

SPE Major Project

Talent Bridge

Ritik Kumar Gupta (IMT2021098)

Pandey Shourya Prasad (IMT2021535)

GitHub Repo -

1. https://github.com/ritik-rkg/Talent_Bridge_Compose
2. https://github.com/ritik-rkg/Talent_Bridge_Kubernetes

Below GitHub repo also contains the working code on which we experimented a lot: https://github.com/shouryap1/Talent_Bridge_K8s

Innovation:

1. Trivy for Vulnerability Scanning:

Implemented *Trivy*, a lightweight vulnerability scanner, to ensure the Docker images used in the project are secure. This proactive approach enhances the security and reliability of the deployed services.

2. Prometheus and Grafana for Kubernetes Monitoring:

Opted for Prometheus and Grafana for monitoring and logging due to their seamless integration with Kubernetes, providing real-time insights and alerts. This approach was chosen over ELK due to its complexity in Kubernetes setups, ensuring faster and more efficient deployment with minimal overhead.

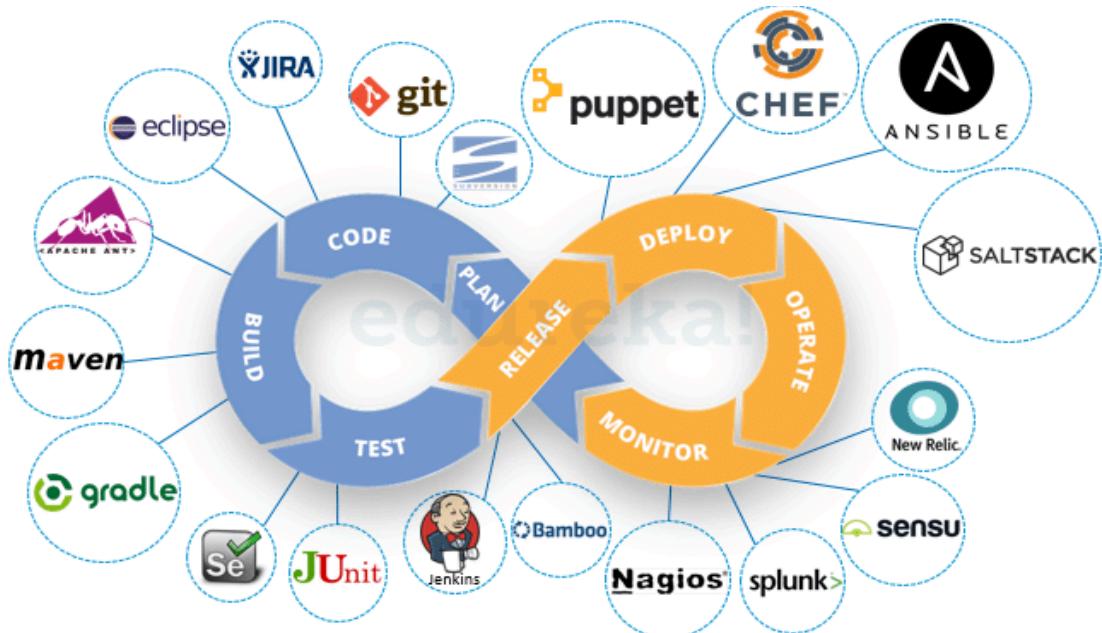
Introduction

A full-stack job portal web application developed using the MERN stack (MongoDB, Express.js, React.js, Node.js). The platform enables recruiters to post job openings, candidates to apply for positions, and administrators to manage users and job postings effectively. Candidates can upload their resumes, manage their profiles, and track their applications seamlessly. The application also features an automated CI/CD pipeline, implemented using Git, GitHub, Jenkins, Maven, Ansible, Docker, and Kubernetes, ensuring robust deployment and continuous delivery. For monitoring and observability, the ELK Stack (Elasticsearch, Logstash, Kibana), Prometheus, and Grafana are integrated to provide comprehensive logging, metrics visualization, and performance tracking.

DevOps?

What is DevOps?

- DevOps is a set of practices, tools, and a cultural philosophy that automates and integrates the processes between software development and IT teams. It emphasizes team empowerment, cross-team communication and collaboration, and technology automation.
- A DevOps team includes developers and IT operations working collaboratively throughout the product lifecycle, to increase the speed and quality of software deployment. It's a new way of working, a cultural shift, with significant implications for teams and their organizations.



DevOps Practices:

1. **Continuous Integration:** Continuous integration is the practice of automating the integration of code changes into a software project. It allows developers to frequently merge code changes into a central repository where builds and tests are executed. This helps DevOps teams address bugs quicker, improve software quality, and reduce the time it takes to validate and release new software updates.

2. **Continuous delivery:** Continuous delivery expands upon continuous integration by automatically deploying code changes to a testing/production environment. It follows a continuous delivery pipeline, where automated builds, tests, and deployments are orchestrated as one release workflow.
3. **Infrastructure as Code:** Whether your organization has an on-premise data center or is completely in the cloud, having the ability to quickly and consistently provision, configure, and manage infrastructure is key to successful DevOps adoption. Infrastructure as Code (IaC) goes beyond simply scripting infrastructure configuration to treating your infrastructure definitions as actual code: using source control, code reviews, tests, etc.
4. **Monitoring:** DevOps teams monitor the entire development lifecycle — from planning, development, integration and testing, deployment, and operations. This allows teams to respond to any degradation in the customer experience, quickly and automatically. More importantly, it allows teams to “shift left” to earlier stages in development and minimize broken production changes.

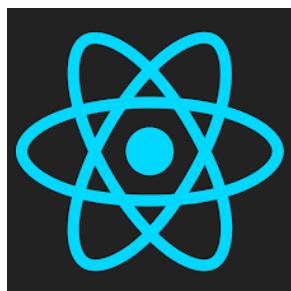
Why DevOps?

1. **Shorter Development Cycles, Faster Innovation:** When development and operations teams are in separate silos, it's usually difficult to tell if an application is ready for operations. When development teams simply turn over an application, the operations' cycle times are extended needlessly.
2. **Reduced Deployment Failures, Rollbacks, and Time to Recover:** Part of the reason teams experience deployment failures is due to programming defects. The shorter development cycles with DevOps promote more frequent code releases. This, in turn, makes it easier to spot code defects. Therefore, teams can reduce the number of deployment failures using agile programming principles that call for collaboration and modular programming.
3. **Increased Efficiencies:** Increased efficiency helps to speed the development process and make it less prone to error. There are ways to automate DevOps tasks. Continuous integration servers automate the process of testing code, reducing the amount of manual work required.

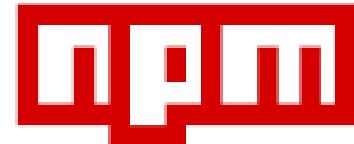
Technology Stack

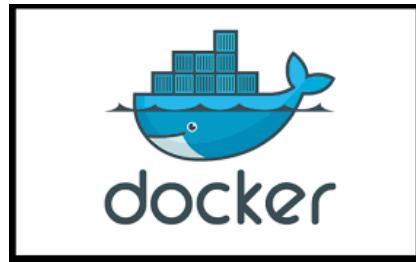
The tools used to create the application & DevOps chain and for implementing the same are as follows:

- **Frontend:** ReactJs
- **Backend:** NodeJs and ExpressJs
- **Database:** MongoDB
- **IDE:** VSCode
- **Dependency and Package Management:** npm, pip, nvm
- **Testing:** Chai, Mocha
- **Version Control System:** Git
- **Continuous Integration:** Jenkins
- **Containerization:** Docker
- **Container Orchestration:** Kubernetes
- **Configuration Management:** Ansible
- **Monitoring:** ELK stack, Prometheus and Grafana

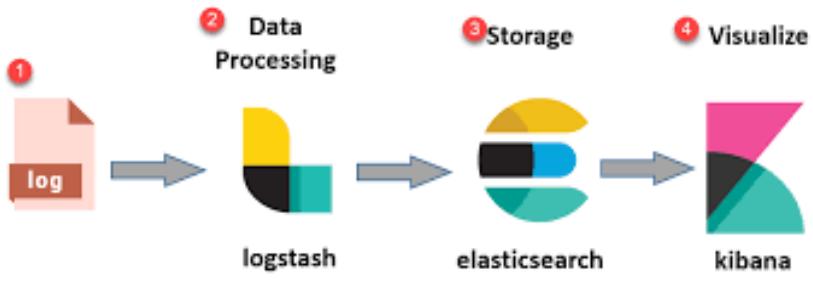


Mongoose { }





Jenkins



© gun09.com



Grafana



Installation Process

Node.js Installation

- sudo apt-get update
- curl -o- <https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.3/install.sh>
- curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.3/install.sh | bash
- source /.bashrc
- nvm install lts/iron
- node --version

```
ritik@ritik:~$ node --version
v20.12.1
ritik@ritik:~$ npm --version
10.5.0
ritik@ritik:~$ nvm --version
0.39.7
ritik@ritik:~$
```

Here the long-term support version of Node.js is used as the de-facto version of the language. If some other version of Node.js has to be installed it can be changed.

- nvm list (List all available Node.js versions)
- nvm install <version>
- nvm use <version>

```
ritik@ritik:~$ nvm list
-> v20.12.1
default -> 20 (> v20.12.1)
iojs -> N/A (default)
unstable -> N/A (default)
node -> stable (> v20.12.1) (default)
stable -> 20.12 (> v20.12.1) (default)
lts/* -> lts/iron (> v20.12.1)
lts/argon -> v4.9.1 (> N/A)
lts/boron -> v6.17.1 (> N/A)
lts/carbon -> v8.17.0 (> N/A)
lts/dubnium -> v10.24.1 (> N/A)
lts/eradium -> v12.22.12 (> N/A)
lts/fermium -> v14.21.3 (> N/A)
lts/gallium -> v16.20.2 (> N/A)
lts/hydrogen -> v18.20.1 (> N/A)
lts/iron -> v20.12.1
ritik@ritik:~$
```

React.js Installation:

Here the long-term support version of Node.js is used as the de-facto version of the language. If some other version of Node.js has to be installed it can be changed.

- npm create vite@latest TalentBridge --template react
- cd TalentBridge
- npm install
- npm run dev
- Access the app in your browser at the URL provided (typically
`http://localhost:5173`)

```
ritik@ritik:~/Downloads/atalent_bridge/talentbridge/frontend$ npm install
up to date, audited 812 packages in 14s
149 packages are looking for funding
  run `npm fund` for details
5 vulnerabilities (2 moderate, 3 high)

To address all issues, run:
  npm audit fix

Run `npm audit` for details.
```

MongoDB Atlas Setup:

Here the long-term support version of Node.js is used as the de-facto version of the language. If some other version of Node.js has to be installed it can be changed.

Sign up or log in:

- Visit [MongoDB Atlas](#) and create an account or log in.

Create a new cluster:

- In the Atlas dashboard, click "Create Cluster."
- Choose a cloud provider, region, and cluster tier.
- Click "Create Cluster."

Set up a database user:

- Go to "Database Access" and click "Add New Database User."
- Set a username and password.

Configure network access:

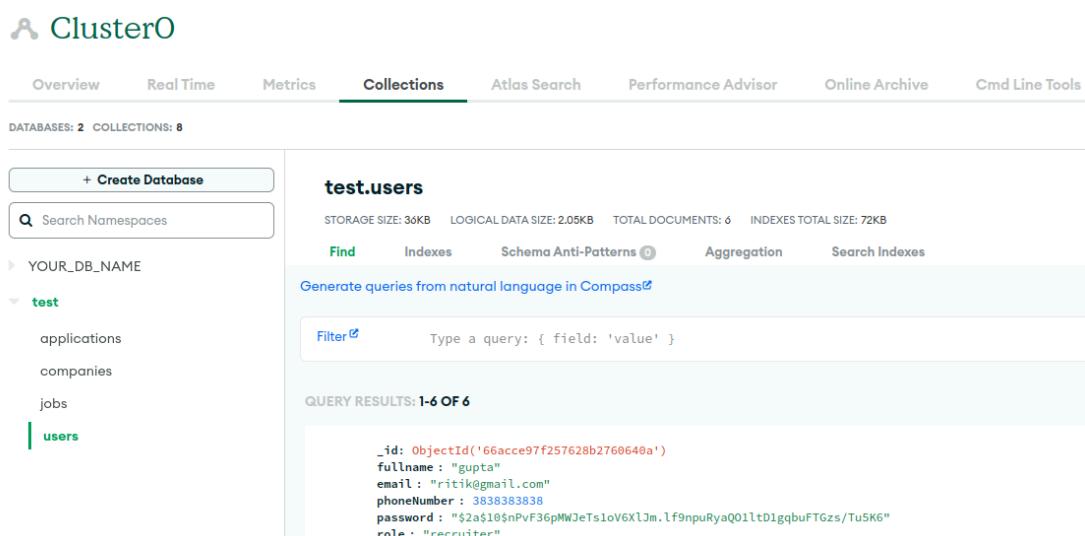
- Go to "Network Access" and click "Add IP Address."
- Allow access from your IP address (or use `0.0.0.0/0` for unrestricted access during development).

Connect to your cluster:

- Go to "Clusters" and click "Connect."
- Select "Connect Your Application."
- Copy the connection string (e.g.,
`mongodb+srv://<username>:<password>@cluster0.mongodb.net/myFirstDatabase?retryWrites=true&w=majority`).

Integrate with the application:

- Replace `<username>`, `<password>`, and `myFirstDatabase` in the connection string with your user credentials and database name.
- Use the string in your backend application's database connection setup (e.g., using Mongoose).



The screenshot shows the Cluster0 interface with the "Collections" tab selected. On the left, there is a sidebar with a tree view of databases and collections. Under the "test" database, the "users" collection is selected. The main panel displays the "test.users" collection details, including storage size, logical data size, total documents, and index size. It also includes tabs for Find, Indexes, Schema Anti-Patterns, Aggregation, and Search Indexes. Below these tabs, there is a search bar and a query builder. The "QUERY RESULTS" section shows 1-6 OF 6 documents, with one document listed:

```
_id: ObjectId('66acce97f257628b2760640a')
fullname : "gupta"
email: "ritik@gmail.com"
phoneNumber : 3838383838
password : "$2a$10$nPvF36pMWJeTs1oV6XlJm.lf9npuRyaQ0iltDiggbuFTGzs/Tu5K6"
role : "recruiter"
```

test.users

STORAGE SIZE: 36KB LOGICAL DATA SIZE: 2.05KB TOTAL DOCUMENTS: 6 INDEXES TOTAL SIZE: 72KB

Find

Indexes

Schema Anti-Patterns 0

Aggregation

Search Indexes

Generate queries from natural language in Compass ↗

Filter ↗

Type a query: { field: 'value' }

QUERY RESULTS: 1-6 OF 6

```
_id: ObjectId('66acce97f257628b2760640a')
fullname : "gupta"
email : "ritik@gmail.com"
phoneNumber : 3838383838
password : "$2a$10$nPvF36pMWJeTsloV6XlJm.lf9npuRyaQ01ltD1gqbuFTGzs/Tu5K6"
role : "recruiter"
profile : Object
createdAt : 2024-08-02T12:18:31.240+00:00
updatedAt : 2024-08-02T12:33:04.466+00:00
__v : 0
```

Git Installation:

- sudo apt-get update
- sudo apt-get install git
- git --version

```
ritik@ritik:~$ git --version
git version 2.34.1
ritik@ritik:~$
```

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Required fields are marked with an asterisk (*).

Owner *



Repository name *

/ Talent_Bridge

Talent_Bridge is available.

Great repository names are short and memorable. Need inspiration? How about [turbo-octo-winner](#) ?

Description (optional)

Public

Anyone on the internet can see this repository. You choose who can commit.

Private

You choose who can see and commit to this repository.

Initialize this repository with:

Add a README file

This is where you can write a long description for your project. [Learn more about READMEs.](#)

Jenkins Installation:

- wget -q -O - https://pkg.jenkins.io/debian/jenkins.io.key | sudo apt-key add -
- sudo sh -c 'echo deb https://pkg.jenkins.io/debian binary/etc/apt/sources.list.d/jenkins.list'
- sudo apt install ca-certificates
- sudo apt-get update
- sudo apt-get install jenkins
- sudo service jenkins status
- jenkins --version

```
ritik@ritik:~$ jenkins --version
2.479.2
ritik@ritik:~$
```

Sign in to Jenkins

Username
Ritik

Password
.....

Keep me signed in

Sign in

Docker Installation:

- sudo apt-get update
- sudo apt-get install docker.io
- sudo dockerd (To start the docker service)
- docker --version
- sudo docker images (To list all local docker images)

```
ritik@ritik:~$ sudo systemctl status docker
● docker.service - Docker Application Container Engine
  Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enabled)
  Active: active (running) since Mon 2024-12-09 19:30:48 IST; 20min ago
    TriggeredBy: ● docker.socket
      Docs: https://docs.docker.com
   Main PID: 1611 (dockerd)
     Tasks: 14
    Memory: 51.5M
       CPU: 2.609s
      CGROUP: /system.slice/docker.service
                  └─1611 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock
```

```
ritik@ritik:~$ docker --version
Docker version 27.3.1, build ce12230
```

Ansible Installation:

- sudo apt-get install openssh-server
- sudo apt-get update
- sudo apt-get install ansible

- pip install docker
- ansible --version

```
ritik@ritik:~$ ansible --version
ansible [core 2.17.7]
  config file = /etc/ansible/ansible.cfg
  configured module search path = ['~/home/ritik/.ansible/plugins/modules', '/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/lib/python3/dist-packages/ansible
  ansible collection location = /home/ritik/.ansible/collections:/usr/share/ansible/collections
  executable location = /usr/bin/ansible
  python version = 3.10.12 (main, Nov 6 2024, 20:22:13) [GCC 11.4.0] (/usr/bin/python3)
  jinja version = 3.1.3
  libyaml = True
ritik@ritik:~$
```

Prometheus Installation:

- helm repo add prometheus-community <https://prometheus-community.github.io/helm-charts>
- helm repo update
- helm install prometheus prometheus-community/prometheus
- curl https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3 | bash
- helm repo add grafana <https://grafana.github.io/helm-charts>

NAME	CHART VERSION	APP VERSION	DESCRIPTION
prometheus-community/alertmanager	1.13.1	v0.27.0	The Alertmanager handles alerts sent by client ...
prometheus-community/alertmanager-snmp-notifier	0.4.0	v1.6.0	The SNMP Notifier handles alerts coming from Pr...
prometheus-community/jiralert	1.7.1	v1.3.0	A Helm chart for Kubernetes to install jiralert
prometheus-community/kube-prometheus-stack	66.3.1	v0.78.2	kube-prometheus-stack collects Kubernetes manifest...
prometheus-community/kube-state-metrics	5.27.0	v2.14.0	Install kube-state-metrics to generate and expose...
prometheus-community/prom-label-proxy	0.10.0	v0.11.0	A proxy that enforces a given label in a given ...
prometheus-community/prometheus	26.0.0	v3.0.0	Prometheus is a monitoring system and time series...
prometheus-community/prometheus-adapter	4.11.0	v0.12.0	A Helm chart for k8s prometheus adapter
prometheus-community/prometheus-blackbox-exporter	9.1.0	v0.25.0	Prometheus Blackbox Exporter

Grafana Installation:

- sudo apt-get install -y adduser libfontconfig1 musl

- wget

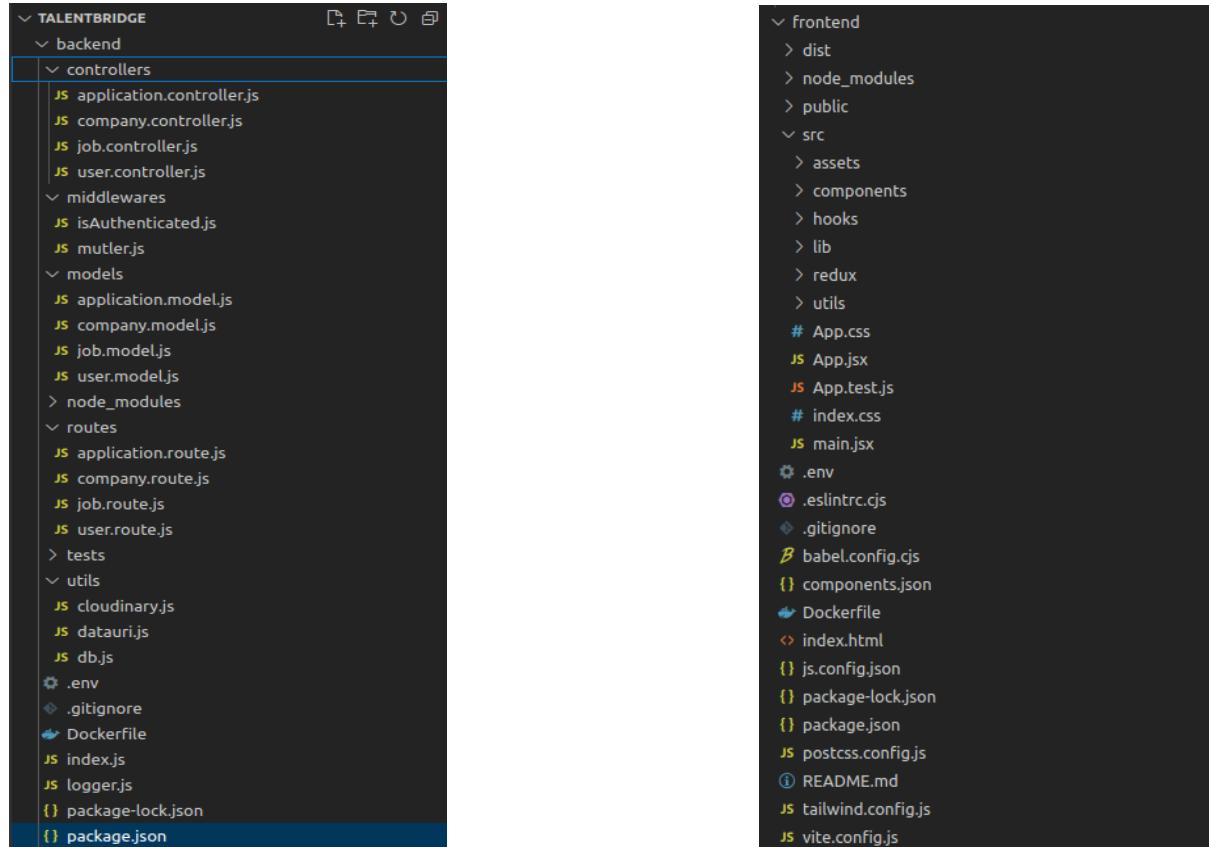
https://dl.grafana.com/enterprise/release/grafana-enterprise_11.4.0_amd64.
deb
- sudo dpkg -i grafana-enterprise_11.4.0_amd64.deb

```
ritik@ritik:~$ sudo systemctl status grafana-server
● grafana-server.service - Grafana instance
   Loaded: loaded (/lib/systemd/system/grafana-server.service; enabled; vendor preset: enabled)
   Active: active (running) since Mon 2024-12-09 19:30:45 IST; 31min ago
     Docs: http://docs.grafana.org
 Main PID: 1612 (grafana)
    Tasks: 24 (limit: 6946)
   Memory: 86.0M
      CPU: 6.900s
     CGroup: /system.slice/grafana-server.service
             └─1612 /usr/share/grafana/bin/grafana server --config=/etc/grafana/grafana.ini --pidfile=>

Dec 09 19:30:47 ritik grafana[1612]: logger=grafana-apiserver t=2024-12-09T19:30:47.44889713+05:30 lev>
Dec 09 19:30:47 ritik grafana[1612]: logger=grafana-apiserver t=2024-12-09T19:30:47.451705238+05:30 lev>
Dec 09 19:32:02 ritik grafana[1612]: logger=infra.usagestats t=2024-12-09T19:32:02.630170116+05:30 lev>
Dec 09 19:40:47 ritik grafana[1612]: logger=cleanup t=2024-12-09T19:40:47.680098489+05:30 level=info m>
Dec 09 19:40:47 ritik grafana[1612]: logger=grafana-apiserver t=2024-12-09T19:40:47.680098489+05:30 lev>
```

Web Application

Directory Structure

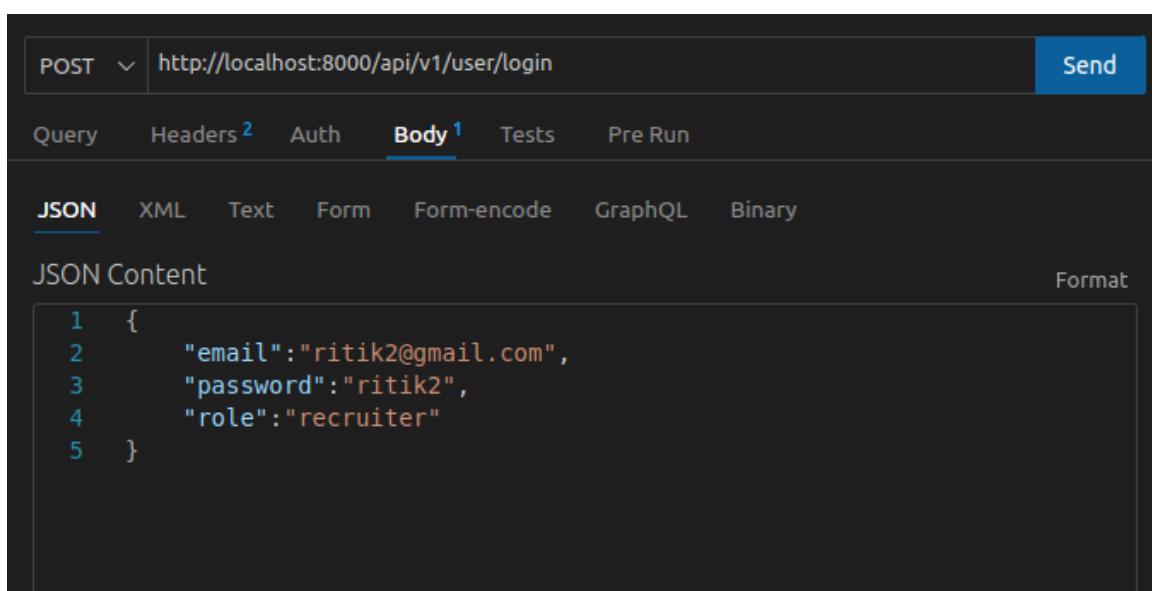


API Endpoints:

Base URL: We have made 4 base urls, user, company, job and application.

```
// api's
app.use("/api/v1/user", userRoute);
app.use("/api/v1/company", companyRoute);
app.use("/api/v1/job", jobRoute);
app.use("/api/v1/application", applicationRoute);
```

1. POST /user/login



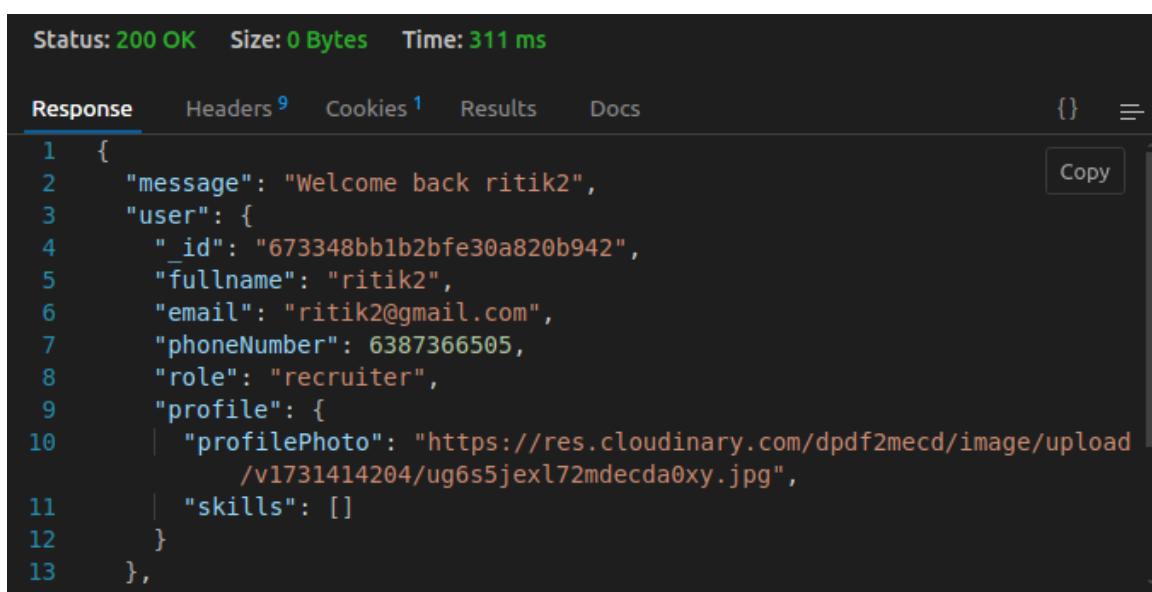
POST <http://localhost:8000/api/v1/user/login> Send

Query Headers ² Auth Body ¹ Tests Pre Run

JSON XML Text Form Form-encode GraphQL Binary

JSON Content Format

```
1  {
2    "email": "ritik2@gmail.com",
3    "password": "ritik2",
4    "role": "recruiter"
5 }
```



Status: 200 OK Size: 0 Bytes Time: 311 ms

Response Headers ⁹ Cookies ¹ Results Docs { } ⚡

```
1  {
2    "message": "Welcome back ritik2",
3    "user": {
4      "_id": "673348bb1b2bfe30a820b942",
5      "fullname": "ritik2",
6      "email": "ritik2@gmail.com",
7      "phoneNumber": 6387366505,
8      "role": "recruiter",
9      "profile": {
10        "profilePhoto": "https://res.cloudinary.com/dpdf2mecd/image/upload/v1731414204/ug6s5jexl72mdecda0xy.jpg",
11        "skills": []
12      }
13    },
14  }
```

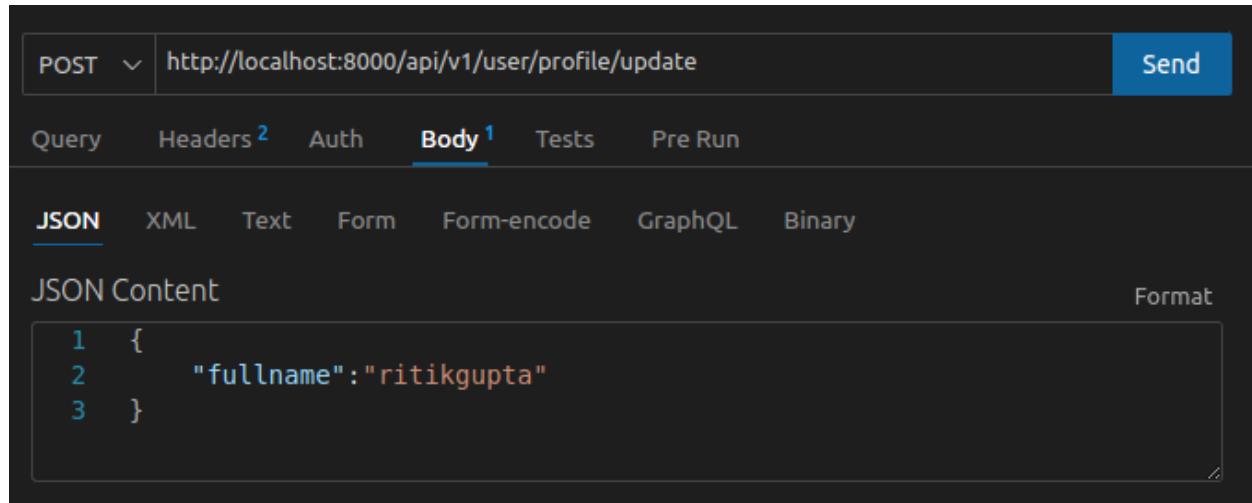
2. GET /user/logout

The screenshot shows the POSTMAN interface with a GET request to `http://localhost:8000/api/v1/user/logout`. The 'Query' tab is selected, showing an empty table for query parameters. A 'Send' button is visible in the top right.

The screenshot shows the POSTMAN response section. The status is **200 OK**, size is **0 Bytes**, and time is **16 ms**. The 'Response' tab is selected, displaying the following JSON data:

```
1  {
2      "message": "Logged out successfully.",
3      "success": true
4  }
```

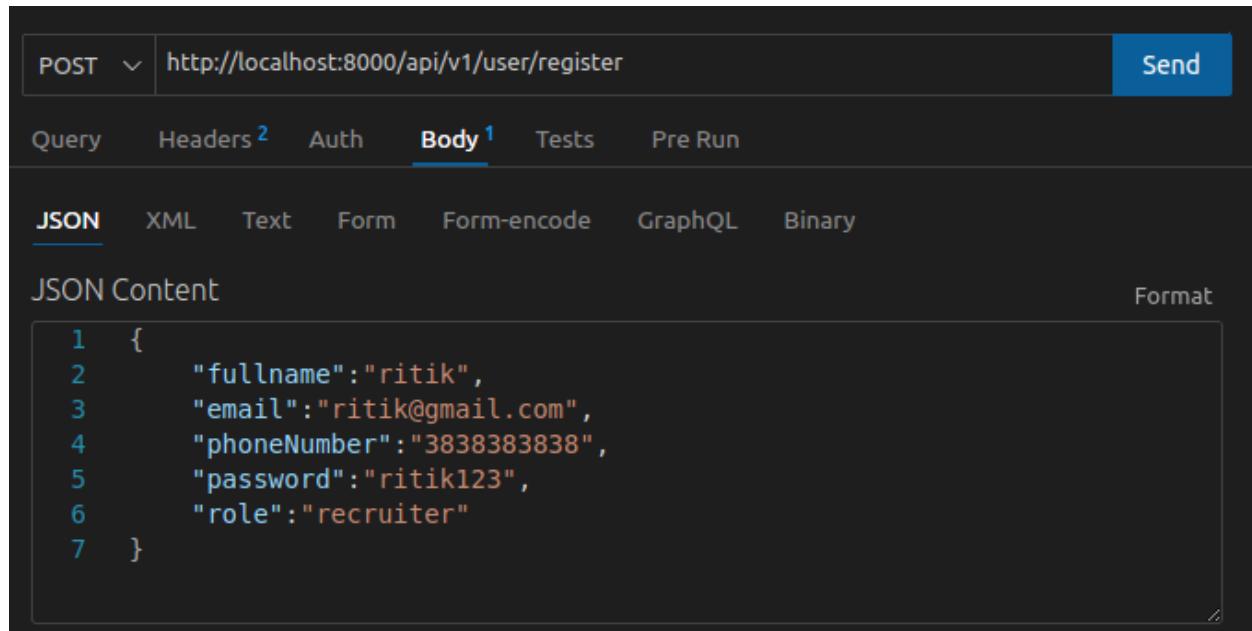
3. POST /user/profile/update



The screenshot shows the Postman application interface. At the top, there is a header bar with a dropdown menu set to "POST", a URL input field containing "http://localhost:8000/api/v1/user/profile/update", and a blue "Send" button. Below the header, there are tabs for "Query", "Headers 2", "Auth", "Body 1", "Tests", and "Pre Run". The "Body" tab is currently selected and has a sub-tab "JSON" which is also selected. In the "JSON Content" area, there is a code block with the following JSON:

```
1 {  
2     "fullname": "ritikgupta"  
3 }
```

4. POST /user/register



The screenshot shows the Postman application interface. At the top, there is a header bar with a dropdown menu set to "POST", a URL input field containing "http://localhost:8000/api/v1/user/register", and a blue "Send" button. Below the header, there are tabs for "Query", "Headers 2", "Auth", "Body 1", "Tests", and "Pre Run". The "Body" tab is currently selected and has a sub-tab "JSON" which is also selected. In the "JSON Content" area, there is a code block with the following JSON:

```
1 {  
2     "fullname": "ritik",  
3     "email": "ritik@gmail.com",  
4     "phoneNumber": "3838383838",  
5     "password": "ritik123",  
6     "role": "recruiter"  
7 }
```

User Interface

The screenshot shows the JobPortal application running in a browser window. The title bar indicates "Vite + React" and the address bar shows "Not secure 192.168.49.2:30008". The header features the "JobPortal" logo, navigation links for "Home", "Jobs", and "Browse", and buttons for "Login" and "Signup". A banner at the top reads "No. 1 Job Hunt Website" and "Search, Apply & Get Your Dream Jobs". Below the banner is a search bar with placeholder text "Find your dream jobs" and a magnifying glass icon. To the right of the search bar is a "Logout" button with the message "Logged out successfully.". On the left, there's a card for a job listing: "Google India SDE Intern Full stack intern 5 Positions Intern 150000LPA". On the right, there are navigation arrows for "Frontend Developer" and "Backend Developer".

Home Page

The screenshot shows the JobPortal application displaying search results. The header includes the "JobPortal" logo, navigation links for "Home", "Jobs", and "Browse", and buttons for "Login" and "Signup". The main content area is titled "Search Results (1)". It shows a single job listing from Google India: "SDE Intern" position, posted 27 days ago, with "5 Positions" and "Intern" status, offering "150000LPA". There are "Details" and "Save For Later" buttons below the listing.

Can Browse Different Jobs

Vite + React Error Not secure 192.168.49.2:30008/signup

JobPortal

Home Jobs Browse Login Signup

Sign Up

Full Name: shourya

Email: shourya@gmail.com

Phone Number: 9999955555

Password: *****

Student Recruiter

Profile Choose File Screenshot f...7-43-31.png

Signup

Already have an account? [Login](#)

Login Page

Vite + React Error Not secure 192.168.49.2:30008

JobPortal

Home Jobs Browse

No. 1 Job Hunt Website

Search, Apply & Get Your Dream Jobs

Find your dream jobs

Frontend Developer Backend Developer

Latest & Top Job Openings

Google India SDE Intern Full stack intern 5 Positions Intern 150000LPA

Welcome back ritik1

After User Logged in to

JobPortal

Filter Jobs

Location

- Delhi NCR
- Bangalore
- Hyderabad
- Pune
- Mumbai

Industry

- Frontend Developer
- Backend Developer
- FullStack Developer

Salary

- 0-40k
- 42-1lakh
- 1lakh to 1.5lakh

27 days ago

Google India

SDE Intern

Full stack intern

5 Positions | Intern | 150000LPA

[Details](#) [Save For Later](#)

Home Jobs Browse Login Signup

Jobs Search Page

JobPortal

[Companies](#) [Jobs](#)

Filter by name

New Company

Logo	Name	Date	Action
	Google	2024-11-12	...
	Salesforce	2024-11-12	...
	Amazon	2024-11-12	...

A list of your recent registered companies

Welcome back ritik2

Recruiter Page Showcasing Different Companies

**Your Company Name**

What would you like to give your company name? You can change this later.

Company Name

NetApp

[Cancel](#) [Continue](#)

Adding another company on the platform

[← Back](#) Company Setup

Company Name

NetApp

Description

Website

Location

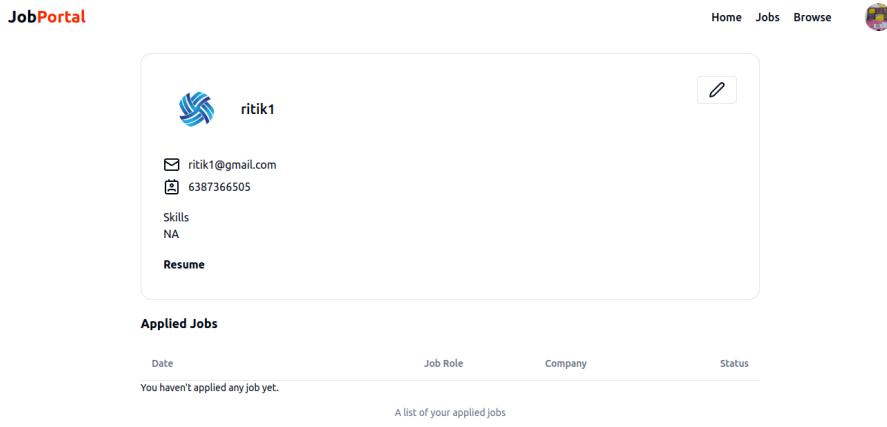
Logo

Choose file No file chosen

[Update](#)

Company registered successfully.

Filling further details related to the company.



User Profile page

Project Design and Framework

Version Control and Project Setup

1. Create an account and a new repository at <https://github.com/>. Click on Code and copy the repository URL.
2. Clone the Repository:
 - a. `git clone <repository url>`
 - b. `cd <project-directory>`
3. Open the project in your IDE (VSCode, IntelliJ IDEA, etc)
4. Install Dependencies:
 - a. For Backend -
 - i. `cd backend`
 - ii. `npm install`
 - b. For Frontend -
 - i. `cd frontend`
 - ii. `npm install`
5. Setup Environment Variables:

- a. Create an `.env` file in both the backend and frontend directories if not already present.
 - i. For Backend:

```
MONGO_URI=
PORT=
SECRET_KEY=
CLOUD_NAME=
API_KEY=
API_SECRET=
CLIENT_ORIGIN_URL=http://localhost:5173 # this is for local
# CLIENT_ORIGIN_URL="http://192.168.49.2:30008" #this is for kubernetes
```

- ii. For Frontend:

```
⚙ .env backend ⚙ .env frontend ×
frontend > ⚙ .env
1 VITE.REACT_APP_API_BASE_URL=http://localhost:8000
2 #this is for local
3 # VITE.REACT_APP_API_BASE_URL=http://192.168.49.2:30007
4 #this is for kubernetes
```

6. Run the application:

- a. Start the Backend server:
 - i. `npm run dev`

```
❖ ritik@ritik:~/Downloads/atalent_bridge/talentbridge/backend$ npm run dev
> backend@1.0.0 dev
> nodemon index.js

[nodemon] 3.1.3
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): ***!
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node index.js`
Servers running at port 8000
mongodb connected successfully
```

- b. Start the frontend server:
 - i. `npm run dev`

```
* History restored

○ ritik@ritik:~/Downloads/atalent_bridge/talentbridge/frontend$ npm run dev

> frontend@0.0.0 dev
> vite --host 0.0.0.0

VITE v5.3.1  ready in 350 ms

→ Local: http://localhost:5173/
→ Network: http://172.16.228.168:5173/
→ press h + enter to show help
```

For committing the change to the application, Run the following commands in your terminal:

1. git init
2. git add .
3. git commit -m " initial commit "
4. git remote add origin <REPOSITORY_URL>
5. git remote add upstream <REPOSITORY_URL>
6. git push origin main

Testing:

To run Test:

1. cd backend
2. npm run test

Key Scenarios:

1. **applyJob API**
 - Success: Creates a new application.
 - Errors: Handles duplicate applications and non-existent jobs.
2. **updateStatus API**
 - Success: Updates application status.
 - Errors: Handles missing status and invalid application ID.

```

backend > tests > js application.spec.js > describe('Job Application Controller') callback > describe('updateStatus') callback
  1 import { expect } from 'chai';
  2 import sinon from 'sinon';
  3 import { Application } from '../models/application.model.js';
  4 import { Job } from '../models/job.model.js';
  5 import { applyJob, getAppliedJobs, getApplicants, updateStatus } from '../controllers/application.controller.js';
  6
  7 describe('Job Application Controller', () => {
  8   afterEach(() => {
  9     sinon.restore();
 10   });
 11
 12   describe('applyJob', () => {
 13     it('should create a new application if the user has not applied before', async () => {
 14       const req = {
 15         id: '123456',
 16         params: { id: '789012' }
 17     };
 18     const res = {
 19       status: sinon.stub().returns({
 20         json: sinon.stub()
 21       })
 22     };
 23   });
  
```

```

ritik@ritik:~/Downloads/atalent_bridge/talentbridge/backend$ npm run test

> backend@1.0.0 test
> mocha ./tests/*.spec.js

Job Application Controller
  applyJob
    ✓ should create a new application if the user has not applied before
    ✓ should return an error if the user has already applied for the job
    ✓ should return an error if the job is not found
  updateStatus
    ✓ should update the status of an application
    ✓ should return an error if the status is not provided
    ✓ should return an error if the application is not found

  6 passing (44ms)

ritik@ritik:~/Downloads/atalent_bridge/talentbridge/backend$ 

```

Containerization:

Containerization allows your application to run consistently across different environments by encapsulating the application and its dependencies into a container.

1. Docker File for Frontend:

```
Frontend > 📄 Dockerfile > ...
1  # Dockerfile for React frontend
2
3  # Build react frontend
4  FROM node:20-alpine
5
6  # Working directory be app
7  WORKDIR /app
8
9
10 COPY package*.json ./
11
12 ### Installing dependencies
13
14 RUN npm install
15
16 # copy local files to app folder
17 COPY . .
18
19 EXPOSE 5173
20
21 CMD [ "npm", "run", "dev" ]
22
23
```

Build the Docker Image: From the root of the frontend directory, build the Docker image using the following command:

- a. docker build -t ritikgupta0114/frontend .
- b. docker run -p 5173:5173 frontend
- c. docker ps

```

ritik@ritik:~/Downloads/atalent_bridge/talentbridge/frontend$ docker build -t ritikgupta0114/frontend .
[+] Building 20.9s (8/10)
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 309B
=> [internal] load metadata for docker.io/library/node:20-alpine
=> [auth] library/node:pull token for registry-1.docker.io
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load build context
=> => transferring context: 1.74MB
=> [1/5] FROM docker.io/library/node:20-alpine@sha256:426f843809ae05f324883afceebaa2b9cab9cb697097dbb1a2a7a41c5701de72
=> => resolve docker.io/library/node:20-alpine@sha256:426f843809ae05f324883afceebaa2b9cab9cb697097dbb1a2a7a41c5701de72
=> sha256:426f843809ae05f324883afceebaa2b9cab9cb697097dbb1a2a7a41c5701de72
=> sha256:effclee0b93c4db0aab5729f3bc25d9df841e9b888ec3c4683d5a948a37553244 1.72kB / 1.72kB
=> sha256:2215267afb33d93392d8d16b41c461a569bf52a442e5af4c4add1371b73f26e3 6.18kB / 6.18kB
=> sha256:38a8310d387e0ecfcfab047a9149e8eb214073db9f461fee6251fd936a75 3.64MB / 3.64MB
=> sha256:65052e355180ad17acd4b61613338090c587b931124cd2946b9dcf0e945b76cb 42.54MB / 42.54MB
=> sha256:9cf3157062e6e87bb84921361434e0c930e8f73584eb06114cf7563178355253 1.26MB / 1.26MB
=> sha256:dcbf660a4e70b11e73b64689666db5db05706d6d6a9bb950d9fea8593712af3c 446B / 446B
=> => extracting sha256:38a8310d387e375e0ecfcfab047a9149e8eb214073db9f461fee6251fd936a75
=> => extracting sha256:65052e355180ad17acd4b61613338090c587b931124cd2946b9dcf0e945b76cb
=> => extracting sha256:9cf3157062e6e87bb84921361434e0c930e8f73584eb06114cf7563178355253
=> => extracting sha256:dcbf660a4e70b11e73b64689666db5db05706d6d6a9bb950d9fea8593712af3c
=> [2/5] WORKDIR /app
=> [3/5] COPY package*.json .
=> [4/5] RUN npm install

```

```

ritik@ritik:~/Downloads/atalent_bridge/talentbridge/frontend$ docker run -p 5173:5173 ritikgupta0114/frontend
> frontend@0.0.0 dev
> vite --host 0.0.0.0

Re-optimizing dependencies because vite config has changed

  VITE v5.3.1 ready in 438 ms

  → Local: http://localhost:5173/
  → Network: http://172.17.0.2:5173/

```

This will expose the app on port 5173. You can now access the React app at <http://localhost:5173> in your browser.

2. Docker File for Backend:

```

backend > 🚀 Dockerfile > ...
1  # Dockerfile for Backend
2
3  FROM node:20-alpine
4
5  # Create App Directory
6  RUN mkdir -p /app
7  WORKDIR /app
8
9  # Install Dependencies
10 COPY package*.json .
11
12 RUN npm install
13
14 RUN npm install morgan
15
16 # Copy app source code
17 COPY . .
18
19 # Exports
20 EXPOSE 8000
21
22 CMD ["npm", "run", "dev"]

```

Build the Docker Image: From the root of the frontend directory, build the Docker image using the following command:

- a. `docker build -t ritikgupta0114/backend .`
- b. `docker run -p 8000:8000 frontend`
- c. `docker ps`

```
○ ritik@ritik:~/Downloads/atalent_bridge/talentbridge/backend$ docker build -t ritikgupta0114/backend .
[+] Building 4.3s (8/11)
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 310B
=> [internal] load metadata for docker.io/library/node:20-alpine
=> [internal] load .dockerignore
=> => transferring context: 2B
=> CACHED [1/7] FROM docker.io/library/node:20-alpine@sha256:426f843809ae05f324883afceebaa2b9cab9cb697097dbb1a2a7a41c5701de72
=> [internal] load build context
=> => transferring context: 826.09kB
=> [2/7] RUN mkdir -p /app
=> [3/7] WORKDIR /app
=> [4/7] COPY package*.json ./
=> [5/7] RUN npm install
```



```
○ ritik@ritik:~/Downloads/atalent_bridge/talentbridge/backend$ docker run -p 8000:8000 ritikgupta0114/backend
> backend@1.0.0 dev
> nodemon index.js

[nodemon] 3.1.3
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): ***!
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node index.js`
Servers running at port 8000
mongodb connected successfully
```

This will expose the app on port 5173. You can now access the React app at <http://localhost:8000> in your browser.

Pushing to DockerHub:

General Tags Builds Collaborators Webhooks Settings

ritikgupta0114/frontend 🐳

Last pushed about 10 hours ago

Add a description 🎧 INCOMPLETE

Add a category 🎧 INCOMPLETE

Tags

This repository contains 1 tag(s).

Tag	OS	Type	Pulled	Pushed
latest	🐧	Image	10 hours ago	10 hours ago

[See all](#)

General Tags Builds Collaborators Webhooks Settings

ritikgupta0114/backend 🐳

Last pushed about 10 hours ago

Add a description 🎧 INCOMPLETE

Add a category 🎧 INCOMPLETE

Tags

This repository contains 1 tag(s).

Tag	OS	Type	Pulled	Pushed
latest	🐧	Image	10 hours ago	10 hours ago

[See all](#)

Docker Compose:

Docker Compose is used to define and manage multi-container Docker applications. Below is the configuration for your application, which includes services for the backend, frontend, Elasticsearch, Logstash, and Kibana.

```
⚙ .env backend ⚙ docker-compose.yml ✘ ⚙ .env ./
```

```
docker-compose.yml
 1 services:
 2   server:
 3     image: ritikgupta0114/backend:latest
 4     container_name: myapp-node-server
 5     ports:
 6       - "8000:8000"
 7     environment:
 8       - NODE_ENV=development
 9       - MONGO_URI
10       - LOGSTASH_HOST=logstash:5044
11     networks:
12       - app-network
13     depends_on:
14       - client
15       - logstash
16     volumes:
17       - /app/node_modules
18       - /home/ritik/Downloads/atalent_bridge/talentbridge/app.log:/app/app.log
19     dns:
20       - 8.8.8.8
21
22   client:
23     image: ritikgupta0114/frontend:latest
24     container_name: myapp-react-client
25     ports:
26       - "5173:5173"
27     environment:
28       - NODE_ENV=development
29     networks:
30       - app-network
31     volumes:
32       - /app/node_modules
33
```

```

34     elasticsearch:
35       image: docker.elastic.co/elasticsearch/elasticsearch:7.17.10
36       container_name: elasticsearch
37       environment:
38         - discovery.type=single-node
39         - ES_JAVA_OPTS=-Xms512m -Xmx512m
40       networks:
41         - app-network
42       volumes:
43         - elasticsearch-data:/usr/share/elasticsearch/data
44       ports:
45         - "9200:9200"
46
47   logstash:
48     image: docker.elastic.co/logstash/logstash:7.17.10
49     container_name: logstash
50     volumes:
51       - /home/ritik/Downloads/atalent_bridge/talentbridge/app.log:/app/app.log
52       - /home/ritik/Downloads/atalent_bridge/talentbridge/logstash.conf:/usr/share/logstash/pipeline/logstash.conf:ro
53     environment:
54       - xpack.monitoring.enabled=false
55     networks:
56       - app-network
57     depends_on:
58       - elasticsearch
59     ports:
60       - "5044:5044"
61
62   kibana:
63     image: docker.elastic.co/kibana/kibana:7.17.10
64     container_name: kibana
65     environment:
66       - ELASTICSEARCH_HOSTS=http://elasticsearch:9200
67     networks:
68       - app-network
69     depends_on:
70       - elasticsearch
71     ports:
72       - "5601:5601"
73
74   networks:
75     app-network:
76       driver: bridge
77
78   volumes:
79     node_modules:
80     elasticsearch-data:
81       driver: local

```

Explanation of the Configuration

1. **server (Backend):**

- The backend service uses the **ritikgupta0114/backend:latest** Docker image.
- The server runs on port **8000** and communicates with Logstash for log shipping.
- Dependencies such as MongoDB and Logstash are also included as **depends_on**.

2. **client (Frontend):**

- The frontend service uses the **ritikgupta0114/frontend:latest** Docker image.

- The frontend runs on port **5173** and uses shared network configurations.

3. **elasticsearch:**

- Elasticsearch is configured to run as a single-node instance, exposed on port **9200**.

4. **logstash:**

- Logstash is configured to process logs, with a configuration file **logstash.conf**.
- It runs on port **5044** to receive logs from the backend.

5. **kibana:**

- Kibana connects to Elasticsearch for visualizing and managing logs.
- It is exposed on port **5601**.

6. **Networks and Volumes:**

- The services are connected through a bridge network named **app-network**.
- Volumes are used for persistent storage (**node_modules** and **elasticsearch-data**).

Explanation of Volumes

In Docker Compose, volumes are used to persist data across container restarts and to share data between containers. Volumes can also help in mapping files from your host machine to containers for easy access and modification.

Here's how volumes are used in your configuration:

1. **Backend Volume (`/app/node_modules`):** This volume ensures that the **node_modules** directory in the backend container is persisted and is shared between the host machine and the container, preventing the need to re-install dependencies every time the container starts.
2. **Log Volume (`/app/app.log`):** The log file from your backend application is shared with the **logstash** service to ship logs to Elasticsearch for analysis. This volume maps the log file from your local system

(`/home/ritik/Downloads/atalent_bridge/talentbridge/app.log`) to the container's log file path.

3. **Elasticsearch Data Volume (`elasticsearch-data`):** This volume ensures that Elasticsearch's data is persistent, meaning data won't be lost when the container is stopped or restarted. The volume is mapped to Elasticsearch's data directory inside the container.
4. **Logstash Configuration Volume (`logstash.conf`):** This volume maps the local `logstash.conf` file (configuration for Logstash) to the Logstash container. The file is used to configure how Logstash processes the logs from the backend.

Note: The `:ro` suffix ensures that the file is mounted as read-only, meaning the container cannot modify the host file.

Steps to Use Docker Compose:

1. **Create the `.env` File:** In the root directory of your project, create a file named `.env`. This file will store environment variables that will be used in the Docker Compose setup.

```
⚙ .env
1 NODE_ENV=
2 MONGO_URI=
3
```

2. **Start All Services:** From the project directory where the `docker-compose.yml` file is located, run:
 - a. `docker-compose up`
3. **Verify Running Containers:** Check the status of the containers:
 - a. `docker ps`
4. **Stop All Services:** To stop all services, use:
 - a. `docker-compose down`

```
ritik@ritik:~/Downloads/atalent_bridge/talentbridge$ docker compose up
[+] Running 5/0
  ✓ Container elasticsearch    Created          0.0s
  ✓ Container myapp-react-client Created          0.0s
  ✓ Container kibana           Created          0.0s
  ✓ Container logstash         Created          0.0s
  ✓ Container myapp-node-server Created          0.0s
Attaching to elasticsearch, kibana, logstash, myapp-node-server, myapp-react-client
logstash      | 2024/12/10 04:40:57 Setting 'xpack.monitoring.enabled' from environment.
logstash      | Using bundled JDK: /usr/share/logstash/jdk
myapp-react-client | > frontend@0.0.0 dev
myapp-react-client | > vite --host 0.0.0.0
myapp-react-client | > vite --host 0.0.0.0
logstash      | OpenJDK 64-Bit Server VM warning: Option UseConcMarkSweepGC was deprecated in version 9.0 and will likely be removed in a future r
elease.
myapp-node-server | > backend@1.0.0 dev
myapp-node-server | > nodemon index.js

myapp-react-client | VITE v5.3.1 ready in 867 ms
myapp-react-client | → Local: http://localhost:5173/
myapp-react-client | → Network: http://172.23.0.2:5173/
myapp-node-server | [nodemon] 3.1.3
myapp-node-server | [nodemon] to restart at any time, enter `rs`
myapp-node-server | [nodemon] watching path(s): ***!
myapp-node-server | [nodemon] watching extensions: js,mjs,cjs,json
myapp-node-server | [nodemon] starting `node index.js`
myapp-node-server | Servers running at port 8000
myapp-node-server | mongodb connected successfully
```

Configuration Management:

Configuration management is the process of automating the management of system configurations to ensure that they are consistent, reliable, and up-to-date. It involves managing infrastructure as code, which means that configurations are defined in code and can be version-controlled, tested, and automated.

Running Multi-Container setup - Docker-Compose:

```
☒ inventory
1 [ansible_nodes]
2 localhost ansible_user=ritik ansible_password=@my_vault.yml ansible_python_interpreter=/usr/bin/python3
3
4 [ansible_nodes:vars]
5 ansible_connection=local
6
```

```
! playbook-new.yaml
1   - name: Deploying with Docker-Compose
2     hosts: all
3     become: true
4     roles:
5       - git_clone
6       - deploy_docker_compose
7
```

Playbook Explanation:

1. Purpose:

The playbook delegates tasks such as cloning the repository and deploying services to predefined Ansible roles for better organization and reusability.

2. Hosts:

The playbook runs on all specified hosts in the Ansible inventory.

3. Tasks:

- **Git Clone Role:** Handles cloning the repository containing the Docker Compose configuration.
- **Docker Compose Deployment Role:** Executes the deployment using Docker Compose.

4. Privilege Escalation:

The `become: true` directive ensures that tasks requiring elevated privileges (e.g., Docker commands) are executed with sudo.

1.

Run the Playbook:

Execute the playbook using the `ansible-playbook` command:

ansible-playbook -i inventory playbook-new.yaml

Running Container Orchestration- Kubernetes:

```
≡ inventory-k8
1 [ansible_nodes]
2 localhost ansible_user=ritik ansible_password=@my_vault.yml ansible_python_interpreter=/usr/bin/python3
3
4 [ansible_nodes:vars]
5 ansible_connection=local
6
```

```
! playbook-k8-new.yaml
1 - name: Deploy Frontend and Backend with Kubernetes
2   hosts: all
3   tasks:
4     - name: Show ansible_user
5       debug:
6         msg: "The ansible_user is {{ ansible_user }}"
7
8     - name: Clone Repository
9       include_role:
10         name: clone_repo
11
12    - name: Deploy Frontend Resources
13      include_role:
14        name: deploy_frontend
15
16    - name: Deploy Backend Resources
17      include_role:
18        name: deploy_backend
```

Playbook Explanation

1. Purpose:

Automates the deployment of frontend and backend Kubernetes resources, separating tasks into modular roles.

2. Hosts:

The playbook is executed on all specified hosts in the Ansible inventory.

3. Tasks:

- **Show Ansible User:** Verifies and displays the Ansible user running the playbook.
- **Clone Repository:** Uses the `clone_repo` role to fetch the codebase from a specified repository.
- **Deploy Frontend Resources:** Uses the `deploy_frontend` role to apply the necessary Kubernetes manifests for the frontend application.
- **Deploy Backend Resources:** Uses the `deploy_backend` role to apply the Kubernetes manifests for the backend application.

Run the Playbook:

Execute the playbook using the `ansible-playbook` command:

```
ansible-playbook -i inventory-k8 playbook-k8-new.yaml
```

Ngrok and Github Webhooks

Whenever the developer builds the code commits it and push that to Github, webhooks automatically builds the code in Jenkins. To perform a webhook, the private IP address of the local machine needs to be converted to a public IP address. Automated messages known as "webhooks" are triggered when changes occur. Specifically, in our situation, updates made to the GitHub repository will activate the Jenkins pipeline through the webhook.

Jenkins can use webhooks for triggering builds. A webhook (or web callback) is a way for an app to provide other applications with real-time information. This can ensure that GitHub will send a webhook to our Jenkins server and our Jenkins application will build and test the changes and send the build status for the commit that was made. However the IP address must be a public one, hence we use ngrok to create a public address and use that to configure the webhook. Once the webhook is set up, all commits to the remote repository, send a POST

request to the payload ngrok URL, which in turn tells the Jenkins server to trigger the build.

To start ngrok: **ngrok http 8080**

```
ngrok
(Ctrl+C to quit)

👋 Goodbye tunnels, hello Agent Endpoints: https://ngrok.com/r/aep

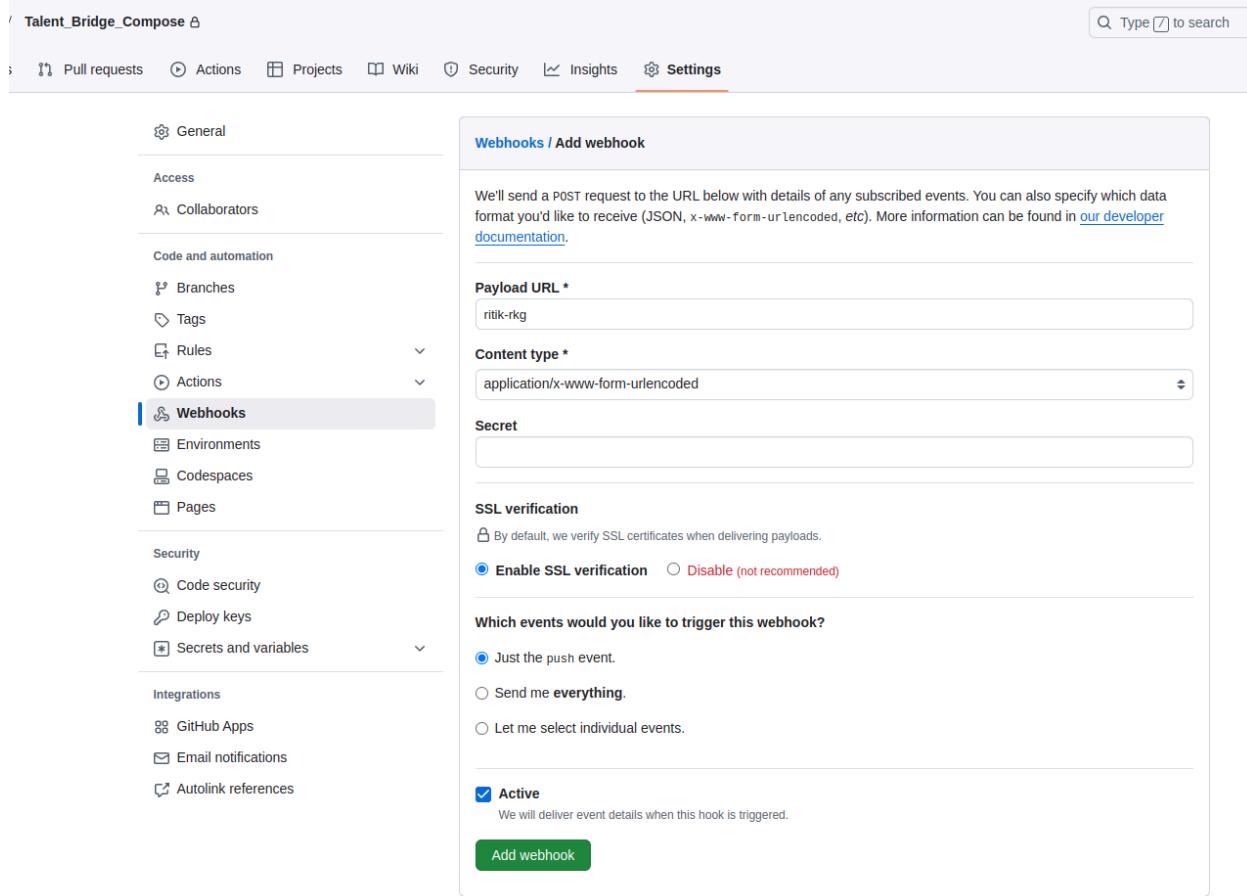
Session Status          online
Account                 Ritik (Plan: Free)
Version                3.18.4
Region                 India (in)
Web Interface          http://127.0.0.1:4040
Forwarding             https://f653-119-161-98-68.ngrok-free.app -> http://localhost:8080

Connections            ttl     opn     rt1     rt5     p50     p90
                        0       0       0.00   0.00   0.00   0.00
```

Setting Up WebHooks :

Navigate to the settings section of the Github repository, then proceed to the Webhooks tab and select Add webhook. Add the ngrok address in payload URL and the personal access token in secret.

The screenshot shows the GitHub repository settings interface for 'Talent_Bridge_Compose'. The top navigation bar includes 'Pull requests', 'Actions', 'Projects', 'Wiki', 'Security', 'Insights', and 'Settings'. The 'Settings' tab is active. On the left, there's a sidebar with 'General', 'Access', 'Collaborators', 'Code and automation' (with 'Branches', 'Tags', 'Rules', 'Actions', and 'Webhooks' listed), and 'Environments'. The 'Webhooks' tab is currently selected. A sub-section titled 'Webhooks' explains that they allow external services to be notified for certain events, with a link to the 'Webhooks Guide'. A 'Add webhook' button is visible in the top right of this section.



If Webhook is already set up then just edit it and copy and paste the new ngrok public IP address.

Continuous Integration Pipeline for Docker-Compose Setup:

This Jenkins pipeline automates the build, test, and deployment process for a project using Docker Compose. Below is a detailed breakdown of the steps required to set up Jenkins, an explanation of each stage in the Jenkinsfile, and instructions to execute the pipeline.

Steps to Set Up Jenkins

1. Configure Node.js:

- Go to **Manage Jenkins > Global Tool Configuration**.
- Under **NodeJS**, add a Node.js version and name it (e.g., "NODEJS").

NodeJS installations

The screenshot shows the Jenkins 'NodeJS installations' configuration page. At the top, there's a header with 'NodeJS installations ^' and an 'Edited' status. Below it is a button labeled 'Add NodeJS'. The main section is titled 'NodeJS' and contains a 'Name' field with 'NODEJS' and an 'Install automatically' checkbox which is checked. A dashed box encloses the 'Install from nodejs.org' section, which includes a 'Version' field with 'NodeJS 20.12.1'. Below this is a note about forcing 32-bit architecture if available, with a corresponding checkbox.

2. Set Up Credentials:

- Navigate to **Manage Jenkins > Credentials**.
- Add the following credentials:
 - **Docker Hub Credentials** (ID: `Docker_Credentials_ritik`).
 - **MongoDB URI** (ID: `mongo-uri`).
 - **Cloudinary Credentials** (`cloud_secret_key`, `cloud_name`, `cloud_api_key`, `cloud_api_secret`).
 - **Ansible Vault Password** (ID: `ansible_vault_pass`).

	System	(global)	githubcredential	ritik-rkg/*****
	System	(global)	cloud_api_key	cloud_api_key
	System	(global)	cloud_name	cloud_name
	System	(global)	mongo-uri	mongo-uri
	System	(global)	cloud_secret_key	cloud_secret_key
	System	(global)	cloud_api_secret	cloud_api_secret
	System	(global)	Docker_Credentials_ritik	ritikgupta0114/*****
	System	(global)	ansible_vault_pass	ansible_vault_pass

Jenkinsfile Explanation

Pipeline Configuration

- **Environment Variables:**
 - Securely fetch credentials like Docker Hub credentials, MongoDB URI, and Cloudinary keys from Jenkins.
 - Set custom environment variables (e.g., `PORT`, `MINIKUBE_HOME`).
- **Tools:**
 - Specify the Node.js version configured in Jenkins

```
 Jenkinsfile
1 pipeline {
2     environment {
3         DOCKERHUB_CRED = credentials("Docker_Credentials_ritik")
4         MONGO_URI = credentials("mongo-uri")
5         SECRET_KEY = credentials("cloud_secret_key")
6         CLOUD_NAME = credentials("cloud_name")
7         API_KEY = credentials("cloud_api_key")
8         API_SECRET = credentials("cloud_api_secret")
9         PORT = "8000"
10        MINIKUBE_HOME = '/home/jenkins/.minikube'
11        VAULT_PASS = credentials("ansible_vault_pass")
12        BECOME_PASS = credentials("ansible_vault_pass")
13    }
14    agent any
15    tools {nodejs "NODEJS"}
```

Stages

1. Stage 1: Git Clone

- Deletes any existing project directory and clones the repository.
- Ensures a clean workspace.

```
 stages {
    stage("Stage 1: Git Clone") {
        steps {
            sh '''
                [ -d Talent_Bridge_Compose ] && rm -rf Talent_Bridge_Compose
                git clone https://github.com/ritik-rkg/Talent_Bridge_Compose.git
                '''
        }
    }
}
```

2. Stage 2: Backend Testing

- Installs backend dependencies and testing tools (e.g., Mocha, Chai).
- Runs backend test scripts to validate functionality.

```
stage("Stage 2: Backend Testing") {
    steps {
        sh ''
        cd backend
        npm i
        cd tests
        npm install mocha chai sinon prom-client
        npm test
        ''
    }
}
```

3. Stage 3: Build Frontend

- Installs frontend dependencies and builds the project.
- Generates production-ready static files.

```
stage("Stage 3: Build frontend") {
    steps {
        sh ''
        cd Talent_Bridge_Compose/frontend
        npm install
        npm run build
        ''
    }
}
```

4. Stage 4: Remove Docker Images and Containers

- Cleans up unused Docker images and containers to free space.

```
stage("Stage 4: Remove docker images and container") {
    steps {
        sh "docker container prune -f"
        sh "docker image prune -a -f"
    }
}
```

5. Stage 5: Create Docker Image for Frontend

- Builds a Docker image for the frontend application using its Dockerfile.

```
stage("Stage 5: Creating Docker Image for frontend") {
    steps {
        sh ''
        cd Talent_Bridge_Compose/frontend
        docker build -t ritikgupta0114/frontend:latest .
        ''
    }
}
```

6. Stage 6: Scan Docker Image for Frontend

- Uses Trivy to scan the Docker image for vulnerabilities.

```
stage("Stage 6: Scan Docker Image for frontend") {
    steps {
        sh ''
        trivy image ritikgupta0114/frontend:latest
        ''
    }
}
```

7. Stage 7: Push Frontend Docker Image

- Logs into Docker Hub and pushes the frontend image to the repository.

```
stage("Stage 7: Push Frontend Docker Image") {
    steps {
        sh ''
        docker login -u ${DOCKERHUB_CRED_USR} -p ${DOCKERHUB_CRED_PSW}
        docker push ritikgupta0114/frontend:latest
        ''
    }
}
```

8. Stage 8: Create Docker Image for Backend

- Builds a Docker image for the backend application using its Dockerfile.

```
stage("Stage 8: Creating Docker Image for backend") {
    steps {
        sh ''
        cd Talent_Bridge_Compose/backend
        docker build -t ritikgupta0114/backend:latest .
        ''
    }
}
```

9. Stage 9: Scan Docker Image for Backend

- Scans the backend Docker image for vulnerabilities using Trivy.

```
stage("Stage 9: Scan Docker Image for backend") {
    steps {
        sh ''
        trivy image ritikgupta0114/backend:latest
        ''
    }
}
```

10. Stage 10: Push Backend Docker Image

- Pushes the backend Docker image to Docker Hub.

```
stage("Stage 10: Push Backend Docker Image") {
    steps {
        sh ''
        docker login -u ${DOCKERHUB_CRED_USR} -p ${DOCKERHUB_CRED_PSW}
        docker push ritikgupta0114/backend:latest
        ''
    }
}
```

11. Stage 11: Ansible Deployment

- Executes an Ansible playbook to deploy the application using Docker Compose.
- Handles sensitive credentials securely and cleans them after use.

```

stage("Stage 11: Ansible") {
    steps {
        sh ''
        cd Talent_Bridge_Compose
        echo "$VAULT_PASS" > /tmp/vault_pass.txt
        chmod 600 /tmp/vault_pass.txt

        # Export become password if needed
        echo "$BECOME_PASS" > /tmp/become_pass.txt
        chmod 600 /tmp/become_pass.txt

        # Run the Ansible playbook with sudo privilege
        ansible-playbook -i inventory --vault-password-file /tmp/vault_pass.txt --extra-vars "ansible_become_pass=$(cat /tmp/become_pass.txt)" playbook-new.yaml

        # Clean up sensitive files
        rm -f /tmp/vault_pass.txt /tmp/become_pass.txt
        ...
    }
}

```

Steps to Run the Pipeline

1. Create a New Pipeline Job:

- Go to [Jenkins Dashboard](#) > [New Item](#).
- Choose "Pipeline" and provide a name for the job.

2. Configure the Pipeline:

- Under the "Pipeline" section, choose "Pipeline script" and paste the Jenkinsfile.

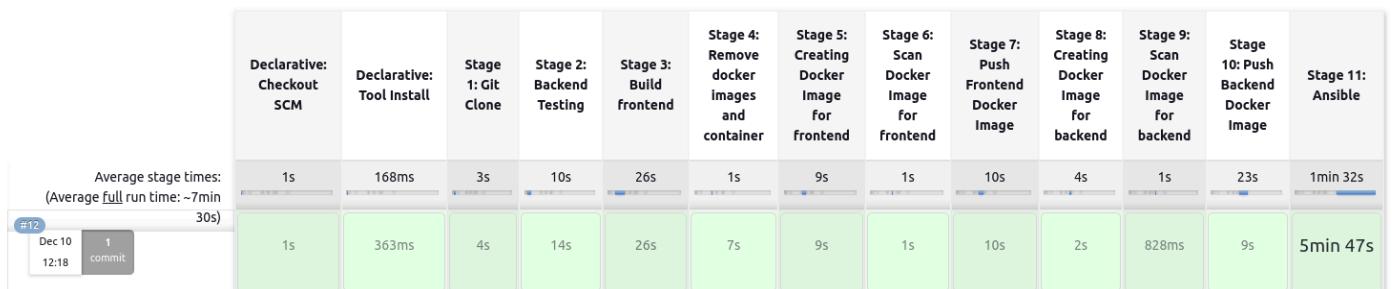
3. Trigger the Build:

- Save the job configuration and click "Build Now" to execute the pipeline.

Talent_Bridge_Compose

[Add description](#)

Stage View



```
Run `npm audit` for details.
```

```
+ npm test
```

```
> backend@1.0.0 test
```

```
> mocha ./tests/*.spec.js
```

Job Application Controller

applyJob

- ✓ should create a new application if the user has not applied before
- ✓ should return an error if the user has already applied for the job
- ✓ should return an error if the job is not found

updateStatus

- ✓ should update the status of an application
- ✓ should return an error if the status is not provided
- ✓ should return an error if the application is not found

```
6 passing (33ms)
```

```
[Pipeline] }
```

```
[Pipeline] // withEnv
```

```
+ cd Talent_Bridge_Compose
+ echo ****
+ chmod 600 /tmp/vault_pass.txt
+ echo ****
+ chmod 600 /tmp/become_pass.txt
+ cat /tmp/become_pass.txt
+ ansible-playbook -i inventory --vault-password-file /tmp/vault_pass.txt --extra-vars ansible_become_pass=**** playbook-new.yaml

PLAY [Deploying with Docker-Compose] ****
TASK [Gathering Facts] ****
ok: [localhost]

TASK [git_clone : Clone the repository for Docker Compose] ****
changed: [localhost]

TASK [deploy_docker_compose : Set COMPOSE_HTTP_TIMEOUT to 300] ****
changed: [localhost]

TASK [deploy_docker_compose : Deploy the application using Docker Compose] ****
changed: [localhost]

PLAY RECAP ****
localhost : ok=4    changed=3    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

+ rm -f /tmp/vault_pass.txt /tmp/become_pass.txt
```

```
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // withCredentials
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

Continuous Integration Pipeline for Kubernetes Setup:

This Jenkins pipeline automates the build, test, and deployment process for a project using Kubernetes. The stages from **1 to 10** are the same as in the Docker Compose setup, focusing on code cloning, building, testing, and pushing Docker images. In **Stage 11**, the application is deployed to a Kubernetes cluster using `kubectl` commands and manifests.

Stage 11: Ansible Deployment

- Executes an Ansible playbook to deploy the application using Docker Compose.
- Handles sensitive credentials securely and cleans them after use.

```
stage("Stage 11: Ansible"){
    steps {
        sh ...
        cd Talent_Bridge_K8s
        echo "$VAULT_PASS" > /tmp/vault_pass.txt
        chmod 600 /tmp/vault_pass.txt
        ansible-playbook -i inventory-k8 --vault-password-file /tmp/vault_pass.txt playbook-k8-new.yaml
        rm -f /tmp/vault_pass.txt
        ...
    }
}
```

Talent_Bridge_Kubernetes

[Add description](#)

Stage View



```
PLAY [Deploying with Kubernetes] ****
```

```
TASK [Gathering Facts] ****
ok: [localhost]
```

```
TASK [Show ansible_user] ****
ok: [localhost] => {
    "msg": "The ansible_user is ritik"
}
```

```
TASK [Clone the repository] ****
changed: [localhost]
```

```
TASK [Apply Kubernetes manifests] ****
changed: [localhost]
```

```
PLAY RECAP ****
localhost : ok=4    changed=2    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

```
+ rm -f /tmp/vault_pass.txt
```

```
+ rm -f /tmp/vault_pass.txt
[Pipeline]
[Pipeline] // withEnv
[Pipeline]
[Pipeline] // stage
[Pipeline]
[Pipeline] // withEnv
[Pipeline]
[Pipeline] // withEnv
[Pipeline]
[Pipeline] // withEnv
[Pipeline]
[Pipeline] // withEnv
[Pipeline]
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

Monitoring and Logging

Once deployed Monitoring is critical for maintaining the health and security of the deployment as well as the deployment infrastructure, as it allows us to quickly detect and respond to potential problems before they become critical.

Docker Compose: ELK Stack for Monitoring and Logging:

Backend Logging Code

The backend application utilizes **Morgan**, **Winston**, and **elasticsearch-winston** for structured logging. Here's how the logging is implemented:

`index.js`

```
app.use(cors(corsOptions));
app.use(
  morgan(morganFormat, {
    stream: {
      write: (message) => {
        const logObject = {
          method: message.split(" ")[0],
          url: message.split(" ")[1],
          status: message.split(" ")[2],
          responseTime: message.split(" ")[3],
        };
        logger.info(JSON.stringify(logObject));
      },
    },
  })
);
```

Explanation of the Code

1. Morgan:

- Used to log HTTP requests in a structured format.
- Outputs method, URL, status, and response time.
- Logs are passed to Winston via a custom `stream`.

2. Winston:

- A powerful logging library that supports multiple transports.

- Logs are sent to the console and Elasticsearch for centralized logging.
3. **elasticsearch-winston:**
- Sends logs to Elasticsearch for storage and analysis.
 - Configured with the Elasticsearch URL.
4. **Log Object:**
- Contains `method`, `url`, `status`, and `responseTime`.
 - JSON format ensures compatibility with Elasticsearch.

Explanation of `logger.js`

The `logger.js` file uses the **Winston** library to configure and manage application logging. It provides a flexible and customizable logging mechanism for the application.

1. Importing Modules
 - `createLogger`: Creates a logger instance.
 - `format`: Provides formatting utilities to structure log output.
 - `transports`: Defines where the logs are written (e.g., console, file).
 - Deconstruct specific formatting methods: `combine`, `timestamp`, `json`, `colorize`.
2. Custom Console Log Format
 - `format.colorize()`:
 - Adds color to log levels for better readability in the console.
 - `format.printf()`:
 - Customizes the log message format.
 - Outputs logs in the format: `level: message`
3. Creating the Logger
 - `level: "info"`:
 - Sets the minimum log level to `info`. Lower levels like `debug` won't be logged.
 - `format: combine()`:
 - Combines multiple formats:
 - `colorize()`: Adds color for console logs.
 - `timestamp()`: Adds timestamps to logs.
 - `json()`: Converts log messages to JSON format for structured logging.
 - `transports`:

- Specifies where the logs are written:
 - **Console**: Logs with a custom colorful format for debugging during development.
 - **File**: Logs are also written to `app.log` in the default JSON format for persistent storage.

```
Job Portal > backend > js logger.js > ...
1 import { createLogger, format, transports } from "winston";
2 const { combine, timestamp, json, colorize } = format;
3
4 // Custom format for console logging with colors
5 const consoleLogFormat = format.combine(
6   format.colorize(),
7   format.printf({ level, message, timestamp }) => {
8     return `${level}: ${message}`;
9   }
10 );
11
12 // Create a Winston logger
13 const logger = createLogger({
14   level: "info",
15   format: combine(colorize(), timestamp(), json()),
16   transports: [
17     new transports.Console({
18       format: consoleLogFormat,
19     }),
20     new transports.File({ filename: "app.log" })
21   ],
22 });
23
24 export default logger;
```

Setting up ElasticSearch, Logstash, and Kibana (ELK Stack) with Docker Compose:

```

34      elasticsearch:
35        image: docker.elastic.co/elasticsearch/elasticsearch:7.17.10
36        container_name: elasticsearch
37        environment:
38          - discovery.type=single-node
39          - ES_JAVA_OPTS=-Xms512m -Xmx512m
40      networks:
41        - app-network
42      volumes:
43        - elasticsearch-data:/usr/share/elasticsearch/data
44      ports:
45        - "9200:9200"
46
47    logstash:
48      image: docker.elastic.co/logstash/logstash:7.17.10
49      container_name: logstash
50      volumes:
51        - /home/ritik/Downloads/atalent_bridge/talentbridge/app.log:/app/app.log
52        - /home/ritik/Downloads/atalent_bridge/talentbridge/logstash.conf:/usr/share/logstash/pipeline/logstash.conf:ro
53      environment:
54        - xpack.monitoring.enabled=false
55      networks:
56        - app-network
57      depends_on:
58        - elasticsearch
59      ports:
60        - "5044:5044"
61
62  kibana:
63    image: docker.elastic.co/kibana/kibana:7.17.10
64    container_name: kibana
65    environment:
66      - ELASTICSEARCH_HOSTS=http://elasticsearch:9200
67    networks:
68      - app-network
69    depends_on:
70      - elasticsearch
71    ports:
72      - "5601:5601"

```

Steps to Run the ELK Stack:

- 1. Create Necessary Files:** Place your application logs (`app.log`) in the specified directory.
- 2. Create a `logstash.conf` file for Logstash pipeline configuration**

```

Job Portal > logstash.conf
1, input {
2,   file {
3,     path => "/app/app.log"
4,     start_position => "beginning"
5,     since_db_path => "/dev/null"
6,   }
7, }
8, filter {
9,   grok {
10,     match => { "message" =>
11,       '{"level": "\u001b[%"NUMBER:color_code}minfo\u001b[%"NUMBER:end_code]m", "message": {"method": "%{WORD:method}", "url": "%{DATA:url}", "status": "%{NUMBER:status}", "responseTime": "%{NUMBER:response_time}"}, "timestamp": "%{TIMESTAMP_ISO8601:timestamp}"}
12,     }
13,   }
14,
15,
16,   mutate {
17,     remove_field => ["color_code", "end_code"]
18,   }
19,
20,
21,   mutate {
22,     convert => {
23,       "response_time" => "float"
24,       "status" => "integer"
25,     }
26,   }
27, }
28, output {
29,   elasticsearch {
30,     hosts => ["http://elasticsearch:9200"]
31,     index => "logstash_indexing" # Daily index
32,   }
33,   stdout {
34,     codec => rubydebug
35,   }
36, }
37, }

```

Explanation of logstash.conf:

This `logstash.conf` file defines how Logstash ingests, processes, and outputs log data. It reads log files from the specified path (`/app/app.log`), processes the log content using the **Grok filter** to parse structured information (like method, URL, status, and response time), and converts certain fields to appropriate data types (e.g., `response_time` to float, `status` to integer). The processed logs are sent to an Elasticsearch instance under the index `logstash_indexing` for storage and analysis. Additionally, debug logs are output to the console in a human-readable format using the `rubydebug` codec.

The **Grok filter** in the `logstash.conf` file is used to extract and structure log data based on a specific pattern. The pattern matches the log message's format, breaking it into meaningful fields such as `method`, `url`, `status`, `response_time`, and `timestamp`.

In your configuration, the Grok pattern decodes ANSI color codes and JSON-like content to extract structured data. For example:

- `method` captures the HTTP method (e.g., GET, POST).

- **url** captures the endpoint being accessed.
- **status** captures the HTTP response status code.
- **response_time** captures the response time of the request.
- **timestamp** captures when the log entry was recorded.

This structured data is then processed and stored for easier querying and analysis in Elasticsearch.

3. Start the Containers:

- Run the following command in the directory containing **docker-compose.yaml**:
 - docker-compose up

```
ritik@ritik:~/Downloads/Job_Portal_K8s_Vault_Scan_Jenkins/Job Portal$ docker compose up
[+] Running 20/3
  : logstash [          ] Pulling
  : kibana [██████████] 174.8MB / 336MB  Pulling
  : elasticsearch [██████████]  259MB / 384.2MB Pulling
```

```
ritik@ritik:~/Downloads/Job_Portal_K8s_Vault_Scan_Jenkins/Job Portal$ docker compose up
[+] Running 35/3
✓ logstash Pulled
✓ kibana Pulled
✓ elasticsearch Pulled
[+] Running 6/6
✓ Volume "jobportal_elasticsearch-data"  Created
✓ Container elasticsearch             Created   301.1s
✓ Container myapp-react-client       Created   189.1s
✓ Container logstash                Created   175.1s
✓ Container kibana                  Created   0.1s
✓ Container myapp-node-server      Created   4.4s
Attaching to elasticsearch, kibana, logstash, myapp-node-server, myapp-react-client
```

```
Attaching to elasticsearch, kibana, logstash, myapp-node-server, myapp-react-client
logstash | 2024/12/10 11:09:45 Setting 'xpack.monitoring.enabled' from environment.
logstash | Using bundled JDK: /usr/share/logstash/jdk
myapp-react-client | > frontend@0.0.0 dev
myapp-react-client | > vite --force --host 0.0.0.0
myapp-react-client | OpenJDK 64-Bit Server VM warning: Option UseConcMarkSweepGC was deprecated in version 9.0 and
will likely be removed in a future release.
myapp-node-server | > backend@1.0.0 dev
myapp-node-server | > nodemon index.js
myapp-node-server | Forced re-optimization of dependencies
myapp-react-client | VITE v5.4.11 ready in 995 ms
myapp-react-client | → Local: http://localhost:5173/
myapp-react-client | → Network: http://172.20.0.2:5173/
myapp-node-server | [nodemon] 3.1.3
myapp-node-server | [nodemon] to restart at any time, enter `rs`
myapp-node-server | [nodemon] watching path(s): ***!
myapp-node-server | [nodemon] watching extensions: js,mjs,cjs,json
myapp-node-server | [nodemon] starting `node index.js`
myapp-node-server | Servers running at port 8000
myapp-node-server | mongodb connected successfully
elasticsearch | {"type": "server", "timestamp": "2024-12-10T11:09:57,988Z", "level": "INFO", "component": "o.e.n.Node", "cluster.name": "docker-cluster", "node.name": "dbd66cc5e66c", "message": "version[7.17.10], pid[7], build [default/docker/fecd68e3150eda0c307ab9a9d7557f5d5fd71349/2023-04-23T05:33:18.1382755972], OS[Linux/6.8.0-49-generic/jamd64], JVM[Oracle Corporation/OpenJDK 64-Bit Server VM/20.0.1/20.0.1+9-29]" }
```

4. Access the Services:

- **Elasticsearch:** Access at <http://localhost:9200>.
- **Kibana:** Access at <http://localhost:5601>.

The screenshot shows the Elastic Stack home page. At the top, there's a banner asking for user feedback on the stack. Below it, the main heading is "Welcome home". There are four cards representing different services:

- Enterprise Search:** Create search experiences with a refined set of APIs and tools.
- Observability:** Consolidate your logs, metrics, application traces, and system availability with purpose-built UIs.
- Security:** Prevent, collect, detect, and respond to threats for unified protection across your infrastructure.
- Analytics:** Explore, visualize, and analyze your data using a powerful suite of analytical tools and applications.

Below these cards, there's a section titled "Get started by adding integrations" with three buttons: "Add Integrations", "Try sample data", and "Upload a file". To the right of this text is a colorful illustration of various data visualization components like charts and graphs.

5. Verify Logs in Kibana:

- Open Kibana and navigate to the **Discover** section.
- Configure an index pattern matching Elasticsearch indices (e.g., **logstash-***).
- View logs from your application in Kibana.

The screenshot shows the Kibana Settings page for the "logstash_indexing" index pattern. At the top, it says "Time field: '@timestamp'". Below that, it says "View and edit fields in **logstash_indexing**. Field attributes, such as type and searchability, are based on [field mappings](#) in Elasticsearch." There are tabs for "Fields (16)", "Scripted fields (0)", and "Field filters (0)".

Name	Type	Format	Searchable	Aggregatable	Excluded
@timestamp	date		●	●	<input type="button" value="Edit"/>
@version	text		●		<input type="button" value="Edit"/>
@version.keyword	keyword		●	●	<input type="button" value="Edit"/>
_id	_id		●	●	<input type="button" value="Edit"/>
_index	_index		●	●	<input type="button" value="Edit"/>
_score					
_source	_source				
_type	_type		●	●	
host	text		●		
host.keyword	keyword		●	●	

A modal window is open in the bottom right corner with the title "⚠ Your data is not secure". It says "Don't lose one bit. Enable our free security features." with a checkbox "Don't show again" and buttons "Enable security" and "Dismiss".

logstash_indexing

X

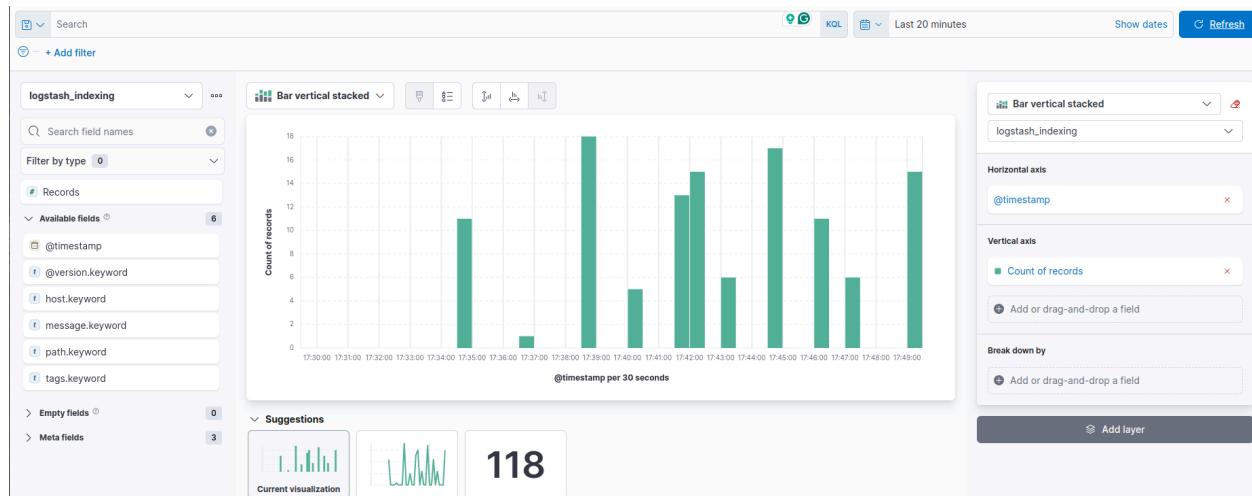
[Summary](#) [Settings](#) [Mappings](#) [Stats](#) [Edit settings](#)

General

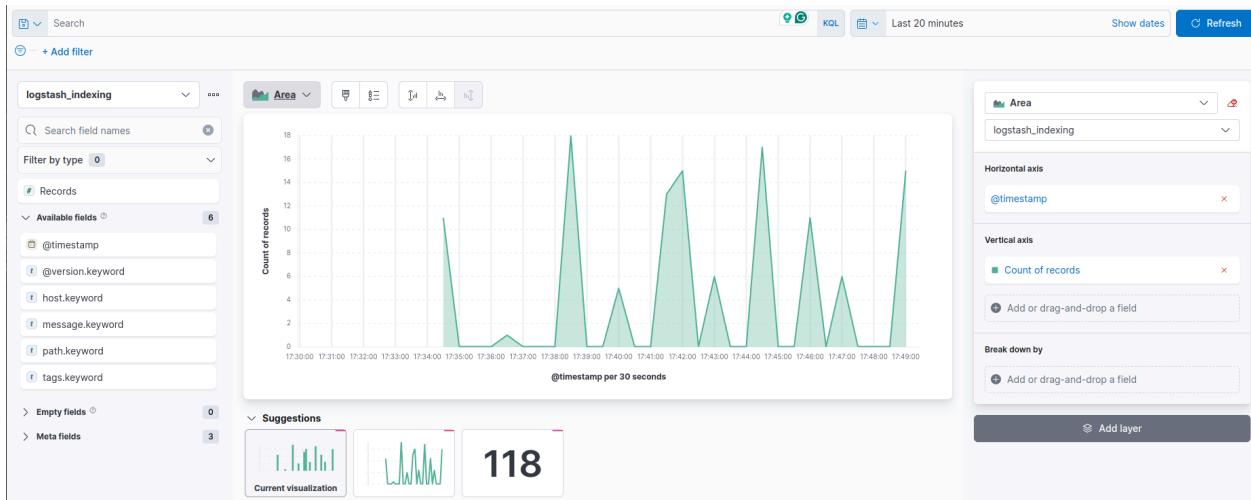
Health	● yellow	Status	open
Primaries	1	Replicas	1
Docs count	133	Docs deleted	
Storage size	114.5kb	Primary storage size	
Aliases	none		

Manage

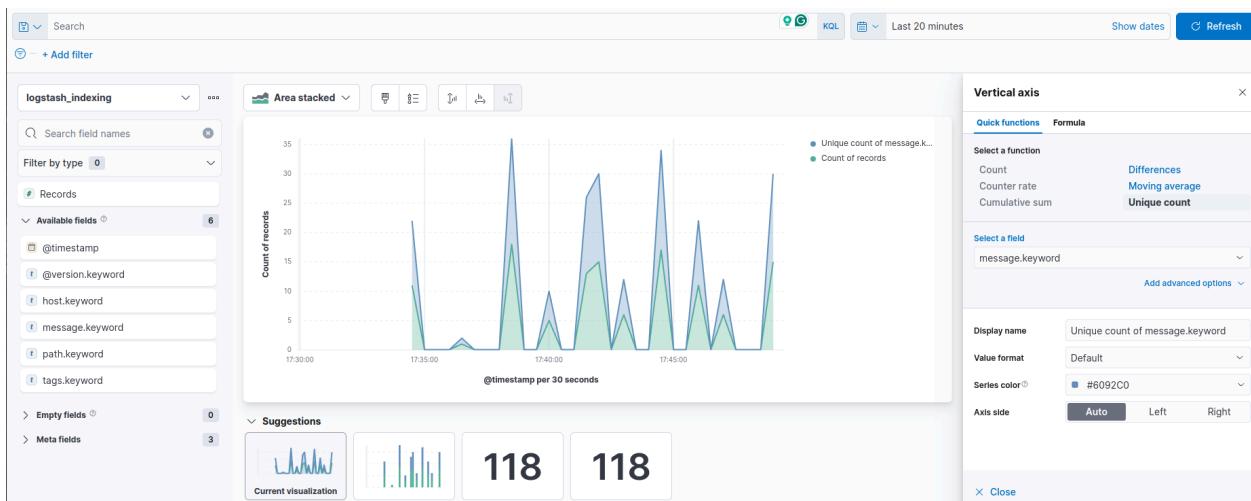
Visualization - 1: Bar Vertical Stacked



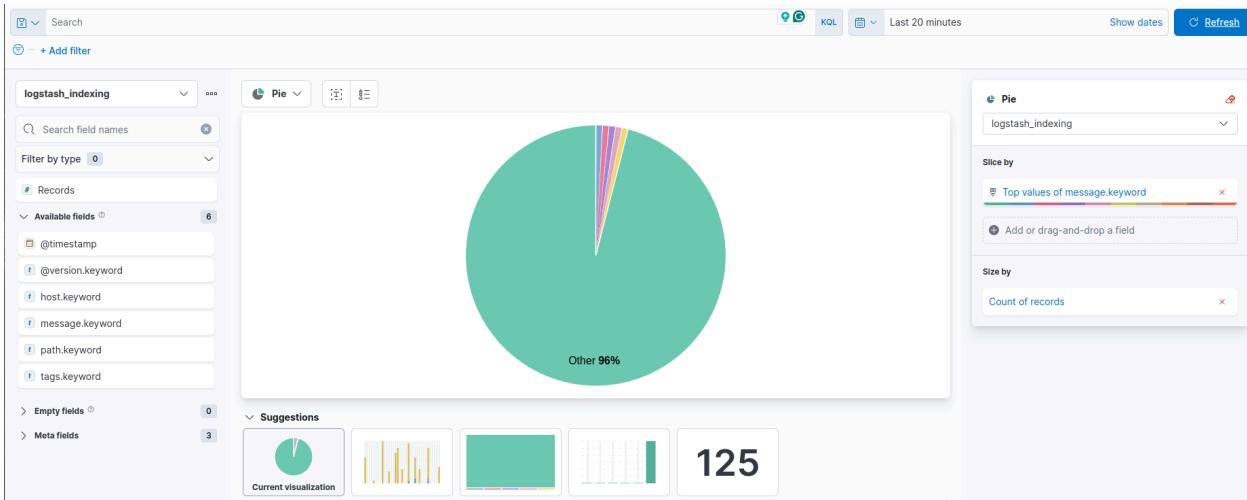
Visualization - 2: Area Plot



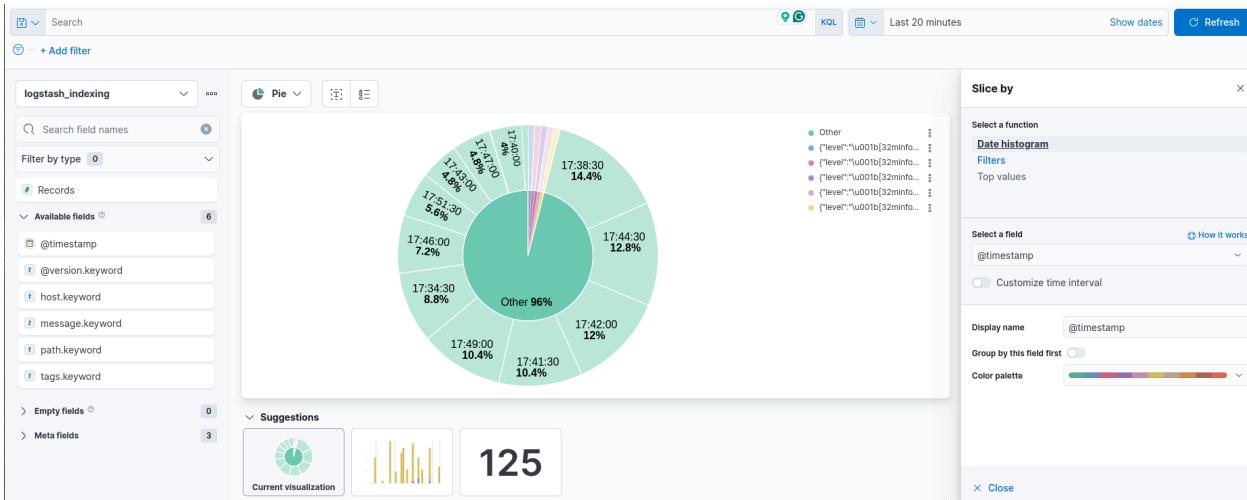
Visualization - 3: Area Stacked Plot



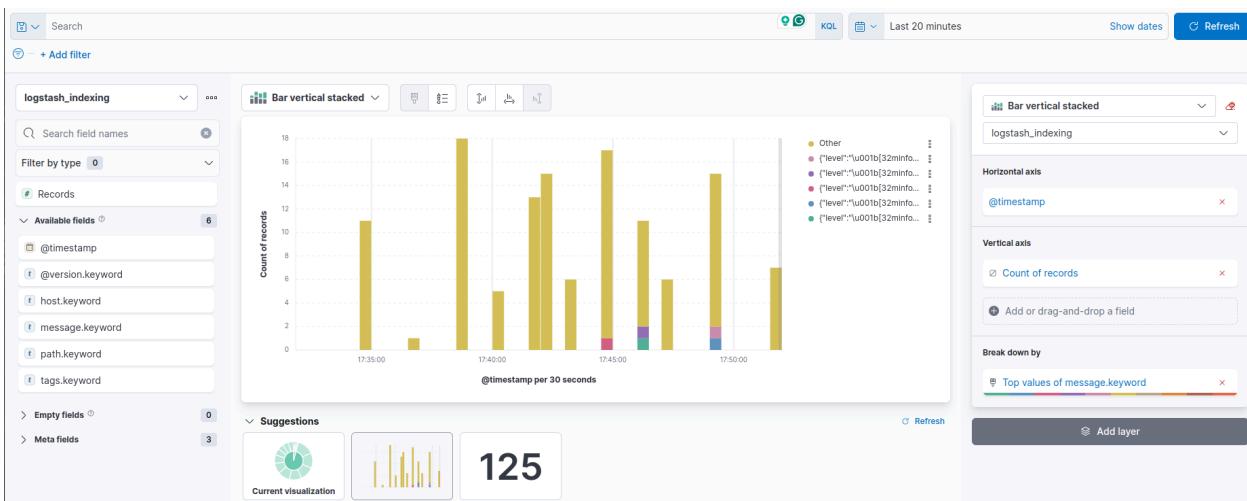
Visualization - 4: Pie Chart

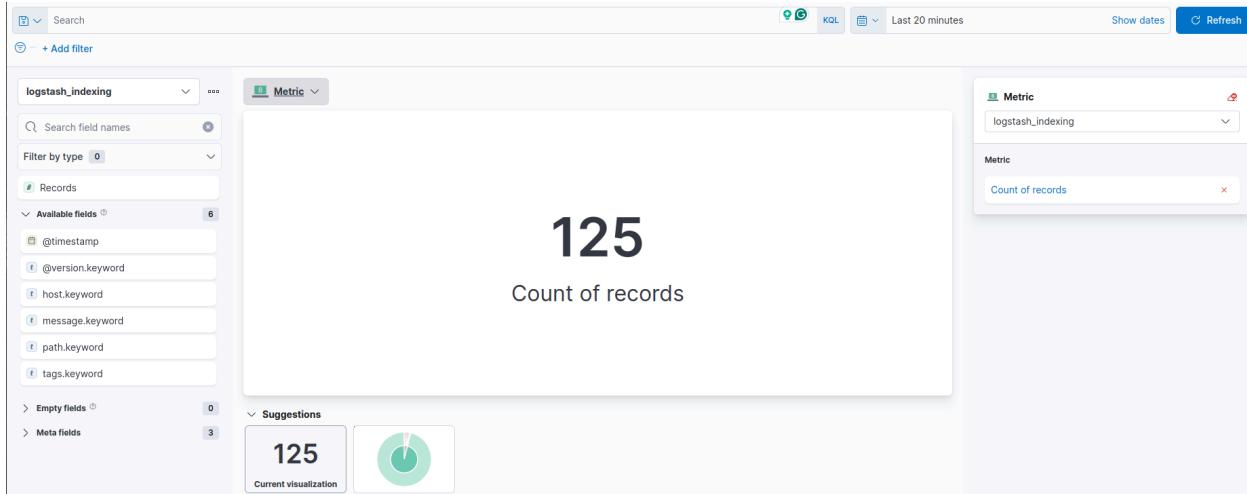


Visualization - 5: Pie Chart - 2

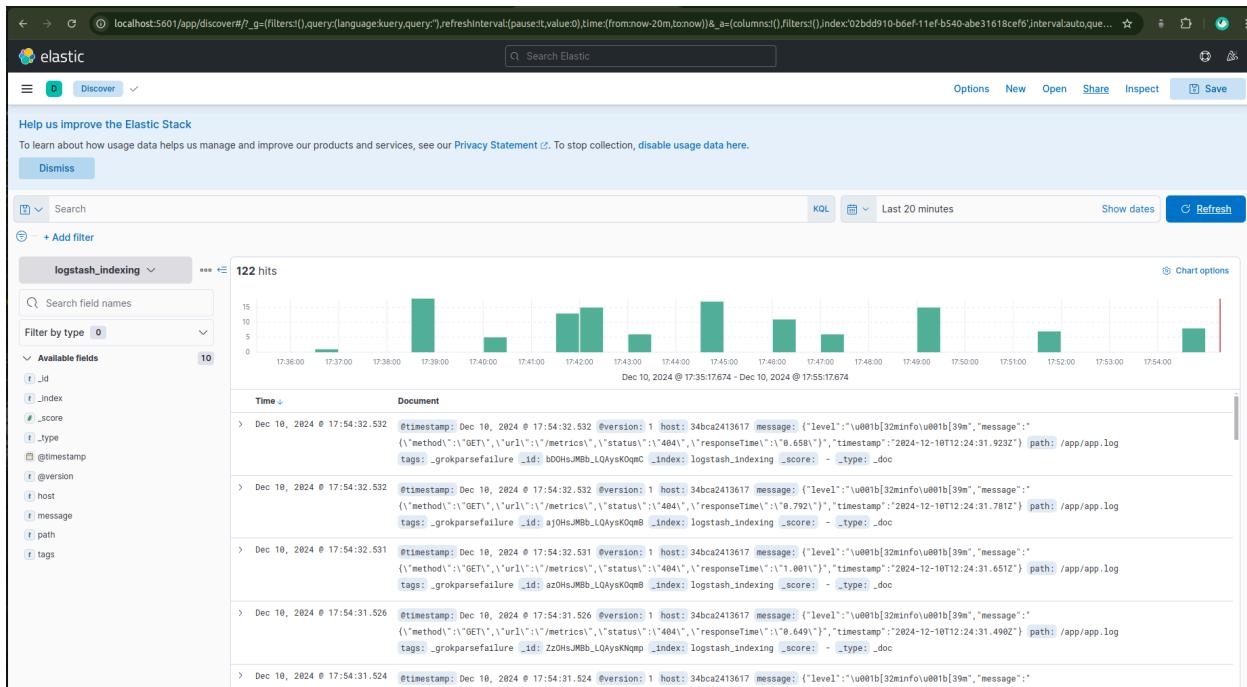


Visualization - 6: Bar Vertical Stacked





Kibana Dashboard



```

Job Portal > app.log
1  {"method": "\GET", "url": "\/", "status": "404", "responseTime": "\11.155\\"", "timestamp": "2024-12-10T11:42:06.604Z"}
2  {"method": "\GET", "url": "\/", "status": "404", "responseTime": "\12.861\\"", "timestamp": "2024-12-10T11:53:33.978Z"}
3  {"method": "\GET", "url": "\/", "status": "404", "responseTime": "\3.788\\"", "timestamp": "2024-12-10T11:54:38.465Z"}
4  {"method": "\GET", "url": "\/", "status": "404", "responseTime": "\1.028\\"", "timestamp": "2024-12-10T11:55:00.068Z"}
5  {"method": "\GET", "url": "\/", "status": "404", "responseTime": "\2.967\\"", "timestamp": "2024-12-10T11:55:00.888Z"}
6  {"method": "\GET", "url": "\/", "status": "404", "responseTime": "\1.415\\"", "timestamp": "2024-12-10T11:56:40.410Z"}
7  {"method": "\GET", "url": "\/", "status": "404", "responseTime": "\0.741\\"", "timestamp": "2024-12-10T11:56:41.128Z"}
8  {"method": "\GET", "url": "\/", "status": "404", "responseTime": "\19.241\\"", "timestamp": "2024-12-10T11:59:55.377Z"}
9  {"method": "\GET", "url": "\/", "status": "404", "responseTime": "\1.996\\"", "timestamp": "2024-12-10T11:59:56.484Z"}
10 {"method": "\GET", "url": "\/", "status": "404", "responseTime": "\0.785\\"", "timestamp": "2024-12-10T12:00:21.032Z"}
11 {"method": "\GET", "url": "\/", "status": "404", "responseTime": "\0.750\\"", "timestamp": "2024-12-10T12:00:22.028Z"}
12 {"method": "\GET", "url": "\/", "status": "404", "responseTime": "\12.501\\"", "timestamp": "2024-12-10T12:06:53.846Z"}
13 {"method": "\GET", "url": "\/", "status": "404", "responseTime": "\1.714\\"", "timestamp": "2024-12-10T12:08:35.209Z"}
14 {"method": "\GET", "url": "\/", "status": "404", "responseTime": "\1.034\\"", "timestamp": "2024-12-10T12:08:35.362Z"}
15 {"method": "\GET", "url": "\/", "status": "404", "responseTime": "\1.038\\"", "timestamp": "2024-12-10T12:08:35.561Z"}
16 {"method": "\GET", "url": "\/", "status": "404", "responseTime": "\0.923\\"", "timestamp": "2024-12-10T12:08:35.690Z"}
17 {"method": "\GET", "url": "\/", "status": "404", "responseTime": "\0.859\\"", "timestamp": "2024-12-10T12:08:35.835Z"}
18 {"method": "\GET", "url": "\/", "status": "404", "responseTime": "\1.310\\"", "timestamp": "2024-12-10T12:08:35.994Z"}
19 {"method": "\GET", "url": "\/", "status": "404", "responseTime": "\0.671\\"", "timestamp": "2024-12-10T12:08:36.125Z"}
20 {"method": "\GET", "url": "\/", "status": "404", "responseTime": "\1.949\\"", "timestamp": "2024-12-10T12:08:36.253Z"}
21 {"method": "\GET", "url": "\/", "status": "404", "responseTime": "\1.514\\"", "timestamp": "2024-12-10T12:08:36.382Z"}
22 {"method": "\GET", "url": "\/", "status": "404", "responseTime": "\0.856\\"", "timestamp": "2024-12-10T12:08:36.517Z"}
23 {"method": "\GET", "url": "\/", "status": "404", "responseTime": "\0.880\\"", "timestamp": "2024-12-10T12:08:36.648Z"}
24 {"method": "\GET", "url": "\/", "status": "404", "responseTime": "\1.630\\"", "timestamp": "2024-12-10T12:08:36.797Z"}
25 {"method": "\GET", "url": "\/", "status": "404", "responseTime": "\0.960\\"", "timestamp": "2024-12-10T12:08:36.956Z"}
26 {"method": "\GET", "url": "\/", "status": "404", "responseTime": "\0.845\\"", "timestamp": "2024-12-10T12:08:37.088Z"}
27 {"method": "\GET", "url": "\/", "status": "404", "responseTime": "\1.222\\"", "timestamp": "2024-12-10T12:08:37.238Z"}
28 {"method": "\GET", "url": "\/", "status": "404", "responseTime": "\0.872\\"", "timestamp": "2024-12-10T12:08:37.365Z"}
29 {"method": "\GET", "url": "\/", "status": "404", "responseTime": "\1.129\\"", "timestamp": "2024-12-10T12:08:37.495Z"}
30 {"method": "\GET", "url": "\/", "status": "404", "responseTime": "\0.781\\"", "timestamp": "2024-12-10T12:08:37.651Z"}
31 {"method": "\GET", "url": "\/", "status": "404", "responseTime": "\0.801\\"", "timestamp": "2024-12-10T12:10:12.344Z"}
32 {"method": "\GET", "url": "\/", "status": "404", "responseTime": "\0.961\\"", "timestamp": "2024-12-10T12:10:12.497Z"}
33 {"method": "\GET", "url": "\/", "status": "404", "responseTime": "\0.667\\"", "timestamp": "2024-12-10T12:10:12.618Z"}
34 {"method": "\GET", "url": "\/", "status": "404", "responseTime": "\1.272\\"", "timestamp": "2024-12-10T12:10:12.777Z"}
35 {"method": "\GET", "url": "\/", "status": "404", "responseTime": "\0.651\\"", "timestamp": "2024-12-10T12:10:12.887Z"}
36 {"method": "\GET", "url": "\/", "status": "404", "responseTime": "\0.817\\"", "timestamp": "2024-12-10T12:11:35.795Z"}
37 {"method": "\GET", "url": "\/", "status": "404", "responseTime": "\1.068\\"", "timestamp": "2024-12-10T12:11:36.001Z"}
38 {"method": "\GET", "url": "\/", "status": "404", "responseTime": "\1.386\\"", "timestamp": "2024-12-10T12:11:36.156Z"}
39 {"method": "\GET", "url": "\/", "status": "404", "responseTime": "\0.710\\"", "timestamp": "2024-12-10T12:11:36.201Z"}

```

Kubernetes: Prometheus and Grafana for Monitoring and Logging

Prometheus and Grafana are commonly used together for robust monitoring and visualization in Kubernetes environments.

Setup Steps:

- Install Helm:** Helm is a package manager for Kubernetes. Install Helm using the following command:
 - `curl https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3 | bash`
- Pull Loki Docker Image:** Loki is a log aggregation tool that integrates seamlessly with Grafana. Pull the Loki image:
 - `docker pull grafana/loki:2.9.3`
- Pull Promtail Docker Image:** Promtail is an agent used to collect logs and forward them to Loki. Pull the Promtail image:
 - `docker pull grafana/promtail:2.9.3`

4. **Pull Grafana Docker Image:** Grafana provides visualizations and dashboards. Pull the latest Grafana image:
 - a. `docker pull grafana/grafana:latest`
5. **Add Grafana Helm Repository:**
 - a. `helm repo add grafana https://grafana.github.io/helm-charts`
6. **Install Loki Stack Using Helm:** Deploy Loki, Promtail, Prometheus, and Grafana together in the `monitoring` namespace. Customize the configuration using the Helm command:
 - a. `helm install loki grafana/loki-stack \ --namespace monitoring \ --create-namespace \ --set loki.image.tag=2.9.3 \ --set promtail.enabled=true \ --set promtail.config.server.http_listen_port=3101 \ --set promtail.config.clients[0].url=http://loki:3100/loki/api/v1/push \ --set promtail.config.positions.filename=/run/promtail/positions.yaml \ --set promtail.config.scrape_configs[0].job_name="kubernetes-pods" \ --set promtail.config.scrape_configs[0].kubernetes_sd_configs[0].role="pod" \ --set promtail.config.scrape_configs[0].relabel_configs[0].action="keep" \ --set promtail.config.scrape_configs[0].relabel_configs[0].source_labels="[_meta_kubernetes_namespace]" \ --set promtail.config.scrape_configs[0].relabel_configs[0].regex=".*" \ --set promtail.config.scrape_configs[0].relabel_configs[1].source_labels="[meta_kubernetes_pod_name]" \ --set promtail.config.scrape_configs[0].relabel_configs[1].target_label="job" \ --set promtail.config.scrape_configs[0].relabel_configs[2].source_labels="[meta_kubernetes_namespace]" \ --set promtail.config.scrape_configs[0].relabel_configs[2].target_label="namespace" \ --set promtail.config.scrape_configs[0].relabel_configs[3].source_labels="[_meta_kubernetes_pod_name]" \ --set promtail.config.scrape_configs[0].relabel_configs[3].target_label="pod" \ --set grafana.enabled=true \ --set prometheus.enabled=true`
7. **Port-Forward Prometheus, Grafana, and Loki:**
 - a. `kubectl port-forward --namespace monitoring svc/loki-prometheus-server 9090:80`

- b. kubectl port-forward --namespace monitoring service/loki-grafana 3000:80
- c. kubectl port-forward svc/loki -n monitoring 3100:3100

8. Retrieve Grafana Admin Password:

- a. kubectl get secret loki-grafana -n monitoring -o jsonpath="{.data.admin-password}" | base64 --decode

```
ritik@ritik:~/Downloads/Talent_Bridge_Kubernetes$ kubectl get all -n monitoring
NAME                                         READY   STATUS    RESTARTS   AGE
pod/loki-0                                     1/1     Running   0          4m10s
pod/loki-alertmanager-0                         1/1     Running   0          4m10s
pod/loki-grafana-847987cd7-h8qp2                2/2     Running   0          4m11s
pod/loki-kube-state-metrics-85d4bdcbc5-5qkl2   1/1     Running   0          4m11s
pod/loki-prometheus-node-exporter-5wvvd        1/1     Running   0          4m11s
pod/loki-prometheus-pushgateway-799988c99-dn69m  1/1     Running   0          4m11s
pod/loki-prometheus-server-598f8bc7d5-bgnpm     2/2     Running   0          4m11s
pod/loki-promtail-5gl2k                         1/1     Running   0          4m11s

NAME                           TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)      AGE
service/loki                   ClusterIP   10.99.136.107 <none>        3100/TCP    4m11s
service/loki-alertmanager      ClusterIP   10.99.109.79  <none>        9093/TCP    4m11s
service/loki-alertmanager-headless  ClusterIP   None           <none>        9093/TCP    4m11s
service/loki-grafana           ClusterIP   10.100.210.95 <none>        80/TCP      4m11s
service/loki-headless          ClusterIP   None           <none>        3100/TCP    4m11s
service/loki-kube-state-metrics ClusterIP   10.103.144.25 <none>        8080/TCP    4m11s
service/loki-memberlist         ClusterIP   None           <none>        7946/TCP    4m11s
service/loki-prometheus-node-exporter ClusterIP   10.98.140.106 <none>        9100/TCP    4m11s
service/loki-prometheus-pushgateway  ClusterIP   10.109.100.71 <none>        9091/TCP    4m11s
service/loki-prometheus-server  ClusterIP   10.100.128.143 <none>        80/TCP      4m11s
```

```
ritik@ritik:~/Downloads/Talent_Bridge_Kubernetes$ kubectl port-forward --namespace monitoring svc/loki-prometheus-server 9090:80
Forwarding from 127.0.0.1:9090 -> 9090
Forwarding from [::1]:9090 -> 9090
```

```
ritik@ritik:~/Downloads/Talent_Bridge_Kubernetes$ kubectl port-forward --namespace monitoring service/loki-grafana 3001:80
Forwarding from 127.0.0.1:3001 -> 3000
Forwarding from [::1]:3001 -> 3000
```

```
ritik@ritik:~/Downloads/Talent_Bridge_Kubernetes$ kubectl port-forward svc/loki -n monitoring 3100:3100
Forwarding from 127.0.0.1:3100 -> 3100
Forwarding from [::1]:3100 -> 3100
```

```
ritik@ritik:~/Downloads/Talent_Bridge_Kubernetes$ kubectl get secret loki-grafana -n monitoring -o jsonpath="{.data.admin-password}" | base64 --decode
xFOFagYJiaJ6gZqC7yzbrfjgzhrLm22I028EGs2dritik@ritik:~/Downloads/Talent_Bridge_Kubernetes$
```

localhost:9090/targets?search=

Prometheus Alerts Graph Status Help

Targets

All Unhealthy Expand All Filter by endpoint or labels

kubernetes-apiservers (1/1 up)		
Endpoint	State	Labels
https://192.168.49.2:8443/metrics	UP	instance="192.168.49.2:8443" job="kubernetes-apiservers"

kubernetes-nodes (1/1 up)		
Endpoint	State	Labels
https://kubernetes.default.svc/api/v1/nodes/minikube/proxy/metrics	UP	beta.kubernetes.io_arch="amd64" beta.kubernetes.io_os="linux" instance="minikube" job=kubernetes-nodes kubernetes.io_arch="amd64" kubernetes.io_hostname="minikube" kubernetes.io_os="linux" minikube_k8s_io_commit="210b148df92a0deb972ebeb7e35281b3c582cd1" minikube_k8s_io_name="minikube" minikube_k8s_io_primary=true minikube_k8s_io_updated_at="2024-12-10T16:05:10.9700" minikube_k8s_io_version="v1.34.0"

kubernetes-nodes-cadvisor (1/1 up)		
Endpoint	State	Labels

localhost:3001/login



Welcome to Grafana

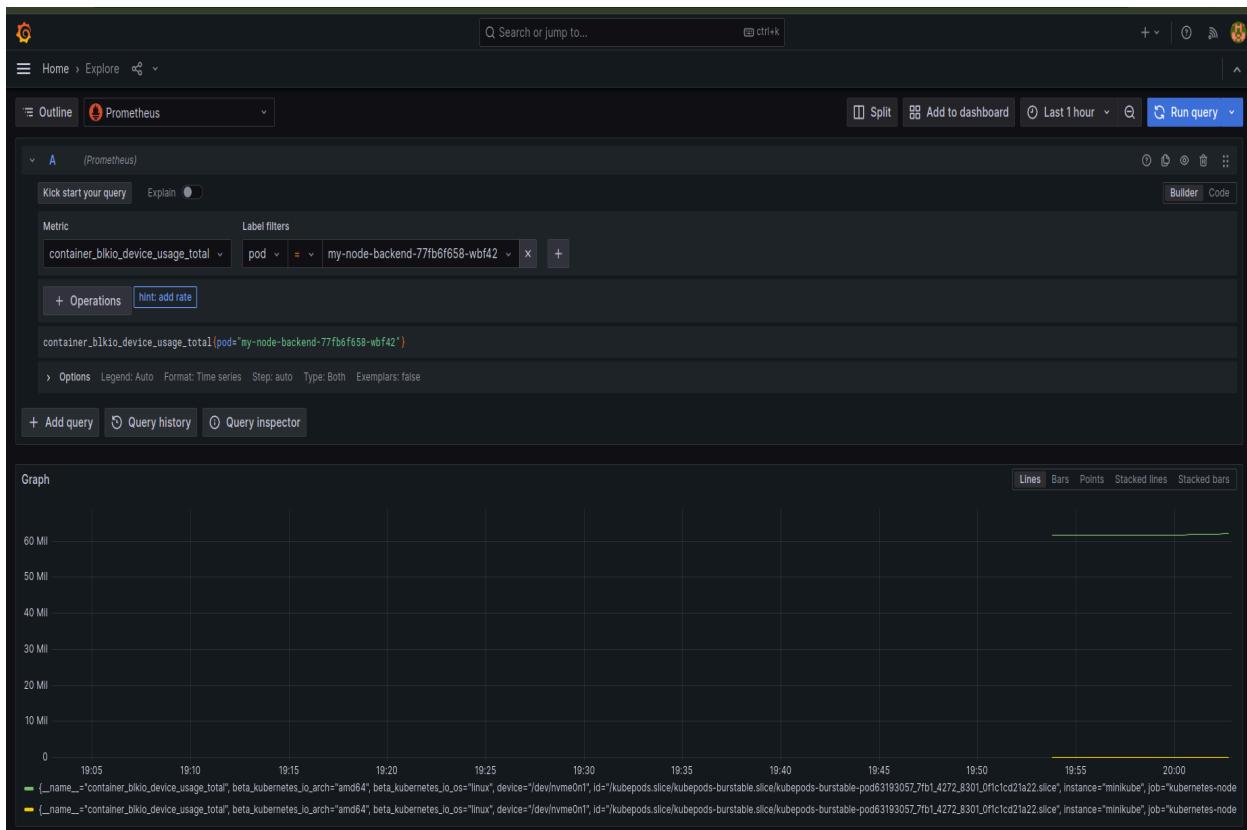
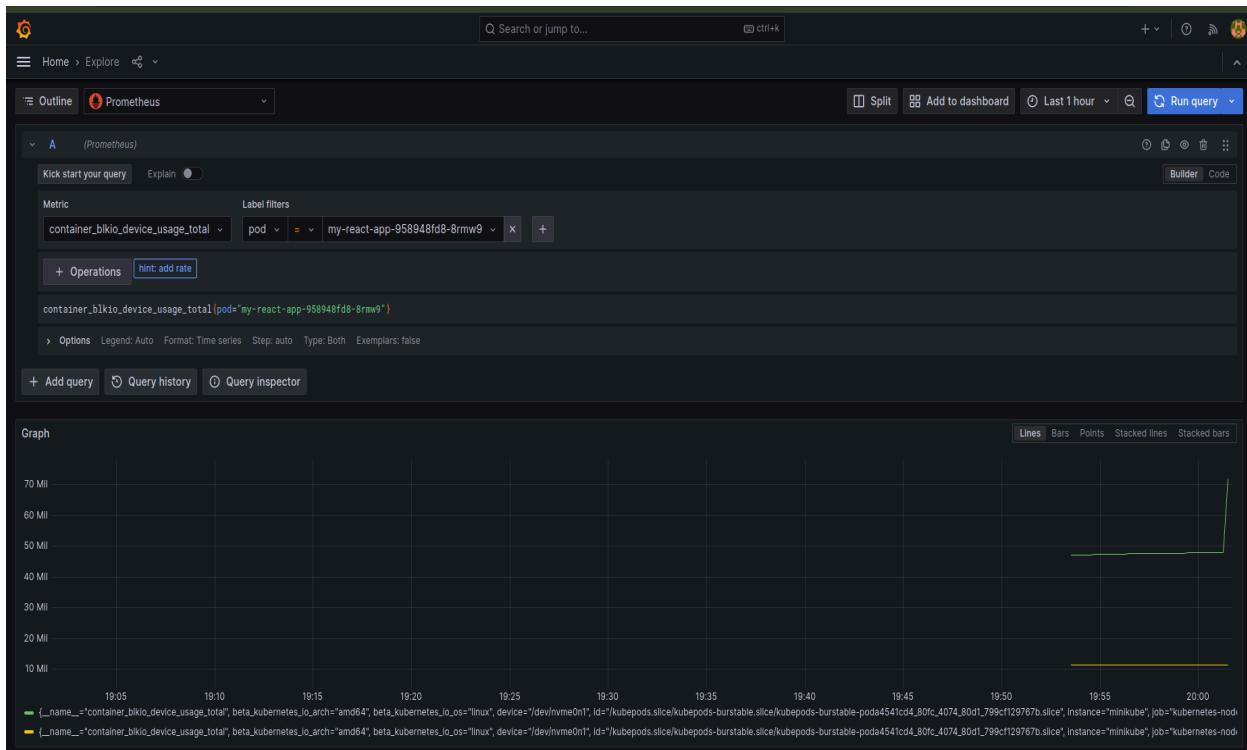
Email or username
admin

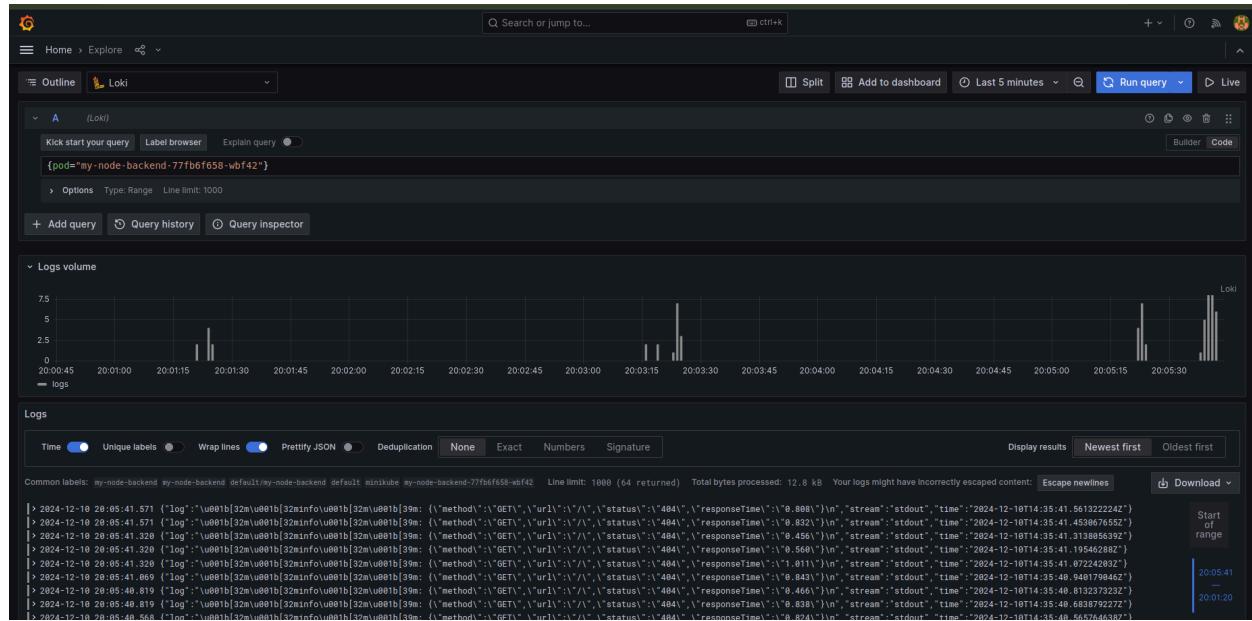
Password
.....

Log in

Forgot your password?

```
ritik@ritik:~/Downloads/Talent_Bridge_Kubernetes$ kubectl get pods
NAME                               READY   STATUS    RESTARTS   AGE
my-node-backend-77fb6f658-wbf42   1/1     Running   0          14m
my-react-app-958948fd8-8rmw9     1/1     Running   0          14m
```





Advanced Features:

1. Ansible Vault -

```
! my_vault.yml
1 $ANSIBLE_VAULT;1.1;AES256
2 62623064653532313362323235343066623461383633313632613331363165336466373064396639
3 6365663939646131656131663665643236353961313365610a373239396634636563353132326435
4 39393432626662386162346635613166396163666664303766653738356335313532623934623063
5 6661643535653362660a336265353132313663633066336231656161306665376166313233383930
6 6335
7
```

2. Modularity: Roles in Ansible -

```
✓ roles
  ✓ clone_repo/tasks
    ! main.yml
  ✓ deploy_backend
    > files
    > tasks
    > deploy_frontend
```

```
✓ roles
  ✓ deploy_docker_compose/tasks
    ! main.yaml
  ✓ docker_install/tasks
    ! main.yaml
  ✓ git_clone/tasks
    ! main.yaml
```

3. HPA -

```
ub > ! backend-hpa.yaml
1 apiVersion: autoscaling/v2
2 kind: HorizontalPodAutoscaler
3 metadata:
4   name: my-node-backend-hpa
5 spec:
6   scaleTargetRef:
7     apiVersion: apps/v1
8     kind: Deployment
9     name: my-node-backend
10    minReplicas: 1
11    maxReplicas: 5
12    metrics:
13      - type: Resource
14        resource:
15          name: cpu
16          target:
17            type: Utilization
18            averageUtilization: 1
19
```

```
ub > ! frontend-hpa.yaml
1 apiVersion: autoscaling/v2
2 kind: HorizontalPodAutoscaler
3 metadata:
4   name: my-react-app-hpa
5 spec:
6   scaleTargetRef:
7     apiVersion: apps/v1
8     kind: Deployment
9     name: my-react-app
10    minReplicas: 1
11    maxReplicas: 5
12    metrics:
13      - type: Resource
14        resource:
15          name: cpu
16          target:
17            type: Utilization
18            averageUtilization: 1
19
```

Challenges Faced

1. Setting Up Kubernetes on Jenkins

Integrating Kubernetes with Jenkins was challenging due to the complexity of managing Kubernetes configurations within the Jenkins pipeline.

Configuring Kubernetes credentials, ensuring compatibility between Jenkins agents and Kubernetes clusters, and setting up Minikube as the local Kubernetes environment required extensive troubleshooting.

Additionally, automating the deployment of Kubernetes resources via Helm charts and maintaining proper security for sensitive data, like Kubernetes secrets, was intricate and time-consuming.

2. ELK Stack Setup on Kubernetes

Deploying the ELK (Elasticsearch, Logstash, Kibana) stack on Kubernetes proved to be difficult due to resource constraints and the intricate nature of its configuration. Elasticsearch's memory requirements and Logstash's dependency on pipeline files made the setup prone to errors. Furthermore, ensuring smooth intercommunication between ELK components within a Kubernetes cluster was tedious. This challenge led us to shift to

Prometheus and Grafana, which offered a more lightweight and Kubernetes-friendly monitoring and logging solution. Their native support for Kubernetes metrics and easier setup proved to be a practical alternative.

3. Setting Up the SSH Server for Testing Ansible

Establishing an SSH server to test Ansible configurations was complex due to networking and authentication issues. Configuring SSH keys for secure, password-less authentication and ensuring compatibility across multiple environments required meticulous attention. Managing the server environment, ensuring it mimicked production systems, and dealing with connectivity issues while testing playbooks made this task especially challenging.

Future Scope

In the future, the Job Portal application can incorporate machine learning models to enhance user experience and platform reliability. ML algorithms can be used to **identify fake job postings** by analyzing patterns and anomalies in job descriptions, recruiter profiles, and posting behavior. Additionally, **personalized job recommendations** can be provided to users by leveraging their profiles, skills, and past interactions, improving job search efficiency and satisfaction.

References:

1. <https://kubernetes.io/docs/home/>
2. <https://docs.docker.com/>
3. <https://github.com/prometheus-community/helm-charts>
4. <https://prometheus.io/docs/prometheus/latest/installation/>
5. <https://trivy.dev/v0.18.3/installation/>