Ain Shams University - Faculty of Engineering

CESS

CSE351: Distributed Systems –Spring 25
Phase 3

Ali Mohamed Refaat	21P0105
Ezz Eldin Alaa Eldin Omar	21P0100
Karim Sherif Louis	21P0223
Omar Mohamed Korkor	21P0065

Table of Contents

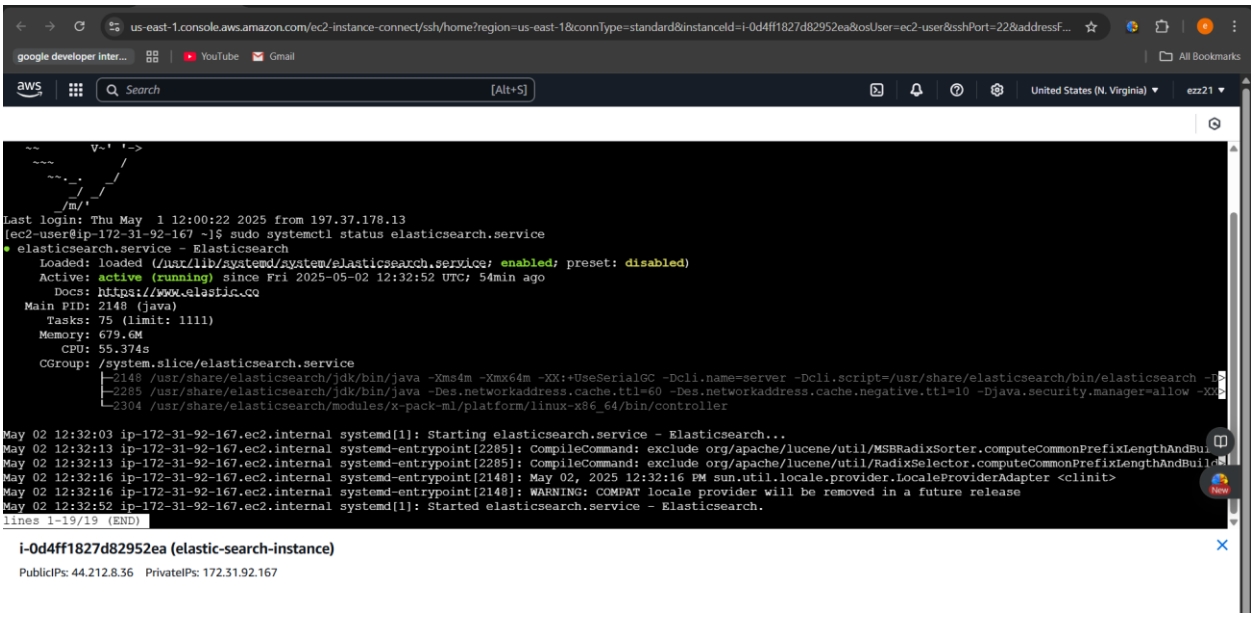
1 Introduction

In this phase of development, our focus was on enhancing the scalability, reliability, and performance of the distributed web crawler system. We significantly improved the indexing pipeline by integrating Elasticsearch, enabling faster and more powerful search capabilities. To validate the system's ability to scale, we deployed multiple crawler instances using Kubernetes, demonstrating a proof of concept for horizontal scalability. Persistent data storage was established through MongoDB, ensuring robustness in task tracking and fault recovery. Furthermore, we enhanced the threading model for improved concurrency and implemented thread safety mechanisms along with fault tolerance features to gracefully handle crawler node failures.

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv
master-node	i-042d59af52dabd3e5	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1a	ec2-54-2
crawler2	i-032ec896e1c2d8fb6	Stopped	t2.micro	–	View alarms +	us-east-1a	–
indexer-node	i-0291dbfdc14a370f5	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1a	ec2-3-80-
elastic-search-...	i-0d4ff1827d82952ea	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1a	ec2-44-2
rabbitmq-server	i-05902baf535c5851a	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1a	ec2-34-2
basic-cralwer-...	i-08d2275a512253500	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1a	ec2-44-2
crawler pods	i-0ecbc8a8b81a27f55	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1a	ec2-18-2
mongo-db	i-0ca2f1f74e15f671c	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1a	ec2-54-1

2 Indexer

To improve the capability of indexing, we deployed elastic search on an ec2 as a service.



Then we integrated that service into our indexer node. Ensuring that elastic search is configured to use stemming and filter stop words for better indexing of the parsed content. Elastic search receives requests in https and needs to be authenticated with a username and password. But for testing we disabled verifying certificates for ease of use.

```python
indexer > indexer.py > ElasticIndexer > __init__
class ElasticIndexer:
    def __init__(self, index_name="webpages"):
        self.index_name = index_name
        self.es = Elasticsearch(
            hosts=[{"host": "172.31.92.167", "port": 9200, "scheme": "https"}],
            basic_auth=("elastic", "sl_ZN64F4UN4uoDw5AIC"),
            verify_certs=False,
        )
        try:
            print(self.es.info())
        except Exception as e:
            print("Failed to connect:", e)

        if not self.es.ping():
            raise RuntimeError(" Elasticsearch ping failed. Check URL/auth.")
        logging.info("Elasticsearch cluster is up!")

        settings = {
            "settings": {
                "analysis": {
                    "analyzer": {
                        "custom_english_analyzer": {
                            "type": "custom",
                            "tokenizer": "standard",
                            "filter": ["lowercase", "english_stop", "english_stemmer"],
                        }
                    },
                    "filter": {
                        "english_stop": {"type": "stop", "stopwords": "_english_"},
                        "english_stemmer": {"type": "stemmer", "language": "english"},
                    },
                }
            },
```

```python
            raise

    def add_document(self, url, text):
        self.es.index(index=self.index_name, document={"url": url, "text": text})
        logging.info(f"Document indexed: {url}")

    def search(self, keyword):
        query = {"query": {"match": {"text": keyword}}}
        response = self.es.search(index=self.index_name, body=query)
        hits = response.get("hits", {}).get("hits", [])
        return [hit["_source"]["url"] for hit in hits]
```

It also offers interfaces for adding documents and searching the deployed elastic search.

The rest of our indexer node has the same logic and capabilities as before, receiving messages through the SQS queue from the crawlers by polling the queue.

```python

# IndexerNode class using SQS and ElasticIndexer
class IndexerNode:
    def __init__(self, queue_url):
        self.queue_url = queue_url
        self.sqs = boto3.client("sqs", region_name="us-east-1")
        self.indexer = ElasticIndexer()

    def poll_messages(self):
        while True:
            try:
                response = self.sqs.receive_message(
                    QueueUrl=self.queue_url,
                    MaxNumberOfMessages=5,
                    WaitTimeSeconds=10,
                    MessageAttributeNames=["All"],
                )
                messages = response.get("Messages", [])
                if messages:
                    logging.info(f"Received {len(messages)} messages from SQS")
                for msg in messages:
                    body = json.loads(msg["Body"])
                    self.indexer.add_document(body["url"], body["text"])
                    logging.info(f"Indexed document from URL: {body['url']}")
                    self.sqs.delete_message(
                        QueueUrl=self.queue_url, ReceiptHandle=msg["ReceiptHandle"]
                    )
                    logging.info(f"Deleted message from SQS for URL: {body['url']}")
            except Exception as e:
                logging.error(f"Error while polling messages: {e}")
            time.sleep(1)

```

```python
    def start(self):
        logging.info("IndexerNode started. Polling for messages...")
        threading.Thread(target=self.poll_messages, daemon=True).start()

    def cli(self):
        while True:
            keyword = input("Enter keyword to search (or 'exit'): ")
            if keyword.lower() == "exit":
                logging.info("Exiting CLI...")
                break
            results = self.indexer.search(keyword)
            if results:
                logging.info(f"Search results for '{keyword}': {results}")
            else:
                logging.info(f"No results found for keyword: {keyword}")


if __name__ == "__main__":
    queue_url = (
        "https://sqs.us-east-1.amazonaws.com/022499012946/crawler-indexer-sqs.fifo"
    )
    node = IndexerNode(queue_url)
    node.start()
    node.cli()
```

3 Crawler

We had a lot of progress both in crawler logic and scalability concerns.

3.1 Scalability

We created another ec2 instance for crawler deployment, but we went on our initial plan of launching a crawler cluster per ec2. The free tier machines that amazon offers aren't computationally powerful so to preserve storage and processing power we deployed k3s, a lightweight distro of Kubernetes. Then we created an image for our crawler .

```
FROM python:3.10-slim

WORKDIR /app

COPY . /app

RUN pip install --no-cache-dir -r requirements.txt

CMD ["python", "crawler.py"]

~
~
~
```

And created a deployment and launched it so crawlers can be easily scaled in this ec2.

```
A newer release of "Amazon Linux" is available.
  Version 2023.7.20250428:
Run "/usr/bin/dnf check-release-update" for full release and version update info
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
Last login: Fri May  2 14:00:40 2025 from 18.206.107.28
[ec2-user@ip-172-31-94-112 ~]$ sudo kubectl get deployment
NAME      READY   UP-TO-DATE   AVAILABLE   AGE
crawler   2/2     2            2           48m
[ec2-user@ip-172-31-94-112 ~]$ sudo kubectl get pods
NAME                       READY   STATUS    RESTARTS      AGE
crawler-8668d6dc76-qs96f   0/1     Unknown   1             48m
crawler-8668d6dc76-z6dms   0/1     Unknown   1 (11m ago)   48m
[ec2-user@ip-172-31-94-112 ~]$ cd crawler
[ec2-user@ip-172-31-94-112 crawler]$ vim Dockerfile
[ec2-user@ip-172-31-94-112 crawler]$
```

3.2 Crawler Implementation Changes

Our crawler implementation went through many changes to get better performance and introduce fault tolerance.

The crawler uses threads that work concurrently to provide faster crawling and communication with indexer and master node.

```python
    def _handle_message(self, ch, method, properties, body):
        reporter = Reporter(self.config)
        try:

            subtask_id, url, crawler_ip, depth = body.decode().split("|")
            logging.info(f"Processing {subtask_id} for {url}")

            reporter.report_status(subtask_id, crawler_ip, "PROCESSING")

            html = self.fetcher.fetch_page(url)
            time.sleep(self.config.crawl_delay)

            if not html:
                reporter.report_status(
                    subtask_id, crawler_ip, "ERROR", {"reason": "fetch failed"}
                )
                return

            content = self.fetcher.extract_content(html, url)
            self.save_to_mongo(url, html)
            logging.info("sent to mongo")
            self.indexer.send_to_indexer(url, content["text"])

            reporter.report_status(
                subtask_id, crawler_ip, "DONE", {"url": url, **content}
            )
            reporter.report_urls(
                content.get("extracted_urls", [])[:3],
                self.queue_name,
                subtask_id,
                int(depth) + 1,
            )

        except Exception as e:
            logging.error(f"Message processing failed: {e}")
```

```python
def run(self):
    Thread(target=self._heartbeat_loop, daemon=True).start()

    while not self._shutdown_flag:
        try:
            connection, channel = self._setup_rabbitmq()
            channel.basic_consume(
                queue=f"queue_{self.queue_name}",
                on_message_callback=self._handle_message,
                auto_ack=True,
            )
            logging.info(f"Crawler {self.queue_name} started consuming")
            channel.start_consuming()
        except pika.exceptions.AMQPConnectionError:
            logging.error("RabbitMQ connection failed, retrying in 5s...")
            time.sleep(5)
        except Exception as e:
            logging.error(f"Unexpected error: {e}")
            time.sleep(1)
        finally:
            if "connection" in locals() and connection.is_open:
                connection.close()

def stop(self):
    self._shutdown_flag = True
    self.mongo_client.close()
```

The main function of the crawler node is receiving messages from RabbitMQ and extract the needed content. It is handled in the main loop when it consumed a message from the intended queue, it then sends it to the call back function _handle_message.

_handle_message extracts the content and the URLs with frequent status reports to the master node like processing, done, error if the server refuses to send html content. It also sends the extracted content to the MongoDB for persistence. Status updates and new URLs are sent in different queues so no tight coupling or issues happen in the master node.

The Reporter class is responsible for all reporting related tasks. It handles sending status updates and new URLs to the master node.

```python
class Reporter:
    def __init__(self, config: CrawlerConfig):
        self.config = config

    def _get_connection(self):
        return pika.BlockingConnection(
            pika.ConnectionParameters(
                host=self.config.rabbitmq_host,
                port=self.config.rabbitmq_port,
                credentials=pika.PlainCredentials(
                    self.config.rabbitmq_username, self.config.rabbitmq_password
                ),
                heartbeat=300,
                blocked_connection_timeout=300,
            )
        )
```

```python
    def report(self, queue: str, body: Dict[str, Any]):
        try:
            connection = self._get_connection()
            channel = connection.channel()
            channel.queue_declare(queue=queue, durable=False)
            channel.basic_publish(exchange="", routing_key=queue, body=json.dumps(body))
            connection.close()
        except Exception as e:
            logging.error(f"Failed to report to {queue}: {e}")

    def report_status(
        self, subtask_id: str, crawler_ip: str, status: str, data: Optional[Dict] = None
    ):
        self.report(
            "crawler_updates",
            {
                "subtask_id": subtask_id,
                "crawler_ip": crawler_ip,
                "status": status,
                "data": data or {},
            },
        )

    def report_urls(
        self, new_urls: list[str], crawler_ip: str, parent_subtask_id: str, depth: int
    ):
        self.report(
            "crawler_urls",
            {
                "urls": new_urls,
                "crawler_ip": crawler_ip,
                "parent_subtask_id": parent_subtask_id,
                "depth": depth,
            },
        )
        logging.info(
```

Every operation that sends to a queue opens a separate connection and closes it after execution because pica has strict rules with threading. Only one thread can access a connection at a time.

The crawler node also has a thread for heartbeat messages, periodically sending pings to the master so it can be monitored for failures.

```python
def _heartbeat_loop(self):
    logging.info("Starting heartbeat loop")
    reporter = Reporter(self.config)
    while not self._shutdown_flag:
        try:
            reporter.report(
                "crawler_registry",
                {"queue_name": self.queue_name, "timestamp": time.time()},
            )
            logging.info(f"Heartbeat sent from {self.queue_name}")
            time.sleep(15)
        except Exception as e:
            logging.error(f"Heartbeat failed: {e}")
            time.sleep(5)
```

4 Master Node

The master node also has a lot of changes to handle failing crawler nodes and redistribution of tasks.

```python
class RabbitMQManager:
    _connection_lock = Lock()

    @classmethod
    def get_channel(cls):
        """Get a new channel with thread-safe connection"""
        params = pika.ConnectionParameters(
            host=RABBITMQ_HOST,
            port=RABBITMQ_PORT,
            credentials=RABBITMQ_CREDS,
            heartbeat=HEARTBEAT_INTERVAL,
            blocked_connection_timeout=CONNECTION_TIMEOUT,
            socket_timeout=10,
        )

        with cls._connection_lock:
            connection = pika.BlockingConnection(params)
            channel = connection.channel()
            channel.exchange_declare(
                exchange="crawler_exchange", exchange_type="direct", durable=False
            )
            channel.queue_declare(queue="crawler_registry", durable=False)
            channel.queue_declare(queue="crawler_updates", durable=False)
            return channel, connection
```

```python

    @classmethod
    def publish_message(cls, exchange, routing_key, body, max_retries=3):
        """Thread-safe message publishing with retries"""
        for attempt in range(max_retries):
            try:
                channel, connection = cls.get_channel()
                channel.basic_publish(
                    exchange=exchange,
                    routing_key=routing_key,
                    body=body,
                    properties=pika.BasicProperties(
                        delivery_mode=2,  # make message persistent
                    ),
                )
                connection.close()
                return True
            except Exception as e:
                logging.error(f"Publish failed (attempt {attempt+1}): {str(e)}")
                time.sleep(2**attempt)
        return False

```

The RabbitMQ class is used to enforce thread safety since the master has a lot of threads running in parallel, so we need to be very careful with open connections.

To allow for scaling, available crawler nodes aren't hardcoded anymore. The master node receives heartbeats on the registration queue. If a heartbeat comes from an unknown crawler the master node registers it as part of its crawler map.

master > new-master-node.py > RabbitMQManager > publish_message

```python


def handle_registration(ch, method, properties, body):
    try:
        msg = json.loads(body.decode())
        qn = msg["queue_name"]
        now = msg.get("timestamp", time.time())

        with threading.Lock():
            if qn not in crawler_map:
                crawler_map[qn] = Crawler(qn)
                logging.info(f"Registered crawler {qn}")
            crawler = crawler_map[qn]
            crawler.last_ping = now
            if crawler.status == CrawlerStatus.FAILED:
                crawler.status = CrawlerStatus.IDLE
                logging.info(f"Crawler {qn} recovered from failed state")
    except Exception as e:
        logging.error(f"Error in registration handler: {str(e)}")


def start_registration_listener():
    while True:
        try:
            channel, connection = RabbitMQManager.get_channel()
            channel.basic_consume(
                queue="crawler_registry",
                on_message_callback=handle_registration,
                auto_ack=True,
            )
            logging.info("Registration listener started")
            channel.start_consuming()
        except Exception as e:
            logging.error(f"Registration listener crashed: {str(e)}, restarting in 5s")
            time.sleep(5)
            continue
```

master > new-master-node.py > RabbitMQManager > publish_message

```python
def health_monitor(timeout=60):
    logging.info("Health monitor started")
    while True:
        now = time.time()
        with threading.Lock():

            for qn, crawler in list(crawler_map.items()):
                if (
                    now - crawler.last_ping > timeout
                    and crawler.status != CrawlerStatus.FAILED
                ):
                    logging.info(
                        f"{qn} timed out, reassigning {len(crawler.assigned_subtasks)} subtasks"
                    )
                    crawler.status = CrawlerStatus.FAILED

                    if crawler.assigned_subtasks:

                        for st_id in crawler.assigned_subtasks[:]:  # Iterate over copy
                            sub = subtask_map.get(st_id)
                            if not sub:
                                continue

                            sub.status = SubtaskStatus.QUEUED
                            new_crawler = get_least_loaded_crawler()
                            if new_crawler:
                                sub.assign_to_crawler(new_crawler.ip)
                                new_crawler.assign_task(sub.id)
                                success = RabbitMQManager.publish_message(
                                    exchange="crawler_exchange",
                                    routing_key=new_crawler.ip,
                                    body=f"{sub.id}|{sub.url}|{new_crawler.ip}|{sub.depth}",
                                )
                                if success:
                                    logging.info(
                                        f"Reassigned {sub.id} to {new_crawler.ip}"
```

There also is health monitor for the crawlers. It checks whether a crawler has been sending heartbeats or not. If a heartbeat has not been sent for 60 seconds, the crawler is considered FAILED and all its tasks are reassigned.

The new master node also handles getting the new URLs sent by the crawlers, checks for depth and assigns them or not based on the level.

```python
def handle_crawler_new_urls(ch, method, properties, body):
    try:
        message = json.loads(body.decode())
        parent_subtask_id = message.get("parent_subtask_id")
        if parent_subtask_id is None:
            logging.error(f"Malformed message: {message}")
            return
        crawler_ip = message["crawler_ip"]
        urls = message["urls"]
        depth = int(message["depth"])

        if depth >= 3:
            logging.info(f"Depth limit reached for {parent_subtask_id}")
            return

        logging.info(f"New URLs from {crawler_ip}: {len(urls)} URLs")

        with threading.Lock():
            parent_subtask = subtask_map.get(parent_subtask_id)
            if not parent_subtask:
                logging.warning(f"Unknown parent subtask {parent_subtask_id}")
                return

            for url in urls:
                new_subtask = Subtask(
                    parent_task_id=parent_subtask.parent_task_id,
                    url=url,
                    depth=depth + 1,
                )
                subtask_map[new_subtask.id] = new_subtask

                assigned_crawler = get_least_loaded_crawler()
                if assigned_crawler:
                    new_subtask.assign_to_crawler(assigned_crawler.ip)
                    assigned_crawler.assign_task(new_subtask.id)

                    success = RabbitMQManager.publish_message(
                        exchange="crawler_exchange",
                        routing_key=assigned_crawler.ip,
                        body=f"{new_subtask.id}|{new_subtask.url}|{crawler_ip}|{new_subtask.depth}",
                    )
                    if success:
                        logging.info(
                            f"Dispatched {new_subtask.id} to {assigned_crawler.ip}"
                        )
                    else:
                        logging.error(f"Failed to dispatch {new_subtask.id}")
    except Exception as e:
        logging.error(f"Error in new URLs handler: {str(e)}")
```

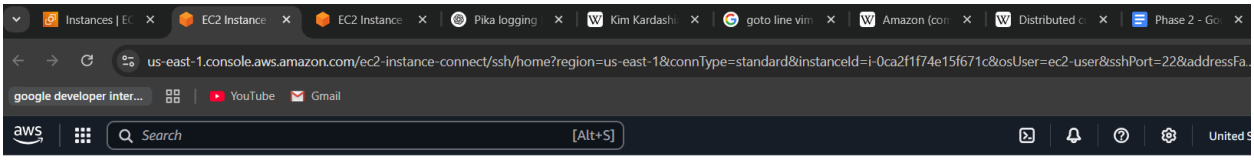
```python
if __name__ == "__main__":
    Thread(target=start_registration_listener, daemon=True).start()
    Thread(target=health_monitor, daemon=True).start()
    Thread(target=start_crawler_update_listener, daemon=True).start()
    Thread(target=start_crawler_urls_listener, daemon=True).start()
    master_process()
```

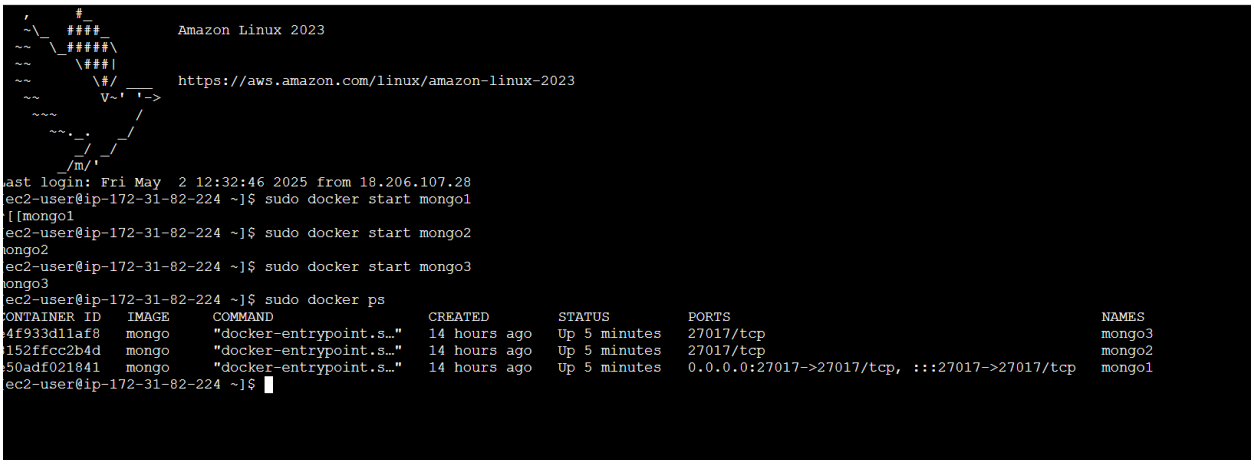
All these services are launched as separate threads.

5 MongoDB Persistence

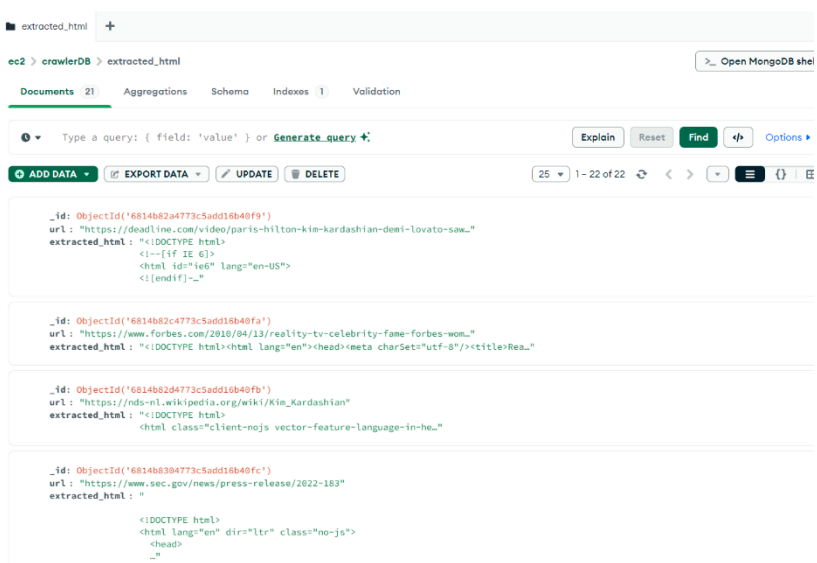
To be able to have a persistent data storage. The number of requests for S3 buckets is low. So we decided to deploy a MongoDB replication set in an ec2 instance.

Using docker containers we ran 3 mongo containers that have a volume attached to them to not lose the data every time the instance doesn't lose data





MongoDB can be accessed through a GUI MongoDB Compass, official gui for MongoDb.



```python
class CrawlerNode:
    def __init__(self, config: CrawlerConfig):
        self.config = config
        self.queue_name = socket.gethostbyname(socket.gethostname())
        self.fetcher = PageFetcher(config)
        self.indexer = IndexerClient(config.sqs_queue_url)
        self.mongo_client = MongoClient("mongodb://172.31.82.224:27017")
        self.db = self.mongo_client["crawlerDB"]
        self._shutdown_flag = False
```

The crawlers connect to the main mongo container through the pymongo library.

The other containers of the replica set work as backups of the main container, replicating the data and replacing the main container if it falls.

6 Output Logs

6.1 Master Node starting

The master node starts and registers the crawlers from the Kubernetes deployment and a crawler from another ec2

```
2025-05-02 13:22:08,801 - INFO - Found credentials from IAM Role: ezz-kiwi-sqs
2025-05-02 13:22:08,845 - INFO - Health monitor started
2025-05-02 13:22:08,845 - INFO - Master process started
2025-05-02 13:22:08,856 - INFO - Registration listener started
2025-05-02 13:22:08,866 - INFO - Crawler update urls started
2025-05-02 13:22:08,876 - INFO - Crawler update listener started
2025-05-02 13:22:08,885 - INFO - Crawler update urls started
2025-05-02 13:22:20,663 - INFO - Registered crawler 10.42.0.68
2025-05-02 13:22:20,663 - INFO - Registered crawler 10.42.0.67
2025-05-02 13:22:22,251 - INFO - Registered crawler 172.31.84.194
ubuntu@ip-172-31-88-139:~$
```

6.2 Crawler with no tasks fails & recovery

We turned off one of the crawlers and turned it on again.

```
2025-05-02 13:18:35,325 - INFO - Dispatched c16c7863-4bfe-45a1-9f52-e20336fa68dc to 172.31.84.194
2025-05-02 13:18:35,332 - INFO - Update from 172.31.84.194: Subtask c16c7863-4bfe-45a1-9f52-e20336f
2025-05-02 13:18:35,336 - INFO - Dispatched eb758974-f27b-49e7-aacd-4f982e596d60 to 10.42.0.68
2025-05-02 13:18:35,345 - INFO - Dispatched 81625774-acf5-4125-806e-203ace240e10 to 10.42.0.67
2025-05-02 13:18:35,551 - INFO - Update from 172.31.84.194: Subtask eb758974-f27b-49e7-aacd-4f982e5
2025-05-02 13:18:35,552 - INFO - Update from 172.31.84.194: Subtask 81625774-acf5-4125-806e-203ace2
2025-05-02 13:18:36,609 - INFO - Update from 172.31.84.194: Subtask c16c7863-4bfe-45a1-9f52-e20336f
2025-05-02 13:18:36,615 - INFO - Depth limit reached for c16c7863-4bfe-45a1-9f52-e20336fa68dc
2025-05-02 13:18:46,891 - INFO - Update from 172.31.84.194: Subtask eb758974-f27b-49e7-aacd-4f982e5
2025-05-02 13:18:46,892 - INFO - Update from 172.31.84.194: Subtask 81625774-acf5-4125-806e-203ace2
2025-05-02 13:22:08,801 - INFO - Found credentials from IAM Role: ezz-kiwi-sqs
2025-05-02 13:22:08,845 - INFO - Health monitor started
2025-05-02 13:22:08,845 - INFO - Master process started
2025-05-02 13:22:08,856 - INFO - Registration listener started
2025-05-02 13:22:08,866 - INFO - Crawler update urls started
2025-05-02 13:22:08,876 - INFO - Crawler update listener started
2025-05-02 13:22:08,885 - INFO - Crawler update urls started
2025-05-02 13:22:20,663 - INFO - Registered crawler 10.42.0.68
2025-05-02 13:22:20,663 - INFO - Registered crawler 10.42.0.67
2025-05-02 13:22:22,251 - INFO - Registered crawler 172.31.84.194
2025-05-02 13:33:08,850 - INFO - 172.31.84.194 timed out, reassigning 0 subtasks
ubuntu@ip-172-31-88-139:~$
```

```
2025-05-02 13:18:36,615 - INFO - Depth limit reached for c16c7863-4bfe-45a1-9f52-e20336fa68dc
2025-05-02 13:18:46,891 - INFO - Update from 172.31.84.194: Subtask eb758974-f27b-49e7-aacd-4f9
2025-05-02 13:18:46,892 - INFO - Update from 172.31.84.194: Subtask 81625774-acf5-4125-806e-203
2025-05-02 13:22:08,801 - INFO - Found credentials from IAM Role: ezz-kiwi-sqs
2025-05-02 13:22:08,845 - INFO - Health monitor started
2025-05-02 13:22:08,845 - INFO - Master process started
2025-05-02 13:22:08,856 - INFO - Registration listener started
2025-05-02 13:22:08,866 - INFO - Crawler update urls started
2025-05-02 13:22:08,876 - INFO - Crawler update listener started
2025-05-02 13:22:08,885 - INFO - Crawler update urls started
2025-05-02 13:22:20,663 - INFO - Registered crawler 10.42.0.68
2025-05-02 13:22:20,663 - INFO - Registered crawler 10.42.0.67
2025-05-02 13:22:22,251 - INFO - Registered crawler 172.31.84.194
2025-05-02 13:33:08,850 - INFO - 172.31.84.194 timed out, reassigning 0 subtasks
2025-05-02 13:33:46,689 - INFO - Crawler 172.31.84.194 recovered from failed state
```

6.3 Dispatching tasks

```
2025-05-02 13:22:08,801 - INFO - Found credentials from IAM Role: ezz-kiwi-sqs
2025-05-02 13:22:08,845 - INFO - Health monitor started
2025-05-02 13:22:08,845 - INFO - Master process started
2025-05-02 13:22:08,856 - INFO - Registration listener started
2025-05-02 13:22:08,866 - INFO - Crawler update urls started
2025-05-02 13:22:08,876 - INFO - Crawler update listener started
2025-05-02 13:22:08,885 - INFO - Crawler update urls started
2025-05-02 13:22:20,663 - INFO - Registered crawler 10.42.0.68
2025-05-02 13:22:20,663 - INFO - Registered crawler 10.42.0.67
2025-05-02 13:22:22,251 - INFO - Registered crawler 172.31.84.194
2025-05-02 13:33:08,850 - INFO - 172.31.84.194 timed out, reassigning 0 subtasks
2025-05-02 13:33:46,689 - INFO - Crawler 172.31.84.194 recovered from failed state
2025-05-02 13:34:50,581 - INFO - New task: 1 seeds, depth=2
2025-05-02 13:34:50,581 - INFO - Created 1 subtasks for Task dba981cd-6547-48af-9c11-12dd522e470b
2025-05-02 13:34:50,589 - INFO - Dispatched 7cb6f520-202a-48e5-b466-3a659f119bb6 to 10.42.0.68
2025-05-02 13:34:50,981 - INFO - Update from 10.42.0.68: Subtask 7cb6f520-202a-48e5-b466-3a659f119bb6 -> PROCESSING
2025-05-02 13:35:12,698 - INFO - Update from 10.42.0.68: Subtask 7cb6f520-202a-48e5-b466-3a659f119bb6 -> ERROR
2025-05-02 13:37:52,208 - INFO - New task: 1 seeds, depth=2
2025-05-02 13:37:52,208 - INFO - Created 1 subtasks for Task b246077b-6088-40d4-9a97-f241a23101b2
2025-05-02 13:37:52,218 - INFO - Dispatched c97785a9-75c5-4e8b-9b22-c8f9e1432bfe to 10.42.0.68
2025-05-02 13:37:52,363 - INFO - Update from 10.42.0.68: Subtask c97785a9-75c5-4e8b-9b22-c8f9e1432bfe -> PROCESSING
2025-05-02 13:40:19,181 - INFO - Update from 10.42.0.68: Subtask c97785a9-75c5-4e8b-9b22-c8f9e1432bfe -> DONE
2025-05-02 13:40:19,431 - INFO - New URLs from 10.42.0.68: 3 URLs
2025-05-02 13:40:19,442 - INFO - Dispatched dcd448e6-439a-485a-a3a9-31fd1ac6b81b to 10.42.0.68
2025-05-02 13:40:19,449 - INFO - Dispatched 0dcf9c49-23ca-4278-bd52-5c4610e5eb42 to 10.42.0.67
2025-05-02 13:40:19,456 - INFO - Dispatched 40f7e3c1-4407-40b8-940e-a48a990bbf18 to 172.31.84.194
2025-05-02 13:40:19,463 - INFO - Update from 10.42.0.68: Subtask 40f7e3c1-4407-40b8-940e-a48a990bbf18 -> PROCESSING
2025-05-02 13:40:19,848 - INFO - Update from 10.42.0.68: Subtask 0dcf9c49-23ca-4278-bd52-5c4610e5eb42 -> PROCESSING
2025-05-02 13:40:19,849 - INFO - Update from 10.42.0.68: Subtask dcd448e6-439a-485a-a3a9-31fd1ac6b81b -> PROCESSING
2025-05-02 13:40:21,174 - INFO - Update from 10.42.0.68: Subtask 40f7e3c1-4407-40b8-940e-a48a990bbf18 -> DONE
2025-05-02 13:40:21,182 - INFO - Depth limit reached for 40f7e3c1-4407-40b8-940e-a48a990bbf18
ubuntu@ip-172-31-88-139:~$
```

6.4 Reassigning tasks

```
2025-05-02 13:42:08,854 - INFO - 172.31.84.194 timed out, reassigning 1 subtasks
2025-05-02 13:42:08,863 - INFO - Reassigned 40f7e3c1-4407-40b8-940e-a48a990bbf18 to 10.42.0.68
2025-05-02 13:42:09,356 - INFO - Update from 10.42.0.68: Subtask 40f7e3c1-4407-40b8-940e-a48a990bbf18 -> PROCESSING
```

6.5 Crawler logs

```
2025-05-02 13:36:46,809 - INFO - Heartbeat sent from 172.31.84.194
2025-05-02 13:37:01,818 - INFO - Heartbeat sent from 172.31.84.194
2025-05-02 13:37:16,828 - INFO - Heartbeat sent from 172.31.84.194
2025-05-02 13:37:31,837 - INFO - Heartbeat sent from 172.31.84.194
2025-05-02 13:37:46,847 - INFO - Heartbeat sent from 172.31.84.194
2025-05-02 13:38:01,856 - INFO - Heartbeat sent from 172.31.84.194
2025-05-02 13:38:16,867 - INFO - Heartbeat sent from 172.31.84.194
2025-05-02 13:38:31,878 - INFO - Heartbeat sent from 172.31.84.194
2025-05-02 13:38:46,890 - INFO - Heartbeat sent from 172.31.84.194
2025-05-02 13:39:01,898 - INFO - Heartbeat sent from 172.31.84.194
2025-05-02 13:39:16,907 - INFO - Heartbeat sent from 172.31.84.194
2025-05-02 13:39:31,918 - INFO - Heartbeat sent from 172.31.84.194
2025-05-02 13:39:46,929 - INFO - Heartbeat sent from 172.31.84.194
2025-05-02 13:40:01,940 - INFO - Heartbeat sent from 172.31.84.194
2025-05-02 13:40:16,950 - INFO - Heartbeat sent from 172.31.84.194
2025-05-02 13:40:19,456 - INFO - Processing 40f7e3c1-4407-40b8-940e-a48a990bbf18 for https://uk.wikipedia.org/wiki/%D0%9C%D0%B0%D1%88%D0%B8%D0%BD%D0%BD%D0%B5_%D0%BD%D0%B0%D0%B2%D1%87
D0%BD%D0%BD%D1%8F
2025-05-02 13:40:19,704 - INFO - Fetched page: https://uk.wikipedia.org/wiki/%D0%9C%D0%B0%D1%88%D0%B8%D0%BD%D0%BD%D0%B5_%D0%BD%D0%B0%D0%B2%D1%87%D0%B0%D0%BD%D0%BD%D1%8F
2025-05-02 13:40:21,015 - INFO - Extracted 689 links from https://uk.wikipedia.org/wiki/%D0%9C%D0%B0%D1%88%D0%B8%D0%BD%D0%BD%D0%B5_%D0%BD%D0%B0%D0%B2%D1%87%D0%B0%D0%BD%D0%BD%D1%8F
2025-05-02 13:40:21,097 - INFO - sent to mongo
2025-05-02 13:40:21,183 - INFO - Reported 3 new URLs from 40f7e3c1-4407-40b8-940e-a48a990bbf18 at depth 3
2025-05-02 13:40:31,962 - INFO - Heartbeat sent from 172.31.84.194
2025-05-02 13:40:46,971 - INFO - Heartbeat sent from 172.31.84.194
```

6.6 Indexer Start

```
ec2-user@ip-172-31-87-169:~/elastic
[ec2-user@ip-172-31-87-169 elastic]$ python3 indexer.py
/usr/lib/python3.9/site-packages/requests/__init__.py:87: RequestsDependencyWarning: urllib3 (2.4.0) or chardet (4.0.0) doesn't match a supported version!
  warnings.warn("urllib3 ({}) or chardet ({}) doesn't match a supported "
INFO:botocore.credentials:Found credentials from IAM Role: ezz-kiwi-sqs
/home/ec2-user/.local/lib/python3.9/site-packages/elasticsearch/_sync/client/__init__.py:399: SecurityWarning: Connecting to 'https://172.31.92.167:9200' using TLS with verify_certs=False i
s insecure
  _transport = transport_class(
/home/ec2-user/.local/lib/python3.9/site-packages/urllib3/connectionpool.py:1097: InsecureRequestWarning: Unverified HTTPS request is being made to host '172.31.92.167'. Adding certificate
verification is strongly advised. See: https://urllib3.readthedocs.io/en/latest/advanced-usage.html#tls-warnings
  warnings.warn(
INFO:elastic_transport.transport:GET https://172.31.92.167:9200/ [status:200 duration:0.050s]
{'name': 'ip-172-31-92-167.ec2.internal', 'cluster_name': 'elasticsearch', 'cluster_uuid': 'vxeV3sr7T8u9XuBVQCil_A', 'version': {'number': '8.12.2', 'build_flavor': 'default', 'build_type':
 'rpm', 'build_hash': '48a287ab9497e852de30327444b0809e55d46466', 'build_date': '2024-02-19T10:04:32.774273190Z', 'build_snapshot': False, 'lucene_version': '9.9.2', 'minimum_wire_compatibi
lity_version': '7.17.0', 'minimum_index_compatibility_version': '7.0.0'}, 'tagline': 'You Know, for Search'}
/home/ec2-user/.local/lib/python3.9/site-packages/urllib3/connectionpool.py:1097: InsecureRequestWarning: Unverified HTTPS request is being made to host '172.31.92.167'. Adding certificate
verification is strongly advised. See: https://urllib3.readthedocs.io/en/latest/advanced-usage.html#tls-warnings
  warnings.warn(
INFO:elastic_transport.transport:HEAD https://172.31.92.167:9200/ [status:200 duration:0.005s]
INFO:root:☑ Elasticsearch cluster is up!
/home/ec2-user/elastic/indexer.py:54: DeprecationWarning: Passing transport options in the API method is deprecated. Use 'Elasticsearch.options()' instead.
  resp = self.es.indices.create(
/home/ec2-user/.local/lib/python3.9/site-packages/urllib3/connectionpool.py:1097: InsecureRequestWarning: Unverified HTTPS request is being made to host '172.31.92.167'. Adding certificate
verification is strongly advised. See: https://urllib3.readthedocs.io/en/latest/advanced-usage.html#tls-warnings
  warnings.warn(
INFO:elastic_transport.transport:PUT https://172.31.92.167:9200/webpages [status:400 duration:0.015s]
INFO:root:☑ Index 'webpages' ready.
INFO:root:IndexerNode started. Polling for messages...
Enter keyword to search (or 'exit'): INFO:root:Received 1 messages from SQS
/home/ec2-user/.local/lib/python3.9/site-packages/urllib3/connectionpool.py:1097: InsecureRequestWarning: Unverified HTTPS request is being made to host '172.31.92.167'. Adding certificate
verification is strongly advised. See: https://urllib3.readthedocs.io/en/latest/advanced-usage.html#tls-warnings
  warnings.warn(
INFO:elastic_transport.transport:POST https://172.31.92.167:9200/webpages/_doc [status:201 duration:0.117s]
INFO:root:Document indexed: https://en.wikipedia.org/wiki/Distributed_computing
INFO:root:Indexed document from URL: https://en.wikipedia.org/wiki/Distributed_computing
INFO:root:Deleted message from SQS for URL: https://en.wikipedia.org/wiki/Distributed_computing
INFO:root:Received 1 messages from SQS
/home/ec2-user/.local/lib/python3.9/site-packages/urllib3/connectionpool.py:1097: InsecureRequestWarning: Unverified HTTPS request is being made to host '172.31.92.167'. Adding certificate
verification is strongly advised. See: https://urllib3.readthedocs.io/en/latest/advanced-usage.html#tls-warnings
  warnings.warn(
INFO:elastic_transport.transport:POST https://172.31.92.167:9200/webpages/_doc [status:201 duration:0.017s]
INFO:root:Document indexed: https://et.wikipedia.org/wiki/Hajusarvutus
INFO:root:Indexed document from URL: https://et.wikipedia.org/wiki/Hajusarvutus
INFO:root:Deleted message from SQS for URL: https://et.wikipedia.org/wiki/Hajusarvutus
```

6.7 Indexer search

```
Enter keyword to search (or 'exit'): distribute
/home/ec2-user/.local/lib/python3.9/site-packages/urllib3/connectionpool.py:1097: InsecureRequestWarning: Unverified HTTPS request is being made to host '172.31.92.167'. Adding certificate
verification is strongly advised. See: https://urllib3.readthedocs.io/en/latest/advanced-usage.html#tls-warnings
  warnings.warn(
INFO:elastic_transport.transport:POST https://172.31.92.167:9200/webpages/_search [status:200 duration:0.018s]
INFO:root:Search results for 'distribute': ['https://en.wikipedia.org/wiki/Distributed_computing', 'https://en.wikipedia.org/wiki/Distributed_computing', 'https://en.wikipedia.org/wiki/Dist
ributed_computing', 'https://ja.wikipedia.org/wiki/%E5%88%86%E6%95%A3%E3%82%B3%E3%83%B3%E3%83%94%E3%83%A5%E3%83%BC%E3%83%86%E3%82%A3%E3%83%B3%E3%82%B0', 'https://es.wikipedia.org/wiki/Compu
taci%C3%B3n_distribuida', 'https://en.wikipedia.org/wiki/Amazon_(company)', 'https://web.archive.org/web/20230801073911/https://www.marketscreener.com/quote/stock/AMAZON-COM-INC-12864605/co
mpany/', 'https://apps.bostonglobe.com/metro/investigations/spotlight/2019/11/21/seeing-red/convenience-culture-makes-traffic-worse/']
Enter keyword to search (or 'exit'):
```