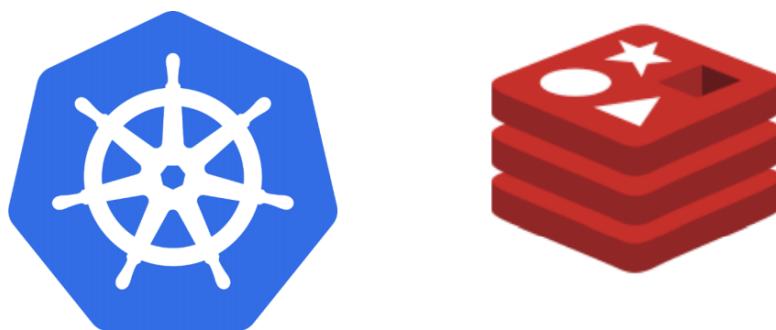


# DevOps Final Project



--Ravi Kumar Pilla  
--Priyam Mukund Modi  
--Dinakara Sai Santosh Burugupalli

## Introduction

### Application Features – Twitter Clone / Chatter

- Token based authentication for user Sign in and Sign up
- Web application where user can tweet and see the recent tweet from all other users.
- Integrated News API on the home page for latest updates
- Tweets include photo, username, tweet body, comments and likes on a particular post.
- User can comment and like the tweet and view tweets posted by others
- Ability to delete offensive tweets before any user can see them
- Tweets are stored in a database, so they never get lost
- Ability to deploy front-end as a PWA so that it can run on devices
- All the tweets are stored to the backend database, which will help the user find their past tweets.
- Redis-json as the middle layer.
- React and material UI for frontend development
- Session Based user management using cookies

### Application Deployment

- Terraform to provision infrastructure
- Kubernetes cluster on AWS
- Logging using ELK stack
- Monitoring using Prometheus
- Continuous Integration to Docker Hub (A new image is created and deployed to docker hub)
- Ingress Controller using NGINX
- Application scaling based on number of clients

## About various Technologies

### Frontend Technologies

- React: <https://react-cn.github.io/react/downloads.html>
- React Native: <https://reactnative.dev/>
- Command to create react native project- npx react-native init {Application Name}

### Backend Technologies

- Node Js: <https://nodejs.org/en/download/>
- Mongo DB: [MongoDB Download Link](#)
- Redis Server: <https://redis.io/download>

### Server Technologies

- Docker CLI: <https://docs.docker.com/docker-for-mac/install/>
- Kubernetes: <https://kubernetes.io/docs/tasks/tools/install-kubectl/>
- EKS: <https://minikube.sigs.k8s.io/docs/start/>
- Helm Chart: [https://helm.sh/docs/helm/helm\\_pull/](https://helm.sh/docs/helm/helm_pull/)
- AWS CLI: <https://awscli.amazonaws.com/AWSCLIV2.msi>
- Prometheus: <https://prometheus.io/download/>

## Final Project – Twitter Application

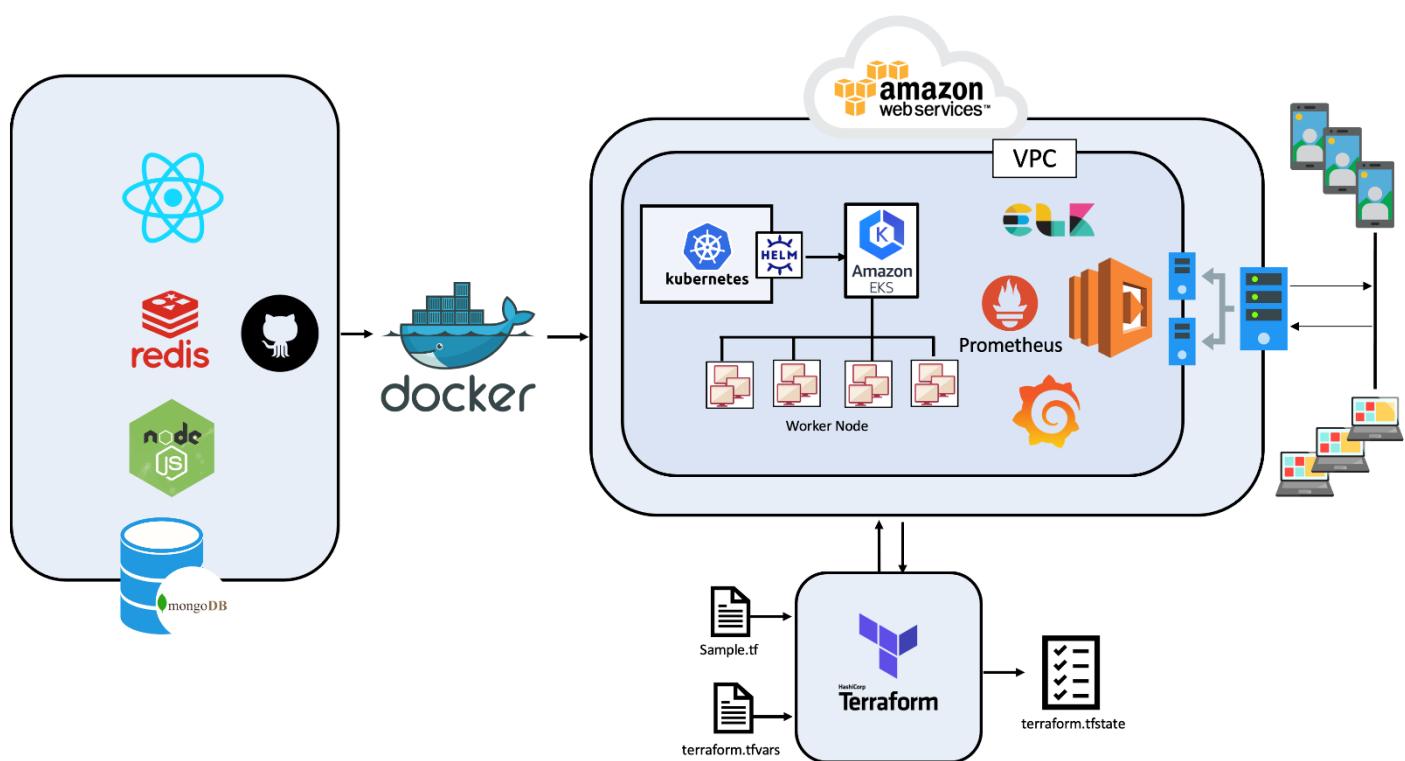
- Terraform: <https://www.terraform.io/downloads.html>

## Logging Technologies

- ELK Stack

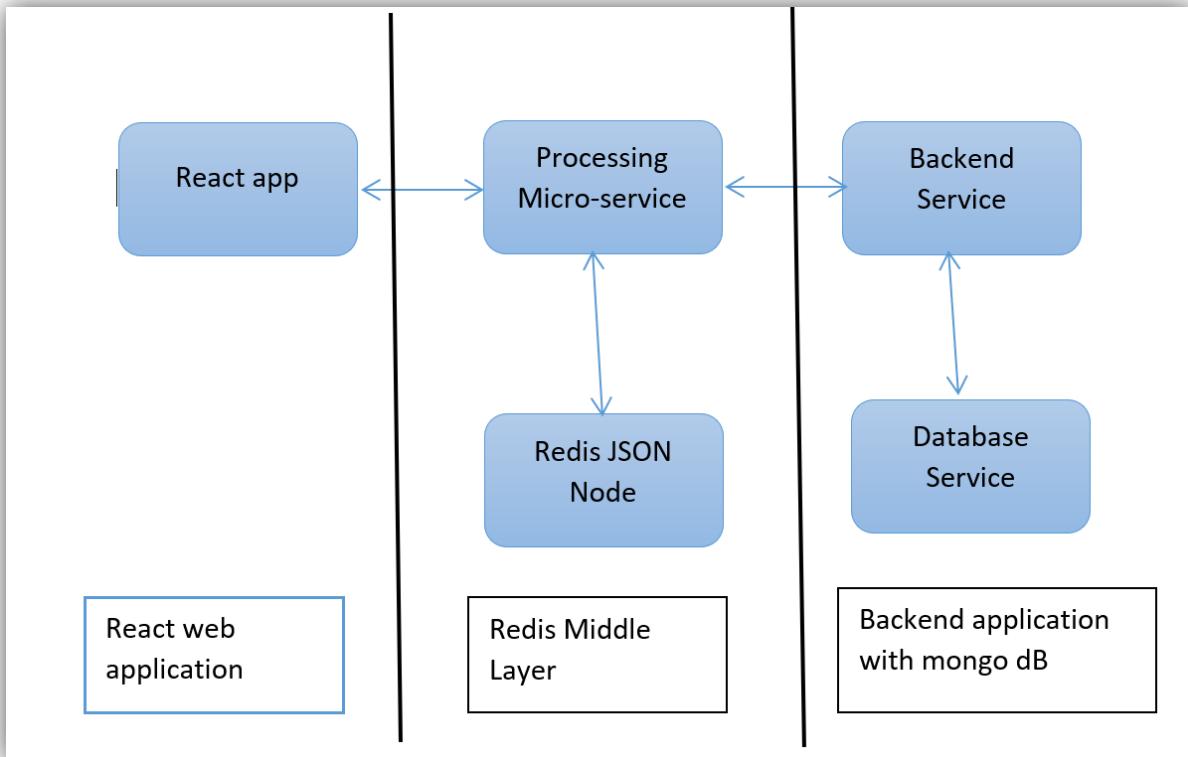
## Project Architecture

- The Front end is created in the React, Redis JSON is working as the middleware for the system.
- All the requests are clustered in the middleware and after every interval it is transferred to the backend i.e., Node server.
- The Node server relates to the MongoDB for saving and fetching user data.
- All the layers are separately containerized in the Docker Container.
- The Docker containers are further deployed as the Kubernetes pods.
- The Kubernetes pods are deployed on the AWS EKS using the Terraform.
- The Prometheus server is connected to the AWS to track down the request load on the server
- Metrics are displayed using Grafana dashboard
- All the incoming requests from the client are load balanced on the AWS using the ELB service of the AWS.
- Pod autoscaling based on number of client requests. And custom metrics using service monitor and Prometheus client.
- Continuous Integration using GitHub Operations and Docker
- Developed helm chart to deploy application at a faster pace



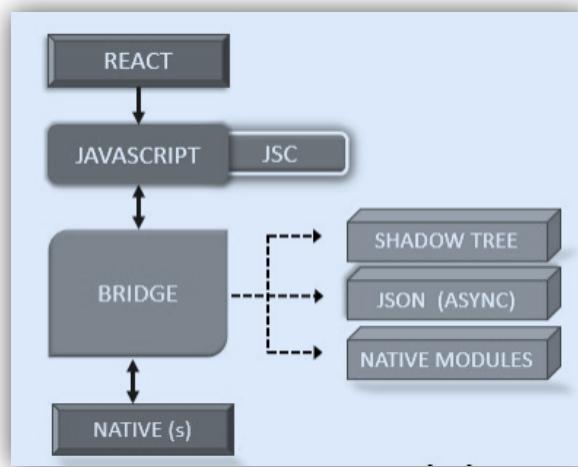
## Application architecture

## Final Project – Twitter Application



### 1. Frontend

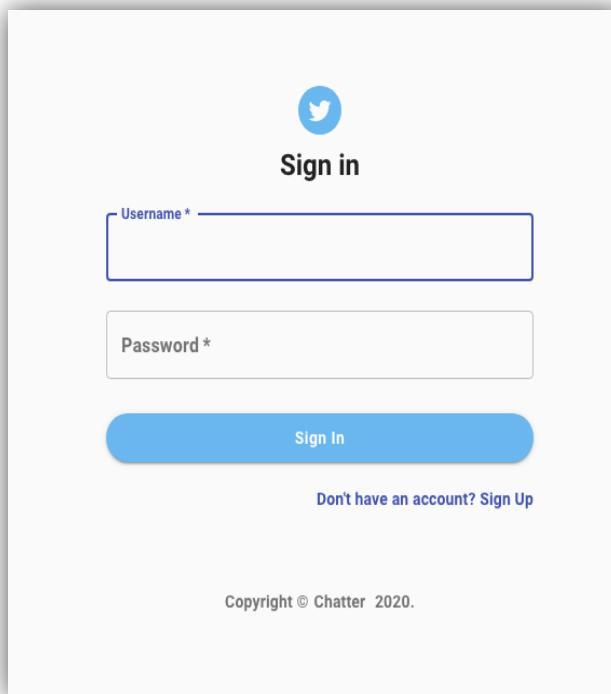
The frontend application is created in React, which enables the application to run on both Mobile and webapp.



# Final Project – Twitter Application

## Login Page

User can login to the existing account.



Copyright © Chatter 2020.

```
// SignIn function definition
export default function SignIn({ changeActiveRoute, history }) {
  const classes = useStyles();

  // State for username and password
  const [username, setUsername] = useState("");
  const [password, setPassword] = useState("");

  // State to show auth failed/succeeded
  const [isUserEntryValid, setIsUserEntryValid] = useState(true);

  // Authenticate a user using API endpoint
  const authenticateUser = async (e) => {
    try {
      setIsUserEntryValid(true);
      e.preventDefault();

      // post api call trigger to verify user identity
      const userRequest = {
        username,
        password,
      };

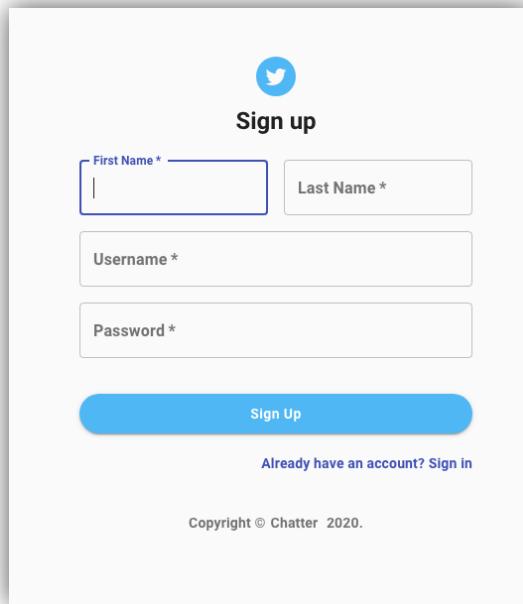
      const userInfoReceived = await authenticateUserInfo(userRequest);
      userInfoReceived.displayName = userInfoReceived.firstName
        .concat(" ")
        .concat(userInfoReceived.lastName);

      // Set session for user
      setSessionCookie(userInfoReceived);
      history.push("/home");
    } catch (error) {
      console.log(error);
      setIsUserEntryValid(false);
    }
  };
}
```

## Registration Page

Create a new user just by entering Name and Email ID.

Password field on the Login and Registration page is Token based authentication.



```
// Function to register a user using Register API
const registerUser = async (e) => {
  try {
    setIsUserEntryValid(true);
    e.preventDefault();

    // post api call trigger to register user
    const userRequest = {
      firstName: firstName,
      lastName: lastName,
      username: username,
      password: password,
    };

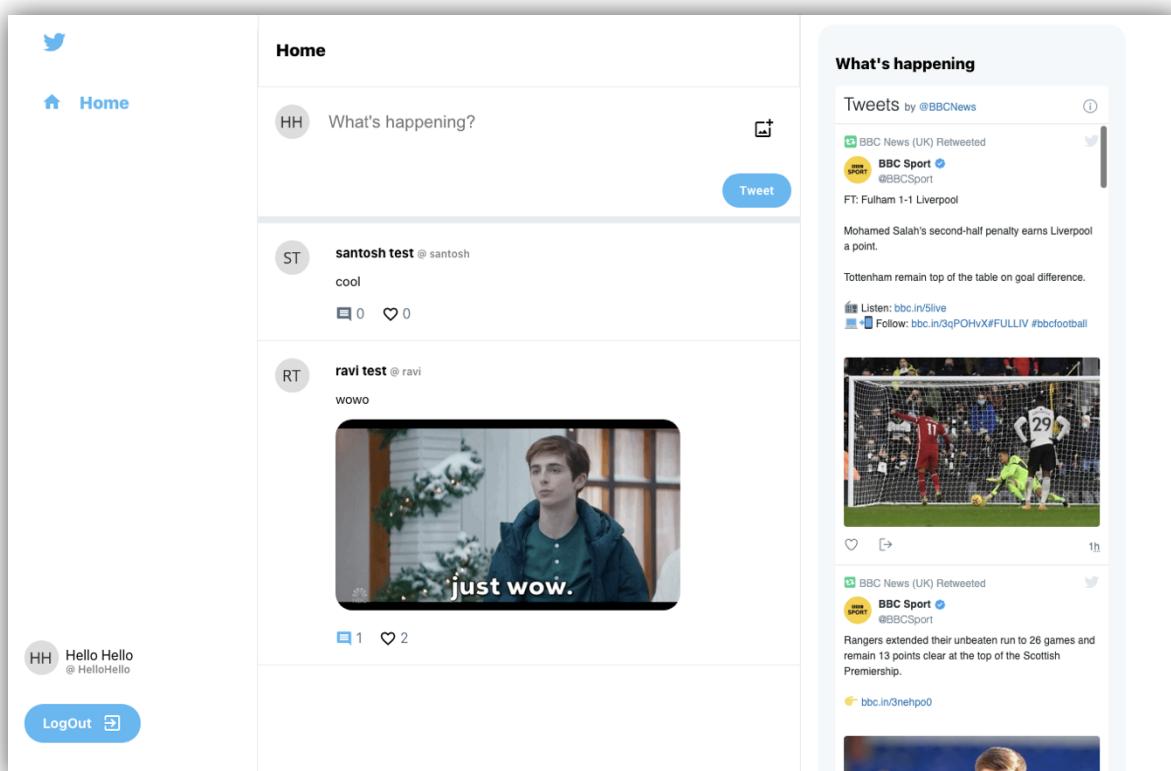
    // register API call
    await registerUserInfo(userRequest);
    // move to login screen
    changeActiveRoute();
  } catch (error) {
    console.log(error);
    setIsUserEntryValid(false);
  }
};
```

## Final Project – Twitter Application

```
▼ Request Payload    view source
  ▼ {firstName: "prudhvi", lastName: "test", username: "prudhvi", password: ".....3"}
    firstName: "prudhvi"
    lastName: "test"
    password: ".....3"
    username: "prudhvi"
```

Name	Headers	Preview	Response	Initiator	Timing
profile?callback=__twtr.callbacks.tl_0...					
profile?callback=__twtr.callbacks.tl_0...			1 {"message":"User Registered successfully"}		
profile?callback=__twtr.callbacks.tl_1...					

## Home Page

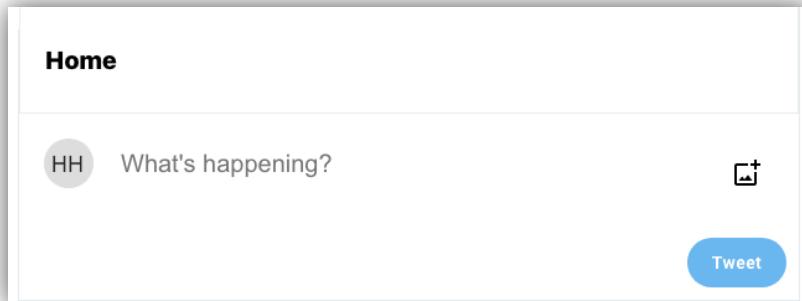


## Final Project – Twitter Application

```
21 lines (20 sloc) | 488 Bytes

1  /**
2   * Component Responsible to hold all the 3 major
3   * components Sidebar, Feed, Widgets
4   */
5  import React from "react";
6  import Feed from "../Feed/Feed";
7  import Sidebar from "../Sidebar/Sidebar";
8  import Widgets from "../Widgets/Widgets";
9  import "../../App.css";
10 // Home function definition
11 function Home({ userInfo }) {
12   return (
13     <div className="app">
14       <Sidebar userInfo={userInfo} />
15       <Feed userInfo={userInfo} />
16       <Widgets />
17     </div>
18   );
19 }
20
21 export default Home;
```

- User can Tweet what is happening around.



### Request for Creating Tweet

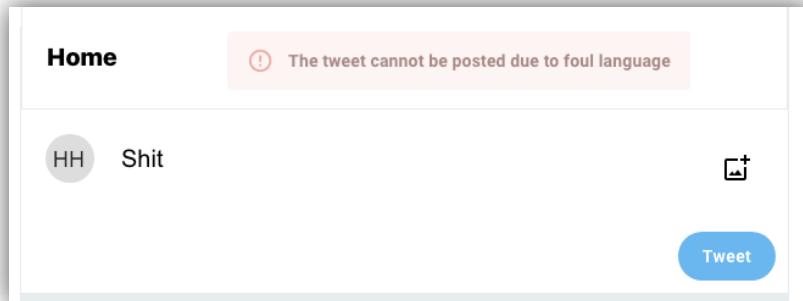
```
▼ Request Payload view source
  ▼ {tweetId: "c00b79e9-597f-4bfc-940f-b9d7476a5a4f", tweet: "creating a new tweet",...}
    comments: []
    createdAt: "2020-12-13T14:27:36-05:00"
    ▶ createdBy: {displayName: "priyam test", userName: "priyam", avatarLink: ""}
    imageLink: ""
    likes: []
    tweet: "creating a new tweet"
    tweetId: "c00b79e9-597f-4bfc-940f-b9d7476a5a4f"
```

## Final Project – Twitter Application

### Response for Creating Tweet

```
object {7}
  tweetId : c00b79e9-597f-4bfc-940f-b9d7476a5a4f
  tweet : creating a new tweet
  imageLink : [value]
  createdBy {3}
    displayName : priyam test
    userName : priyam
    avatarLink : [value]
  createdAt : 2020-12-13T14:27:36-05:00
  likes [0]
    (empty array)
  comments [0]
    (empty array)
```

- User will not be able to post abuse words.



- User can like any of the tweets and can also comment on tweets.

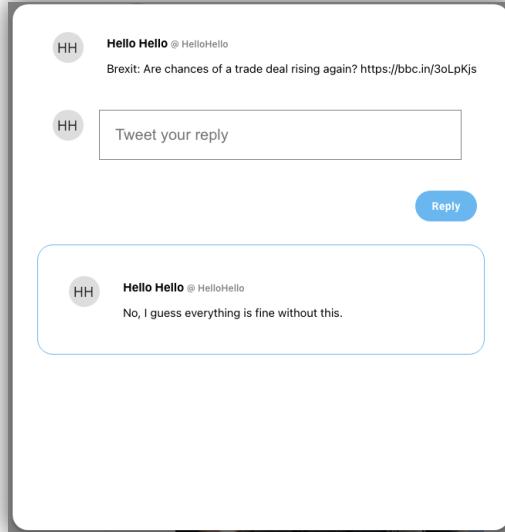
### JSON Request for likes

```
General
Request URL: http://a3eeb7e1a7db54183bb136b82dc3dca-1941134974.us-east-2.elb.amazonaws.com/v1/c00b79e9-597f-4bfc-940f-b9d7476a5a4f/likes
Request Method: PUT
Status Code: 200 OK
Remote Address: 3.129.41.38:80
Referer Policy: strict-origin-when-cross-origin
Response Headers
  Access-Control-Allow-Credentials: true
  Access-Control-Allow-Origin: *
  Connection: keep-alive
  Content-Length: 48
  Content-Type: application/json; charset=utf-8
  Date: Sun, 13 Dec 2020 19:38:06 GMT
  ETag: W/"28-YXplicr+3se1P093Yy1fpakgnC8"
  X-Powered-By: Express
```

# Final Project – Twitter Application

## JSON Request for likes on tweet

```
▼ Request Headers View source
Accept: */*
Accept-Encoding: gzip, deflate
Accept-Language: en-US, en;q=0.9
Access-Control-Allow-Origin: *
Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cC16IkpxVCJ9.eyJzdWIiOiI1ZmQ2Njk2001lM2NhMTU00WMMmE5Zdk1LCJpYXQiOjE2M0c400cyMTgsImV4cCI6MTYwODQSMjAxOH8.9SpMnR7tDmsqoJsRA_EeGb-uP
EmaYmKnKzhrRIVf4s
Connection: keep-alive
Content-Length: 47
Content-Type: application/json
Cookie: session=%22firstName%22%22lastName%22%22test%22%22username%22%22priyam%22%22createdDate%22%222020-12-13T19:20:08.521Z%22%2C%22id%22%225fd669689e3ca1549c02a9d9%22%22token%22%22eyJhbGciOiJIUzI1NiIsInR5cC16IkpxVCJ9.eyJzdWIiOiI1ZmQ2Njk2001lM2NhMTU00WMMmE5Zdk1LCJpYXQiOjE2M0c400cyMTgsImV4cCI6MTYwODQSMjAxOH8.9SpMnR7tDmsqoJsRA_EeGb-uP
Host: a3eeb7e1a7db54103bb136b02dc3dca-1941134974.us-east-2.elb.amazonaws.com
Origin: http://a3eeb7e1a7db54103bb136b02dc3dca-1941134974.us-east-2.elb.amazonaws.com
Referer: http://a3eeb7e1a7db54103bb136b02dc3dca-1941134974.us-east-2.elb.amazonaws.com/home
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/87.0.4288.88 Safari/537.36
```



After User click on the Like Button it will add to the backend

```
▼ Request Payload View source
▼ {userId: "5fd669689e3ca1549c02a9d9", liked: 1}
  liked: 1
  userId: "5fd669689e3ca1549c02a9d9"
```

```
x Headers Preview Response Initiator Timing Cookies
1 {"message": "Tweet updated successfully"}
```

## Final Project – Twitter Application

### Comment Request to Get Comments from the backend

```
▼ General
Request URL: http://a3eeb7e1a7db54103bb136b02dcb3dca-1941134974.us-east-2.elb.amazonaws.com/v1/c00b79e9-597f-4bfc-940f-b9d7476a5a4f/comments
Request Method: PUT
Status Code: 200 OK
Remote Address: 3.129.41.38:80
Referrer Policy: strict-origin-when-cross-origin

▼ Response Headers    view source
Access-Control-Allow-Credentials: true
Access-Control-Allow-Origin: *
Connection: keep-alive
Content-Length: 48
Content-Type: application/json; charset=utf-8
Date: Sun, 13 Dec 2020 19:32:09 GMT
ETag: W/"28-YXwpicr+3selF093Yy1fpakgnC8"
X-Powered-By: Express
```

### Adding Comment Request

```
▼ Request Payload    view source
{
  "commentId": "e3c282a8-6011-4d9e-8813-145dd383c795",
  "commentedBy": "priyam test",
  "userName": "priyam",
  "comment": "hello world",
  "commentId": "e3c282a8-6011-4d9e-8813-145dd383c795",
  "commentedBy": "priyam test",
  "userName": "priyam"
}
```

### Response from backend after request are submitted

Headers	Preview	Response	Initiator	Timing	Cookies
		1 {"message": "Tweet updated successfully"}			

- Included BBC News API to show the recent news.
- Logout from the account.

## Final Project – Twitter Application

The screenshot shows a Twitter-like application interface. At the top left is a profile icon with the letters 'HH'. Below it, the word 'Home' is displayed. Two tweets are listed:

- Hello Hello @HelloHello**  
Brexit: Are chances of a trade deal rising again? <https://bbc.in/3oLpkjs>
- Hello Hello @HelloHello**  
Rangers extended their unbeaten run to 26 games and remain 13 points clear at the top of the Scottish Premiership.

Each tweet has a '0' under the reply icon and a '0' under the like icon.

The screenshot shows a 'What's happening' feed. At the top right is a refresh button. Below it, the heading 'Tweets by @BBCNews' is followed by a BBC News tweet:

**BBC News (UK) Retweeted**  
**BBC Sport** @BBCSport  
FT: Fulham 1-1 Liverpool  
Mohamed Salah's second-half penalty earns Liverpool a point.  
Tottenham remain top of the table on goal difference.  
Listen: <bbc.in/5live>  
Follow: <bbc.in/3qPOHvX#FULLIV> #bbcfootball

Below this is a thumbnail image of a football match with a player in red about to score a goal. There are like and share icons below the image.

## Get All Tweets Request

This screenshot shows the Network tab of a browser developer tools window. A successful GET request for '/v1/tweets' is listed. The response body contains JSON data representing the tweets from the previous screenshots.

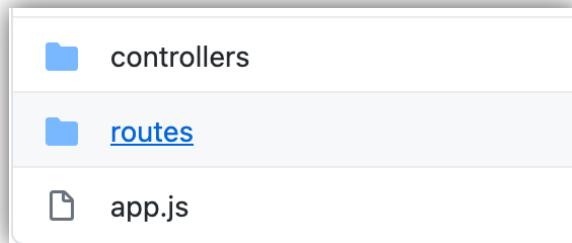
## JSON Data

This screenshot shows a JSON viewer tool displaying the tweet data. The structure is as follows:

- array [2]:
  - 0 {7}:
    - tweetId: 102e679e-4ab3-4460-a96b-1751c6ecd351
    - tweet: wooo
    - imageLink: <https://res.cloudinary.com/ravi479/image/upload/v1607887100/jzusxtspdixsvxiop08.gif>
    - createdBy {3}:
      - displayName: ravi test
      - userName: ravi
      - avatarLink: value
    - likes [2]:
      - 0: 5fd668d2d601ae99346df47e
      - 1: 5fd669379e3ca1b61402a9d8
    - comments [1]:
      - 0 {5}:
        - commentId: 97409e04-ac29-4f94-97e6-207044162a38
        - commentedBy: ravi test
        - userName: ravi
        - avatarLink: value
        - comment: testwooo
    - 1 {7}:
      - tweetId: 148e197f-fa86-428d-9ea7-5483daba39c0
      - tweet: cool

## 2. Redis Server

The Redis server will act as the middleware for the application. All the request from the frontend will be forwarded to the Redis server and it will act as the queuing part.



The Set interval method is used to queue the api calls and after every 6000ms it will be pushed to the backend server.

```
setInterval(clearCache, 1000 * 60 * cacheTimeOut);

// Function used to clear cache after every cacheTimeOut minutes and calls the main backend to
// update the records
function clearCache() {
  // get the values from cache
  client.lrange(myKey, 0, -1, function (err, reply) {
    if (!err) {
      const tweets = reply.map(JSON.parse);
      if (tweets && tweets.length > 0) {
        // call the node server to dump the data
        fetch(`${baseUrl}/tweets`, {
          method: "POST",
          body: JSON.stringify(tasks),
          headers: { "Content-Type": "application/json" },
        })
        .then((res) => res.json())
        .then((json) => console.log(json));
      }
    }
  });
}
```

## Final Project – Twitter Application

### Redis Server Code

```
const express = require("express"),
app = express(),
bodyParser = require("body-parser"),
port = process.env.PORT || 3000,
cors = require("cors");
// enable cors
app.use(cors());

//Adding body parser for handling request and response objects.
app.use(
  bodyParser.urlencoded({
    extended: true,
  })
);
app.use(bodyParser.json({ limit: "50mb" }));
app.use(
  bodyParser.urlencoded({
    limit: "50mb",
    extended: true,
    parameterLimit: 50000,
  })
);

//Initialize app
let initApp = require("./api/app");
initApp(app);

app.listen(port);
console.log(`ChatterApp Redis server started on: ${port}`);
```

### 3. Backend

The backend of the application relates to the Redis Server, where all the request is gathered and further it is transported to the backend Node JS server.

Server Code:

```
const utilConstants = require("./api/utils/Constants");
const express = require("express"),
app = express(),
port = process.env.PORT || utilConstants.PORT,
mongoose = require("mongoose"), //created model loading here
bodyParser = require("body-parser");
const cors = require("cors");
const jwt = require("./api/utils/jwt");
const errorHandler = require("./api/utils/error_handler");

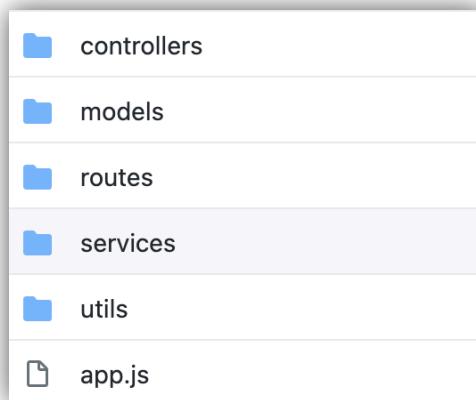
// Mongo Atlas
const uri = utilConstants.MONGODB_URL;
mongoose.connect(uri, {
useNewUrlParser: true,
useUnifiedTopology: true,
useFindAndModify: false,
useCreateIndex: true,
promiseLibrary: global.Promise,
})
.then(() => {
  console.log("MongoDB Connected...")
})
.catch(err => console.log(err))

// enable cors
app.use(cors());
app.use(jwt());
app.use(errorHandler);
//Adding body parser for handling request and response objects.
app.use(
  bodyParser.urlencoded({
    extended: true,
  })
);
app.use(bodyParser.json({ limit: "50mb" }));
app.use(
  bodyParser.urlencoded({
    limit: "50mb",
    extended: true,
    parameterLimit: 50000,
  })
);

//Initialize app
let initApp = require("./api/app");
initApp(app);

app.listen(port);
console.log("ChatterApp RESTful API server started on: " + port);
```

## Server API folder Structure



The Server Controllers API contains the Tweet methods which are responsible for creating and displaying tweets, adding comments to the tweet, adding like to the tweet and User controller methods which are responsible for creating a new user or fetching the existing user details.

The Server Model API contains the Schema of the tweet and user data.

The Route services API contains the routes through which the data will be communicated between frontend and backend.

## Final Project – Twitter Application

We have created the following endpoint for the backend communication –  
Create User: Post method - <http://localhost:3000/v1/register>

```
object {4}
  username : ravi
  firstName : Ravi
  lastName : Kumar
  password : Admin@123
```

If the user is successfully created, the server will respond with the success message.

```
1 {
2   "message": "User Registered successfully"
3 }
```

If there is some issue while creating the user, the server will respond with the failure message.

```
1 {
2   "message": "Some issue occurred while registering the user"
3 }
```

User Login: Post Method - <http://localhost:3000/v1/authenticate>

```
object {2}
  username : ravi
  password : Admin@123
```

## Final Project – Twitter Application

If the credentials provided are correct, the server will respond with the success message as follow

```
{  
    "username": "ravi123",  
    "firstName": "ravi",  
    "lastName": "test1",  
    "createdDate": "2020-12-06T03:34:54.186Z",  
    "id": "5fcc515e6b3aed35842e3d59",  
    "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.  
eyJzdWIiOiIzMjNjNTE1ZTUiM2FlZDM10DQyZTNkNTkiLCJpYXQiOjE2MDcyMjc5NDQsImV4cCI6MTYwNzgzMjc0NH0.  
GEqHgUCHNK8YjZjsI-qadSWYVnn4Z2nxEi50o2MapVA"  
}
```

If the credentials provided are not correct, the server will respond with the failure message

```
{"message": "Invalid username or password"}
```

Creating Tweet: Post method - <http://localhost:3000/v1/tweets>

The parameters used for the creating tweets are as below –

TweetID – uuid

CreatedBy – Name of the user who created the tweet

CreatedAt – TimeStamp

```
{  
    "tweetId": "3",  
    "tweet": "Some tweet description from frontend3",  
    "createdBy": {  
        "userName": "username from frontend",  
        "avatarLink": "user image link from frontend"  
    },  
    "createdAt": "TimeStamp from frontend",  
    "imageLink": "Link of Any image for the tweet"  
}
```

If the entered data validated and the tweet is stored in the database, the server will acknowledge using the below success message.

## Final Project – Twitter Application

```
{  
    "tweetId": "3",  
    "tweet": "Some tweet description from frontend3",  
    "imageLink": "Link of Any image for the tweet",  
    "createdBy": {  
        "userName": "username from frontend",  
        "avatarLink": "user image link from frontend"  
    },  
    "createdAt": "timeStamp from frontend",  
    "likes": [],  
    "comments": []  
}
```

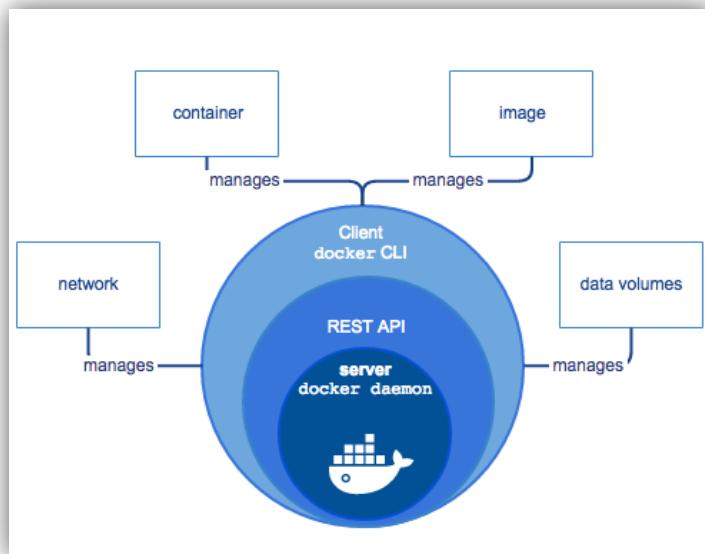
If it fails, the server will acknowledge with the below error message

```
{  
    "message": "Invalid Token"  
}
```

## 4. Infrastructure

### Docker

- Docker is an open platform for developing, shipping, and running applications. Docker enables you to separate your applications from your infrastructure so you can deliver software quickly.
- With Docker, you can manage your infrastructure in the same ways you manage your applications.
- By taking advantage of Docker's methodologies for shipping, testing, and deploying code quickly, you can significantly reduce the delay between writing code and running it in production.



## Final Project – Twitter Application

All of the microservices will containerized as a separate container  
Create a separate docker files for frontend, middleware, and Backend.

- Docker file for React Application

```
2 lines (2 sloc) | 43 Bytes
1 FROM nginx
2 COPY build /usr/share/nginx/html
```

- Docker file for Redis Server

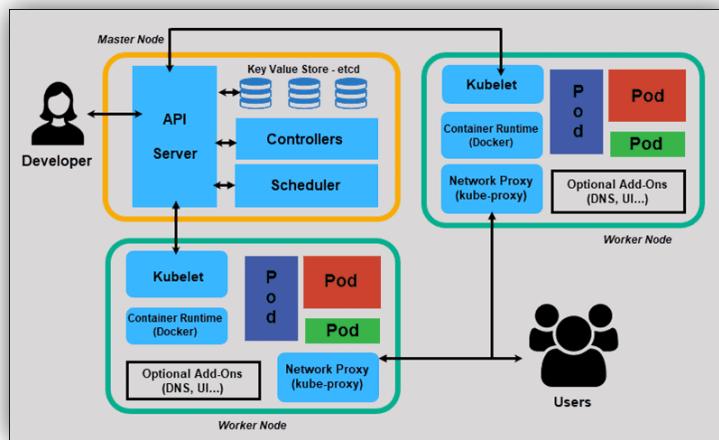
```
7 lines (7 sloc) | 118 Bytes
1 FROM node:14-slim
2 WORKDIR /usr/src/app
3 COPY ./package.json ./
4 RUN npm install
5 COPY . .
6 EXPOSE 3000
7 CMD [ "server.js" ]
```

- Docker file for the Node Server

```
7 lines (7 sloc) | 118 Bytes
1 FROM node:14-slim
2 WORKDIR /usr/src/app
3 COPY ./package.json ./
4 RUN npm install
5 COPY . .
6 EXPOSE 3000
7 CMD [ "server.js" ]
```

## Kubernetes

- Kubernetes is a portable, extensible, open-source platform for managing containerized workloads and services, that facilitates both declarative configuration and automation. It has a large, rapidly growing ecosystem. Kubernetes services, support, and tools are widely available.



- In Kubernetes, we can expose the services publicly by choosing the type Load Balancer. That will create a public IP address for each service.
- But we want to reduce the number of IP addresses to make some saving and we want to map a URL like mycompany.com/login and mycompany.com/products to the right. Well, this could be done through Kubernetes Ingress resources.
- Kubernetes API does not provide an implementation for an ingress controller. So, we need to install it ourselves.

Many ingress controllers are supported for Kubernetes:

1. Nginx Controller
2. HAProxy Ingress, Contour, Citrix Ingress Controller
3. API Gateways like Traffic, Kong and Ambassador
4. Service mesh like Istio
5. Cloud managed ingress controllers like Application Gateway Ingress Controller (AGIC), AWS ALB Ingress Controller, Ingress GCE

The first part will start by configuring Ingress:

1. Installing an ingress controller (NGINX) into Kubernetes.
2. Deploying different applications/services.
3. Configuring ingress to route traffic depending on the URL.
4. Configure SSL/TLS using Cert Manager

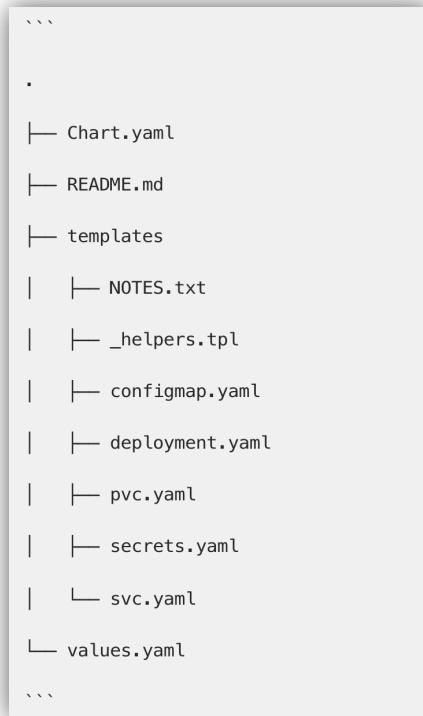
```
csye7220-vm@ubuntu:~/chatter$ kubectl get pods
NAME                      READY   STATUS    RESTARTS   AGE
mongo-f7d8b86bd-8pwv      1/1     Running   0          87m
nginx-ddfc8464-bqb9z     1/1     Running   0          86m
twitter-backend-5899fcf9c4-9gqwt 1/1     Running   0          87m
twitter-backend-5899fcf9c4-tkmn4 1/1     Running   0          87m
twitter-redis-5587855cc7-j85ff 1/1     Running   0          86m
twitter-redis-5587855cc7-lj45j 1/1     Running   0          86m
twitter-redis-node-78f9cc978f-92ndp 1/1     Running   0          87m
csye7220-vm@ubuntu:~/chatter$
```

### Helm Chart

- Helm is a package Manager for Kubernetes. Package managers (like yum and APT) are a big reason many of the popular Linux distributions are as successful as they are.
- They provide a standard, opinionated way to install, configure, upgrade, and run an application in a matter of minutes.
- The packages themselves are open source, and anyone can contribute to them.
- Helm is made of two components:
  - A server called Tiller, which runs inside your Kubernetes cluster and a client called helm that runs on your local machine.
  - A package is called a chart to keep with the maritime theme.
  - With the Helm client, you can browse package repositories and deploy those Charts on your Kubernetes cluster.
- Helm will pull the Chart and talking to Tiller will create a *release* (an instance of a Chart).

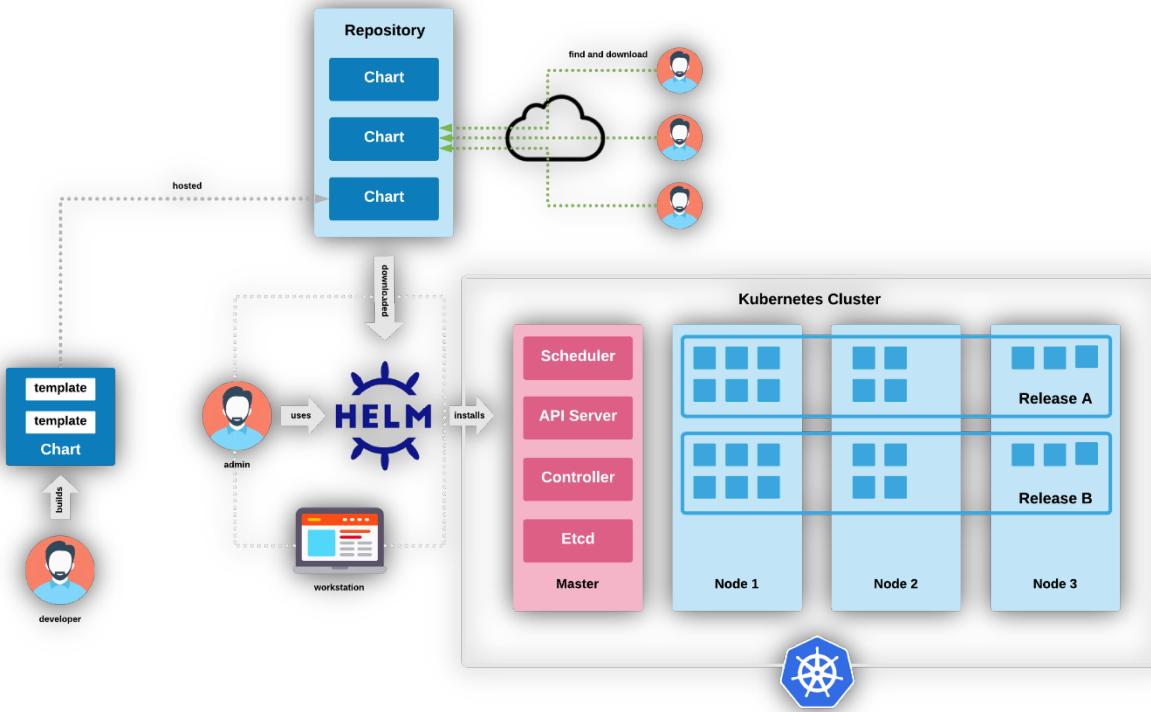
### Structure of a Chart

A Chart is easy to demystify; it is an archive of a set of Kubernetes resource manifests that make up a distributed application.



- `Chart.yaml`, contains some metadata about the Chart, such as its name, version, keywords.
- `values.yaml`, contains keys and values that are used to generate the release in your Cluster. These values are replaced in the resource manifests.
- `configMap.yaml`, contains database configuration.
- `secrets.yaml`, database passwords are stored in secret file.

## Final Project – Twitter Application



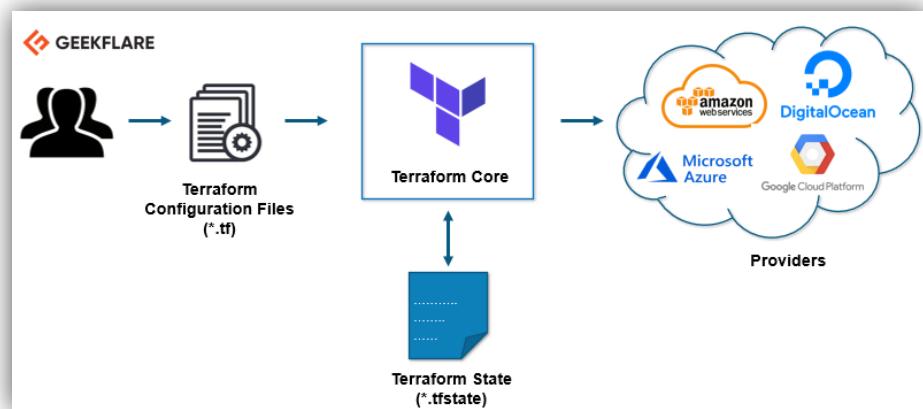
### Commands to Run:

- Check all the values in the Helm chart are correct –  
`helm lint ./twitterhelmchart`
- Dry Run (Check everything is running well without deploying actual services) –  
`helm install --dry-run --debug ./twitterhelmchart --generate-name`
- Run the Helm chart - `helm install example ./twitterhelmchart/ --set service.type=NodePort`

## Final Project – Twitter Application

### Terraform

- Terraform is a tool for building, changing, and versioning infrastructure safely and efficiently.
- Terraform can manage existing and popular service providers as well as custom in-house solutions.
- Configuration files describe to Terraform the components needed to run a single application or your entire datacenter.
- Terraform generates an execution plan describing what it will do to reach the desired state, and then executes it to build the described infrastructure.
- As the configuration changes, terraform can determine what changed and create incremental execution plans which can be applied.



## Final Project – Twitter Application

### Ingress Controller

Kubernetes ingress is a collection of routing rules that govern how external users access services running in a Kubernetes cluster? However, in real-world Kubernetes deployments, there are frequently additional considerations beyond routing for managing ingress. We'll discuss these requirements in more detail below.

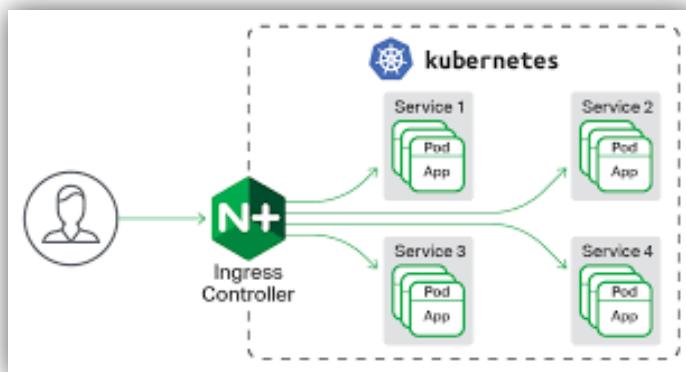
### Ingress in Kubernetes

In Kubernetes, there are three general approaches to exposing your application.

Using a Kubernetes service of type Node Port, which exposes the application on a port across each of your nodes.

Use a Kubernetes service of type Load Balancer, which creates an external load balancer that points to a Kubernetes service in your cluster

Use a Kubernetes Ingress Resource

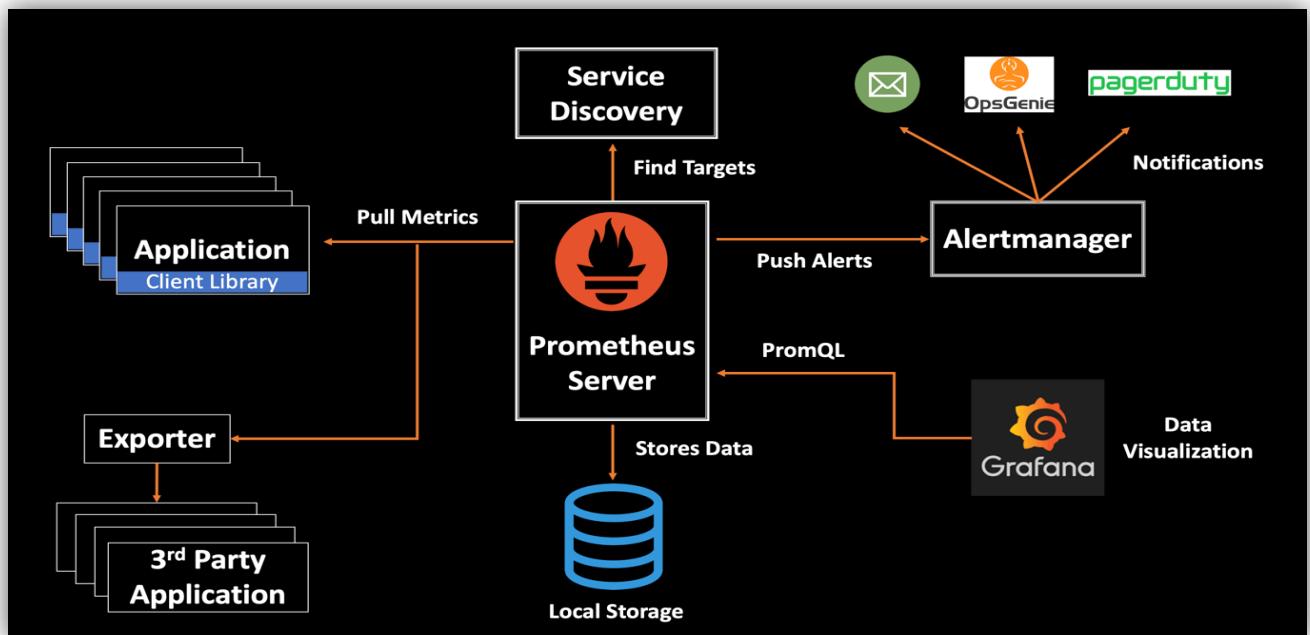


### Ingress resource Configuration

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: ingress-node-sample-helloworld
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /
    nginx.ingress.kubernetes.io/enable-cors: "true"
    kubernetes.io/ingress.class: nginx
spec:
  rules:
    # Uncomment the below to only allow traffic from this domain and route based on it
    # - host: my-host # your domain name with A record pointing to the nginx-ingress-controller IP
    - http:
        paths:
          - path: / # Everything on this path will be redirected to the rewrite-target
            backend:
              serviceName: nginx # the exposed svc name and port
              servicePort: 80
          - path: /auth # Everything on this path will be redirected to the rewrite-target
            backend:
              serviceName: nginx # the exposed svc name and port
              servicePort: 80
          - path: /home
            backend:
              serviceName: nginx
              servicePort: 80
```

## Prometheus Server

Prometheus is an open-source system monitoring and alerting toolkit originally built at SoundCloud. Since its inception in 2012, many companies and organizations have adopted Prometheus, and the project has a very active developer and user community.



## Prometheus Service yaml file

15 lines (15 sloc)   290 Bytes
<pre> 1  apiVersion: v1 2  kind: Service 3  metadata: 4    name: prometheus-service 5    namespace: monitoring 6    annotations: 7      prometheus.io/scrape: 'true' 8      prometheus.io/port: '9090' 9  spec: 10   selector: 11     app: prometheus-server 12     type: LoadBalancer 13   ports: 14     - port: 80 15       targetPort: 9090   </pre>

## Final Project – Twitter Application

### Prometheus Dashboard

```
csye7220-vm@ubuntu:~/chatter/infrastructure/kubernetes$ kubectl get pods --namespace monitoring
NAME                                         READY   STATUS    RESTARTS   AGE
alertmanager-prometheus-oper-alertmanager-0   2/2     Running   0          78m
prometheus-grafana-85b4dbb556-kjpsv          2/2     Running   0          78m
prometheus-kube-state-metrics-6df5d44568-r2qhp 1/1     Running   0          78m
prometheus-prometheus-node-exporter-2t4z5      1/1     Running   0          78m
prometheus-prometheus-node-exporter-p7xwl       1/1     Running   0          78m
prometheus-prometheus-node-exporter-tw9cp       1/1     Running   0          78m
prometheus-prometheus-operator-85cc758cdb-wqmlm 2/2     Running   0          78m
prometheus-prometheus-prometheus-0             3/3     Running   1          78m
csye7220-vm@ubuntu:~/chatter/infrastructure/kubernetes$
```

```
csye7220-vm@ubuntu:~/chatter/infrastructure/kubernetes$ kubectl get all
NAME           TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
service/kubernetes   ClusterIP   172.28.0.1   <none>        443/TCP   83m
NAME           REFERENCE  TARGETS  MINPODS  MAXPODS  REPLICAS  AGE
horizontalpodautoscaler.autoscaling/nginx   Deployment/nginx  1% / 5%  1         10        1         50m
```

```
csye7220-vm@ubuntu:~/chatter/infrastructure/kubernetes$ kubectl get ingress
NAME          HOSTS   ADDRESS   PORTS   AGE
ingress-node-sample-helloworld   *          80        8s
csye7220-vm@ubuntu:~/chatter/infrastructure/kubernetes$
```

### Prometheus using Custom Metrics

Example is the custom metric, and I have also enabled metrics server to get started.

Targets						
All		Unhealthy				
default/example/0 (2/2 up) <a href="#">show less</a>						
Endpoint	State	Labels		Last Scrape	Scrape Duration	Error
<a href="http://10.0.1.225:3000/metrics">http://10.0.1.225:3000/metrics</a>	<span>UP</span>	<code>endpoint="web" instance="10.0.1.225:3000" job="twitter-redis-db" namespace="default" pod="twitter-redis-5587855c7c85f9"</code>		15.513s ago	3.942ms	
<a href="http://10.0.3.50:3000/metrics">http://10.0.3.50:3000/metrics</a>	<span>UP</span>	<code>endpoint="web" instance="10.0.3.50:3000" job="twitter-redis-db" namespace="default" pod="twitter-redis-5587855c7c85f9"</code>		22ms ago	6.264ms	
monitoring/prometheus-prometheus-oper-alertmanager/0 (1/1 up) <a href="#">show less</a>						
Endpoint	State	Labels		Last Scrape	Scrape Duration	Error
<a href="http://10.0.2.40:9093/metrics">http://10.0.2.40:9093/metrics</a>	<span>UP</span>	<code>endpoint="web" instance="10.0.2.40:9093" job="prometheus-prometheus-oper-alertmanager" namespace="monitoring" pod="alertmanager-prometheus-prometheus-oper-alertmanager-0" service="prometheus-prometheus-oper-alertmanager"</code>		22.73s ago	4.474ms	
monitoring/prometheus-prometheus-oper-apiserver/0 (2/2 up) <a href="#">show less</a>						
Endpoint	State	Labels		Last Scrape	Scrape Duration	Error
<a href="https://10.0.1.221:443/metrics">https://10.0.1.221:443/metrics</a>	<span>UP</span>	<code>endpoint="https" instance="10.0.1.221:443" job="apiserver" namespace="default" service="kubernetes"</code>		8.07s ago	69.41ms	
<a href="https://10.0.3.71:443/metrics">https://10.0.3.71:443/metrics</a>	<span>UP</span>	<code>endpoint="https" instance="10.0.3.71:443" job="apiserver" namespace="default" service="kubernetes"</code>		23.039s ago	112ms	
monitoring/prometheus-prometheus-oper-coredns/0 (2/2 up) <a href="#">show less</a>						
Endpoint	State	Labels		Last Scrape	Scrape Duration	Error
<a href="http://10.0.3.35:9153/metrics">http://10.0.3.35:9153/metrics</a>	<span>UP</span>	<code>endpoint="http-metrics" instance="10.0.3.35:9153" job="coredns" namespace="kube-system" pod="coredns-670d075c5-2vewt" service="prometheus-prometheus-oper-coredns"</code>		26.128s ago	6.475ms	
<a href="http://10.0.3.68:9153/metrics">http://10.0.3.68:9153/metrics</a>	<span>UP</span>	<code>endpoint="http-metrics" instance="10.0.3.68:9153" job="coredns" namespace="kube-system" pod="coredns-670d075c5-2vewt" service="prometheus-prometheus-oper-coredns"</code>		14.659s ago	4.069ms	
monitoring/prometheus-prometheus-oper-grafana/0 (1/1 up) <a href="#">show less</a>						
Endpoint	State	Labels		Last Scrape	Scrape Duration	Error
<a href="http://10.0.1.67:3000/metrics">http://10.0.1.67:3000/metrics</a>	<span>UP</span>	<code>endpoint="service" instance="10.0.1.67:3000" job="grafana" namespace="monitoring" pod="grafana-85b4dbb556-kjpsv" service="grafana"</code>		16.46s ago	3.076ms	

## Final Project – Twitter Application

```
# HELP my_application:nodejs_heap_space_size_available_bytes Process heap space size available from Node.js in bytes.
# TYPE my_application:nodejs_heap_space_size_available_bytes gauge
my_application:nodejs_heap_space_size_available_bytes{space="read_only"} 0
my_application:nodejs_heap_space_size_available_bytes{space="new"} 133352
my_application:nodejs_heap_space_size_available_bytes{space="old"} 366528
my_application:nodejs_heap_space_size_available_bytes{space="code"} 18752
my_application:nodejs_heap_space_size_available_bytes{space="map"} 70768
my_application:nodejs_heap_space_size_available_bytes{space="large_object"} 0
my_application:nodejs_heap_space_size_available_bytes{space="code_large_object"} 0
my_application:nodejs_heap_space_size_available_bytes{space="new_large_object"} 1047424

# HELP my_application:nodejs_version_info Node.js version info.
# TYPE my_application:nodejs_version_info gauge
my_application:nodejs_version_info{version="v14.15.1",major="14",minor="15",patch="1"} 1

# HELP my_application:nodejs_gc_duration_seconds Garbage collection duration by kind, one of major, minor, incremental or weakcb.
# TYPE my_application:nodejs_gc_duration_seconds histogram
my_application:nodejs_gc_duration_seconds_bucket{le="0.001",kind="minor"} 67
my_application:nodejs_gc_duration_seconds_bucket{le="0.01",kind="minor"} 69
my_application:nodejs_gc_duration_seconds_bucket{le="0.1",kind="minor"} 69
my_application:nodejs_gc_duration_seconds_bucket{le="1",kind="minor"} 69
my_application:nodejs_gc_duration_seconds_bucket{le="5",kind="minor"} 69
my_application:nodejs_gc_duration_seconds_bucket{le="Inf",kind="minor"} 69
my_application:nodejs_gc_duration_seconds_sum{kind="minor"} 0.03412943200000001
my_application:nodejs_gc_duration_seconds_count{kind="minor"} 69
my_application:nodejs_gc_duration_seconds_bucket{le="0.001",kind="incremental"} 4
my_application:nodejs_gc_duration_seconds_bucket{le="0.01",kind="incremental"} 6
my_application:nodejs_gc_duration_seconds_bucket{le="0.1",kind="incremental"} 6
my_application:nodejs_gc_duration_seconds_bucket{le="1",kind="incremental"} 6
my_application:nodejs_gc_duration_seconds_bucket{le="5",kind="incremental"} 6
my_application:nodejs_gc_duration_seconds_bucket{le="Inf",kind="incremental"} 6
my_application:nodejs_gc_duration_seconds_sum{kind="incremental"} 0.013888233
my_application:nodejs_gc_duration_seconds_count{kind="incremental"} 6
my_application:nodejs_gc_duration_seconds_bucket{le="0.001",kind="major"} 0
my_application:nodejs_gc_duration_seconds_bucket{le="0.01",kind="major"} 3
my_application:nodejs_gc_duration_seconds_bucket{le="0.1",kind="major"} 3
my_application:nodejs_gc_duration_seconds_bucket{le="1",kind="major"} 3
my_application:nodejs_gc_duration_seconds_bucket{le="5",kind="major"} 3
my_application:nodejs_gc_duration_seconds_bucket{le="Inf",kind="major"} 3
my_application:nodejs_gc_duration_seconds_sum{kind="major"} 0.01288049499999999
my_application:nodejs_gc_duration_seconds_count{kind="major"} 3

# HELP my_application:hello_duration Duration of HTTP requests in ms
# TYPE my_application:hello_duration histogram

# HELP get_tweets_route To get all the tweets
# TYPE get_tweets_route counter
get_tweets_route 11

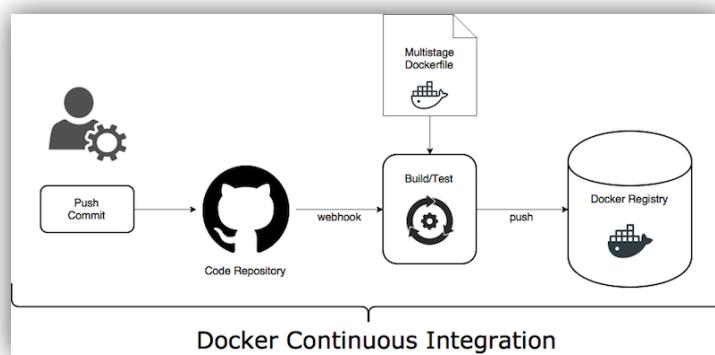
# HELP metric_name metric_help
# TYPE metric_name counter
metric_name 0
```

## Service Discovery

- [default/example/0](#) (2/13 active targets)
- [monitoring/prometheus-prometheus-oper-alertmanager/0](#) (1/16 active targets)
- [monitoring/prometheus-prometheus-oper-apiserver/0](#) (2/13 active targets)
- [monitoring/prometheus-prometheus-oper-coredns/0](#) (2/25 active targets)
- [monitoring/prometheus-prometheus-oper-grafana/0](#) (1/16 active targets)
- [monitoring/prometheus-prometheus-oper-kube-controller-manager/0](#) (0/25 active targets)
- [monitoring/prometheus-prometheus-oper-kube-etcd/0](#) (0/25 active targets)
- [monitoring/prometheus-prometheus-oper-kube-proxy/0](#) (3/25 active targets)
- [monitoring/prometheus-prometheus-oper-kube-scheduler/0](#) (0/25 active targets)
- [monitoring/prometheus-prometheus-oper-kube-state-metrics/0](#) (1/16 active targets)
- [monitoring/prometheus-prometheus-oper-kubelet/0](#) (3/25 active targets)
- [monitoring/prometheus-prometheus-oper-kubelet/1](#) (3/25 active targets)
- [monitoring/prometheus-prometheus-oper-kubelet/2](#) (3/25 active targets)
- [monitoring/prometheus-prometheus-oper-kubelet/3](#) (3/25 active targets)
- [monitoring/prometheus-prometheus-oper-node-exporter/0](#) (3/16 active targets)
- [monitoring/prometheus-prometheus-oper-operator/0](#) (1/16 active targets)
- [monitoring/prometheus-prometheus-oper-prometheus/0](#) (1/16 active targets)

### Continuous Integration using GitHub actions

Continuous Integration (CI) is a development practice where developers integrate code into a shared repository frequently, preferably several times a day. Each integration can then be verified by an automated build and automated tests. While automated testing is not strictly part of CI it is typically implied.



Screenshot of the GitHub Workflows interface showing the "All workflows" page. It displays 23 results, including recent runs for "ci-changes" and a merge pull request. The interface includes filters for Event, Status, Branch, and Actor.

Event	Status	Branch	Actor
ci #1: Commit 0829f0c pushed by dburugupalli	dev		33 seconds ago In progress
ci #7: Commit 0829f0c pushed by dburugupalli	dev		34 seconds ago In progress
ci #1: Commit 0829f0c pushed by dburugupalli	dev		41 seconds ago 41s
Merge pull request #16 from dburugupalli/feature/infr...	dev		17 hours ago 40s
updated infrastructure README.md	feature/infrastructure		17 hours ago 40s

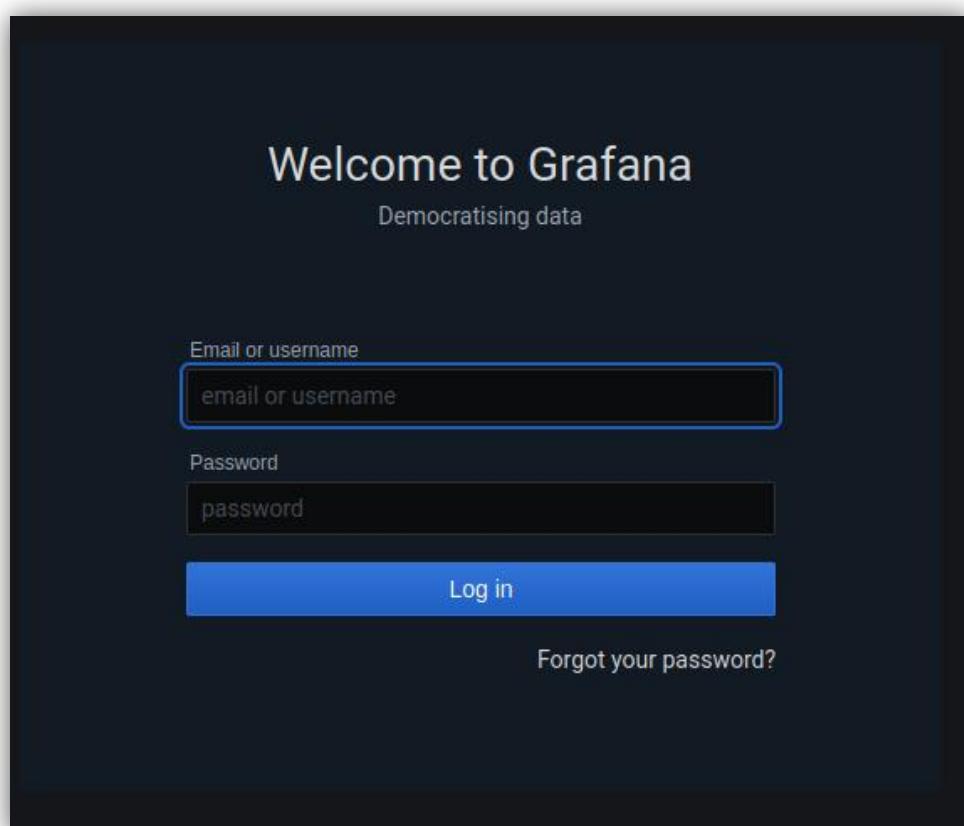
Screenshot of the GitHub Repositories interface for the "devopschatter" organization. It shows three repositories: "twitter-client", "chatter", and "twitter-server". Each repository is listed with its name, last updated time, and visibility status (Public).

NAME	LAST UPDATED	VISIBILITY
devopschatter / twitter-client	3 minutes ago	Public
devopschatter / chatter	4 minutes ago	Public
devopschatter / twitter-server	4 minutes ago	Public

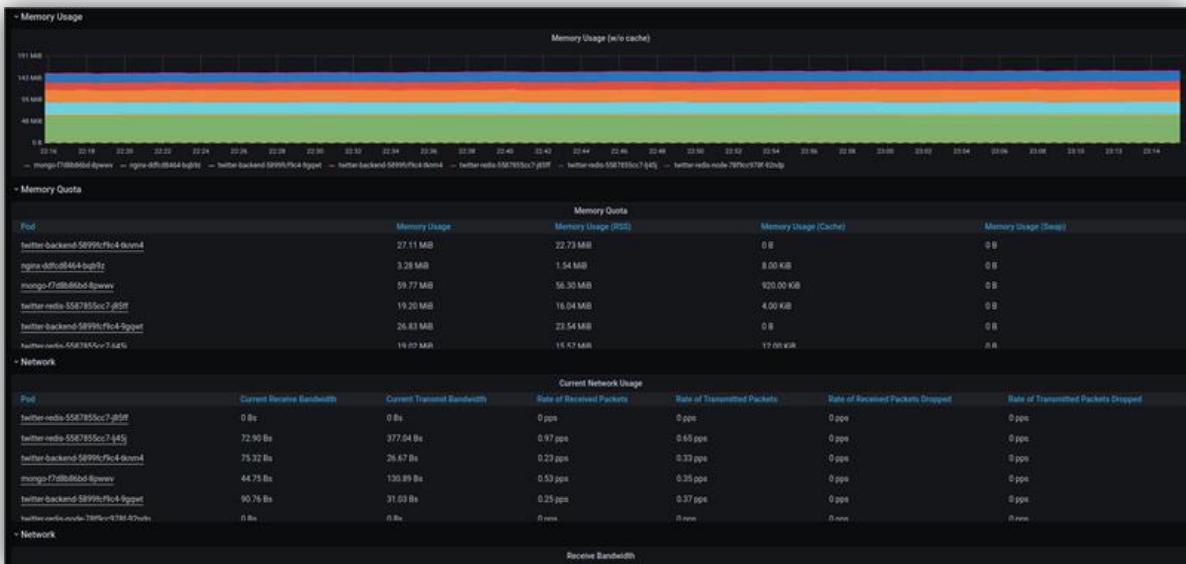
## Designing Monitoring dashboard Grafana

Grafana is an open-source visualization tool that can be used on top of a variety of different data stores but is most commonly used together with Graphite, InfluxDB, Prometheus, Elasticsearch and Logz.io. As it so happens, Grafana began as a fork of Kibana, trying to supply support for metrics (a.k.a. monitoring) that Kibana (at the time) did not provide much if any such support for.

Essentially, Grafana is a feature-rich replacement for Graphite-web, which helps users to easily create and edit dashboards. It contains a unique Graphite target parser that enables easy metric and function editing. Users can create comprehensive charts with smart axis formats (such as lines and points) as a result of Grafana's fast, client-side rendering — even over long ranges of time — that uses Flot as a default option.



# Final Project – Twitter Application



## Logging using ELK stack

### Elastic Search, Logstash and Kibana

Elastic Search is analysis and search engine and Logstash is a server-side data processing pipeline that ingests the data from multiple sources simultaneously, transforms and sends it to a stash like

Elastic search, Kibana lets the users visualize the charts and graphs in elastic search

```
csye7220-vm@ubuntu:~/chatter/infrastructure/efk$ kubectl get pods
NAME                               READY   STATUS    RESTARTS   AGE
elasticsearch-856f49d4b4-tmwvs      1/1     Running   0          54s
kibana-584b78d8bd-jq657            1/1     Running   0          52s
logstash-7f8f44cf8-h4jc4           1/1     Running   0          49s
mongo-f7d8b86bd-8pwwv              1/1     Running   0          3h16m
nginx-ddfc8464-bqb9z               1/1     Running   0          3h16m
twitter-backend-5899fcf9c4-9gqwt   1/1     Running   0          3h16m
twitter-backend-5899fcf9c4-tknm4   1/1     Running   0          3h16m
twitter-redis-5587855cc7-j85ff    1/1     Running   0          3h16m
twitter-redis-5587855cc7-lj45j    1/1     Running   0          3h16m
twitter-redis-node-78f9cc978f-92ndp 1/1     Running   0          3h16m
```

### Kibana logging dashboard

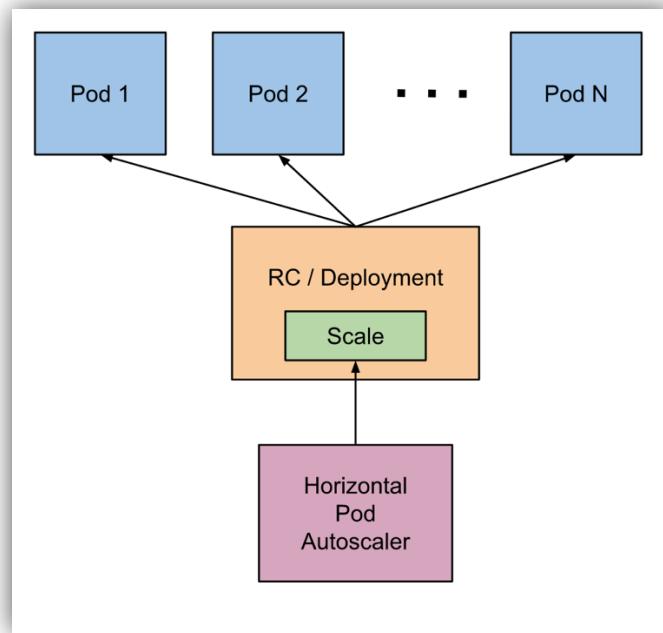
The screenshot shows the Kibana Settings interface for the '\_source' field. On the left, there's a sidebar with 'Selected Fields' containing '\_score'. Below it is a list of 'Available Fields' including '\_id', '\_index', '\_score', '\_type', 'buildNum', 'cluster\_name', 'cluster\_state.master\_n...', 'cluster\_state.nodes.ja...', 'cluster\_state.nodes.ja...', 'cluster\_state.nodes.ja...', 'cluster\_state.nodes.ja...', 'cluster\_state.nodes.ja...', 'cluster\_state.state\_uuid', 'cluster\_state.status', 'cluster\_state.version', 'cluster\_stats.indices.co...', 'cluster\_stats.indices.co...', and 'cluster\_stats.indices.do...'. A blue 'add' button is positioned next to '\_score'. The main area displays the JSON configuration for '\_source':

```
_source
  buildNum: 15554 defaultIndex: AXZezppBb3bve8TCpy2tc _id: 5.6.3 _type: config _index: .kibana _score: 1
  title: "notExpandable: true fields: [{"name": "_id", "type": "string", "count": 0, "scripted": false, "searchable": true, "aggregatable": false, "readFromDocValues": false}, {"name": "_index", "type": "string", "count": 0, "scripted": false, "searchable": true, "aggregatable": true, "readFromDocValues": false}, {"name": "_score", "type": "number", "count": 0, "scripted": false, "searchable": false, "aggregatable": false, "readFromDocValues": false}, {"name": "_source", "type": "source", "count": 0, "scripted": false, "searchable": false, "aggregatable": false, "readFromDocValues": false}, {"name": "_type", "type": "string", "count": 0, "scripted": false, "searchable": true, "aggregatable": false, "readFromDocValues": false}], _index: ".monitoring-alerts", _score: 1
  metadata.severity: 1,100 metadata.cluster_uuid: tEyI3NfpTTGjf4mRpI6A metadata.version_created: 5050199 metadata.watch: elasticsearch_cluster_status metadata.link: @elasticsearch/_indices metadata.alert_index: .monitoring-alerts-6 metadata.type: monitoring update_timestamp: December 13th 2020, 20:13:06.751 prefix: Elasticsearch cluster status is yellow. message: Allocate missing replica shards. timestamp: December 13th 2020, 20:10:07.023 _id: tEyI3NfpTTGjf4mRpI6A_elasticsearch_cluster_status _type: doc _index: .monitoring-alerts-6 _score: 1
  cluster_uuid: tEyI3NfpTTGjf4mRpI6A timestamp: December 13th 2020, 20:10:24.773 type: cluster_stats source_node.uuid: JaEjsE71RnSI4Q321PanPw source_node.host: 10.0.1.8 source_node.transport_address: 10.0.1.8:9300 source_node.ip: 10.0.1.8 source_node.name: JaEjsE7 source_node.attributes.ml.max_open_jobs: 10 source_node.attributes.ml.enabled: true cluster_name: docker-cluster version: 5.6 license.status: active license.uid: ee69b88e-f783-42cf-a53f-72e7759fd44a license.type: trial license.issue_date: 2020-12-14T01:08:59.9982 license.issue_date_in_millis: 1607908139998 license.expiry_date: 2021-01-13T01:08:59.9982 license.expiry_date_in_millis: 1610500139998
  cluster_uuid: tEyI3NfpTTGjf4mRpI6A timestamp: December 13th 2020, 20:10:24.778 type: index_recovery source_node.uuid: JaEjsE71RnSI4Q321PanPw source_node.host: 10.0.1.8 source_node.transport_address: 10.0.1.8:9300 source_node.ip: 10.0.1.8 source_node.name: JaEjsE7 source_node.attributes.ml.max_open_jobs: 10 source_node.attributes.ml.enabled: true index_recovery.shards: { "index_name": ".triggered_watches", "id": 0, "type": "EMPTY_STORE", "stage": "DONE", "primary": true, "start_time_in_millis": 1607908207036, "stop_time_in_millis": 16079082070705, "total_time_in_millis": 38, "source": {}, "target": { "id": "JaEjsE71RnSI4Q321PanPw", "host": "10.0.1.8", "transport_address": "10.0.1.8" } }
  cluster_uuid: tEyI3NfpTTGjf4mRpI6A timestamp: December 13th 2020, 20:10:24.778 type: node_stats source_node.uuid: JaEjsE71RnSI4Q321PanPw source_node.host: 10.0.1.8 source_node.transport_address: 10.0.1.8:9300 source_node.ip: 10.0.1.8 source_node.name: JaEjsE7 source_node.attributes.ml.max_open_jobs: 10 source_node.attributes.ml.enabled: true node_stats.node_id: JaEjsE71RnSI4Q321PanPw node_stats.node_master: true node_stats.blockall: false node_stats.indices.docs.count: 72 node_stats.indices.store.size_in_bytes: 479,885 node_stats.indices.store.throttle_time_in_millis: 0 node_stats.indices.indexing.index_total: 99 node_stats.indices.indexing.index_time_in_millis: 765
```

## Final Project – Twitter Application

### HPA

The Horizontal Pod Autoscaler automatically scales the number of Pods in a replication controller, deployment, replica set or stateful set based on observed CPU utilization (or, with custom metrics support, on some other application-provided metrics). Note that Horizontal Pod Autoscaling does not apply to objects that can't be scaled, for example, DaemonSets.



```
apiVersion: autoscaling/v1
kind: HorizontalPodAutoscaler
metadata:
  name: nginx
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: nginx
  minReplicas: 1
  maxReplicas: 10
  targetCPUUtilizationPercentage: 5
```

```
csye7220-vm@ubuntu:~/chatter/infrastructure/efk$ kubectl get hpa
NAME      REFERENCE      TARGETS      MINPODS      MAXPODS      REPLICAS      AGE
nginx    Deployment/nginx  1%/5%       1           10          1           4h46m
csye7220-vm@ubuntu:~/chatter/infrastructure/efk$
```