

**UNIVERSIDADE FEDERAL DE PERNAMBUCO**  
**CENTRO DE INFORMÁTICA - 2018.1**

Technical Report - IF1007 Microserviços  
PORQUIN

Bruno Filho (brgccf)  
Matheus de La Rocque (mlrbc)  
Vinícius de Moraes (vrn)

# Sumário

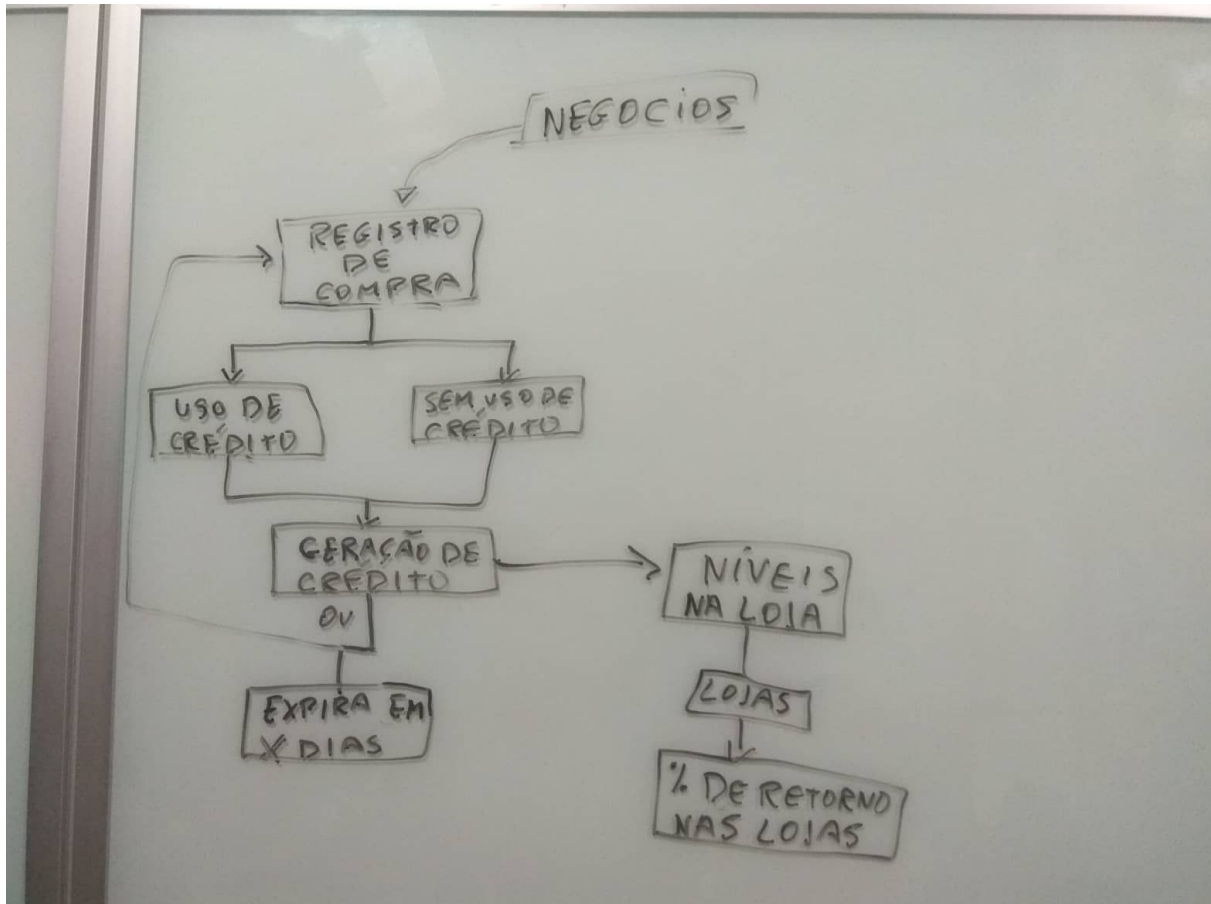
<b>Sumário</b>	<b>2</b>
<b>Introdução</b>	<b>3</b>
<b>Arquitetura</b>	<b>4</b>
<b>Projeto</b>	<b>5</b>
<b>Execução</b>	<b>10</b>
<b>Referências</b>	<b>11</b>

# Introdução

Projeto que utiliza de base a proposta da startup Porquin, voltada à fidelização de clientes de varejistas de moda feminina. A proposta se baseia no conceito de “créditos para compras seguintes”, no qual quando um consumidor realiza uma compra numa das lojas parceiras Porquin, uma parte do valor gasto (porcentagem) na compra volta para sua carteira virtual (aplicativo) em forma de crédito, disponibilizado para ser utilizado na compra seguinte em forma de descontos. Quanto mais um mesmo consumidor realiza compras num mesmo estabelecimento parceiro, maior é sua taxa de créditos retornada no aplicativo, estimulando-o a continuar comprando na mesma loja envolvida.

# Arquitetura

Segue abaixo um MindMap que detalha a ideia da arquitetura geral do projeto:



A divisão do projeto foi feita da seguinte forma:

- Foi criado um microserviço específico para lidar com os usuários e suas respectivas carteiras (onde ficam armazenados seus créditos), o creditService.
- Foi criado outro microserviço específico para lidar com as lojas, informação de níveis e porcentagens de retorno, o storeService.

Quando um registro de compra é feito no creditService, o mesmo consulta o storeService, para buscar informações sobre os níveis da loja e poder calcular quanto de crédito aquela compra vai gerar para o usuário.

# Projeto

Cada microserviço tem 3 arquivos de configuração de ambiente (application-\*.yml), prod, test, dev. Em ambiente de produção eles utilizam o mongoDB da mLab e caso contrário tentam se conectar em um local.

Para a documentação dos microserviços utilizamos o Swagger:



## Api Documentation

Api Documentation

[Apache 2.0](#)

### store-controller : Store Controller

[Show/Hide](#) | [List Operations](#) | [Expand Operations](#)

PUT	/levels	Update levels of a store
GET	/login	Get a store login token
GET	/registerinfo	Infos to make a register
DELETE	/store	Remove the store itself
POST	/store	Create a store

[ BASE URL: / , API VERSION: 1.0 ]



## Api Documentation

Api Documentation

[Apache 2.0](#)

### credit-controller : Credit Controller

[Show/Hide](#) | [List Operations](#) | [Expand Operations](#)

GET	/credit	Return user credit
POST	/purchase	Register a purchase
DELETE	/wallet	Remove a wallet

[ BASE URL: / , API VERSION: 1.0 ]

Inicialmente rodamos o Kubernetes localmente com o minikube:

Workloads

Workloads Statuses

100.00%

Deployments

100.00%

Pods

100.00%

Replica Sets

Deployments

Name ↕	Labels	Pods	Age ↕	Images	
✓ <a href="#">creditservice</a>	k8s-app: creditservice	3 / 3	15 hours	mlrbc/creditservice	⋮
✓ <a href="#">storeservice</a>	k8s-app: storeservice	3 / 3	15 hours	mlrbc/storeservice	⋮

Pods

Name ↕	Node	Status ↕	Restarts	Age ↕	
✓ <a href="#">creditservice-b6cdf6fb-7gmb4</a>	minikube	Running	1	15 hours	⋮
✓ <a href="#">creditservice-b6cdf6fb-m69nj</a>	minikube	Running	1	15 hours	⋮
✓ <a href="#">creditservice-b6cdf6fb-wl7tm</a>	minikube	Running	1	15 hours	⋮
✓ <a href="#">storeservice-d79f8954-hgps9</a>	minikube	Running	1	15 hours	⋮
✓ <a href="#">storeservice-d79f8954-v55l4</a>	minikube	Running	1	15 hours	⋮
✓ <a href="#">storeservice-d79f8954-zb9kh</a>	minikube	Running	1	15 hours	⋮

Replica Sets

Name ↕	Labels	Pods	Age ↕	Images	
✓ <a href="#">creditservice-b6cdf6fb</a>	k8s-app: creditservice pod-template-hash: 627899296	3 / 3	15 hours	mlrbc/creditservice	⋮
✓ <a href="#">storeservice-d79f8954</a>	k8s-app: storeservice pod-template-hash: 83594510	3 / 3	15 hours	mlrbc/storeservice	⋮

Discovery and Load Balancing

Services

Name ↕	Labels	Cluster IP	Internal endpoints	External endpoints	Age ↕	
✓ <a href="#">kubernetes</a>	component: apiserver provider: kubernetes	10.96.0.1	kubernetes:443 TCP kubernetes:0 TCP	-	16 hours	⋮

Config and Storage

Secrets

Name ↕	Type	Age ↕	
<a href="#">default-token-ggnp</a>	kubernetes.io/service-account-token	16 hours	⋮

Creation Time: 2018-06-24T06:40 UTC

Selector: [k8s-app: creditservice](#)

Strategy: RollingUpdate

Min ready seconds: 0

Revision history limit: 10

Rolling update strategy: Max surge: 25%, Max unavailable: 25%

Status: 3 updated, 3 total, 3 available, 0 unavailable

New Replica Set

Name	Labels	Pods	Age	Images	
✓ <a href="#">creditservice-b6cdf6fb</a>	k8s-app: creditservice pod-template-hash: 627899296	3 / 3	15 hours	mlrbc/creditservice	⋮

Old Replica Sets

There is nothing to display here

This Deployment does not have any old replica sets

Horizontal Pod Autoscalers

There is nothing to display here

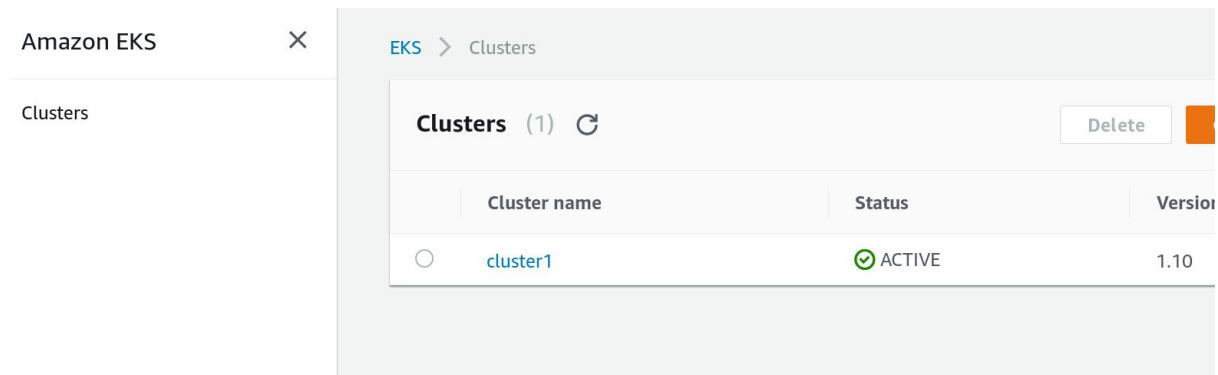
There are currently no Horizontal Pod Autoscalers targeting this Deployment.

Events

Message	Source	Sub-object	Count	First seen	Last seen
Scaled up replica set creditservice-b6cdf6fb to 3	deployment-controller	-	1	2018-06-24T05:45 UTC	2018-06-24T05:45 UTC
Scaled up replica set creditservice-b6cdf6fb to 3	deployment-controller	-	1	2018-06-24T06:40 UTC	2018-06-24T06:40 UTC

6

Porém depois conseguimos configurá-lo no EKS da Amazon, criando um cluster com 3 máquinas do EC2 como hosts:



<input type="checkbox"/>	Name	Instance ID	Instance Type	Availability Zone	Instance Status	Status Check	Alarm Status	Public DNS (IPv4)
<input checked="" type="checkbox"/>	cluster1-no...	i-05d929e16d51a...	t2.medium	us-east-1b	running	2/2 check...	None	ec2-54-224-62-244.c...
<input type="checkbox"/>	cluster1-no...	i-0c78591afef12f...	t2.medium	us-east-1a	running	2/2 check...	None	ec2-18-207-120-255....
<input type="checkbox"/>	cluster1-no...	i-0f4b5156e48d5...	t2.medium	us-east-1a	running	2/2 check...	None	ec2-54-152-218-175....

E então pelo kubectl enviamos jsons para criar o replication controller com seus pods e um service para o mesmo<sup>1</sup>:

```
wilson@apple ~ % kubectl get all -o wide
NAME                                READY    STATUS    RESTARTS   AGE    IP                                NODE
pod/creditservice-fx654             1/1     Running   0           18h    192.168.143.89                   ip-192-168-142-228.ec2.internal
pod/creditservice-q52jr             1/1     Running   0           18h    192.168.68.92                    ip-192-168-84-249.ec2.internal
pod/creditservice-z4z2l             1/1     Running   0           18h    192.168.121.112                  ip-192-168-104-195.ec2.internal
pod/logservice-vxptr               1/1     Running   0           18h    192.168.135.200                  ip-192-168-142-228.ec2.internal
pod/storeservice-5qn9b             1/1     Running   0           18h    192.168.69.15                    ip-192-168-104-195.ec2.internal
pod/storeservice-cmfhv             1/1     Running   0           18h    192.168.99.64                    ip-192-168-84-249.ec2.internal
pod/storeservice-z54xs             1/1     Running   0           18h    192.168.128.118                  ip-192-168-142-228.ec2.internal

NAME                                DESIRED   CURRENT   READY    AGE    CONTAINERS    IMAGES                SELECTOR
replicationcontroller/creditservice 3          3         3        18h    creditservice wison27/creditservice app=creditservice
replicationcontroller/logservice     1          1         1        18h    logservice    wison27/logservice    app=logservice
replicationcontroller/storeservice   3          3         3        18h    storeservice  wison27/storeservice  app=storeservice

NAME                                TYPE          CLUSTER-IP    EXTERNAL-IP    PORT(S)                AGE    SELECTOR
service/creditservice               LoadBalancer  10.100.174.94  ad15425b478bd11e8b1530e7f1a24d73-2110805670.us-east-1.elb.amazonaws.com 9090:30232/TCP 18h    app=creditservice
service/kubernetes                  ClusterIP      10.100.0.1     <none>          443/TCP                2d     <none>
service/logservice                  LoadBalancer  10.100.75.56  a91dbce7e78bc11e8b9110a405e61255-1646121970.us-east-1.elb.amazonaws.com 5601:31441/TCP,9200:31758/TCP 18h    app=logservice
service/storeservice                LoadBalancer  10.100.188.131 a1b9b465c78bd11e8b9110a405e61255-501333875.us-east-1.elb.amazonaws.com 8080:31753/TCP 18h    app=storeservice
```

O acesso a eles ainda está disponível em:

StoreService ->

<http://a1b9b465c78bd11e8b9110a405e61255-501333875.us-east-1.elb.amazonaws.com:8080/swagger-ui.html>

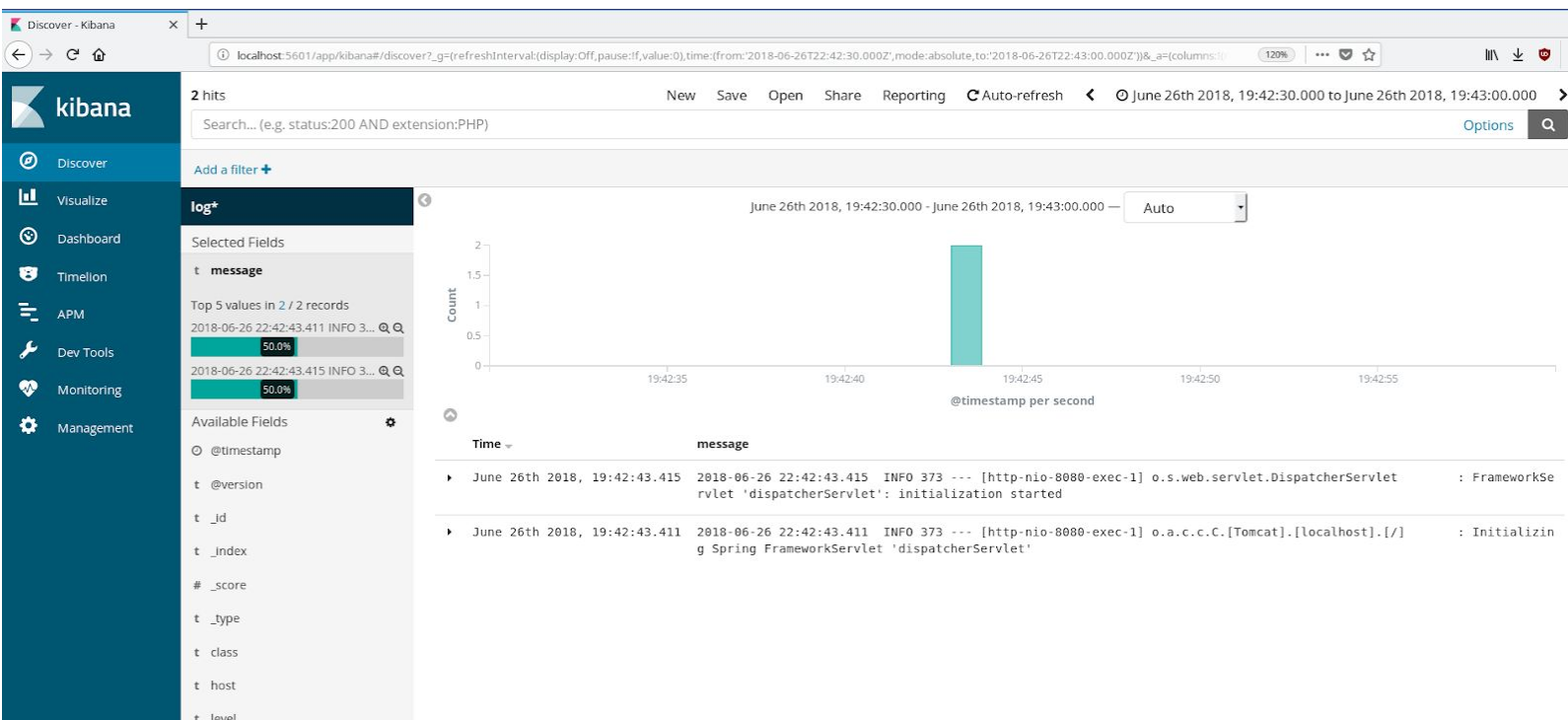
CreditService ->

<http://ad15425b478bd11e8b1530e7f1a24d73-2110805670.us-east-1.elb.amazonaws.com:9090/swagger-ui.html>

Para log e monitoramento utilizamos o conjunto denominado ELK, composto por Elasticsearch, Logstash e Kibana.

<sup>1</sup> <https://github.com/if1007/storeService/tree/master/eks>,  
<https://github.com/if1007/creditService/tree/master/eks>

Criamos um container denominado logservice que contém o Elasticsearch e o Kibana. E então colocamos o logstash em conjunto com o packetbeat nos containers do storeService<sup>2</sup> e creditService<sup>3</sup>, dessa forma conseguimos guardar logs e monitorar cada instância dos containers. Porém não conseguimos rodar o projeto corretamente no EKS, mesmo utilizando como base do logservice o container oficial do elasticsearch<sup>4</sup>, pois o mesmo requer que se altere algumas configurações default do kernel linux na máquina host<sup>5</sup>.



Para a API Gateway utilizamos o Amazon API Gateway, que permite que tudo seja configurado pela dashboard deles:

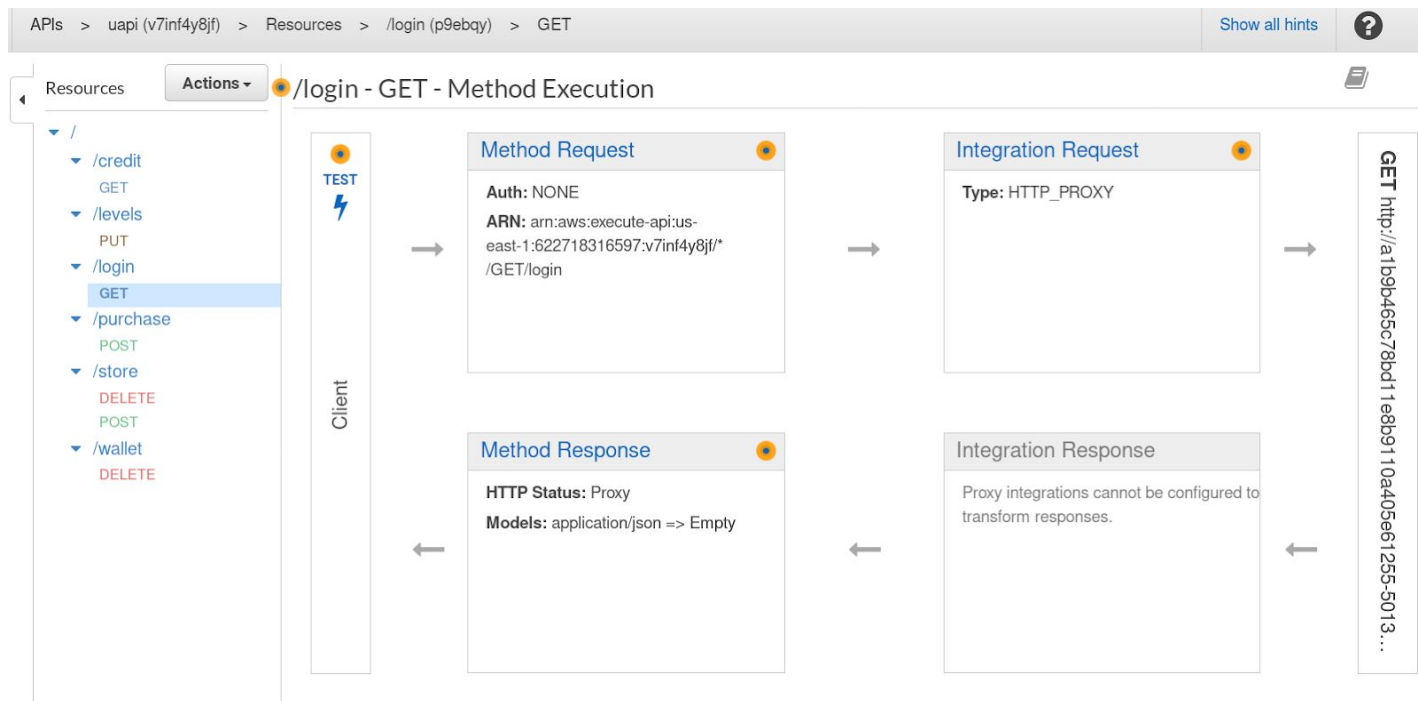
<sup>2</sup> <https://github.com/if1007/storeService/blob/master/log.sh>

<sup>3</sup> <https://github.com/if1007/creditService/blob/master/log.sh>

<sup>4</sup> <https://github.com/if1007/logservice/blob/master/Dockerfile#L1>

<sup>5</sup> <https://www.elastic.co/guide/en/elasticsearch/reference/current/vm-max-map-count.html>,  
<https://github.com/docker-library/elasticsearch/issues/111>





Estando disponível em: <https://v7inf4y8jf.execute-api.us-east-1.amazonaws.com/test/>

# Execução

Foram criados scripts para otimizar a execução dos projetos de várias maneiras. Todas as instruções para rodá-los localmente com (prod) e sem docker (dev), os testes (test) e ainda subí-los facilmente no EKS (se houver um arquivo kubeconfig do cluster configurado conforme a especificação do EKS<sup>6</sup>) encontram-se atualizadas nos seus respectivos README.md:

<https://github.com/if1007/storeService>

<https://github.com/if1007/creditService>

<https://github.com/if1007/logservice>

---

<sup>6</sup> [https://docs.aws.amazon.com/pt\\_br/eks/latest/userguide/getting-started.html](https://docs.aws.amazon.com/pt_br/eks/latest/userguide/getting-started.html)

# Referências

Abaixo seguem objetos de consulta utilizados ao longo do desenvolvimento do projeto para auxílio e solução de dúvidas.

- Java

<https://stackoverflow.com/questions/2607289/converting-array-to-list-in-java>  
<http://www.javarepl.com/term.html>

- Eclipse

<https://stackoverflow.com/questions/10722773/import-existing-gradle-git-project-into-eclipse/44715981#44715981>

- Spring Boot

<https://spring.io/guides/gs/actuator-service/>  
<https://stackoverflow.com/questions/23027315/does-application-yml-support-environment-variables/47527525#47527525>  
<https://emmanuelneri.com.br/2017/05/14/profiles-no-spring-boot/>

- Spring Boot Profile Config

<https://stackoverflow.com/questions/46331059/reading-application-yml-in-service>  
<https://www.mkyong.com/spring-boot/spring-boot-profile-based-properties-and-yaml-example/>

- Requisições HTTP

<https://spring.io/guides/gs/consuming-rest/>  
<https://stackoverflow.com/questions/42592440/how-to-change-response-http-header-in-get-request-by-spring-resttemplate>  
<https://stackoverflow.com/questions/10308452/how-to-convert-the-following-json-string-to-java-object>  
<https://stackoverflow.com/questions/23205213/how-to-extract-http-status-code-from-the-resttemplate-call-to-a-url>

- Swagger

<https://github.com/swagger-api/swagger-core/wiki/annotations>  
<https://dzone.com/articles/spring-boot-swagger-ui>  
<http://www.baeldung.com/swagger-2-documentation-for-spring-rest-api>

- Testes com Spring Boot

<https://www.youtube.com/watch?v=8S8o46avgAw>  
<https://docs.spring.io/spring-boot/docs/current/reference/html/boot-features-testing.html>  
<https://springframework.guru/unit-testing-spring-mvc-spring-boot-1-4-part-1/>

<https://www.petrikainulainen.net/programming/gradle/getting-started-with-gradle-integration-testing/>

<https://stackoverflow.com/questions/24199393/how-to-send-a-mock-object-as-json-in-mock-mvc/24200590#24200590>

<https://stackoverflow.com/questions/18336277/how-to-check-string-in-response-body-with-mockmvc/18336481>

- Api Gateway

[https://docs.aws.amazon.com/pt\\_br/apigateway/latest/developerguide/api-gateway-create-a-pi-as-simple-proxy-for-http.html](https://docs.aws.amazon.com/pt_br/apigateway/latest/developerguide/api-gateway-create-a-pi-as-simple-proxy-for-http.html)

<https://spring.io/guides/gs/gateway/>

- EKS

[https://docs.aws.amazon.com/pt\\_br/eks/latest/userguide/getting-started.html](https://docs.aws.amazon.com/pt_br/eks/latest/userguide/getting-started.html)

- Dockerfile

<https://stackoverflow.com/questions/19537645/get-environment-variable-value-in-dockerfile>

<https://docs.docker.com/engine/reference/builder/#workdir>

- Docker

<https://www.mundodocker.com.br/link-entre-containers-parte-2/>

<https://docs.docker.com/docker-cloud/builds/push-images/>

<https://stackoverflow.com/questions/38118791/can-t-delete-image-with-children>

<https://stackoverflow.com/questions/37782505/is-it-possible-to-show-the-workdir-when-building-a-docker-image>

<https://gist.github.com/bastman/5b57ddb3c11942094f8d0a97d461b430>

- Kubernetes

<https://www.mundodocker.com.br/kubernetes-parte-i/>

<https://www.mundodocker.com.br/kubernetes-parte-ii/>

<https://churrops.io/2018/06/19/kubernetes-criando-um-cluster-simples-em-cloud-com-o-kubernetes-adm/>

<https://www.youtube.com/watch?v=1ENtPSKjD2I&t=16s>

<https://github.com/vinicius3w/if1007-Microservices/blob/master/lectures/if1007-microservices-13-withnotes.pdf>

<https://aws.amazon.com/pt/kubernetes/>

- ELK

<https://howtodoinjava.com/microservices/elk-stack-tutorial-example/>

<https://www.elastic.co/guide/en/beats/packetbeat/current/packetbeat-getting-started.html>