

INSTITUTO INFNET
ESCOLA SUPERIOR DE TECNOLOGIA
GRADUAÇÃO EM ENGENHARIA DE SOFTWARE



Microserviços e DevOps com Spring Boot e
Spring Cloud

AT

Alunos: [Daniel Gomes Lipkin](#)

23 de set. de 2024

Questões

1 e 2) API Restful com Eureka e Gateway e testes dos endpoints

Instances currently registered with Eureka

Application	AMIs	Availability Zones	Status
LS-FUNCIONARIO-SERVICE	n/a (1)	(1)	UP (1) - LS-funcionario-service_0:9fb7f4dd12f1986494a7656d36e18c18
LS-GATEWAY-SERVICE	n/a (1)	(1)	UP (1) - 16d93945db25:LS-gateway-service:8080
LS-MISSAO-SERVICE	n/a (1)	(1)	UP (1) - LS-missao-service_0:4811447a9a577ed1270e98a7f2a6f6b3

LS-Missao-Service :

 POST ▼ http://localhost:8080/missao

FORM

RAW

```
1 {
2   "titulo": "Terminate Marcous",
3   "desc": "Terminate Marcous with no witnesses",
4   "ativa": true,
5   "prioridade": 5
6 }
```

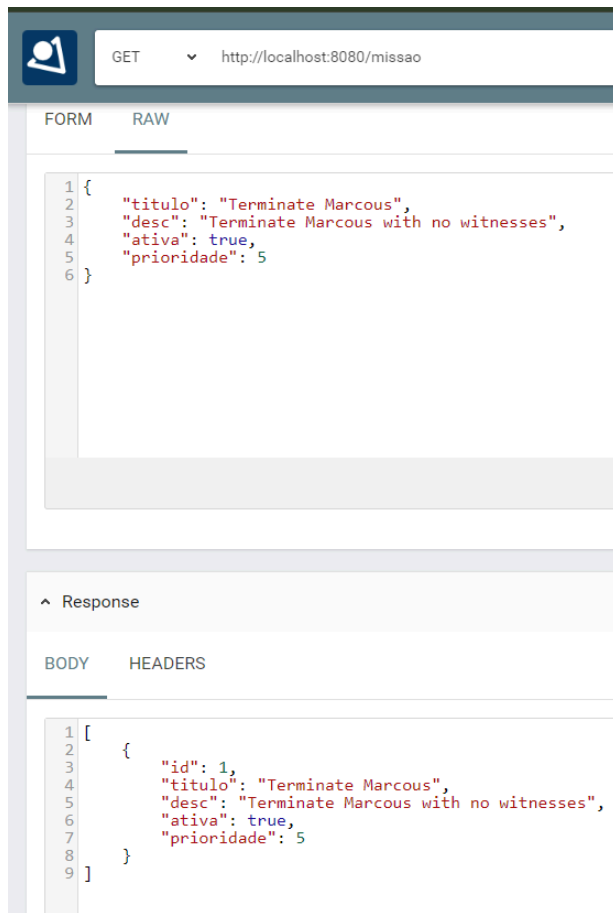
^ Response

BODY

HEADERS

```
1 {
2   "id": 1,
3   "titulo": "Terminate Marcous",
4   "desc": "Terminate Marcous with no witnesses",
5   "ativa": true,
6   "prioridade": 5
7 }
```

2



3) Para migrar uma aplicação Docker para Kubernetes é necessário criar para cada imagem dos micro serviços uma configuração de Serviço (para expor a porta á redes externas e gerenciar o endereçamento) e uma configuração de Deployment (para fazer o trabalho de balanceamento de carga e escalonamento). Isso é geralmente feito em um arquivo .yaml que depois é passado para o comando *"kubectl apply -f nome.yaml"*. Vera que com *"kubectl get pods"* ou *"kubectl get services"* aparecera os serviços e pods criados para as imagens. Volumes para banco de dados e redes internas tambem podem ser definidos opcionalmente.

Os beneficios estão principalmente no fato de ser nativo a nuvem, com varias empresas (Google, Amazon, etc) fornecendo plataformas para hostear clusters de Kubernetes. Alem disso ha tambem o fato que um container pode ser hosteado varias vezes (pods) em diversos hosts diferentes (nodes), replicas de imagens são ajustadas automaticamente dependendo do trafego, balanceamento de carga automática entre diferentes hosts e containers, reinicialização automática caso caia um dos sistemas, e integração sem dificuldades com workflows de CI/CD quando for atualizado alguma imagem ou uma parte da configuração.

4) Container Docker

<



devopsat

C:\Users\donke\Documents\lnfn\devopsat

Open



 devopsat-missao-... devopsat-missao-ser Exited (143)	▶ : 	2024-09-23 01:04:05 gateway-server-1 2024-09-23T04:04:05.342Z INFO 1 --- teaway-service] [reshExecutor-%d] com.netflix.discovery.DiscoveryClient : The re .status is 200
 devopsat-funcion... devopsat-funcionaric Exited (143)	▶ : 	2024-09-23 01:04:05 gateway-server-1 2024-09-23T04:04:05.355Z INFO 1 --- teaway-service] [beatExecutor-%d] com.netflix.discovery.DiscoveryClient : Discov ent_LS-GATEWAY-SERVICE/16d93945db25:LS-gateway-service:8080 - registration status: 2024-09-23 01:07:20 gateway-server-1 2024-09-23T04:07:20.112Z INFO 1 --- teaway-service] [ionShutdownHook] o.s.c.n.e.s.EurekaServiceRegistry : Unregi g application LS-GATEWAY-SERVICE with eureka with status DOWN
 devopsat-gateway... devopsat-gateway-sf Exited (143) 8080:8080	▶ : 	2024-09-23 01:07:20 gateway-server-1 2024-09-23T04:07:20.113Z INFO 1 --- teaway-service] [ionShutdownHook] com.netflix.discovery.DiscoveryClient : Saw lo atus change event StatusChangeEvent [timestamp=1727064440113, current=DOWN, previc 2024-09-23 01:07:20 gateway-server-1 2024-09-23T04:07:20.114Z INFO 1 --- teaway-service] [foReplicator-%d] com.netflix.discovery.DiscoveryClient : Discov ent_LS-GATEWAY-SERVICE/16d93945db25:LS-gateway-service:8080: registering service..
 devopsat-msr-pla... devopsat-msr-planet Exited (143) 8082:8082	▶ : 	2024-09-23 01:07:20 gateway-server-1 2024-09-23T04:07:20.131Z INFO 1 --- teaway-service] [foReplicator-%d] com.netflix.discovery.DiscoveryClient : Discov ent_LS-GATEWAY-SERVICE/16d93945db25:LS-gateway-service:8080 - registration status: 2024-09-23 01:07:22 gateway-server-1 2024-09-23T04:07:22.160Z INFO 1 --- teaway-service] [ionShutdownHook] com.netflix.discovery.DiscoveryClient : Shutt n DiscoveryClient ...
 devopsat-eureka-... devopsat-eureka-ser Exited (143) 8761:8761	▶ : 	2024-09-23 01:07:25 gateway-server-1 2024-09-23T04:07:25.161Z INFO 1 --- teaway-service] [ionShutdownHook] com.netflix.discovery.DiscoveryClient : Unregi g ... 2024-09-23 01:07:25 gateway-server-1 2024-09-23T04:07:25.174Z INFO 1 --- teaway-service] [ionShutdownHook] com.netflix.discovery.DiscoveryClient : Discov

5) Implementação Kubernetes

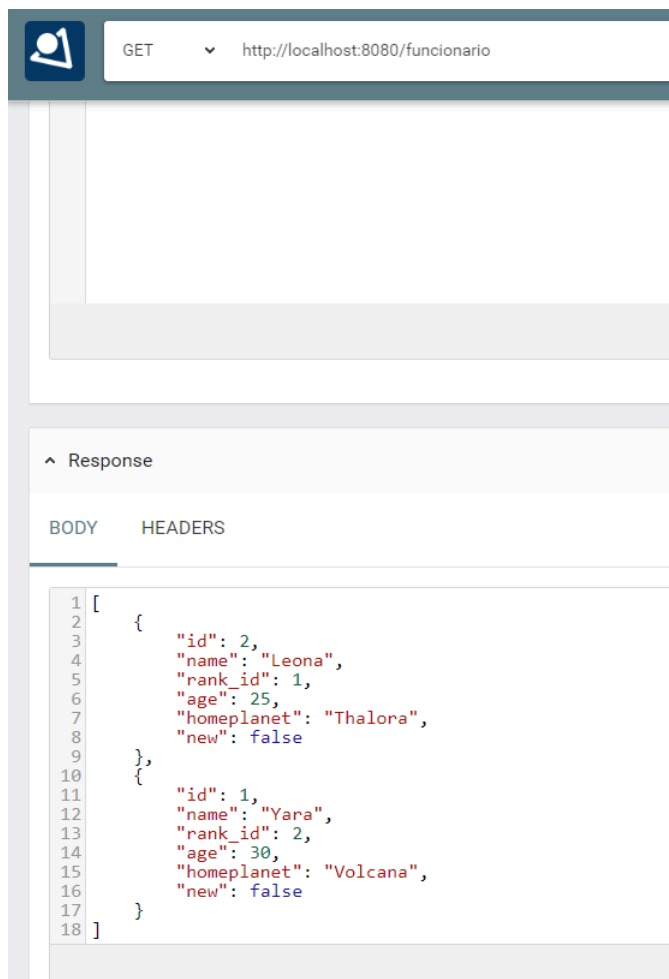
LS-Funcionario-Service :

```
deploy.yml x
Kubernetes files are supported by IntelliJ IDEA Ultimate

1  apiVersion: apps/v1
2  kind: Deployment
3  metadata:
4    name: funcionario-service
5    labels:
6      app: funcionario-service
7  spec:
8    replicas: 1
9    selector:
10     matchLabels:
11       app: funcionario-service
12    template:
13     metadata:
14       labels:
15         app: funcionario-service
16     spec:
17       containers:
18         - name: funcionario-service
19           image: bagelboi/funcionario-service:latest
20           ports:
21             - containerPort: 8080
22           env:
23             - name: JAVA_OPTS
24               value: "-Xms512m -Xmx1024m"
25             - name: EUREKA_CLIENT_SERVICEURL_DEFAULTZONE
26               value: http://eureka-server-service:8761/eureka
27
28  ---
29  apiVersion: v1
30  kind: Service
31  metadata:
32    name: funcionario-service
33  spec:
34    type: LoadBalancer
35    ports:
36      - port: 80
37        targetPort: 8080
38    selector:
39      app: funcionario-service
40
```

6) Cliente reativo com GET e POST

LS-Funcionario-Service :



7) Consumir API externa e Test Containers

LS-Missao-Service consome da aplicação MSR-Planets

```
@SpringBootTest(webEnvironment = SpringBootTest.WebEnvironment.RANDOM_PORT)
public class PlanetContainerTest {

    public static GenericContainer<?> planetService = new GenericContainer<>(
        dockerImageName: "bagelboi/msrplanets:latest")
        .withExposedPorts(8082);

    private static WebClient webClient;

    @BeforeAll
    public static void setUp() {
        String baseUrl = "http://" + planetService.getHost() + ":8082" + "/msr/planeta";
        webClient = WebClient.builder().baseUrl(baseUrl).build();
    }

    @Test
    public void testGetAllPlanets() {
        Mono<List<PlanetaDTO>> planetListMono = webClient.get()
            .retrieve()
            .bodyToFlux(PlanetaDTO.class)
            .collectList();
    }

    @Test
    public void testGetPlanetById() {
        Long planetId = 1L; // Assuming a planet with ID 1 exists
        Mono<PlanetaDTO> planetMono = webClient.get()
            .uri(uri:("/{id}", planetId)
            .retrieve()
            .bodyToMono(PlanetaDTO.class);
    }
}
```

 GET  http://localhost:8080/planeta

^ Response

BODYHEADERS

```
1  [
2    {
3      "id": 1,
4      "nome": "Zogara"
5    },
6    {
7      "id": 2,
8      "nome": "Thalora"
9    },
10   {
11     "id": 3,
12     "nome": "Aquarion"
13   },
14   {
15     "id": 4,
16     "nome": "Volcana"
17   },
18   {
19     "id": 5,
20     "nome": "Frostus"
21   }
22 ]
```


8) Commit no Github

```
C:\Users\donke\Documents\Infnet\devopsat\LS-funcionario-service>git add .

C:\Users\donke\Documents\Infnet\devopsat\LS-funcionario-service>git commit -m "push test"
[main ae61bc2] push test
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 push.txt

C:\Users\donke\Documents\Infnet\devopsat\LS-funcionario-service>git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
(use "git push" to publish your local commits)

nothing to commit, working tree clean

C:\Users\donke\Documents\Infnet\devopsat\LS-funcionario-service>git add .

C:\Users\donke\Documents\Infnet\devopsat\LS-funcionario-service>git commit -m "actions fix"
[main b074401] actions fix
1 file changed, 1 insertion(+)

C:\Users\donke\Documents\Infnet\devopsat\LS-funcionario-service>git push
Enumerating objects: 12, done.
Counting objects: 100% (12/12), done.
Delta compression using up to 8 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (8/8), 700 bytes | 350.00 KiB/s, done.
Total 8 (delta 3), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (3/3), completed with 2 local objects.
To https://github.com/Bagelboi/ls-funcionario-service.git
   ac4d0df..b074401  main -> main
```

9) O Github Actions automatiza o processo de CI/CD fornecendo maquinas virtuais (Runners) que executam um conjunto de comandos estilo cmd/Powershell (Steps) quando um evento for disparado no repositório (Pull, Push, etc). Esse conjunto de comandos se chama Job e 1 ou mais Jobs podem ser executados em um Workflow definido por um arquivo .yaml em “.github/workflows”.

Os beneficios são o fato de você poder atualizar seu código-fonte e o repositório sem precisar manualmente testar ou atualizar outros aspectos de produção como a imagem Docker ou a configuração do Kubernetes graças aos Jobs, alem de testar a compatibilidade do programa em outras plataformas com as opções de Runners.

10) Crie um arquivo .yaml de configuração Kubernetes para um dos serviços Spring Boot e depois coloque em um dos Workflows do repositório.

```
- name: Nome do Step
  run: kubectl apply -f ./deploy.yml
```

Antes disso podem ser executados Steps para se autenticar e conectar com serviços de nuvem para clusters a fim de utilizá-los, como exemplo o Google Kubernetes Engine usando

```
- name: Set up Google Cloud
```

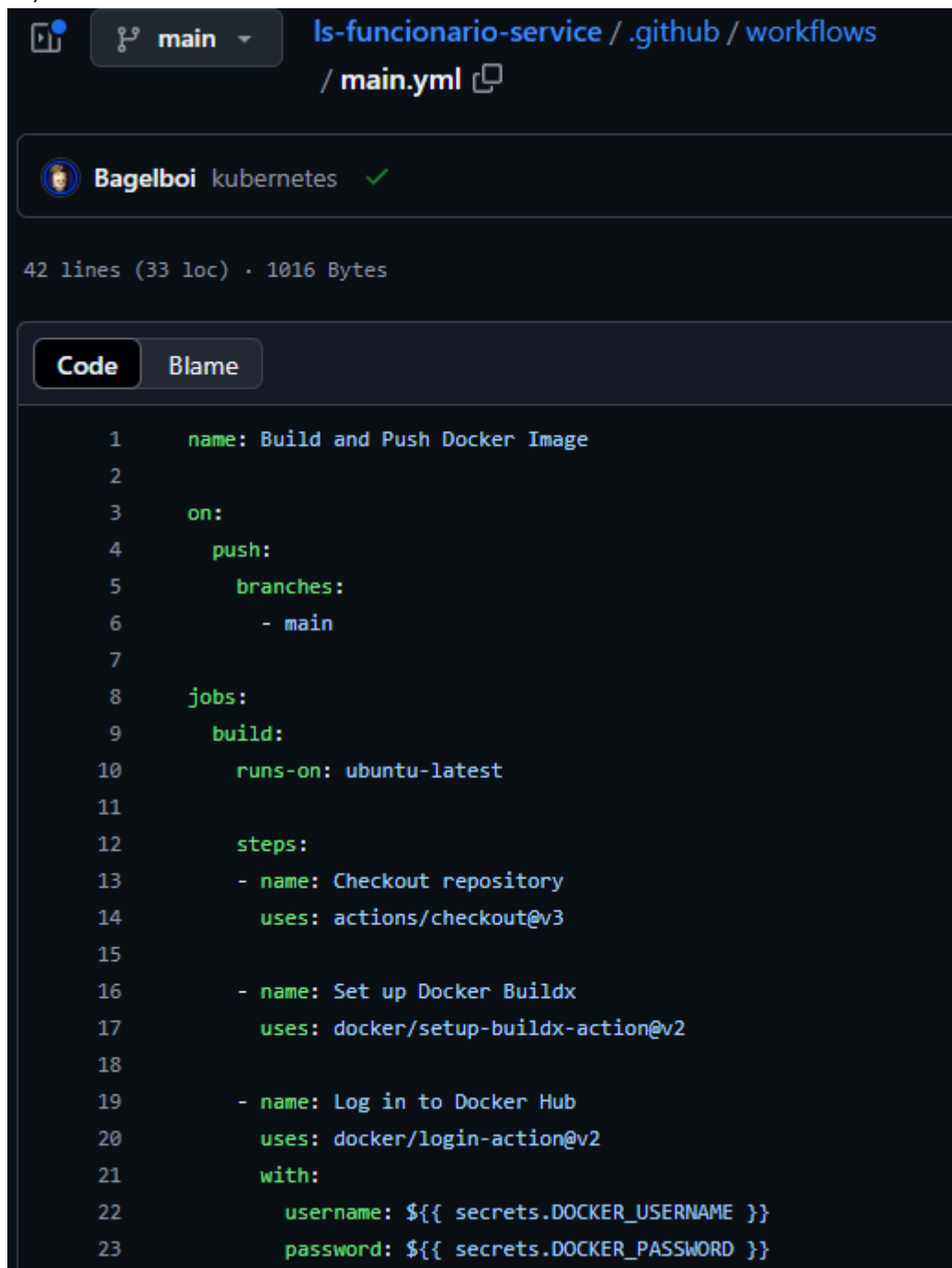
uses: google-github-actions/setup-gcloud@v0.2.0
with:
service_account_key: \${{secrets.GKE_LOGIN}} #Colocar credencias do GKE nos Secrets
do Github

- name: Deploy to GKE

run:

gcloud container clusters get-credentials ls-cluster --zone us-central1-a

11) Actions



```
1  name: Build and Push Docker Image
2
3  on:
4    push:
5      branches:
6        - main
7
8  jobs:
9    build:
10     runs-on: ubuntu-latest
11
12     steps:
13       - name: Checkout repository
14         uses: actions/checkout@v3
15
16       - name: Set up Docker Buildx
17         uses: docker/setup-buildx-action@v2
18
19       - name: Log in to Docker Hub
20         uses: docker/login-action@v2
21         with:
22           username: ${{ secrets.DOCKER_USERNAME }}
23           password: ${{ secrets.DOCKER_PASSWORD }}
```

```

- name: Build and push Docker image
  uses: docker/build-push-action@v5
  with:
    context: .
    file: ./Dockerfile
    push: true
    tags: ${ secrets.DOCKER_USERNAME }}/funcionario-service:latest

```

12) GKE

```

# - name: Set up Google Cloud
#   uses: google-github-actions/setup-gcloud@v0.2.0
#   with:
#     service_account_key: ${ secrets.GKE_LOGIN }

# - name: Deploy to GKE
#   run: |
#     gcloud container clusters get-credentials ls-cluster --zone us-central1-a
#     kubectl apply -f ./deploy.yml

```

Código

Github :

- <https://github.com/Bagelboi/ls-funcionario-service>
- <https://github.com/Bagelboi/ls-missao-service>
- <https://github.com/Bagelboi/ls-eureka-server>
- <https://github.com/Bagelboi/ls-gateway-server>

Docker :

- bagelboi/funcionario-service
- bagelboi/missao-service
- bagelboi/ls-eureka-server
- bagelboi/ls-gateway-service
- bagelboi/msrplanets