



# VisionHive

*DEVOPS TOOLS & CLOUD COMPUTING*

João Victor Michaeli - 555678  
Larissa Muniz - 557197  
Henrique Garcia - 558062

# Sumário

**Links**

**Azure Board**

**Dockerfile**

**Criar Vm**

**Excluir Vm**

**Diagrama**

# Links

**Link para o vídeo da  
criação da Vm**

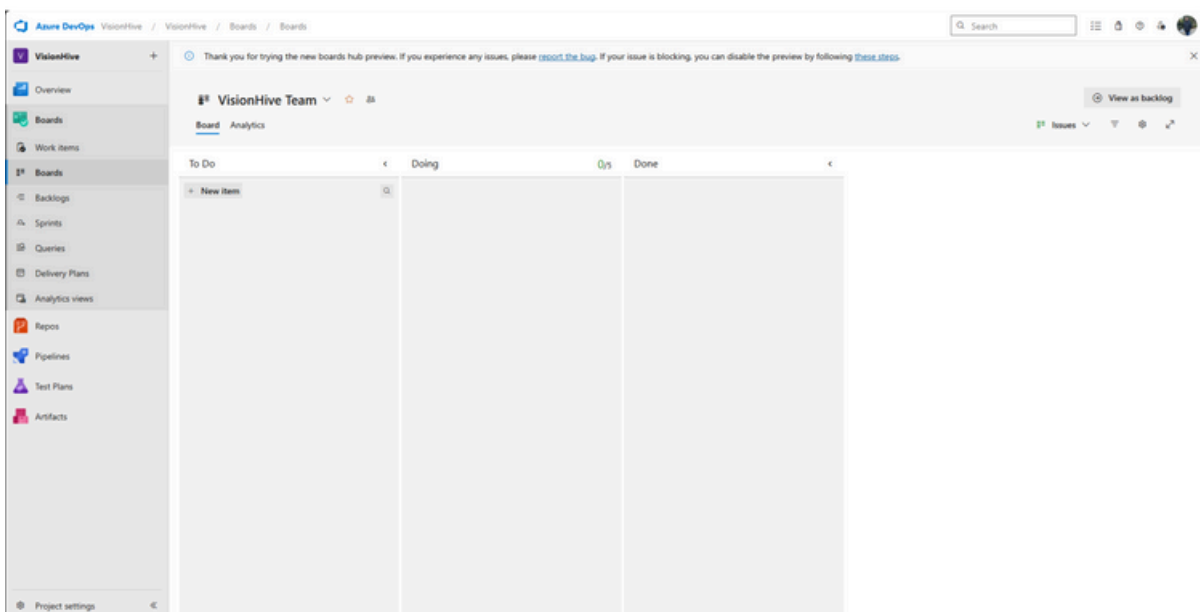
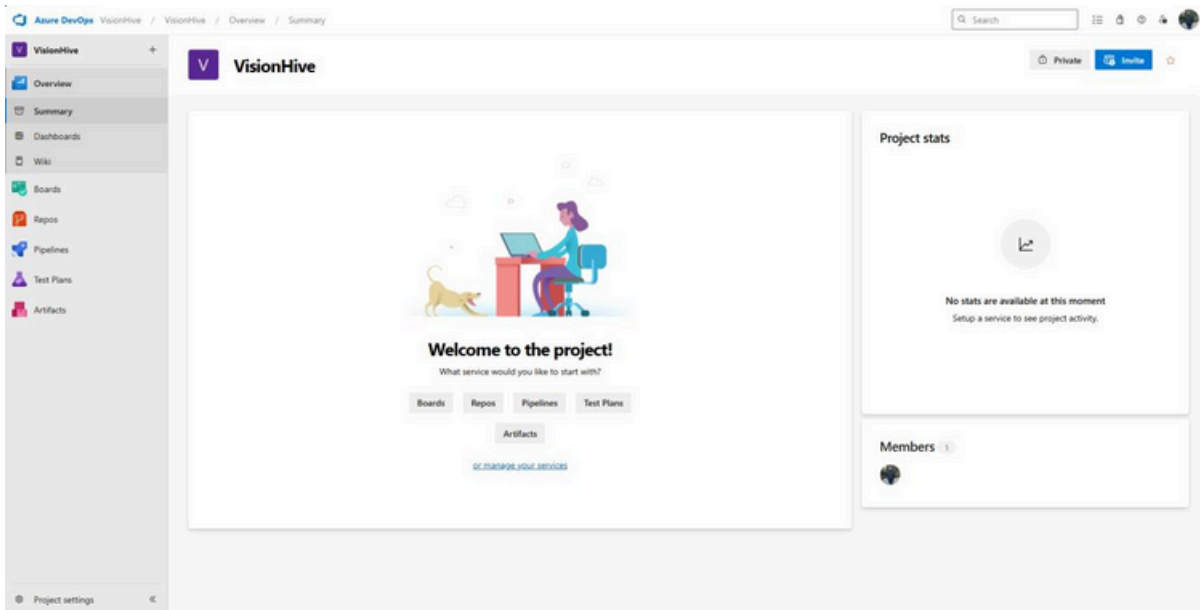
<https://youtu.be/fqSrBkhhbOAg>

**Link do repositório  
github**

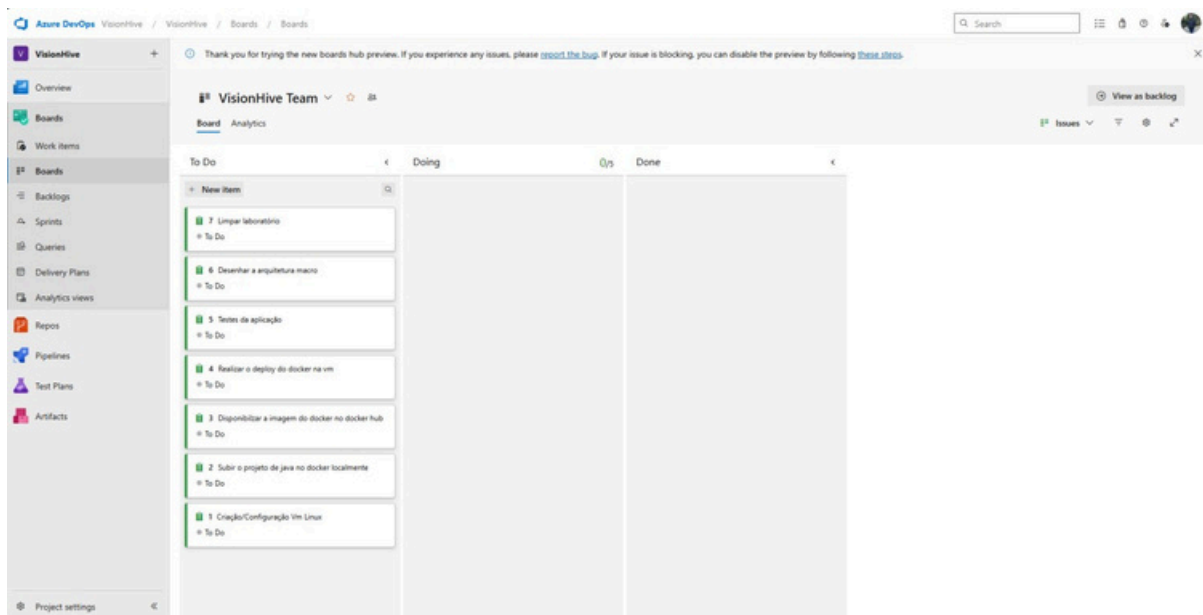
<https://github.com/JoaoMichaeli/VisionHive-DevOps>

# Azure Board

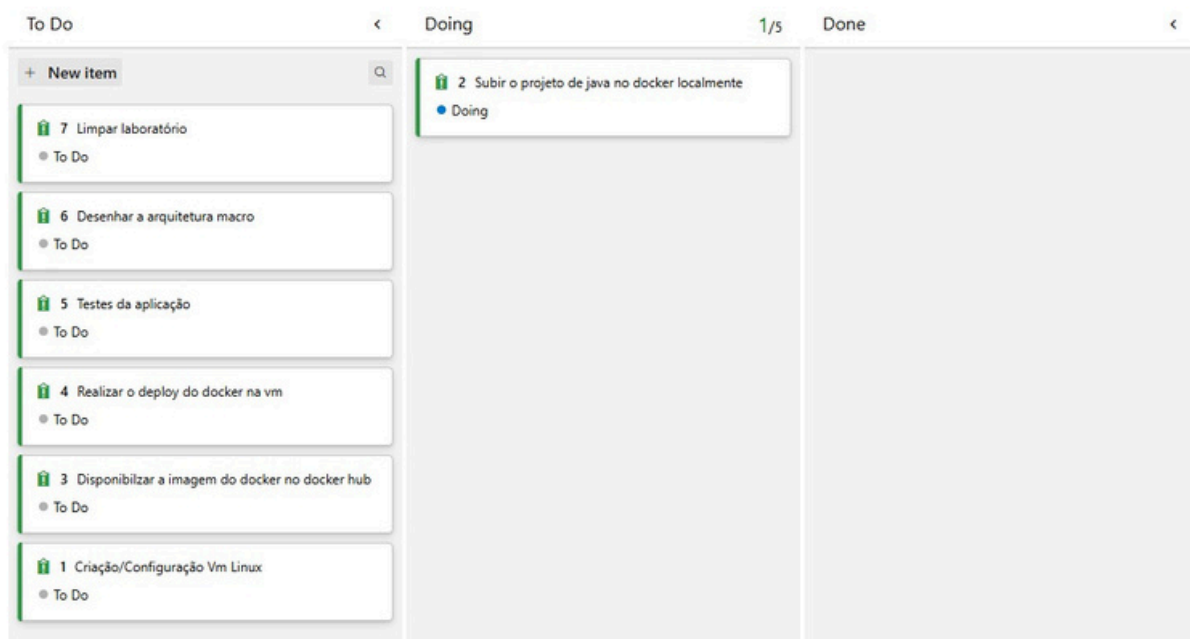
Criação do Azure board para melhor organização



## Preenchimento da lista To Do



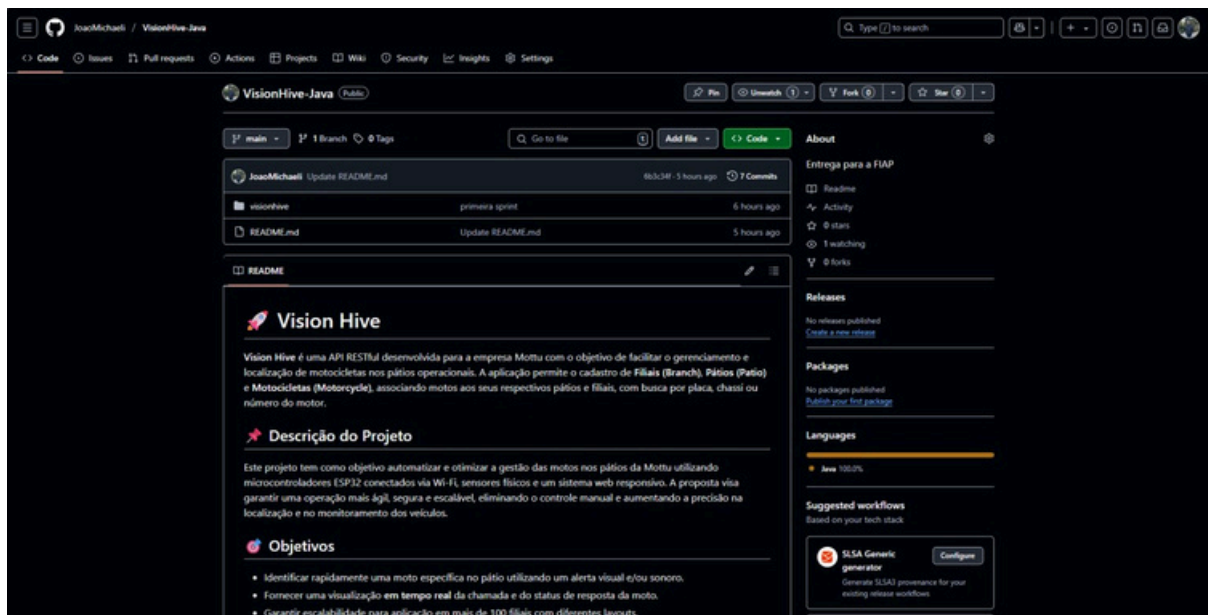
Primeiro vamos subir o java no docker local, para posteriormente subir no docker hub



# Dockerfile

O projeto java está nesse repositório do github:

<https://github.com/JoaoMichaeli/VisionHive-Java>



Dockerfile do projeto

```
Dockerfile X
C: > Users > joao > Desktop > VisionHive-DevOps > visionhive > Dockerfile
1 FROM maven:3.8.5-openjdk-17 AS build
2 WORKDIR /app
3 COPY . .
4 RUN mvn clean package -DskipTests
5
6
7 FROM openjdk:17-alpine
8 WORKDIR /app
9 COPY --from=build /app/target/*.jar app.jar
10 EXPOSE 8080
11 ENTRYPOINT ["java", "-jar", "app.jar"]
```

## Criar e testar a imagem Docker localmente

```
Terminal
PS C:\Users\joaov\Desktop\VisionHive-DevOps\visionhive> docker build -t joamichaeli/visionhive-java:v1 .

[+] Building 17.9s (15/15) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 280B
=> [internal] load metadata for docker.io/library/openjdk:17-alpine
=> [internal] load metadata for docker.io/library/maven:3.8.5-openjdk-17
=> [auth] library/openjdk:pull token for registry-1.docker.io
=> [auth] library/maven:pull token for registry-1.docker.io
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [build 1/4] FROM docker.io/library/maven:3.8.5-openjdk-17@sha256:3a9c30b3af6278a8ae0007d3a3bf00fff80ec3ed
=> => resolve docker.io/library/maven:3.8.5-openjdk-17@sha256:3a9c30b3af6278a8ae0007d3a3bf00fff80ec3ed7ae4eb
=> [stage-1 1/3] FROM docker.io/library/openjdk:17-alpine@sha256:4b6abae565492dbe9e7a894137c966a748515423890
=> => resolve docker.io/library/openjdk:17-alpine@sha256:4b6abae565492dbe9e7a894137c966a7485154238902f2f25e9
=> [internal] load build context
=> => transferring context: 55.80kB
=> CACHED [stage-1 2/3] WORKDIR /app
=> CACHED [build 2/4] WORKDIR /app
=> [build 3/4] COPY .
=> [build 4/4] RUN mvn clean package -DskipTests
=> [stage-1 3/3] COPY --from=build /app/target/*.jar app.jar

=> exporting to image
=> => exporting layers
=> => exporting manifest sha256:2ad7e5b3c8569518b474b305fe4e0a3d12d16f1340bde32130dcd46436c40d52
=> => exporting config sha256:1ec13dd6250f14552f903adee4dcce834d6eadbb4bd7c12791d456e820d53573
=> => exporting attestation manifest sha256:7f48ffd913268aacc79806528b0e69965cf88f3aef78d9be89020c7aff1c1ec5
=> => exporting manifest list sha256:9a82dd24a38e6529b03d0de549390be0e6476fe229cda4c4e09db14e92b7434c
=> => naming to docker.io/joamichaeli/visionhive-java:v1
=> => unpacking to docker.io/joamichaeli/visionhive-java:v1

View build details: docker-desktop://dashboard/build/desktop.linux/desktop.linux/y9kb6nmvfzznbdh1hqw59eta1
PS C:\Users\joaov\Desktop\VisionHive-DevOps\visionhive>
```

## Imagem criada

```
PS C:\Users\joaov\Desktop\VisionHive-DevOps\visionhive> docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
joamichaeli/visionhive-java	v1	9a82dd24a38e	About a minute ago	629MB

```
PS C:\Users\joaov\Desktop\VisionHive-DevOps\visionhive>
```

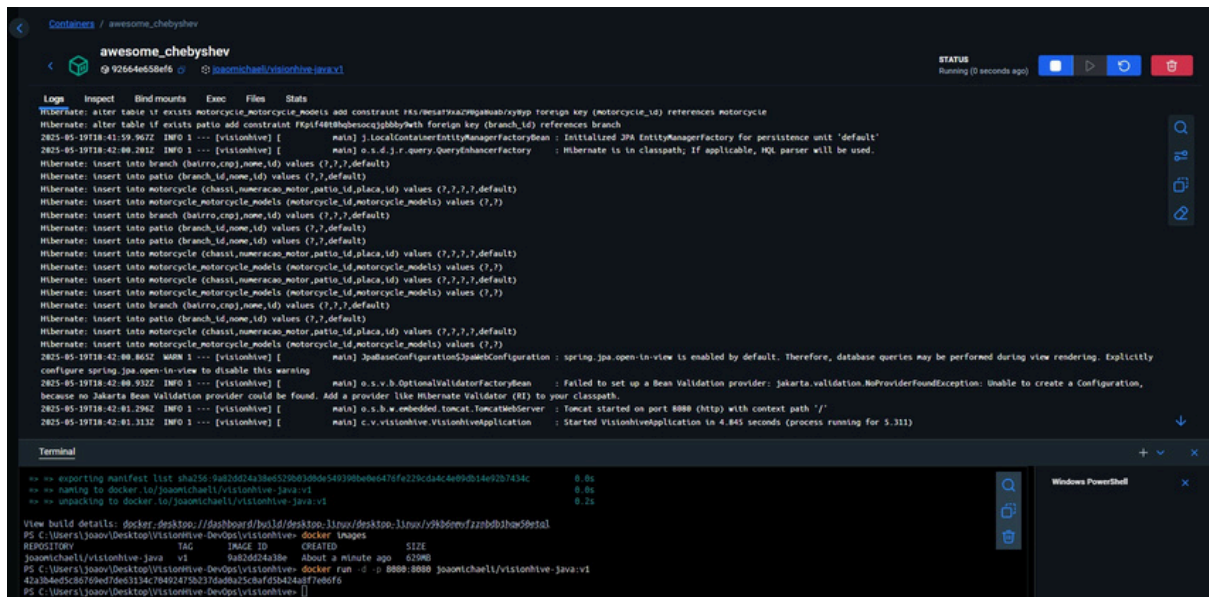
**Images** [Give feedback](#)

View and manage your local and Docker Hub images. [Learn more](#)

Search

Name	Tag	Image ID	Created	Size	Actions
joamichaeli/visionhive-java	v1	9a82dd24a38e	3 seconds ago	629.24 MB	

Como podemos ver, o projeto já está funcionando localmente



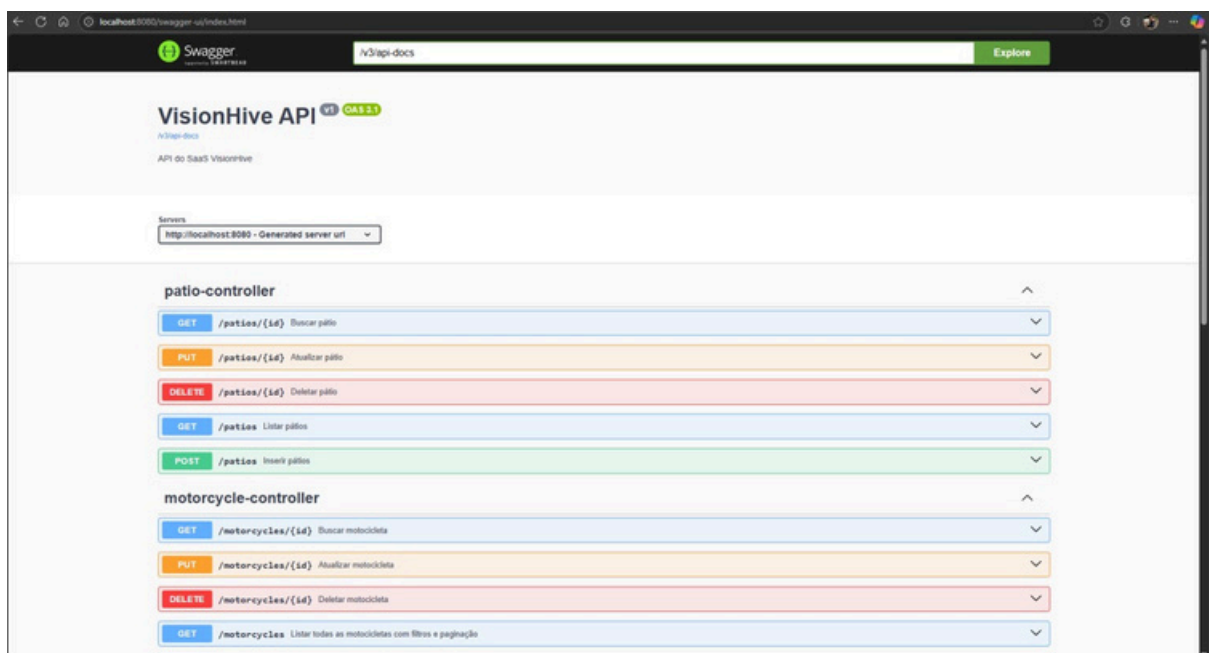
```
Container awesome_chebyshev
awesome_chebyshev
@ 92664ed58ef6  java:1
STATUS
Running (0 seconds ago)

Logs Inspect Bind mounts Exec Files Stats
Hibernate: alter table if exists motorcycle_motorcycle_models add constraint FKq1f48t8h0qocjpbby7wh foreign key (branch_id) references branch
2025-05-19T18:42:59.962Z INFO 1 --- [visionhive] [ main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit 'default'
2025-05-19T18:42:59.281Z INFO 1 --- [visionhive] [ main] o.s.d.j.r.query.QueryHintsFactory : Hibernate is in classpath; if applicable, HQ parser will be used.
Hibernate: insert into branch (bairro,cnpj,nome,id) values (7,7,7,default)
Hibernate: insert into patio (branch_id,nome,id) values (7,7,default)
Hibernate: insert into motorcycle (chassi,numeracao_motor_patio_id,placa,id) values (7,7,7,7,default)
Hibernate: insert into motorcycle_motorcycle_models (motorcycle_id,motorcycle_models) values (7,7)
Hibernate: insert into branch (bairro,cnpj,nome,id) values (7,7,7,default)
Hibernate: insert into patio (branch_id,nome,id) values (7,7,default)
Hibernate: insert into motorcycle (chassi,numeracao_motor_patio_id,placa,id) values (7,7,7,7,default)
Hibernate: insert into motorcycle_motorcycle_models (motorcycle_id,motorcycle_models) values (7,7)
Hibernate: insert into motorcycle_motorcycle_models (motorcycle_id,motorcycle_models) values (7,7)
Hibernate: insert into branch (bairro,cnpj,nome,id) values (7,7,7,default)
Hibernate: insert into patio (branch_id,nome,id) values (7,7,default)
Hibernate: insert into motorcycle (chassi,numeracao_motor_patio_id,placa,id) values (7,7,7,7,default)
Hibernate: insert into motorcycle_motorcycle_models (motorcycle_id,motorcycle_models) values (7,7)
2025-05-19T18:42:59.865Z INFO 1 --- [visionhive] [ main] 3pabaseConfiguration5pabaseConfiguration : spring.jpa.open-in-view is enabled by default. Therefore, database queries may be performed during view rendering. Explicitly configure spring.jpa.open-in-view to disable this warning
2025-05-19T18:42:59.932Z INFO 1 --- [visionhive] [ main] o.s.v.b.OptionalValidatorFactoryBean : failed to set up a Bean Validation provider: jakarta.validation.spi.ProviderException: Unable to create a Configuration, because no Jakarta Bean Validation provider could be found. Add a provider like Hibernate Validator (RI) to your classpath.
2025-05-19T18:42:59.294Z INFO 1 --- [visionhive] [ main] o.s.w.embedded.TomcatWebServer : Tomcat started on port 8080 (http) with context path '/'
2025-05-19T18:42:59.312Z INFO 1 --- [visionhive] [ main] c.v.visionhive.VisionhiveApplication : Started VisionhiveApplication in 4.845 seconds (process running for 5.312)

Terminal
==> exporting Manifest list sha256:9a826d24a38e579b03d9d5493909eb647ef275cd4c4e9db14e2b7434c 0.0s
==> naming to docker.io/joamichael/visionhive-java-v1 0.0s
==> unpacking to docker.io/joamichael/visionhive-java-v1 0.2s

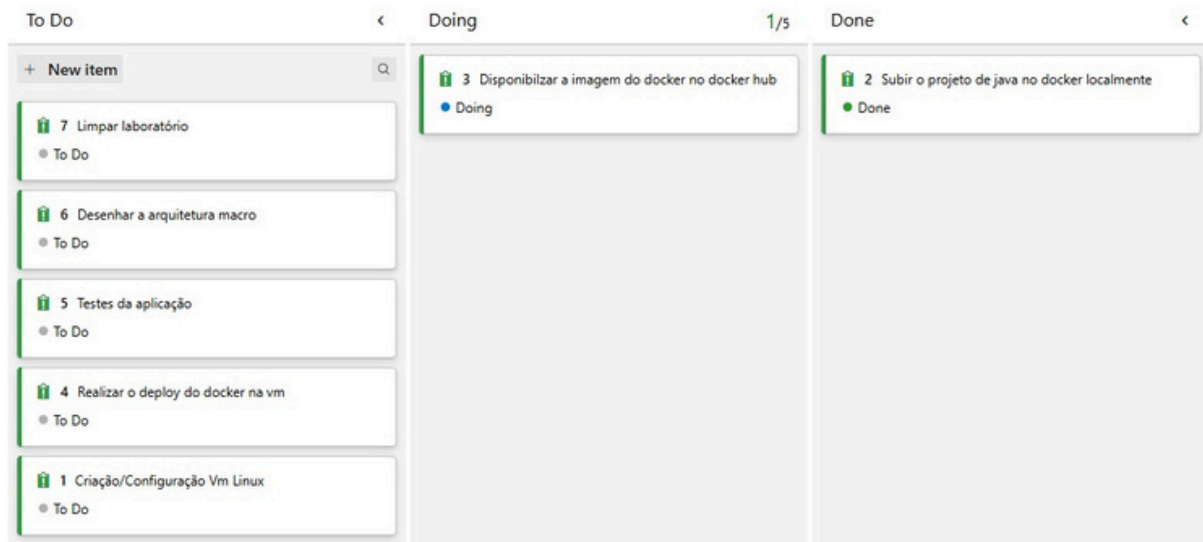
View build details: docker desktop://dashboards/build/desktop.linux/desktop.linux/v9b6revfzrbd3hwe9beta1
PS C:\Users\joao\Desktop\Visionhive> docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
joamichael/visionhive-java v1 9a826d24a38e About a minute ago 620MB
PS C:\Users\joao\Desktop\Visionhive> docker run -d -p 8080:8080 joamichael/visionhive-java:v1
42a3b4e5c8679ed7d6a3134c76492475b237da0a25c4fd5b424a8f7e0f6
PS C:\Users\joao\Desktop\Visionhive> docker ps
PS C:\Users\joao\Desktop\Visionhive> docker logs -f 42a3b4e5c8679ed7d6a3134c76492475b237da0a25c4fd5b424a8f7e0f6
```

Como essa é a primeira entrega da solução em java, ainda não temos integração web gráfica, então para testarmos o deploy será utilizado via swagger





Após o sucesso do teste, vou subir a imagem no docker hub para o envio posterior a vm.



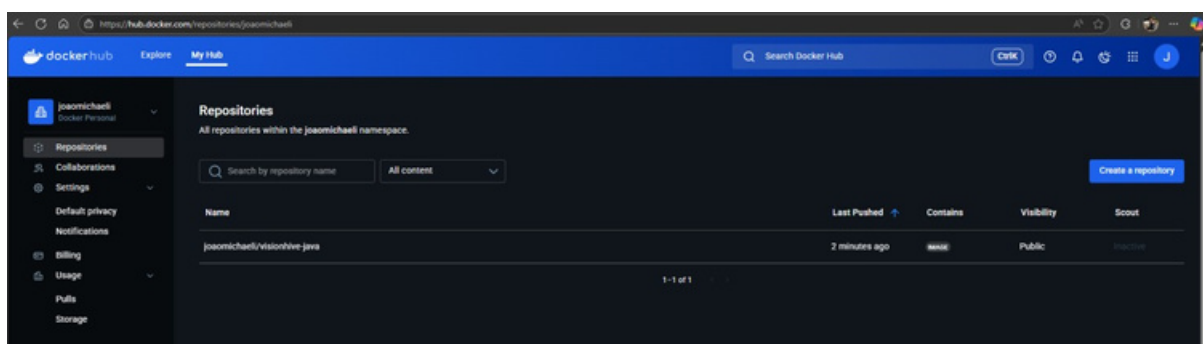
Agora vamos realizar a disponibilidade da imagem local para o hub

```
PS C:\Users\joaov\Desktop\VisionHive-DevOps\visionhive> docker login
Authenticating with existing credentials... [Username: joaomichaeli]

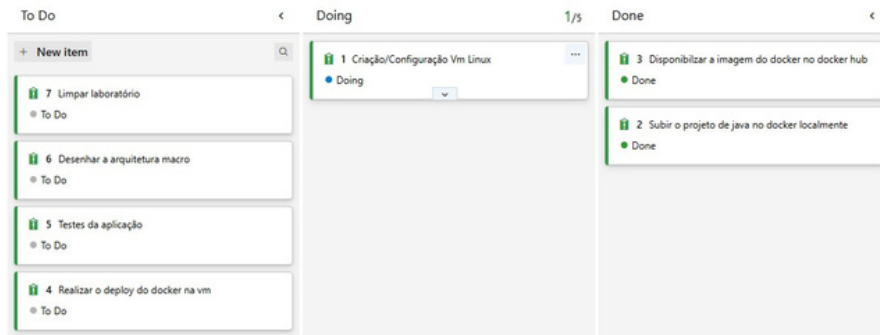
Info → To login with a different account, run 'docker logout' followed by 'docker login'

Login Succeeded
PS C:\Users\joaov\Desktop\VisionHive-DevOps\visionhive> docker push joaomichaeli/visionhive-java:v1
The push refers to repository [docker.io/joaomichaeli/visionhive-java]
c3b66361bb95: Pushed
7992fe636f76: Pushed
d8d715783b80: Pushed
53c9466125e4: Pushed
5843afab3874: Pushed
d902d246dc1f: Pushed
v1: digest: sha256:9a82dd24a38e6529b03d0de549390be0e6476fe229cda4c4e09db14e92b7434c size: 856
PS C:\Users\joaov\Desktop\VisionHive-DevOps\visionhive> 
```

Imagem postada no hub

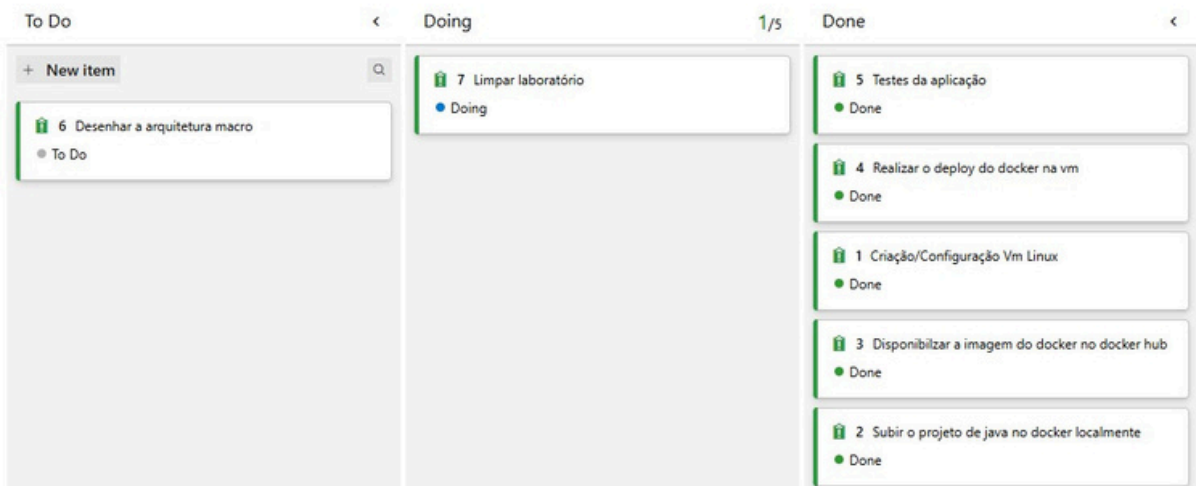


# Criar a maquina virtual



Com a imagem postada no docker hub, agora vamos criar e configurar a máquina virtual

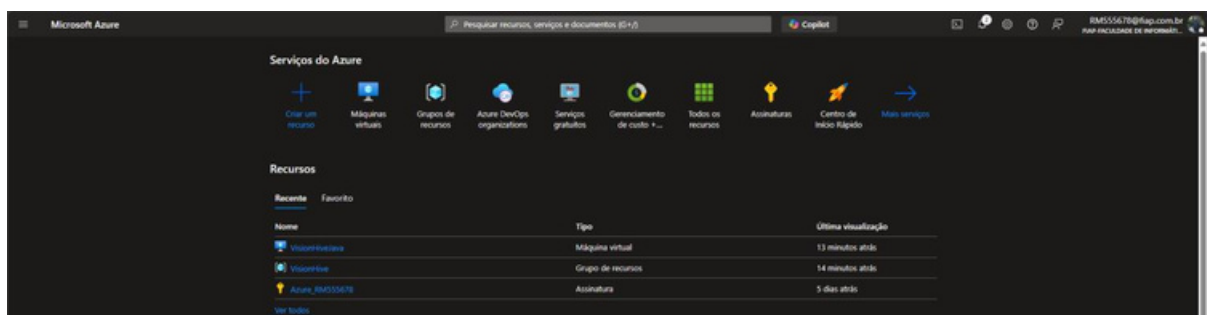
<https://youtu.be/fqSrBkhhbOAg>



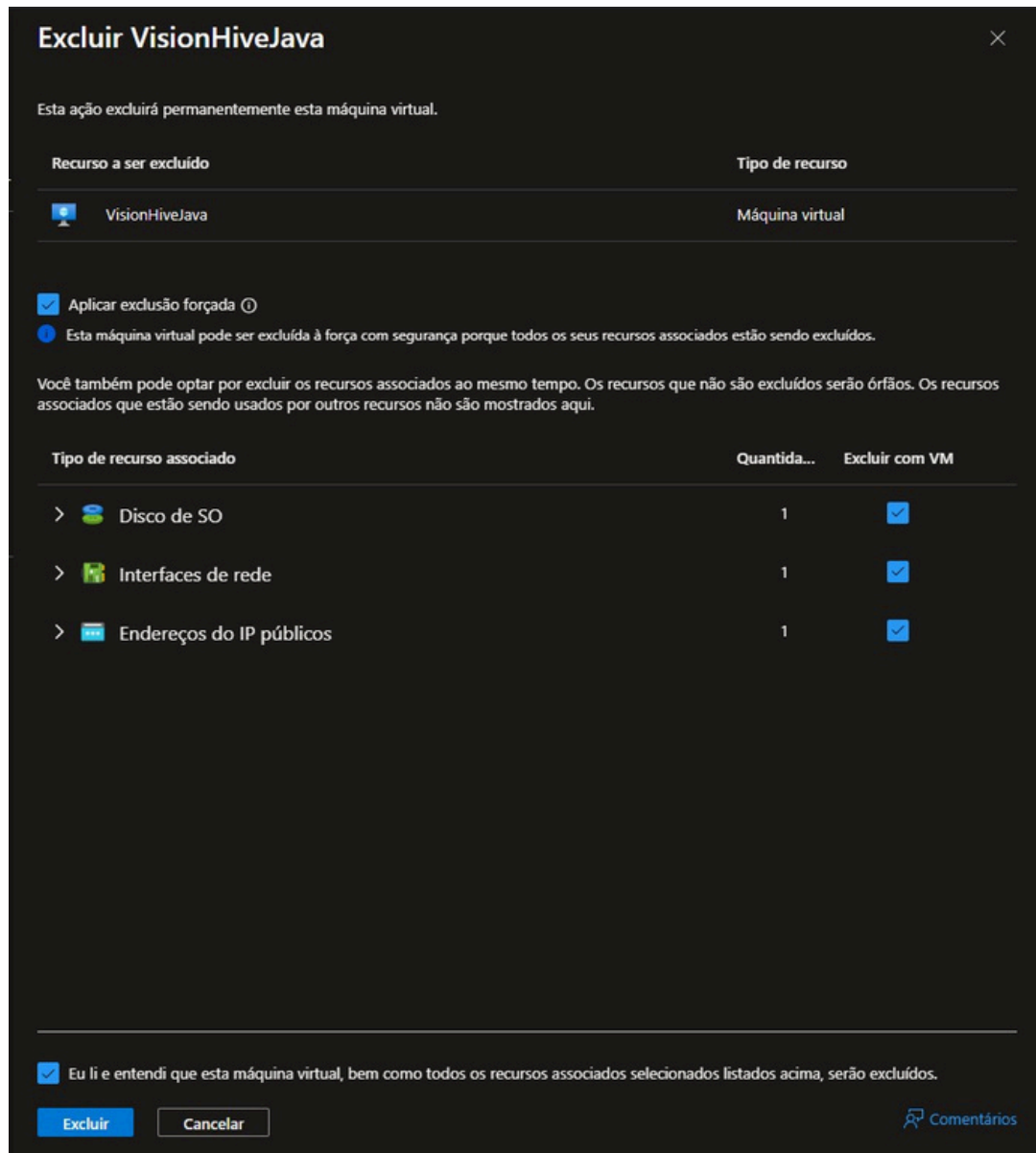
Agora é a limpeza do laboratório

Primeiro vamos limpar o Azure

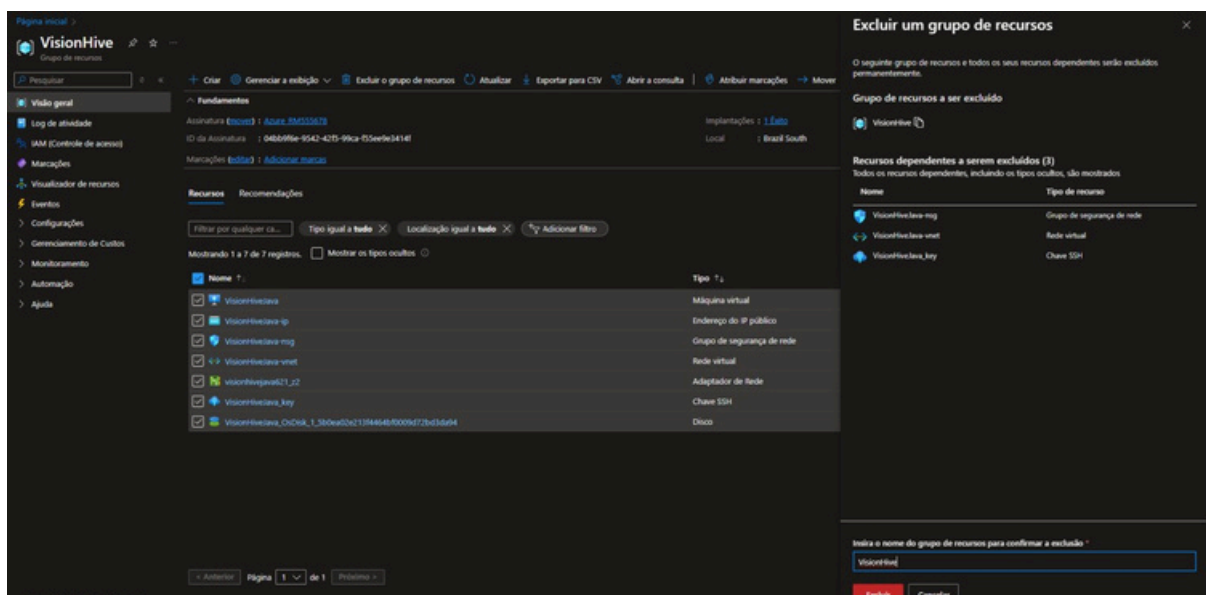
Serviços criados



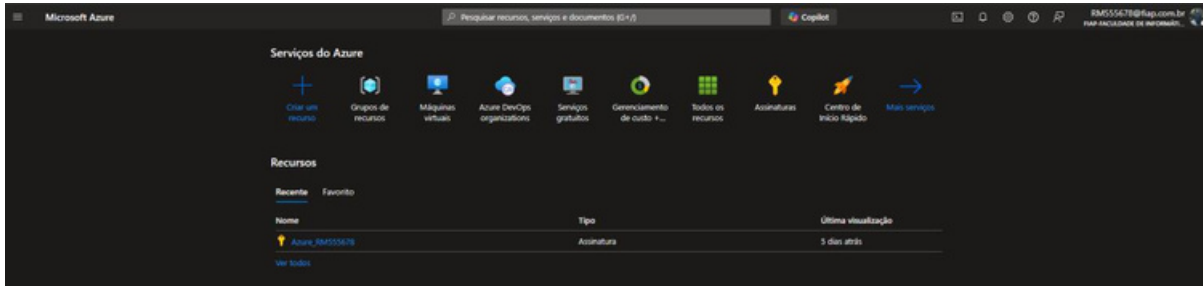
# Exclur Vm



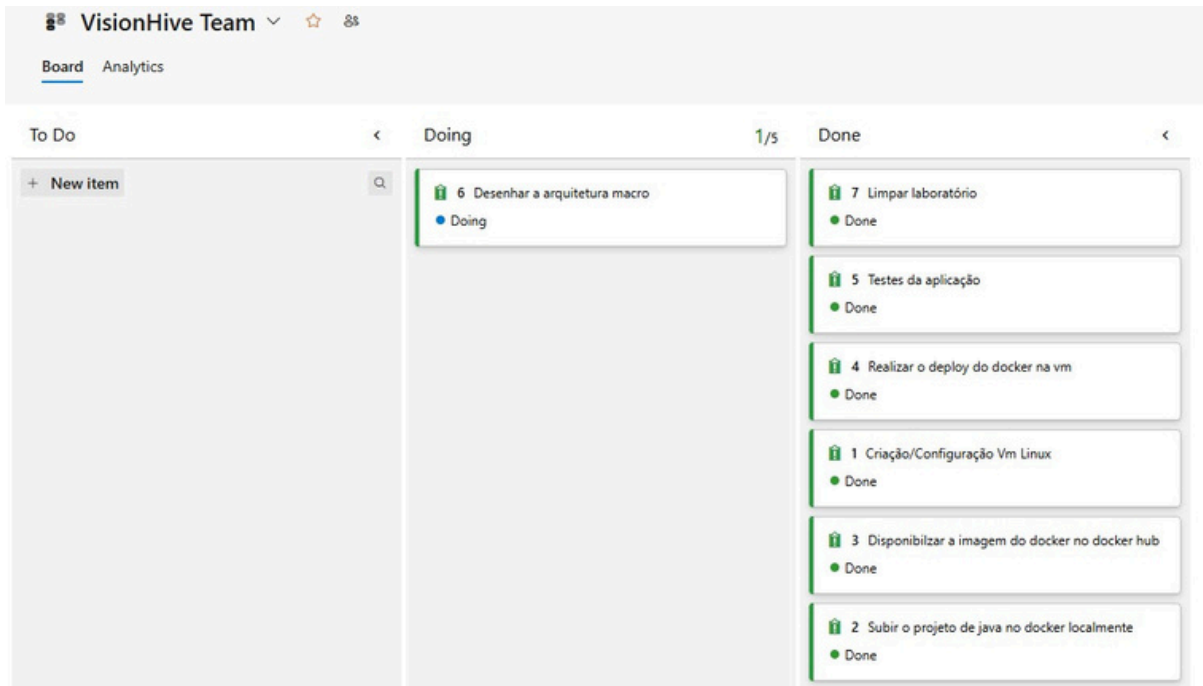
### Excluindo grupo de recurso



## Serviços limpos



Agora vamos fazer o diagrama da solução



## Diagrama

