Covid19 Data Visualization

CONTENTS

---

1. ABSTRACT

The recent outbreak of COVID-19 has taken the world by surprise, forcing lockdown and straining public health care systems. COVID-19 is known to be a highly infectious virus, and infected individuals do not initially exhibit symptoms, while some remain asymptomatic.Our main of the project was to make awareness among the people by making Covid19 Data Visualization web app. By visualizing the data using graphical representation of information and data by using visual elements like graph, charts and maps. Data visualization help us to better understand the trend. The web app is build on Angular 11 and data is coming from the API. The data is updated within an interval of 15-30 minutes. The web app show visualization of covid19 disease data

2 INTRODUCTION

2.1 Overview

Everyone is aware of Covid19 disease One of the most and important times for people and countries who are fighting with this virus. So this is my small initiative to bring awareness among people by showing them statistics of confirmed/active cases of all the states and districts of India through my angular application where data is updated every 15–30 minutes. Our team has made an angular app named **Covid19 Data Visualization**

2.2 Features

- Real Time data of the number of cases of COVID-19
- Shows confirmed active cases, recovered cases, and number of deaths
- Shows data from all affected states
- Shows data from the affected districts of states
- Show the India Map which has most affected states
- Show the bar and pie charts for better understanding of data
- Data is updated every 15–30 minutes.

2.3 Why Devops ?

DevOps describes a culture and set of processes that bring development and operations teams together to complete software development. It allows organizations to create and improve products at a faster pace than they can with traditional software development approaches. And, it’s gaining popularity at a rapid rate. DevOps tools consist of configuration management, test and build systems, application deployment, version control and monitoring tools. Continuous integration, continuous delivery and continuous deployment require different tools.

2.3.1. DevOps Features

- Shorter Development Cycles, Faster Innovation
- Reduced Deployment Failures, Rollbacks, and Time to Recover
- Improved Communication and Collaboration
- Increased Efficiencies
- Reduced Costs and IT Headcount

3. SYSTEM CONFIGURATION

3.1 Operating System

Ubuntu 18.04.04 Bionic Beaver.

3.2 CPU and RAM

4 core processor and RAM 8 GB

3.3 Language

- Angular 11
- Typescript
- Bootstrap

3.4 Kernel Version

Linux Machine 5.4.0-66-generic

3.5 Database

we have used the public data API which provide access to all of our data at a national and state level .The data is coming in JSON format links of API used are

- https://api.covid19india.org/data.json
- https://api.covid19india.org/state_district_wise.json

3.6 DevOps Tools

- Source Control Management - GitHub
- Continuous Integration - Jenkins
- Containerization - Docker
- Continuous deployment - Ansible
- Monitoring - ELK Stack (Elastic Search, Logstash, Kibana)

4. SOFTWARE DEVELOPMENT LIFE CYCLE

4.1 Installation

Install Node.js

Node.js is an open-source server environment that uses JavaScript on the server.Node.js lets developers use JavaScript to develop wide variety of applications like network applications, command line tools, web API, web applications. We need nodejs for dev tooling (like local web server with live reloading features) and dev experience, we do not need nodejs to run react in production.

```
curl -sL https://deb.nodesource.com/setup_12.x | sudo -E bash -
sudo apt-get install -y nodejs
```

Update NPM

Npm stands for node package manager, it is a dependency management tool for javascript applications. This tool will help to install the libraries and other tools to support angular development

Normally, NPM will be installed with the Node.js itself. However, we can update it to the latest version using the below command.

```
sudo npm install npm@latest -g
```

Install Angular CLI

```
sudo npm install -g @angular/cl
```

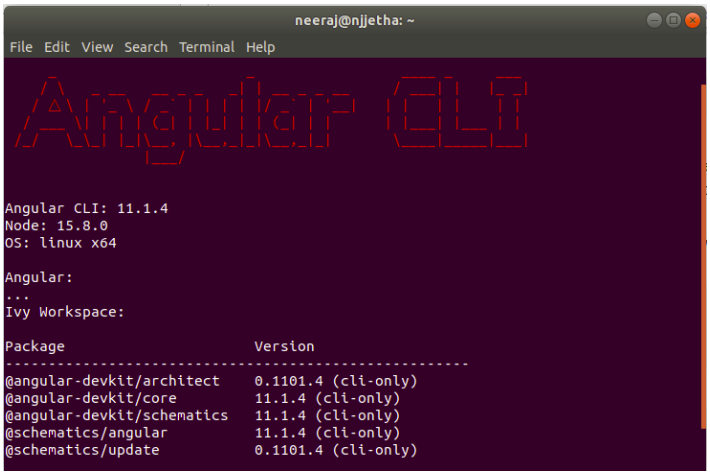


Figure 1 Angular CLI version

Creating new project

```
ng new covid19
```

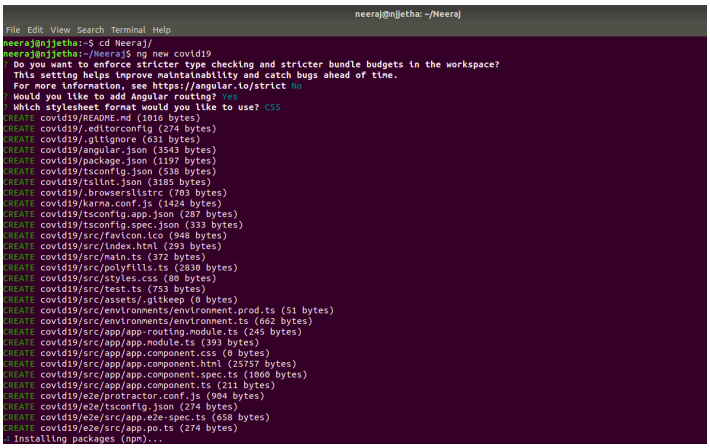


Figure 2 Creating new project

Commands used to generate components

```
ng g c <name>
```

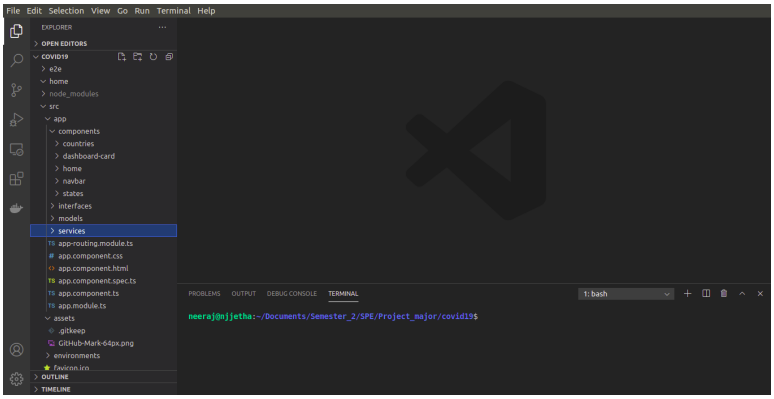


Figure 3 creating all the components

Commands to generate interface

```
ng g interface <name> <type>
```

Commands to generate service

```
ng g service <name>
```

4.2 Tools used in Project

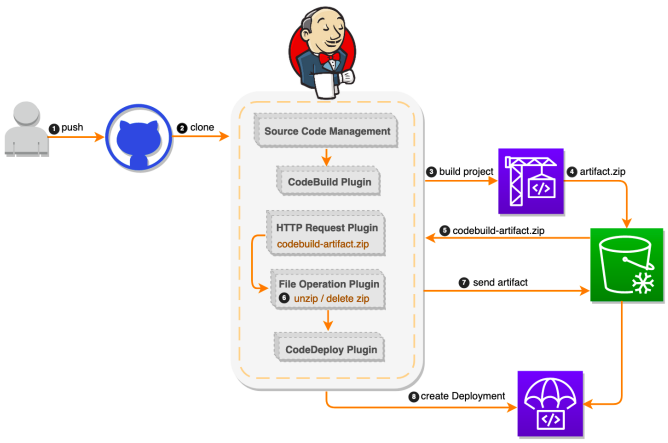
Github:- A Source Code Management (SCM) is a software tool used by programmers to manage the source codes. For our project every team member would clone the repository from github. Create a different branch locally on their system and then merge it with master after pulling the latest code from git, resolving any conflicts and then push the changes to git.

- **git clone <repository url>**- This command copies the entire data on the git url
- **git checkout -b <branch_name>**- This command creates a new branch with the name as in 'branch_name'
- **git add <changed files>**- This command adds changes in the working directory to the staging area
- **git commit -m "message while committing"**- This command is used to save your changes to the local repository with -m used to provide a concise description that helps your teammates (and yourself) understand what happened.
- **git checkout master**- This command switches to master branch
- **git pull**- This command is used to update the local version of a repository from a remote.
- **git merge <branch_name>** - This command is used to integrate changes from another branch.
- **git push**- This command will push all the latest code to the repository

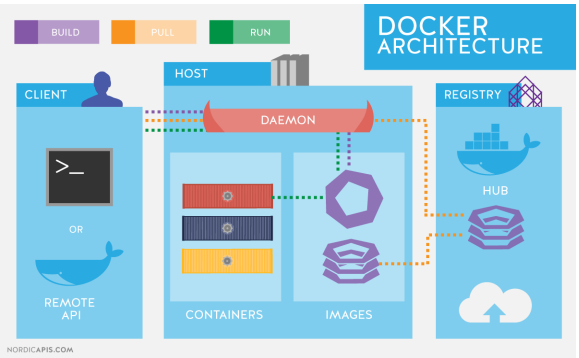
```
commit 2c6d6592247d1f8f69fc177fb45cefa6fe57b344 (HEAD -> master, origin/master)
Author: njjetha <neeraj.jetha@iiitb.org>
Date:   Wed Apr 28 00:38:42 2021 +0530

    Frontend improved

commit cbfc9ec03f4f23774bbdc6886a78d962fa9c1e2e
Author: njjetha <neeraj.jetha@iiitb.org>
Date:   Tue Apr 27 16:07:34 2021 +0530

    move the dockerfile

commit 5c1ef786743399176f8631a46a0bf54eb3aba9aa
Author: njjetha <neeraj.jetha@iiitb.org>
Date:   Tue Apr 27 15:47:10 2021 +0530

    added docker file

commit 6d4fe80a151e3d224596cac65220530f7a28017a
Author: njjetha <neeraj.jetha@iiitb.org>
Date:   Tue Apr 27 00:40:12 2021 +0530

    improvement in cards

commit bdc2fbec609f034cbd6f730981821f79816b353e
Merge: 9d01e47 e183a1c
Author: njjetha <neeraj.jetha@iiitb.org>
Date:   Sat Apr 17 08:29:37 2021 +0530

    Merge branch 'master' of https://github.com/njjetha/Covid19-Data-Visualization

    Geochart added

commit 9d01e4777863cc26d7cfebb95c45dc8d0b8ec309
Author: njjetha <neeraj.jetha@iiitb.org>
Date:   Sat Apr 17 08:29:29 2021 +0530
```

Figure 4 git logs

Jenkins :- Jenkins is a powerful application that allows continuous integration and continuous delivery of projects, regardless of the platform you are working on. It is a free source that can handle any kind of build or continuous integration. You can integrate Jenkins with a number of testing and deployment technologies.

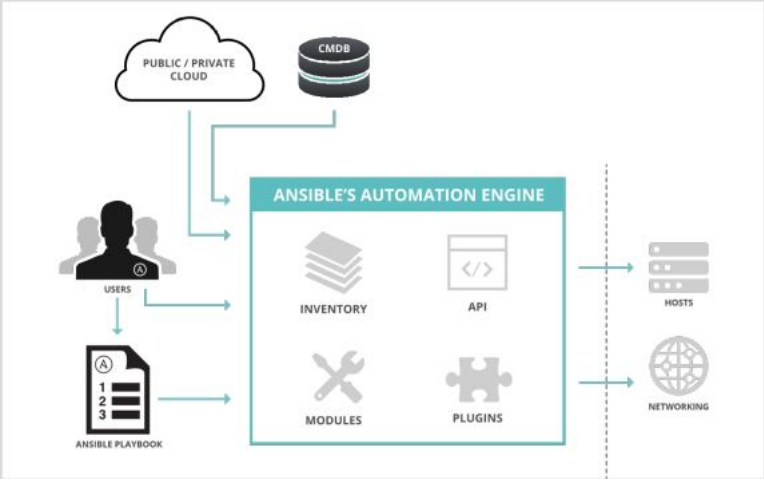


Docker: Docker is a tool designed to make it easier to create, deploy, and run applications by using containers. Containers allow a developer to package up an application with all of the parts it needs, such as libraries and other dependencies, and deploy it as one package

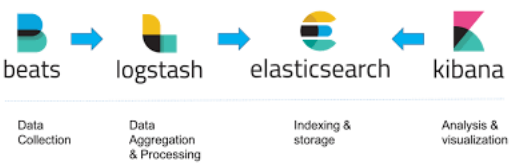


Ansible :- Open Source automation platform. Automation Engine that runs ansible playbooks, playbooks are defined tasks, where we define environments and workflows. There are two types of machines in the Ansible architecture: control nodes and managed hosts. Ansible is installed and run from a control node, and this machine also has copies of your Ansible project files.

Each task runs a module, a small piece of code (written in Python, PowerShell, or some other language), with specific arguments. Each module is essentially a tool in your toolkit. Ansible ships with hundreds of useful modules that can perform a wide variety of automation tasks. They can act on system files, install software, or make API calls.



ELK :- "ELK" is the acronym for three open source projects: Elasticsearch, Logstash, and Kibana. Elasticsearch is a search and analytics engine. Logstash is a server-side data processing pipeline that ingests data from multiple sources simultaneously, transforms it, and then sends it to a "stash" like Elasticsearch. Kibana lets users visualize data with charts and graphs in Elasticsearch.



4.3 CI/CD Pipeline Screenshots

Pipeline Script

```
pipeline {
    agent any
    environment{
        imageName=""
    }
    stages {
        stage('git cloning') {
            steps {
                script{
                    git 'https://github.com/njjetha/Covid19-Data-Visualization'
                }
            }


         stage('Docker Build to image') {
            steps {
                script{
                    imageName=docker.build "njjetha/covid19:latest"
                }
            }

         stage('Push Docker image') {
            steps {
                script{
                    docker.withRegistry('','docker-jenkins'){
                        imageName.push()
                    }
                }
            }
        }


        stage('Ansible Pull Docker image') {
            steps {
                ansiblePlaybook becomeUser: null, colorized: true, disableHostKeyChecking: true,
                installation:'Ansible',inventory: 'remote-server', playbook: 'covid19-playbook.yml', sudoUser: null
            }
        }

    }
}
```

Stage: 1 →Git Cloning

In this stage the code is cloned from the github repo

Stage: 2 → Docker Build to image

In this stage we create the docker image where we first install all the dependencies required to run the docker image

Step 3: → Push Docker image

In this stage we push the docker image to docker hub by providing the credential of docker hub

Step 4: → Pulling Docker image

In this stage we pull the docker image on the local host using ansible.

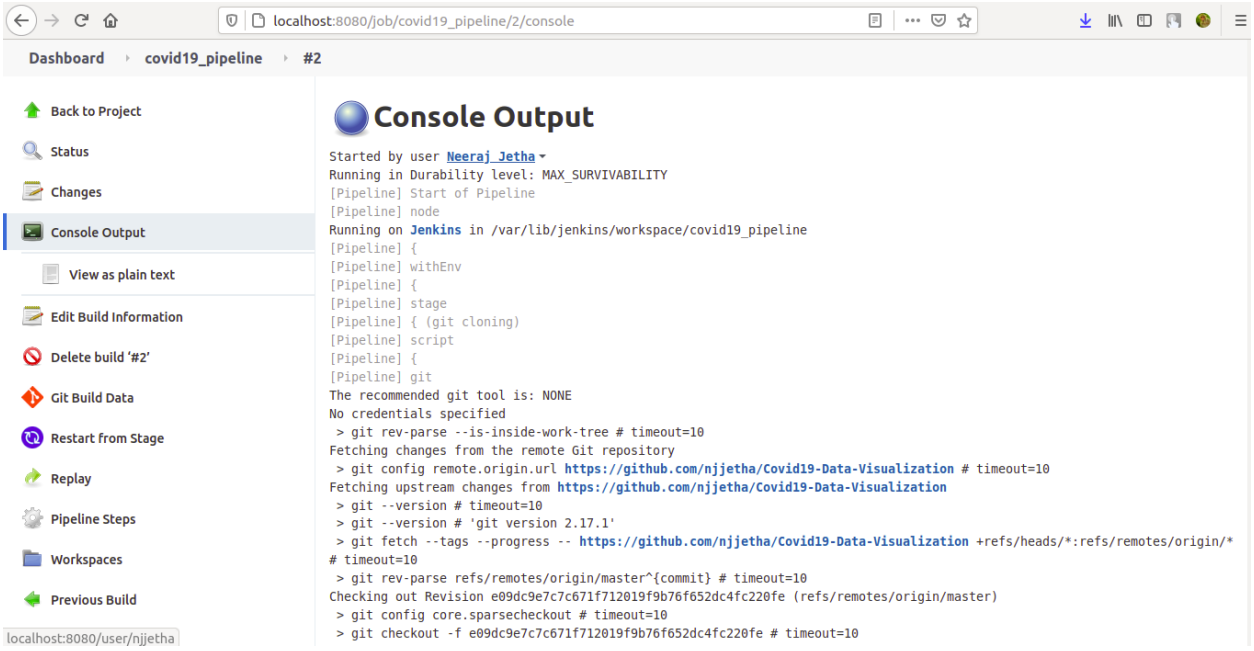


Figure 5 Git Cloning

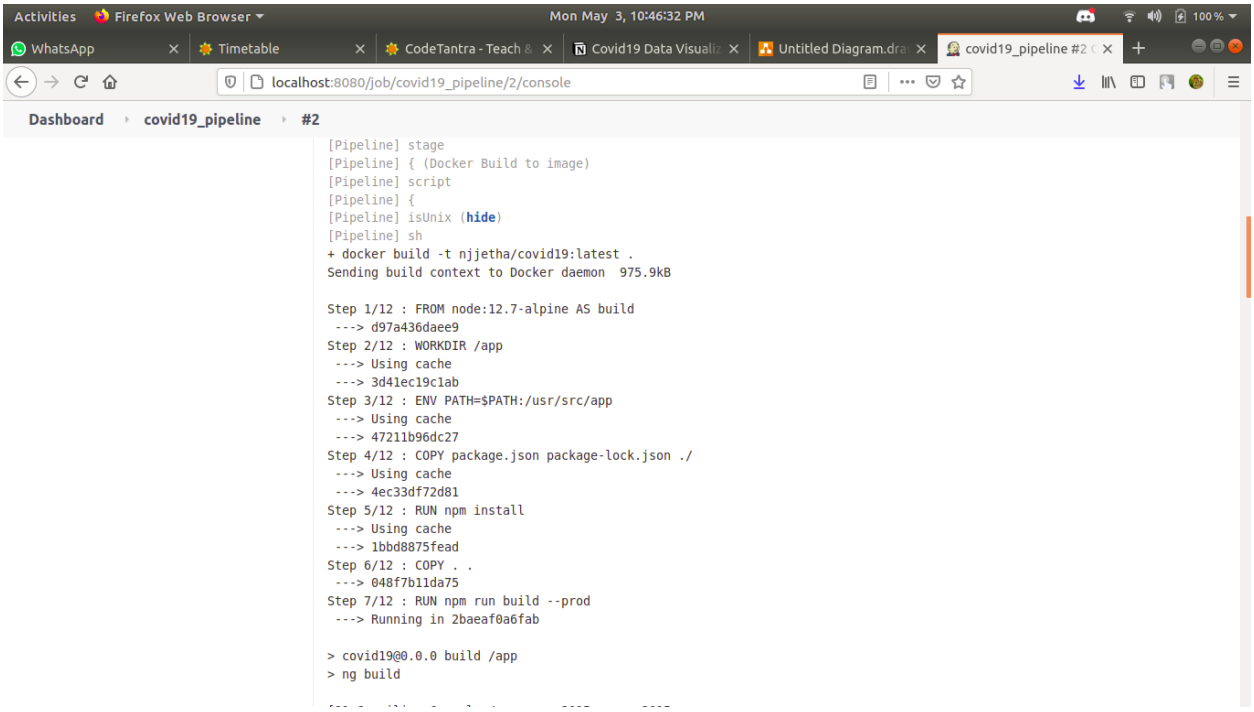


Figure 6 Building Docker image

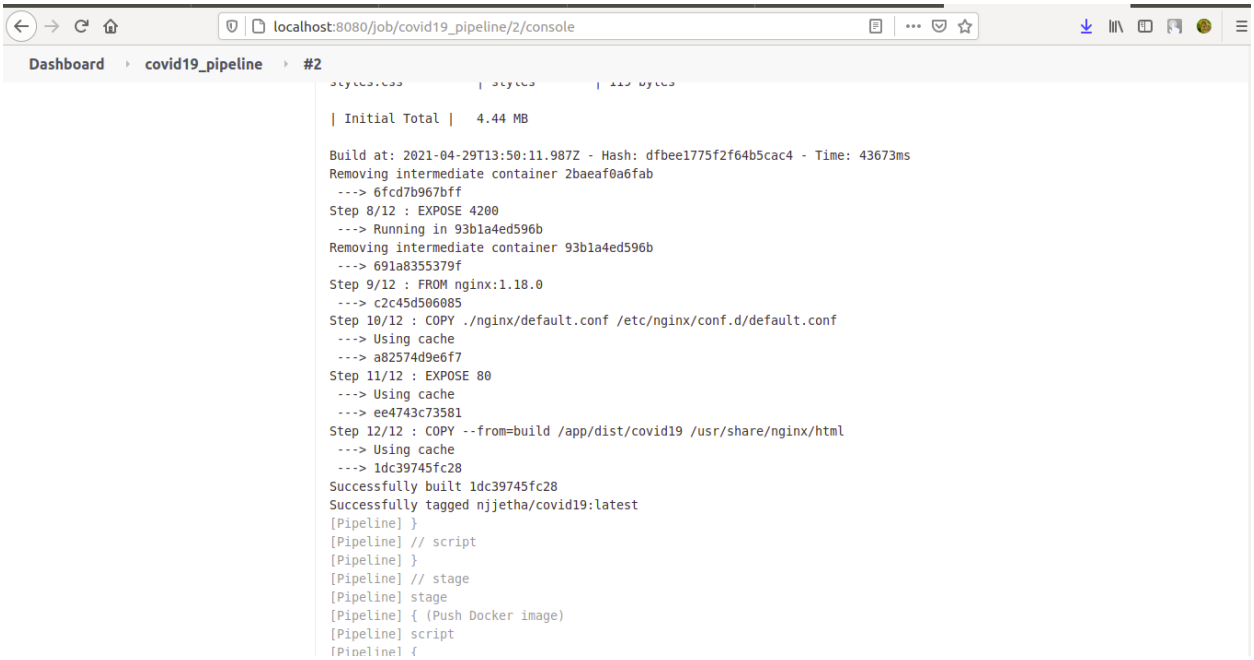


Figure 7 Pushing Docker image

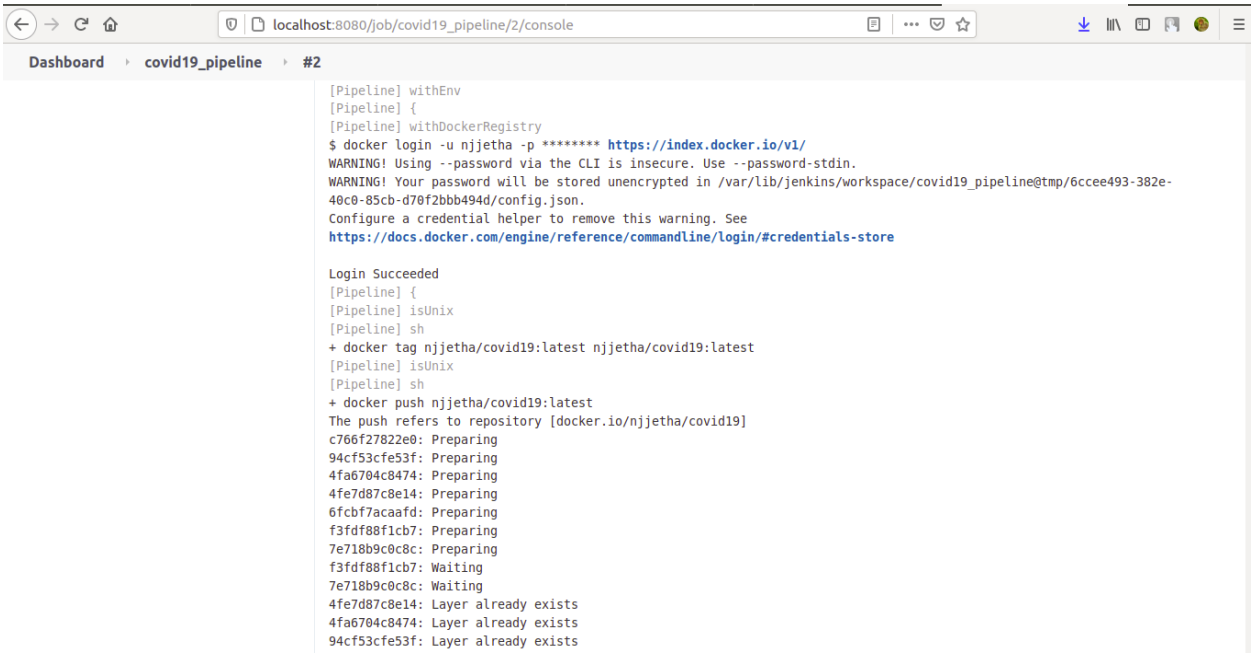


Figure 8 Docker Hub login successful and pushing image

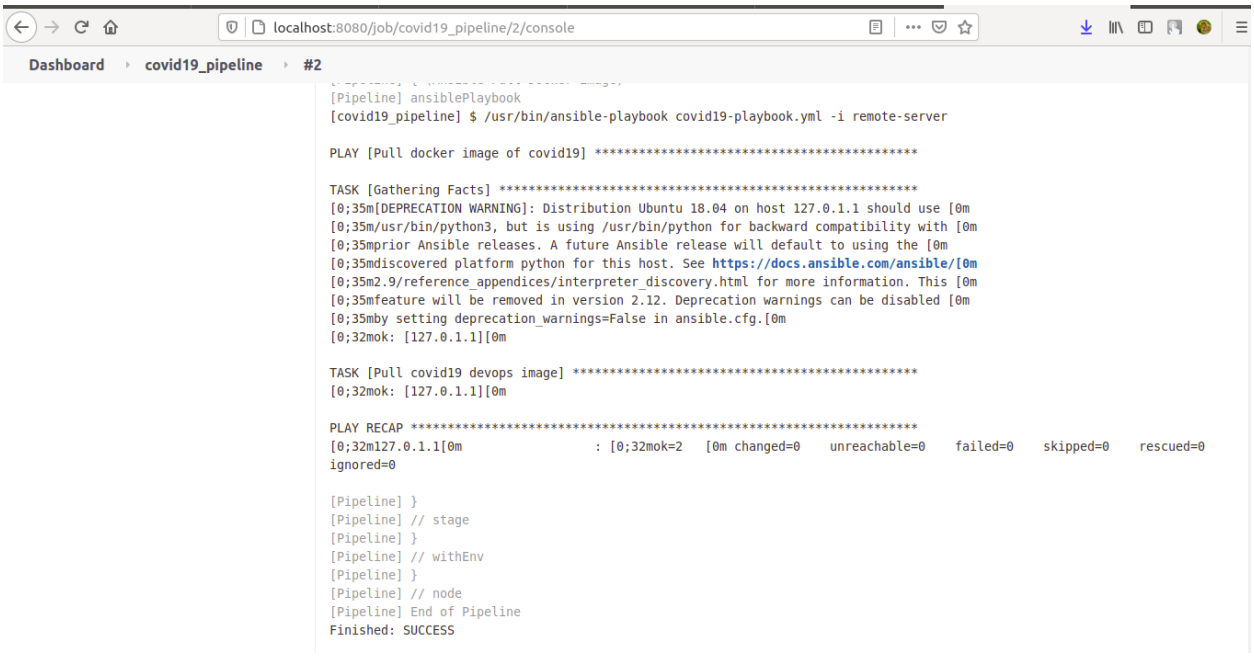


Figure 9 Docker image pulling through Ansible

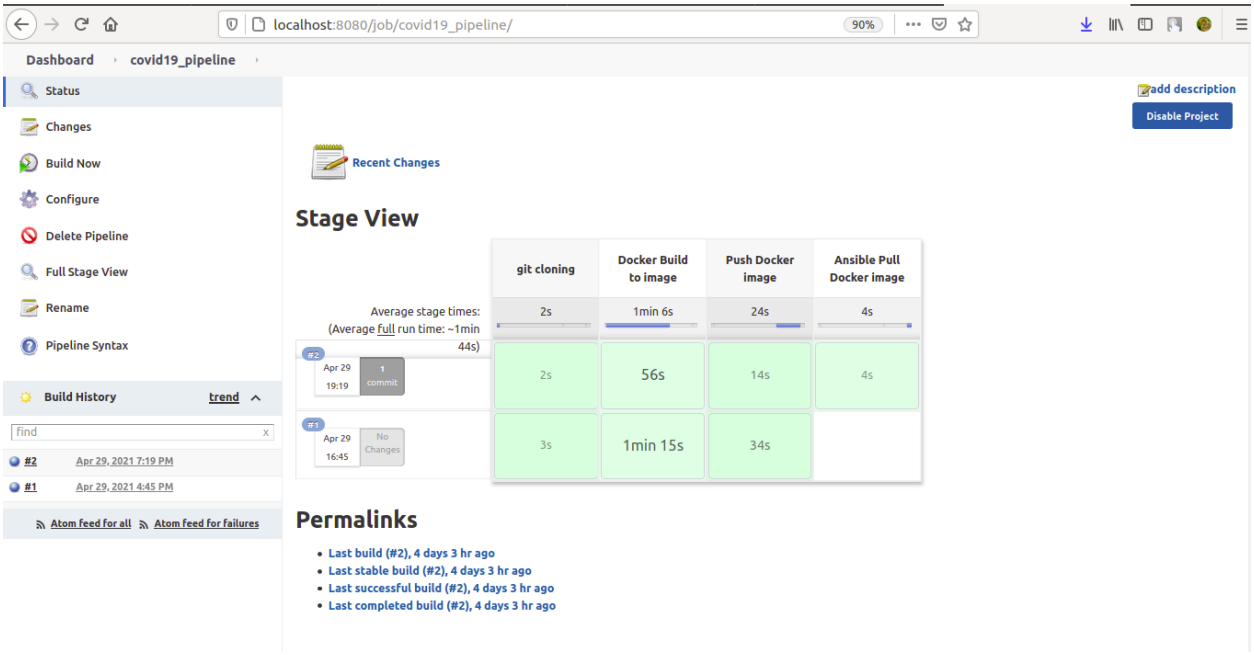


Figure 10 Stages of Pipeline

- **jasmine-core.** Jasmine is the framework we are going to use to create our tests. It has a bunch of functionalities to allow us the write different kinds of tests.
- **karma.** Karma is a task runner for our tests. It uses a configuration file in order to set the startup file, the reporters, the testing framework, the browser among other things.

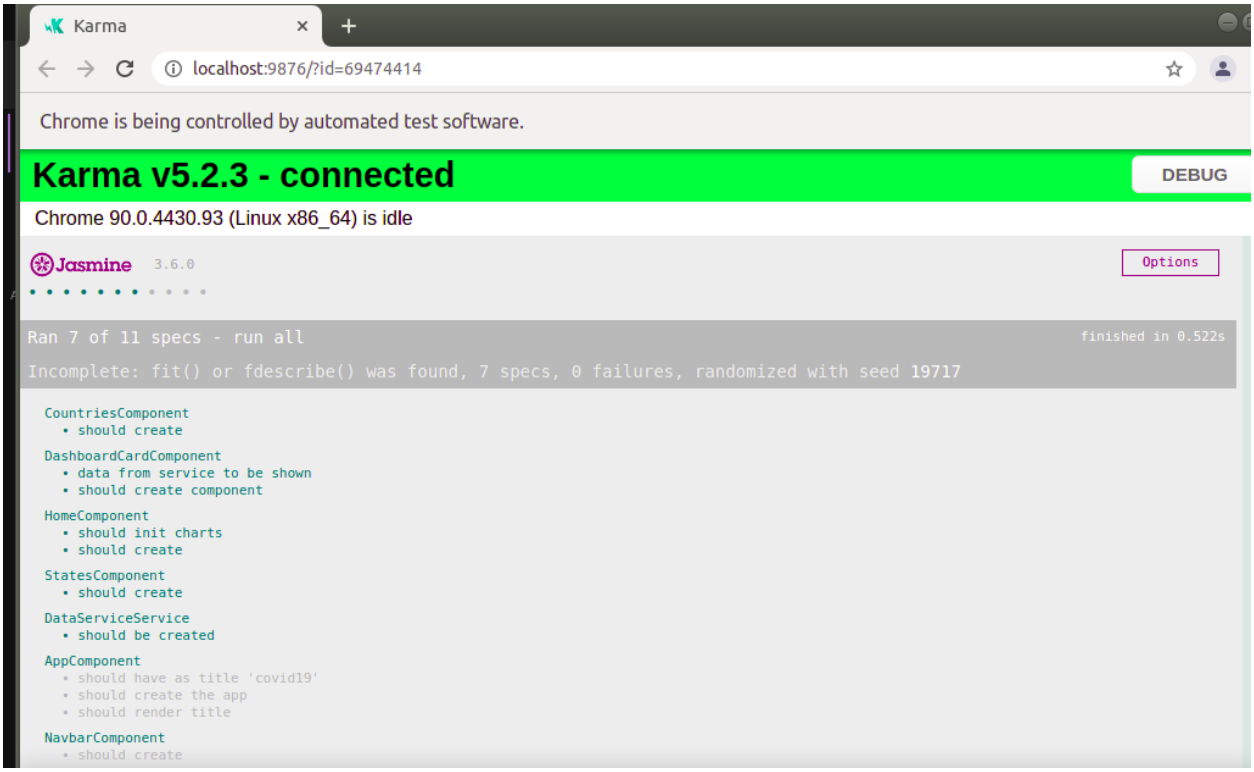


Figure 11 Testing

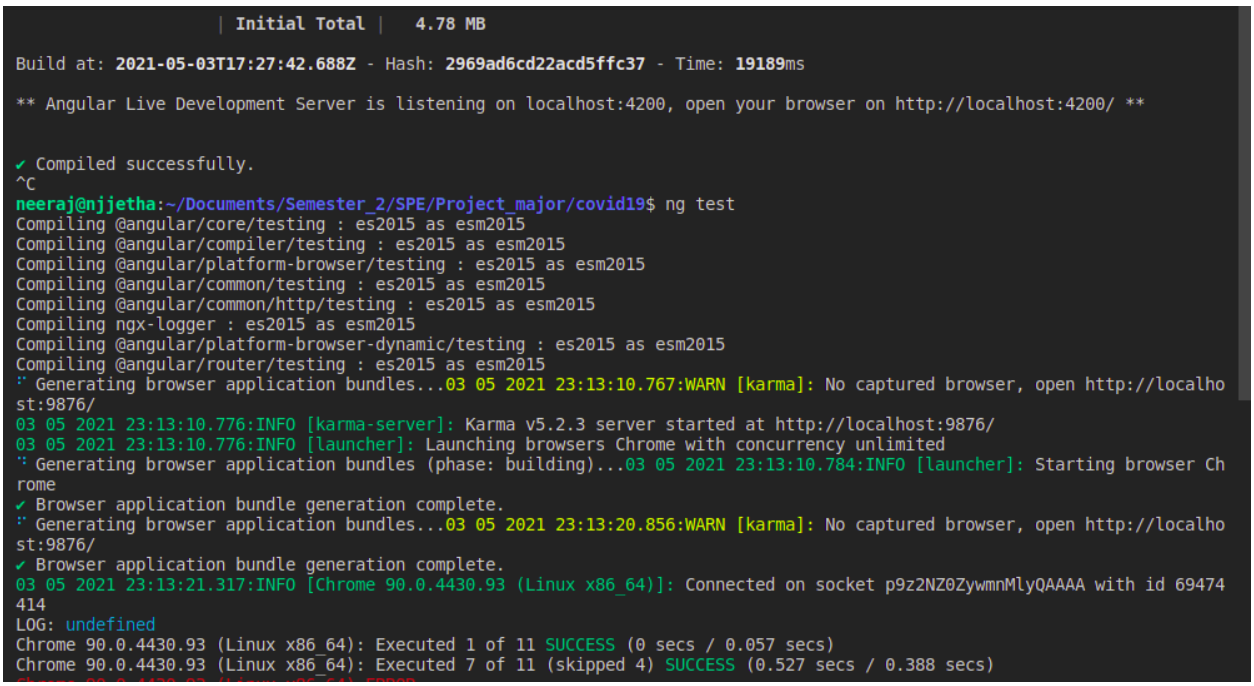


Figure 12 Testing file running

NGX Logger is a simple logging module for angular (currently supports angular 6+). It offers the following features:

- Pretty printing to the console
- Enable logging based on the level specified

- Control log level based on the current environment
- Send log messages to server via HTTP for centralised logging
- Indicates log location and line number

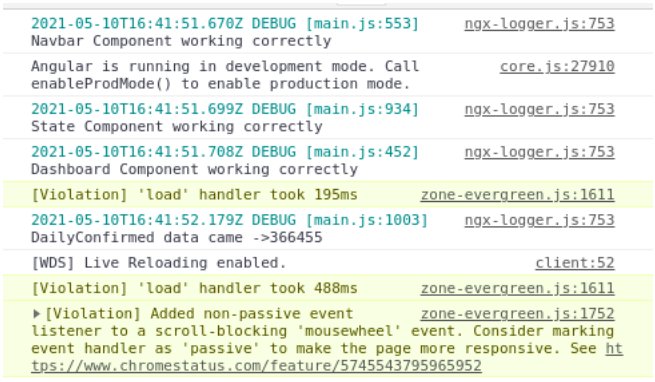


Figure 12 Logs at browser

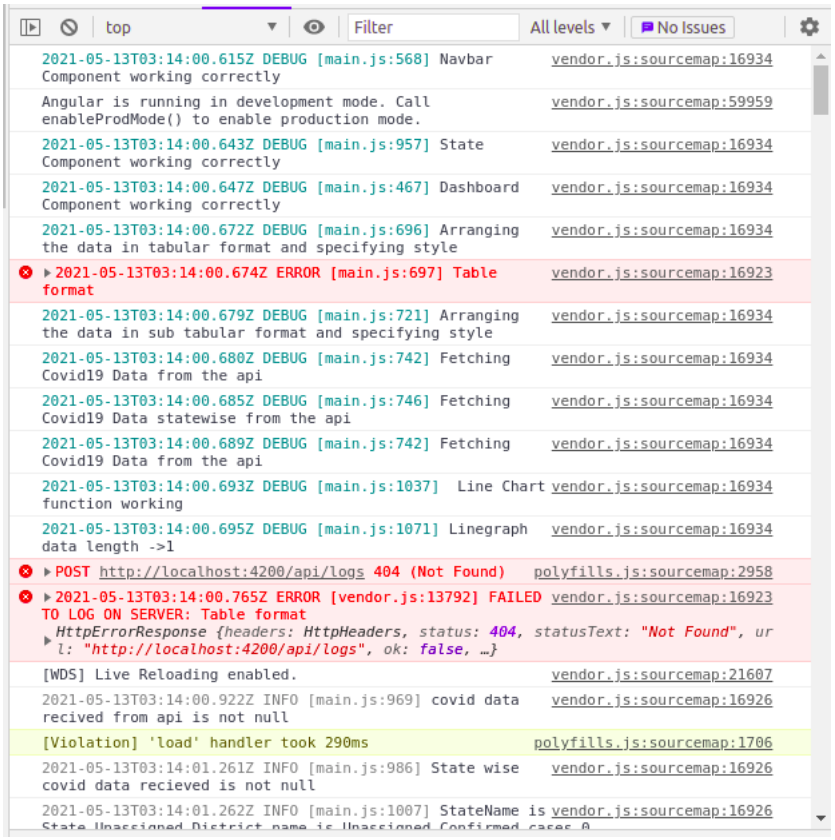


Figure 13 Logs at different level are of different color

Figure 14 Logs at different level are of different color

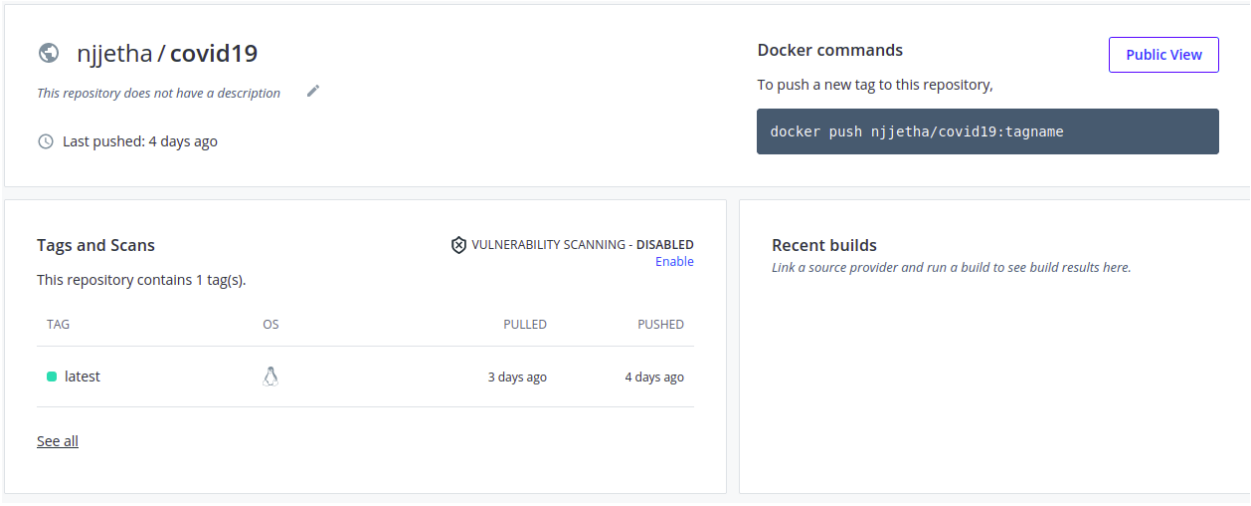


Figure 15 Docker Hub image

A **Dockerfile** is a text file that Docker reads in from top to bottom. It contains a bunch of instructions which informs Docker HOW the Docker image should get built.

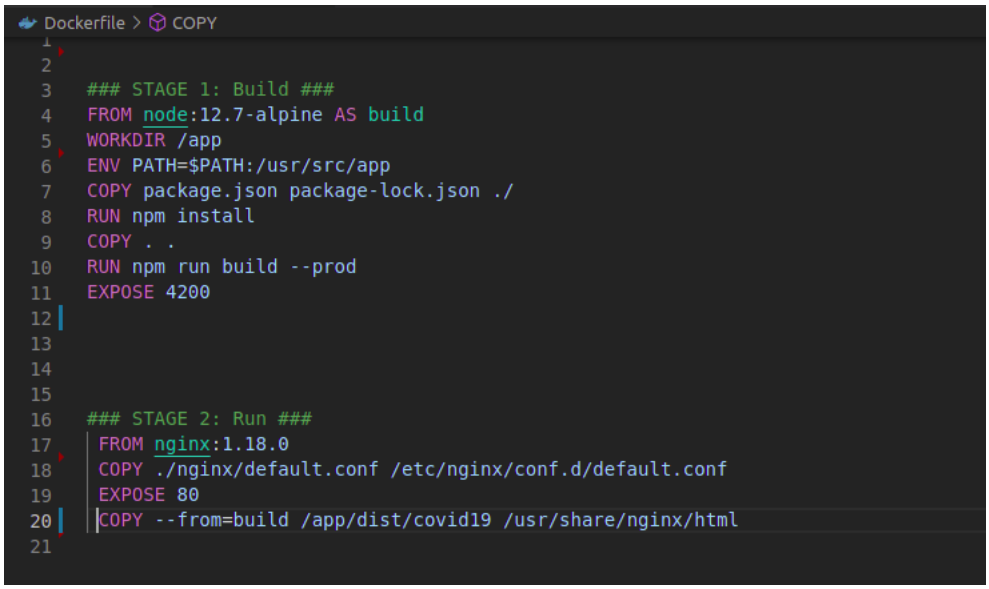


Figure 16 Dockerfile

Playbooks are one of the core features of Ansible and tell Ansible what to execute.Playbooks contain the steps which the user wants to execute on a particular machine. Playbooks are run sequentially. Playbooks are the building blocks for all the use cases of Ansible.

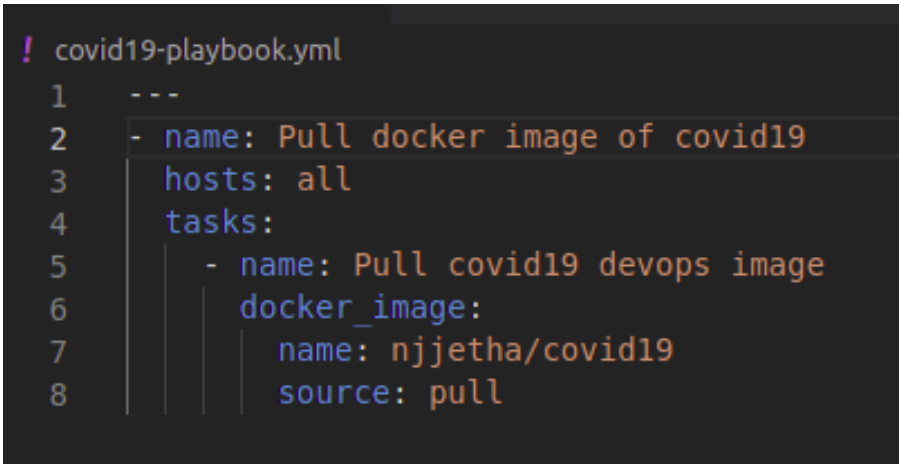


Figure 17 Ansible playbook

Command to run docker image

```
docker run -p 4200:80 --name covid19 njjetha/covid19
```

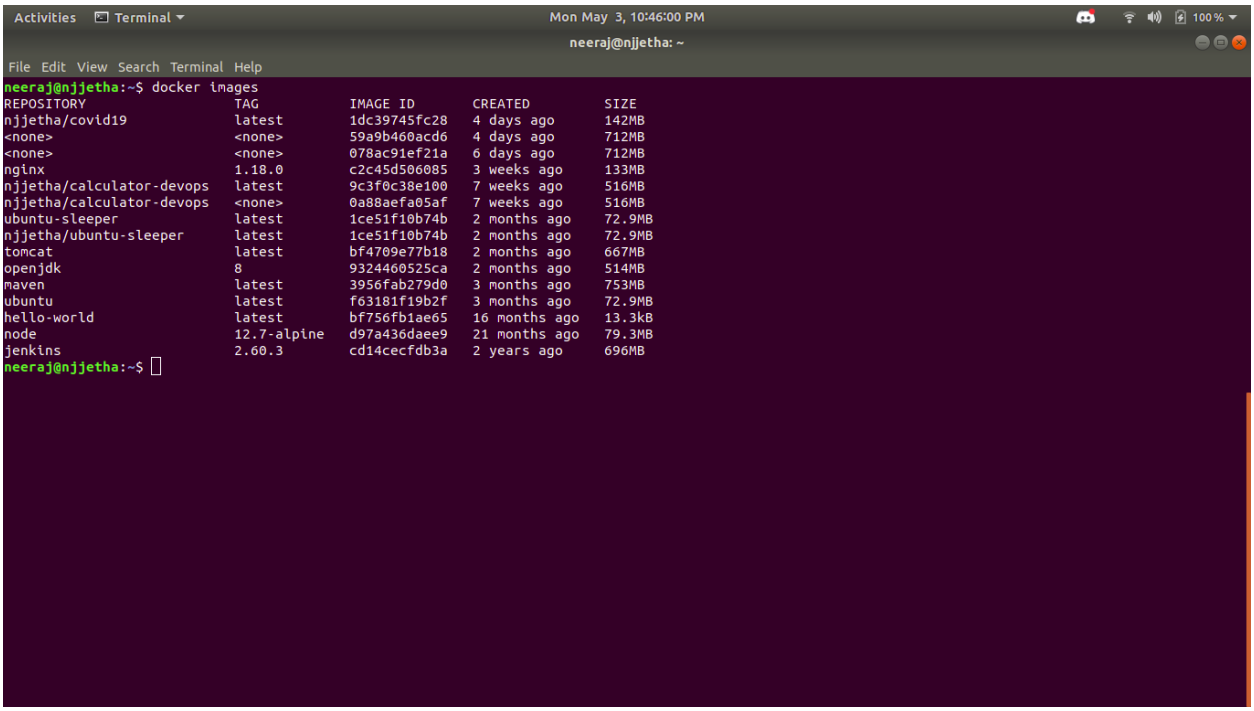


Figure 18 Docker images

5. EXPERIMENTAL SETUP

5.1 Non Functional Requirements

Usability

Being an app centered around visualization and the target audience being the everyday user of the internet,the interface should be graphical and very intuitive to use.The users should be able to figure out the different features and available customization with little use.

Security

The app should rely on anonymous data that is available to the public.It should not reveal the private information of any individuals.

Performance

The app should be responsive.The visualizations to be displayed should be loaded and rendered quickly and should change dynamically based on the users actions.

Scalability

The app should be able to scale to deal with large amount of data without impacting the performance .If the application is modified from client side to server side than the server should be able to handle the users traffic without a drop in performance.

5.2 Functional Requirements

- The application should display time series COVID-19 data using plots.
- It should use a live data source that is updated within an interval of 15-30 minutes.
- The application should show the confirmed, recovered, and death cases separately.
- The graph and map automatically updated while clicking on the bullet of cases

- The state table should be expanded on click on the row for showing district cases .

5.3 Architecture and Workflow

Here in the reference design diagram of our application and how the data flow .Data coming from the public API in the json format

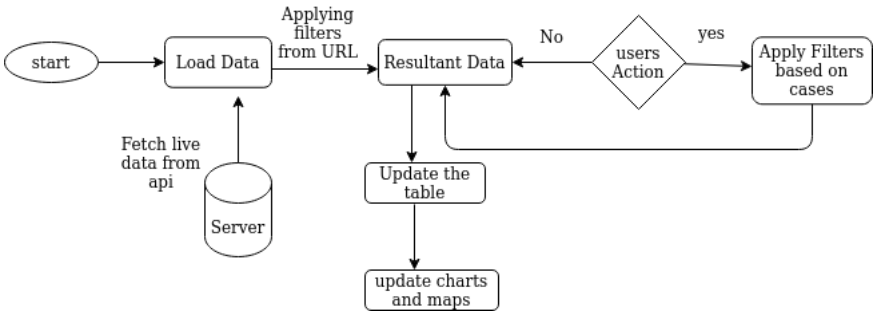


Figure 19 Workflow

5.4 Code Walkthrough

Their are 4 components , one service file and one interface

Components

- **AppComponent**: This is first component which run and call the NavbarComponent
- **NavbarComponent** :- This is component which contains links on the navbar
- **HomeComponent** : This component used to display the charts and maps of different cases
- **DashboardComponent** :- This component contains cards of different cases which is updated within interval of 15-30 minutes
- **stateComponent**: Contains a list of the district for specific States and loaded dynamically On Click of a particular state.
- **dataservice** : It is used for fetching data from api
- **dataInterface** : It is used to define the structure of the data coming from api.

API for fetching data

- **Daily-Indian-Case**: https://api.covid19india.org/data.json
- **State-district-wise** : https://api.covid19india.org/state_district_wise.json

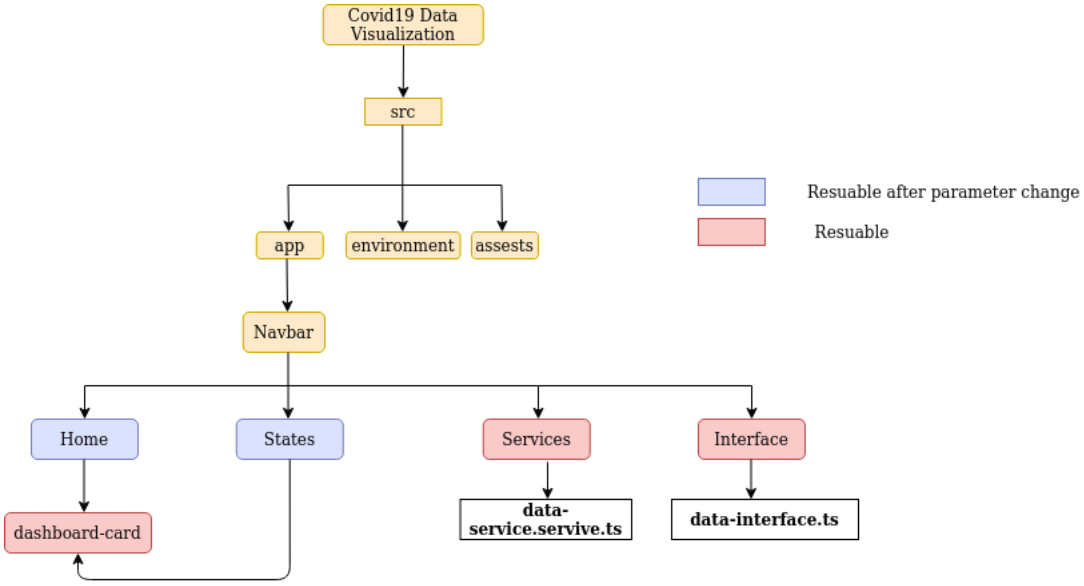


Figure 20 CodeFlow

5.5 Screenshots and code snippets

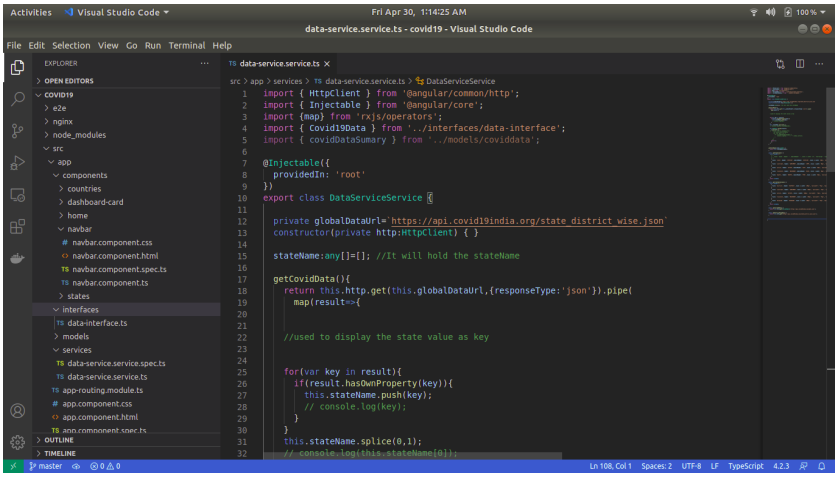


Figure 21 dataservice.service.ts

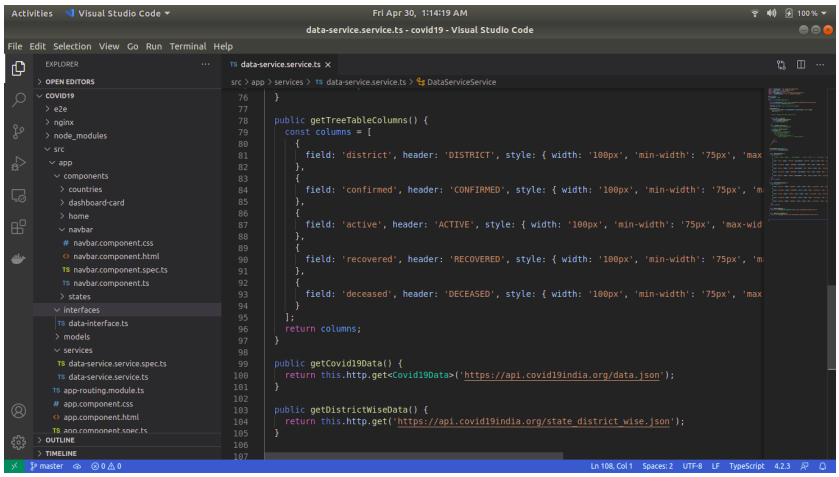


Figure 22 dataservice.service.ts

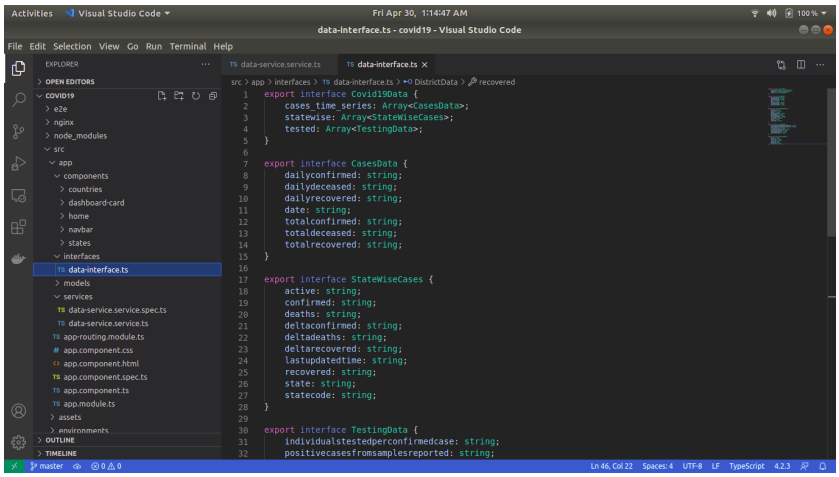


Figure 23 data-interface.ts

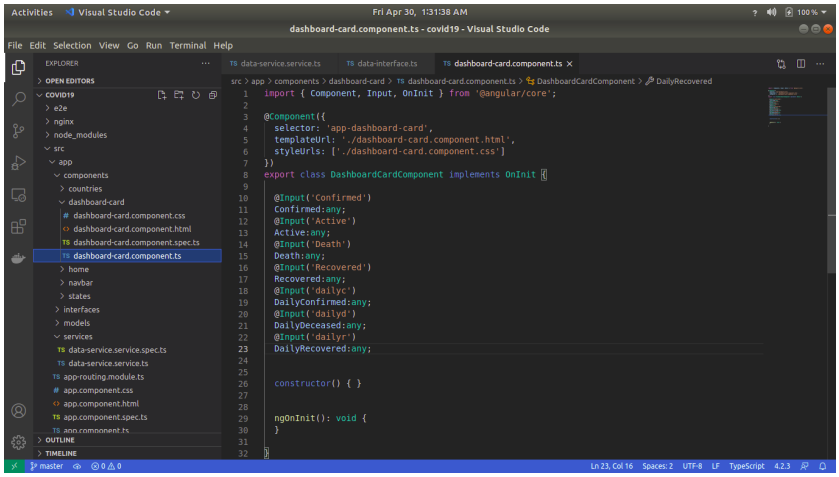


Figure 24 dashboard-card.component.html

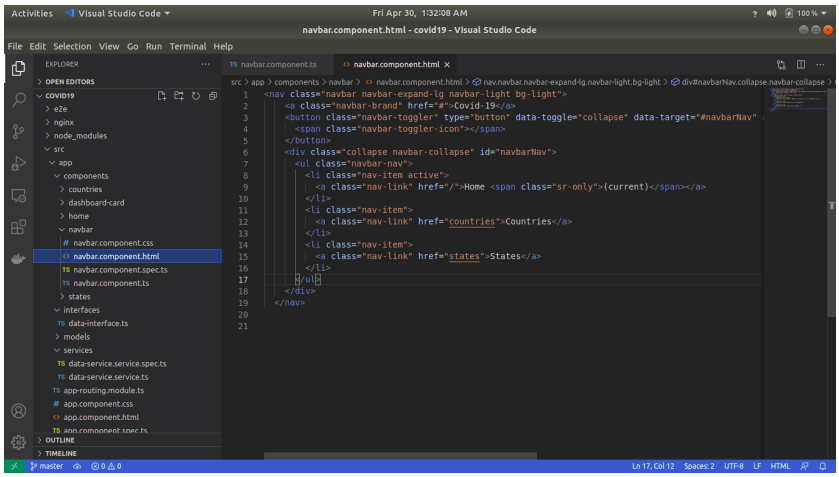


Figure 25 navbar.component.html

Passing the value of cases variable to the dashboard components in Indian format

```
public getDailyData():void{
    this.dataservice.getCovid19Data().subscribe((data:Covid19Data)=>{
      this.covid19Data = data.cases_time_series;
      for(const[key,value]of Object.entries(this.covid19Data)){
        var x:any=value;
        this.DailyConfirmed=parseInt(x.dailyconfirmed);
        this.DailyDeceased=parseInt(x.dailydeceased);
        this.DailyRecovered=parseInt(x.dailyrecovered);
      }
      // converting number to string and this is value passed to dashboard component
      this.DailyConfirmed=this.DailyConfirmed.toLocaleString('en-IN', {maximumFractionDigits:2});
      this.DailyDeceased=this.DailyDeceased.toLocaleString('en-IN', {maximumFractionDigits:2});
      this.DailyRecovered=this.DailyRecovered.toLocaleString('en-IN', {maximumFractionDigits:2});
    });
  }
```

It will fetch the state wise data

```
transformDistrictWiseData(districtWiseData:any): void {
      for (const [key, value] of Object.entries(districtWiseData)) {
          var x:any=value
        for (const [key1, value1] of Object.entries(x)) {
            var y:any=value1
          for (const [key2, value2] of Object.entries(y)) {
              var z:any=value2;
            if (key1 !== 'statecode') {
              this.totalConfirmed=this.totalConfirmed+z.confirmed;
              this.totalDeath+=z['deceased'];
              this.totalRecovered+=z['recovered'];
              this.totalActive+=z['active'];
              this.transformedDistrictWiseData.push({ state: key, district: key2, confirmed: z['confirmed'],active:z['active'],recovered:z[
            }
        }
      }
    }
  }
```

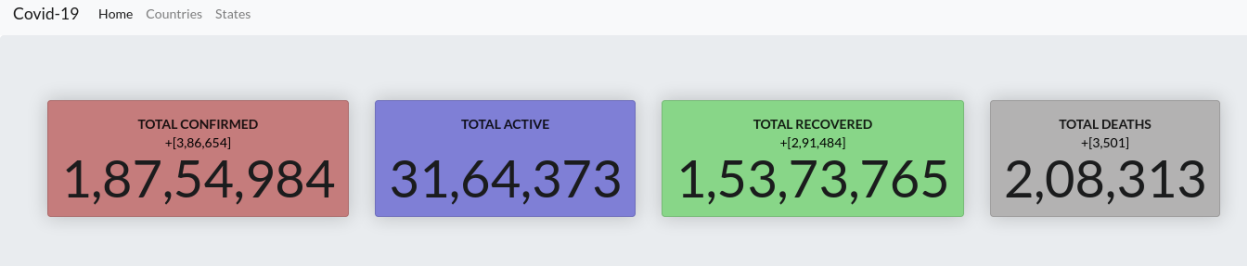


Figure 26 Dashboard-card

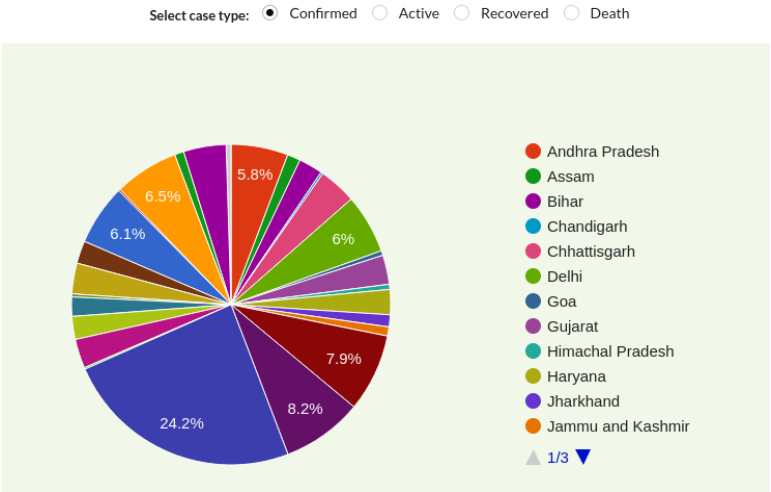


Figure 27 Pie chart of confirmed cases

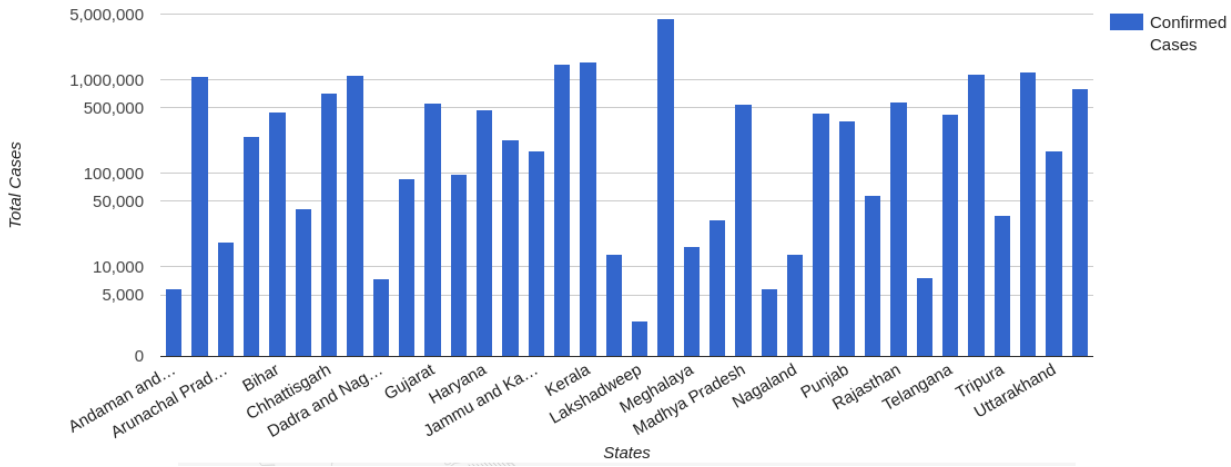


Figure 28 Barcharts of confirmed cases

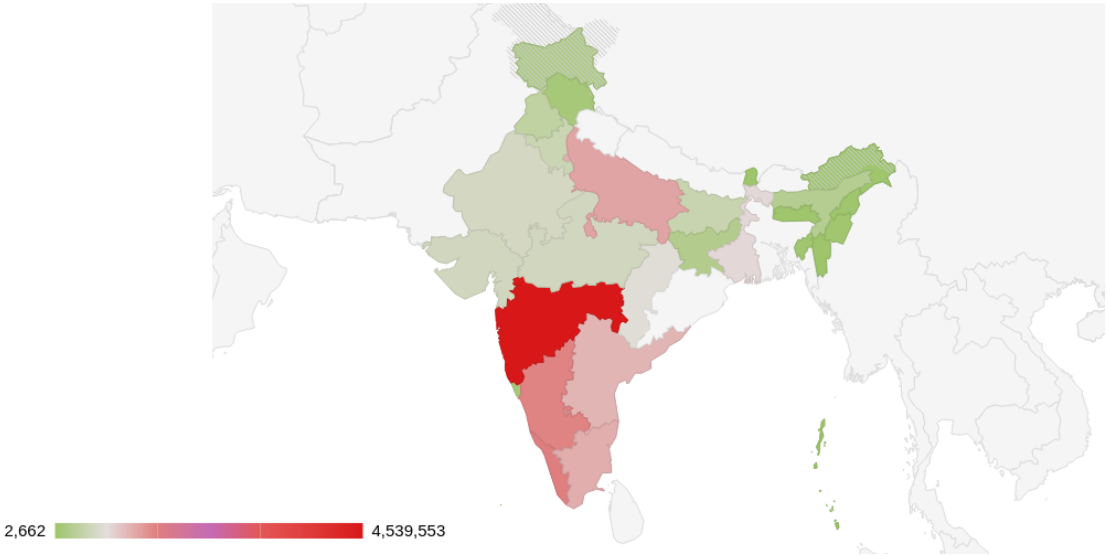


Figure 29 Map of Confirmed cases

STATE DATA

STATE/UT	CONFIRMED	ACTIVE	RECOVERED	DEATHS
Maharashtra	4539553	670301	3799266	67985
Kerala	1533985	284085	1244301	5260
Karnataka	1474846	349496	1110025	15306
Andhra Pradesh	1084336	114158	962250	7928
Tamil Nadu	1148064	112556	1021575	13933
Delhi	1122286	97977	1008537	15772
Uttar Pradesh	1217952	309237	896477	12238
West Bengal	810955	110241	689466	11248
Odisha	435513	53031	380400	2082
Rajasthan	580846	169519	407243	4084
Chhattisgarh	713706	117910	587484	8312
Telangana	427960	76060	349692	2208

Figure 30 State wise cases

STATE/UT	CONFIRMED	ACTIVE	RECOVERED	DEATHS
Maharashtra	4539553	670301	3799266	67985

DISTRICT	CONFIRMED	ACTIVE	RECOVERED	DECEASED
Ahmednagar	167199	23124	142124	1950
Akola	40186	4741	34831	610
Amravati	63299	7243	55184	870
Aurangabad	123533	14649	106925	1945
Beed	53470	10788	41795	878
Bhandara	49737	11309	37961	463
Buldhana	44339	7927	36046	361
Chandrapur	60016	26613	32784	617
Dhule	37535	4050	33042	433
Gadchiroli	19421	4214	15014	184
Gondia	32054	7311	24404	333
Hingoli	13363	1935	11254	174
Jalgaon	117416	13393	102108	1887
Jalna	43713	6848	36233	631
Kolhapur	64809	7803	55215	1788
Latur	70413	12526	56788	1095
Mumbai	644583	67255	562738	13036
Mumbai Suburban	0	0	0	0
Nagpur	413981	80028	328817	5090
Nanded	80628	9360	69682	1578
Nandurbar	34491	7377	26571	542
Nashik	308118	52954	252129	3034
Osmanabad	38456	7606	29957	875
Other State	146	26	0	118
Palghar	86961	16759	68967	1225
Parbhani	35371	10059	24733	568
Pune	833243	104529	719221	9438

Figure 31 District wise cases

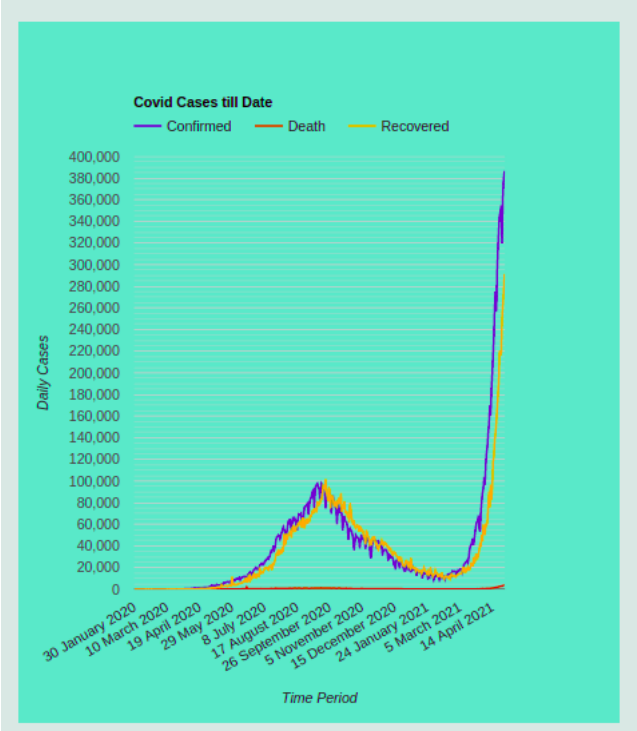


Figure 32 cases till date

6. CONCLUSION AND FUTURE WORK

We have showcased the web application that plots Covid19 time series data using a live data source . It allow user to select the state and view the district data .However more functionality related to state like showing the graph when clicking on the state, adding more information on the column table other than cases like population , number of testing per millions. Later on if public api for vaccination data is available than based on this information more things can be drawn

Here are the key takeaways of this Project

- Working with an HTTP request
- Working with JSON Data (Arrays and Objects)
- Sorting Numeric Data
- Sorting Datatypes.
- Sorting String and Object arrays
- Doing arithmetic Calculation on Object types
- Nesting Table inside Template
- Loading component Dyanamically on Click
- Cross Component communication

REFERENCES

- https://www.npmjs.com/package/ng2-google-charts
- https://angular.io/docs
- https://medium.com/@kamalaujla4vr/display-data-on-indias-geo-chart-in-angular-8-332e6f1b01dd

- https://documenter.getpostman.com/view/10724784/SzYXXKmA?version=latest

LINKS

Github Repo → https://github.com/njjetha/Covid19-Data-Visualization

DockerHub Repo → https://hub.docker.com/repository/docker/njjetha/covid19