1 Msregistry (discovery microservice) 8010 eclipse

2 MRCConfigServer (Spring cloud config server) coupled with Git (MRC Config)..assymetric encryption used with JWT token 8012 eclipse

3 Zuul API Gateway – Eclipse --- 8011

4 JWTMS(in E/copy) ..main user account details (JWT auth) (random port with discovery service) intellij

5 MICRO1 (Angular appp)

6 Micro3possim (eclipse) – Position simulator …..8080 (queue name --- fleetman) (couple it with eureka later)

7 Micro4possim --- Position tracker (which recives data from queue) intellij

8 MICRONOD (Nodejs backend) (coap server and websocket client) messages received from coap client in pi is routed to angular frontend usinf websockets visualcode

9 Msvehicle ms with vehicle details (feign operation) intellij

ELK Updated (in E/Thesis private/ELK….) (kibana 5601….elastic 9200…logstash 9600) skipped due to time constarints and exisiting useless problems

COap client in Raspberry coap_ms .. done

Web app locally deployed in container..set up angular ansiblke and deploy web app blue/black///try using docket community

Separate container for queue, ELK, Database

User, postion, vehicle fresh start

Micro1 (in intellij) ---- for testing

The global config server ami has been saved and need to be autpscaled alteast one running in any AZ to avoid MS connection blackout however we are skiiping tis to save costs.

Auto scaling group and launch configuration script ready..load balancer to be created properly

Naram.yaml will deploy config server (ansible notebook)

Registry.yaml will deploy eureka

Pending bus refresh in config server

Rabbit new yaml works perfectly

Config server --- naram and asg also elb also

Rabbit – only ansible pla rabbitnew.yaml

Eureka only script registry.yaml

Ec2_instance.yaml for Jenkins

Try a simple web app Rahul Shetty deployment using Jenkins and blue/green deployment

Guess the number game understand and push to git thymeleaf

FIRST STEP

Manual..create elastic ip for the same and provide it in rabbitnew.yaml

Start rabbit Ms..rabbitnew.yaml…

Rabbit done

SECOND STEP

Start config server

Asg..start conf..laod balancer(default manually created coupled created)..can be accessed with elb

Config server running

THIRD STEP

Eureka registry start..no asg or elb..just registry.yaml which will take already created elastic ip….

FOURTHE STEP

Sample web srarted (with elastic ip in docker env..eureka issue solved)

RUN CONFIG ASG WHEN STARTING FREE

FIFTH

FLEETMAN TRACKER AND VEHICLE DETAILS(FEIGN) DONE

SIXTH

HYSTRIX FALL BACK

SEVENTH

GATEWAY

EIGHT

USER SERVICE

NINTH

ANGULAR DONE

NEXT ELK FOR USER

GUAGE COAP

TENTH

COAP/CONNECT (NODEJS TO CLOUD)

ONCE FULL CONNECT K8S

IT-huawei-8383884

SR320200921181045204

Docker run -d -v esdata1:/usr/share/elasticsearch/data –name elasticsearch -p 9200:9200 -p 9300:9300 -e “discovery.type=single-node” elasticsearch:7.2.0

Docker run -d –link elasticsearch:elasticsearch -p 5601:5601 kibana:7.3.0

Docker Maven Plugin for latest uses

```
<plugin>
                                        <groupId>org.springframework.boot</groupId>
                                        <artifactId>spring-boot-maven-
            plugin</artifactId>

                    <configuration><executable>true</executable></configuration>
                                </plugin>

                                <plugin>
                                        <groupId>io.fabric8</groupId>
                                        <artifactId>docker-maven-plugin</artifactId>
                                        <version>0.21.0</version>

                                        <configuration>
                                <verbose>true</verbose>

            <authConfig>
            <username></username>  // your dockerhub username
            <password></password> //your dockerhub pass
            </authConfig>
                                                <images>
                                                        <image>

                    <name>virtualpairprogrammers/fleetman-position-tracker</name>
                                                                <build>

                    <dockerFileDir>${project.basedir}/src/main/docker/</dockerFileDir>
                                                                        <assembly>

                    <descriptorRef>artifact</descriptorRef>
                                                                        </assembly>
                                                                        <tags>

                    <tag>latest</tag>
                                                                        </tags>
```

```
                                                    </build>
                                                </image>
                                            </images>
                                        </configuration>
                <!—optional>// to avoid manual build of pom
                <executions>
                <execution>
                <phase>package</phase>
                <goals>
                <goal>build</goal>
                </goals>
                </execution>

                </executions>
                                    </plugin>
```

Build in eclipse with goals “clean package docker:build”..also note the jar is inside maven folder in target when run from plugin..refer docker file of Richard..the above opetional can be used to avoid docker:build..if to push use docker:push

K8S

Config

docker build --tag configserver:minikube .

docker run --restart always --network sasi --publish 8012:8012 --detach --name miniconfig configserver:minikube

Rabbit

docker run --restart always -p 15672:15672 -p 5672:5672 -p 15671:15671 -p 5671:5671 -p 4369:4369 --network sasi --detach --name minirabbit rabbitmq:3-management

DiscoveryServer

docker run --restart always --publish 8010:8010 --network sasi --detach --name minieureka registry:minieureka

PoSSimulatpr

docker build --tag sim:minisim .

docker run --restart always --network sasi --detach --name minisim sim:minisim

Coap

docker run --restart always --network sasi -p 5683:5683/udp -p 3002:3002 --detach --name minicoap coap:minicoap

docker run --restart always --network sasi -p 5683:5683/udp --detach --name minicoap coap:minicoap

Postracker

docker build --tag posra:miniposra .

docker run --restart always --network sasi -p 8888:8888 --detach --name miniposra posra:miniposra

PosVehicle

docker build --tag vehicle:minivehicle .

docker run --restart always --network sasi -p 8080:8080 --detach --name minivehicle vehicle:minivehicle

user, gateway(check), elk…

push to docker rep

login to kubectl mini

user

docker run --restart always --network sasi -e "logging.file.name=/api-logs/usersmicroservice.log" -v c:/users/manis/docklog:/api-logs -p 8500:8888 --detach --name miniuser user:miniuser

docker run --restart always --network sasi -v c:/users/manis/docklog:/api-logs -p 8500:8888 --detach --name miniuser user:miniuser

docker run --restart always --network sasi -v c:/users/manis/docklog:/api-logs --name miniuserlogstash logstash

elasticsearch

docker run -d -v esdata1:/usr/share/elasticsearch/data --name elasticsearch -p 9200:9200 -p 9300:9300 -e "discovery.type=single-node" elasticsearch:7.3.0

docker run -d --network sasi -p 5601:5601 --name kibana kibana:7.3.0

kibana

CUSTOM GATEWAY

docker run --restart always --network sasi -p 8012:8080 --detach --name minigateway customgateway:minigateway

IMPORTNT

cloud bee plugin anme for giving secret

key : AWS_ACCESS_KEY_ID

secret: AWS_SECRET_ACCESS_KEY

SSH-Agent plugin for connecting to EC2

in jenkins after ssh with private key (they give the .pem file from aws in jenkins to connect)

username is ec2-user

sftp -i .pem ec2-user@xx.xxx.xxx.xx

put ..file or path..file

export ANSIBLE_INVENTORY=/etc/ansible/ec2.py

/etc/ansible/ec2.py – list

This can be run to get the list of ec2 instance in the configured area