Microservice Patterns

The following list of Microservice Patterns was applied so far.

Server-side service discovery - Consul

API Gateway - Spring Cloud Zuul

Externalized configuration - Consul using Spring Cloud Config yaml format(with spring profiles), more details look at docker/spring-cloud-config

Exception Tracking - Spring Boot Admin

Access token - Spring Oauth2 with JWT

Health Check API - Spring Boot Actuator Starter

Distributed tracing - Jaeger

Application metrics - Spring Micrometer Prometheus

Database per service - MongoDB an instance per service

Shared database - Redis for sharing http sessions

To know more about each pattern look at Microservice Architecture

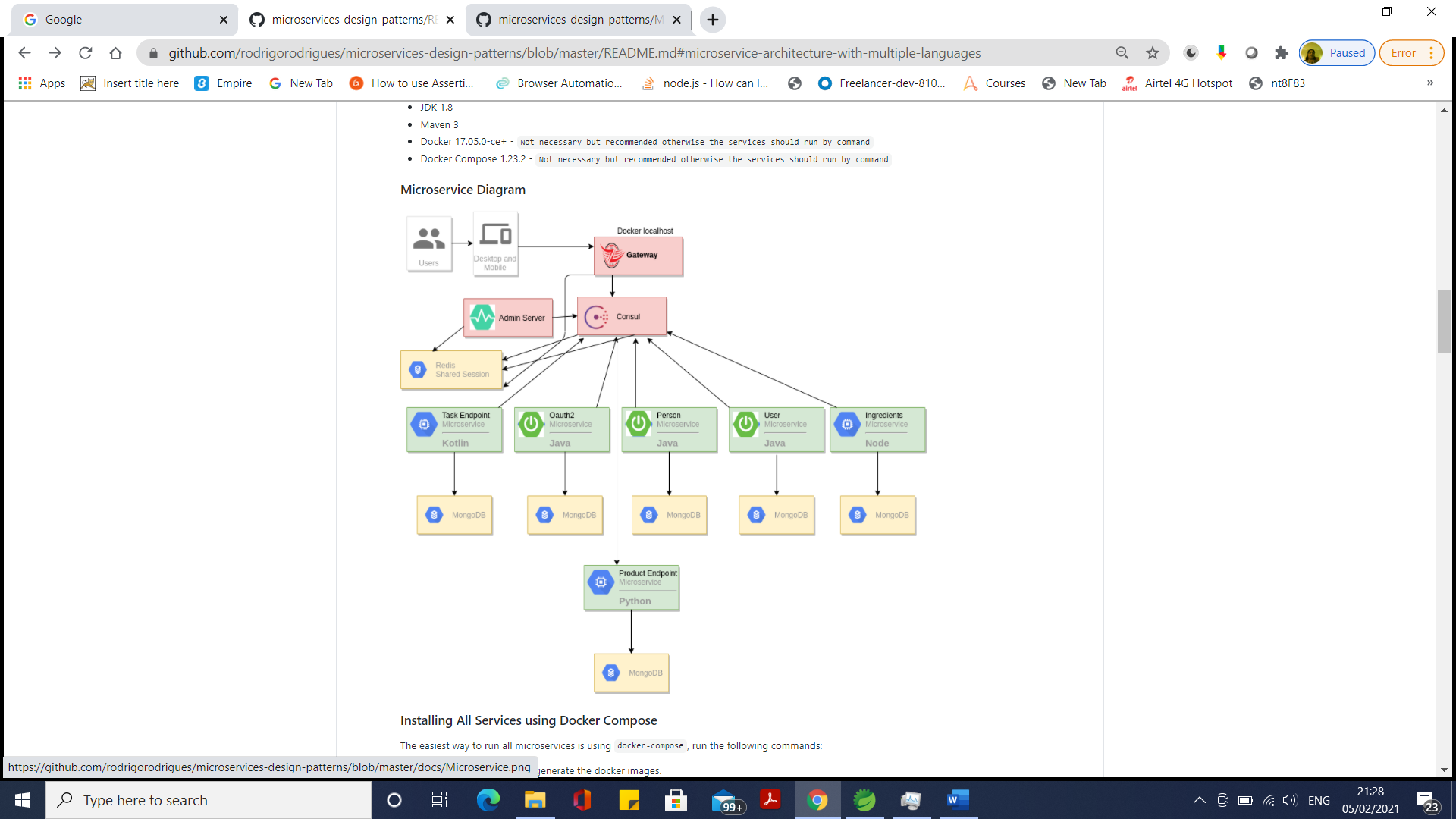
Prerequisites

JDK 1.8

Maven 3

Docker 17.05.0-ce+ - Not necessary but recommended otherwise the services should run by command

Docker Compose 1.23.2 - Not necessary but recommended otherwise the services should run by command



Installing All Services using Docker Compose

The easiest way to run all microservices is using docker-compose, run the following commands:

On root folder first need to generate the docker images.

# at once for building the docker images

mvn clean install docker:build

On docker folder run all microservices

docker-compose up -d

Docker Commands

To see logs for a specific docker container:

docker logs -f SERVICE\_NAME

PS: Service names are on docker-compose.yml -> container\_name

To execute a command inside the container:

docker exec -it week-menu-api sh

To stop and remove all containers:

docker-compose down -v

To restart/start/stop/remove specific container:

docker-compose restart SERVICE\_NAME

docker-compose up SERVICE\_NAME

docker-compose stop SERVICE\_NAME

docker-compose rm SERVICE\_NAME

**Manual Installation - NOT RECOMMENDED**

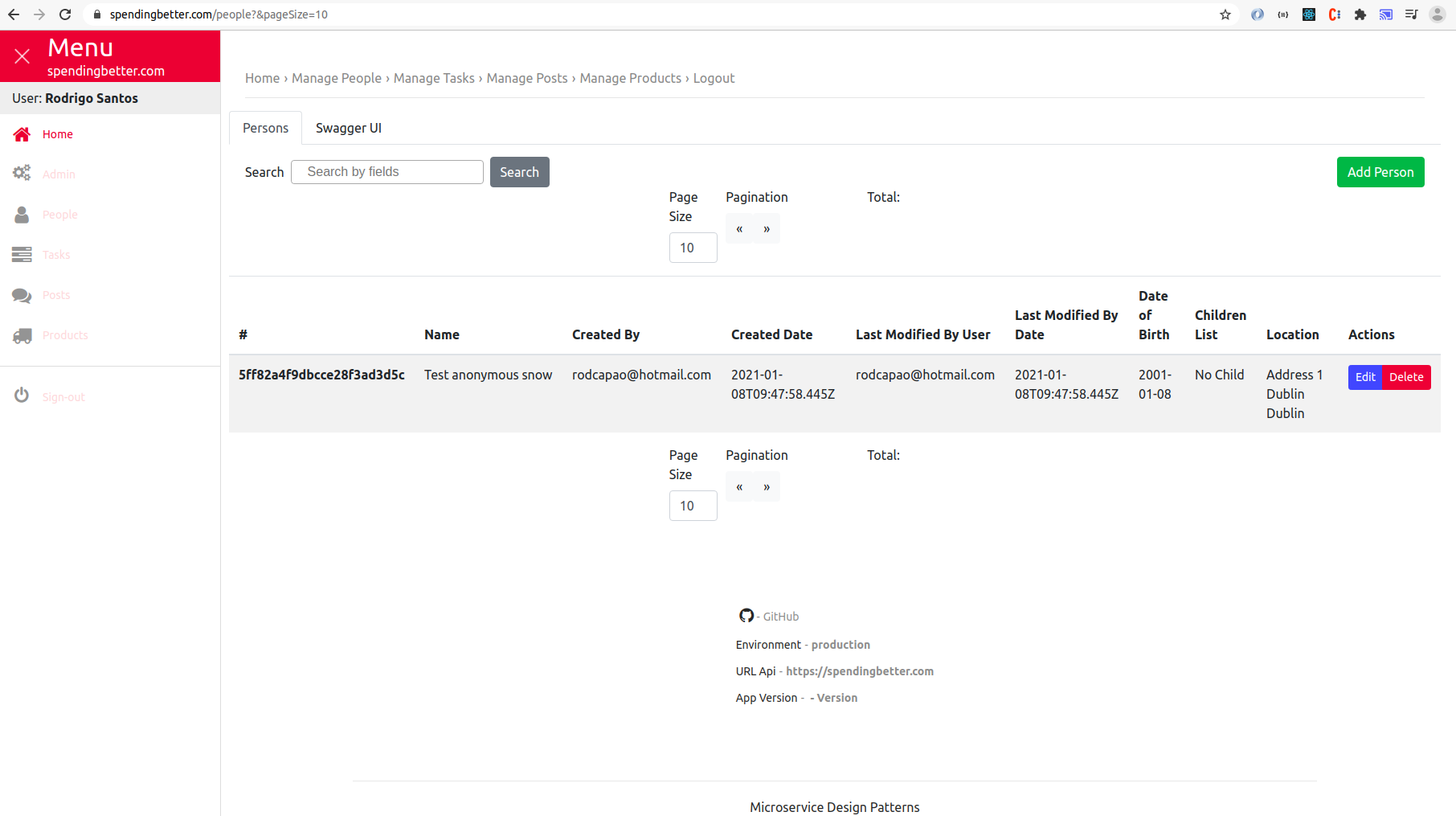
If for some reason you cannot install docker/docker-compose you can run all services manually using the following command for Java applications.

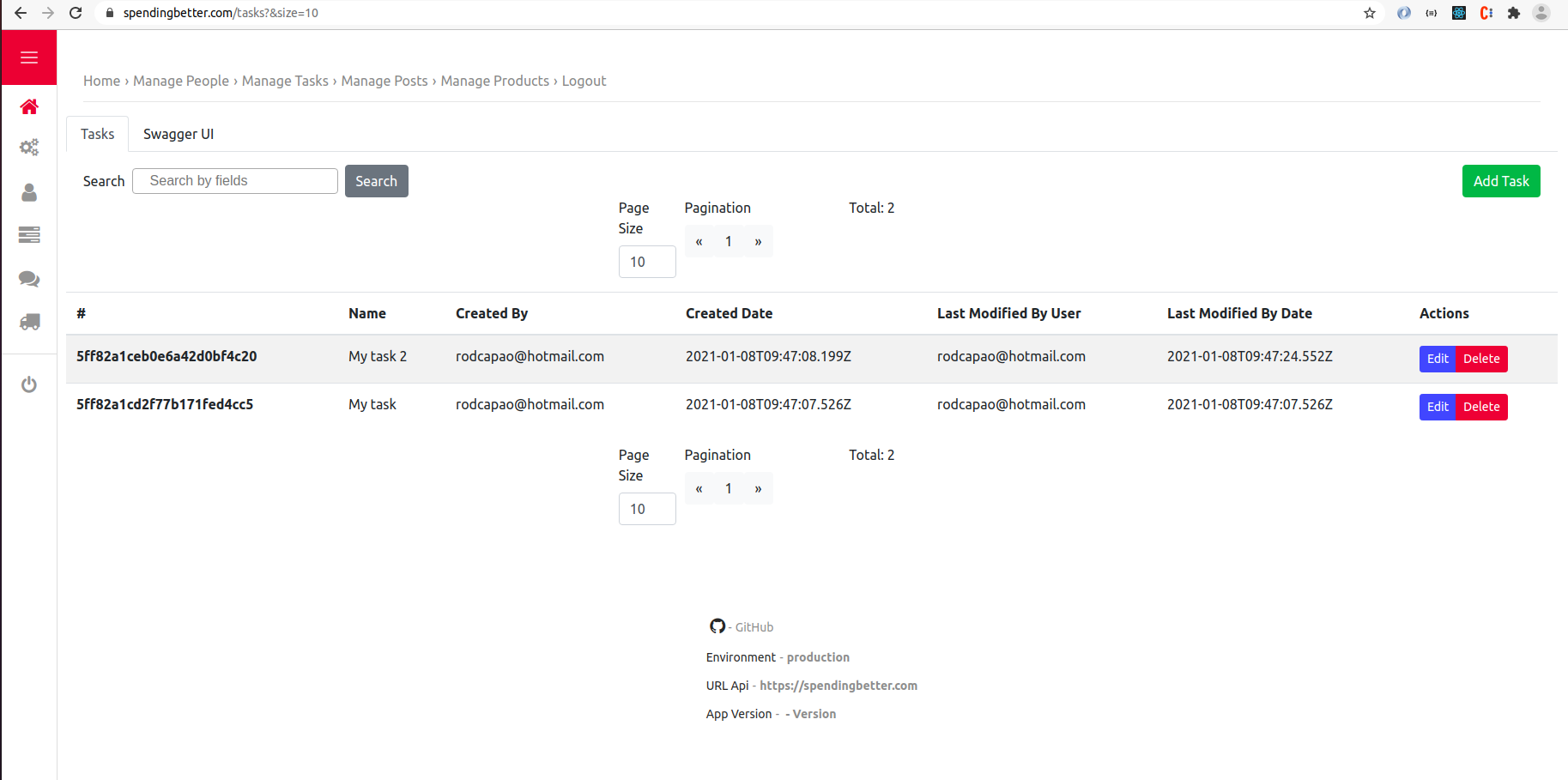
mvn spring-boot:run -Dspring-boot.run.arguments="--server.port={PORT}"

To run NodeJS and React applications on folders nodejs-service and react-webapp:

sudo npm install

sudo npm start





### Default Users

admin@gmail.com/password - ROLE\_ADMIN

master@gmail.com/password123 - ROLE\_PERSON\_CREATE, ROLE\_PERSON\_READ, ROLE\_PERSON\_SAVE

anonymous@gmail.com/test - ROLE\_PERSON\_READ

PS: Moved default users for Integration Tests only.

### Kubernetes - Google Cloud Platform

The code in Raspberry Pi Cluster using microk8s.

Following useful commands for kubernetes

Installation

#helm create ingress - RBAC enabled

kubectl create serviceaccount --namespace kube-system tiller

kubectl create clusterrolebinding tiller-cluster-rule --clusterrole=cluster-admin --serviceaccount=kube-system:tiller

helm init --service-account tiller

helm list

helm init --tiller-tls-verify

helm init

kubectl get deployments -n kube-system

helm install --name nginx-ingress stable/nginx-ingress --set rbac.create=true --set controller.publishService.enabled=true

#helm list

helm list

#create tls

kubectl create secret tls ingress-tls --cert /etc/sslmate/www.spendingbetter.com.chained.crt --key /etc/sslmate/www.spendingbetter.com.key

#create generic certs

kubectl create secret generic spendingbetter-p12 --from-file=/etc/sslmate/www.spendingbetter.com.p12

kubectl create secret generic spendingbetter-crt --from-file=/etc/sslmate/www.spendingbetter.com.crt

kubectl create secret generic spendingbetter-jks --from-file=/etc/sslmate/www.spendingbetter.com.jks

#list certs

kubectl get secrets

#list specific cert

kubectl describe secret ingress-tls

#show ingress

kubectl get ing

kubectl describe ingress

# Istio

# Get Grafana Configuration

kubectl get service grafana --namespace istio-system -o yaml

# Update Grafana Configuration

kubectl edit service grafana --namespace istio-system

Deployment

cd kubernetes

#create docker image

docker tag eureka-server:latest eu.gcr.io/spring-boot-gke-243520/eureka-server:4.0

docker tag docker\_react-webapp:latest eu.gcr.io/spring-boot-gke-243520/react-webapp:6.0

#push docker image

docker push eu.gcr.io/spring-boot-gke-243520/eureka-server:4.0

docker push eu.gcr.io/spring-boot-gke-243520/react-webapp:6.0

#Deploy

kubectl apply -f deployment-admin-server.yml

#Undeploy

kubectl delete -f deployment-admin-server.yml

#see logs

kubectl logs admin-server-XXXXX -f

#exec command

kubectl exec -it redis-5b4699dd74-qckm9 -- sh

#show all pods

kubectl get pods --show-labels

#create config map

kubectl create configmap prometheus --from-file=../docker/prometheus-prod.yml

kubectl create configmap grafana-dashboard --from-file=../docker/create-datasource-and-dashboard.sh

kubectl create configmap grafana-datasource --from-file=../docker/grafana-datasource.yaml

#port forward

kubectl port-forward $(kubectl get pod --selector="app=eureka-server" --output jsonpath='{.items[0].metadata.name}') 8761:8761

#delete specific ingress

kubectl delete ingress ingress-gateway-forward-https

#cpu usage

kubectl get nodes --show-labels

kubectl describe nodes gke-your-first-cluster

kubectl top nodes

Enable Ingress

Example Ingress Configuration

Install Helm

Kubernetes + Zuul

Example Spring Boot 2 + Kubernetes + Zuul

Secure Discovery Example

Travis CI/CD

Used travis-ci for building pull requests only.

Github Actions CI/CD

Using GitHub Actions for deploying services for multiple platforms(linux/amd64,linux/arm64).

More details look at .github/workflows/docker-build-push-\*.

Configuration(Deployment/Services) for Kubernetes look at .github/workflows/kubernetes.

Swagger UI

Swagger UI is available for Authentication, Person and User Services

Access it Swagger UI - http://localhost:{SERVICE\_PORT}/swagger-ui.html

TODO List

Java - Split Person and User in different entities

Java - Split back-end and front-end in two different folders

Java - Split Java 8 Learning in another folder

Java - Add Test for Users Classes

Java - Add Spring Cloud Config

Java - Add Service Discovery(Eureka)

Java - Add Zuul(Gateway)

Java - Add Maven Docker Plugin

Java - Add Redis for Shared Session between applications

Java - Add Authentication for all applications

Java - Add Prometheus/Grafana for docker compose

Java - Add Oauth2 Security layer

Java - Fix Zuul/Edge Server for working with NodeJS Service

Kotlin - Add Service using Kotlin Language

Quarkus - Add Service using Quarkus framework

Scala - Add Service using Scala Language

C# - Add Service using C# Language

Go - Add Service using Go Language

React - Create User List

React - Create User Page

React - Create User Edit

React - Create Categories Edit

React - Create Recipes Edit

React - Fix User Create/Edit

React - Fix Person Create/Edit

React - Fix Person List to work with @Tailable and EventSource.

React - Fix Docker Web App to use Nginx

Kubernetes/Minikube - Add example to use Kubernetes with Minikube

Deploy - Google Cloud/GKE

CI/CD - Add Travis

CI/CD - - Add Herokuy

CI/CD - Add GitHub Actions for deploy in GCP

Add documentation for libraries used

Add documentation/how-to for each language

Add tests for Python

Add React Legacy

Rename /api/persons to /api/people

Replace Eureka/Spring Config Server to Consul

Add Query DSL