



Summer Fellowship Report

On

Web-Development/DevOps

Submitted by

Akshat Pande

Amity University Uttar Pradesh

Under the guidance of

Prof.Kannan M. Moudgalya

Chemical Engineering Department

IIT Bombay

Mentors

Mr. Thomas Stephen Lee

Mr. Rohan Mhatre

June 20, 2020

Acknowledgment

The journey of FOSSEE Summer Fellowship was one of a kind experience and I consider myself privileged to be a part of such a prestigious program. The efficient workflow of the Internship in an online manner under such pandemic conditions was really appreciable. Also, It gave me a chance to get connected with some Industry professionals and mentors from the organization which was a great assistance for me throughout the fellowship, not only helping me if I am stuck but also encouraged to research better alternatives of a solution to carry out an intuitive learning process.

I am grateful to express gratitude and special thanks to Prof. Kannan M Moudgalya, head of FOSSEE Team, for giving us an opportunity to be a part of the project. Also, the successful completion and valuable learnings while completing our task would not have been made possible without my mentors, Mr. Thomas Stephen Lee and Mr. Rohan Mhatre who made this process more fulfilling, rewarding, and fun along with maintaining a perfect balance between work and breaks.

I have always thought of this program as a source of learning and of course a milestone in my career. The technologies used in the program duration were really interesting to explore and added a valuable skill set to my working methodology and projects. I will for sure be consistent with them in the future and wish to connect for a similar opportunity with the organization.

Content

1. Getting Started with Grafana

1.1	Introduction to Grafana	1
1.2	Installation and Setup	2
1.3	Overview of basic features	5

2. Components of Grafana

2.1	Supported Data Sources	7
2.2	Dashboard and Panels	9
2.3	Users and Teams	12
2.4	Explore	13
2.5	Grafana HTTP API	13

3. Contributions

3.1	Tasks and Solutions	14
3.2	Side-Tasks in the process	19
3.3	Problems Faced	19

4. DevOps

4.1	Introduction to concepts	20
4.2	Category of Tools available	22

5. Containerization

5.1	Importance of containers	24
5.2	Container Engines	25
5.3	Implementation of Docker	26

6. CI/CD

6.1	Workflow and Advantages	29
6.2	Travis CI	30
6.2.1	Implementing Travis on Yaksh App	31
6.2.2	Implementing Travis on a Node app	31

7. Orchestration

7.1	Introduction to Kubernetes	32
7.2	Components Involved	33
7.3	Implementing Kubernetes on Yaksh	34

8. Staytus

7.1	Overview	36
7.2	Implementation	37

9. AWDash

7.1	Overview	38
7.2	Implementation	38

8. References		39
----------------------	--	----

Getting Started with Grafana

Introduction to Grafana

We are aware of the importance of analysis and visualization of data in the present scenario where data science and data scientists are one of the most discussed professions and work roles, respectively. The study of data is extensive ranging from various sources to generate patterns and make the technology smart with the help of such methods.

Grafana is one of the Open Source solutions for monitoring and analyzing data with the help of an interactive user interface which is not only easy to use but also implements team structure as an accessibility feature which is a need of each workplace. The software has the capability to deliver on-premises service and connects to almost all the major data sources available around us. It focuses on studying data at various time intervals also known as the time-series analytics which proves to be a rational concept for monitoring system statistics, errors, etc. Grafana extracts metrics and presents them in dashboards with the help of queries and a vast range of visualizations available for the representation of data.

Development as conversed ranges from third-party plug-ins for non listed data sources or adding new features to a dashboard library with several comprehensive dashboards pre-built and import-ready for their use with data source requirements and import details specified on the platform.

The software is free to use and modify as the tag suggests Open Source but it does offer two paid solutions for the end-users including Grafana Enterprise and Grafana Cloud. Here, the enterprise edition provides you with advanced plug-ins and extended whereas the Cloud refers to a compute instance made available to you with Graphite, Prometheus and Loki pre-installed as your head-start to start with monitoring and analysis.

Installation and Setup

The installation of Grafana on CentOS 8 requires:

Grafana-Server which serves the main application

- Disable SELinux by editing the file `/etc/sysconfig/selinux`
 - Change `SELINUX=enforcing` to `SELINUX=disabled`
 - Reboot the system using ``$ sudo reboot``
- Create a repo file at `/etc/yum/repos.d/grafana.repo` and paste the following

```
[grafana]
name=grafana
baseurl=https://packages.grafana.com/oss/rpm
repo_gpgcheck=1
enabled=1
gpgcheck=1
gpgkey=https://packages.grafana.com/gpg.key
sslverify=1
sslcacert=/etc/pki/tls/certs/ca-bundle.crt
```

- Install it with
`$ sudo dnf install grafana`
- Start and Enable the service
 - `$ sudo systemctl start grafana`
 - `$ sudo systemctl enable grafana`
- Opening required port on firewall (3000 in our case)
 - `$ firewall-cmd --zone=public --add-port=3000/tcp --permanent`
 - `$ firewall-cmd --reload`

Adding SSL to domain

- Install Certbot
 - \$ sudo wget <https://dl.eff.org/certbot-auto>
 - \$ sudo mv certbot-auto /usr/local/bin/certbot-auto
 - \$ sudo chown root /usr/local/bin/certbot-auto
 - \$ sudo chmod 0755 /usr/local/bin/certbot-auto
- Stop Nginx service if running with
 - \$ sudo service nginx stop
- Execute the command below to get certificates as well as automatically change the Nginx config
 - \$ sudo certonly -d <domain_name>

Nginx Web Server to reverse proxy to Grafana Server

- Install Nginx
 - \$ sudo dnf install nginx
 - \$ sudo systemctl enable nginx
 - \$ sudo systemctl start nginx
 - \$ sudo firewall-cmd --permanent --add-service=http
 - \$ sudo firewall-cmd --reload
- Test the server
 - \$ curl localhost
 - Or Visit localhost in a browser
- Configuration for nginx
 - Erase everything in '/etc/nginx/conf.d/default.conf' and place a '#'
 - Add config below to '/etc/nginx/conf.d/<domain_name>.conf'

```

server {
    listen      10.0.2.15:80 ;
    server_name akshat.fosseeapps.in;

    return 301 https://akshat.fosseeapps.in$request_uri;

    access_log /var/log/nginx/akshat.fosseeapps.in.access.log main;
    error_log /var/log/nginx/akshat.fosseeapps.in.error.log;
}

server {
    listen      10.0.2.15:443 ssl;
    server_name akshat.fosseeapps.in;

    #ssl on;
    ssl_certificate /etc/letsencrypt/live/akshat.fosseeapps.in/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/akshat.fosseeapps.in/privkey.pem;

    ssl_protocols TLSv1 TLSv1.1 TLSv1.2;
    ssl_ciphers <add cipher>;
    ssl_prefer_server_ciphers on;
    ssl_dhparam /etc/ssl/certs/dhparam.pem;

    access_log /var/log/nginx/akshat.fosseeapps.in.ssl.access.log main;
    error_log /var/log/nginx/akshat.fosseeapps.in.ssl.error.log;

    client_max_body_size 0; # disable any limits to avoid HTTP 413 for large
    image uploads

    location / {
        proxy_pass http://localhost:3000;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-Host $host;
        proxy_set_header X-Forwarded-Server $host;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_read_timeout 5m;
    }
}

```


- Start Nginx again with the command
 - `$ sudo service nginx start`
 - In case an error of `dhparam.pem` file doesn't exist is observed, create one with the command
 - `$ sudo openssl dhparam -out /etc/nginx/ssl/dhparam.pem 2048`
 - Check the HTTP and https port on your web browser if they're working correctly else debug

Enable SELinux for Grafana

- Create Configurations
 - Edit Selinux ``/etc/selinux/config``
Set: SELINUX=enforcing
 - `$ sudo reboot`
 - `$ sudo dnf install policycoreutils-python-utils`
 - `$ grep den /var/log/audit/audit.log | audit2allow -M mygrafana`
- Restart Grafana
 - `$ sudo service restart grafana`

Overview of Basic Features

- **Data Sources**

These refer to the software, databases, and other cloud-based services that collect and monitor data from some specific resources in their region. The data may include system metrics, logs, errors, database metrics, etc. and extracting them from a particular tool made for the purpose is the best use case for maximized flexibility and efficiency as these tools may provide a similar querying procedure or technology-based features built into them. Later these metrics are in turn fetched from Grafana and visualized in a more elaborated manner. This feature is the main module of the software as no matter which tool you use under the hood to keep data intact, you will usually have an option to pull the data in Grafana which makes it act like a wrapper over metrics and log collection tools.

- **Dashboards**

It is a collection of panels representing versatile information with the help of data pulled from data sources. The dashboards module provides users an option to either import a pre-made dashboard using an ID or make a new one, customized according to the needs of the organization. Each entity has its own name, version number which indeed maintains version control and settings for the same.

- **Panels**

Moving down the branches in the dashboard hierarchy we come across a number of panels which as the name suggests are resizable blocks of visualizations that together build up to make a dashboard. Each panel is capable of being converted to a row and also consists of an add visualization button which opens a new dimension to add queries, play with visualizations, and a lot more discussed in later sections.

- **Teams**

Working in teams is a basic part of workplace culture where generally each time is specified particular roles and hardly do they cross operate on confidential assets unless not necessary. To implement the same as an accessibility control in Grafana a Teams module is provided to the admin where he can create groups termed as teams and add pre-existing users to the same, not only this but control their access levels out of a list of roles granted to each team. This helps in providing a filtered and secure vision to viewers which ease their work.

- **Query & Visualization**

Once the data source is selected, a dashboard in its place and a panel is created then queries play an eminent role in fetching the required data from the complete set. These queries may vary in their schema based on the data source we are using, for example- SQL Queries for MySql and Lucene Queries for ElasticSearch, etc. They work in the same manner as executing them in their default source environment, it's just an additional wrapper of measurement and limits setting along with a set of visualization is provided which makes each analysis accurate and filtered.

- **Use Cases Involved**

If we discuss how the implementation of features conversed above can bring us a one-screen solution we can think of a use case where we need system metrics, which can be fetched using Prometheus with the help of its exporters, namely node-exporter which pushes all the system related metrics into the time series database of Prometheus. A use case where we have to visualize MySql data used in our project, we will query and pull results for example, of the sales in the month of June and later visualize it on a graph. Another use case may be of showing real-time logs occurring in our web-server to keep an eye on the active state of our service. In the end, combining the use case above will result in a dashboard capable of providing most of the information ranging from system metrics, project data analysis to server logs, and much more, added on the go.

Components of Grafana

Supported Data Sources

- **AWS CloudWatch**

The service monitors the resources used on Amazon Web Services in realtime and provide metrics in an AWS panel, also provides features like limit notifications and alarm.

- **Azure Monitor**

A service to serve metrics from all on-premises resources utilized on Azure Cloud Platform encouraging maximized performance

- **Elasticsearch**

It is a search-based server which uses Lucene for queries under the hood. It can be scaled in both directions vertically and horizontally. Usually used within the ELK Stack along with Beats.

- **Google Stackdriver**

A solution for extracting data from multiple cloud platforms or services but focused more on Google Cloud. It fetches and displays logs, metrics, metadata from Google Cloud VM, Amazon EC2, and can do the same for open source services like Cassandra, ElasticSearch, etc.

- **Graphite**

A monitoring tool conversed for running well on cloud infrastructure as well as cheap hardware.

- **InfluxDB**
It is a time-series database and InfluxData developed it with a goal to optimize data retrieval at a very high speed.
- **Loki**
A horizontally scalable log aggregation system inspired from Prometheus and backed by Grafana Labs.
- **Microsoft SQL Server (MSSQL)**
Relational Database Management System developed by Microsoft.
- **MySQL and PostgreSQL**
Open Source, easy to learn, and widely used Relational Database.
- **OpenTSDB**
A scalable time-series database that runs on Hadoop.
- **Prometheus**
A free software application with Time Series Database providing real-time metric and alerting.
- **Testdata**
It is not actually a service but fake data supplied to the Grafana server to test analysis and visualization and make conclusions for similar datasets.

Dashboard and Panels

The role of dashboards and panels play a major role in enhancing the visualization after we have the data required. As discussed earlier a Dashboard is a collection of panels and each panel is a unique entity with attributes like queries and visualizations in its definition. Also, It is marked as the second step to complete the setup where either we add a dashboard using an ID or create one by querying the sources we selected in step one. Each dashboard consists of a name and version number along with a commit message which helps in maintaining the version control and list the changes we made. A dashboard is further divided into visualization and rows, each panel is capable of being added as a visualization or gets converted to a row which may accommodate similar panels focused on visualizing data, in short categorizing the visuals inside a dashboard based on what they represent.

We are able to set a global refresh time for each dashboard and use additional settings using a gear button given on the top of each unit. The setting preview is another view which provides a list of features as described below:

- **General**

Consist of basic meta-data including tags, name, description, folder, and edit permission followed by info related to time zone and cursor preview.

- **Annotations**

These are markings which specify an event. They can be vertical lines or icons depending on the visualization type and give you detailed information when hovered.

- **Variables**

We can define certain strings along with the use of some predefined functions that are provided by Grafana to carry out labeling, assigning queries to name tasks. These are listed under the variables heading.

- **Links**

It provides you a feature to add other dashboards or websites as links to be used in the current dashboard. This can be useful when connecting related visualizations and links associated with them.

- **Versions**

Version Control is the need for any long term task involving frequent changes and the same is listed under this tab, including everything from versions along with their commit message and users who made the same. We can also restore the state of the dashboard to any previous commit with the help of a click.

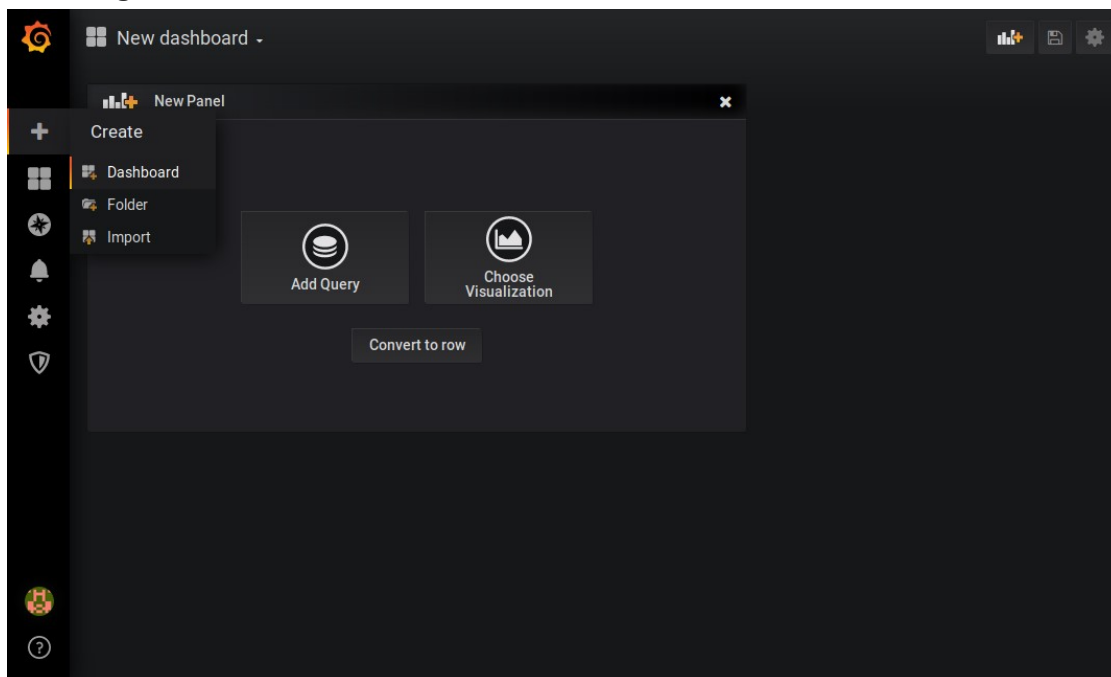
- **Permissions**

This tab is responsible for any grant access provided to an individual or team in the software. We can modify it on the basis of Users, Teams, or Roles and can specify their access as View, Edit and Admin.

- **JSON Model**

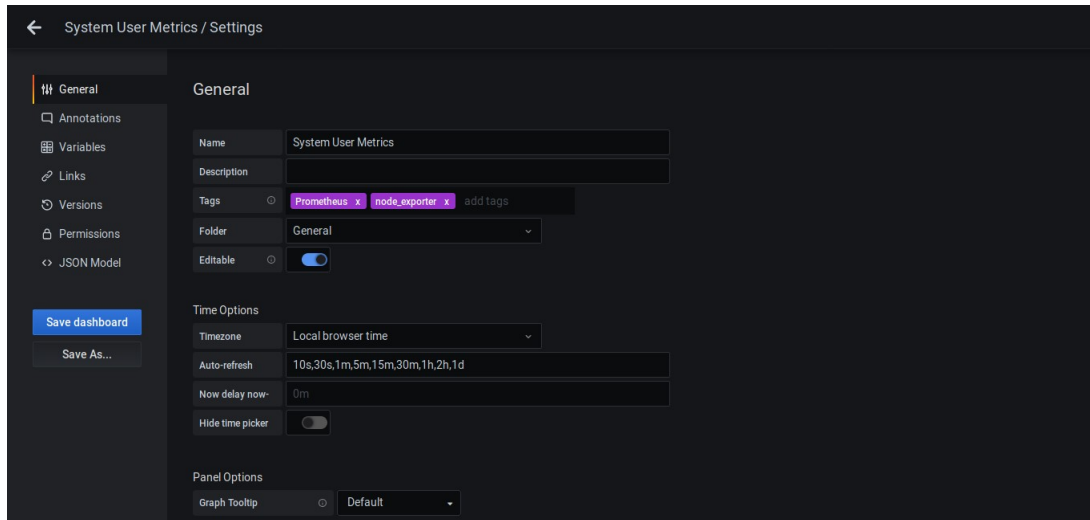
The tab specifies a vast JSON data structure which is basically the complete dashboard converted into a schema that helps in the commute and import/export operations across the software.

Creating a dashboard



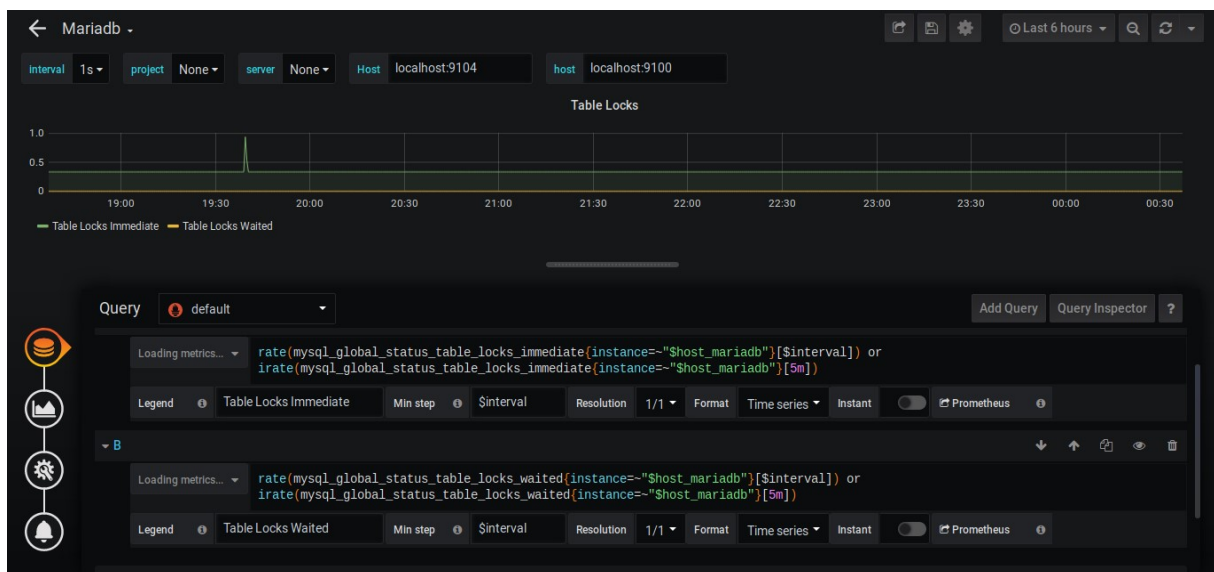
The options displays choice to create a new dashboard, create a new folder where specific dashboards will be accumulated or Import a dashboard from the Grafana Library.

Settings View



Panels are the building blocks of a dashboard and can be transformed into either a row or visualization depending on the option we select. Once we tap for adding a query and a new view is introduced which gives us an interface to implement the query on a data source and watch the results in the display above. Also, we can switch a list of visualizations provided in the menu which includes Graph, Gauge, Bar Gauge, Heat Maps, Logs, etc. We are able to implement multiple queries and choose a suitable representation for the same where each query consists of settings like if the data fetched is to be real-time, the legend of that section, Format, Steps in range, etc. which help in increasing the accuracy and improving the format of metrics displayed.

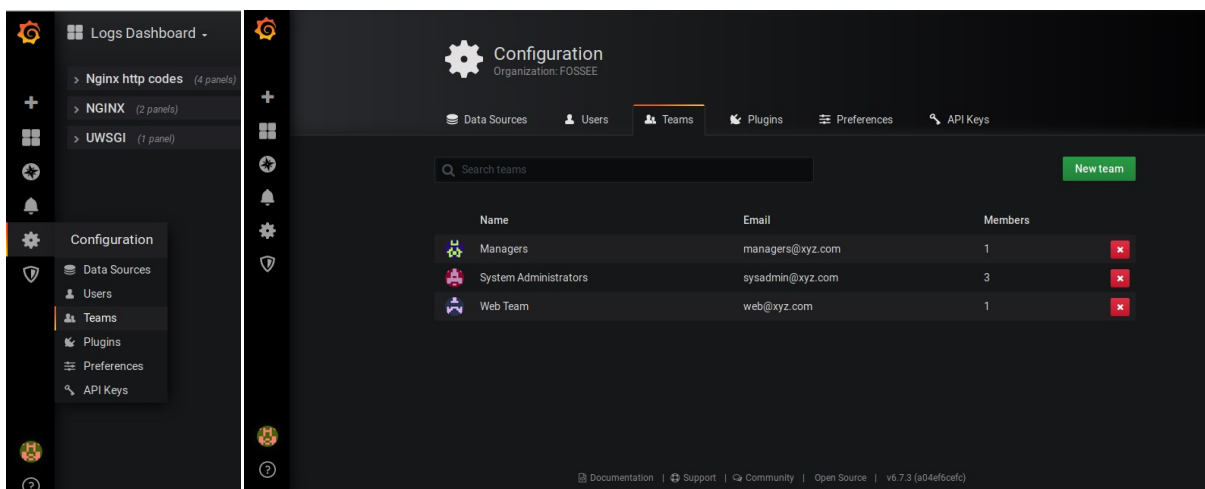
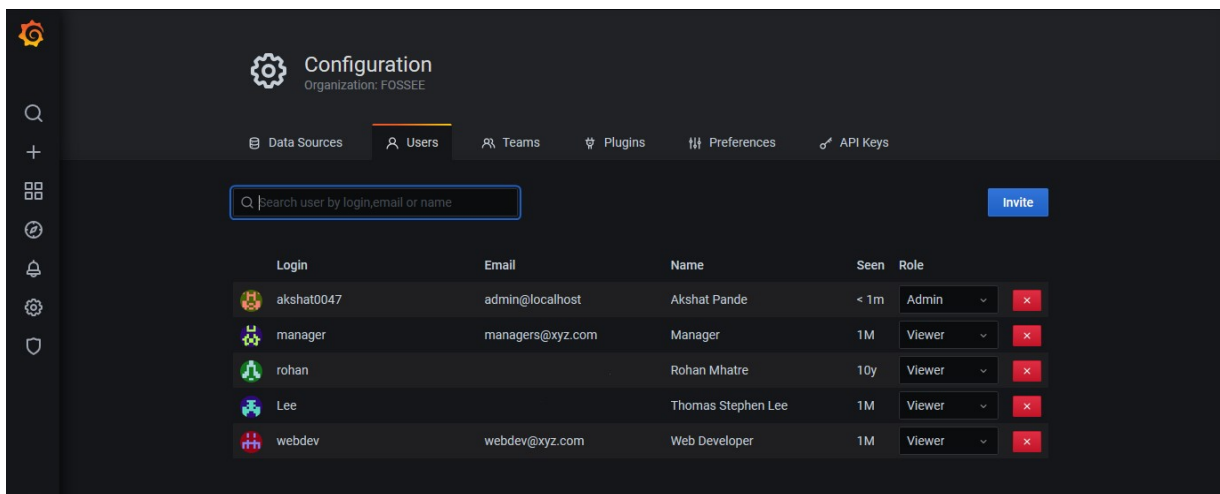
Panel Edit View



Users and Teams

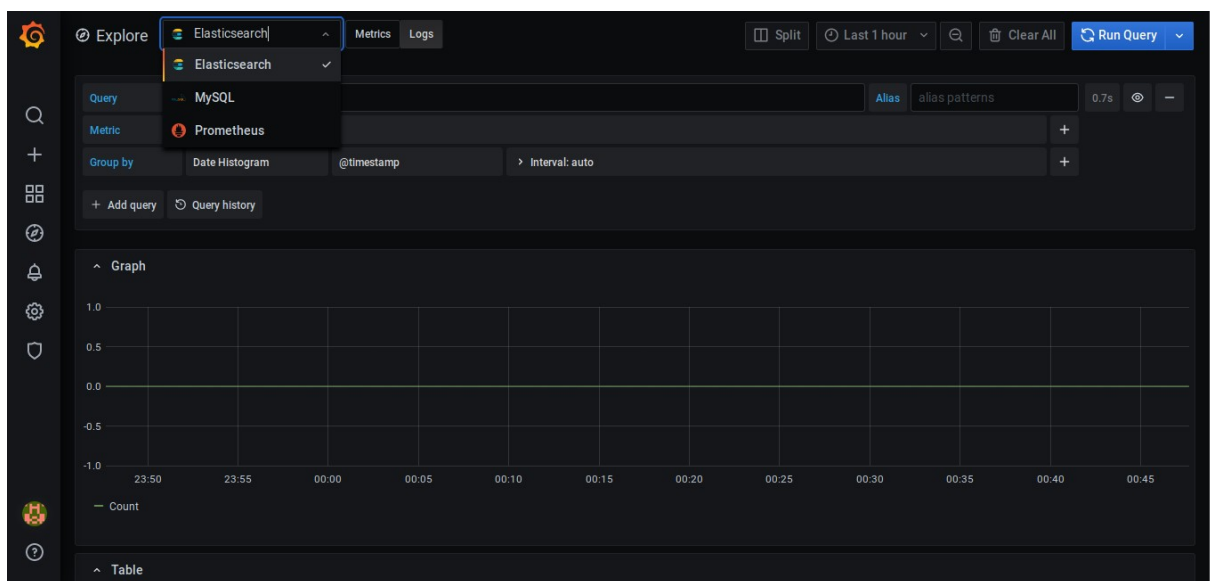
On studying the basic need and fashion of software we can conclude that Users and Groups are one of the most common features available. It's so important because there are always multiple people using the system and most of them belonging to a unique team in the workplace. This feature is also tapped by Grafana in a better and easy way under the option Users and Teams with a minimal adding process to mirror the offline workplace hierarchy on the Grafana Structure in almost no time.

Here, we can add users along with their username/email or Teams which as the name suggests are set of Users who fall under a specific category. These teams do have additional settings that are capable of setting a default home screen dashboard, a UI theme, and a time zone for the team. As discussed earlier, Users and Teams defined here are chosen at the time of granting permissions on a dashboard.



Explore

It is one of the most interesting sections of the UI as the name suggests Explore, it lets you query the selected data sources for freestyle analysis. This refers to the querying of data without saving it in a dashboard instead for test purposes or to get a basic idea of how the visualization looks for a certain time period which may result in valuable predictions. At times we need to look through the current data to solve a bug or study patterns for a temporary purpose, the Explore feature solves this problem. It gives you the complete querying functionality with limited visualization and a split view to compare between query results.



Grafana HTTP API

An API in its basic definition can converse as a set of protocols or functions defined in the code of a system that can be utilized within the shell or from the outside with limited access provided on basis of keys and limits. The HTTP API exposes its endpoints over HTTP to third party users and allows them to make use of the functions to carry out several functionalities of the system without using the User Interface. This consists of a list of endpoints mentioned in the API docs which can be pinged with authorization parameters issued by the provider which are to be included in the header along with other options and later sending the request with the required body for changes to work.

https://grafana.com/docs/grafana/latest/http_api/

Contributions

Note

- ❑ All the tasks are performed on Grafana Version
- ❑ Complete code details are mentioned in the attached Documentation
- ❑ Link to the Documentation with tasks in the sidebar

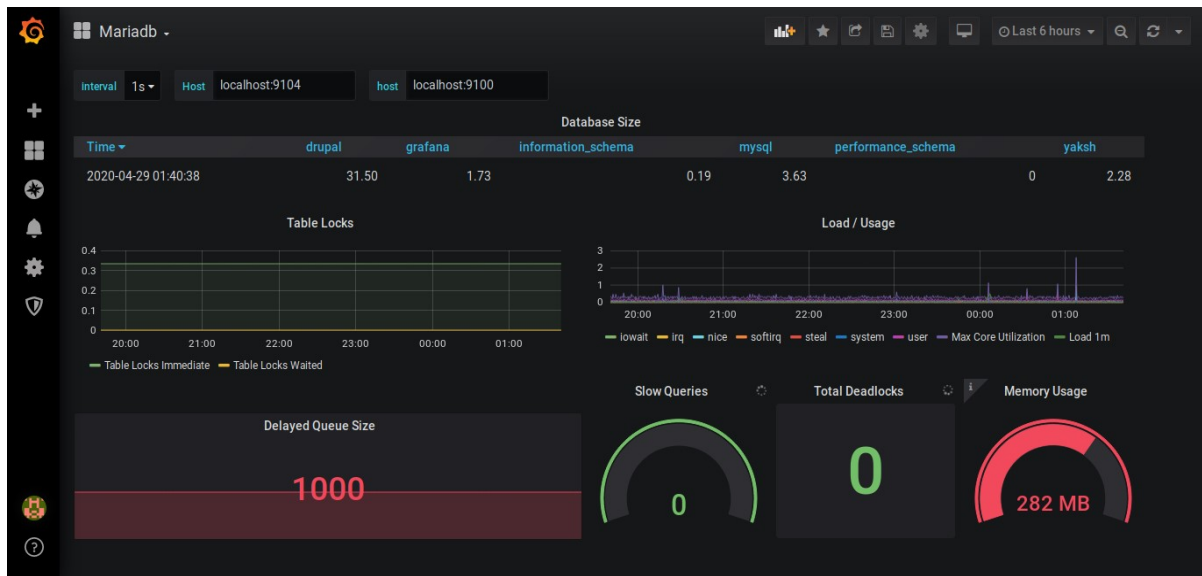
Tasks and Solutions

Task 1: Add a Dashboard to show metrics of MariaDB

Steps:

- Install MariaDB
- Install Prometheus
- Add mysqld_exporter to Prometheus
- Add Prometheus as a Data Source in Grafana
- Create a Dashboard and add Queries
- Choose name and Visualizations accordingly

Result:



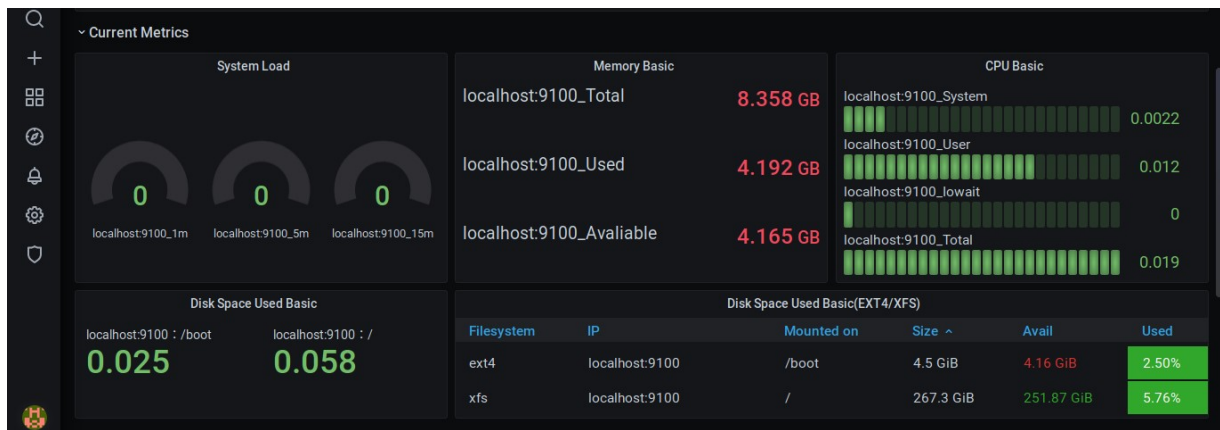
Task 2: Add a Dashboard to show system metrics both in real-time and overtime

Steps:

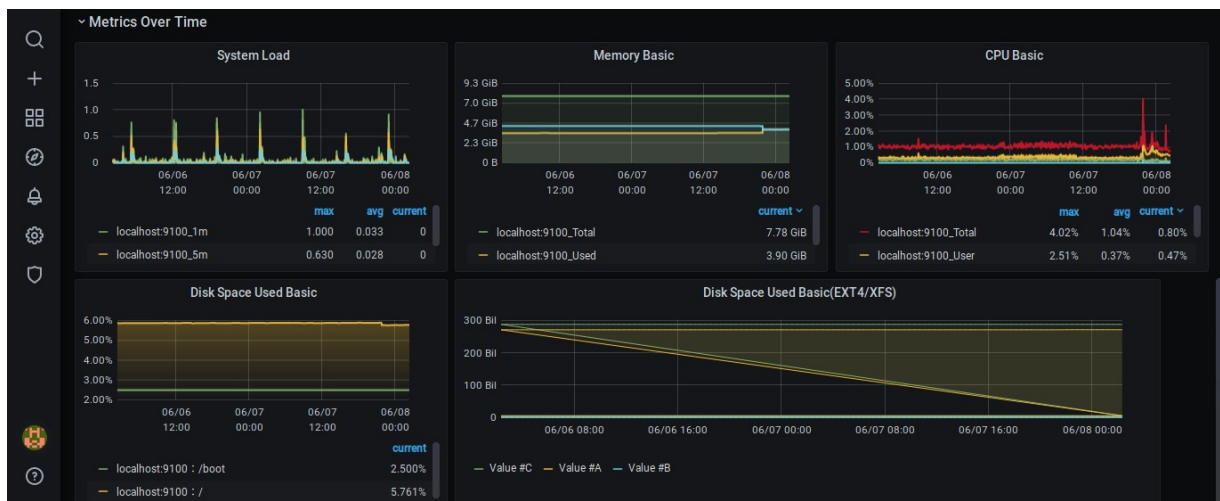
- Install Prometheus
- Add node_exporter to Prometheus
- Add Prometheus as a Data Source in Grafana
- Create a Dashboard and add Queries
- Choose name and Visualizations accordingly

Result:

Real-Time Metrics



Over-Time Metrics



Task 3: Visualize the tail of Nginx and UWSGI logs in a Dashboard

Steps:

- Install ELK Stack (ElasticSearch, LogStash, Kibana)
- Install FileBeat
- Edit configuration files accordingly
- Create Index Pattern in Kibana
- Query the logs in Kibana > Explore using Lucene
- Add ElasticSearch as a Data Source in Grafana
- Create a Dashboard and add Queries
- Choose name and Visualizations accordingly

Result:

Nginx Access Logs

```
2020-05-09 08:59:03 10.0.2.2 - - [09/May/2020:08:59:03 +0530] "POST /api/datasources/proxy/5/_msearch?max_concurrent_shard_requests=5 HTTP/1.0" 200 "https://akshat.fosseeapps.in/d/V4419AeZz/logs-dashboard?orgId=1&refresh=5s" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0" "47.8.27.193"
2020-05-09 08:59:03 10.0.2.2 - - [09/May/2020:08:59:03 +0530] "POST /api/datasources/proxy/5/_msearch?max_concurrent_shard_requests=5 HTTP/1.0" 200 "https://akshat.fosseeapps.in/d/V4419AeZz/logs-dashboard?orgId=1&refresh=5s" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0" "47.8.27.193"
2020-05-09 08:59:02 10.0.2.2 - - [09/May/2020:08:58:59 +0530] "GET /api/dashboards/uid/V4419AeZz HTTP/1.0" 200 "https://akshat.fosseeapps.in/d/V4419AeZz/logs-dashboard" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0" "47.8.27.193"
2020-05-09 08:58:55 10.0.2.2 - - [09/May/2020:08:58:53 +0530] "GET /api/dashboards/tags HTTP/1.0" 200 "https://akshat.fosseeapps.in/dashboards" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0" "47.8.27.193"
```

Nginx Error Logs

```
2020-05-09 08:19:05 2020/05/09 08:19:00 [error] 3836#3836: *3367 connect() failed (111: Connection refused) while connecting to upstream, client: 10.0.2.2, server: drupal.fosseeapps.in, request: "POST /api/kibana/kql_opt_in_telemetry HTTP/1.0", upstream: "http://[::1]:5601/api/kibana/kql_opt_in_telemetry", host: "drupal.fosseeapps.in", referer: "https://drupal.fosseeapps.in/app/kibana"
2020-05-09 08:17:40 2020/05/09 08:17:35 [error] 3836#3836: *3354 connect() failed (111: Connection refused) while connecting to upstream, client: 10.0.2.2, server: drupal.fosseeapps.in, request: "POST /elasticsearch/filebeat-7.6.2-*/_search?rest_total_hits_as_int=true&ignore_unavailable=true&ignore_throttled=true&preference=1588988055137&timeout=3000ms HTTP/1.0", upstream: "http://[::1]:5601/elasticsearch/filebeat-7.6.2-*/_search?rest_total_hits_as_int=true&ignore_unavailable=true&ignore_throttled=true&preference=1588988055137&timeout=3000ms", host: "drupal.fosseeapps.in", referer: "https://drupal.fosseeapps.in/app/kibana"
2020-05-09 08:17:25 2020/05/09 08:17:17 [error] 3836#3836: *3327 connect() failed (111: Connection refused) while connecting to upstream, client: 10.0.2.2, server: drupal.fosseeapps.in,
```

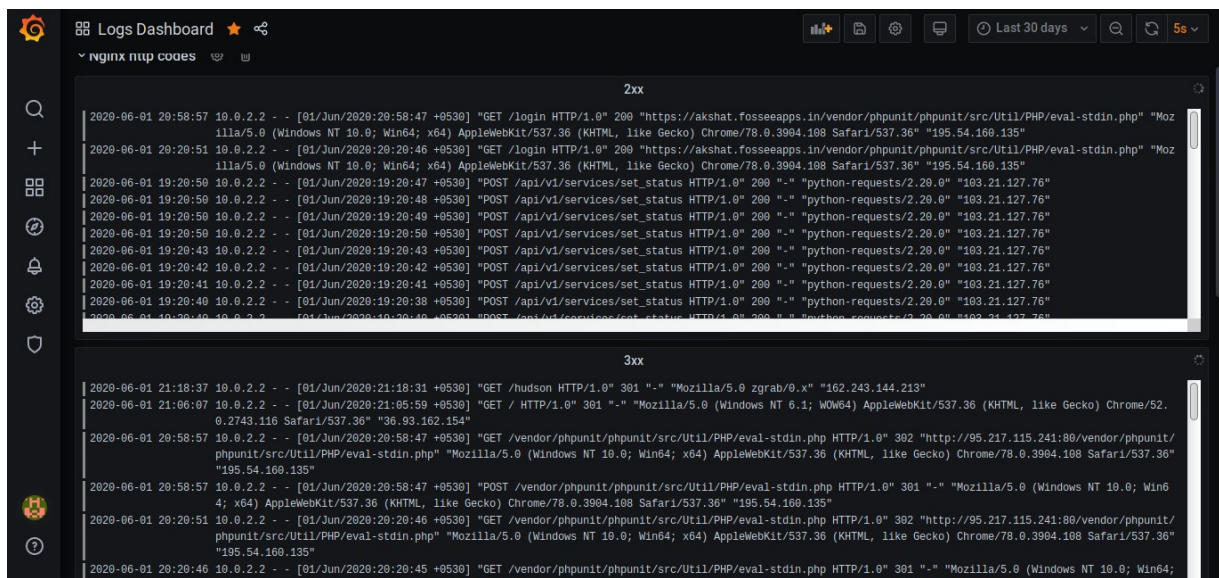
UWSGI Logs

Task 4: Filter the Nginx Logs according to their HTTP Codes

Steps:

- Install ELK Stack (ElasticSearch, LogStash, Kibana)
- Install FileBeat
- Edit configuration files accordingly
- Create Index Pattern in Kibana
- Query the logs in Kibana > Explore using Lucene
- Add ElasticSearch as a Data Source in Grafana
- Create a Dashboard and add **Regex** Queries
- Choose name and Visualizations accordingly

Result:



The screenshot shows the Kibana Logs Dashboard interface. The top navigation bar includes the Kibana logo, the title 'Logs Dashboard', and various utility icons. Below the navigation bar, there is a search bar and a filter bar. The main content area displays a list of log entries, which are filtered by HTTP status codes. The logs are grouped into two sections: '2xx' and '3xx'. Each log entry includes a timestamp, IP address, user agent, and the request details.

```
2xx
[2020-06-01 20:58:57 10.0.2.2 - - [01/Jun/2020:20:58:47 +0530] "GET /login HTTP/1.0" 200 "https://akshat.fosseeapps.in/vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/78.0.3904.108 Safari/537.36" "195.54.160.135"
[2020-06-01 20:20:51 10.0.2.2 - - [01/Jun/2020:20:20:46 +0530] "GET /login HTTP/1.0" 200 "https://akshat.fosseeapps.in/vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/78.0.3904.108 Safari/537.36" "195.54.160.135"
[2020-06-01 19:20:50 10.0.2.2 - - [01/Jun/2020:19:20:47 +0530] "POST /api/v1/services/set_status HTTP/1.0" 200 "-" "python-requests/2.20.0" "103.21.127.76"
[2020-06-01 19:20:50 10.0.2.2 - - [01/Jun/2020:19:20:49 +0530] "POST /api/v1/services/set_status HTTP/1.0" 200 "-" "python-requests/2.20.0" "103.21.127.76"
[2020-06-01 19:20:50 10.0.2.2 - - [01/Jun/2020:19:20:50 +0530] "POST /api/v1/services/set_status HTTP/1.0" 200 "-" "python-requests/2.20.0" "103.21.127.76"
[2020-06-01 19:20:43 10.0.2.2 - - [01/Jun/2020:19:20:43 +0530] "POST /api/v1/services/set_status HTTP/1.0" 200 "-" "python-requests/2.20.0" "103.21.127.76"
[2020-06-01 19:20:42 10.0.2.2 - - [01/Jun/2020:19:20:42 +0530] "POST /api/v1/services/set_status HTTP/1.0" 200 "-" "python-requests/2.20.0" "103.21.127.76"
[2020-06-01 19:20:41 10.0.2.2 - - [01/Jun/2020:19:20:41 +0530] "POST /api/v1/services/set_status HTTP/1.0" 200 "-" "python-requests/2.20.0" "103.21.127.76"
[2020-06-01 19:20:40 10.0.2.2 - - [01/Jun/2020:19:20:38 +0530] "POST /api/v1/services/set_status HTTP/1.0" 200 "-" "python-requests/2.20.0" "103.21.127.76"

3xx
[2020-06-01 21:18:37 10.0.2.2 - - [01/Jun/2020:21:18:31 +0530] "GET /hudson HTTP/1.0" 301 "-" "Mozilla/5.0 zgrab/0.x" "162.243.144.213"
[2020-06-01 21:06:07 10.0.2.2 - - [01/Jun/2020:21:05:59 +0530] "GET / HTTP/1.0" 301 "-" "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/52.0.2743.116 Safari/537.36" "36.93.162.154"
[2020-06-01 20:58:57 10.0.2.2 - - [01/Jun/2020:20:58:47 +0530] "GET /vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php HTTP/1.0" 302 "http://95.217.115.241:80/vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/78.0.3904.108 Safari/537.36" "195.54.160.135"
[2020-06-01 20:58:57 10.0.2.2 - - [01/Jun/2020:20:58:47 +0530] "POST /vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php HTTP/1.0" 301 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/78.0.3904.108 Safari/537.36" "195.54.160.135"
[2020-06-01 20:20:51 10.0.2.2 - - [01/Jun/2020:20:20:46 +0530] "GET /vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php HTTP/1.0" 302 "http://95.217.115.241:80/vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/78.0.3904.108 Safari/537.36" "195.54.160.135"
[2020-06-01 20:20:46 10.0.2.2 - - [01/Jun/2020:20:20:45 +0530] "GET /vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php HTTP/1.0" 301 "-" "Mozilla/5.0 (Windows NT 10.0; Win64;
```

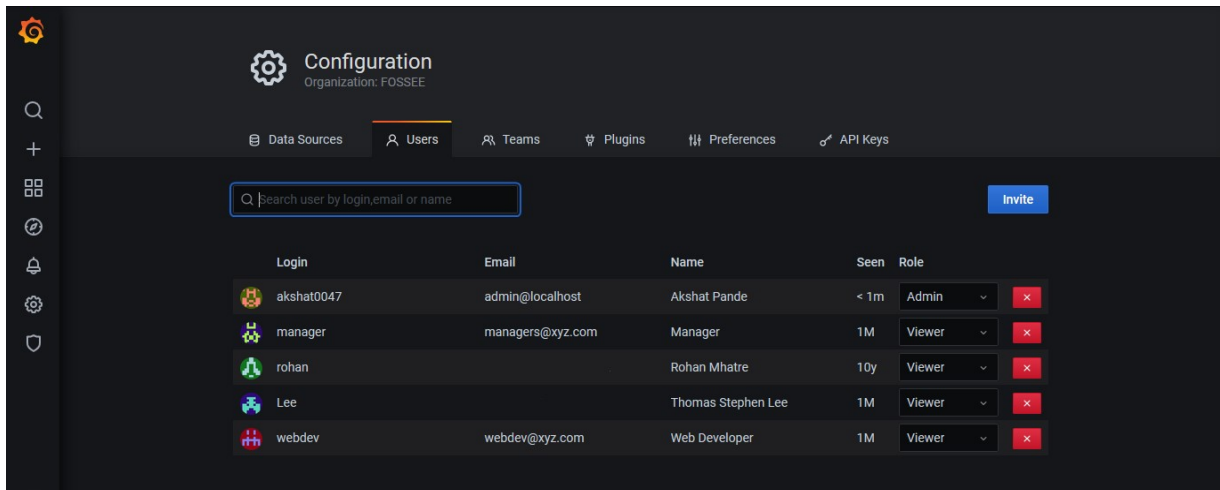
Task 5: Create 3 Teams in Grafana and assign them specific dashboards

Steps:

- Create multiple users
- Create 3 Teams (Managers, System Administrators and Web Team)
- Assign Users to respective Teams
- Edit Team preferences for a default dashboard
- Edit Permission for each Dashboard visible to the specific team
 - Logs > Web Team [View] || System Administrators [Edit]
 - DB Metrics > Web Team [View] || System Administrators [Edit] || Managers [View]
 - System Metrics > System Administrators [Edit] || Managers [View]

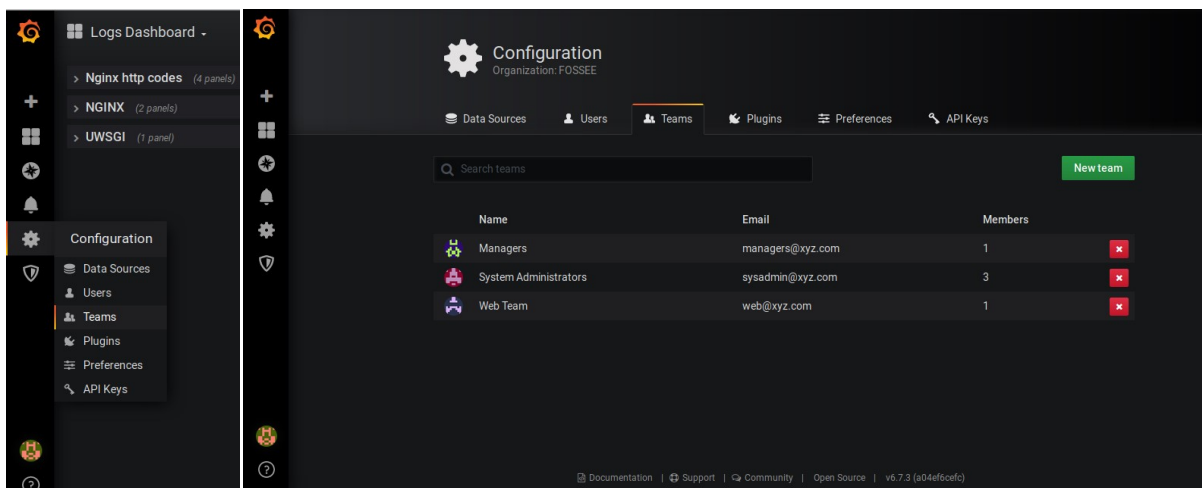
Result:

Users Created



Login	Email	Name	Seen	Role	
akshat0047	admin@localhost	Akshat Pande	< 1m	Admin	
manager	managers@xyz.com	Manager	1M	Viewer	
rohan		Rohan Mhatre	10y	Viewer	
Lee		Thomas Stephen Lee	1M	Viewer	
webdev	webdev@xyz.com	Web Developer	1M	Viewer	

Teams Created



Name	Email	Members	
Managers	managers@xyz.com	1	
System Administrators	sysadmin@xyz.com	3	
Web Team	web@xyz.com	1	

Side-Tasks in the process

The interesting part about solving the tasks mentioned above is I did not only create a new dashboard for every analysis instead I've to follow the whole process of setting up the data source from scratch. This happened because I was creating my use cases and then making Visualizations and Analysis out of them. There were several installations and server setups I went through in the sequence of solving the tasks, these are listed below and the code for the same can be found in the attached Documentation:

- Managing SELinux and Firewall on CentOS 8
- Installing and Setup Nginx 1.17.10
- Adding SSL to Nginx using Letsencrypt
- Installing and Setup MariaDB 10.4
- Install and Setup Prometheus with additional Exporters
- Working with PromQL
- Install and Understand ELK Stack
- Working with Lucene and Kibana

Problems Faced

Problem: Either to use LogStash or not

Solution: LogStash acts as a middleware used to filter logs into a schema but as we required logs directly from the files I used FileBeat which is a simple extension to ELK Stack for logs.

Problem: Unable to pull log data from Elasticsearch

Solution: Initially use Kibana after FileBeat setup and create an Index Pattern using the User Interface. Following this use this index pattern in the Grafana Data Source to specify collection source.

Problem: Install SSL certificate for a particular domain only using certbot

Solution: `$ sudo certonly -d <domain_name>`

Problem: Adding configurations to SELinux

Solution: Mentioned in Installation and Setup Section (**Sec. 1.2**)

DevOps

Introduction to DevOps

It is defined as the software development strategy which bridges the gap between the Development and Operations team of a company. Usually, people confuse DevOps with a framework or technology but it's not the case, it is just a concept/culture implemented in the production environment to relieve the conflicts arising between the two teams. There are a variety of tools available to counter issues in different phases of the code life-cycle, these help the teams as a unit to build a pipeline and automate several processes resulting in the build, test, and release of software in a fast and reliable manner.

Talking about the history of DevOps it dates back to 2007 - 2008 when the development and operations team were a completely different unit. They started being vocal about the issues they faced in the development cycle and addressed it as a serious drawback in the Industry. Soon discussions began to happen from in-person to online forums and meetups which also included several leaders from Industry and soon they found a solution to bridge the gap and made things connected in a sequence. It's mentioned that the transformation of a company to a DevOps culture doesn't happen overnight but once it is finalized the pipeline of the product becomes highly efficient and free of risk.

Mr. Jez Humble, the co-author of the book 'The DevOps Handbook' which is considered as one of the golden learning sources, coined the term CALMS. It is a framework that evaluates a company's potential to shift towards DevOps culture and measure the success quotient along with the transformation. The CALMS define its components as:

Culture

It specifies that DevOps is not all about tools and implementing the same cannot guarantee the achievement of the former instead in its true sense, the focus is about the collaboration of Development and Operations team with common goals and working together to find a solution with the help of a tool designed for the use case.

Automation

One of the important steps involved in the ease of any process is automation as repeated manual work may result in errors and also depicts an inefficient workflow. Build, test, deploy and provisioning are the first steps while moving towards DevOps as it builds a single system that would benefit everyone in different teams. Continuous Delivery and Configuration as code are two major aspects of this step.

Lean

It encourages the workflow to be more focused upon the agile methodology or to minimize the details and continue with the development on the go. The focus is on continuous improvement and dealing with failure, making the system presentable in its simplest form and consistent with your build to the complex state.

Measurement

The result of any effort can only be predicted after studying the data and attributes related to the work and the same principle works with this methodology. There are various tools and techniques to keep track of such data related to the changes implemented in the process and based on this information several teams in the department are capable of making better decisions and introducing features for the welfare of our product.

Sharing

Connection is an important aspect of DevOps culture and it is only possible by sharing information. It helps in easing the friction between development and operations teams where a developer can indulge into the details of how the app is pipelined through production because in the steps involving automation a basic idea of app design is necessary, this can be made possible with proper communication between the two. For the same reasons, there are support based developers whose work is to keep track of issues arising from the client-side and write patches for a solution bridging the gap between the two teams.

Category of Tools Available

There is a wide range of tools available for the problems occurring in the process of adapting to a DevOps culture. Deciding the best tool which may fulfill your requirement is what matters and is the role of the DevOps Engineer in a company. Tools may exist to be a solution of a particular use case or just solving a variety of similar use cases in the scenario, this doesn't mean the complete stack tool is a proven best solution instead it can be a fact that it may result in an overkill as a solution. For eg:- Using Ansible to run a script on two servers, it'll have to work but not needed.

The DevOps life-cycle consists of a set of phases which are the components of a pipeline we get as a result, each such phase has its own set of tools as a solution. Remember, there are a variety of tools available for each phase, and choosing the best is only possible with proper research about the tool and communication with the development team about the application design. Yes, experience does play an important role in this field as there are a number of scenarios, drawbacks of techniques which are only observed/visible after a saturation point.

The various phases involved are listed below with few tools that fall under the category:

Plan & Code

Roots of any application are the code structure it maintains from day one including the versions, solving bugs, planning of tasks, and the complete workflow. This is the initial building block of DevOps process and there are tools like Version control whose responsibility is to maintain code versions according to changes with committing each change with a message, not only this but there are a variety of features packaged along for jumping across commits to connect to platforms like GitHub, BitBucket. Workflow management tools are also a part of this category which lets you form a schedule to keep a track of tasks and additional features integrated according to your needs.

Code

Git, Subversion, Linters

Plan

Jira, PivotalTracker, VersionOne, Targetprocess

Build & Test

This layer can be quoted as equivalent to what middleware is in development. Most of the codes need a build that is to generate files in a certain format suitable for production or make the code work as a unit. Also, the development of each application is complemented by writing suitable test cases for the same to check it thoroughly and keep a track of bugs before delivering it as a product. All of this if done manually will consume a lot of time and may mess up the structure if not done is complete isolation. In order to deal with it we implement automated builds and test checks using tools listed below, these tools vary on the basis of languages and test cases contradicted.

Build

Gradle, Maven, ApacheAnt, Rake, Invoke

Test

Selenium, Junit, TestSigma, Tricentis Tosca

Deploy & Operate

One of the later stages of a pipeline is deployment, depending on the sequence involved at times it is deployment and delivery or just delivery. The difference persists if we're directly making changes to the user-facing product or first doing it in our environment. In the present scenario where scaling is a basic need of most of the technologies the infrastructure scales at a fast pace. Even if we are dependent on some Infrastructure as a service we still have to manage a number of resources with consistency which gets out of hand most of the time. To solve this we cope up with the tools listed below:

Deploy

AWS CodeDeploy, Docker, Rancher, Ansible, Puppet

Monitor

If implemented, the monitoring tools layer is known as the last one, and the DevOps cycle loops back from this position. Almost every successful and production-grade product has its own monitoring interface which consists of just every metric on the system ranging from utilization, database to error logs. These help different teams including the non-technical ones to study the working and performance of products and make decisions accordingly. Few tools popular for the purpose are mentioned below.

Monitoring

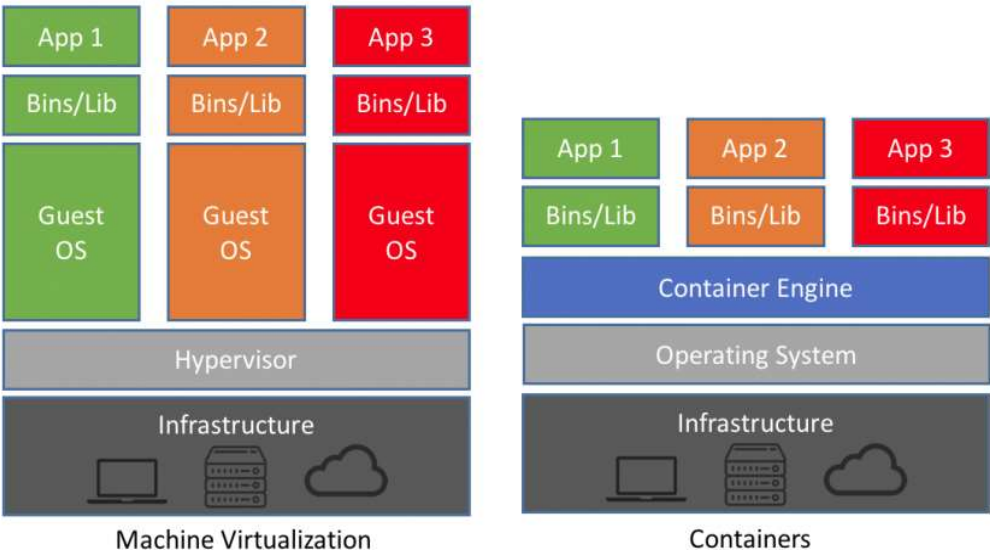
ELK Stack, Splunk, Grafana, Graphite

Containerization

Importance of Containers

Containers are one of the best solutions for problems like “It works on my system but not sure on others” basically, it’s a packaged environment with the root libraries of an operating system necessary for basic functionalities with an added layer of basic utilities on which the code is dependent followed by dependencies of code and directories in the end added with scripts and command to get the job running.

The containerization was introduced as a concept getting inspired from the old cargo ships where irregular-sized goods were tied together and their transport became a challenge to the authorities, later containers helped these to get sorted inside them as bundles and everything was organized in a better manner. The journey of virtualization in the world of systems has a long way, Virtual Machines are still used as one of the ways to create isolation within the systems and the applications running inside them but what they do is segregate application from operating and hardware, each VM gets shot up with a different operating system and allocated resources which drains the master machine or a part of the resource is always wasted.



This was solved with the introduction of containers as they shared the same Operating System and were lighter, better performing, scalable, and easy to move. Above is a picture contrasting the architecture between the two, on one side Virtual Machines sit upon a Hypervisor over infrastructure mechanism with each VM having its own OS, Libraries and Applications whereas in the Docker scenario a Container Engine sits upon an Operating System over Infrastructure mechanism with each container having the essential binaries and libraries with the apps installed. Some of the popular Virtualizations software and Containerization software are listed below:

Virtualization

VMware, VirtualBox

Containerization

Docker, Rocket

Container Engines

A container engine as discussed earlier is a layer above the infrastructure we use which contains the root bins/libs essential for connecting docker with the OS beneath. As the name suggests the working of the whole docker system is dependent on this engine, it comprises of three major operations:

Providing API / User Interface

The operation is a bit self-explanatory as it quotes about providing a way to easily interact with the processes related to container manipulation. So that we do not have to run a script or interact with a language/set of functions, instead it gets a command line connected with it which provides us a list of commands to carry out all of its features using CLI and focus on our plan, get things faster and managed.

Pulling/Expanding Images to disk

This feature manages images and management of layers, it is responsible for pulling images and storing them in the local cache so that if they're required by the container we can supply them from the local storage. Each time a new container is started the engine uses this image and also adds an extra layer in case for the data to be written in the container. The process involves pulling the image layer-wise in order to create a set of files, all of this governed under OCI(Open Containers Initiative). The image spec defines content and metadata in the container repository while the distribution spec specifies a set of rules to grab necessary layers in the Registry Server.

Next, as quoted above by usage of cached images, what actually happens under the hood is a virtual copy of the image files that is created and mapped to the container. Later adding a writable volume layer if necessary. These are achieved using OS functionalities like overlays and device-mapper.

Building a configuration file

This is a configuration file created by container engine and supplied to runc which is an OCI runtime, this tool requires two CLI options

- The directory where contents of the image with layers will be expanded
- The manifest file which contains directives to be followed

As none of the above tasks are simple to follow as pulling images requires maintaining layers and creating the manifest file is complex as it gets vast and complex these are carried out by the docker engine and provided as a utility to users.

Implementation of Docker

Docker is an open-source tool designed to create, deploy, and run containers with packaged applications. This provides an isolated and developer-oriented environment to the code so that it performs in the same manner on any machine like it would on the developer's system, not only this but scaling and moving your application gets easier across servers. We have already discussed the importance of containers and container engines in the article above and docker is a mere implementation to bring all of that in the form of the tool.

The components of the Docker engine which sits upon the infrastructure to provide container-based services consist of:

Dockerd - The user-facing API(A server with long-running daemon process).

Containerd - A runtime to abstract away syscalls or OS-based functionality to run containers is different operating systems.

Runc - An OCI runtime enclosed in containerd which takes parameters for image directory and manifest file later being responsible for defining directives.

Steps to Create a Docker Image

The complete concept is based upon a few components namely Dockerfiles, Container images, containers, commands, etc. talking about the workflow of containerizing a project with docker, the approach to follow concludes that first, we have to understand the architecture of an app and the technology involved to choose a base image in our Dockerfile. There exists a list of commands to be specified in the Dockerfile each with a motive to perform or comprehend an action. For eg- COPY to copy files from drive to container, RUN to execute a command inside the container, CMD to execute a command at the runtime of container etc.

Next, we study about the libraries, packages, and additional languages/techs that will be required by our code as a base to perform and specify their installation using the RUN command inside the file. Once we have the basic dependencies we can move towards specifying commands to copy required directories inside a path in the container, these will include source code as well as the relevant scripts to support the system.

Now we will set the path to the working directory using WORKDIR and install technology-based libraries/packages required by our code to get the application working, additional commands may be issued based on the extra scripts we're copying and the type of application we are using. These may include EXPOSE which is used to expose a port to the outside world of containers and CMD to run the command which fires up our app.

After getting our Dockerfile ready in place we will build the image and add a tag to it, also will push it to an image registry like DockerHub if necessary. Later these images can be pulled instead of building them each time which will ease the hassle of setting up a development environment of the application. We can use these images in Docker Compose or Docker Stack directly to achieve more advanced features supplied by Docker. Docker Compose is an external utility to be installed along with docker which helps us define scripts to run multiple containers at once with configurations supplied to each one of them using a syntax defined by docker. This helps in running a microservices-based system using a single file.

Implementation Work

<https://docs.docker.com/engine/reference/builder/> (Link to complete reference for DockerFile)

<https://github.com/akshat0047/yaksh-devops> (Link to Dockerfiles created)

<https://hub.docker.com/r/akshat0047/yaksh-codeserver> (Link to Yaksh Codeserver Image)

<https://hub.docker.com/r/akshat0047/yaksh-django> (Link to Yaksh Django Image)

The operations related with containers include listing the images on the local machine, starting a container, stopping a container, removing the image from the local system, adding a volume to it, assigning ports and much more, the list is long so I have mentioned a cheat sheet of important commands below along with the link to complete documentation. Once we have the image we can follow these commands to perform any functionality provided by Docker.

Docker Cheat Sheet

Build

Build an image from the Dockerfile in the current directory and tag the image
`docker build -t myimage:1.0 .`

List all images that are locally stored with the Docker Engine
`docker image ls`

Delete an image from the local image store
`docker image rm alpine:3.4`

Share

Pull an image from a registry
`docker pull myimage:1.0`

Retag a local image with a new image name and tag
`docker tag myimage:1.0 myrepo/myimage:2.0`

Push an image to a registry
`docker push myrepo/myimage:2.0`

Run

Run a container from the Alpine version 3.9 image, name the running container "web" and expose port 5000 externally, mapped to port 80 inside the container.
`docker container run --name web -p 5000:80 alpine:3.9`

Stop a running container through SIGTERM
`docker container stop web`

Stop a running container through SIGKILL
`docker container kill web`

List the networks
`docker network ls`

List the running containers (add --all to include stopped containers)
`docker container ls`

Delete all running and stopped containers
`docker container rm -f $(docker ps -aq)`

Print the last 100 lines of a container's logs
`docker container logs --tail 100 web`

Docker Management

All commands below are called as options to the base `docker` command. Run `docker <command> --help` for more information on a particular command.

<code>app*</code>	<i>Docker Application</i>
<code>assemble*</code>	<i>Framework-aware builds (Docker Enterprise)</i>
<code>builder</code>	<i>Manage builds</i>
<code>cluster</code>	<i>Manage Docker clusters (Docker Enterprise)</i>
<code>config</code>	<i>Manage Docker configs</i>
<code>context</code>	<i>Manage contexts</i>
<code>engine</code>	<i>Manage the docker Engine</i>
<code>image</code>	<i>Manage images</i>
<code>network</code>	<i>Manage networks</i>
<code>node</code>	<i>Manage Swarm nodes</i>
<code>plugin</code>	<i>Manage plugins</i>
<code>registry*</code>	<i>Manage Docker registries</i>
<code>secret</code>	<i>Manage Docker secrets</i>
<code>service</code>	<i>Manage services</i>
<code>stack</code>	<i>Manage Docker stacks</i>
<code>swarm</code>	<i>Manage swarm</i>
<code>system</code>	<i>Manage Docker</i>
<code>template*</code>	<i>Quickly scaffold services (Docker Enterprise)</i>
<code>trust</code>	<i>Manage trust on Docker images</i>
<code>volume</code>	<i>Manage volumes</i>

*Experimental in Docker Enterprise 3.0

www.docker.com

There are some important concepts about containers for which the explanation is out of the scope of this report, please refer Docker Official Documentation and refer them for a better understanding of the background features, few of them are:

Volumes, Network, Port mapping, Environment Variables, Exec into container, etc.

<https://docs.docker.com/> (Complete Documentation for Docker)

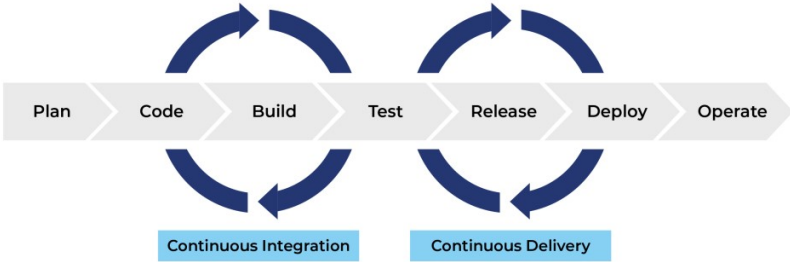
CI / CD

Workflow and Advantages

Continuous Integration and Continuous Delivery wraps a set of cultures and practices concerned with enabling frequent code changes in an automated and reliable manner. It is one of the crucial steps involved in the DevOps lifecycle and a must for successfully adopting Agile Methodology. The reason being it helps in focusing on better code design, security, and features rather than dealing with the hassle of hosting and maintaining different versions on the server.

The goal of the process is to automate build, package, and then test the application, this involves providing a complete module for checking against test cases defined for the app by the testing team. All this is made possible with a series of steps taking place in an isolated environment which is usually a cloud server. It is the place where installation of necessary packages, setup of environment, and Integration takes place. The code is maintained with the help of some version control system with centralized storage from where each change triggers the pipeline designed for the complete process to take place.

The code travels the same pipeline each time and follows the steps/specifications listed in a script along with environment variables which are a must to send in each run. This results in Continuous Integration, later based on the company's workflow they may have multiple deployment environments or just one. Accordingly, the code is moved inside a container or compact format to deployment/testing/delivery machine and accomplishes the purpose of a CI/CD pipeline. All of this results in a workflow where the application is capable of getting automatically tested and hosted as a product with the push of a commit. Always remember, continuous testing is important to deliver a quality product.



Travis CI

It is a Continuous Integration tool that gets integrated with GitHub and BitBucket to provide a platform where you are able to build and test your code and then schedule it for deployment if required at some other platform. Each repository consisting of a Travis webhook has a `.travis.yml` file defined in it which defines the complete configuration of the environment where integration will take place. This webhook is triggered on several events that can be chosen under the settings tab of a repository. We can enable it on a pull request, commit or branch merge and many more detailed settings for customizing the trigger mechanism is provided for every use case.

The best part about this hosted and distributed continuous integration tool is that it provides you with free services if your project is open source. It supports a number of languages and makes the workflow from your development space to deployment server smooth and just a push away. It comes in two versions; the free version requires your code to be open source and hosted on GitHub, the enterprise version is for companies to be used as a part of the workplace and starts from 63\$ per month.



Implementation in Yaksh and Node App

To integrate a project with Travis CI the steps you need to follow are mentioned below:

- SignUp for a Travis Account
- Connect it to your GitHub
- Switch on the repository you want to add a webhook to
- Add a `travis.yml` configuration file to the repository (<https://docs.travis-ci.com/user/customizing-the-build>)
- Specify events that trigger Travis under those repositories settings tab
- Push the code and watch your build in Travis Dashboard

Travis File for Yaksh

https://github.com/FOSSEE/online_test/blob/master/.travis.yml

Travis File for Node App

```
language: node_js
node_js:
  - "12.13.0"

cache:
  - node_modules

before_install:
  - npm install -g npm

script:
  - npm install

deploy:
  provider: heroku
  api_key:
    secure: xxxxxx
  app: alias-chat
  on:
    repo: akshat0047/socket-chat
    branch: master
```

Dashboard

The screenshot shows the Travis CI dashboard. On the left, there is a sidebar with a search bar and a list of repositories under 'My Repositories'. The main area displays the build details for 'akshat0047 / online_test', which is currently passing. The build log shows the following steps:

- swarm added swarm (3 passed)
- Commit d233078
- Compare d23307819b1c
- Branch swarm
- AKSHAT PANDE
- Python: 3.6
- AMD64

The build ran for 8 min 39 sec and finished 23 days ago. A 'Restart build' button is visible.

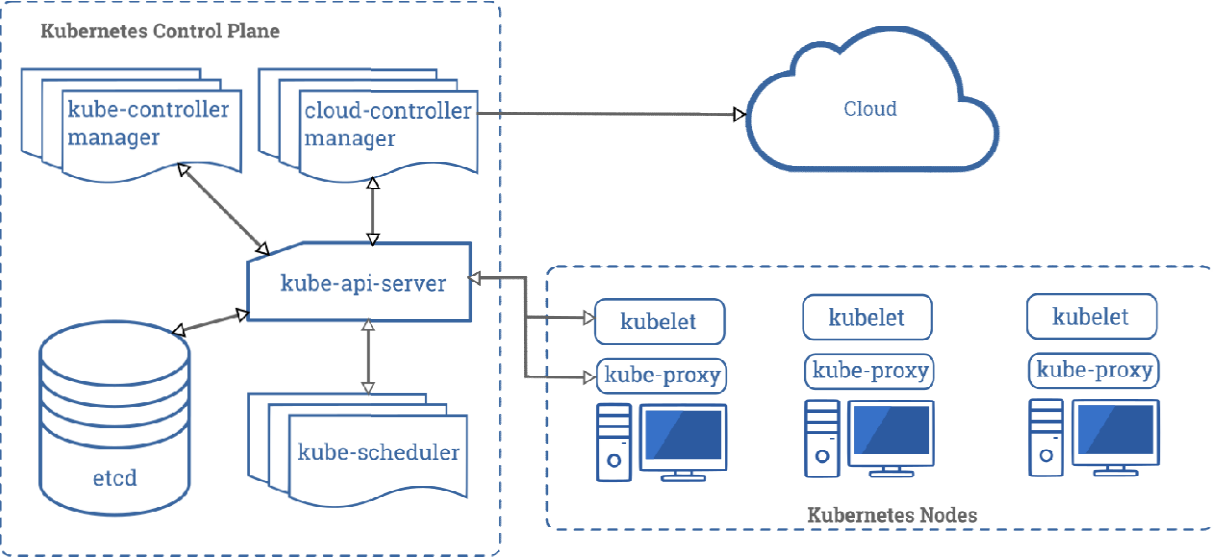
Orchestration

Introduction to Kubernetes

Orchestration can be termed as scaling of containers in an enclosed environment with the automated arrangement, combined working of complex computer systems and services.

It is different from running multiple containers at once as performed by docker-compose utility instead it's more focused on running multiple containers inside pods with automated configuration to work with each other with an added layer to all the pods making it a cluster.

Kubernetes is an open-source tool for orchestration providing vast features with several implementations, a detailed level of customization for those who have mastered it, and provision of clusters using managed-services along with tools to run it on a local or remote environment for beginners. The learning curve of Kubernetes is a bit steep so understanding the whole mechanism and implementing a cluster requires patience. The image below is a mere representation of Kubernetes Structure.



Major Components Involved

Pod

It is a collection of one or more containers bonded together to represent a microservice where the constituent containers share the same space and host along with pre-configured network settings

Replication Controller

A replication controller is a set of rules or a component programmed in such a manner that it ensures x number of pods working at a specific time inside the cluster. If they get lesser or more than the specified limit, the controller will scale them accordingly.

Replica Set

It's a modified version of the Replication Controller with the only difference that it selects the pods to be managed with the use of tags instead of pod name.

Services

A service can be defined as a policy along with specifications that is connected to a set of pods and make them accessible as a microservice inside the cluster.

ConfigMap

This helps us to isolate configuration inputs from the active pods so that we can change the configuration without editing the images or restarting pods.

Secrets

Almost the same as ConfigMap but the only difference is these consists of secrets that are used within Pods like passwords and API secrets are meant to remain confidential

Daemon Set

It is a component which when assigned to a specific pod makes sure that it runs at least an instance of that pod, used when a service is to be made available at each node

Deployment

This is a combined implementation of Replica Set and Pods with features to make the cluster attain the desired state. When deployed these can change the active state to a desirable state by changing replication policies and pods configuration. Also, deployments offer a great feature of rolling out updates for the application.

Load Balancer

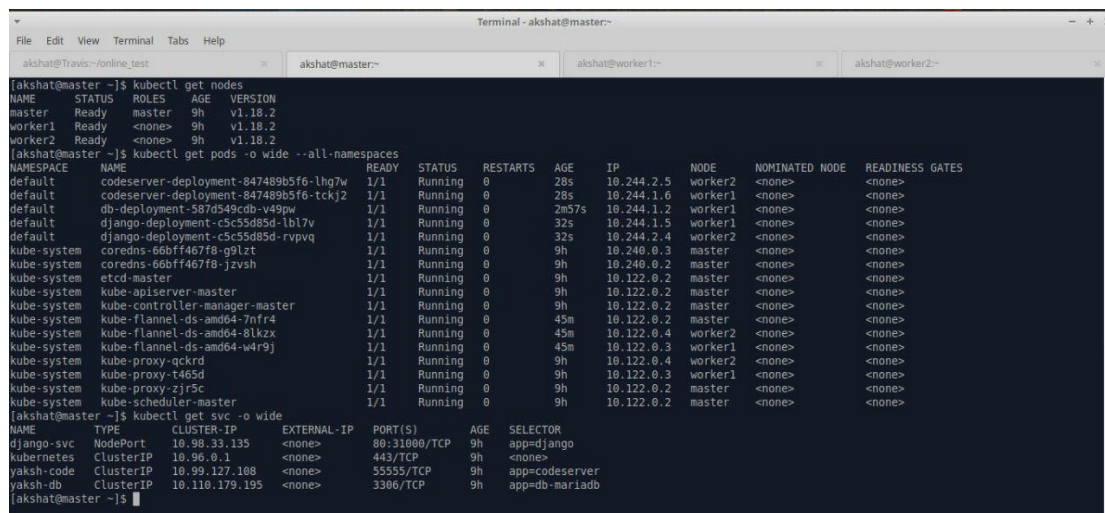
Depending upon the type of implementation we are using for Kubernetes the definition of a load balancer can vary but performing a common final result that is sanctioning an external cloud service that we would choose to distribute traffic across our nodes in the Cluster. In the managed implementation it can be done with the help of running a service while in the manual configuration we have to issue a load balancer on our own. It is an essential component as Kubernetes serves the purpose of running multiple instances and when they're related to any frontend service, it becomes important to have a common domain to access any of the exposed pods for the same.

Implementing Kubernetes on Yaksh/Sample App

There are a list of approaches that I followed to implement Kubernetes on Yaksh as listed below:

<https://github.com/akshat0047/yaksh-devops> (Link to Kubernetes Configuration File)

Using Kubeadm, kubectl, Kubelet and Pod Network to setup from scratch



```
Terminal - akshat@master:~
File Edit View Terminal Tabs Help
akshat@Travis:~/online_test x akshat@master:~ x akshat@worker1:~ x akshat@worker2:~
akshat@master ~$ kubectl get nodes
NAME          STATUS    ROLES    AGE   VERSION
master        Ready    master   9h    v1.18.2
worker1       Ready    <none>   9h    v1.18.2
worker2       Ready    <none>   9h    v1.18.2
akshat@master ~$ kubectl get pods -o wide --all-namespaces
NAMESPACE     NAME                                READY    STATUS    RESTARTS   AGE   IP             NODE     NOMINATED NODE   READINESS GATES
default       coderserver-deployment-847489b5f6-lhg7w  1/1      Running   0           28s   10.244.2.5     worker2  <none>            <none>
default       coderserver-deployment-847489b5f6-tckj2  1/1      Running   0           28s   10.244.1.6     worker1  <none>            <none>
default       db-deployment-587d549cdb-v49pw          1/1      Running   0           2m57s  10.244.1.2     worker1  <none>            <none>
default       django-deployment-c5c55d85d-lbl7v       1/1      Running   0           32s   10.244.1.5     worker1  <none>            <none>
default       django-deployment-c5c55d85d-rvpvq       1/1      Running   0           32s   10.244.2.4     worker2  <none>            <none>
kube-system   coredns-66bff467f8-g9l1t                1/1      Running   0           9h    10.240.0.3     master   <none>            <none>
kube-system   coredns-66bff467f8-jzvsh                1/1      Running   0           9h    10.240.0.2     master   <none>            <none>
kube-system   etcd-master                              1/1      Running   0           9h    10.122.0.2     master   <none>            <none>
kube-system   kube-apiserver-master                    1/1      Running   0           9h    10.122.0.2     master   <none>            <none>
kube-system   kube-controller-manager-master           1/1      Running   0           9h    10.122.0.2     master   <none>            <none>
kube-system   kube-flannel-ds-amd64-7nfr4              1/1      Running   0           45m   10.122.0.2     master   <none>            <none>
kube-system   kube-flannel-ds-amd64-8lkzx              1/1      Running   0           45m   10.122.0.4     worker2  <none>            <none>
kube-system   kube-flannel-ds-amd64-w4r9j              1/1      Running   0           45m   10.122.0.3     worker1  <none>            <none>
kube-system   kube-proxy-qckrd                          1/1      Running   0           9h    10.122.0.4     worker2  <none>            <none>
kube-system   kube-proxy-t465d                          1/1      Running   0           9h    10.122.0.3     worker1  <none>            <none>
kube-system   kube-proxy-zjr5c                          1/1      Running   0           9h    10.122.0.2     master   <none>            <none>
kube-system   kube-scheduler-master                    1/1      Running   0           9h    10.122.0.2     master   <none>            <none>
akshat@master ~$ kubectl get svc -o wide
NAME          TYPE          CLUSTER-IP   EXTERNAL-IP   PORT(S)          AGE   SELECTOR
django-svc    NodePort     10.98.33.135 <none>         80:31000/TCP     9h    app=django
kubernetes    ClusterIP    10.96.0.1    <none>         443/TCP          9h    <none>
yaksh-code    ClusterIP    10.99.127.108 <none>         55555/TCP        9h    app=codeserver
yaksh-db      ClusterIP    10.110.179.195 <none>         3306/TCP         9h    app=db-mariadb
akshat@master ~$
```

Using kops and Terraform on AWS EC2

```

Terminal - kubernet@akshat
kubernet@akshat~$ kubectl get nodes
NAME                                STATUS    ROLES    AGE     VERSION
ip-172-20-43-205.eu-central-1.compute.internal Ready    node     9m24s   v1.16.9
ip-172-20-46-53.eu-central-1.compute.internal Ready    master   20m     v1.16.9
ip-172-20-52-131.eu-central-1.compute.internal Ready    node     6m28s   v1.16.9
kubernet@akshat~$ kubectl get pods
NAME                                READY    STATUS    RESTARTS   AGE
my-nginx-deployment-7ddd8b97f8-mbrmn 1/1      Running   0           6m7s
kubernet@akshat~$ kubectl get svc
NAME                                TYPE        CLUSTER-IP    EXTERNAL-IP  PORT(S)          AGE
kubernetes                          ClusterIP   100.64.0.1     <none>        443/TCP          20m
my-nginx-service                     NodePort    100.66.222.159 <none>        80:31311/TCP    6m3s
kubernet@akshat~$

```

Using Managed Service on Digital Ocean

```

Terminal - @django-deploy
Cluster - DigitalOcean - Mozill... Terminal - @django-deploy
Terminal - @django-deployment-576dcff488-7b9g7/Sites/online_test
akshat0047@akshat0047-HP-Pavilion-15-Notebook-PC:~/kub$ kubectl --kubeconfig="kubernetes-fossee-kubeconfig.yaml" get nodes -o wide
NAME                                STATUS    ROLES    AGE     VERSION    INTERNAL-IP    EXTERNAL-IP    OS-IMAGE                                KERNEL-VERSION    CONTAINER-RUNTIME
fosse-3u5i6                         Ready    <none>   55m    v1.17.5    10.122.0.6     139.59.13.114  Debian GNU/Linux 9 (stretch)          4.19.0-0.bpo.6-amd64  docker://18.9.2
fosse-3u5i1                         Ready    <none>   56m    v1.17.5    10.122.0.7     159.89.161.162 Debian GNU/Linux 9 (stretch)          4.19.0-0.bpo.6-amd64  docker://18.9.2
fosse-3u5it                         Ready    <none>   55m    v1.17.5    10.122.0.8     142.93.208.151 Debian GNU/Linux 9 (stretch)          4.19.0-0.bpo.6-amd64  docker://18.9.2
akshat0047@akshat0047-HP-Pavilion-15-Notebook-PC:~/kub$ kubectl --kubeconfig="kubernetes-fossee-kubeconfig.yaml" get pods -o wide
NAME                                READY    STATUS    RESTARTS   AGE    IP             NODE             NOMINATED NODE    READINESS GATES
codeserver-deployment-58bbdcf4b-b8hn5 1/1      Running   0           16m    10.244.0.192   fosse-3u5i6     <none>             <none>
codeserver-deployment-58bbdcf4b-k9rdp 1/1      Running   0           16m    10.244.1.14    fosse-3u5i6     <none>             <none>
django-deployment-576dcff488-7b9g7    1/1      Running   0           17m    10.244.0.246   fosse-3u5i6     <none>             <none>
django-deployment-576dcff488-g6ps8    1/1      Running   0           17m    10.244.1.15    fosse-3u5i6     <none>             <none>
akshat0047@akshat0047-HP-Pavilion-15-Notebook-PC:~/kub$ kubectl --kubeconfig="kubernetes-fossee-kubeconfig.yaml" get svc -o wide
NAME                                TYPE        CLUSTER-IP    EXTERNAL-IP  PORT(S)          AGE    SELECTOR
django-svc                          NodePort    10.245.90.207 <none>        80:31000/TCP    30m    app=django
kubernetes                          ClusterIP   10.245.0.1     <none>        443/TCP          59m    <none>
yaksh-code                          ClusterIP   10.245.51.193 <none>        55555/TCP       30m    app=codeserver
yaksh-db                             ClusterIP   10.245.207.174 <none>        3306/TCP        31m    app=db-mariadb
akshat0047@akshat0047-HP-Pavilion-15-Notebook-PC:~/kub$ kubectl --kubeconfig="kubernetes-fossee-kubeconfig.yaml" exec -it django-deployment-576dcff488-7b9g7 -- /bin/bash
root@django-deployment-576dcff488-7b9g7 online test|# curl 10.245.51.193:55555
5 processes, 0 running, 0 queued[root@django-deployment-576dcff488-7b9g7 online test]# ^C
root@django-deployment-576dcff488-7b9g7 online test|# curl 10.245.51.193:55555
5 processes, 0 running, 0 queued[root@django-deployment-576dcff488-7b9g7 online test]# curl 10.245.51.193:55555
5 processes, 0 running, 0 queued[root@django-deployment-576dcff488-7b9g7 online test]# curl 10.245.51.193:55555
5 processes, 0 running, 0 queued[root@django-deployment-576dcff488-7b9g7 online test]# curl 10.245.51.193:55555

```

Using Minikube on Local System

```

Terminal - akshat0047@akshat0047-HP-Pavilion-15-Notebook-PC:~/Projects/kubernetes/Deployments
akshat0047@akshat0047-HP-Pavilion-15-Notebook-PC:~/Projects/kubernetes/Deployments$ sudo /usr/local/bin/minikube status
minikube
Type: Control Plane
host: Running
kubelet: Running
apiserver: Running
kubeconfig: Configured
akshat0047@akshat0047-HP-Pavilion-15-Notebook-PC:~/Projects/kubernetes/Deployments$ kubectl get nodes
NAME                                STATUS    ROLES    AGE     VERSION
akshat0047-hp-pavilion-15-notebook-pc Ready    master   19h    v1.18.3
akshat0047@akshat0047-HP-Pavilion-15-Notebook-PC:~/Projects/kubernetes/Deployments$ kubectl get configmap
NAME    DATA    AGE
yaksh-config 13       19h
akshat0047@akshat0047-HP-Pavilion-15-Notebook-PC:~/Projects/kubernetes/Deployments$ kubectl get secrets
NAME                                TYPE        DATA    AGE
db-secret                           opaque      2         19h
default-token-tgqkk                 kubernetes.io/service-account-token 3         19h
akshat0047@akshat0047-HP-Pavilion-15-Notebook-PC:~/Projects/kubernetes/Deployments$ kubectl get svc
NAME                                TYPE        CLUSTER-IP    EXTERNAL-IP  PORT(S)          AGE
django-svc                          NodePort    10.101.78.246 <none>        80:31000/TCP    19h
kubernetes                          ClusterIP   10.96.0.1     <none>        443/TCP          19h
yaksh-code                          ClusterIP   10.103.17.165 <none>        55555/TCP       19h
yaksh-db                             ClusterIP   10.106.45.60 <none>        3306/TCP        19h
akshat0047@akshat0047-HP-Pavilion-15-Notebook-PC:~/Projects/kubernetes/Deployments$ kubectl get deployment
NAME                                READY    UP-TO-DATE    AVAILABLE    AGE
codeserver-deployment              2/2      2              2            11m
db-deployment                      1/1      1              1            4m27s
django-deployment                  2/2      2              2            4m33s

```


StaYtus

Overview

Staytus is an open-source status app that can be installed on your server for showing the statuses of several services present on the server to your clients. This helps in keeping them informed about issues being faced at the moment and track these issues. Also, we can schedule maintenance sessions on the same which is shown as a notification on the site along with email notifications that are sent to users who have subscribed to us.

The Staytus is a very simple project with only limited features that are listed below:

General Settings

This consists of general information about the status site related to its name, email id associated, description, time zone, website-URL, protocol, etc.

Users

Lists users that are allowed to make changes in the admin panel, we can add users here

Design

Takes cover picture and logo as inputs to present it on the status site for a customized look

Services

List of services offered by the organization

Service Statuses

Applicable options of statuses that can be applied to the present service to depict a state

Service Groups

Groups of services to cluster them in a meaningful way

API Tokens

Key-Value pairs required to interact with the site using HTTP requests

E-Mail Templates

Pre-Designed email format to be sent to the subscriber when a change is triggered and the email option is chosen

Subscribers

This panel adds and lists the people who submit their email id to get notified about all the updates

Maintenance

Panel to list and add maintenance sessions

Issues

Panel to list and add issues along with edit the tracking stages according to the progress

Dashboard

Admin side view of the complete site services with buttons to directly add issues and maintenances sessions

Implementation

The installation instruction for Staytus can be found on the official blog(<https://blog.k.io/atech/install-staytus-tutorial>) as well as **mentioned below** though it is suggested to only opt for the method below if the official method doesn't work naturally as my way of accomplishing it involves complex changes that I had to make while debugging the installation.

I've also added a part to the fossee script for updating the statutes using HTTP API, this script is run as a cron job every hour and is present in this repository.

(<https://bitbucket.org/tslee/python.web.sites.checker/src/master/>)

The screenshot displays the AWDASH STATUS dashboard. At the top, it says "AWDASH STATUS" and "This site shows up down status of the links used in statistics." Below this, there are several status indicators: "Not responding" (last updated about 14 hours ago) and "Investigating" (in red), "Rolling new Update" (due to start in about 3 hours) and "Maintenance" (in yellow). At the bottom, there is a table showing the status of three domains: dwsim.fossee.in, esim.fossee.in, and focal.fossee.in, all of which are marked as "UP".

Domain	Status
dwsim.fossee.in	UP
esim.fossee.in	UP
focal.fossee.in	UP

Installation

- Install dependencies
 - \$ sudo -s
 - \$ dnf update
 - \$ dnf install @ruby:2.5
 - \$ ruby-devel nodejs git libmysqlclient-devel
 - \$ dnf groupinstall 'Development Tools'
 - \$ gem install bundler procodile
 - Create Database
 - \$ mysql -u root -p
 - CREATE DATABASE `staytus` CHARSET utf8mb4 COLLATE utf8mb4_unicode_ci;
 - CREATE USER `staytus`@`127.0.0.1` IDENTIFIED BY 'staytus';
 - GRANT ALL ON `staytus`.* TO `staytus`@`localhost`;
 - exit;
 - Add user for staytus
 - \$ useradd staytus
 - \$ passwd staytus
 - \$ usermod -aG wheel staytus
 - \$ su - staytus
 - Clone application and modify code to work with bcrypt 3.1.12
 - \$ git clone https://github.com/adamcooke/staytus
 - \$ cd staytus
 - \$ nano Gemfile
- Change: gem 'bcrypt' → gem 'bcrypt', '3.1.12'
- Build a new Gemfile.lock outside the directory
 - \$ cp Gemfile ../
 - \$ cp Gemfile.lock ../
 - \$ cd ..
 - \$ bundle install
 - Replace the old files with new one
 - \$ rm Gemfile Gemfile.lock
 - \$ cp ../Gemfile .
 - \$ cp ../Gemfile.lock .
 - Edit the code to cope up with the update

- \$ mkdir -p app/assets/config && echo '{}' > app/assets/config/manifest.js
- \$ nano config/initializers/assets.rb

Change: Rails.application.config.assets.precompile += [/\.jpg\z/, /\.png\z/,
 /\.gif\z/, /\.svg\z/]

To: Rails.application.config.assets.precompile << [".jpg", ".png", ".gif", ".svg",
 ".eot", ".woff", ".ttf"]

- Copy icons to be served
 - \$ sudo cp -r app/assets/images public/
- Install gems and add Database credentials
 - \$ bundle install --deployment --without development:test
 - \$ cp config/database.example.yml config/database.yml
 - \$ nano config/database.yml

- **Hostname:** localhost
- **Username:** staytus
- **Password:** staytus
- **Database:** staytus

- Compile assets and build database
 - \$ bundle exec rake staytus:build
 - \$ bundle exec rake staytus:install
- Test in development mode
 - procdile start --dev
- Start in background
 - procdile start
- Add Nginx Configuration as mentioned in Grafana Installation and Setup (**Sec. 1.2**)

AWDash

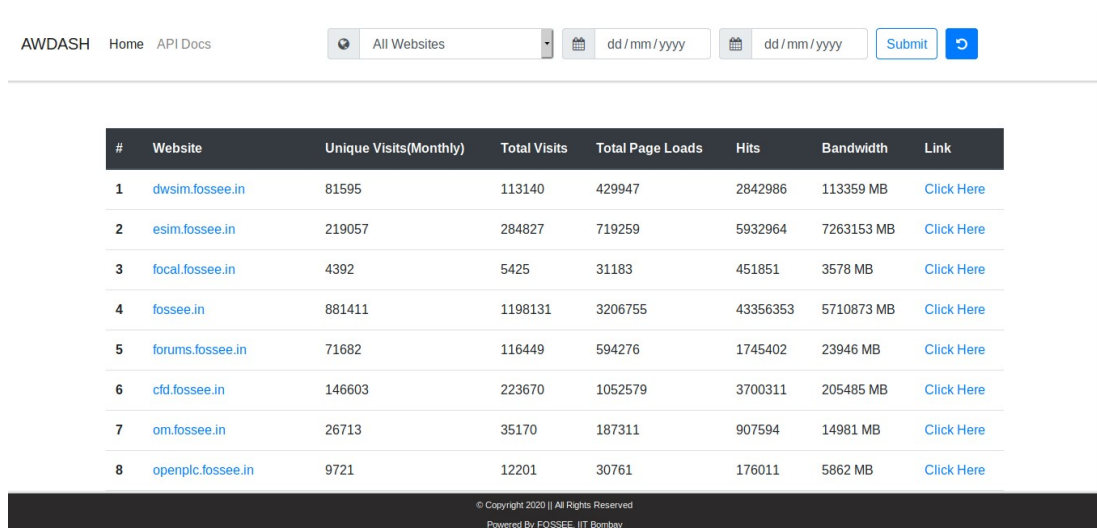
Overview

It is a Wrapper over AWStats which is an open-source monitoring that keeps track of metrics like hits, bandwidth, unique visits, visits, etc. The tool is capable of showing metrics for all the attributes according to months, daily, and weekly that are too complemented with visualization. Though we are able to filter these stats manually according to time, the tools don't provide us with a dashboard interface to monitor multiple sites with the required metrics at once.

AWDash is a solution to the same where we can add domains along with their AWStats link and log filename on the server. Written in PHP, the wrapper has an API structure that returns information as well as views. It presents us with a dashboard consisting of all the websites hosted on AWStats along with metrics Unique Visits, Visits, Hits, Total page loads, and bandwidth. We can also query these metrics for a time period using an integrated AJAX form which queries the API and the backend and returns contact the logs on each server using a PHP Script present on each one of them, in the end, providing us the result.

Implementation

Code - <https://github.com/FOSSEE/AWDash>



The screenshot shows the AWDash dashboard interface. At the top, there are navigation links for 'AWDASH', 'Home', and 'API Docs'. Below these, there is a search bar with a dropdown menu set to 'All Websites', two date input fields for filtering data by date (format dd/mm/yyyy), a 'Submit' button, and a refresh icon. The main content area features a table with the following data:

#	Website	Unique Visits(Monthly)	Total Visits	Total Page Loads	Hits	Bandwidth	Link
1	dwsim.fossee.in	81595	113140	429947	2842986	113359 MB	Click Here
2	esim.fossee.in	219057	284827	719259	5932964	7263153 MB	Click Here
3	focal.fossee.in	4392	5425	31183	451851	3578 MB	Click Here
4	fossee.in	881411	1198131	3206755	43356353	5710873 MB	Click Here
5	forums.fossee.in	71682	116449	594276	1745402	23946 MB	Click Here
6	ctd.fossee.in	146603	223670	1052579	3700311	205485 MB	Click Here
7	om.fossee.in	26713	35170	187311	907594	14981 MB	Click Here
8	openplc.fossee.in	9721	12201	30761	176011	5862 MB	Click Here

At the bottom of the dashboard, there is a footer with the text: © Copyright 2020 | All Rights Reserved. Powered By FOSSEE, IIT Bombay.

Reference

- <https://www.cyberciti.biz/>
- <https://www.digitalocean.com/community/tutorials>
- <https://linuxize.com/>
- <https://medium.com/>
- <https://gravitational.com/>
- <https://kubernetes.io/>