1. **Title** : Shark Information
2. **Description** :

A Simple node.js with frontend coded in express.js and mongodb web application which take input of shark name and shark character and stores it into database.

Deployed in kubernetes cluster with 3 node.js and 3 mongodb pods with health checkup in prometheus and prometheus alertmanager also grafana for time series analysis graphs.

Github Link - <https://github.com/Tikam02/Node-project-kube>

Docker Hub -<https://cloud.docker.com/u/timon02/repository/docker/timon02/node-replicas>

Live Web Application - <http://174.138.122.173/>

1. **Installation & Instructions:**

* **For Running The web app locally:**

1. $ cd /local/node
2. $ npm install
3. $ node app.js

* **For Running web App in Docker**

1. $ mv /local/node ~./home //move node app to home directory because db setup at home for time to time restart.
2. $ docker build -t dockerhub-username/node-project .
3. $ docker images -a //check for build image
4. $ docker push username/repository:tag
5. $ docker-compose up --build
6. $ Open ---> localhost:3000

* **For Running WebApp Clusters and Monitoring with Prometheus and Grafana**

1. Download cluster Config file from owner of cluster and connect to the cluster:

$ cd ~/.kube && kubectl --kubeconfig="clustername-kubeconfig.yaml" get nodes

1. Get kubectl nodes:

$ kubect get nodes

1. Get pods:

$ kubectl get pods

1. Get Services :

$ kubectl get services

1. Get Deployments:

$ kubectl get services

1. See MongoDb pods

$ kubectl exec -it mongo-mongodb-replicaset-0 -- mongo -u tikam -p --authenticationDatabase admin

1. Port Forward Grafans to monitor in localhost:

$ kubectl port-forward -n monitoring svc/doks-cluster-monitoring-grafana 8000:80

1. Port Forward Prometheus to monitor in localhost:

$ kubectl port-forward -n monitoring svc/doks-cluster-monitoring-pr-prometheus 9090:9090

1. listing running Services in the monitoring namespace:

$ kubectl get svc -n monitoring

1. **Use cases and edge Conditions:**

Web App Use Cases:

* Simple landing page
* Enter Shark Name
* Enter Shark Character
* [ These two objects will be saved in mongodb]

Web App edge Condition:

* One of the mongodb cluster has some issue so it takes time to save data

Kubernetes use cases:

### Optimized Resource Usage with Namespaces

* Namespaces create a virtually separated cluster inside the real cluster.

### Self-Healing and Scaling Services

* Replication and Scaling

## Simple Deployment of Stateless Applications

## Deploy Stateful Data Services

Kubernetes Edge Condition :

* For api services access to other services like spekt8 and helm we need to get RBAC ,
* Pods sometimes need a manual start/restart before they start working as intended. This can happen in certain situations such as when running near full capacity.
* Uses a different configuration and YAML definitions and API as it wasn’t designed only for docker clustering

1. **Workflow:**

* Nodejs App -> Docker -> DockerHub -> Deployment.yaml +Services.yaml -> kubernetes -> Prometheus -> Grafana -> Spekt8
* Created web app with node and express.js,connected to the mongodb run it locally.
* Make dockerfile to run node app
* Make docker-compose.yaml file to run node app and mongodb
* Push the built docker image to docker hub repository
* Installed and setup kubernetes CLI - kubectl and kubernetes package manager helm CLI.

## Configuring the MongoDB Helm Chart and Creating a Deployment

## Creating a Custom Application Chart and Configuring Parameters

* Setting up helm in cluster because of RBAC

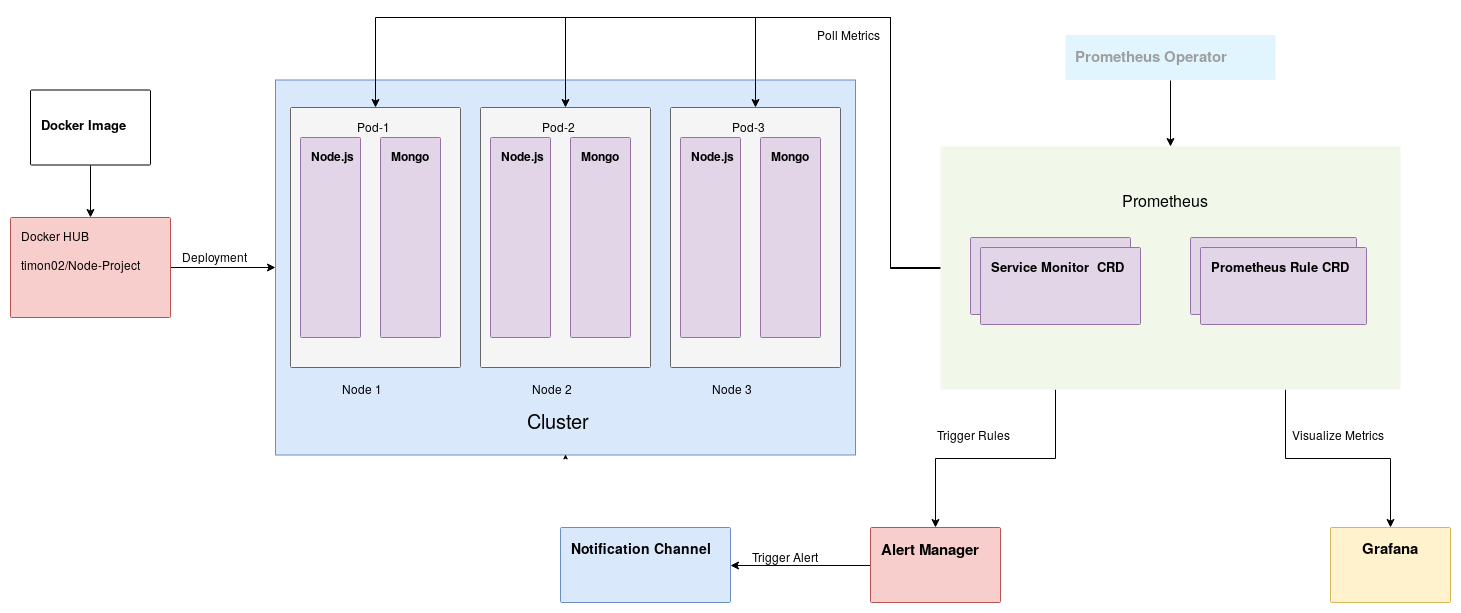
## Installing the prometheus-operator Chart via helm

## Accessing Grafana and Exploring Metrics Data

* forward local port 8000 to port 80 of the doks-cluster-monitoring-grafana for monitoring in localhost

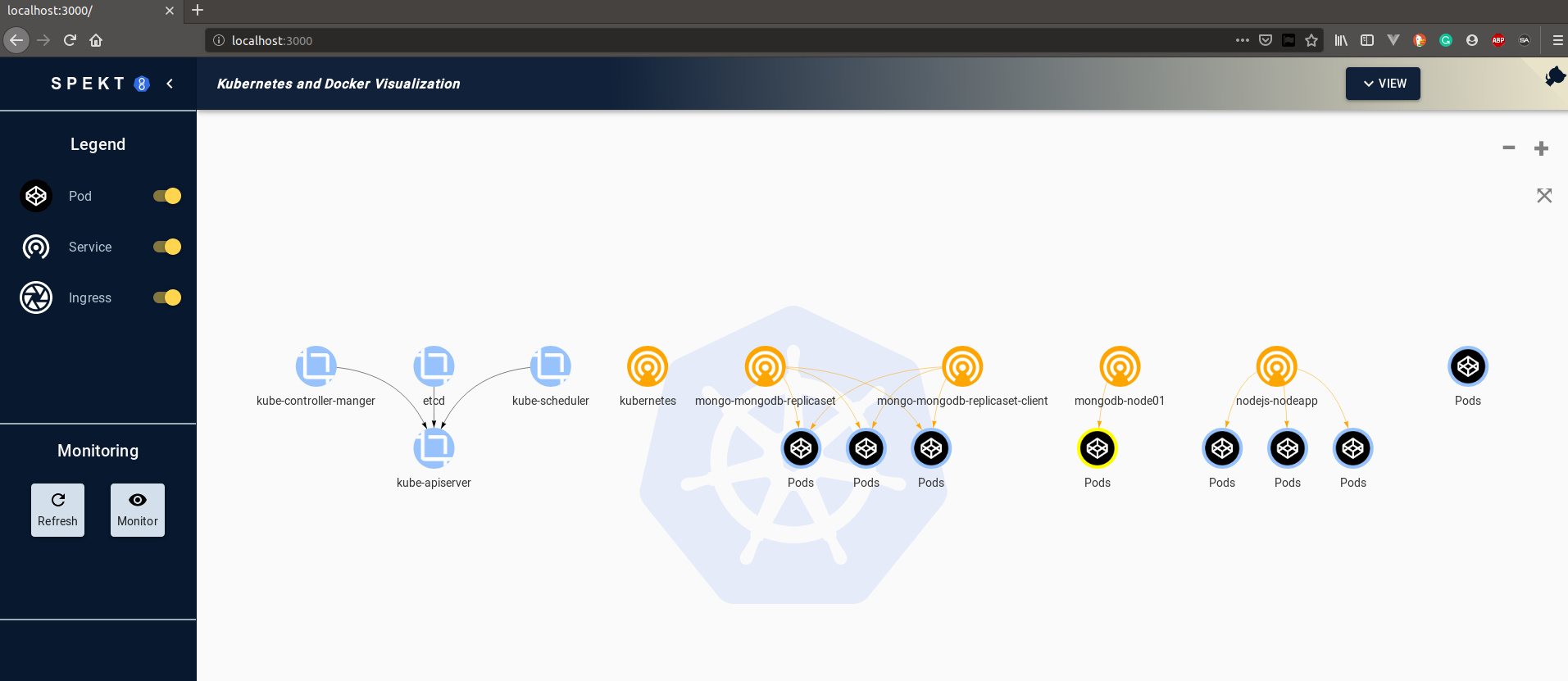
## Accessing Prometheus and Alertmanager

* We are going to forward local port 9093 to port 9093 of the doks-cluster-monitoring-pr-alertmanager Service.

****

Used helm package manager and helm charts for deploying and setting up prometheus and grafana.

When creating multi-service deployments with Kubernetes, many developers opt to use the Helm package manager. Helm streamlines the process of creating multiple Kubernetes resources by offering charts and templates that coordinate how these objects interact. It also offers pre-packaged charts for popular open-source projects.



**Spekt8 Service to monitor the architecture**

1. **Data formats & Reporting:**

Kubernetes - YAML files

MongoDb - json object | Binary JSON (BSON)

Node.js - Json object

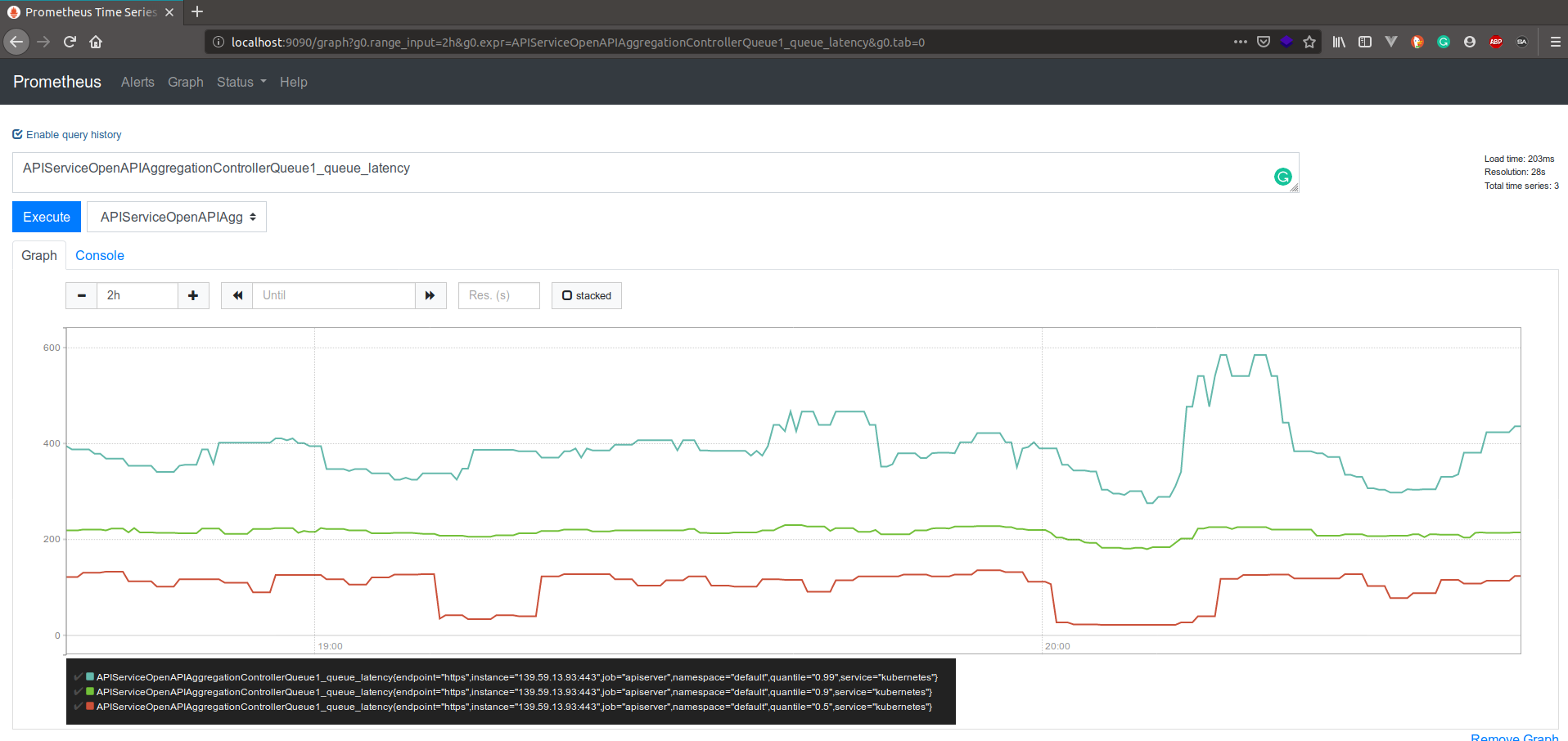
1. **Performance & Scaling:**

Performance - For performance basically if using digitalocean’s kubernetes service we get metric insights with graphs which displays how much resources have been used,CPU, memory, I/O, network, etc.



Secondly, Deploying prometheus we get each cluster’s nodes, pods and many more features.

**Prometheus** is a time series database and monitoring tool that works by polling metrics endpoints and scraping and processing the data exposed by these endpoints.

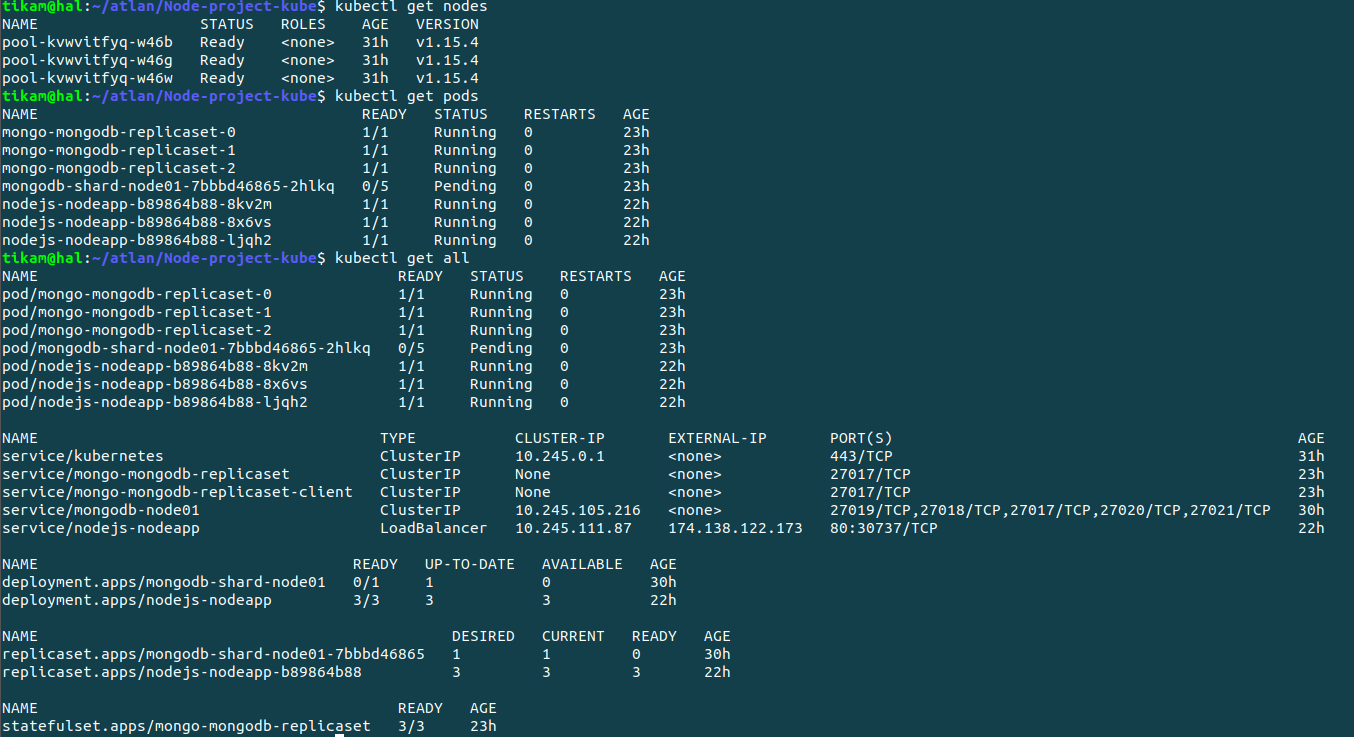


**Grafana** is a data visualization and analytics tool that allows you to build dashboards and graphs for your metrics data.



**Scaling:**

* I followed the basic idea of What to scale? How to scale ? And when to scale?
* Since Digitalocean doesn’t have autoscaler like Google Kubernetes, so we have to implement autoscaler like Horizontal Pod Autoscaler or manual scaling from command:
* $ kubectl scale --replicas=10 deployment/app
* The Horizontal Pod Autoscaler automatically scales the number of pods in a replication controller, deployment or replica set based on observed CPU utilization (or, with custom metrics support, on some other application-provided metrics).
* Right now the deployed app cluster has a Pool of 3 nodes and
* 3 mongodb pods and 3 nodejs app.



* If any of the resource fails or crashes we get the information from prometheus alert manager and according to that I will scale the app, either increase nodes or increase pods.

1. **Unresolved Issues:**

* Cluster config file setting up everytime for a new cluster