LOGGING & MONITORING WORKSHOP

Platform Engineering

Platform Engineering

Table of Contents

Clone the workshop

from codebase **https://github.com/arunma/spring-logmon**

OS Specific Instructions are available at the bottom of the workshop

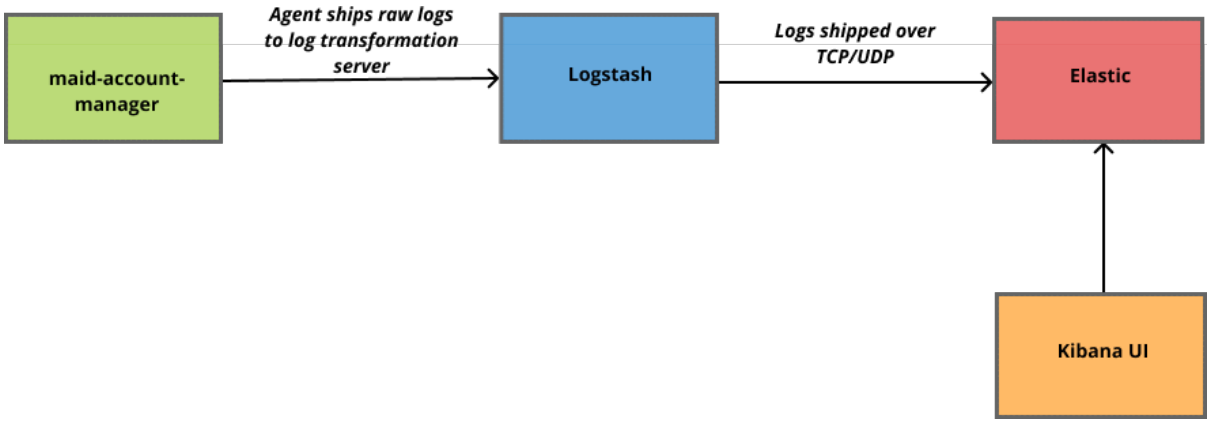
The cloned repository has four containers wrapped in a docker-compose.yml

1. maid-account-manager spring application
2. Elasticsearch datastore
3. Logstash log shipper
4. Kibana UI

Launch all the containers

on your local machine using docker-compose

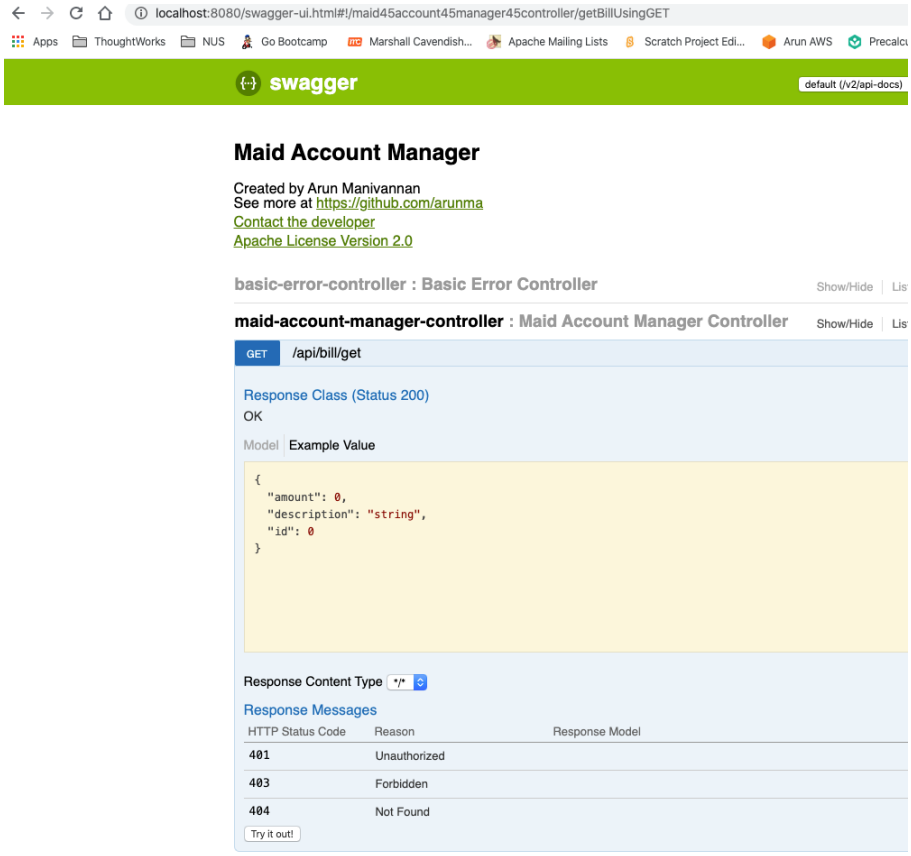
```
cd <location of spring-logmon >

# eg. cd /Users/arunma/projects/spring-logmon

docker-compose -f docker/docker-compose.yml up --build
```



Create dummy logs using Swagger UI

Visit the Swagger UI url on your localhost at http://localhost:8080/swagger-ui.html

Expand the maid-account-manager-controller and click “Try it out” a few times.



The MaidAccountController's implementation creates exception roughly 40% of the time.

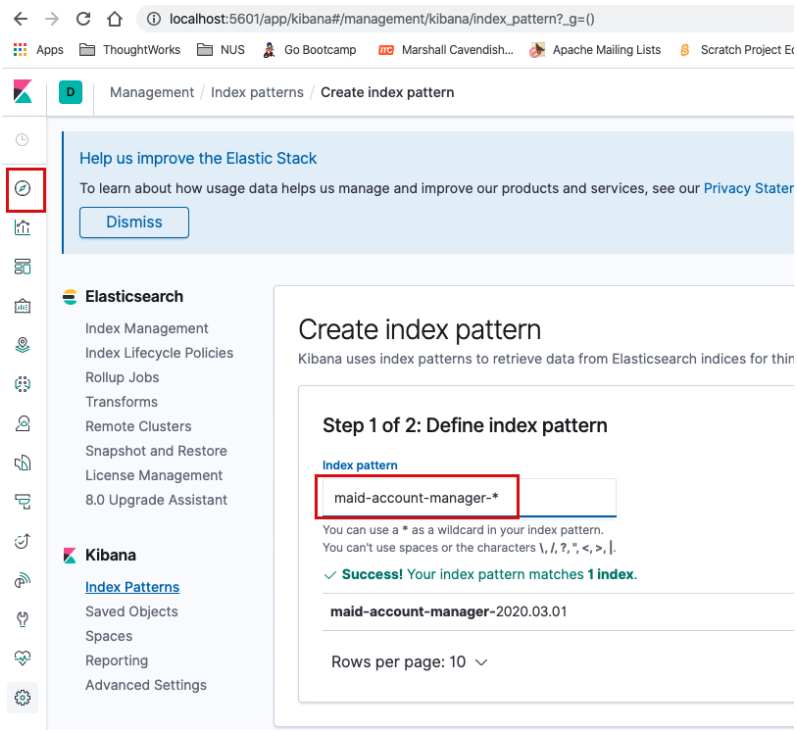
MaidAccountManagerController.java | logback-spring.xml | application.properties | MaidApplication.java

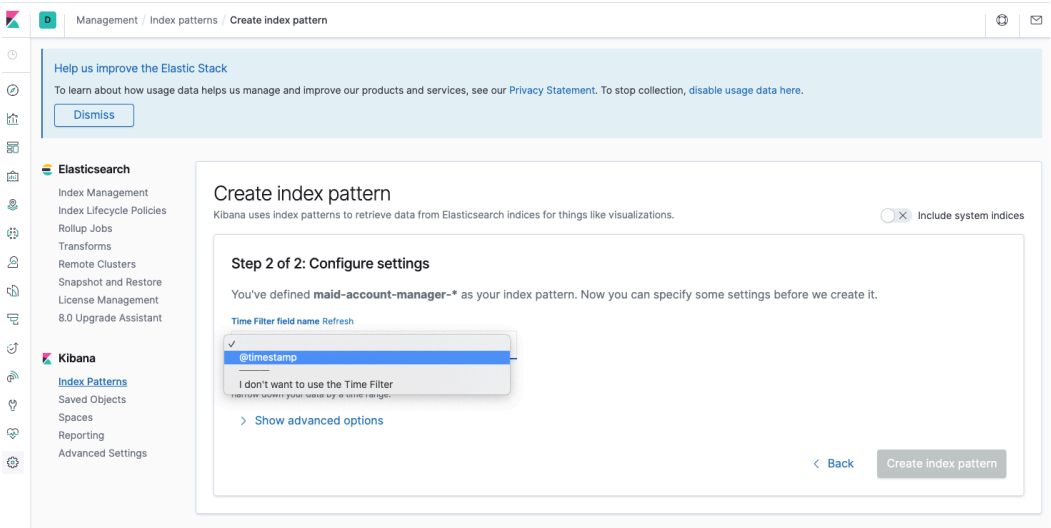
```java
import java.time.Month;
import java.time.format.TextStyle;
import java.util.Date;
import java.util.Locale;
import java.util.Random;
import java.util.concurrent.ThreadLocalRandom;

@RestController
@RequestMapping(MaidAccountManagerController.BASE_URL)
public class MaidAccountManagerController {

    public static final String BASE_URL = "/api/bill";
    private Logger log = LoggerFactory.getLogger(MaidAccountManagerController.class);

    @GetMapping(path = "/get")
    @ResponseStatus(HttpStatus.OK)
    public MaidUsageBill getBill() {
        double random = new Random().nextDouble();
        if (random < 0.4) {
            log.error("Random Exception for value " + random, new RuntimeException("Random Exception is thrown @ " + new Date()));
        }
        String randomMonth = Month.of(ThreadLocalRandom.current().nextInt( origin: 1,  bound: 12)).getDisplayName(TextStyle.FULL, Locale.US);
        String randomYear = String.valueOf(ThreadLocalRandom.current().nextInt( origin: 2000,  bound: 2020));
        int id = Math.abs(ThreadLocalRandom.current().nextInt());
        double amount = Math.ceil(ThreadLocalRandom.current().nextDouble( origin: 500,  bound: 1200));
        log.info("Returning Maid Usage for the period {}, {} with Id {} and Amount {}", randomMonth, randomYear, id, amount);
        return new MaidUsageBill(
                id,
                amount,
                String.format("Version 2 of the API - Payment for %s %s", randomMonth, randomYear)
        );
    }
```

Check your logs in Kibana

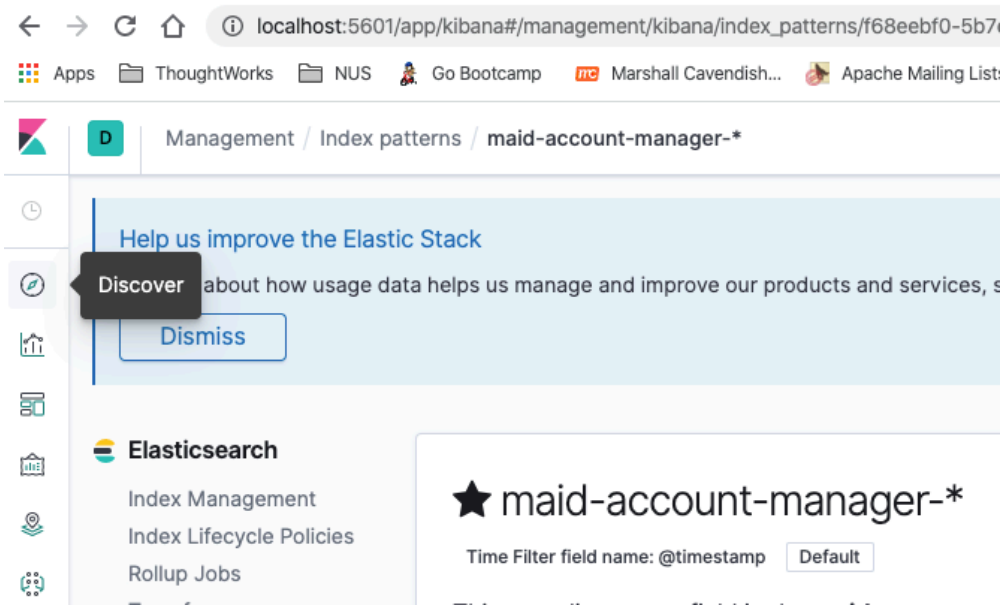
Once enough logs have been created, go to the Kibana UI at http://localhost:5601/app/kibana

Create a new Index Pattern

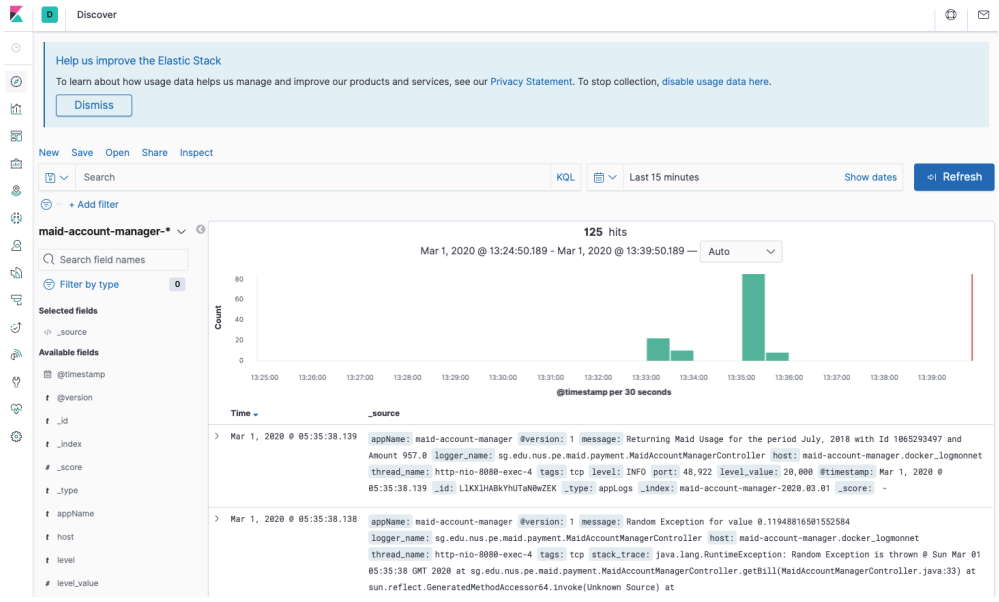




Click the discovery button



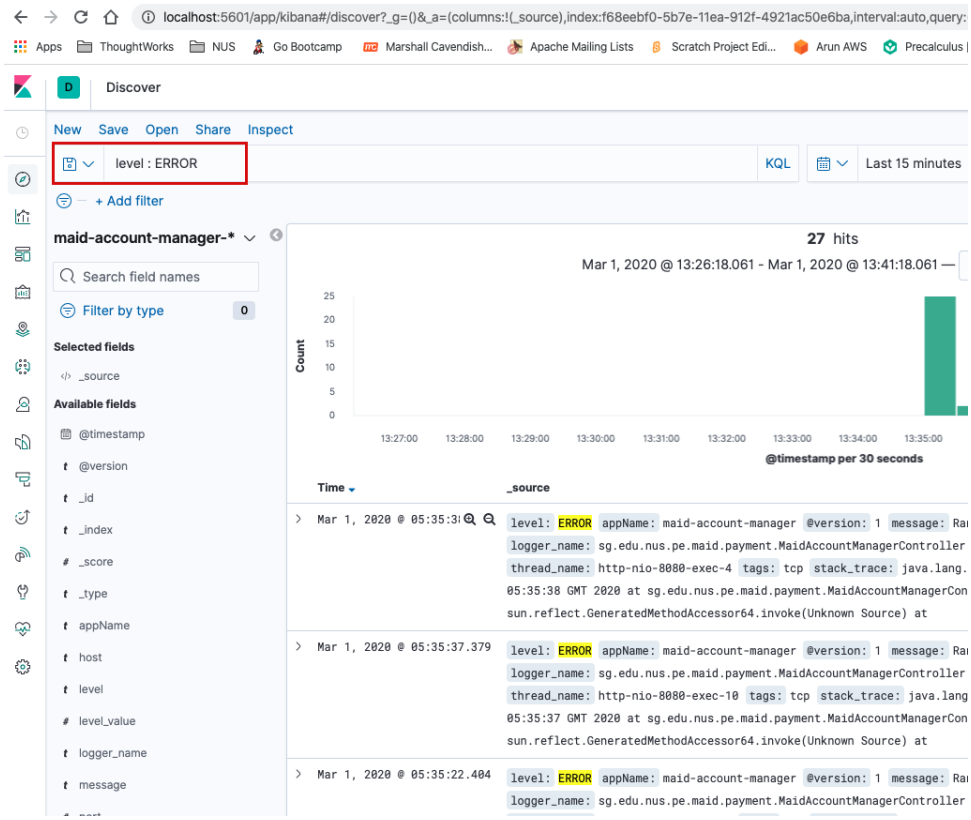
You should see all the logs already captured



Let's filter for Errors from the logs

In the "Search" bar, issue the following search criteria. You can also see the count of errors in the histogram.

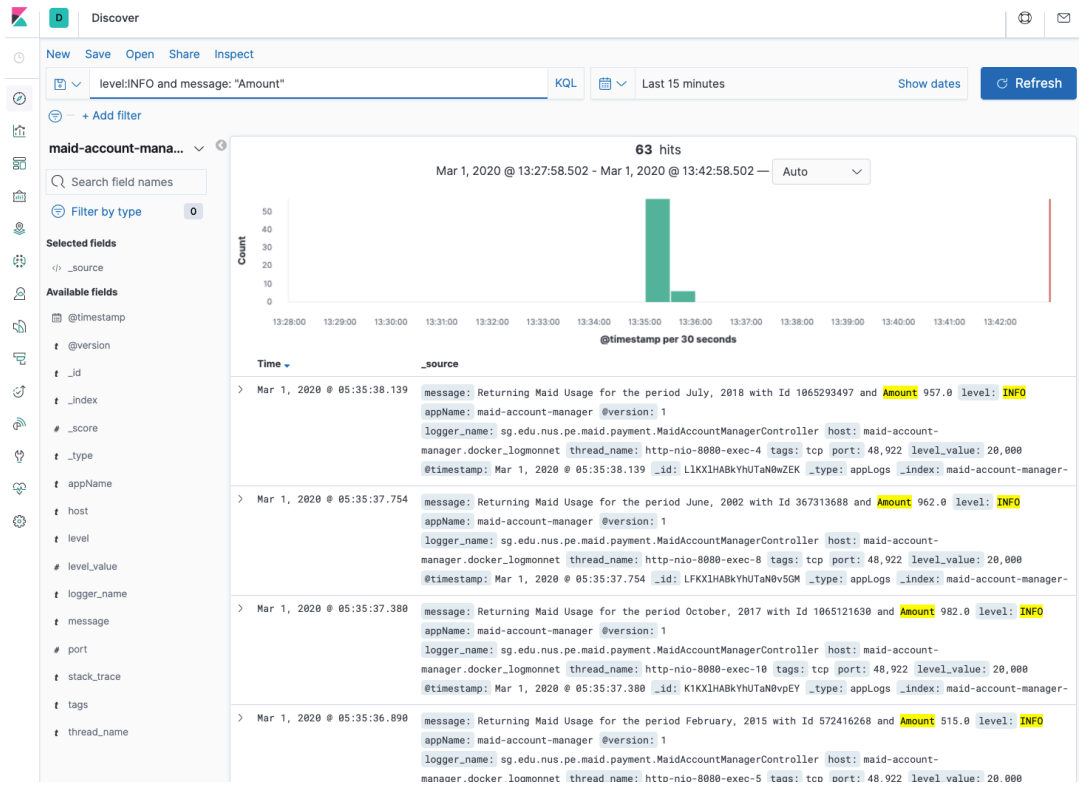
```
level:ERROR
```

Try using a richer query criteria

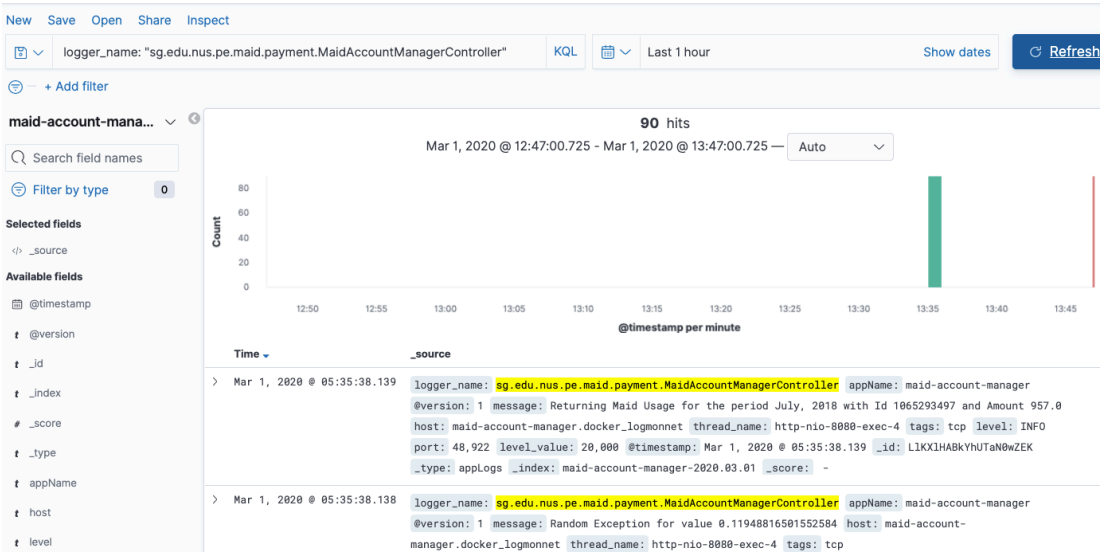
Filter INFO level logs that has “Amount” in the log message

```
level:INFO and message: "Amount"
```

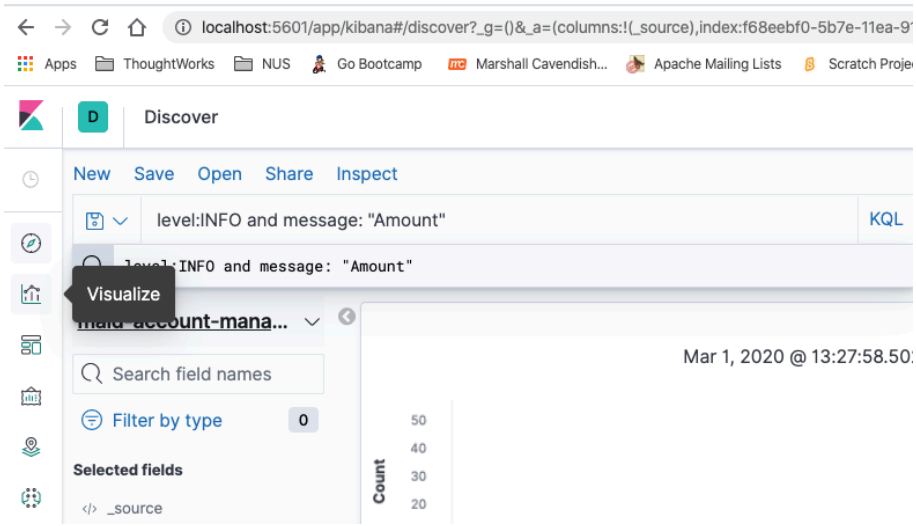
Filtering all logs from Controller

```
logger_name: "sg.edu.nus.pe.maid.payment.MaidAccountManagerController"
```

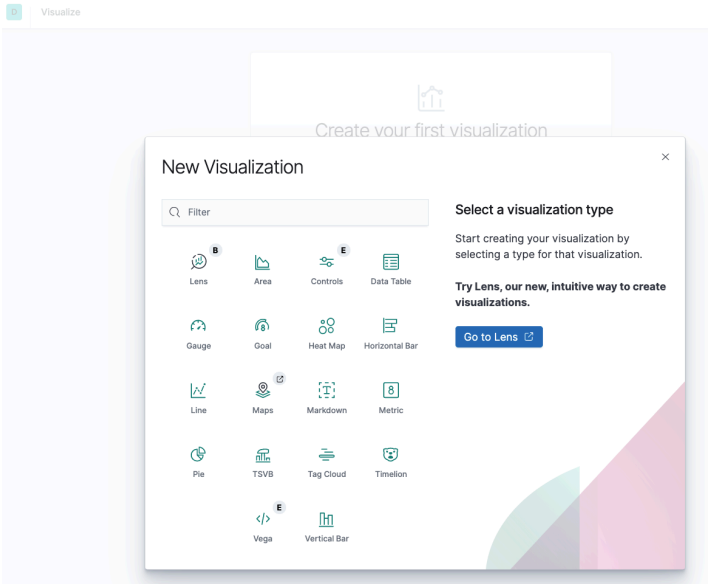


Looks like we have 90 log messages generated from the MaidAccountManagerController. Let's visualize how much of these are errors and how much are good responses.

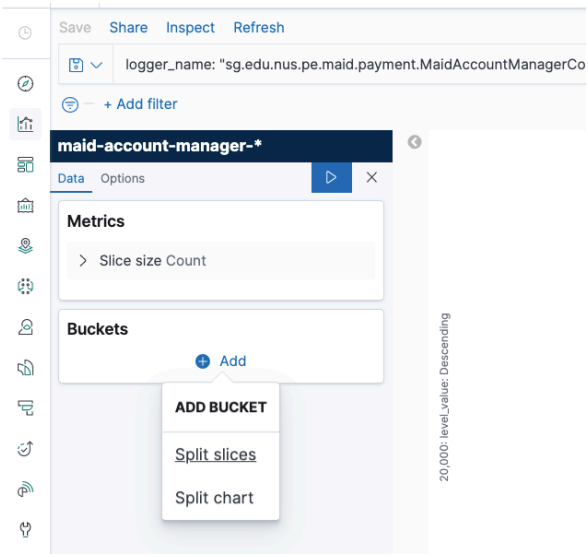
Visualize



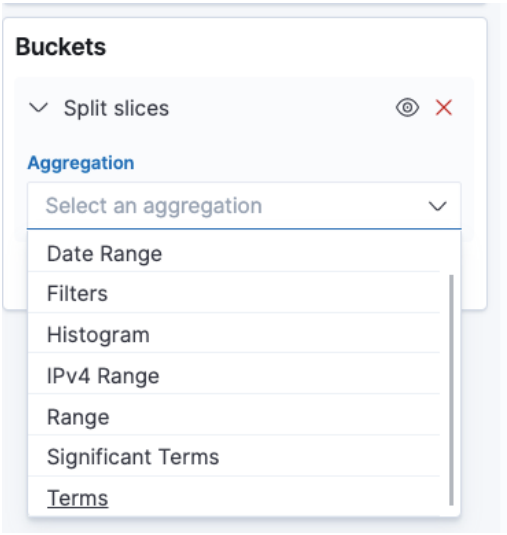
Create a pie chart



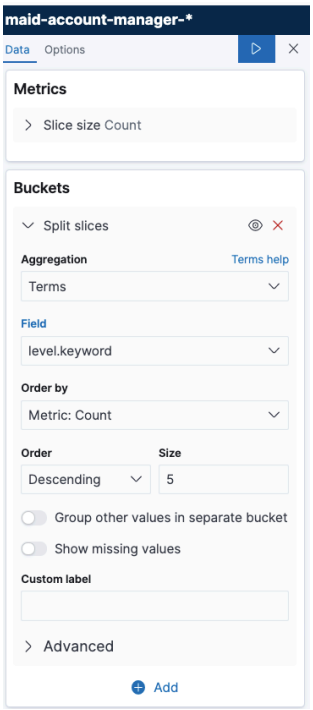
Create a new Split slice



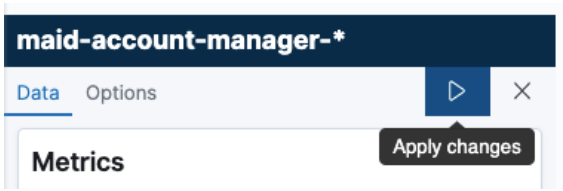
Select “Terms”



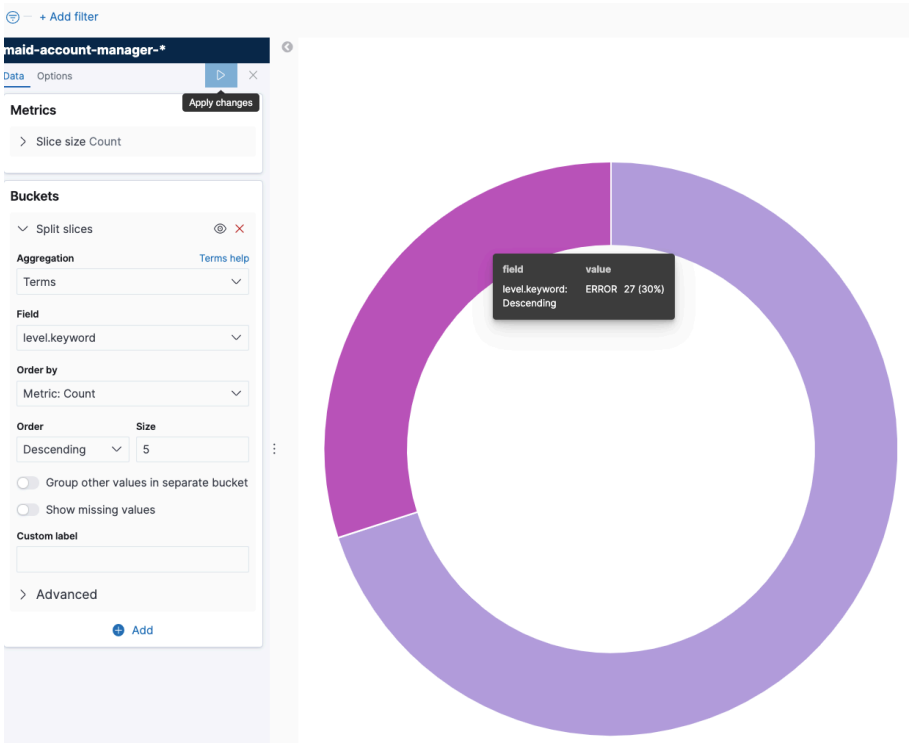
In the Field, select “level_keyword”



Click "Apply changes"



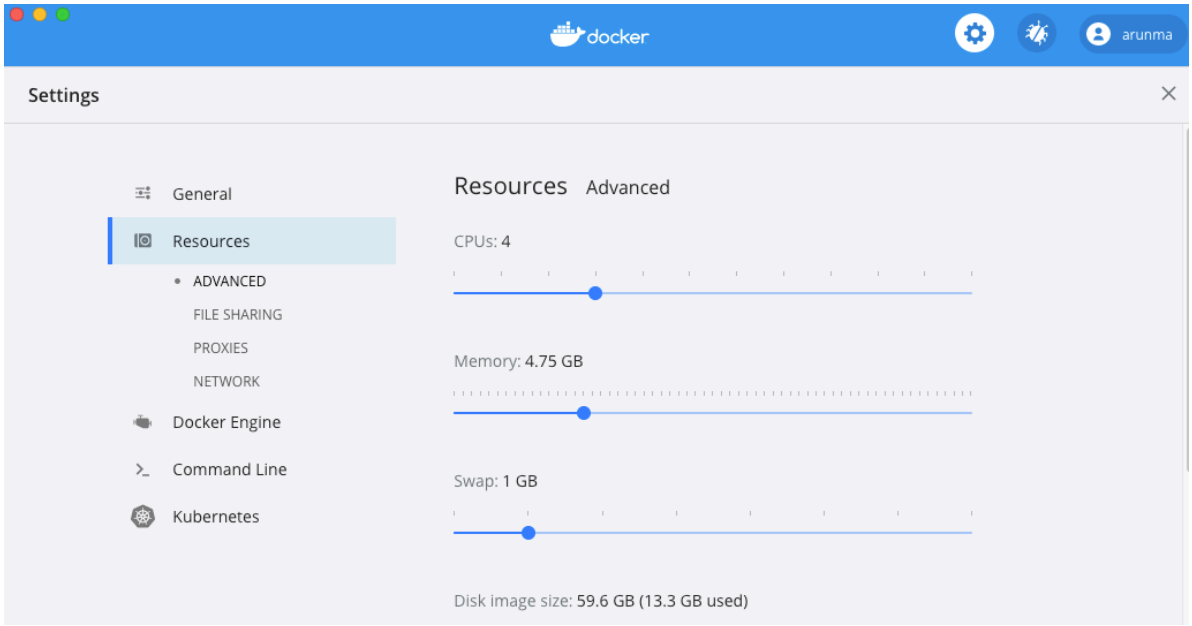
We now have a live visualisation of the error/good responses.



End of workshop

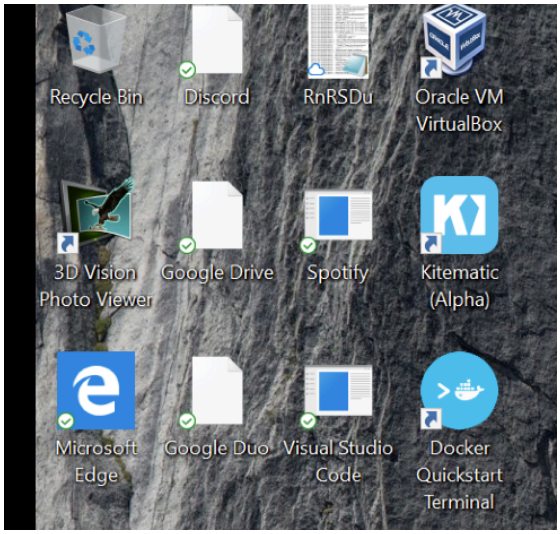
OS Specific instructions

Mac & Windows Pro Special Instructions:
Extra Adjust your docker settings
Give at least 4 cores and 3 GB memory

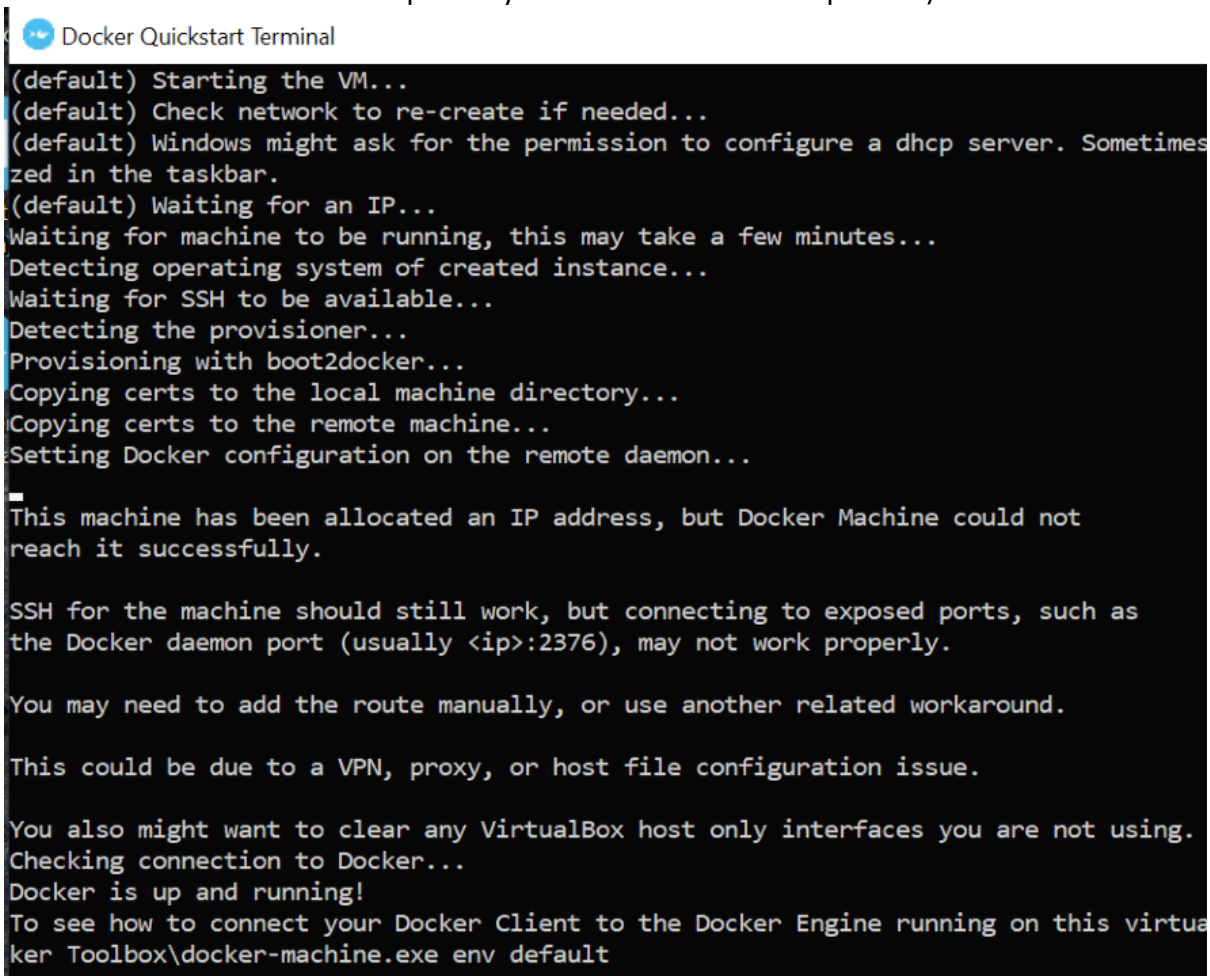


Windows Home Special instructions (for windows >7)

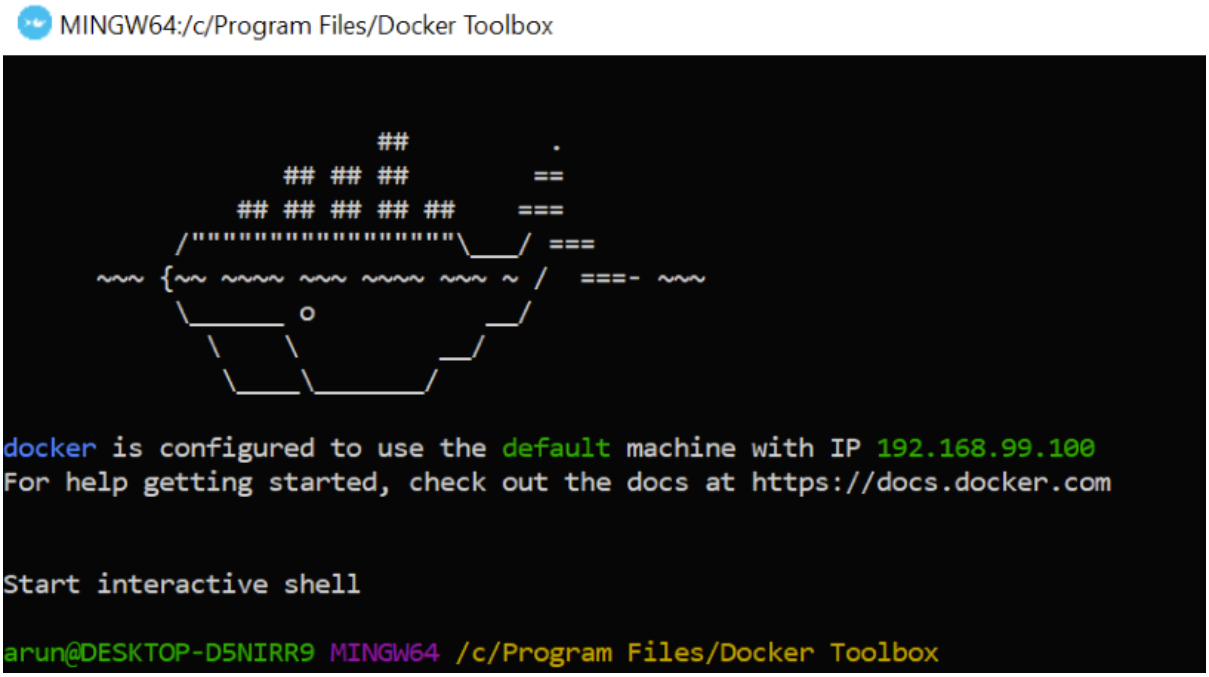
1. Download Docker toolbox from https://github.com/docker/toolbox/releases
2. Install Linux VM
    a. You must see the Docker Quickstart Terminal on your desktop



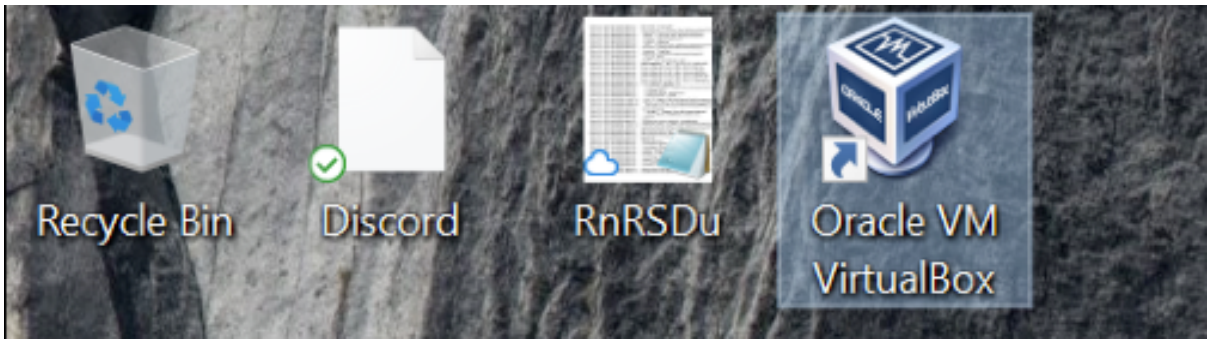
    b. Upon clicking, it would create a new VM inside your Oracle Virtual Box (which was installed as part of your toolbox installation process)



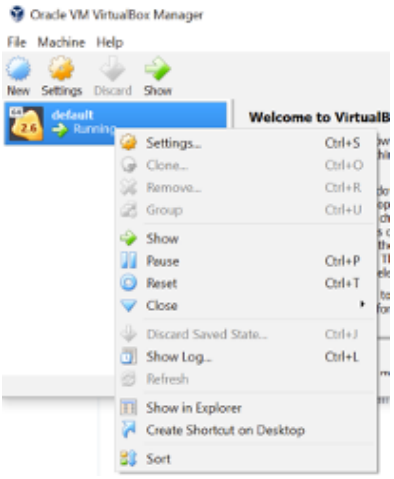
    c. You must see the Docker ASCII art on your screen



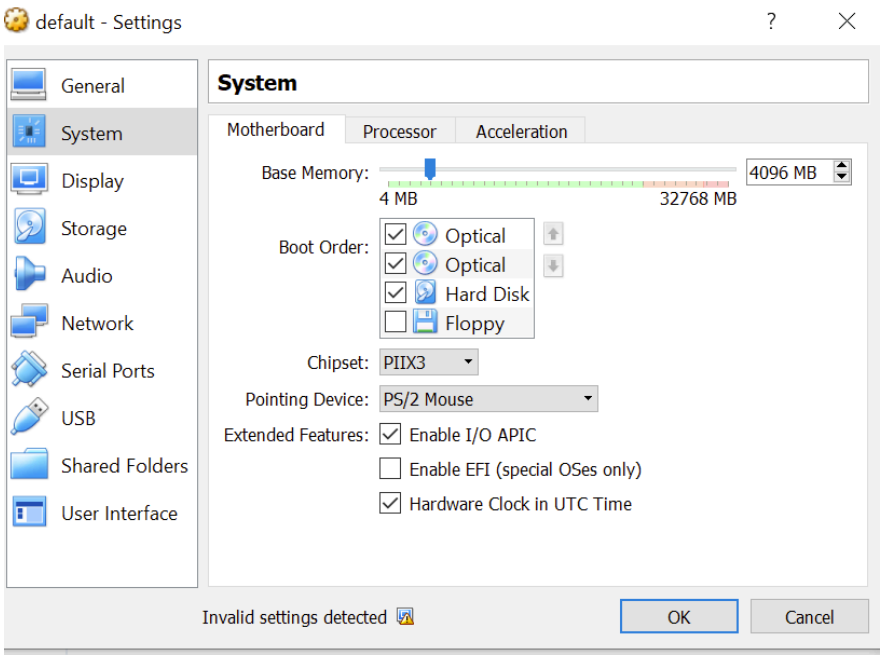
3. Adjust your memory and processor settings:
   a. Open your virtual box



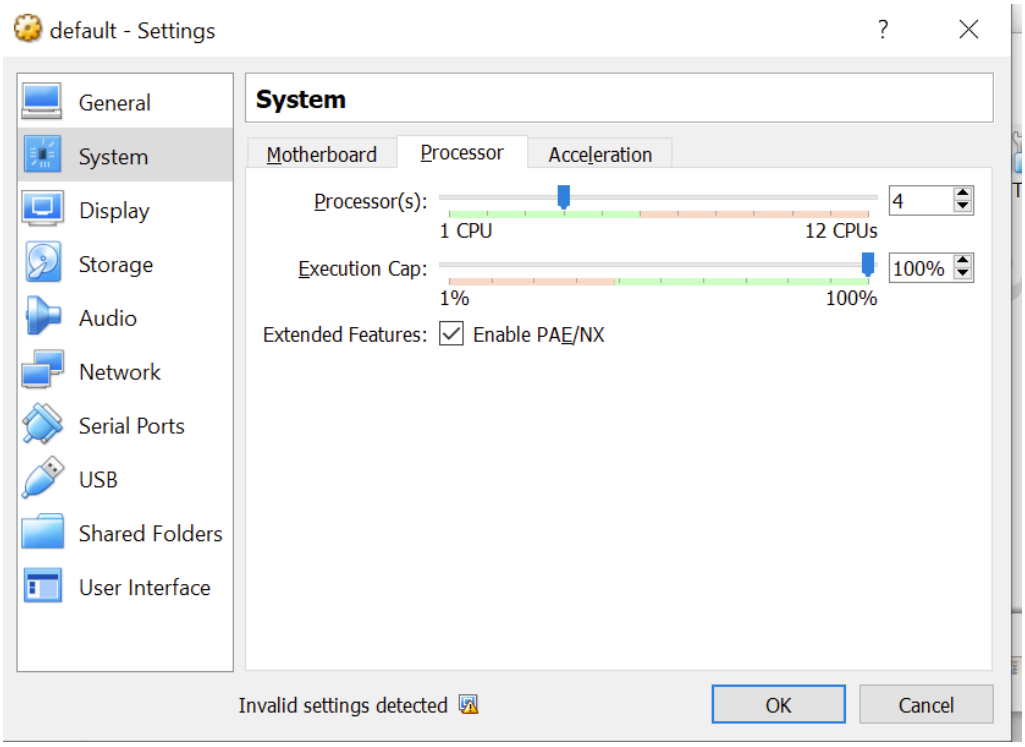
   b. Close and shut down your linux VM



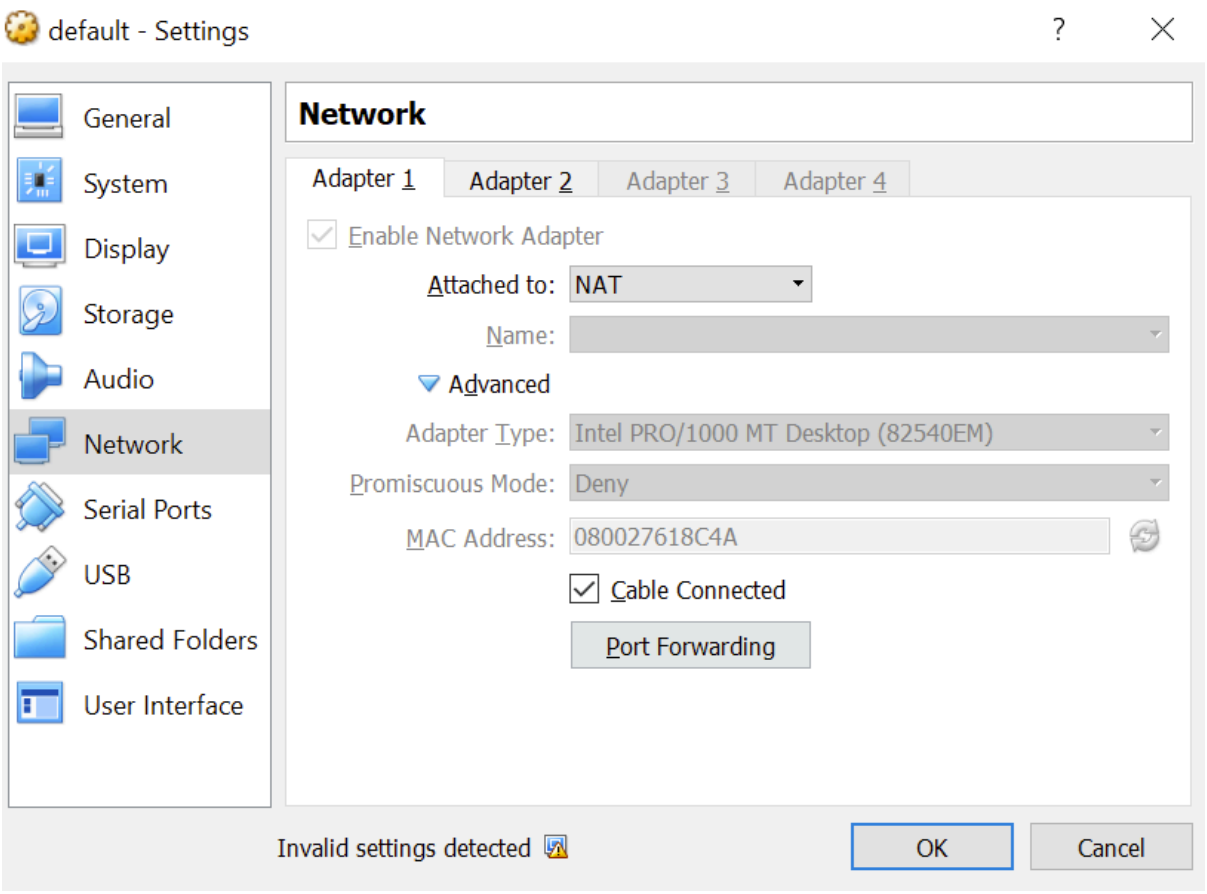
   c. Update memory to be at least 4 GB

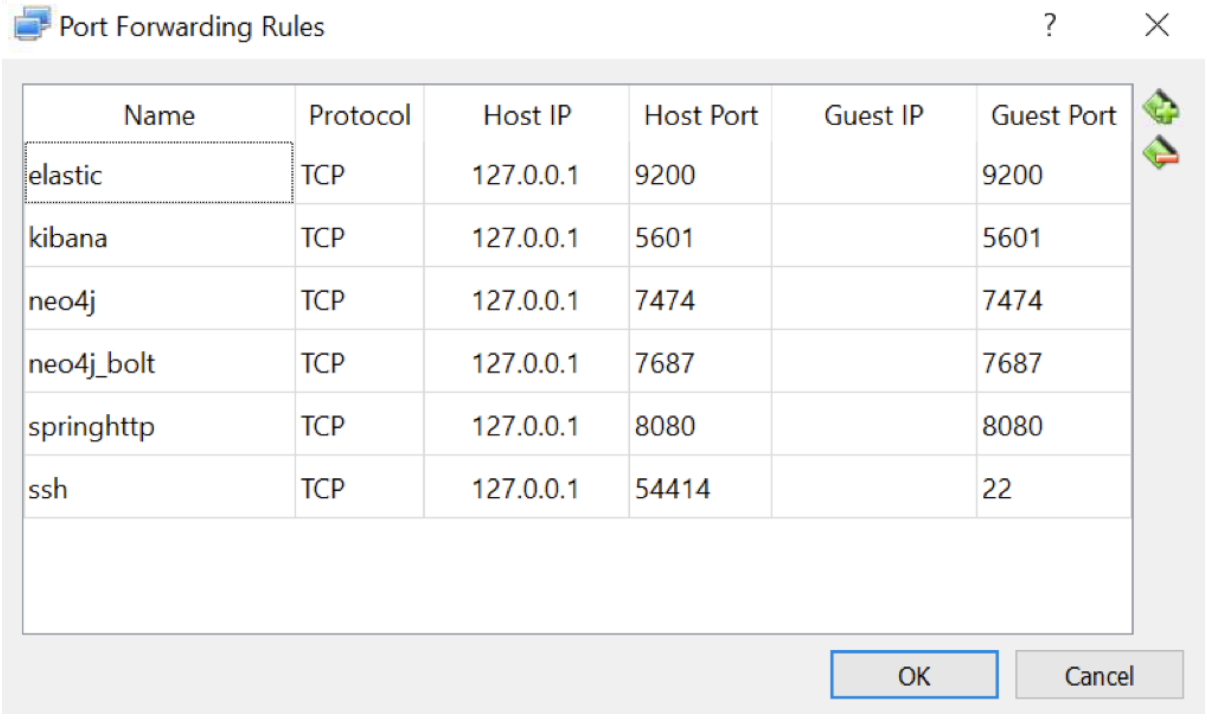


d. Update the number of processors to be 4

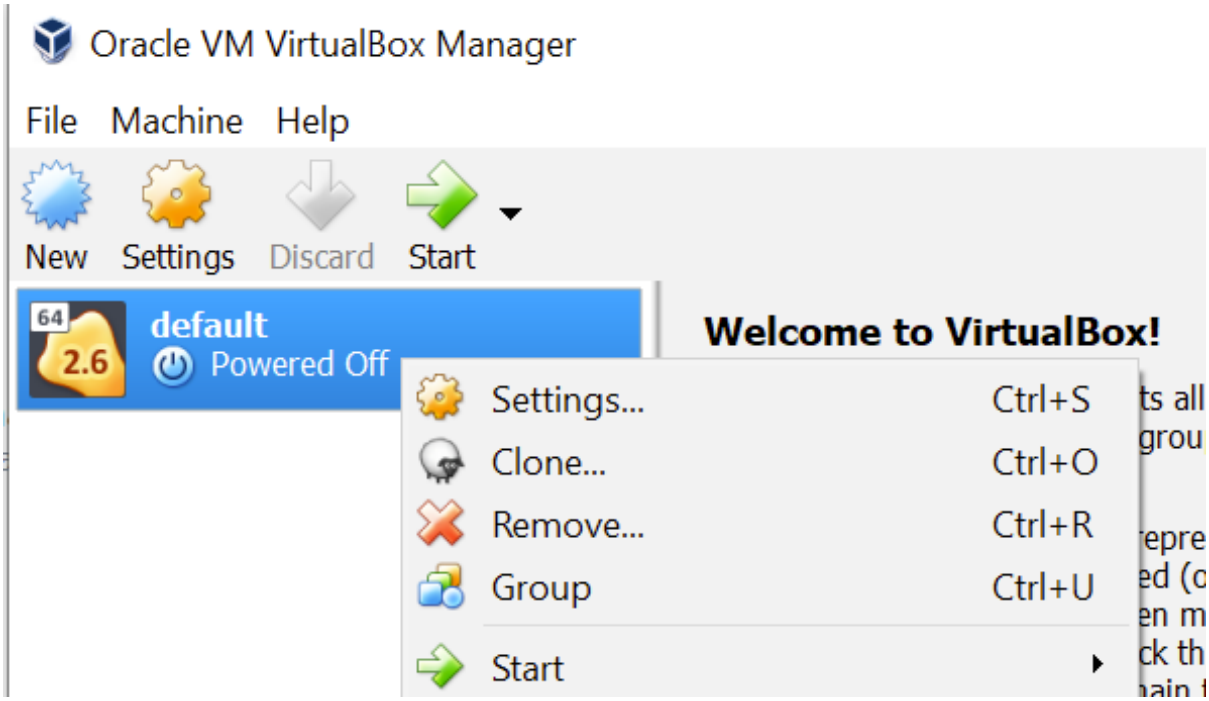


e. Set “Port Forwarding rules” in your Network





Restart your VM (Headless start)



Common Errors and Solutions

1. Problem
"Forbidden" while creating an index in Kibana

Solution
curl -XPUT -H "Content-Type: application/json" http://localhost:9200/_all/_settings -d '{"index.blocks.read_only_allow_delete": null}'

2. Problem
Error 137 or Kibana not coming up

Solution
Check if you have increased the Docker memory to at least 4 GB and your CPU to be 4.

