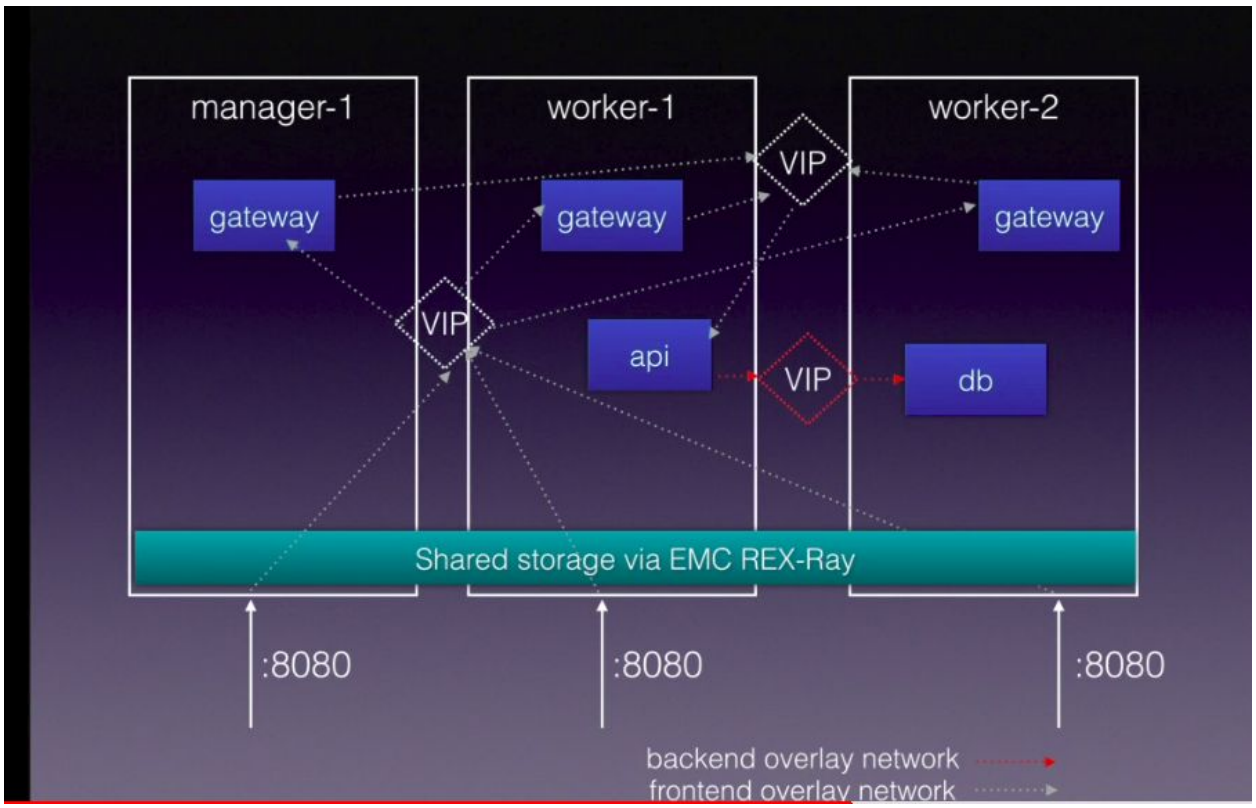
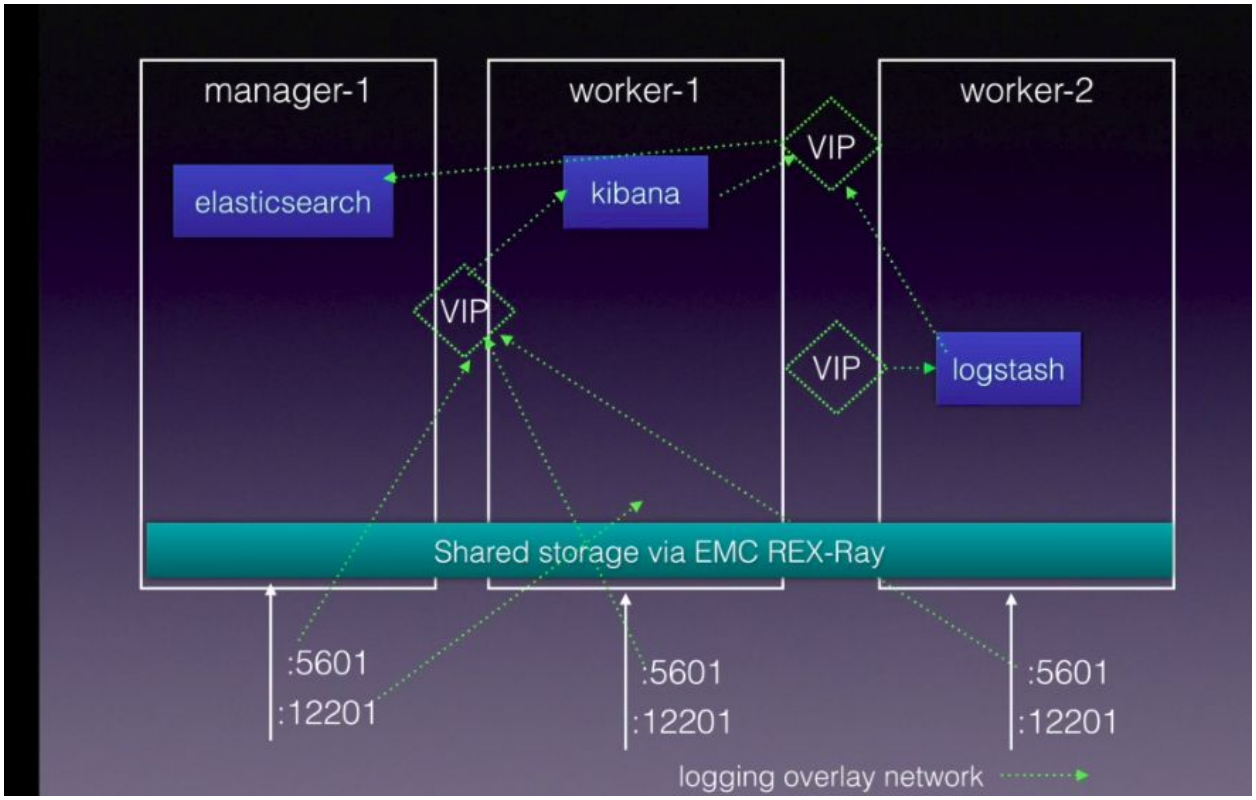
DEPLOYING A WEB APPLICATION USING DOCKER SWARM

1. We have used Docker swarm mode to deploy a web application
2. 3 vms are created viz.Manager 1,worker1 and worker 2
3. We have created shared storage between the 3 vms.via EMV REX-Ray (which has good support for virtual box)
4. First we created a database service which is a mongoDB. It is over backend overlay network
5. The next service is the API, on the front-end overlay network
6. Next service is the gateway or NGINX service which serves as the basis for static content on front-end overlay network.
7. Each of the services will be given a virtual IP that allows the docker swarm to load balance between services
8. Front end nginx service will be available on port 8080
9. We have used ELK which is a popular centralized logging system
10. The logstash service will connect to elasticsearch service(logstash collects the log and sends it to elasticsearch) .Kibana pulls in the visualization of the logs by connecting to elasticsearch
11. Kibana is accessed on port 5601
12. Logstash is on port 12201

WEB APPLICATION ARCHITECTURE



ELK STACK ARCHITECTURE



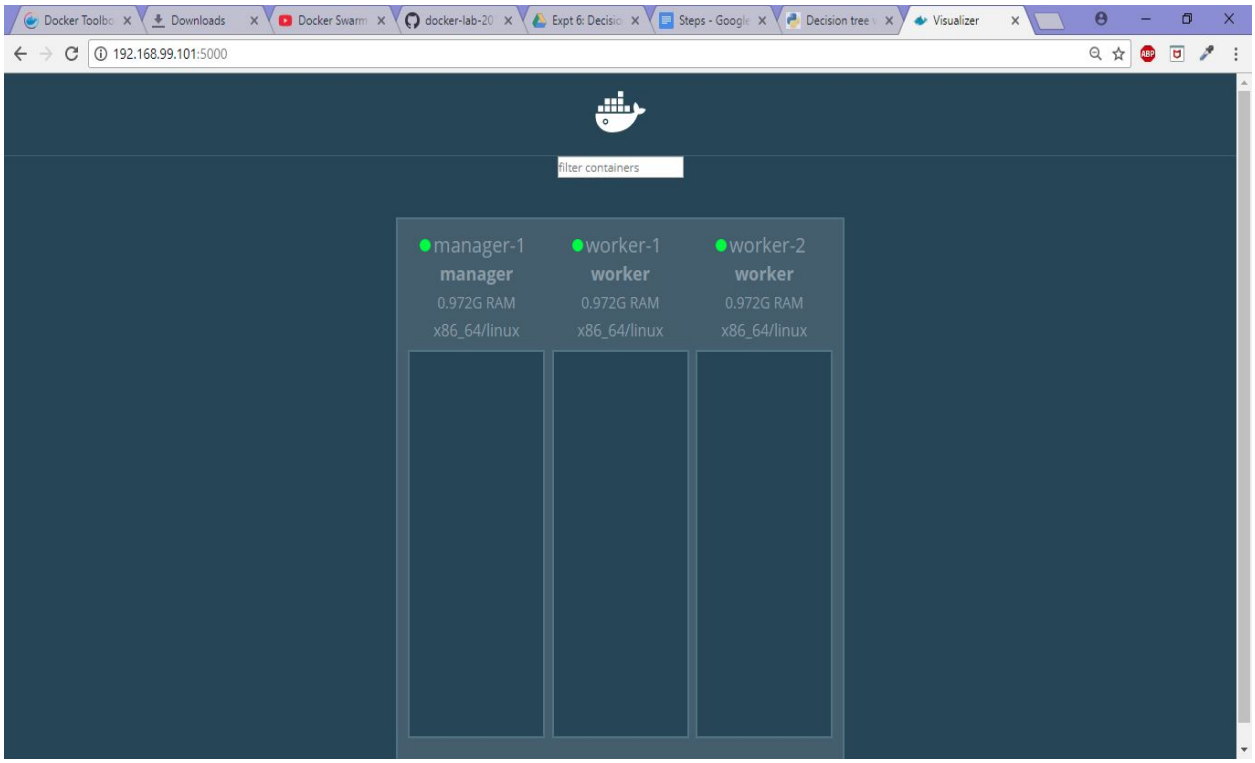
Setup Docker Swarm

1.Use docker-machine to provision Docker engine on three nodes (i.e. vms).
2.Visualizer is an easy way to look at the docker swarm details.
3.Target the manager-1, and make it a swarm manager.
4.Join the worker nodes to the swarm. To do this, target each machine in order, and issue the docker swarm join command that was output by the last step.
5.A simple 3 node swarm is now setup.

```
$ docker-machine create -d virtualbox manager-1
Running pre-create checks...
Creating machine...
(manager-1) Copying C:\Users\DELL_PC\.docker\machine\cache\boot2docker.iso to C:\Users\DELL_PC\.docker\machine\machines\manager-1\boot2docker.iso...
(manager-1) Creating VirtualBox VM...
(manager-1) Creating SSH key...
(manager-1) Starting the VM...
(manager-1) Check network to re-create if needed...
(manager-1) Windows might ask for the permission to configure a dhcp server. Sometimes, such confirmation window is minimized in the taskbar.
(manager-1) Waiting for an IP...
Waiting for machine to be running, this may take a few minutes...
Detecting operating system of created instance...
Waiting for SSH to be available...
Detecting the provisioner...
Provisioning with boot2docker...
Copying certs to the local machine directory...
Copying certs to the remote machine...
Setting Docker configuration on the remote daemon...
Checking connection to Docker...
Docker is up and running!
To see how to connect your Docker Client to the Docker Engine running on this virtual machine, run: C:\Program Files\Docker Toolbox\docker-machine.exe env manager-1
```

```
$ docker-machine create -d virtualbox worker-1
Running pre-create checks...
Creating machine...
(worker-1) Copying C:\Users\DELL_PC\.docker\machine\cache\boot2docker.iso to C:\Users\DELL_PC\.docker\machine\machines\worker-1\boot2docker.iso...
(worker-1) Creating VirtualBox VM...
(worker-1) Creating SSH key...
(worker-1) Starting the VM...
(worker-1) Check network to re-create if needed...
(worker-1) Windows might ask for the permission to configure a dhcp server. Sometimes, such confirmation window is minimized in the taskbar.
(worker-1) Waiting for an IP...
Waiting for machine to be running, this may take a few minutes...
Detecting operating system of created instance...
Waiting for SSH to be available...
Detecting the provisioner...
Provisioning with boot2docker...
Copying certs to the local machine directory...
Copying certs to the remote machine...
Setting Docker configuration on the remote daemon...
Checking connection to Docker...
Docker is up and running!
To see how to connect your Docker Client to the Docker Engine running on this virtual machine, run: C:\Program Files\Docker Toolbox\docker-machine.exe env worker-1

$ docker-machine create -d virtualbox worker-2
Running pre-create checks...
Creating machine...
(worker-2) Copying C:\Users\DELL_PC\.docker\machine\cache\boot2docker.iso to C:\Users\DELL_PC\.docker\machine\machines\worker-2\boot2docker.iso...
(worker-2) Creating VirtualBox VM...
(worker-2) Creating SSH key...
(worker-2) Starting the VM...
(worker-2) Check network to re-create if needed...
(worker-2) Windows might ask for the permission to configure a dhcp server. Sometimes, such confirmation window is minimized in the taskbar.
(worker-2) Waiting for an IP...
Waiting for machine to be running, this may take a few minutes...
Detecting operating system of created instance...
Waiting for SSH to be available...
Detecting the provisioner...
Provisioning with boot2docker...
Copying certs to the local machine directory...
Copying certs to the remote machine...
Setting Docker configuration on the remote daemon...
Checking connection to Docker...
Docker is up and running!
To see how to connect your Docker Client to the Docker Engine running on this virtual machine, run: C:\Program Files\Docker Toolbox\docker-machine.exe env worker-2
```

```
$ docker-machine ls --filter=driver=virtualbox
NAME        ACTIVE   DRIVER       STATE     URL                         SWARM   DOCKER        ERRORS
default     *        virtualbox   Running   tcp://192.168.99.100:2376           v18.03.0-ce
manager-1   -        virtualbox   Running   tcp://192.168.99.101:2376           v18.03.0-ce
worker-1    -        virtualbox   Running   tcp://192.168.99.102:2376           v18.03.0-ce
worker-2    -        virtualbox   Running   tcp://192.168.99.103:2376           v18.03.0-ce

$ docker swarm init --advertise-addr $(docker-machine ip manager-1)
Swarm initialized: current node (oeo5jfenau0a0junovdssnfm3) is now a manager.

To add a worker to this swarm, run the following command:

    docker swarm join --token SWMTKN-1-4v77rcyljnyfuec5ywg7qx8uozd2x1n05k6hqol5avqnzrvocq-44h5lie8tcky8oft38sbouoir 192.168.99.101:2377

To add a manager to this swarm, run 'docker swarm join-token manager' and follow the instructions.


DELL_PC@DESKTOP-1C8FPRM MINGW64 ~
$ eval $(docker-machine env worker-1)

DELL_PC@DESKTOP-1C8FPRM MINGW64 ~
$ docker swarm join --token SWMTKN-1-4v77rcyljnyfuec5ywg7qx8uozd2x1n05k6hqol5avqnzrvocq-44h5lie8tcky8oft38sbouoir 192.168.99.101:2377
This node joined a swarm as a worker.

DELL_PC@DESKTOP-1C8FPRM MINGW64 ~
$ eval $(docker-machine env worker-2)

DELL_PC@DESKTOP-1C8FPRM MINGW64 ~
$ docker swarm join --token SWMTKN-1-4v77rcyljnyfuec5ywg7qx8uozd2x1n05k6hqol5avqnzrvocq-44h5lie8tcky8oft38sbouoir 192.168.99.101:2377
This node joined a swarm as a worker.
```



```
DELL_PC@DESKTOP-1C8FPRM MINGW64 ~
$ eval $(docker-machine env manager-1)

DELL_PC@DESKTOP-1C8FPRM MINGW64 ~
$ docker node ls
ID                            HOSTNAME            STATUS              AVAILABILITY        MANAGER STATUS
oeo5jfenau0a0junovdssnfm3 *   manager-1           Ready               Active              Leader
ienhiqg2eappuvi7p9nooc7ph     worker-1            Ready               Active
fqfq2qkptoz6ufjrm2z5vgyaa     worker-2            Ready               Active


$ docker info
Containers: 4
 Running: 2
 Paused: 0
 Stopped: 2
Images: 2
Server Version: 18.03.0-ce
Storage Driver: aufs
 Root Dir: /mnt/sda1/var/lib/docker/aufs
 Backing Filesystem: extfs
 Dirs: 34
 Dirperm1 Supported: true
Logging Driver: json-file
Cgroup Driver: cgroupfs
Plugins:
 Volume: local
 Network: bridge host macvlan null overlay
 Log: awslogs fluentd gcplogs gelf journald json-file logentries splunk syslog
Swarm: active
 NodeID: oeo5jfenau0a0junovdssnfm3
 Is Manager: true
 ClusterID: g0wko79jlzeugh5taat1nwjap
 Managers: 1
 Nodes: 3
 Orchestration:
  Task History Retention Limit: 5
 Raft:
  Snapshot Interval: 10000
  Number of Old Snapshots to Retain: 0
  Heartbeat Tick: 1
  Election Tick: 3
 Dispatcher:
  Heartbeat Period: 5 seconds
 CA Configuration:
  Expiry Duration: 3 months
  Force Rotate: 0
 Autolock Managers: false
 Root Rotation In Progress: false
 Node Address: 192.168.99.101
 Manager Addresses:
  192.168.99.101:2377
Runtimes: runc
Default Runtime: runc
Init Binary: docker-init
containerd version: cfd04396dc68220d1cecbe686a6cc3aa5ce3667c
runc version: 4fc53a81fb7c994640722ac585fa9ca548971871
init version: 949e6fa
Security Options:
 seccomp
  Profile: default
Kernel Version: 4.9.89-boot2docker
Operating System: Boot2Docker 18.03.0-ce (TCL 8.2.1); HEAD : 404ee40 - Thu Mar 22 17:12:23 UTC 2018
OSType: linux
Architecture: x86_64
CPUs: 1
Total Memory: 995.6MiB
Name: manager-1
ID: PAUV:FHDN:76CE:KPQY:76KK:CPXZ:FU5K:S45V:MQA3:U5UE:CRBQ:TIO3
Docker Root Dir: /mnt/sda1/var/lib/docker
Debug Mode (client): false
Debug Mode (server): false
Registry: https://index.docker.io/v1/
Labels:
 provider=virtualbox
Experimental: false
Insecure Registries:
 127.0.0.0/8
Live Restore Enabled: false
```

Setup shared storage between VMs

1. Install the REX-ray server on manager-1.
2. Install the REX-ray client on worker-1
3. Install the REX-ray client on worker-2

```
$ docker-machine ssh worker-2 \
> "curl -sSL https://dl.bintray.com/emccode/rexray/install | sh -s -- stable 0.5.1"
time="2018-03-23T18:02:17Z" level=info msg="making libStorage directory" mustPerm=true path="/etc/libstorage" perms=-rwxr-xr-x
time="2018-03-23T18:02:17Z" level=info msg="making libStorage directory" mustPerm=true path="/var/lib/libstorage" perms=-rwxr-xr-x
time="2018-03-23T18:02:17Z" level=info msg="making libStorage directory" mustPerm=true path="/var/log/libstorage" perms=-rwxr-xr-x
time="2018-03-23T18:02:17Z" level=info msg="making libStorage directory" mustPerm=true path="/var/run/libstorage" perms=-rwxr-xr-x

rexray has been installed to /usr/bin/rexray

time="2018-03-23T18:02:17Z" level=info msg="making libStorage directory" mustPerm=false path="/home/docker/.libstorage" perms=-rwxr-xr-x
time="2018-03-23T18:02:17Z" level=info msg="making libStorage directory" mustPerm=true path="/home/docker/.libstorage/etc/libstorage" perms=-rwxr-xr-x
time="2018-03-23T18:02:17Z" level=info msg="making libStorage directory" mustPerm=true path="/home/docker/.libstorage/var/lib/libstorage" perms=-rwxr-xr-x
time="2018-03-23T18:02:17Z" level=info msg="making libStorage directory" mustPerm=true path="/home/docker/.libstorage/var/log/libstorage" perms=-rwxr-xr-x
time="2018-03-23T18:02:17Z" level=info msg="making libStorage directory" mustPerm=true path="/home/docker/.libstorage/var/run/libstorage" perms=-rwxr-xr-x
REX-Ray
-------
Binary: /usr/bin/rexray
SemVer: 0.5.1
OsArch: Linux-x86_64
Branch: v0.5.1
Commit: e2c3165ce26cdbd8be76c3f6bde31db8618ae306
Formed: Wed, 14 Sep 2016 19:59:58 UTC

libStorage
----------
SemVer: 0.2.1
OsArch: Linux-x86_64
Branch: v0.5.1
Commit: 2f7210c90252a4e1c9fb011ba29158cb0facb516
Formed: Wed, 14 Sep 2016 19:58:39 UTC
```

```
DELL_PC@DESKTOP-1C8FPRM MINGW64 ~
$ docker-machine ssh worker-2 \
> "sudo tee -a /etc/rexray/config.yml << EOF
> rexray:
>   logLevel: warn
> libstorage:
>   host:    tcp://${REXRAY_SERVER}:7979
>   service: virtualbox
> "
rexray:
  logLevel: warn
libstorage:
  host:    tcp://192.168.99.101:7979
  service: virtualbox

$ docker-machine ssh worker-2 "sudo rexray service start"
Starting REX-Ray...SUCCESS!

  The REX-Ray daemon is now running at PID -1. To
  shutdown the daemon execute the following command:

    sudo /usr/bin/rexray stop
```

Build Docker Images

1. Log into Docker Hub
2. Build and push the database (mongodb), static web server (nginx) and REST backend (strongloop)
3. deploy centralized logging using the ELK stack .

DB DockerFile

```
FROM tutum/mongodb
ENV JOURNALING no
```

Gateway DockerFile

```
FROM nginx
RUN apt-get -y update  && apt-get install -y curl nano
COPY sample_app_nginx.conf /etc/nginx/nginx.conf
```

Strongloop DockerFile

```
FROM sgdpro/nodeslc

COPY ./app/package.json /home/strongloop/app/package.json
WORKDIR /home/strongloop/app
RUN npm install

COPY ./app /home/strongloop/app
VOLUME /home/strongloop/app
# ENV NODE_ENV production
ENTRYPOINT ["./start.sh"]
```

Elasticsearch DockerFile

```
FROM elasticsearch:2.4.0
RUN apt-get -y update  && apt-get install -y curl nano
```

Kibana DockerFile

```
FROM kibana:4.6.0
RUN apt-get -y update  && apt-get install -y curl nano
ENV ELASTICSEARCH_URL=http://elasticsearch:9200
```

Logstash DockerFile

```
FROM logstash:2.4
COPY ./logstash.conf /opt/logstash/conf.d/logstash.conf
```

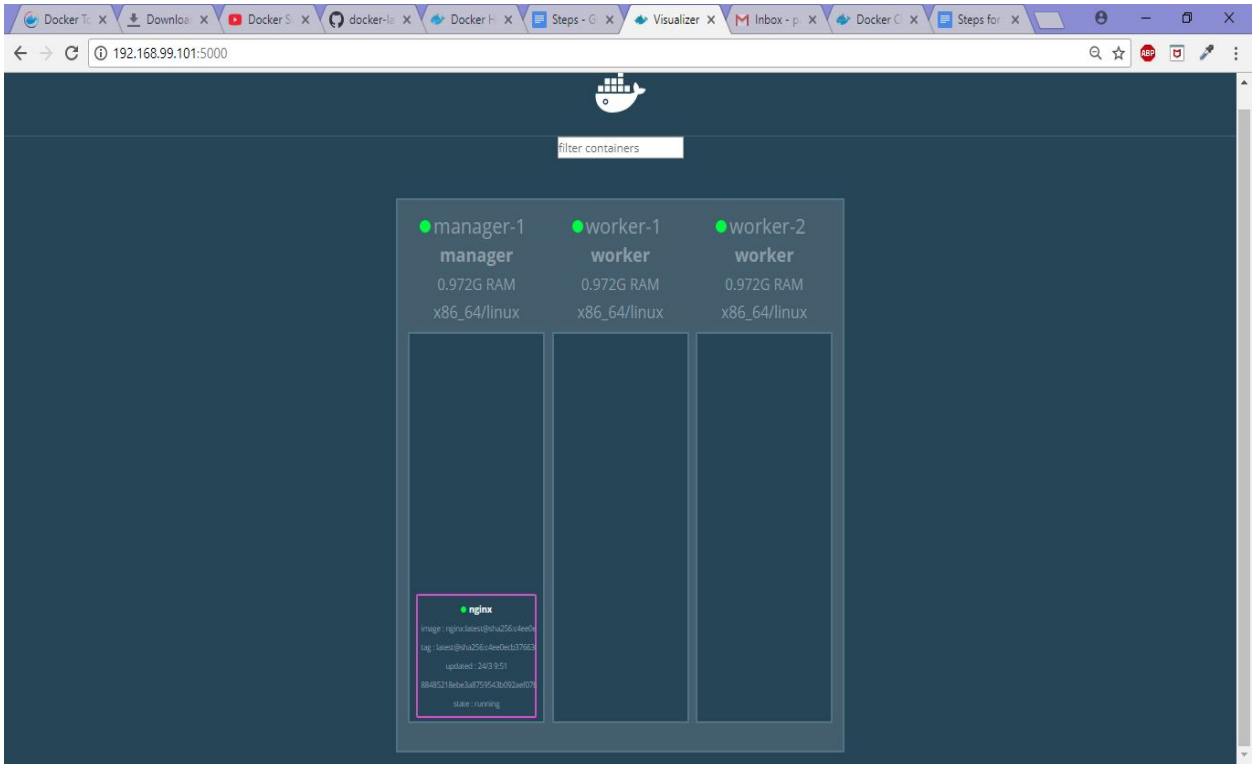
RUN apt-get -y update && apt-get install -y curl nano
EXPOSE 5000 5000/udp 12201 12201/udp

```
$ docker-machine ssh worker-2 "sudo rexray service start"
Starting REX-Ray...SUCCESS!

  The REX-Ray daemon is now running at PID -1. To
  shutdown the daemon execute the following command:

    sudo /usr/bin/rexray stop

DELL_PC@DESKTOP-1C8FPRM MINGW64 ~
$ eval $(docker-machine env manager-1)

DELL_PC@DESKTOP-1C8FPRM MINGW64 ~
$ docker service create --replicas 1 \
> --name nginx \
> -p 8080:80 \
> --mount type=volume,source=hellopersistence,target=/usr/share/nginx/html,volume-driver=rexray \
> nginx
s23r7nwr2ugh80ezgbe62vrwj
overall progress: 1 out of 1 tasks
1/1: running   [==================================================>]
verify: Service converged
```

```
$ docker service inspect --pretty nginx

ID:             s23r7nwr2ugh80ezgbe62vrwj
Name:           nginx
Service Mode:   Replicated
 Replicas:      1
Placement:
UpdateConfig:
 Parallelism:   1
 On failure:    pause
 Monitoring Period: 5s
 Max failure ratio: 0
 Update order:      stop-first
RollbackConfig:
 Parallelism:   1
 On failure:    pause
 Monitoring Period: 5s
 Max failure ratio: 0
 Rollback order:    stop-first
ContainerSpec:
 Image:         nginx:latest@sha256:c4ee0ecb376636258447e1d8effb56c09c75fe7acf756bf7c13efadf38aa0aca
Mounts:
  Target = /usr/share/nginx/html
   Source = hellopersistence
   ReadOnly = false
   Type = volume
Resources:
Endpoint Mode:  vip
Ports:
 PublishedPort = 8080
  Protocol = tcp
  TargetPort = 80
  PublishMode = ingress
```

```
$ docker service ls
ID                  NAME                MODE                REPLICAS            IMAGE               PORTS
s23r7nwr2ugh        nginx               replicated          1/1                 nginx:latest        *:8080->80/tcp
```

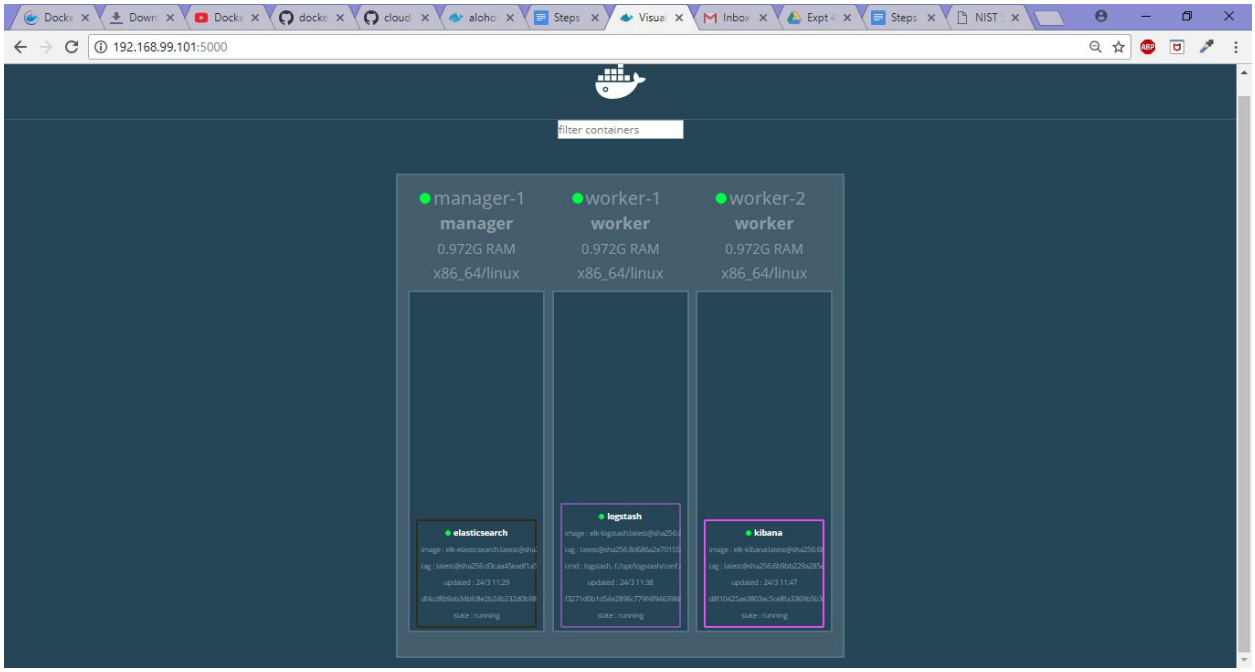
```
$ docker service ps nginx
ID                  NAME                IMAGE               NODE                DESIRED STATE       CURRENT STATE               ERROR               PORTS
li4kmlohyexr        nginx.1             nginx:latest        worker-2            Running             Running about a minute ago
```

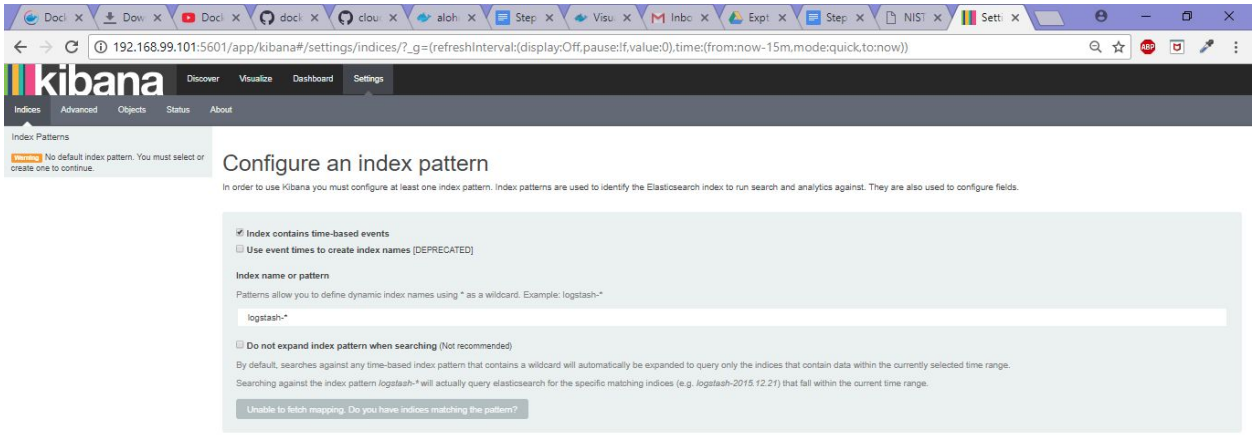


Run the ELK stack

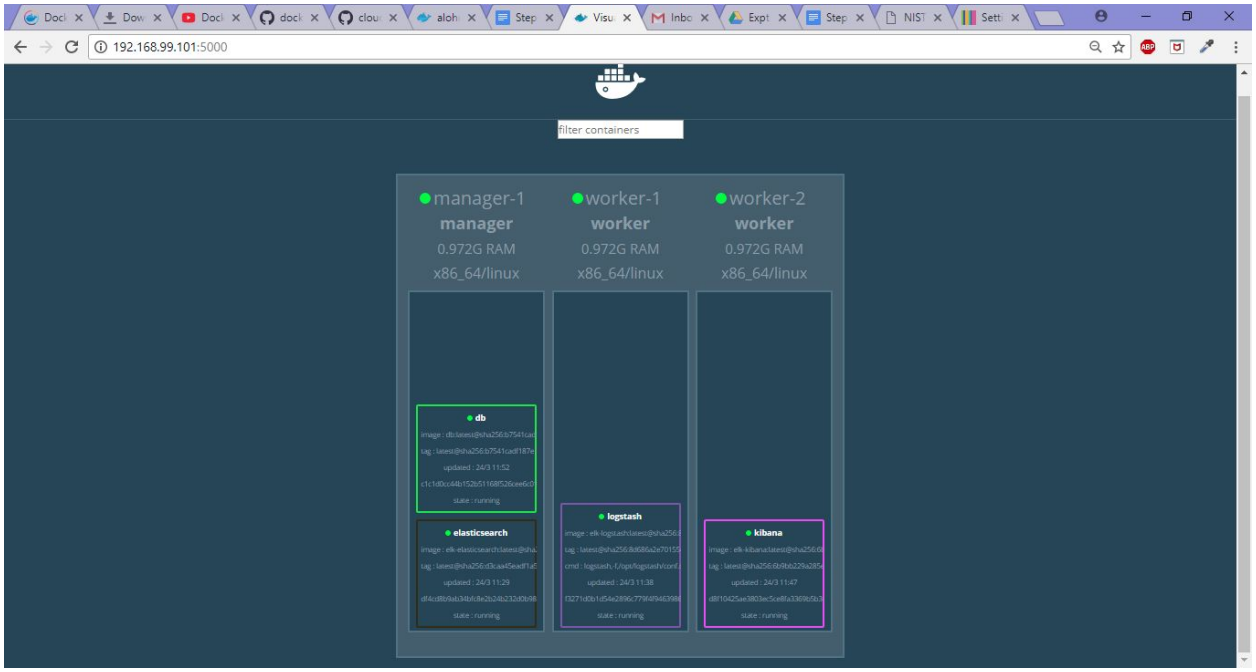
1. Create a network that all of the nodes that require logging will connect to.
2. Create a docker Volume for elastic search data(using ReX RAY DRIVER)
3. Now we have a named network and a named volume.
4. Create the service elastic search.create 1 replica.Connect it to the logging network.(It selected worker 2 to deploy the service on on its own)
5. Create the logstash container.It will go ahead and choose a node.(Manager 1 in our case)
6. Create the Kibana service.

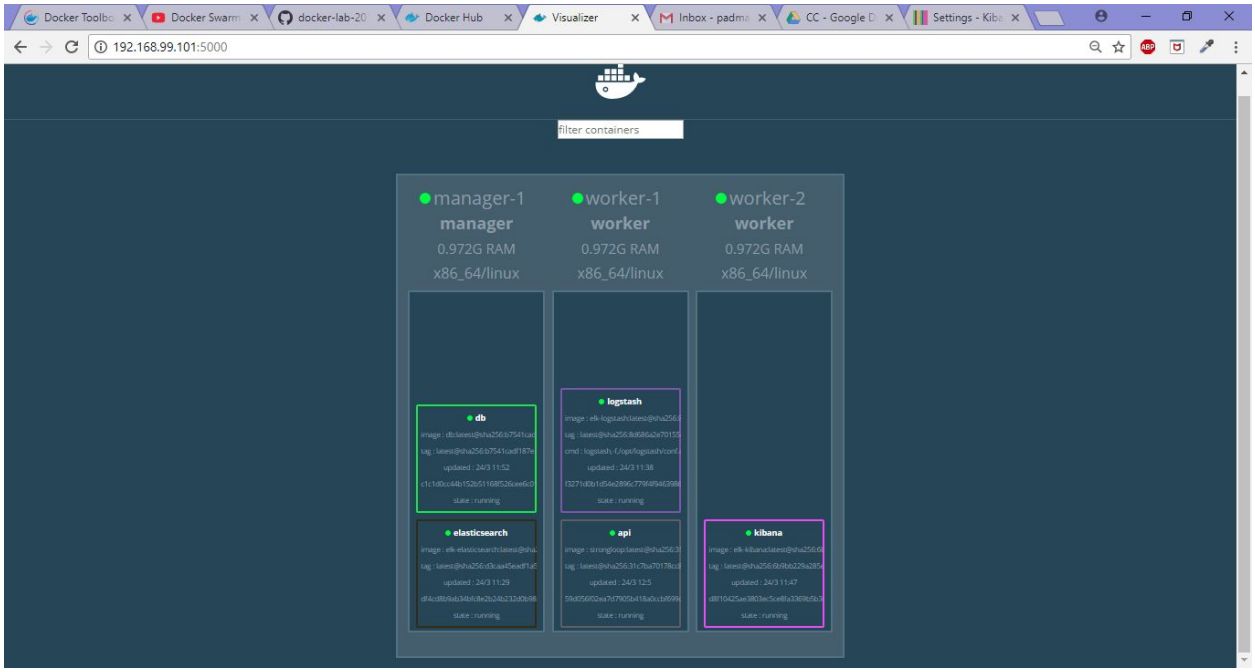
```
DELL_PC@DESKTOP-1C8FPRM MINGW64 ~
$ docker service create \
> --name elasticsearch \
> --replicas 1 \
> --network logging \
> -e LOGSPOUT=ignore \
> --mount type=volume,source=esdata,target=/usr/share/elasticsearch/data,volume-driver=rexray \
> alohomora/elk-elasticsearch:latest
lgcbd5nlfxtkal7f28o2x9wfu
overall progress: 1 out of 1 tasks
1/1: running   [==================================================>]
verify: Service converged
```

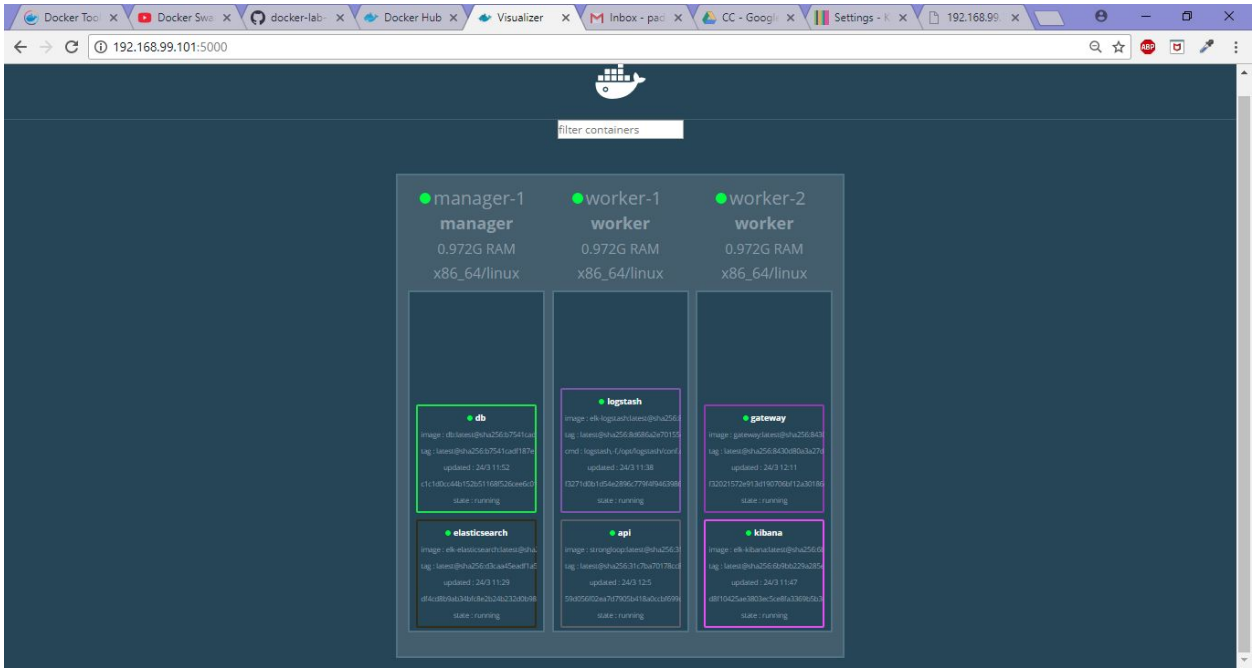

192.168.99.101:5000
Filter containers
manager-1
manager
0.972G RAM
x86_64/linux
worker-1
worker
0.972G RAM
x86_64/linux
worker-2
worker
0.972G RAM
x86_64/linux
elasticsearch
logstash
kibana

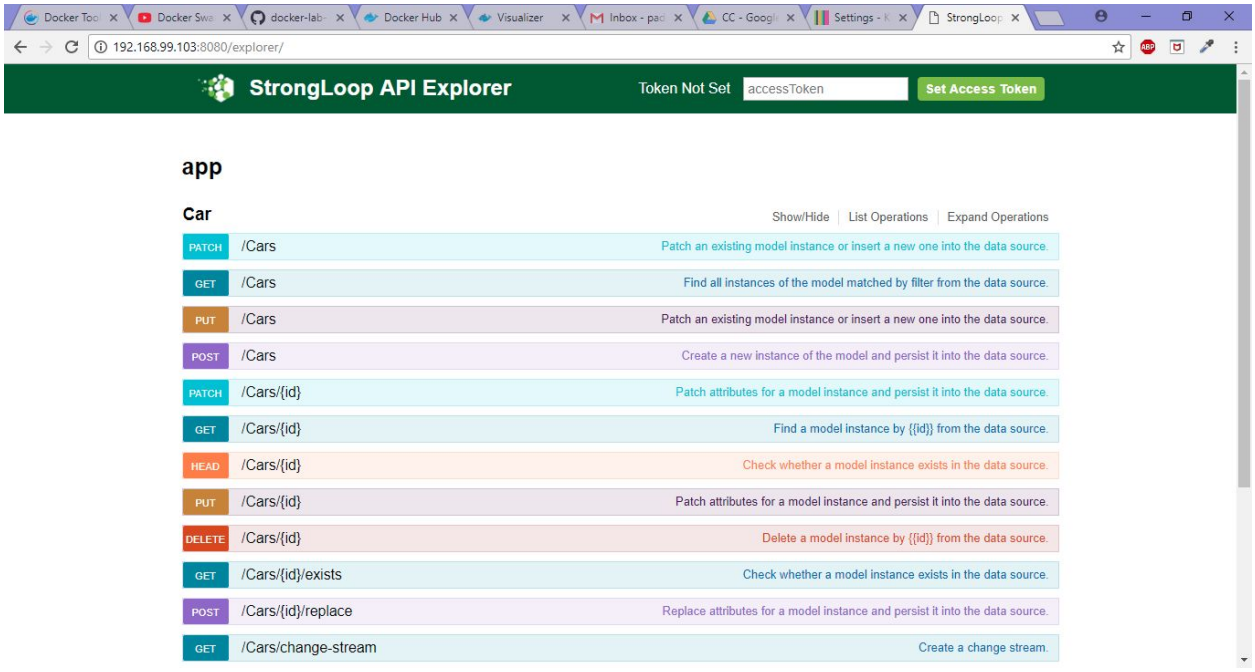

192.168.99.101:5601/app/kibana#/settings/indices/?_g=(refreshInterval:(display:Off,pause:!f,value:0),time:(from:now-15m,mode:quick,to:now))
kibana
Discover
Visualize
Dashboard
Settings
Indices
Advanced
Objects
Status
About
Index Patterns
Warning No default index pattern. You must select or create one to continue.
Configure an index pattern
In order to use Kibana you must configure at least one index pattern. Index patterns are used to identify the Elasticsearch index to run search and analytics against. They are also used to configure fields.
Index contains time-based events
Use event times to create index names [DEPRECATED]
Index name or pattern
Patterns allow you to define dynamic index names using * as a wildcard. Example: logstash-*
logstash-*
Do not expand index pattern when searching (Not recommended)
By default, searches against any time-based index pattern that contains a wildcard will automatically be expanded to query only the indices that contain data within the currently selected time range.
Searching against the index pattern logstash-* will actually query elasticsearch for the specific matching indices (e.g. logstash-2015.12.21) that fall within the current time range.
Unable to fetch mapping. Do you have indices matching the pattern?

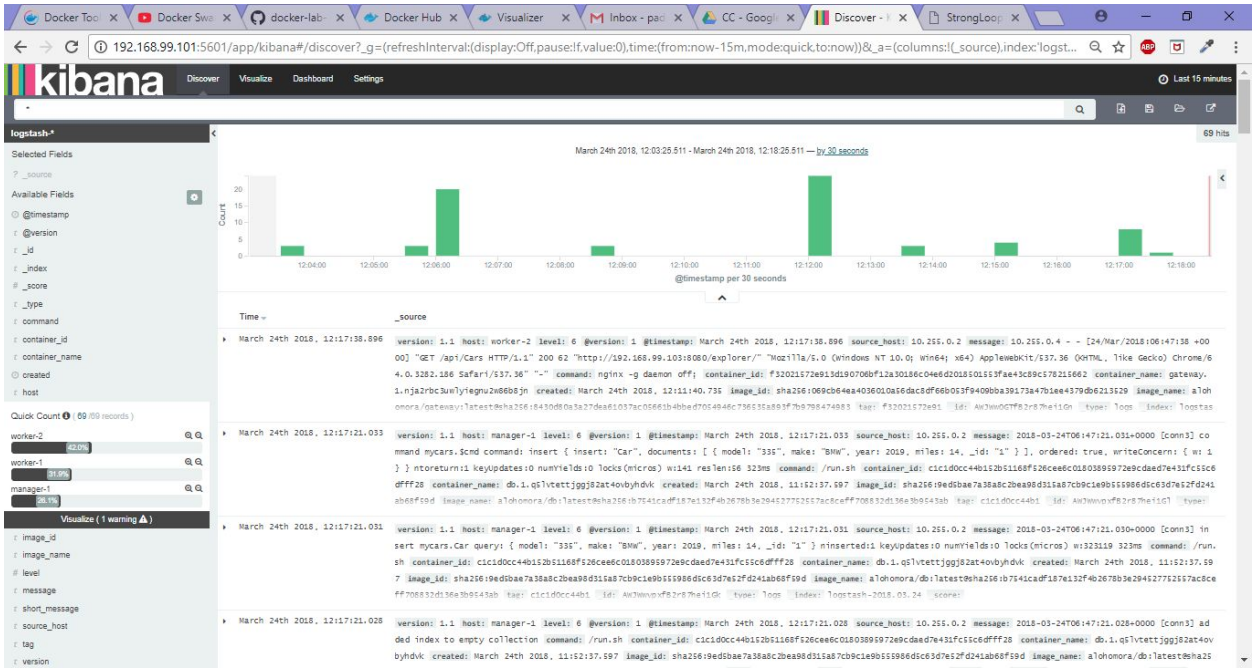
```
DELL_PC@DESKTOP-1C8FPRM MINGW64 ~
$ docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
1utal6azjm6j        backend             overlay             swarm
dbe4c4f3302e        bridge              bridge              local
ecae27c4ac6b        docker_gwbridge     bridge              local
e92ulgygxiwu        frontend            overlay             swarm
d289fb2290b1        host                host                local
r38hddy1ftft        ingress             overlay             swarm
9p7d3mw318gy        logging             overlay             swarm
20d14eb4dc42        none                null                local

DELL_PC@DESKTOP-1C8FPRM MINGW64 ~
$ docker volume ls
DRIVER              VOLUME NAME
rexray              dbdata
rexray              disk.vmdk
rexray              disk.vmdk
rexray              disk.vmdk
rexray              disk.vmdk
rexray              esdata
rexray              hellopersistence
```

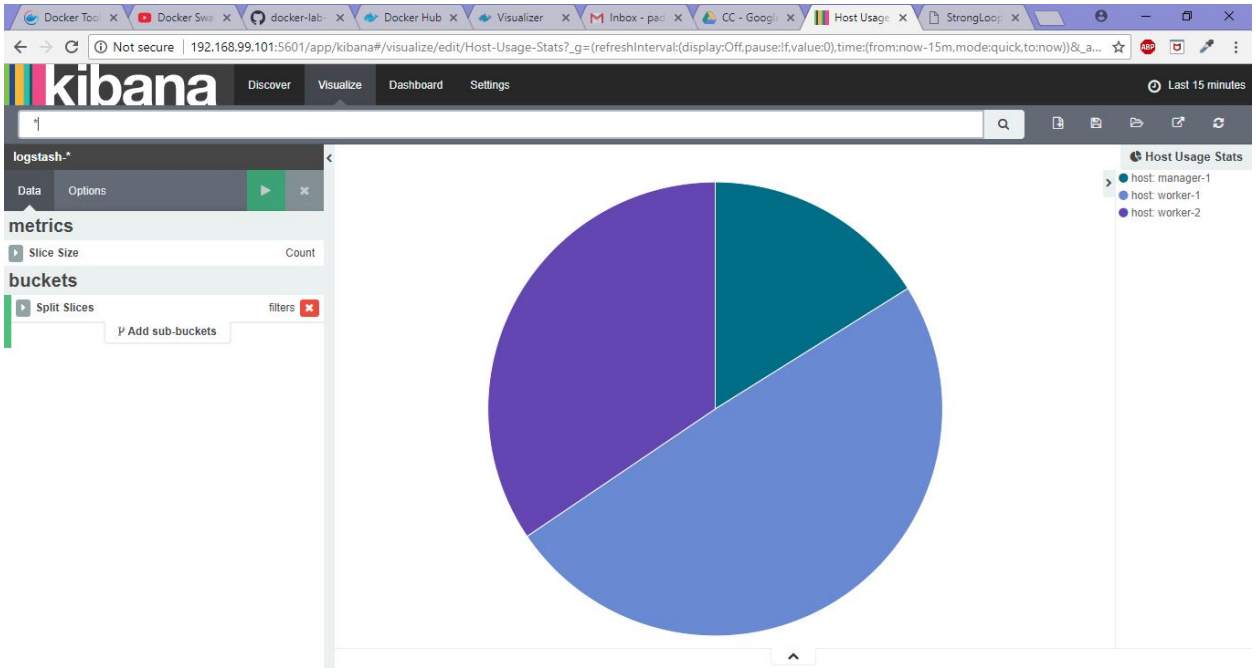


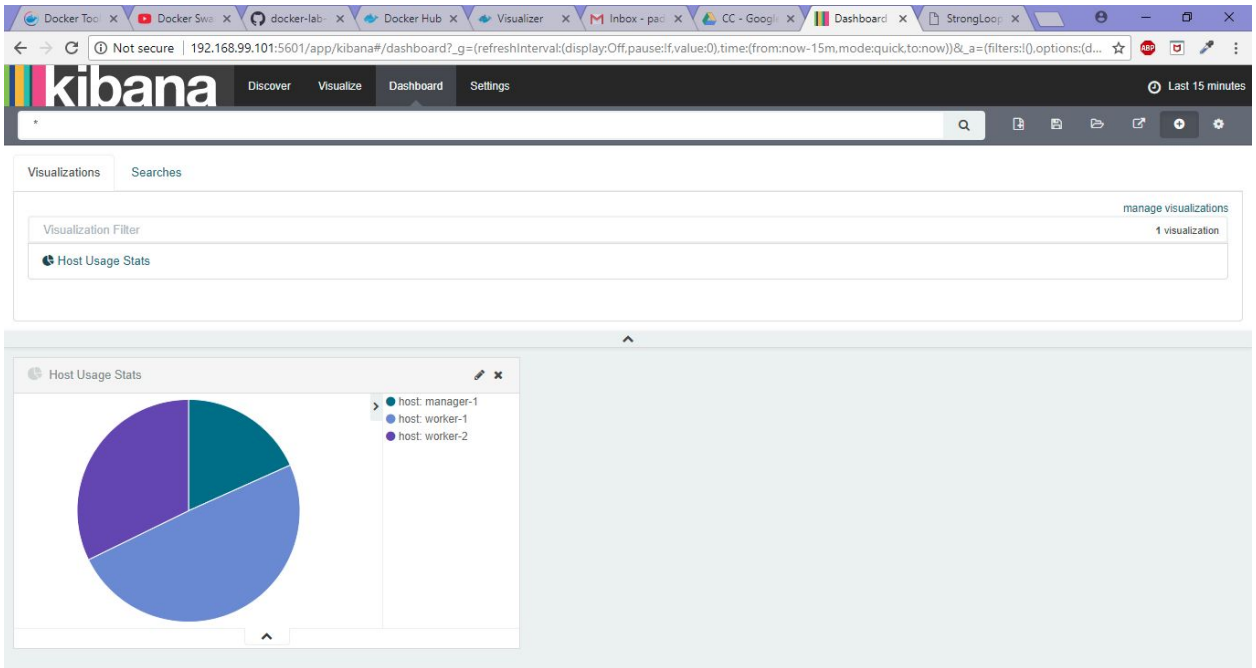

Visualizer
192.168.99.101:5000
filter containers
manager-1
manager
0.972G RAM
x86_64/linux
worker-1
worker
0.972G RAM
x86_64/linux
worker-2
worker
0.972G RAM
x86_64/linux
db
logstash
elasticsearch
api
kibana


Visualizer
192.168.99.101:5000
filter containers
manager-1
manager
0.972G RAM
x86_64/linux
worker-1
worker
0.972G RAM
x86_64/linux
worker-2
worker
0.972G RAM
x86_64/linux
db
logstash
gateway
elasticsearch
api
kibana


StrongLoop API Explorer
Token Not Set
accessToken
Set Access Token
app
Car
Show/Hide
List Operations
Expand Operations
PATCH /Cars Patch an existing model instance or insert a new one into the data source.
GET /Cars Find all instances of the model matched by filter from the data source.
PUT /Cars Patch an existing model instance or insert a new one into the data source.
POST /Cars Create a new instance of the model and persist it into the data source.
PATCH /Cars/{id} Patch attributes for a model instance and persist it into the data source.
GET /Cars/{id} Find a model instance by {{id}} from the data source.
HEAD /Cars/{id} Check whether a model instance exists in the data source.
PUT /Cars/{id} Patch attributes for a model instance and persist it into the data source.
DELETE /Cars/{id} Delete a model instance by {{id}} from the data source.
GET /Cars/{id}/exists Check whether a model instance exists in the data source.
POST /Cars/{id}/replace Replace attributes for a model instance and persist it into the data source.
GET /Cars/change-stream Create a change stream.


kibana
Discover
Visualize
Dashboard
Settings
Last 15 minutes
logstash-*
69 hits
March 24th 2018, 12:03:25.511 - March 24th 2018, 12:18:25.511 — by 30 seconds
@timestamp per 30 seconds
Time
_source


Not secure | 192.168.99.101:5601/app/kibana#/visualize/edit/Host-Usage-Stats?_g=(refreshInterval:(display:Off,pause:!f,value:0),time:(from:now-15m,mode:quick,to:now))&_a...
kibana
Discover
Visualize
Dashboard
Settings
Last 15 minutes
logstash-*
Data
Options
metrics
Slice Size
Count
buckets
Split Slices
filters
Add sub-buckets
Host Usage Stats
host: manager-1
host: worker-1
host: worker-2


Not secure | 192.168.99.101:5601/app/kibana#/dashboard?_g=(refreshInterval:(display:Off,pause:!f,value:0),time:(from:now-15m,mode:quick,to:now))&_a=(filters:!(),options:(d...
kibana
Discover
Visualize
Dashboard
Settings
Last 15 minutes
Visualizations
Searches
manage visualizations
Visualization Filter
1 visualization
Host Usage Stats
Host Usage Stats
host: manager-1
host: worker-1
host: worker-2

Operations

1. Scale nginx to take on more load.(Automatic load balancing by swarm between these 3 instances)
2. Drain the node that has the api. This will force the api to be scheduled on another active node.

Docker service scale service_name = number

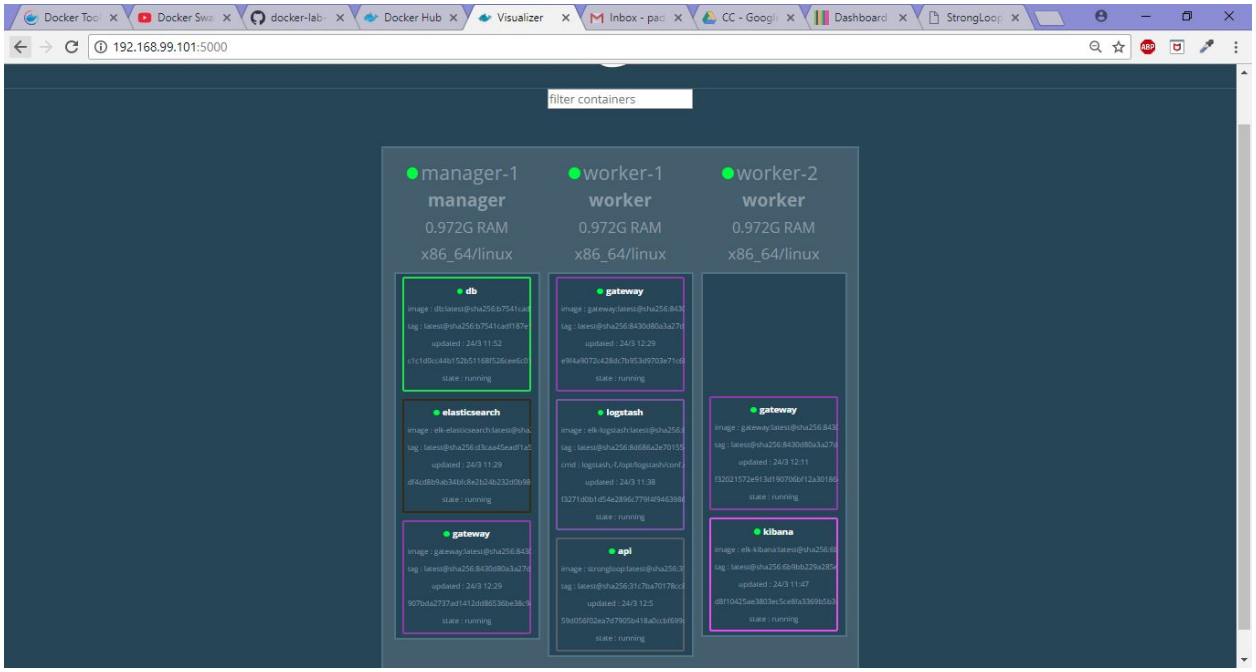
Docker node update --availability drain/active/pause <node>

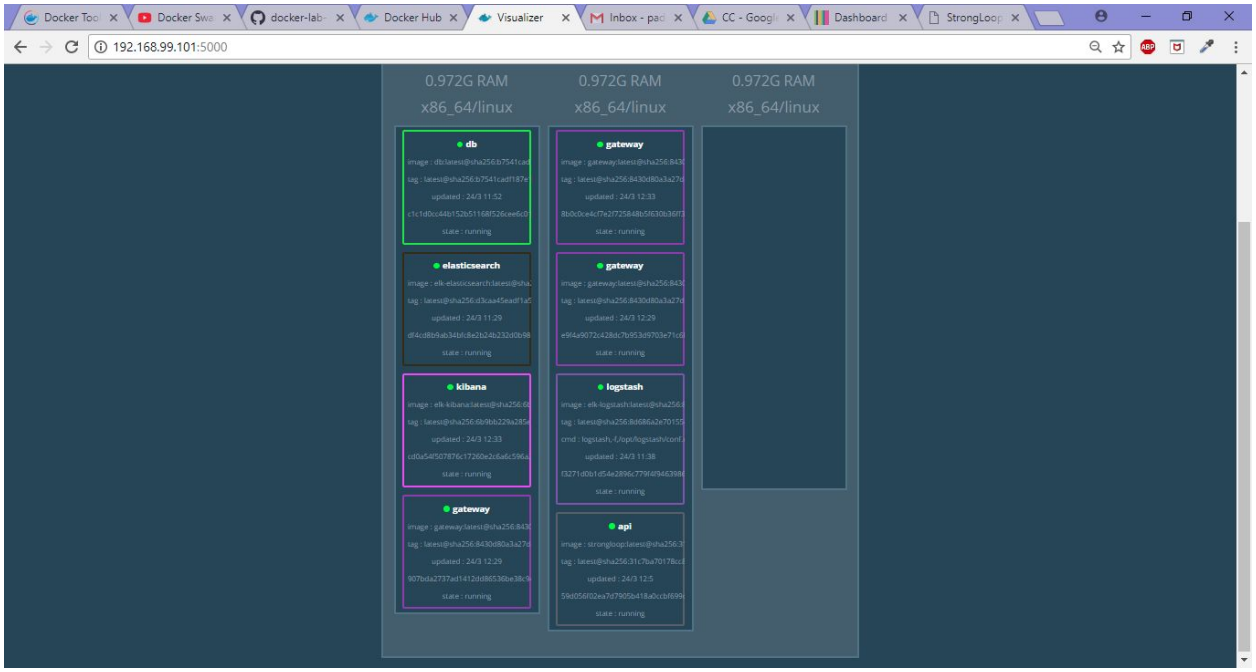
Docker service update -- env-add UPDATE=1 service_name

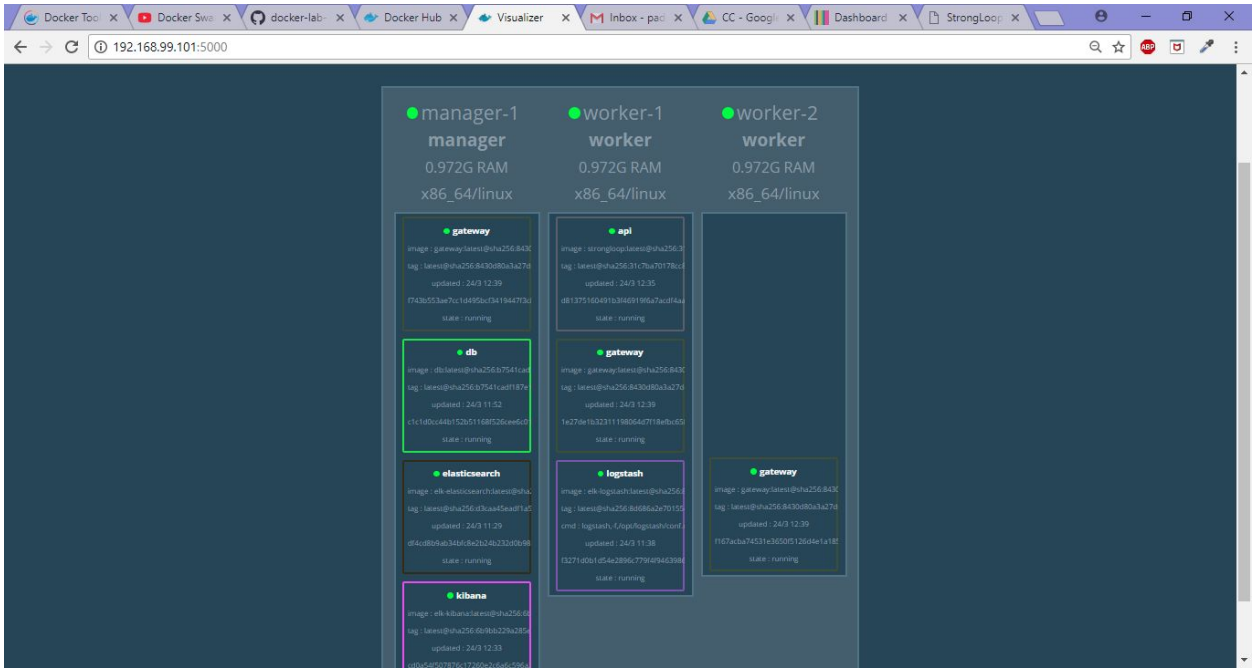
Docker service rm service_name

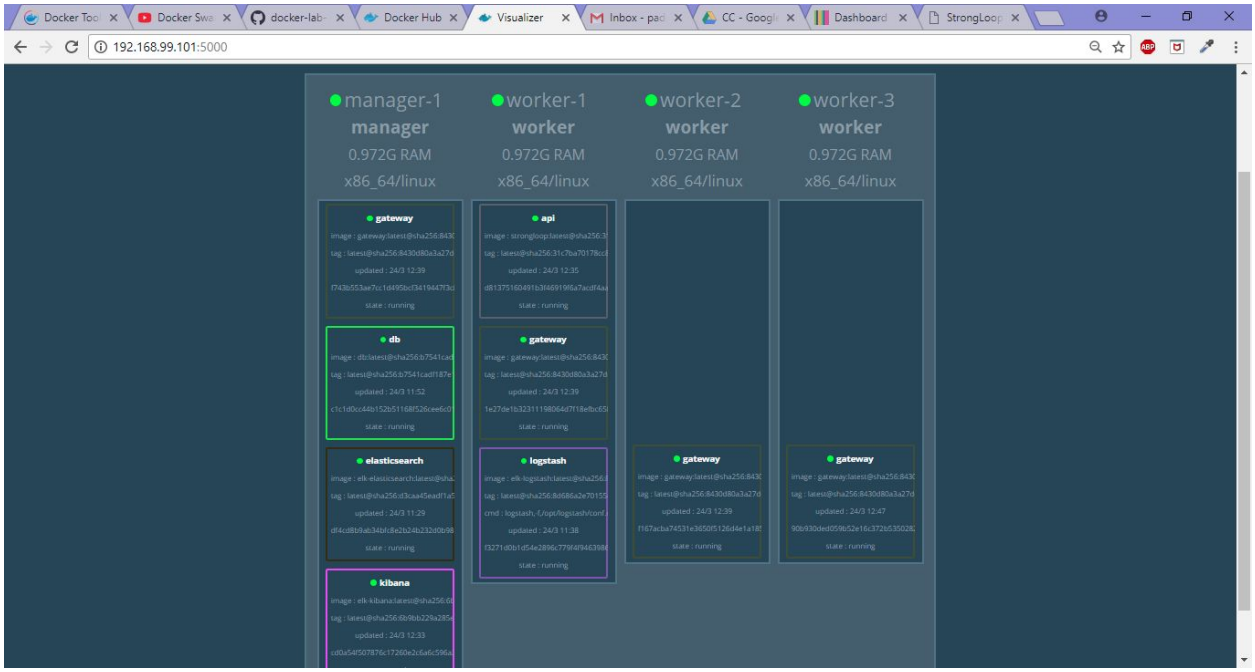
```
 docker service create \
--name gateway \
--network frontend \
--network logging \
--replicas 1 \
--log-driver=gelf --log-opt gelf-address=udp://$(docker-machine ip manager-1):12201 \
-p 8080:80 \
alohomora/gateway:latest
```

```
docker service create \
--name gateway \
--network frontend \
--network logging \
--log-driver=gelf --log-opt gelf-address=udp://$(docker-machine ip manager-1):12201 \
-p 8080:80 \
--mode global \
alohomora/gateway:latest
```


Visualizer
192.168.99.101:5000
filter containers
manager-1
manager
0.972G RAM
x86_64/linux
worker-1
worker
0.972G RAM
x86_64/linux
worker-2
worker
0.972G RAM
x86_64/linux
db
elasticsearch
gateway
gateway
logstash
api
gateway
kibana


Visualizer
192.168.99.101:5000
0.972G RAM
x86_64/linux
0.972G RAM
x86_64/linux
0.972G RAM
x86_64/linux
db
elasticsearch
kibana
gateway
gateway
gateway
logstash
api


Visualizer
192.168.99.101:5000
manager-1
manager
0.972G RAM
x86_64/linux
worker-1
worker
0.972G RAM
x86_64/linux
worker-2
worker
0.972G RAM
x86_64/linux
gateway
api
db
gateway
elasticsearch
logstash
kibana
gateway


Visualizer
192.168.99.101:5000
manager-1
manager
0.972G RAM
x86_64/linux
worker-1
worker
0.972G RAM
x86_64/linux
worker-2
worker
0.972G RAM
x86_64/linux
worker-3
worker
0.972G RAM
x86_64/linux
gateway
api
db
gateway
elasticsearch
logstash
kibana
gateway
gateway