosting:

Elasticsearch calculates a relevance score for each document returned by a query, based on how well it matches the search criteria. The relevance score is influenced by factors such as term frequency, field length, and inverse document frequency.

- **Relevance Boosting:** Allows you to increase or decrease the importance of certain fields or documents in the relevance calculation. For example, you can boost documents with specific terms in the title field to make them more relevant in search results.

Understanding these basic search concepts in Elasticsearch is essential for effectively querying and retrieving data from your indices.

Query DSL (Domain Specific Language) fundamentals:

It is a fundamental component of Elasticsearch that allows users to construct complex queries to retrieve specific documents or perform advanced search operations. Query DSL is expressed in JSON format and provides a flexible and powerful way to define search criteria and filters.

Here's an elaboration on Query DSL fundamentals:

1. Query Types:

Elasticsearch supports various types of queries that can be used individually or combined to create sophisticated search criteria. Some common query types include:

- **Match Query:** Searches for documents that contain a specified term or terms. It analyzes the input text and performs a full-text search.

```
{
  "query": {
    "match": {
      "title": "elasticsearch"
    }
  }
}
```

- **Term Query:** Searches for documents that contain an exact term in a specified field. Unlike the match query, it does not analyze the input text.

```
{
  "query": {
    "term": {
      "status": "published"
    }
  }
}
```

- **Range Query:** Searches for documents with fields that contain values within a specified range.

```
{
  "query": {
    "range": {
      "price": {
        "gte": 100,
        "lte": 500
      }
    }
  }
}
```

- **Bool Query:** Allows combining multiple queries using boolean logic (AND, OR, NOT).

```
{
  "query": {
    "bool": {
      "must": [
        { "match": { "title": "elasticsearch" } },
        { "match": { "author": "John Doe" } }
      ]
    }
  }
}
```

2. Filters:

Filters are used to narrow down search results based on specific criteria. Unlike queries, filters do not affect the relevance score of documents.

- **Term Filter:** Filters documents that contain an exact term in a specified field.
- **Range Filter:** Filters documents with fields that contain values within a specified range.
- **Bool Filter:** Allows combining multiple filters using boolean logic.

Filters are commonly used for exact matches, date ranges, and other criteria that don't require relevance scoring.

3. Aggregations:

Aggregations allow you to perform analysis on your data and generate summaries, histograms, and other statistical information.

- **Terms Aggregation:** Groups documents based on the values of a specified field.
- **Date Histogram Aggregation:** Divides date values into intervals and provides counts for each interval.
- **Range Aggregation:** Similar to range query but used for aggregation purposes.

Aggregations are useful for generating insights from your data and understanding patterns and trends.

Understanding Query DSL fundamentals empowers users to create precise and efficient search queries in Elasticsearch. By mastering Query DSL, you can leverage

Elasticsearch's full potential to retrieve, filter, and analyze data effectively.

Performing simple searches and aggregations:

It allows you to retrieve and analyze data efficiently. Let's delve into each aspect:

Performing Simple Searches:

Simple searches in Elasticsearch involve retrieving documents that match specific criteria or conditions. Here's how you can perform simple searches using Elasticsearch:

Match Query:

The **'match'** query is one of the simplest and most commonly used queries in Elasticsearch. It searches for documents that contain a specified term or terms. Elasticsearch analyzes the input text to generate terms before performing the search.

```
{
  "query": {
    "match": {
      "title": "elasticsearch"
    }
  }
}
```

This query retrieves documents where the **'title'** field contains the term "elasticsearch".

Term Query:

The **'term'** query is used to search for documents that contain an exact term in a specified field. Unlike the **'match'** query, the **'term'** query does not analyze the input text.

```
{
  "query": {
    "term": {
      "status": "published"
    }
  }
}
```

```
}  
}
```

This query retrieves documents where the 'status' field exactly matches "published".

Performing Aggregations:

Aggregations in Elasticsearch allow you to perform analysis on your data and generate summaries, histograms, and other statistical information. Here are some commonly used aggregations:

Terms Aggregation:

The 'terms' aggregation groups documents based on the values of a specified field. It then provides counts for each unique value or term.

```
{  
  "aggs": {  
    "top_authors": {  
      "terms": {  
        "field": "author.keyword"  
      }  
    }  
  }  
}
```

This aggregation groups documents by the 'author' field and provides counts for each unique author.

Date Histogram Aggregation:

The 'date_histogram' aggregation divides date values into intervals (e.g., days, months, years) and provides counts for each interval.

```
{  
  "aggs": {  
    "posts_per_month": {  
      "date_histogram": {  
        "field": "published_date",  
        "calendar_interval": "month"  
      }  
    }  
  }  
}
```

```
}  
}
```

This aggregation groups documents by the ‘**published_date**’ field and provides counts for each month.

Range Aggregation:

The ‘**range**’ aggregation groups documents into specified ranges based on numeric or date fields and provides counts for each range.

```
{  
  "aggs": {  
    "price_ranges": {  
      "range": {  
        "field": "price",  
        "ranges": [  
          { "from": 0, "to": 100 },  
          { "from": 100, "to": 200 },  
          { "from": 200 }  
        ]  
      }  
    }  
  }  
}
```

This aggregation groups documents by the ‘**price**’ field into predefined price ranges.

Performing simple searches and aggregations in Elasticsearch allows you to gain insights into your data and extract valuable information for analysis and decision-making.

Understanding relevance scoring and relevance boosting:

This is crucial for fine-tuning search results and ensuring that the most relevant documents are returned to users. In Elasticsearch, relevance scoring is a fundamental aspect of the search process, and relevance boosting allows you to influence the ranking of search results based on certain criteria. Let’s delve deeper into each concept:

Understanding Relevance Scoring:

When you perform a search in Elasticsearch, each document is assigned a relevance

score that indicates how well it matches the search criteria. The relevance score is calculated based on various factors, including:

- **Term Frequency (TF):** The frequency of the search terms within the document. Documents with a higher frequency of search terms are considered more relevant.
- **Inverse Document Frequency (IDF):** The rarity of the search terms across the entire index. Terms that appear in fewer documents are given more weight.
- **Field Length Norm:** The length of the field in which the search terms appear. Shorter fields tend to receive higher scores.
- **Term Position:** The proximity of the search terms within the document. Documents where the search terms appear closer together are considered more relevant.

Elasticsearch uses a scoring algorithm (typically the TF-IDF algorithm or BM25) to calculate the relevance score for each document. The documents are then ranked based on their relevance scores, and the most relevant documents are returned as search results.

Understanding Relevance Boosting:

Relevance boosting allows you to influence the relevance scores of documents based on certain criteria. By assigning higher weights to specific fields, documents, or terms, you can boost their relevance and ensure that they appear higher in the search results.

Field-Level Boosting:

You can assign different weights (boost values) to individual fields within your documents. This allows you to prioritize certain fields over others when calculating relevance scores.

```
{
  "query": {
    "multi_match": {
      "query": "elasticsearch tutorial",
      "fields": ["title^2", "description"]
    }
  }
}
```

```
}  
}
```

In this example, the **'title'** field is given a higher boost factor (2 times) compared to the **'description'** field, indicating that matches in the title field are more relevant.

Document-Level Boosting:

You can also assign boost values to entire documents. Documents with higher boost values are considered more relevant and are ranked higher in the search results.

Query-Time Boosting:

You can apply boost values directly to query terms or clauses, giving certain search terms or conditions more weight during the search process.

```
{  
  "query": {  
    "bool": {  
      "should": [  
        { "match": { "title": { "query": "elasticsearch", "boost": 2 } } },  
        { "match": { "description": "tutorial" } }  
      ]  
    }  
  }  
}
```

In this example, matches in the **'title'** field are given twice the relevance boost compared to matches in the **'description'** field.

By leveraging relevance boosting, you can fine-tune your search results and ensure that the most relevant documents are presented to users based on their search queries and preferences.

Here are **scenario-based interview questions** covering all the topics discussed regarding searching and querying.

1. You are working on a project where you need to index large volumes of log data. How would you structure your indices to optimize search performance?

2. Your application allows users to search for products based on multiple criteria such as name, category, and price range. How would you design your Elasticsearch queries to handle these search requirements efficiently?
3. You are tasked with implementing a search feature that supports full-text search across multiple fields in your documents. How would you utilize the Query DSL to achieve this functionality?
4. You need to implement a search feature that prioritizes documents containing exact matches over partial matches. How would you approach this using the Query DSL?
5. Your application needs to display statistics such as the number of orders placed per month. How would you use aggregations in Elasticsearch to generate this information?
6. You want to provide users with auto-suggestions as they type in the search bar based on the existing data. How would you implement this feature using Elasticsearch?
7. Your application needs to display search results that prioritize documents containing specific keywords in the title field. How would you boost the relevance of documents based on the presence of these keywords?
8. You want to implement a search feature that gives higher relevance to documents published in the past year. How would you achieve this using Elasticsearch?

Conclusion:

In this journey through Elasticsearch Searching and Querying, we've explored the fundamental concepts, Query DSL, simple searches, aggregations, and relevance scoring. Elasticsearch's robust search capabilities empower you to extract insights from your data with ease.

Join us tomorrow for Day 4, where we'll explore advanced querying techniques in Elasticsearch. Happy searching!

Elasticsearch

DevOps

Course

Learning

Interview



Following



Written by Navya Cloudops

514 Followers

More from Navya Cloudops



 Navya Cloudops

Real-time Interview Questions for 4 years Experience!!

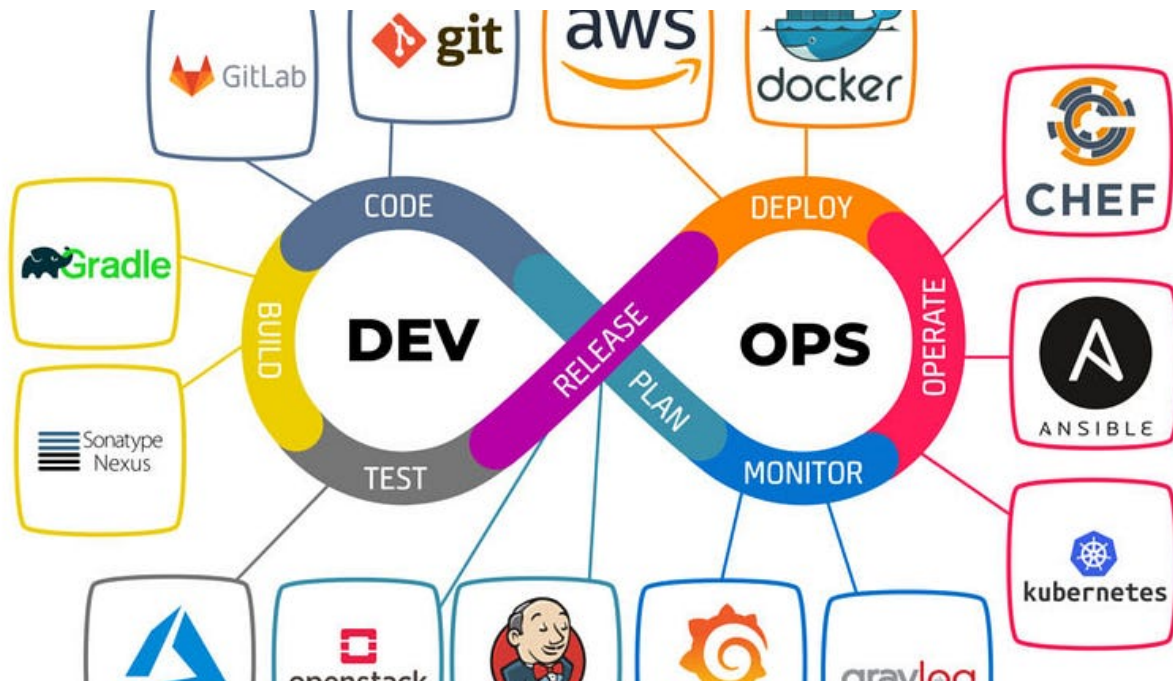
Here are some scenario based interview questions with respect to 4 years experience for a position in CITI Bank.

🌟 · 3 min read · Jan 24, 2024



88





N Navya Cloudops

DevOps Zero to Hero in 30 days!!

Here's a 30-day DevOps course outline with a detailed topic for each day!

4 min read · Jul 12, 2023



26



1

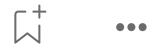


N Navya Cloudops

DevOps Zero to Hero—Day 14: Release Management!!

Welcome to Day 14 of our 30-day course on Software Development Best Practices! In today's session, we'll delve into the critical aspect of...

7 min read · Jul 26, 2023



 Navya Cloudops

DevOps Zero to Hero—Day 20: Deployment Strategies

Welcome to Day 20 of our comprehensive 30-day course on Application Deployment! In this segment, we will delve into various deployment...


8 min read · Aug 3, 2023



See all from Navya Cloudops

Recommended from Medium



 Chameera Dulanga in Bits and Pieces

Best-Practices for API Authorization

4 Best Practices for API Authorization

9 min read · Feb 6, 2024

 1.1K  4



 Navya Cloudops

Elastic Search—Day 4: Advanced Querying!!

Welcome to Day 4 of our 10-day DevOps Elasticsearch course! Today, we're diving deep into Elasticsearch's Advanced Querying capabilities...

🌟 · 7 min read · Feb 10, 2024



Lists



Self-Improvement 101

20 stories · 1364 saves



How to Find a Mentor

11 stories · 422 saves



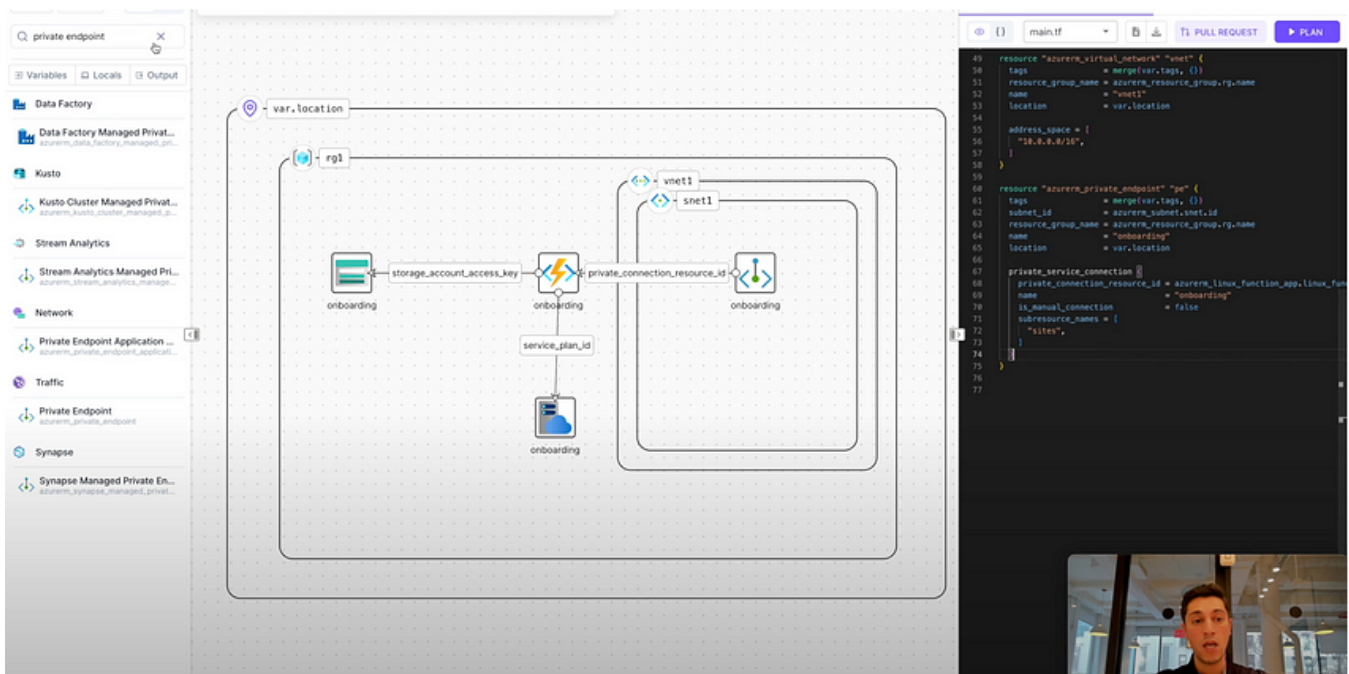
Good Product Thinking

11 stories · 470 saves



The New Chatbots: ChatGPT, Bard, and Beyond

12 stories · 307 saves



Mike Tyson of the Cloud (MTotC)

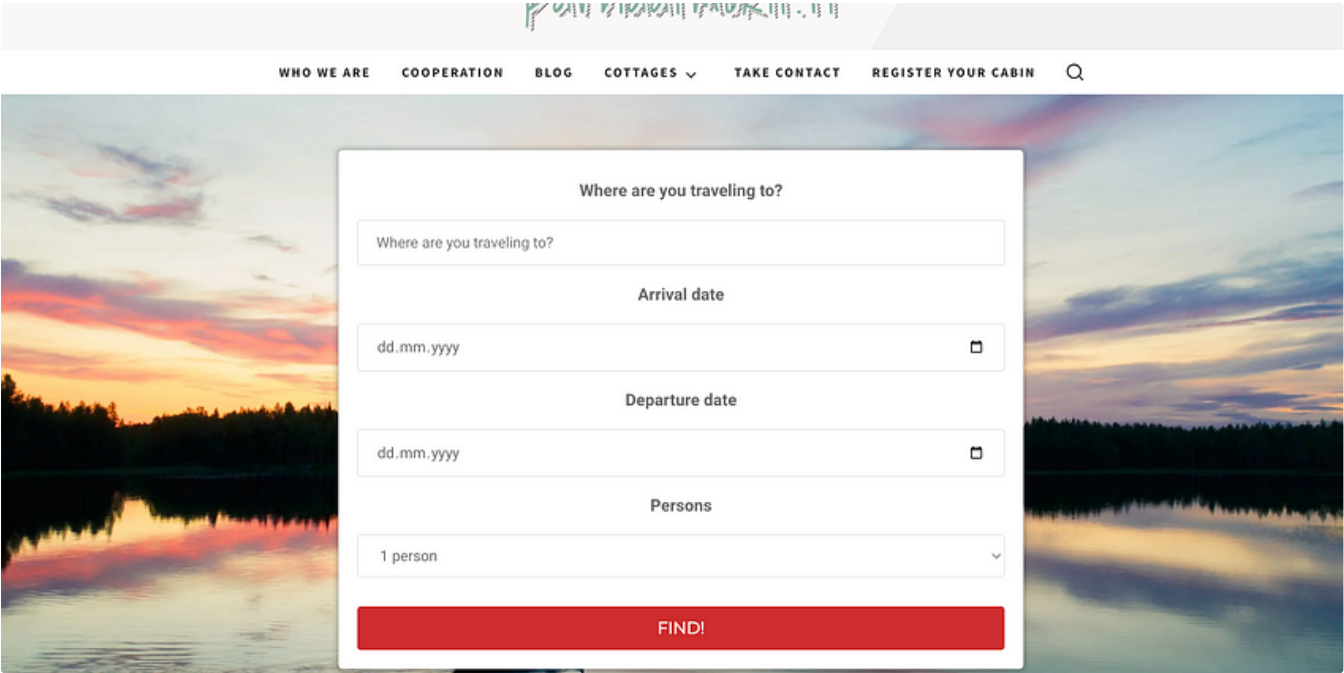
Complete Terraform Tutorial

Your journey in building a cloud infrastructure from scratch and learning Terraform with Brainboard starts here.

25 min read · Feb 8, 2024

 248 




 Artturi Jalli

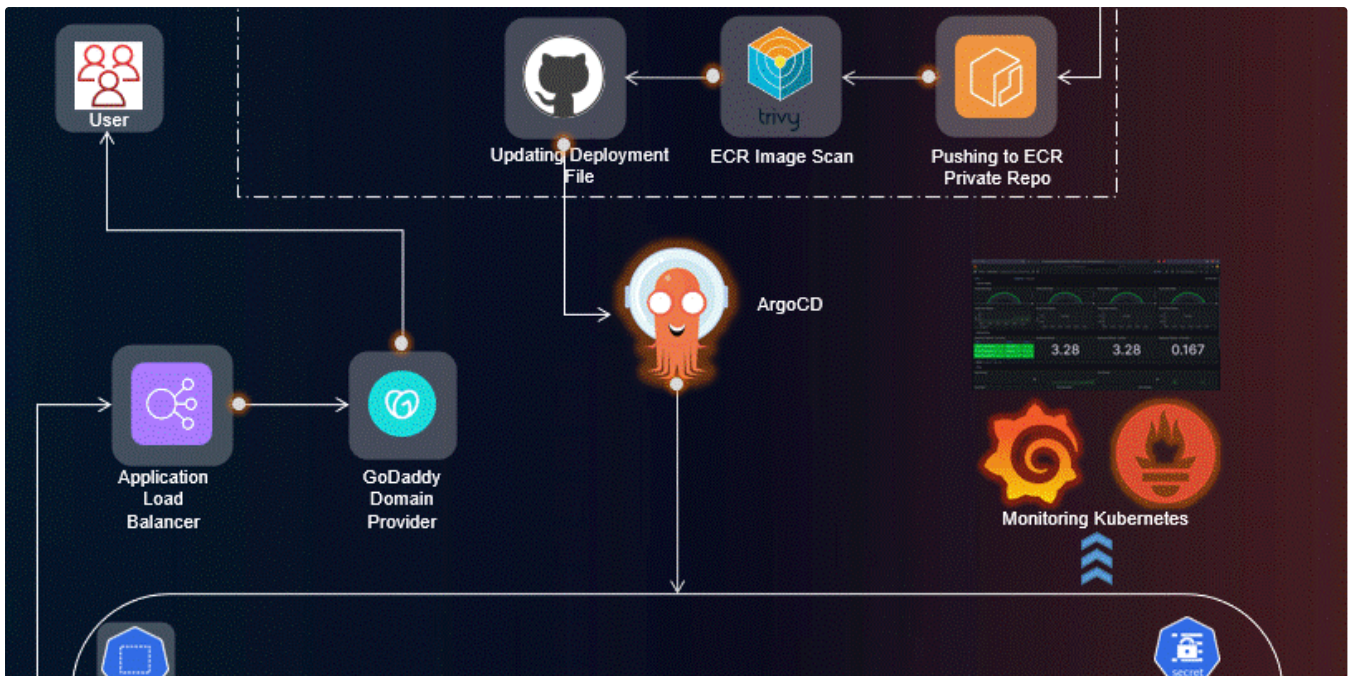
I Built an App in 6 Hours that Makes \$1,500/Mo

Copy my strategy!

★ · 3 min read · Jan 23, 2024

 9.2K  121



 Aman Pathak in Stackademic

Advanced End-to-End DevSecOps Kubernetes Three-Tier Project using AWS EKS, ArgoCD, Prometheus...

Project Introduction:

23 min read · Jan 18, 2024



1.5K



15



Dolan Miu

Why have 100% Test Coverage

100% test coverage is somewhat of a taboo phrase in software. It's “unachievable” with “diminishing returns” they would say. Non-developers...

4 min read · 3 days ago



53



See more recommendations