Open in app ↗

Search

Member-only story

Elastic Search — Day 7: Monitoring and Alerting!!

Navya Cloudops · Following
9 min read · 6 days ago

Listen Share More

Welcome back to our 10-day DevOps Elasticsearch course! Today, we're diving deep into Elasticsearch monitoring and alerting. Monitoring the health and performance of your Elasticsearch cluster is crucial for ensuring its reliability and efficiency. In this session, we'll explore various tools and techniques to monitor and set up alerts for Elasticsearch.

Overview of Elasticsearch monitoring tools and metrics:

Let's delve deeper into the overview of Elasticsearch monitoring tools and metrics.

Elasticsearch Monitoring Tools:

1. Elasticsearch Monitoring API:

- **Description:** Elasticsearch comes with a built-in Monitoring API that provides comprehensive insights into the health and performance of the Elasticsearch cluster.
- **Functionality:** The Monitoring API exposes a wide range of metrics, including node statistics, index performance, search latency, disk usage, JVM heap utilization, and cluster health.
- **Usage:** Developers and administrators can query the Monitoring API to retrieve real-time metrics and monitor the state of their Elasticsearch cluster. These metrics are invaluable for troubleshooting issues, optimizing performance, and ensuring system reliability.

2. Prometheus:

- **Description:** Prometheus is a popular open-source monitoring and alerting toolkit designed for reliability and scalability. It is highly extensible and integrates seamlessly with various systems, including Elasticsearch.
- **Functionality:** Prometheus collects metrics from monitored targets by scraping HTTP endpoints exposed by those targets. It stores collected data in a time-series database and provides a flexible query language for data analysis and visualization.
- **Usage:** By configuring Prometheus to scrape metrics from the Elasticsearch Monitoring API, users can monitor Elasticsearch cluster health, node performance, and resource utilization. Prometheus also supports alerting based on defined thresholds, enabling proactive issue detection and notification.

3. Grafana:

- **Description:** Grafana is a powerful open-source visualization and analytics platform commonly used for monitoring and observability purposes. It supports various data sources, including Prometheus, Elasticsearch, InfluxDB, and more.
- **Functionality:** Grafana allows users to create custom dashboards and visualizations to monitor and analyze metrics collected by Prometheus and

other data sources. Its rich set of features includes interactive graphs, charts, and alerting capabilities.

- **Usage:** By connecting Grafana to Prometheus as a data source, users can create dynamic dashboards to visualize Elasticsearch metrics in real-time. Grafana's intuitive interface makes it easy to design informative dashboards that provide insights into cluster health, performance trends, and system anomalies.

Elasticsearch Monitoring Metrics:

1. Cluster Health:

- **Description:** Cluster health indicates the overall status of the Elasticsearch cluster, including the number of nodes, active shards, unassigned shards, and cluster state.
- **Metrics:** Health status (green, yellow, red), number of nodes, active primary shards, active replica shards, unassigned shards, cluster state metadata, and cluster-wide indices statistics.
- **Importance:** Monitoring cluster health helps identify issues such as node failures, shard relocations, and cluster instability, allowing administrators to take corrective actions and maintain system availability.

2. Node Statistics:

- **Description:** Node statistics provide insights into the performance and resource usage of individual Elasticsearch nodes within the cluster.
- **Metrics:** JVM heap memory usage, CPU load, disk I/O operations, network throughput, thread pool statistics, and garbage collection metrics.
- **Importance:** Monitoring node statistics enables administrators to identify performance bottlenecks, resource constraints, and hardware failures that may impact cluster performance and reliability.

3. Index Performance:

- **Description:** Index performance metrics measure the indexing and search throughput of Elasticsearch indices.

- **Metrics:** Indexing rate (documents indexed per second), indexing latency (time taken to index documents), search throughput (queries executed per second), and search latency (response time for search queries).
- **Importance:** Monitoring index performance helps optimize data ingestion, query performance, and system responsiveness, ensuring efficient data retrieval and analysis.

4. Search Latency:

- **Description:** Search latency represents the time taken to process and execute search queries against Elasticsearch indices.
- **Metrics:** Average search latency, percentiles (e.g., 95th percentile latency), and search request/response times.
- **Importance:** Monitoring search latency allows administrators to assess the responsiveness of the Elasticsearch cluster and identify slow-performing queries or overloaded nodes that may degrade user experience and application performance.

By leveraging monitoring APIs, tools like Prometheus and Grafana, and essential metrics such as cluster health, node statistics, index performance, and search latency, administrators can gain valuable insights into their Elasticsearch infrastructure, detect issues proactively, and optimize system performance for enhanced scalability and efficiency.

Monitoring and alerting tools like Elasticsearch Monitoring API, Prometheus, and Grafana:
Setting up monitoring and alerting using tools like Elasticsearch Monitoring API, Prometheus, and Grafana involves several steps. Let’s break down the process:

1. Install and Configure Elasticsearch Monitoring API:
Installation:

- If you haven’t already, install Elasticsearch on your servers or cluster nodes.
- Ensure that the Monitoring plugin is installed and enabled in Elasticsearch. You can install it using Elasticsearch’s plugin management tool.

Configuration:

- Configure Elasticsearch to expose the Monitoring API endpoint. This can be done by enabling the relevant settings in the **'elasticsearch.yml'** configuration file.
- Set up authentication and security measures to protect access to the Monitoring API, especially in production environments.

2. Install and Configure Prometheus:

Installation:

- Install Prometheus on a dedicated server or container within your infrastructure.
- Prometheus is available as a binary download or can be installed using package managers on various operating systems.

Configuration:

- Configure Prometheus to scrape metrics from the Elasticsearch Monitoring API endpoint.
- Define scraping intervals and HTTP endpoints in the Prometheus configuration file (**'prometheus.yml'**).
- Set up service discovery mechanisms if your Elasticsearch cluster is dynamic and nodes are added or removed frequently.

3. Install and Configure Grafana:

Installation:

- Install Grafana on a separate server or container, or alongside Prometheus.
- Grafana is available as a binary download or can be installed using package managers similar to Prometheus.

Configuration:

- Configure Grafana to connect to Prometheus as a data source.
- Define the Prometheus endpoint URL and access credentials (if applicable) in the Grafana data source settings.

- Import pre-configured Elasticsearch monitoring dashboards or create custom dashboards to visualize Elasticsearch metrics.

4. Set Up Monitoring Dashboards in Grafana:

- Explore available Elasticsearch monitoring dashboards in Grafana's dashboard library or import community-contributed dashboards.
- Customize dashboards to display relevant metrics based on your specific monitoring requirements and Elasticsearch deployment architecture.
- Design visualizations such as line charts, histograms, and gauges to track cluster health, node performance, index statistics, and search latency.

5. Configure Alerting Rules in Prometheus and Grafana:

- Define alerting rules in Prometheus to monitor Elasticsearch metrics and trigger alerts based on predefined thresholds or conditions.
- Configure alerting channels in Prometheus to send notifications via email, Slack, PagerDuty, or other alerting integrations.
- Create alerting rules and notification channels in Grafana using Grafana's alerting engine. Grafana integrates with Prometheus for alerting purposes.

6. Test and Fine-Tune Monitoring and Alerting Setup:

- Conduct thorough testing of the monitoring and alerting setup to ensure that metrics are collected accurately, dashboards display relevant information, and alerts are triggered appropriately.
- Fine-tune alerting thresholds and notification settings based on observed performance patterns, system behavior, and business requirements.
- Regularly review and update monitoring configurations as your Elasticsearch cluster evolves, new metrics become relevant, and monitoring requirements change.

Setting up monitoring and alerting for Elasticsearch using tools like Elasticsearch Monitoring API, Prometheus, and Grafana enables administrators to gain valuable insights into cluster health, performance, and resource utilization.

Let’s take a look at some example code snippets for setting up monitoring and alerting with Prometheus and Grafana:

```
# Prometheus Configuration (prometheus.yml)
scrape_configs:
  - job_name: 'elasticsearch'
    static_configs:
      - targets: ['<elasticsearch_host>:<port>'] # Elasticsearch Monitoring API
```

```
# Grafana Dashboard (elasticsearch_dashboard.json)
{
  "annotations": {
    "list": [
      {
        "builtIn": 1,
        "datasource": "-- Grafana --",
        "enable": true,
        "hide": true,
        "iconColor": "rgba(0, 211, 255, 1)",
        "name": "Annotations & Alerts",
        "type": "dashboard"
      }
    ]
  },
  "editable": true,
  "gnetId": null,
  "graphTooltip": 0,
  "id": null,
  "links": [],
  "panels": [],
  "schemaVersion": 22,
  "style": "dark",
  "tags": [],
  "templating": {
    "list": []
  },
  "time": {
    "from": "now-6h",
    "to": "now"
  },
  "timepicker": {
    "refresh_intervals": [
      "5s",
      "10s",
      "30s",
      "1m",
      "5m",
      "15m",
      "30m",
      "1h",
      "2h",
      "1d"
    ],
    "time_options": [
      "5m",
      "15m",
      "1h",
      "6h",
      "12h",
      "24h",
      "2d",
      "7d",
      "30d"
    ]
  },
  "timezone": "",
  "title": "Elasticsearch Monitoring",
  "uid": "Nn3A7uYRk",
  "version": 1
}
```

Cluster health and performance indicators:

Understanding cluster health and performance indicators is essential for effectively managing and optimizing an Elasticsearch deployment. Here's a detailed explanation of these indicators:

Cluster Health Indicators:

1. Cluster Status:

- **Description:** The cluster status indicates the overall health of the Elasticsearch cluster.
- Indicators:
  — **Green:** All primary and replica shards are active, and the cluster is operating normally.
  — **Yellow:** All primary shards are active, but some replica shards are missing. The cluster is still operational but may be at risk if a node fails.
  — **Red:** Some primary shards are missing, indicating data loss or unavailability. The cluster is in a critical state and requires immediate attention.

2. Node Availability:

- **Description:** Node availability refers to the status of individual nodes within the Elasticsearch cluster.
- **Indicators:**
  — **Up:** Nodes are online and actively participating in the cluster.
  — **Down:** Nodes are offline or unresponsive, indicating potential hardware or network issues.
  — **Unassigned:** Nodes are present but not assigned to any role in the cluster, possibly due to configuration errors or resource constraints.

3. Shard Allocation:

- **Description:** Shard allocation tracks the distribution of primary and replica shards across nodes in the cluster.
- **Indicators:**
  — **Assigned:** Shards are distributed evenly across nodes, ensuring data redundancy and high availability.
  — **Unassigned:** Shards are not allocated to any node, indicating imbalance or insufficient resources.
  — **Relocating:** Shards are being moved or relocated due to node additions, removals, or rebalancing operations.

4. Index Health:

- **Description:** Index health reflects the status of individual indices stored within the Elasticsearch cluster.
- **Indicators:**
  — **Open:** Indices are available for read and write operations.
  — **Closed:** Indices are closed and not actively indexed or searched. This can be intentional for archiving purposes or due to errors.

Performance Indicators:

1. Indexing Throughput:

- **Description:** Indexing throughput measures the rate at which documents are ingested into Elasticsearch indices.
- **Metrics:**
  — **Indexing Rate:** Number of documents indexed per second.

— **Indexing Latency:** Time taken to index documents, including network latency and processing overhead.

2. Search Latency:

- **Description:** Search latency represents the time taken to execute search queries against Elasticsearch indices.
- **Metrics:**
  — **Average Latency:** Mean time to process and return search results.
  — **Percentiles:** Distribution of latency values across different quantiles (e.g., 95th percentile latency).

3. Resource Utilization:

- **Description:** Resource utilization measures the consumption of CPU, memory, disk, and network resources by Elasticsearch nodes.
- **Metrics:**
  — **CPU Usage:** Percentage of CPU cycles consumed by Elasticsearch processes.
  — **Memory Usage:** Amount of physical and virtual memory used by Elasticsearch JVM.
  — **Disk I/O:** Input/output operations per second (IOPS) and throughput rates.
  — **Network Traffic:** Bandwidth usage for inter-node communication and client requests.

4. Query Performance:

- **Description:** Query performance evaluates the efficiency and responsiveness of search and aggregation queries.
- **Metrics:**
  — **Query Execution Time:** Time taken to process and execute search queries.
  — **Cache Hit Ratio:** Percentage of queries served from cache to reduce computation overhead.
  — **Query Complexity:** Analysis of query patterns and optimization opportunities.

By tracking metrics such as cluster status, node availability, shard allocation, indexing throughput, search latency, resource utilization, and query performance, administrators can identify bottlenecks, diagnose issues, and fine-tune system configurations to improve efficiency, reliability, and scalability.

Here are **scenario-based interview questions** for each of the topics mentioned:

1. During a routine monitoring check, you notice that the cluster status has turned yellow. What could be the possible reasons for this, and how would you troubleshoot it?
2. A node in your Elasticsearch cluster is marked as "unassigned". What could be the potential causes of this issue, and how would you resolve it?
3. Your Elasticsearch cluster is experiencing a sudden spike in indexing latency. How would you investigate the root cause of this performance degradation?
4. Users have reported slow response times for search queries on your Elasticsearch cluster. How would you diagnose and troubleshoot this issue?
5. A critical alert is triggered in your Prometheus monitoring system, indicating that the cluster's CPU utilization has exceeded a predefined threshold. What steps would you take to address this issue?
6. Grafana dashboards show a significant drop in indexing throughput over the past hour. How would you investigate the potential causes of this performance anomaly?

Conclusion:

Monitoring and alerting are essential components of any Elasticsearch deployment. By leveraging tools like Prometheus and Grafana, we can gain valuable insights into our cluster's health and performance. Setting up effective monitoring and alerting mechanisms allows us to proactively detect and mitigate issues, ensuring the reliability and efficiency of our Elasticsearch infrastructure.

That wraps up Day 7 of our DevOps Elasticsearch course. Tomorrow, we'll explore about security. Stay tuned for updates!

Elasticsearch DevOps Monitoring Learning Interview

N

Following

Written by Navya Cloudops

514 Followers

More from Navya Cloudops

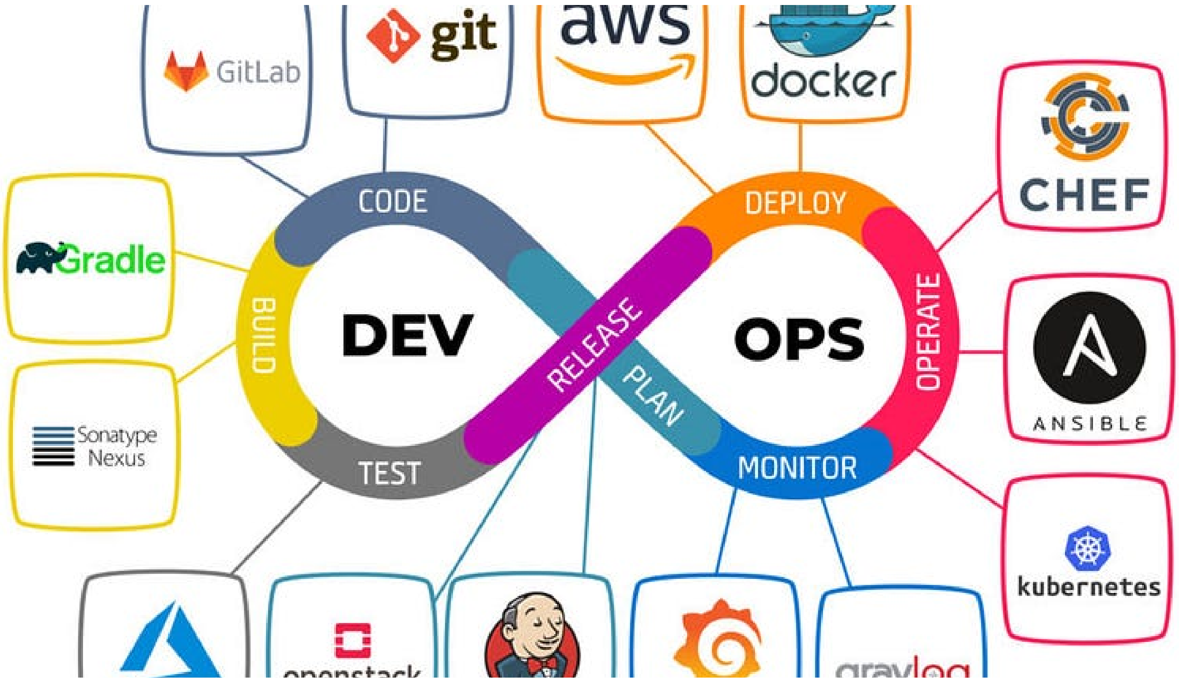
N Navya Cloudops

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



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

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

1

Navya Cloudops

DevOps Zero to Hero — Day 20: Deployment Strategies

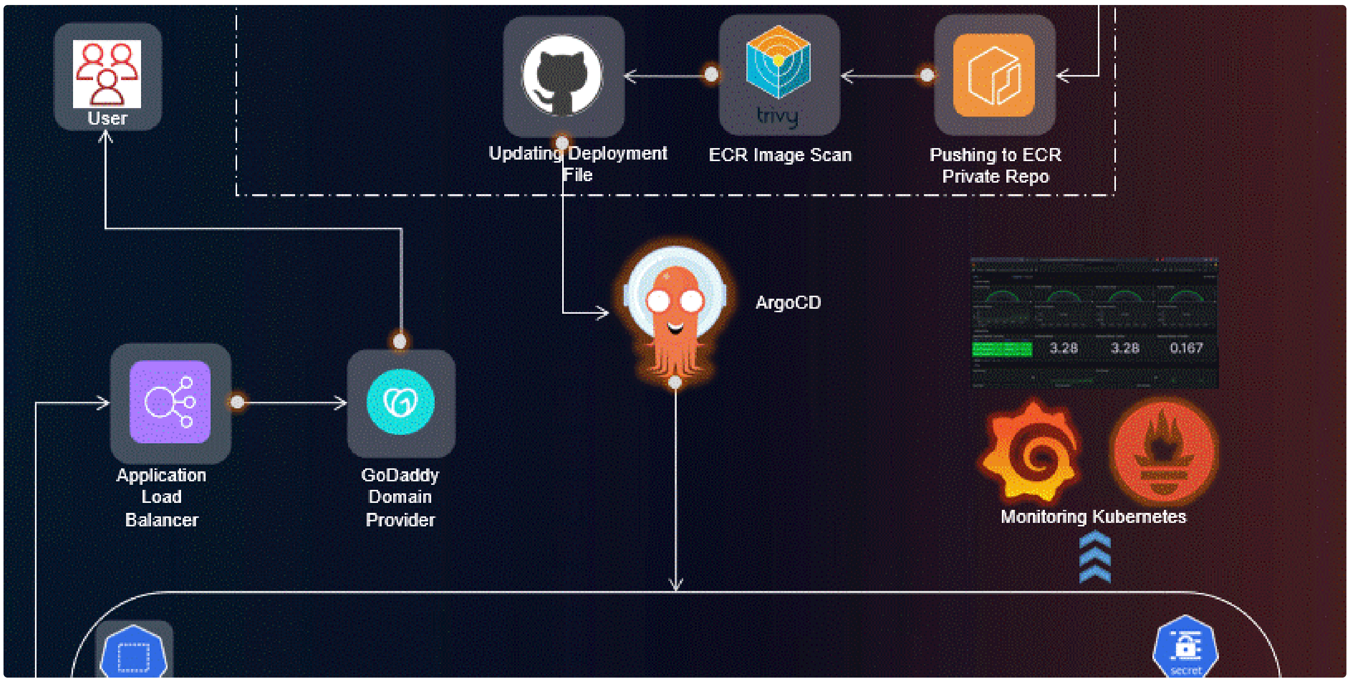
Welcome to Day 20 of our comprehensive 30-day course on Application Deployment! In this segment, we will delve into various deployment...

8 min read · Aug 3, 2023

2

See all from Navya Cloudops

Recommended from Medium



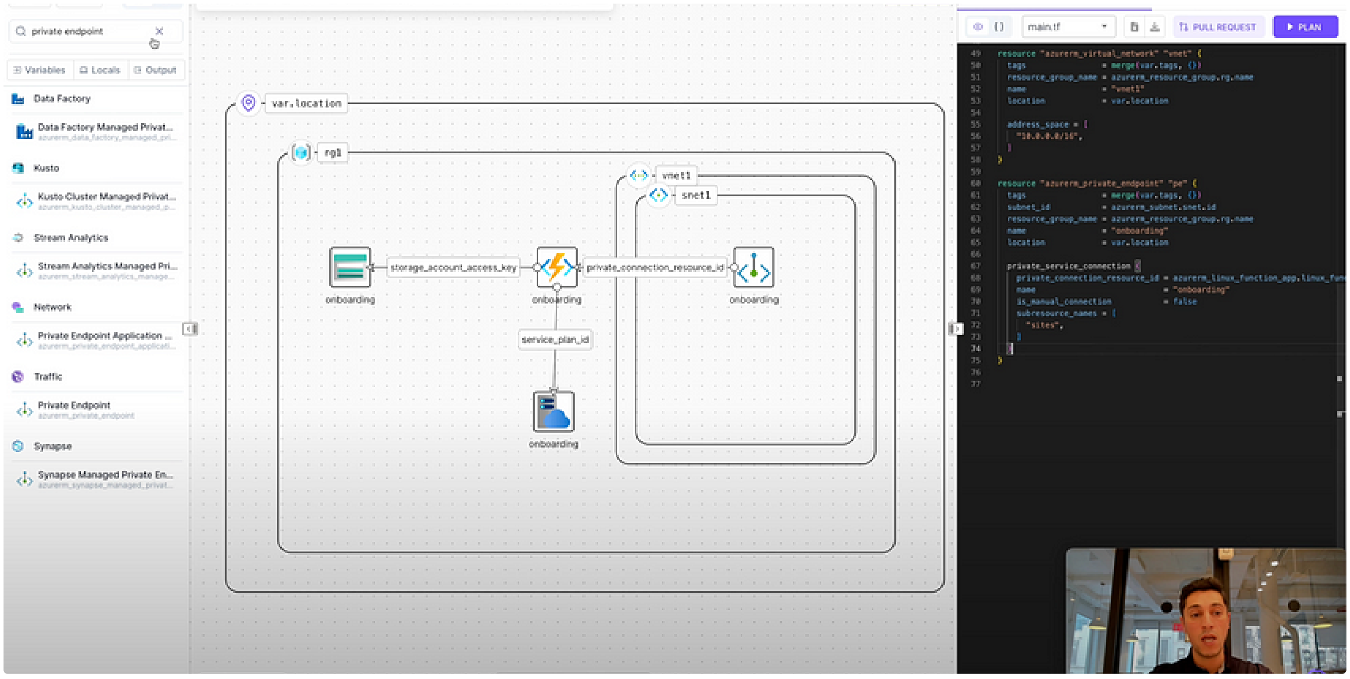
Aman Pathak in Stackademic

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



```
resource "azurerm_virtual_network" "vnet" {
  tags                = merge(var.tags, {})
  resource_group_name = azurerm_resource_group.rg.name
  name                = "vnet1"
  location            = var.location

  address_space = [
    "10.0.0.0/16",
  ]
}

resource "azurerm_private_endpoint" "pe" {
  tags                = merge(var.tags, {})
  subnet_id           = azurerm_subnet.snet.id
  resource_group_name = azurerm_resource_group.rg.name
  name                = "onboarding"
  location            = var.location

  private_service_connection {
    private_connection_resource_id = azurerm_linux_function_app.linux_fun
    name                           = "onboarding"
    is_manual_connection           = false
    subresource_names = [
      "sites",
    ]
  }
}
```

Mike Tyson of the Cloud (MToC)

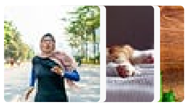
Complete Terraform Tutorial

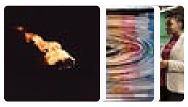
Your journey in building a cloud infrastructure from scratch and learning Terraform with Brainboard starts here.

25 min read · Feb 8, 2024

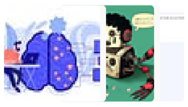
248

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

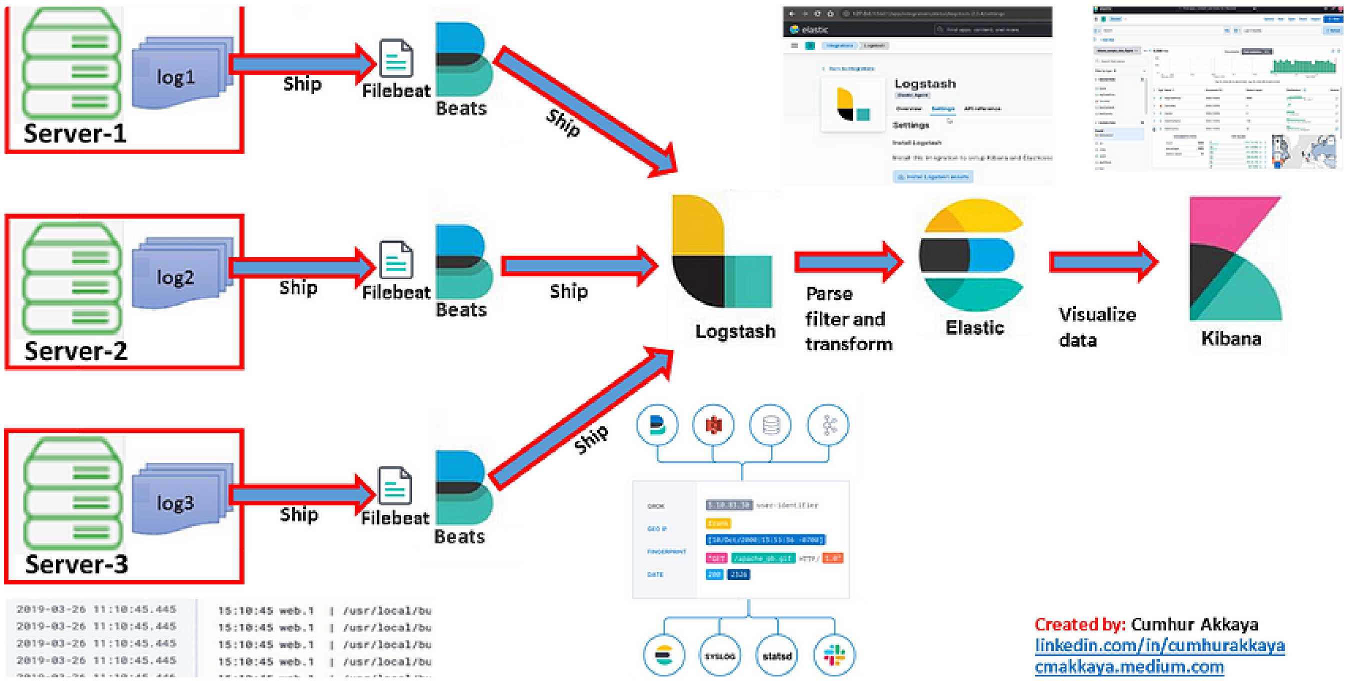
Chameera Dulanga in Bits and Pieces

Best-Practices for API Authorization

4 Best Practices for API Authorization

9 min read · Feb 6, 2024

1.1K 4



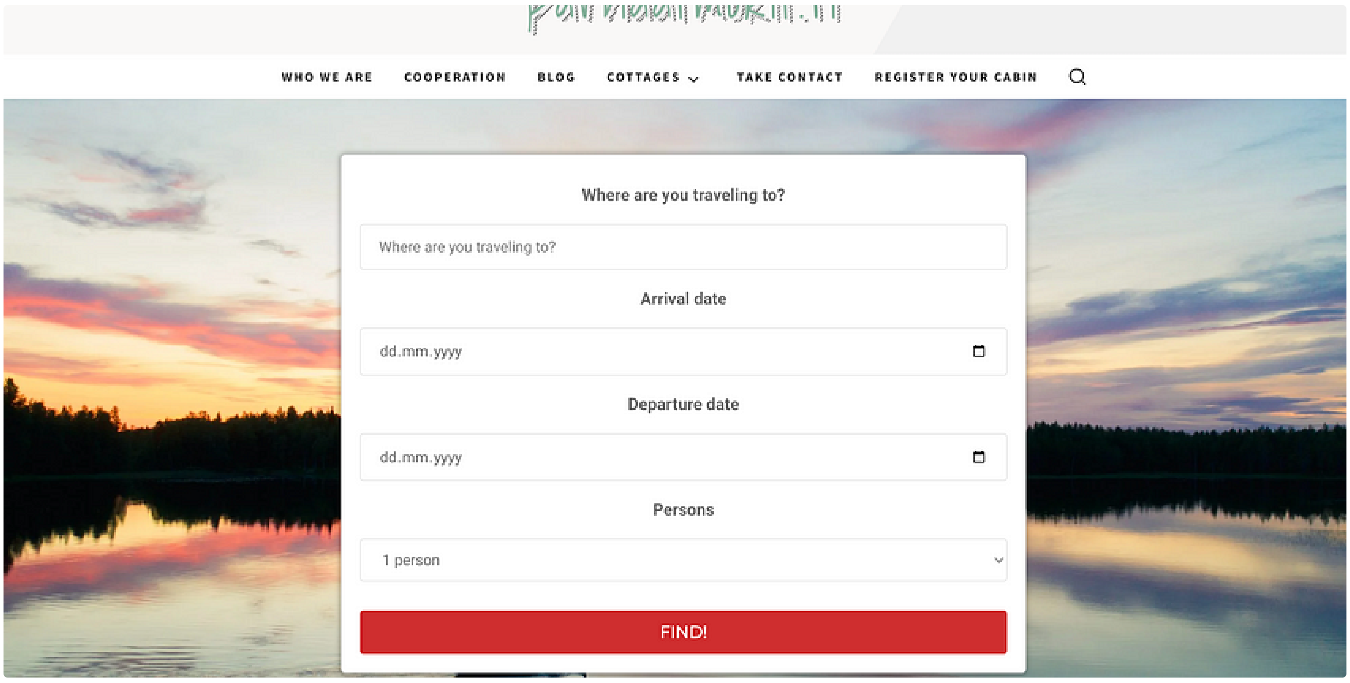
Cumhur Akkaya

Log Search and Analysis Using ELK Stack (Elastic Search, Logstash, and Kibana)

Elasticsearch, Kibana, Beats, and Logstash are known as the ELK Stack. It allows us to reliably and securely take data from any source, in...

12 min read · Oct 18, 2023

57 1



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

Samsor Rahman

#90DaysOfDevOps Challenge — Day 55 — Understanding Configuration Management with Ansible

Table of contents

5 min read · Feb 5, 2024

2

See more recommendations