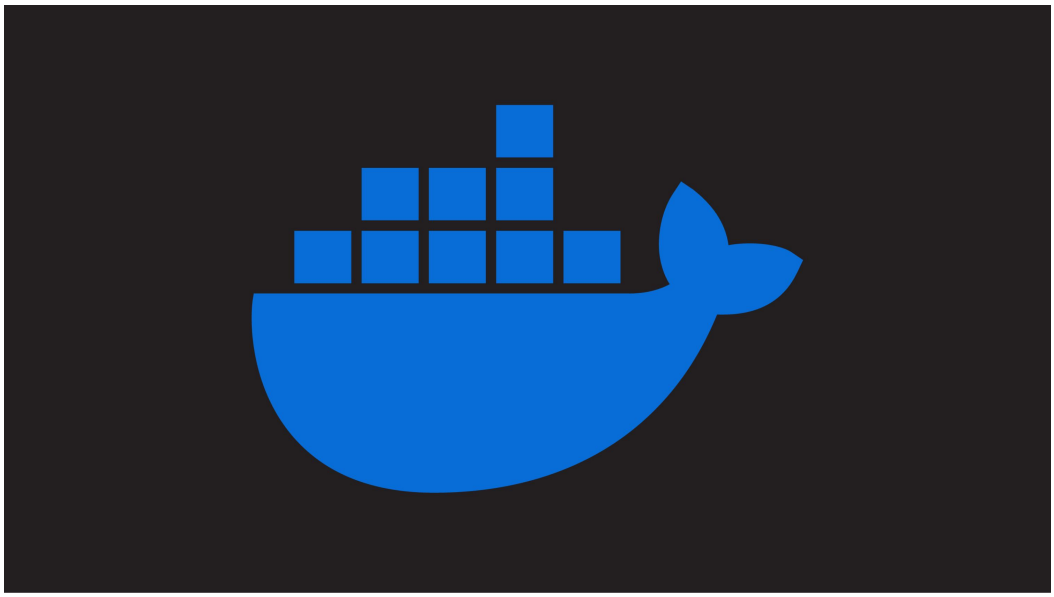
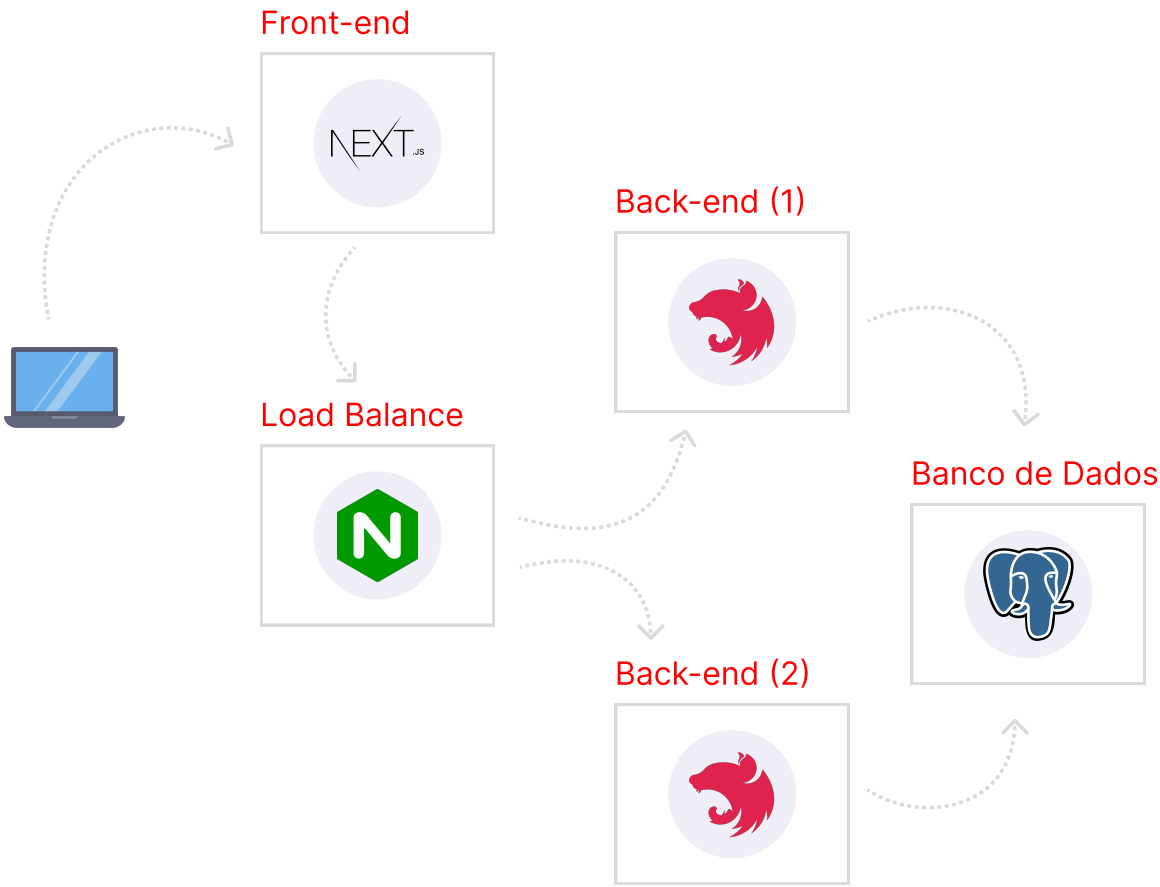
ARGUS ACADEMY

Docker

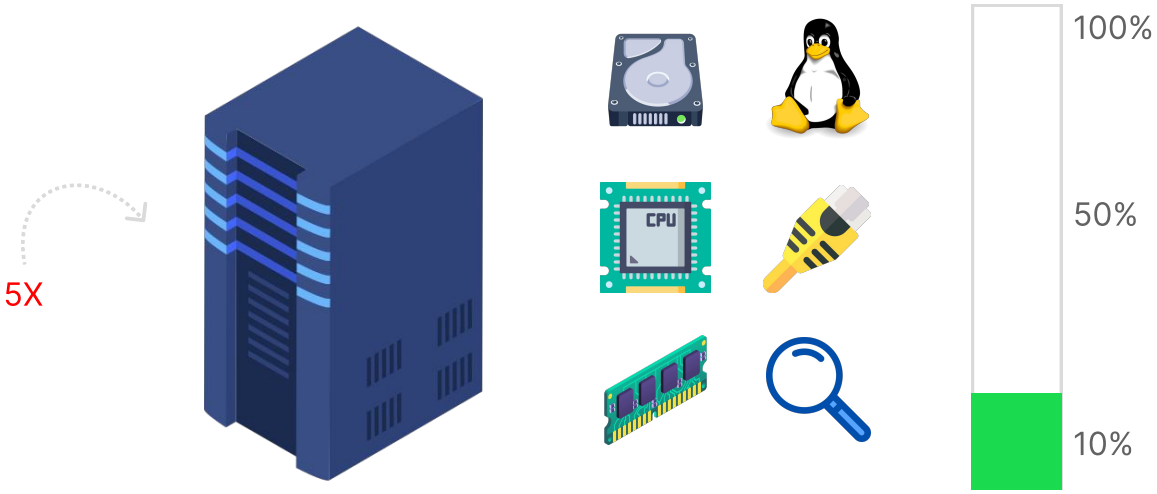
Dominando a Criação e o Gerenciamento de Containe



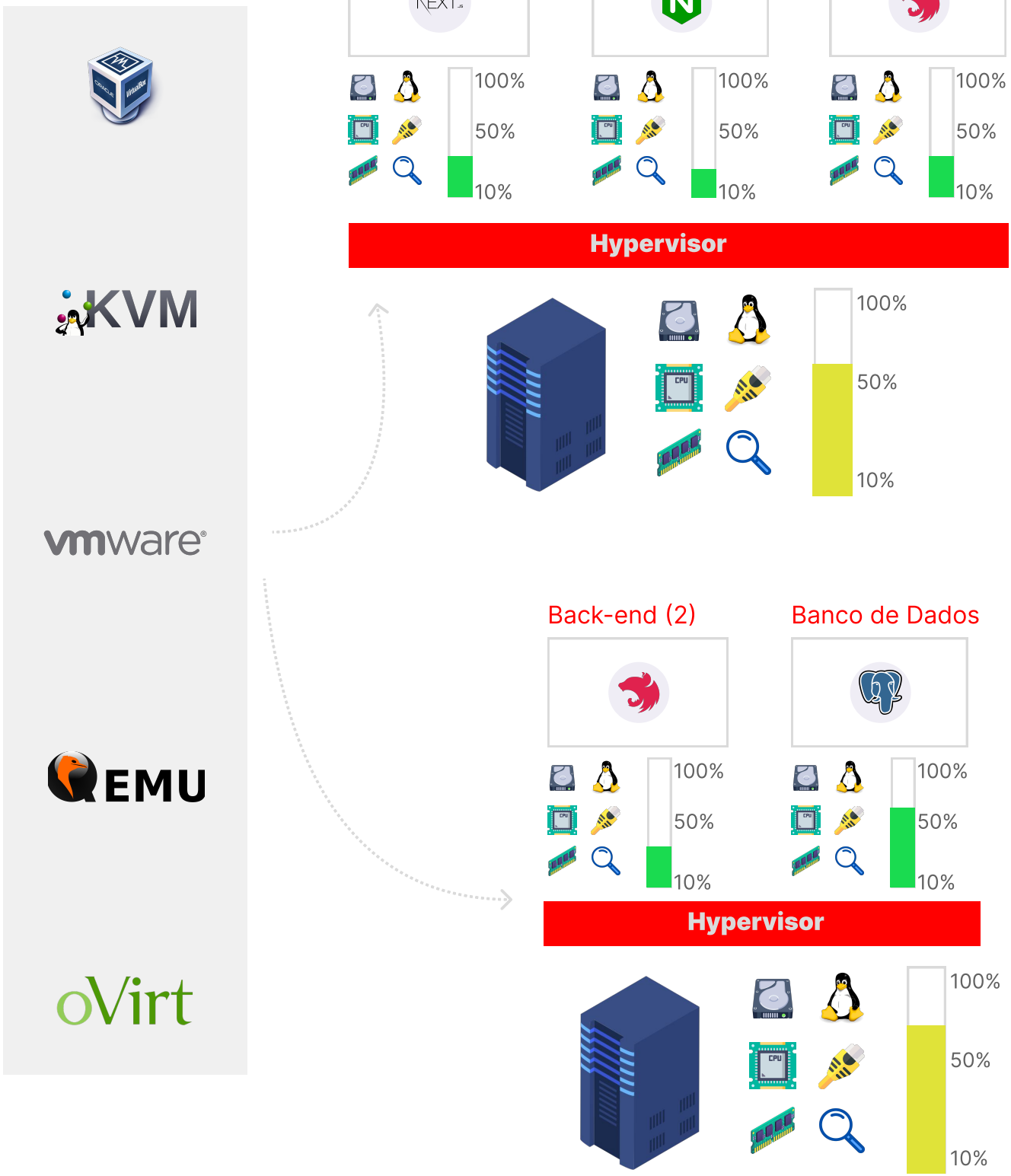
Aplicações vs Arquitetura de Infraestrutura



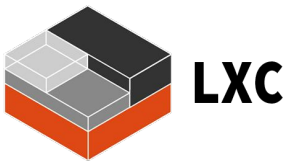
Aplicações vs Servidores Dedicados



Aplicações vs Máquinas Virtuais

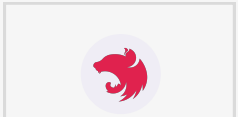
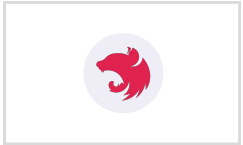
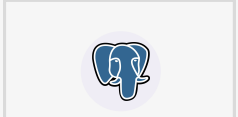


Containers são exclusividade do Docker?

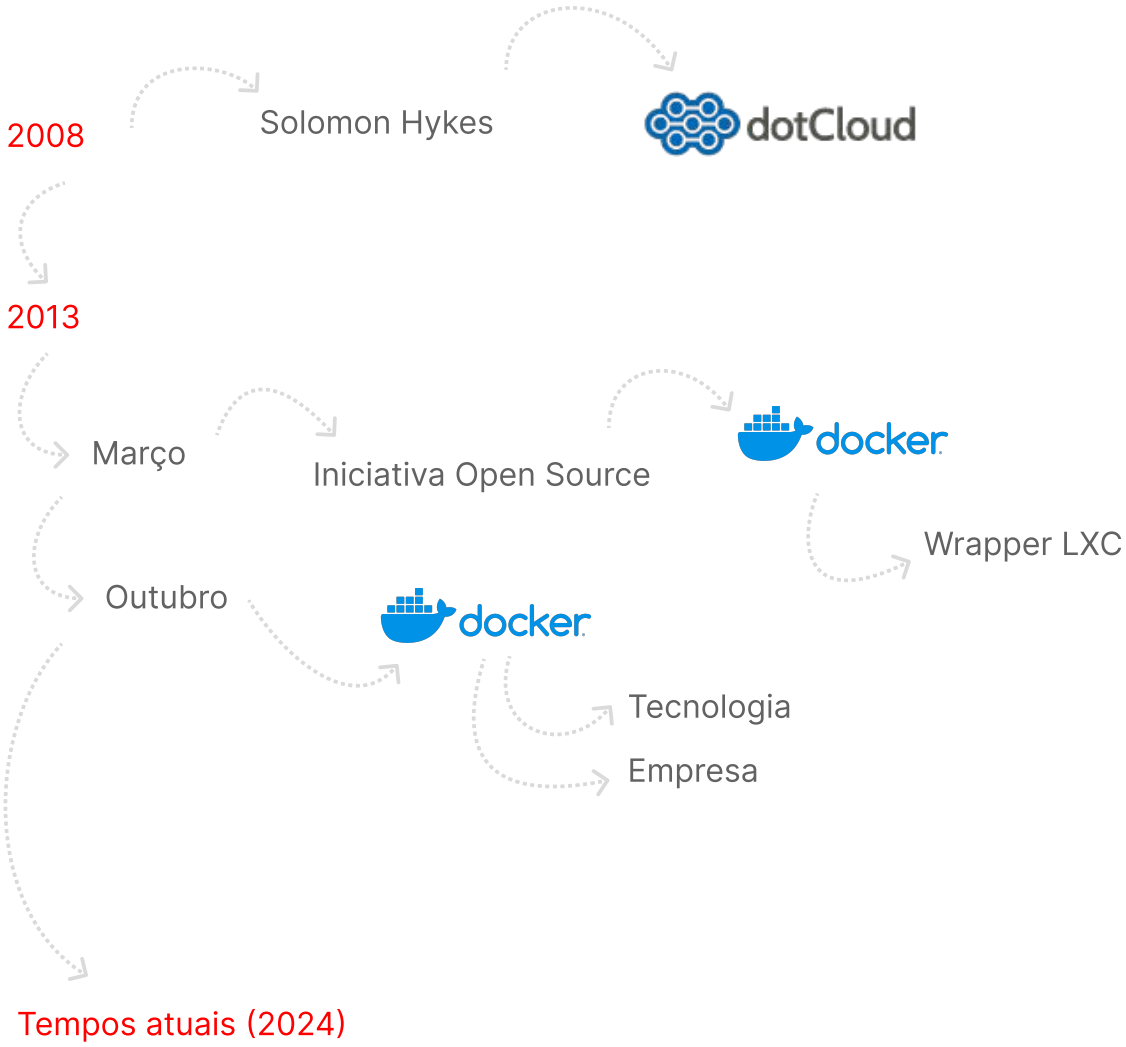


Aplicações vs Containers (produtividade)

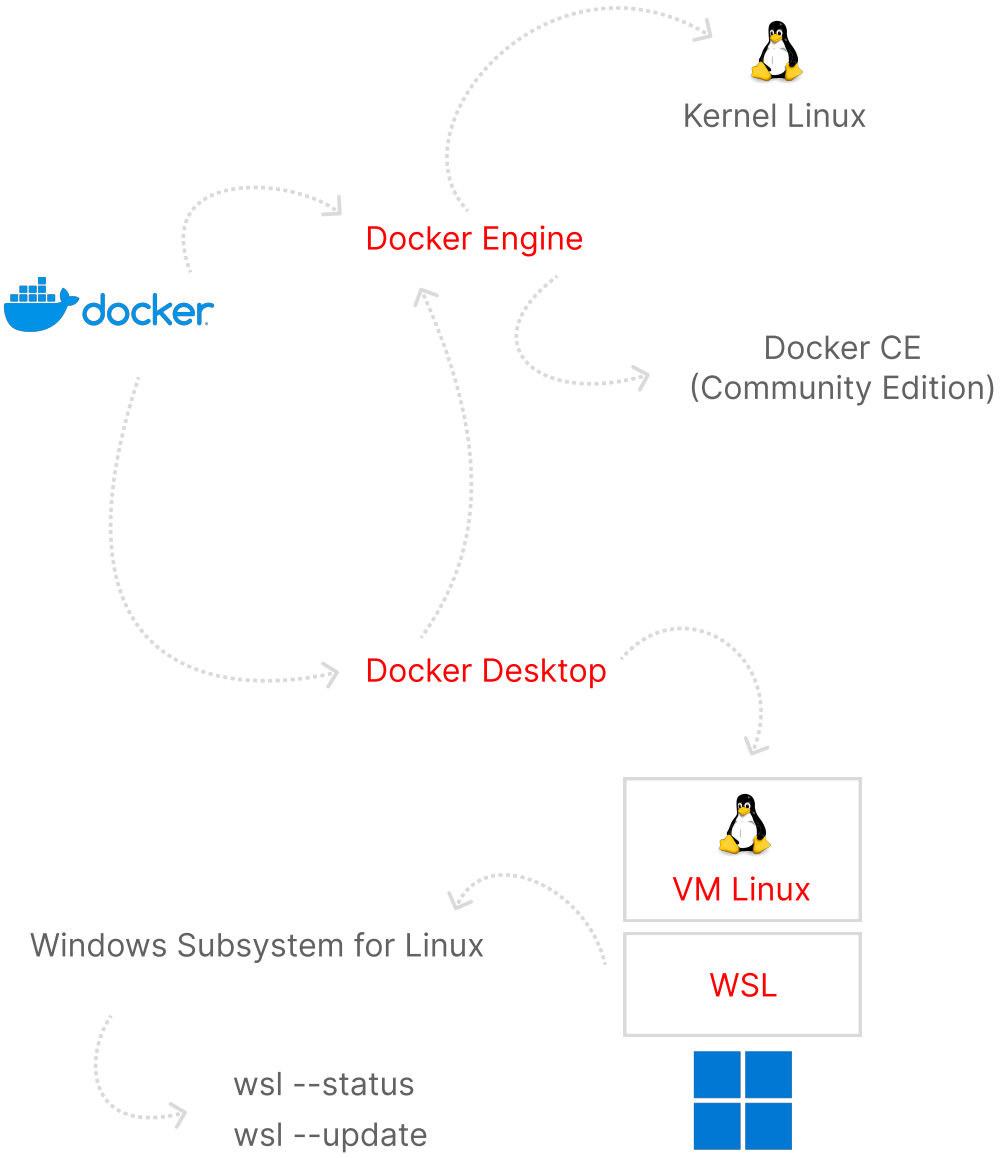
Matrix from Hell

	Devs	HML	PRD
Front-end 	?	?	?
Load Balance 	?	?	?
Back-end (1) 	?	?	?
Back-end (2) 	?	?	?
Banco de Dados 	?	?	?

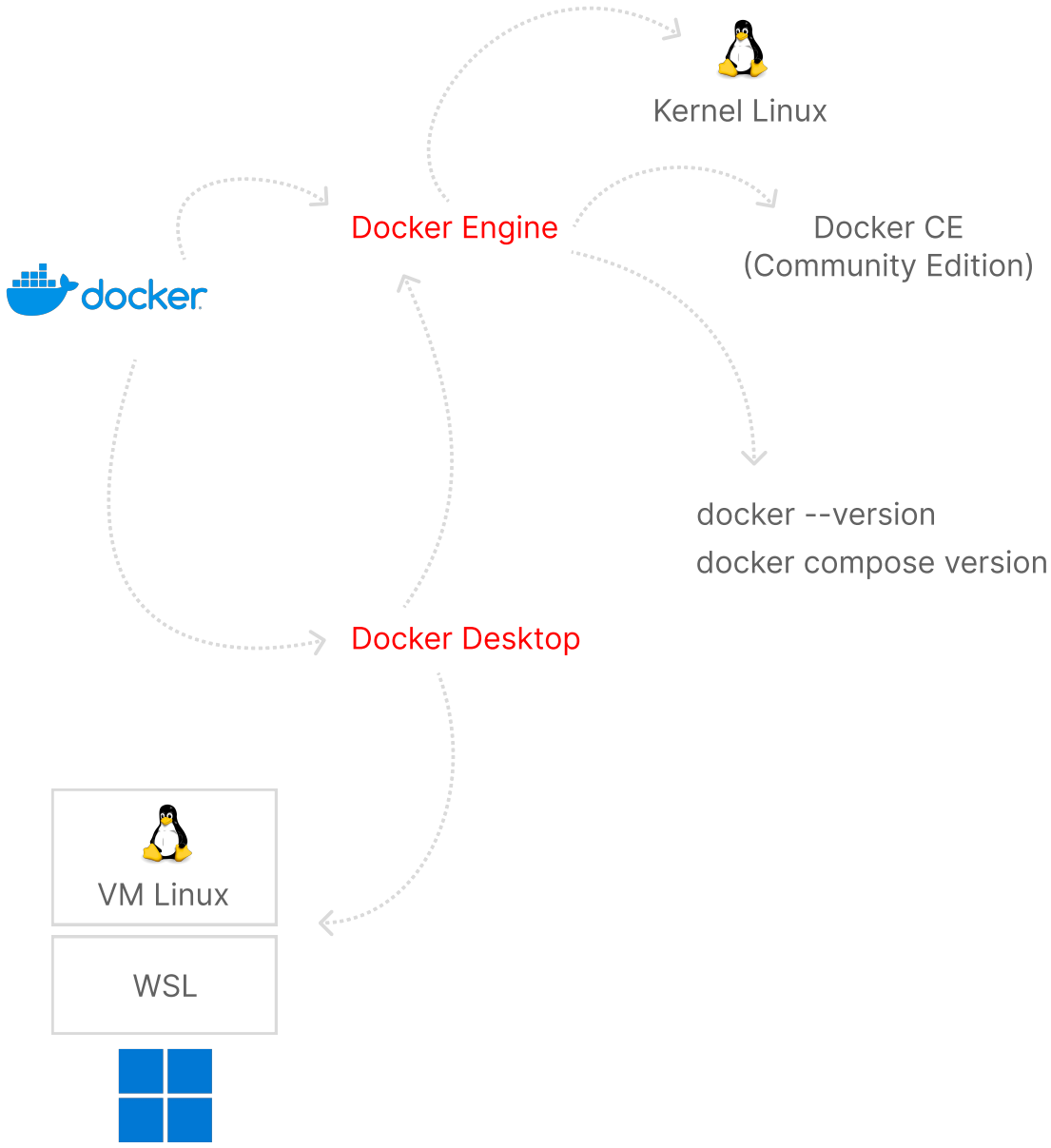
Conhecendo a empresa e a plataforma Docker



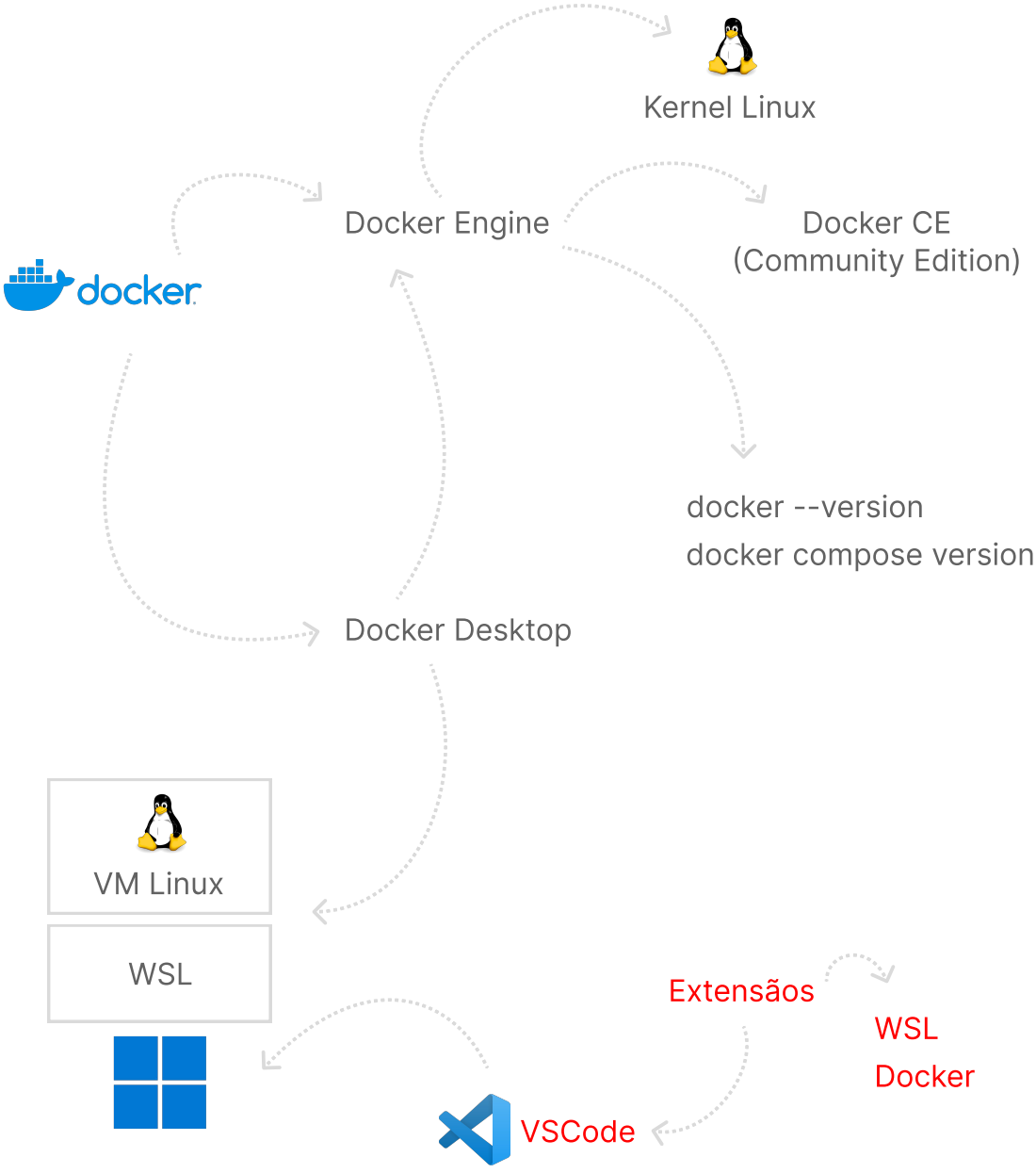
[Windows 11 Pro] Configurando o WSL (Windows Subsystem for Linux)



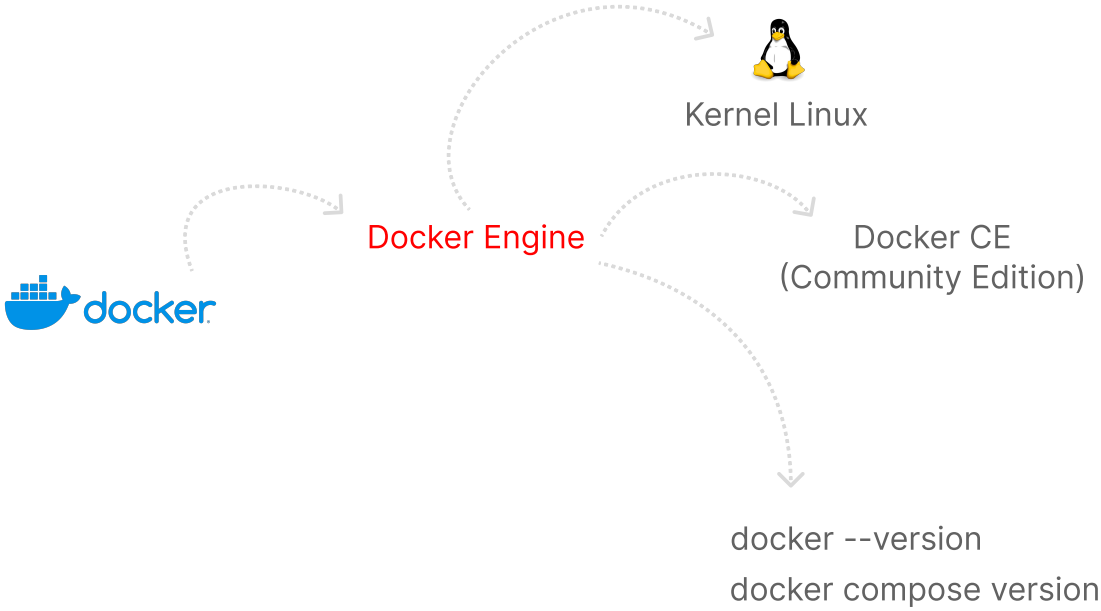
[Windows 11 Pro] Instalando o Docker Desktop



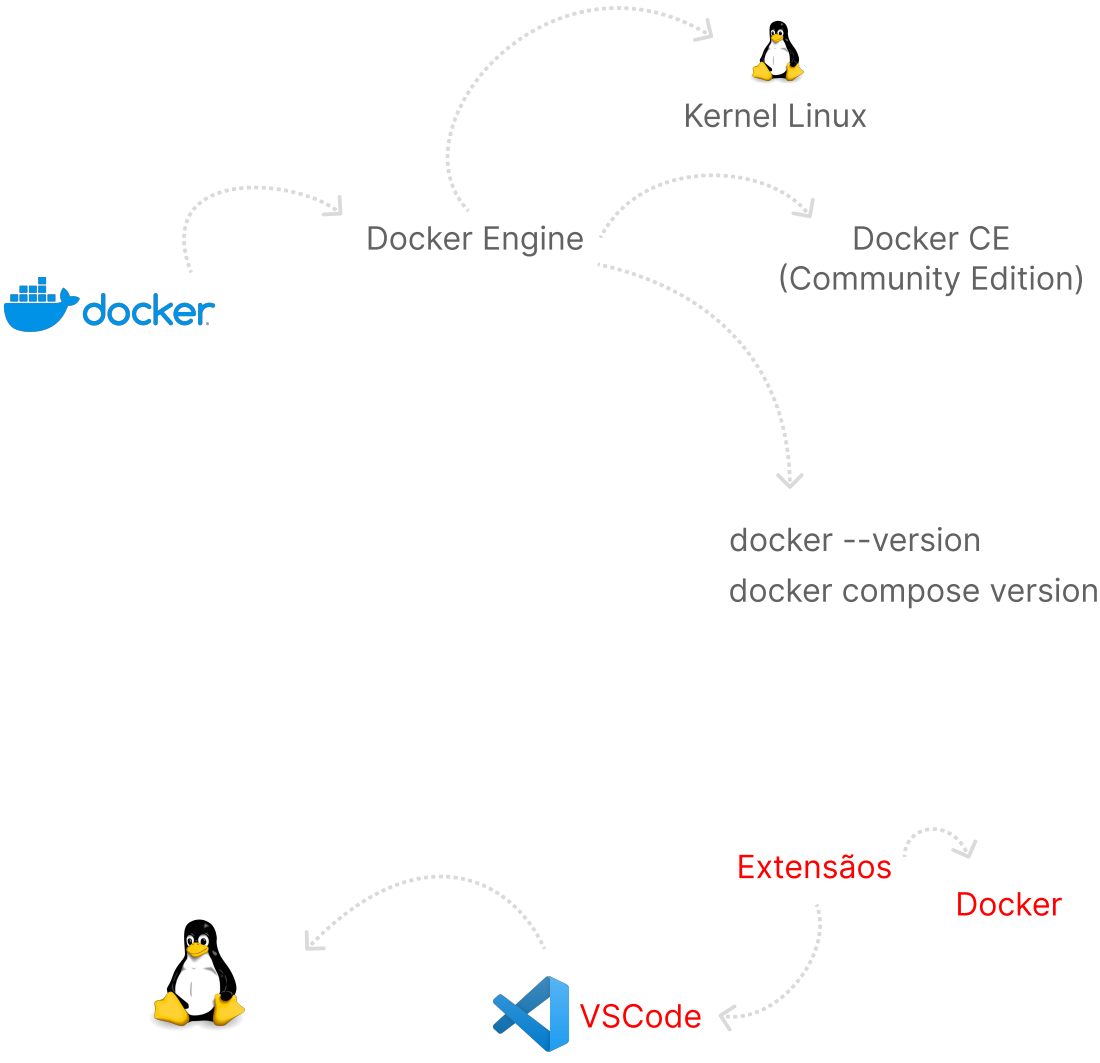
[Windows 11 Pro] Instalando o VSCode



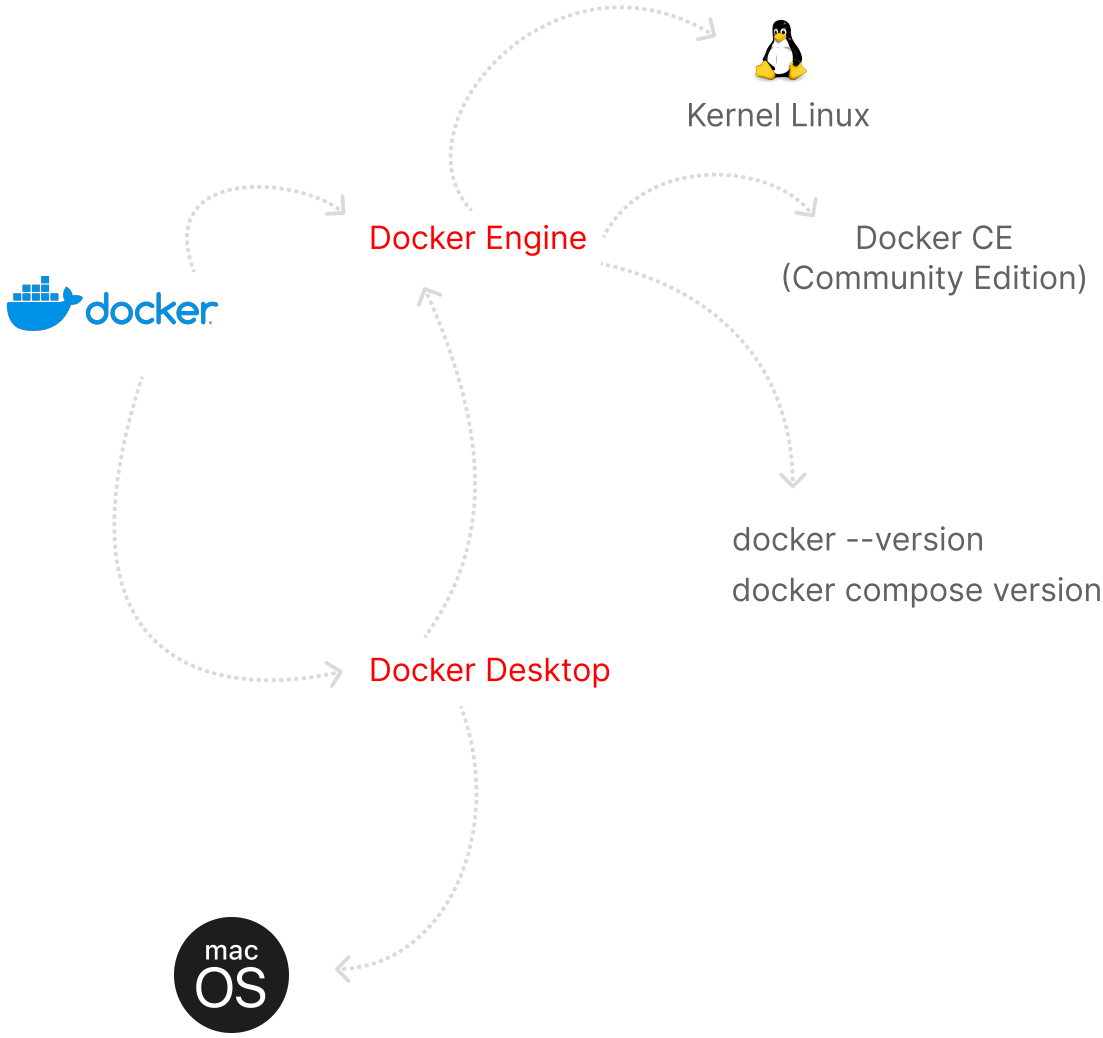
[Linux Ubuntu 22] Instalando o Docker Engine



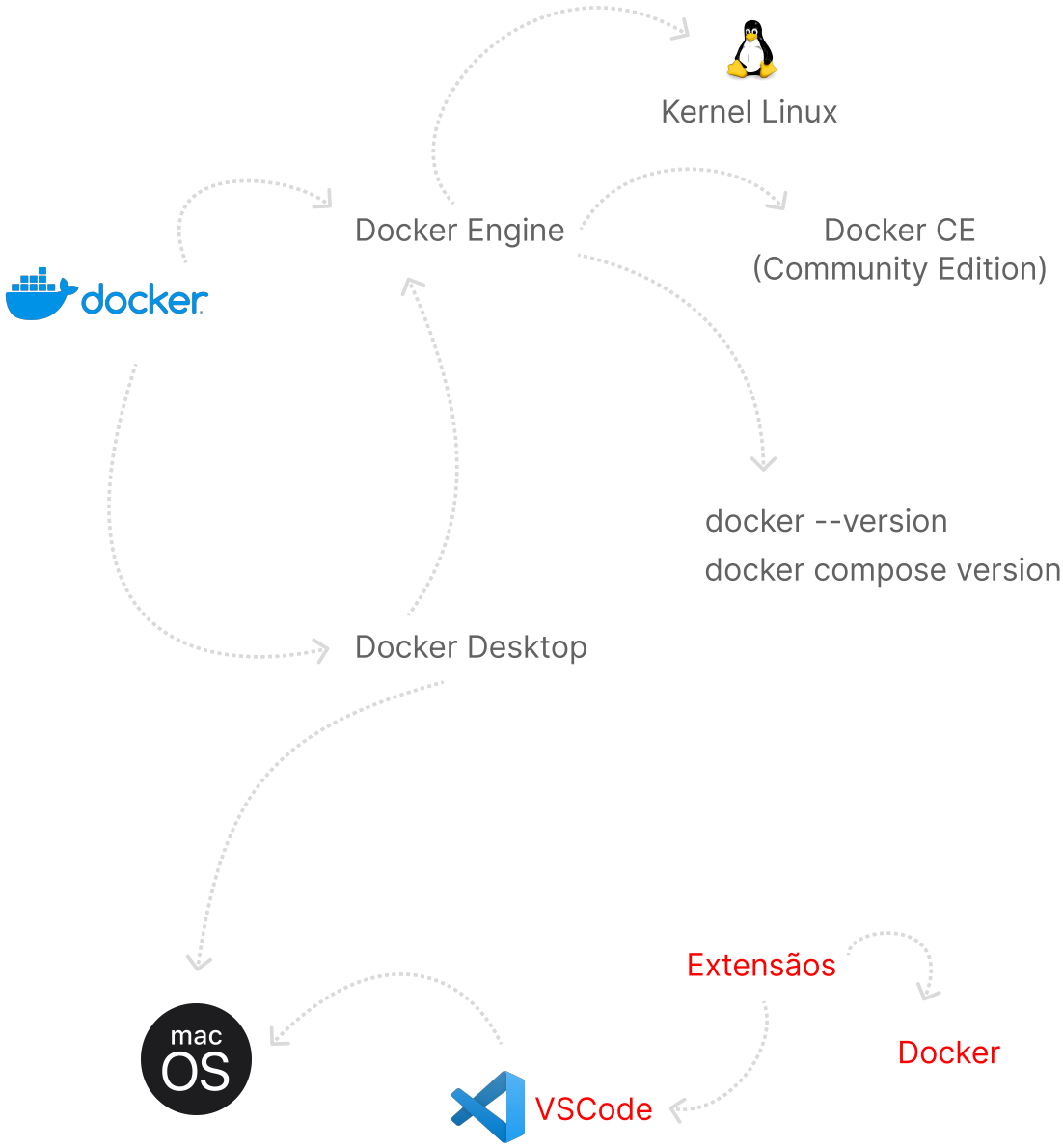
[Windows 11 Pro] Instalando o VSCode



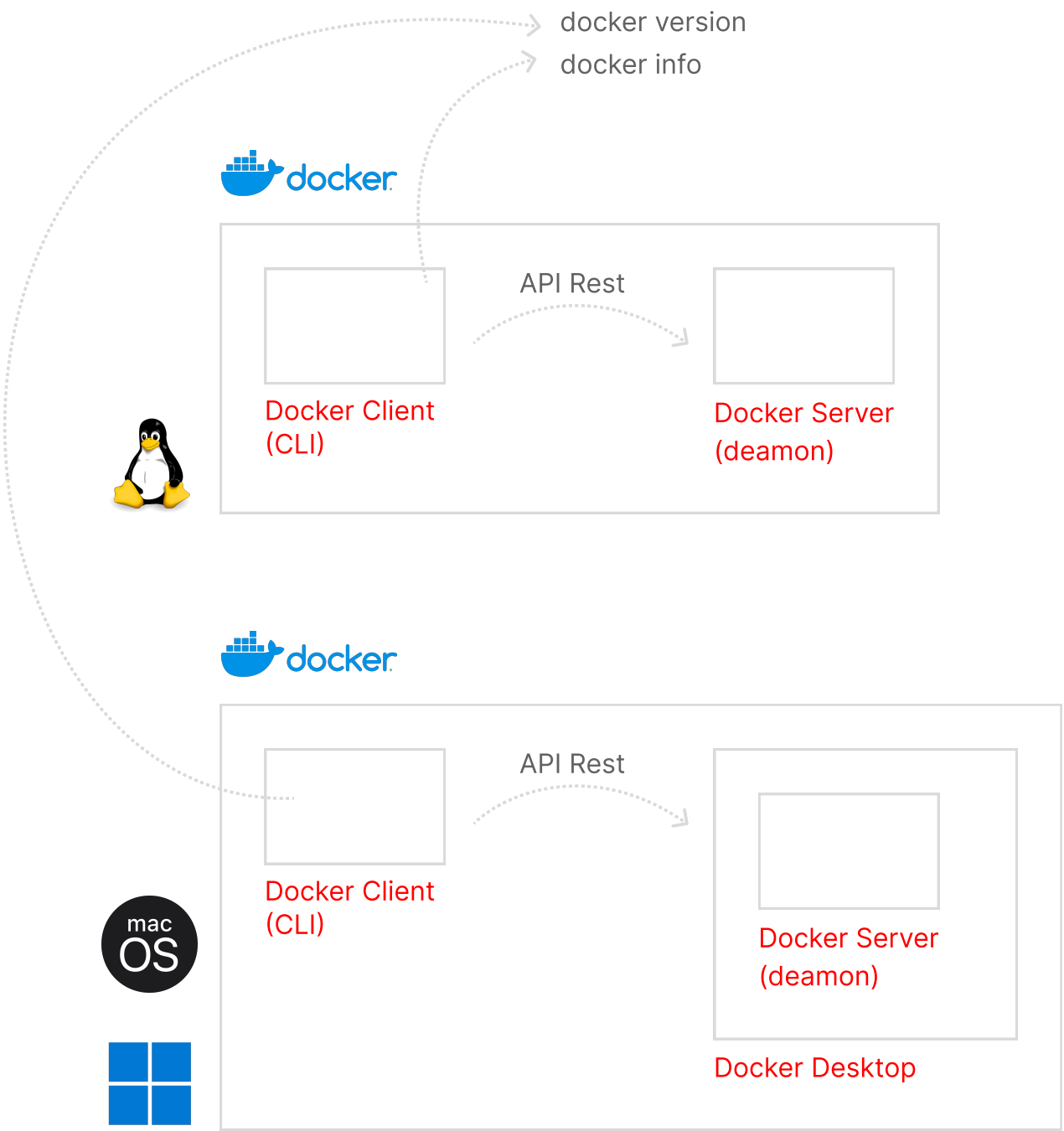
[MacOS] Instalando o Docker Desktop



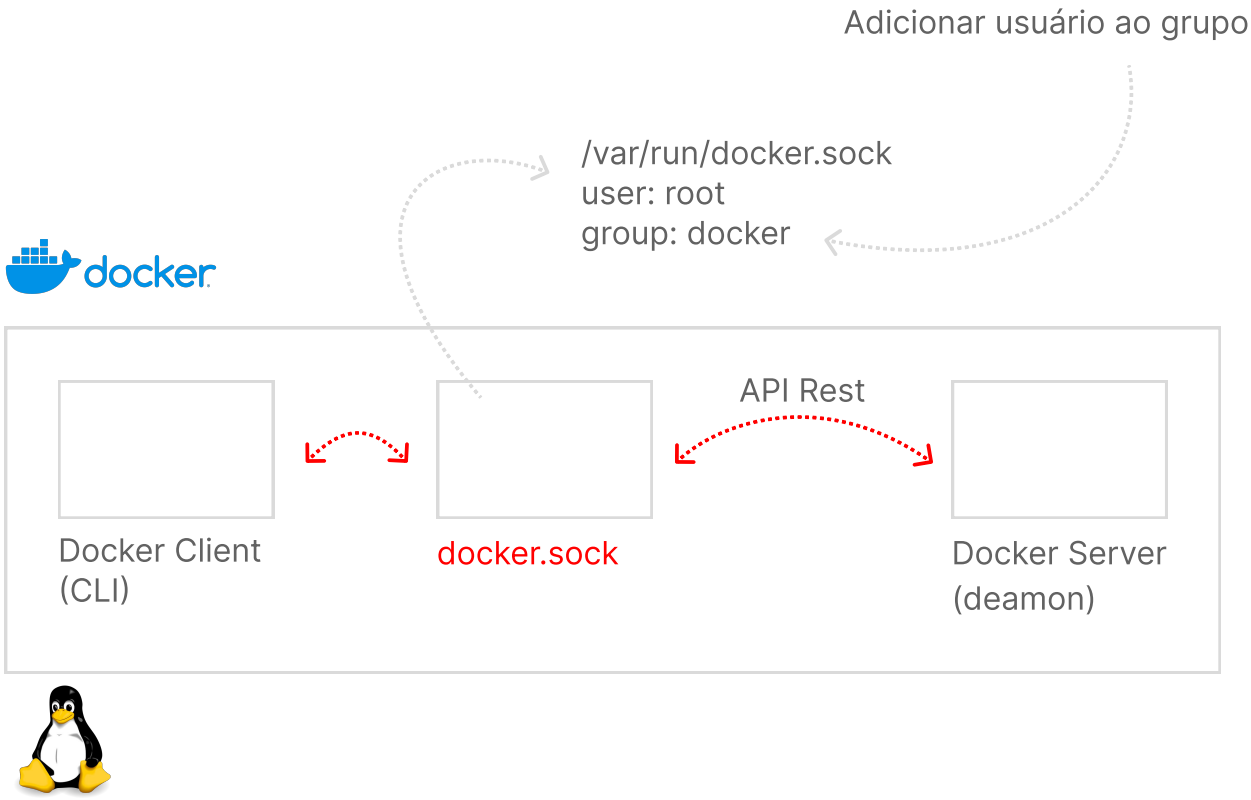
[MacOS] Instalando o VSCode



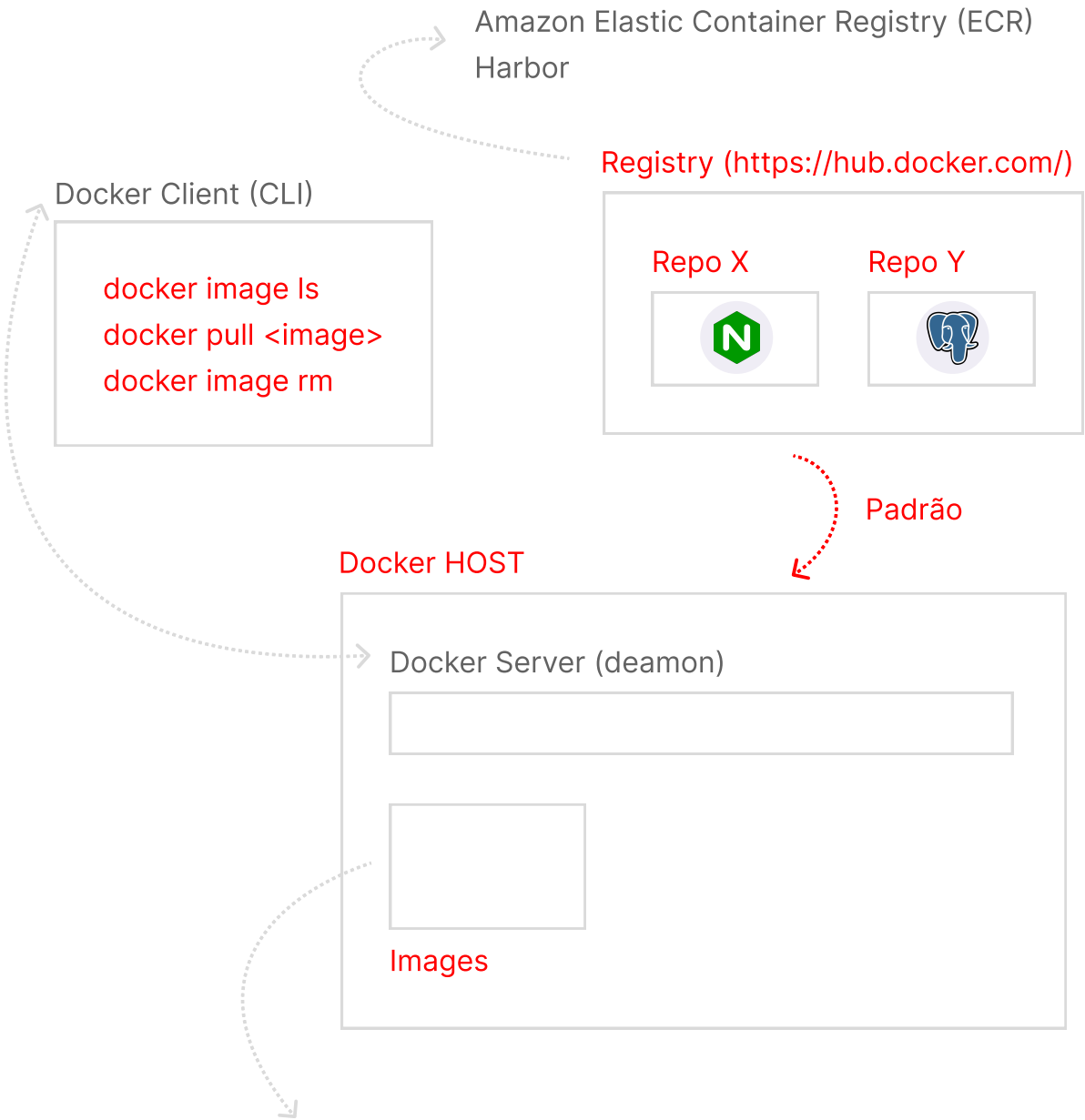
Conhecendo o Docker Client e o Docker Server



Configurando o acesso ao Docker Sock no Linux

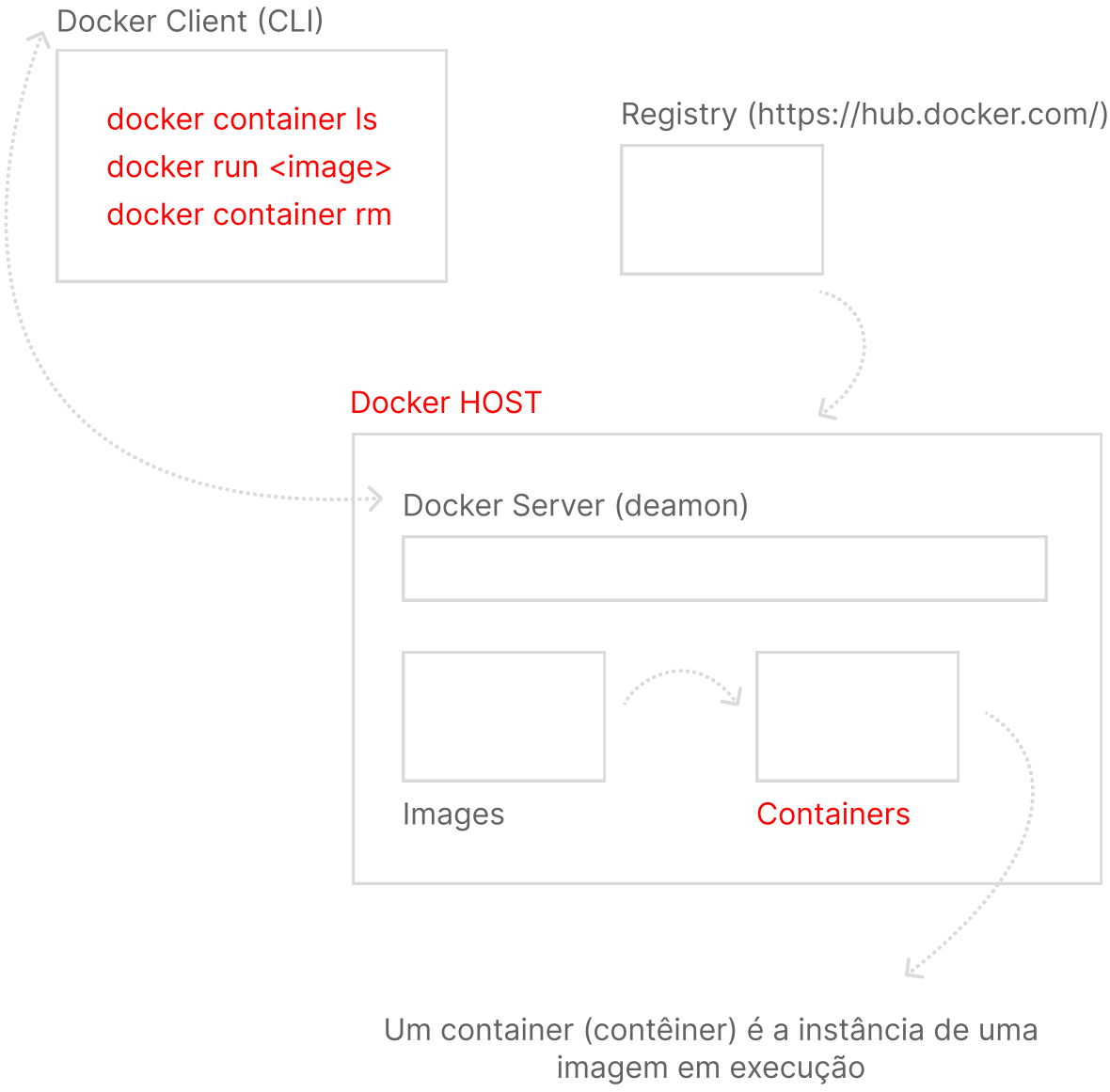


Introdução as Images e ao Docker Hub



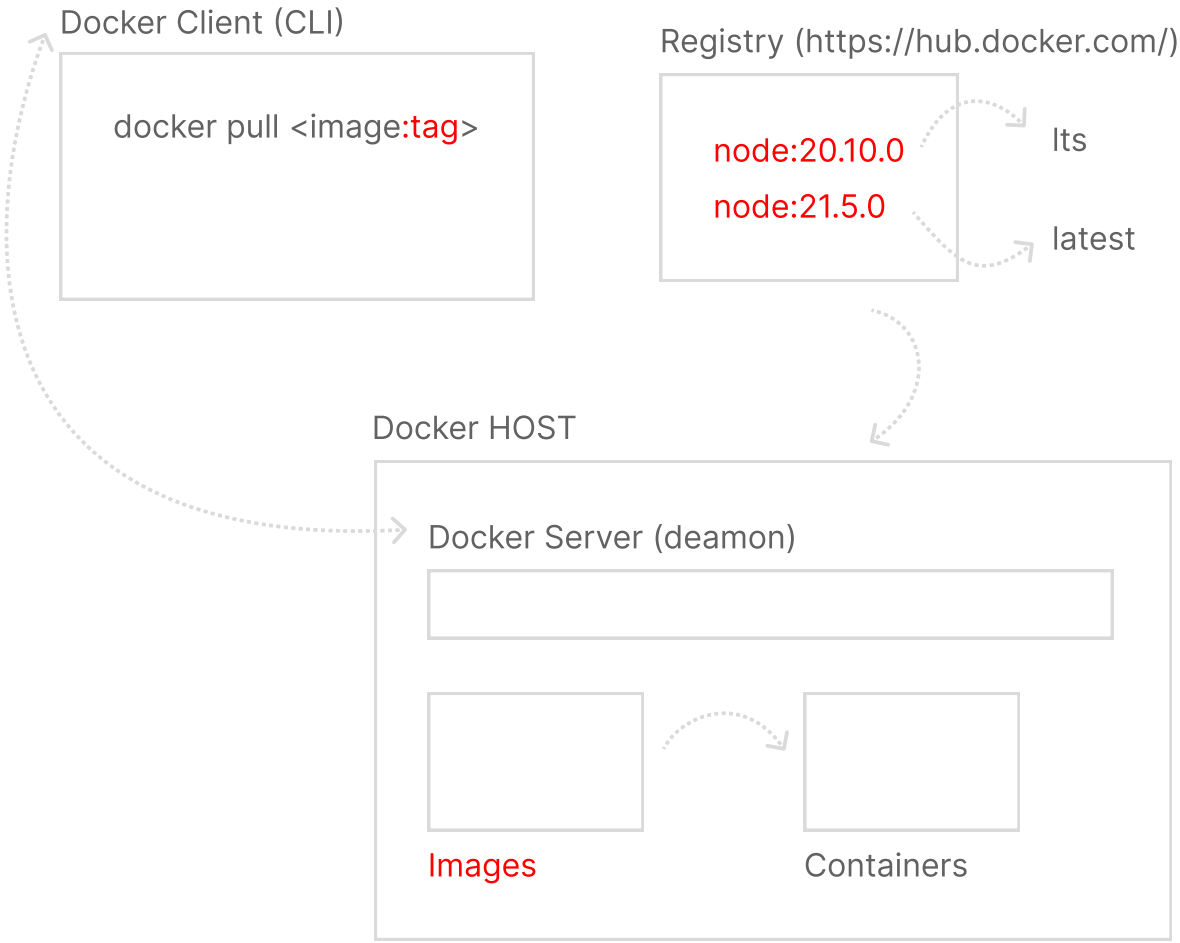
Uma imagem é um arquivo executável e auto-contido, com tudo o que necessário para executar o seu objetivo

Introdução aos Containers

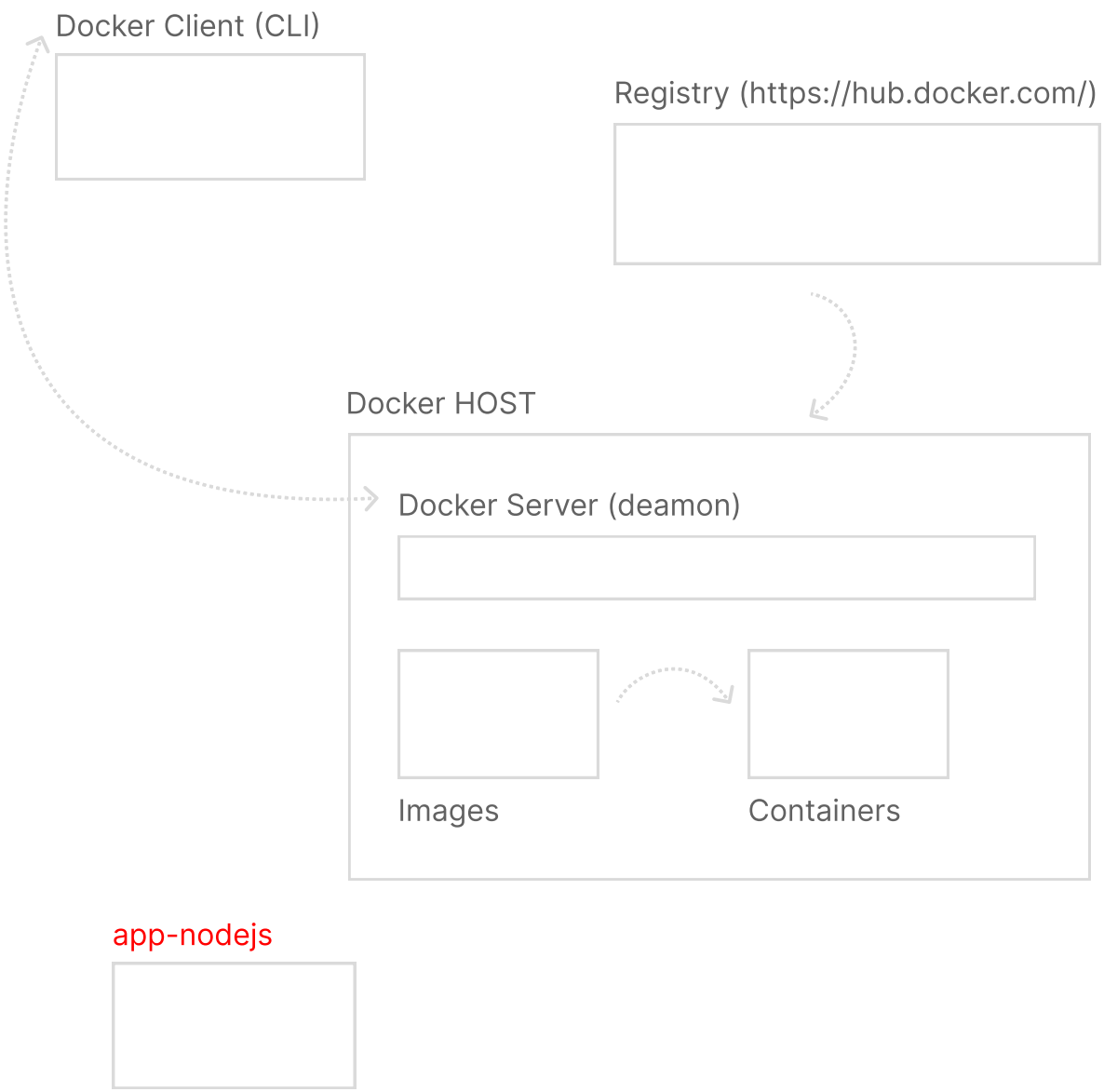


Um container (contêiner) é a instância de uma imagem em execução

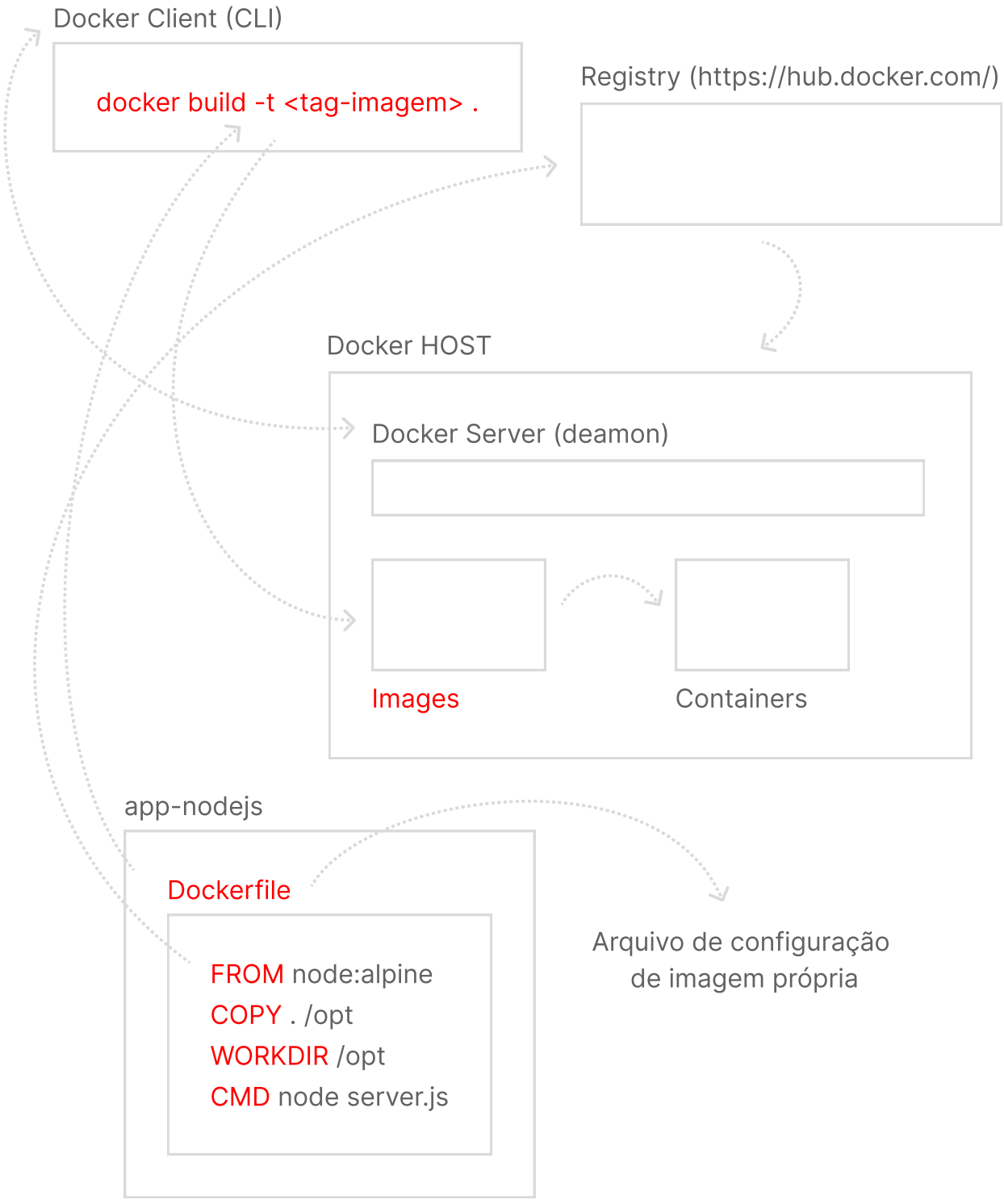
Baixando imagens de tags específicas



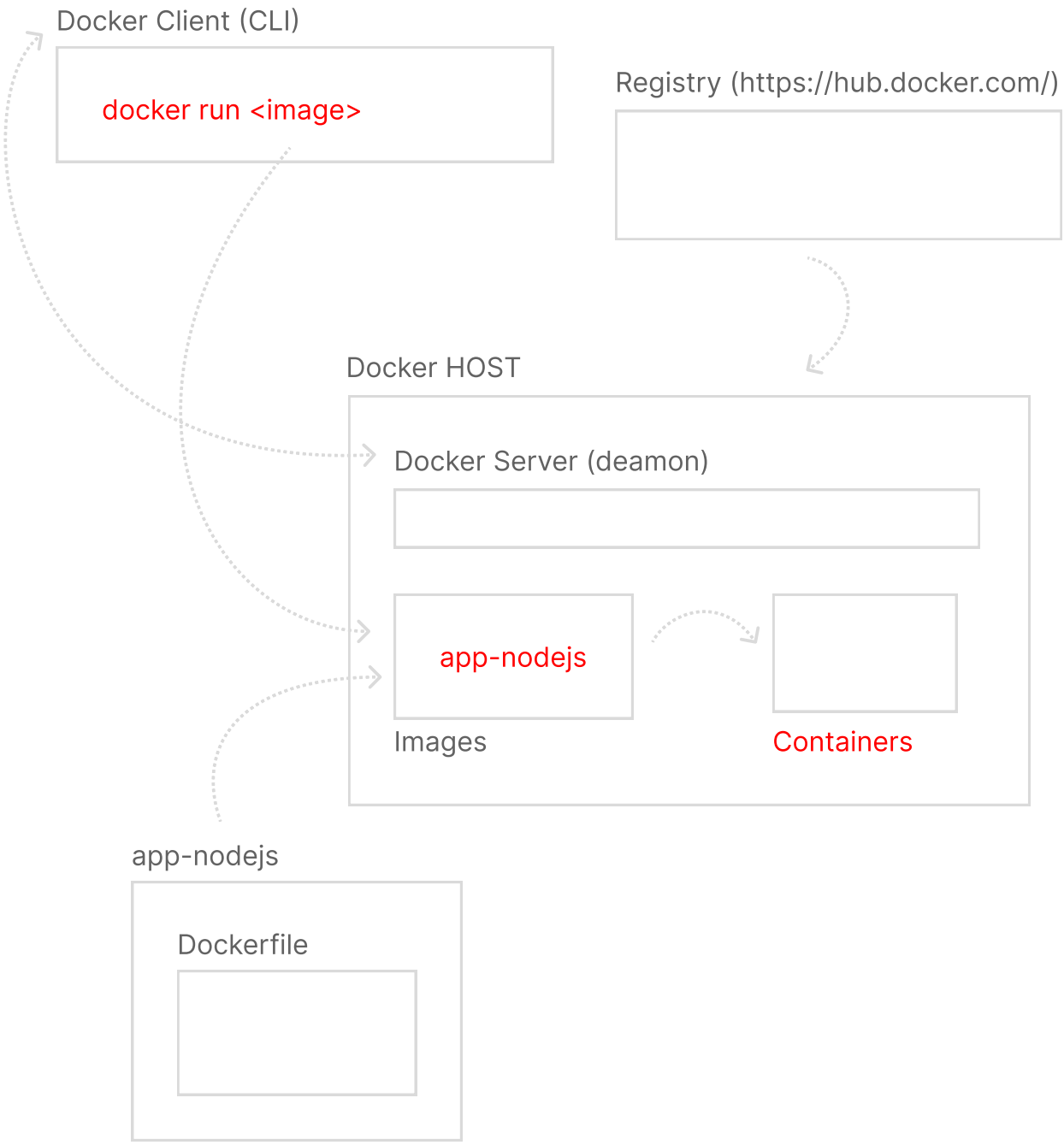
Criando uma imagem: Criando uma aplicação em NodeJS



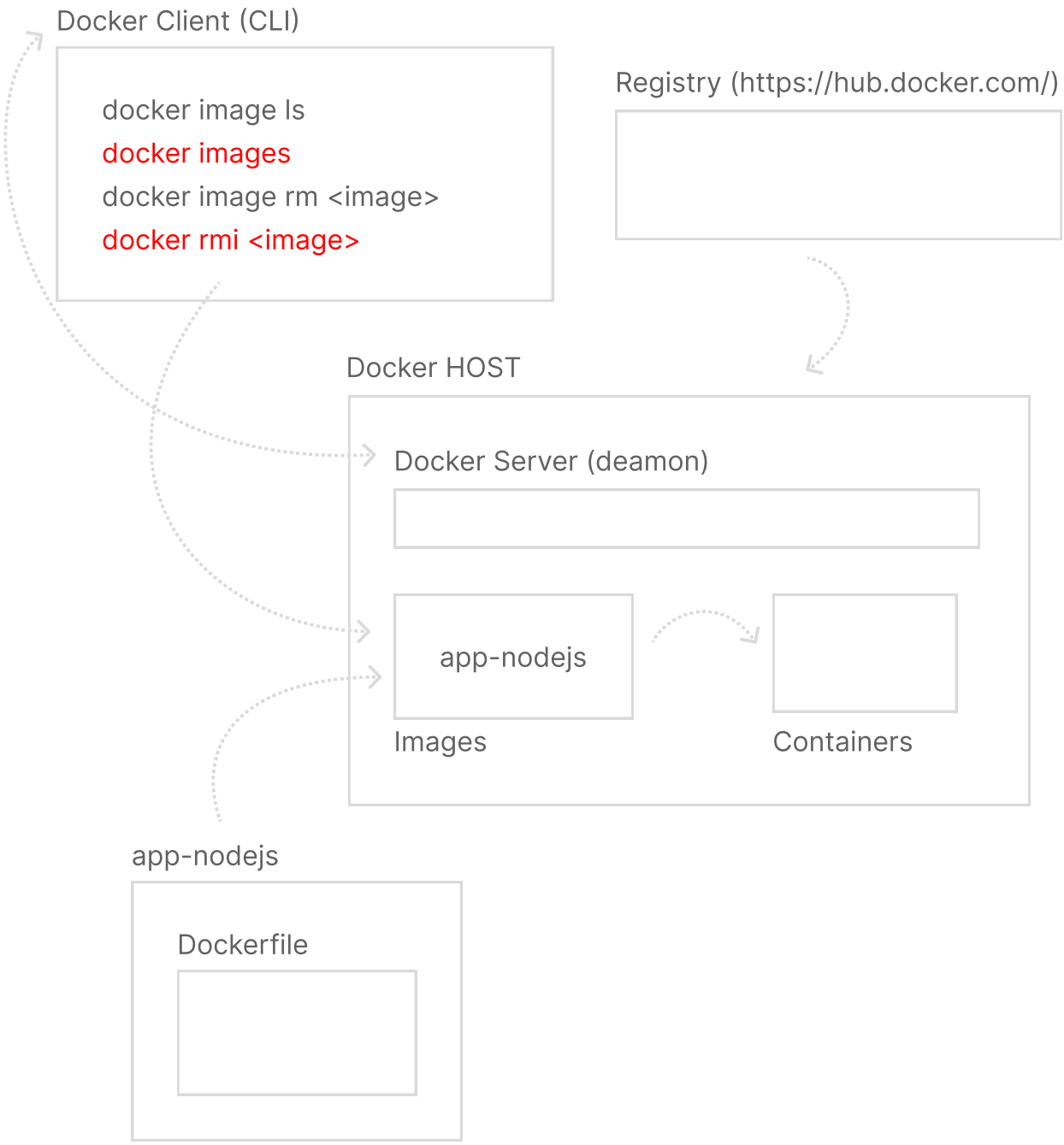
Criando uma imagem: Dockerfile e o build da imagem



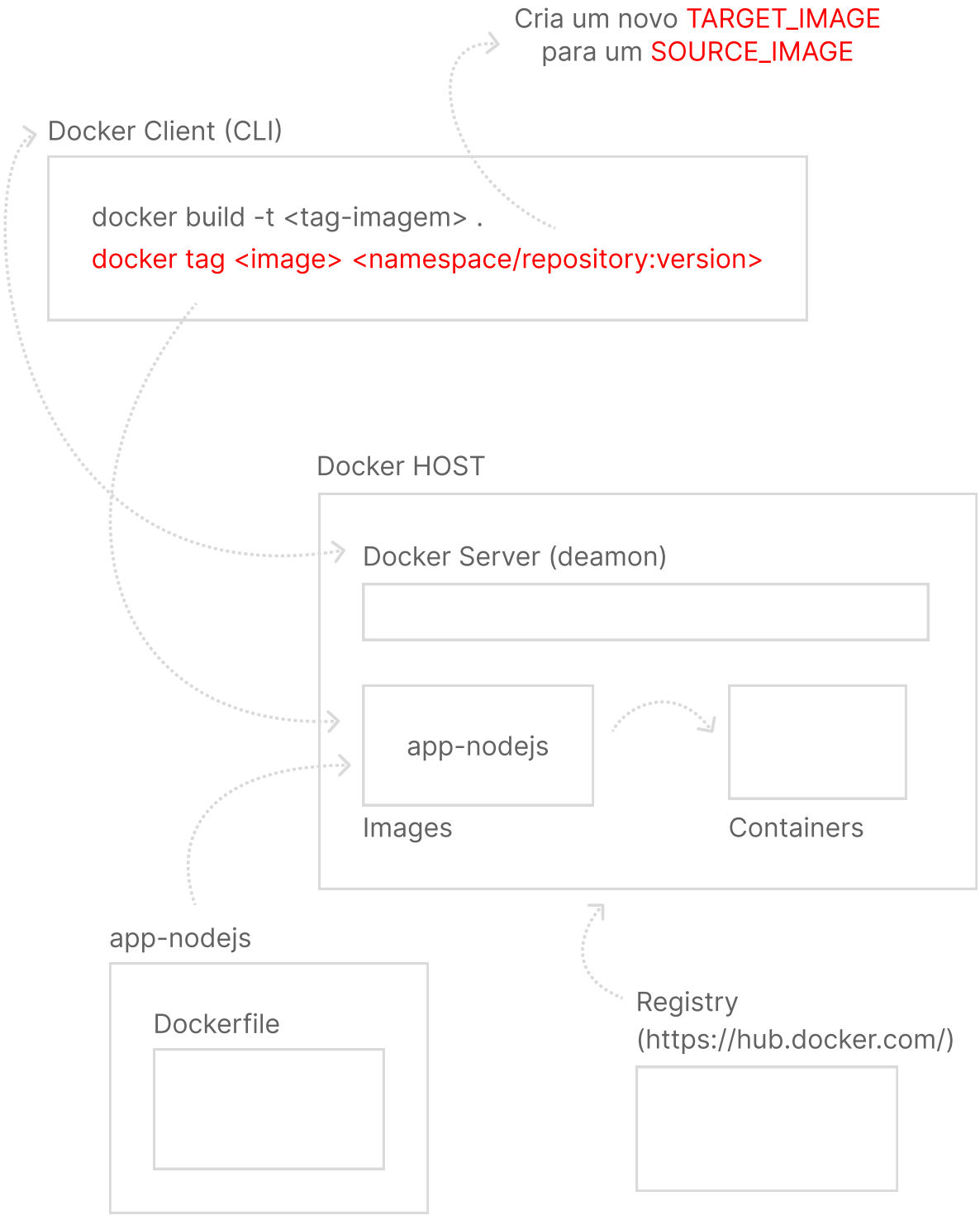
Criando uma imagem: Executando um container à partir da imagem



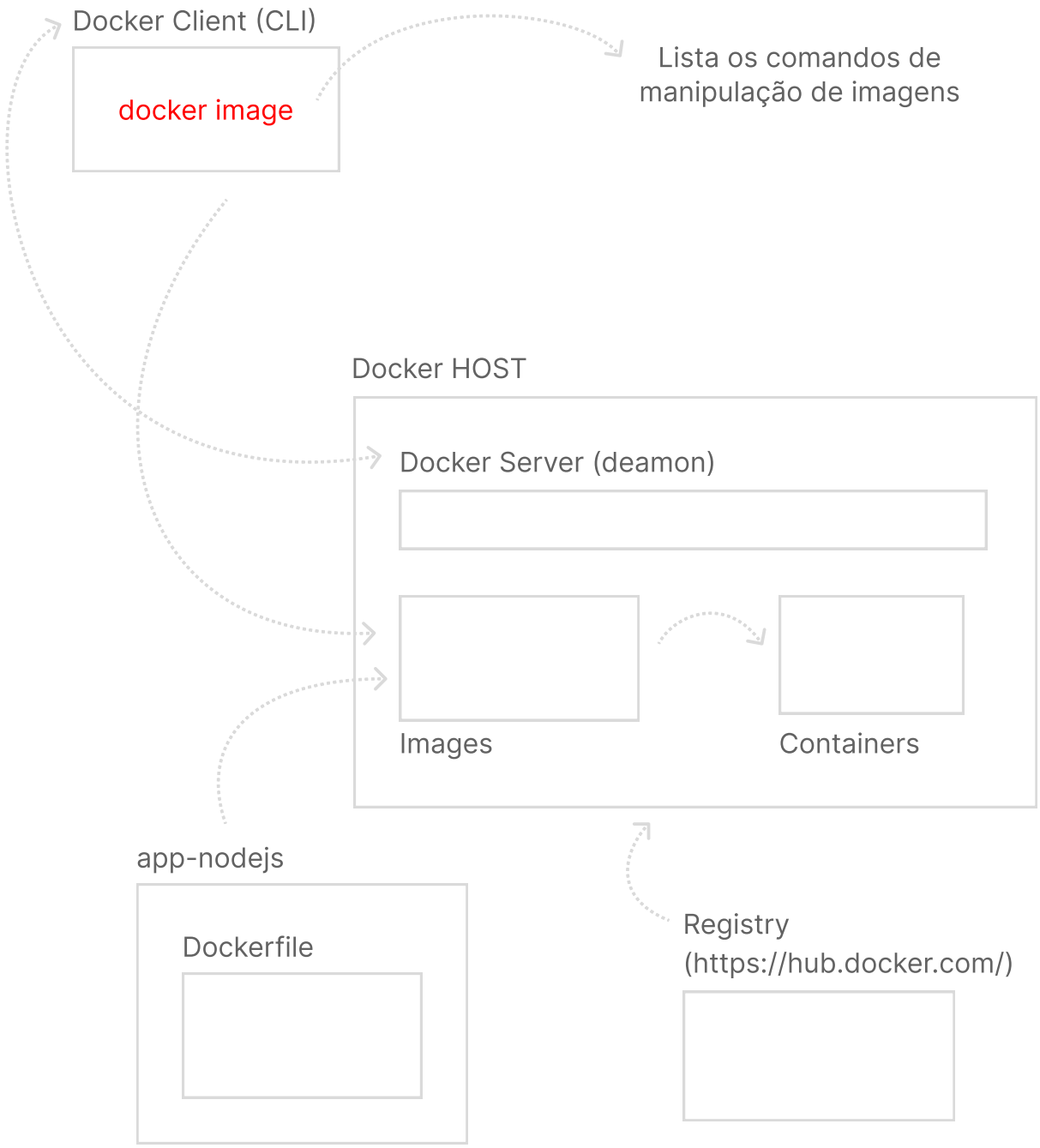
Comandos opcionais para listagem e remoção de imagens



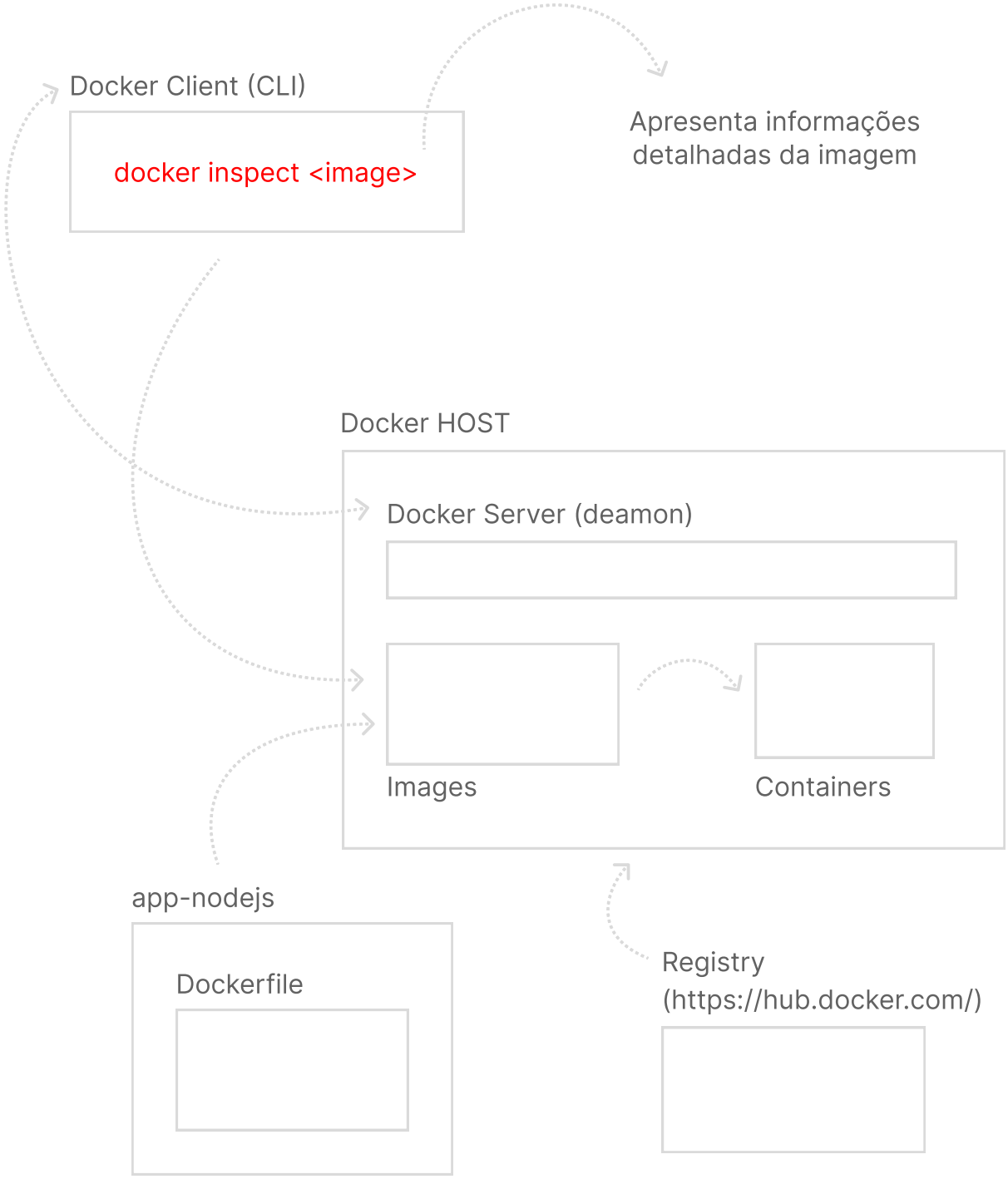
Etiquetando imagens



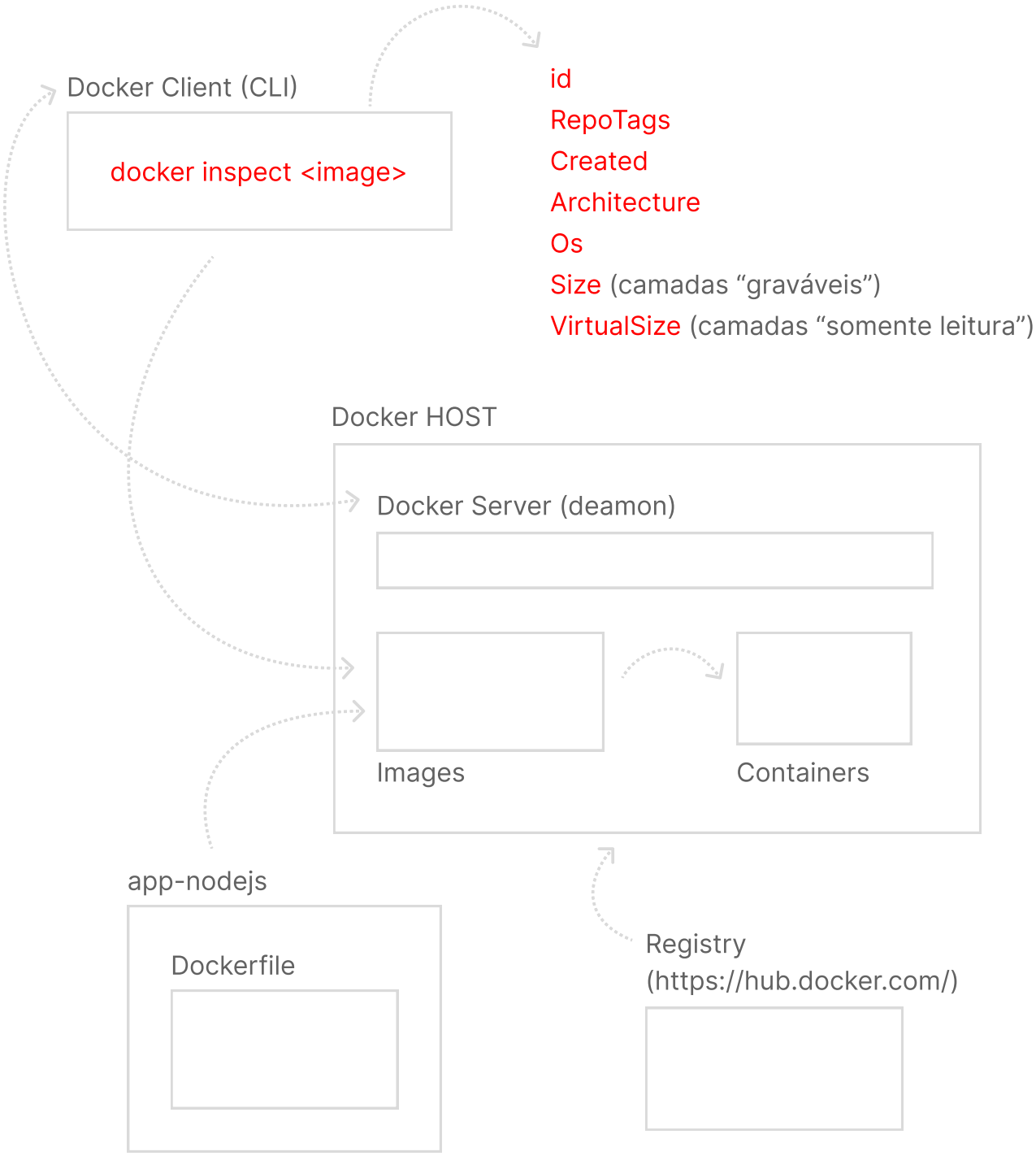
Exibindo os comandos de manipulação de imagens



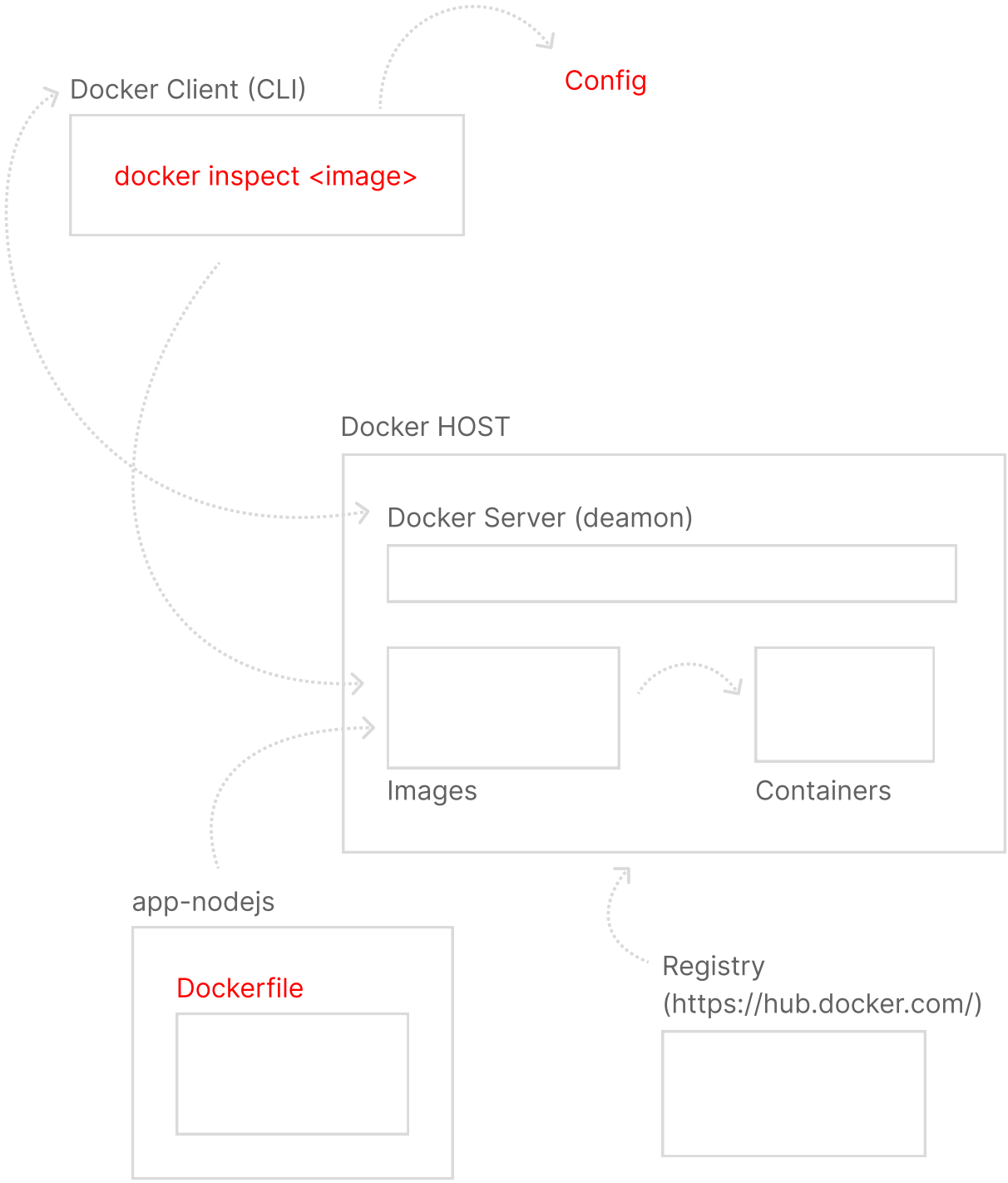
Inspecionando imagens



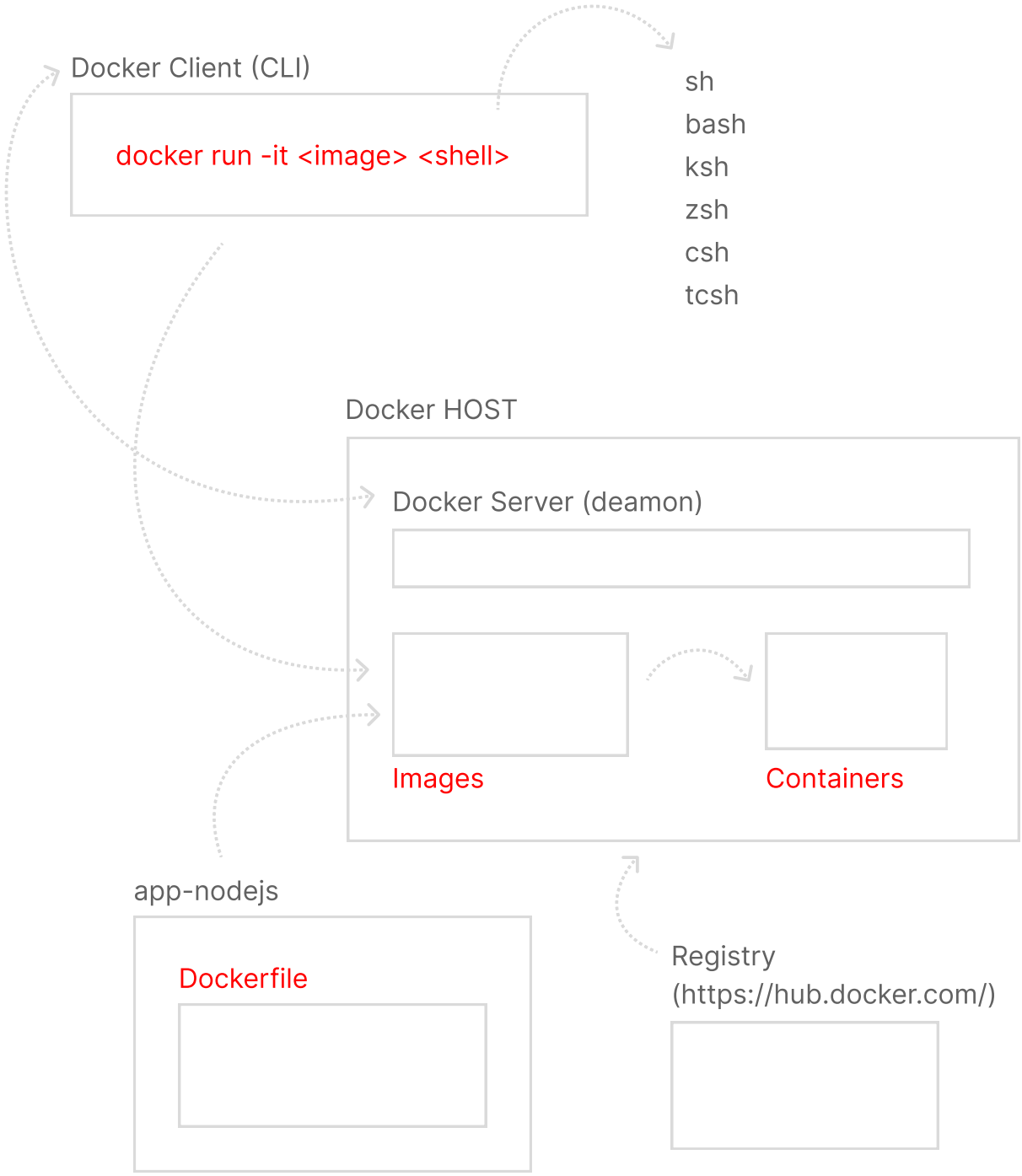
Inspecionando imagens - Detalhes 1



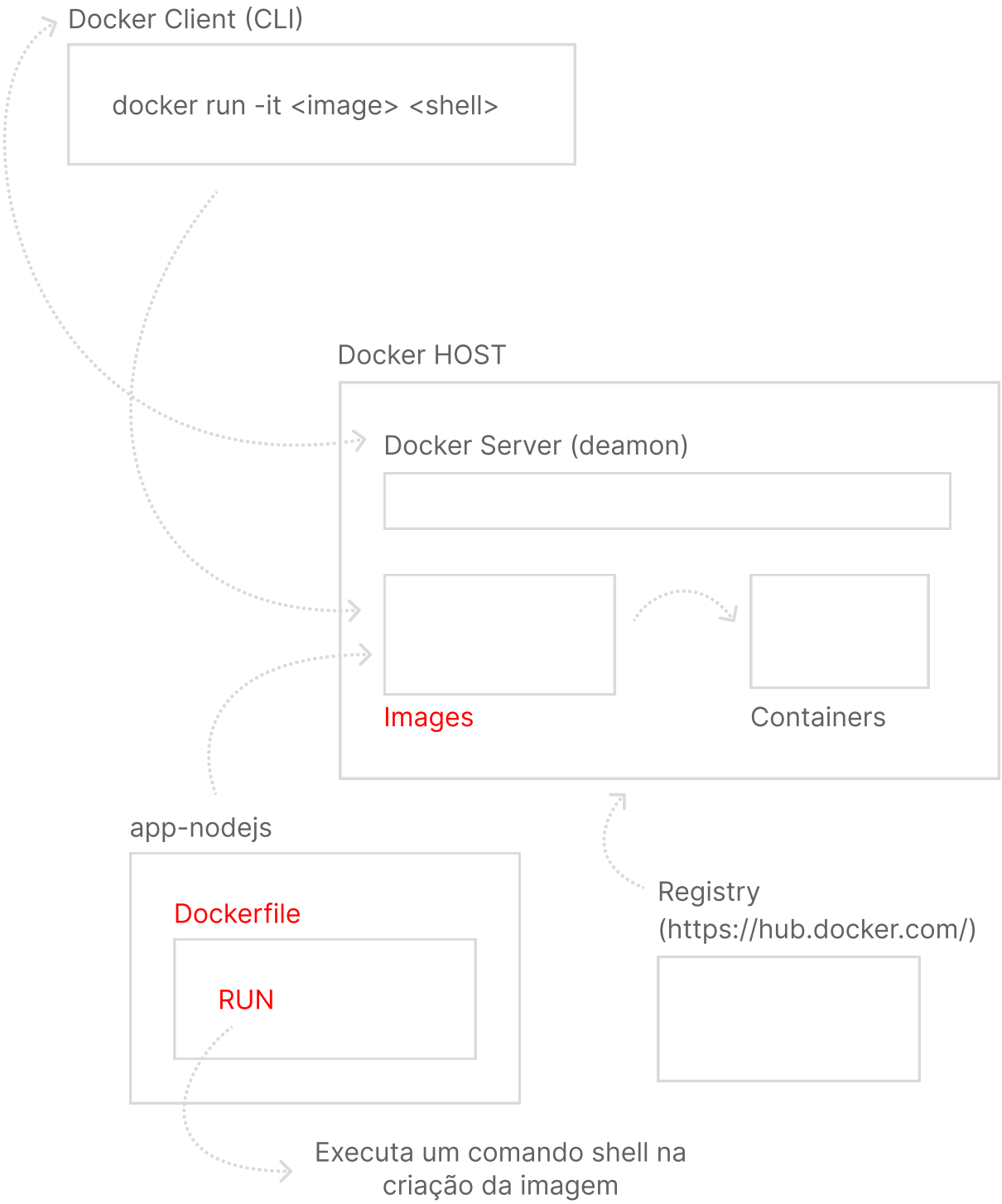
Inspecionando imagens - Detalhes 2



Acessando o container durante a inicialização via terminal interativo



Dockerfile - RUN



Dockerfile - RUN multilinha

Dockerfile

```
RUN comando1
RUN comando2
RUN comando3
```

Dockerfile

```
RUN comando1 && \
    comando2 && \
    comando3
```

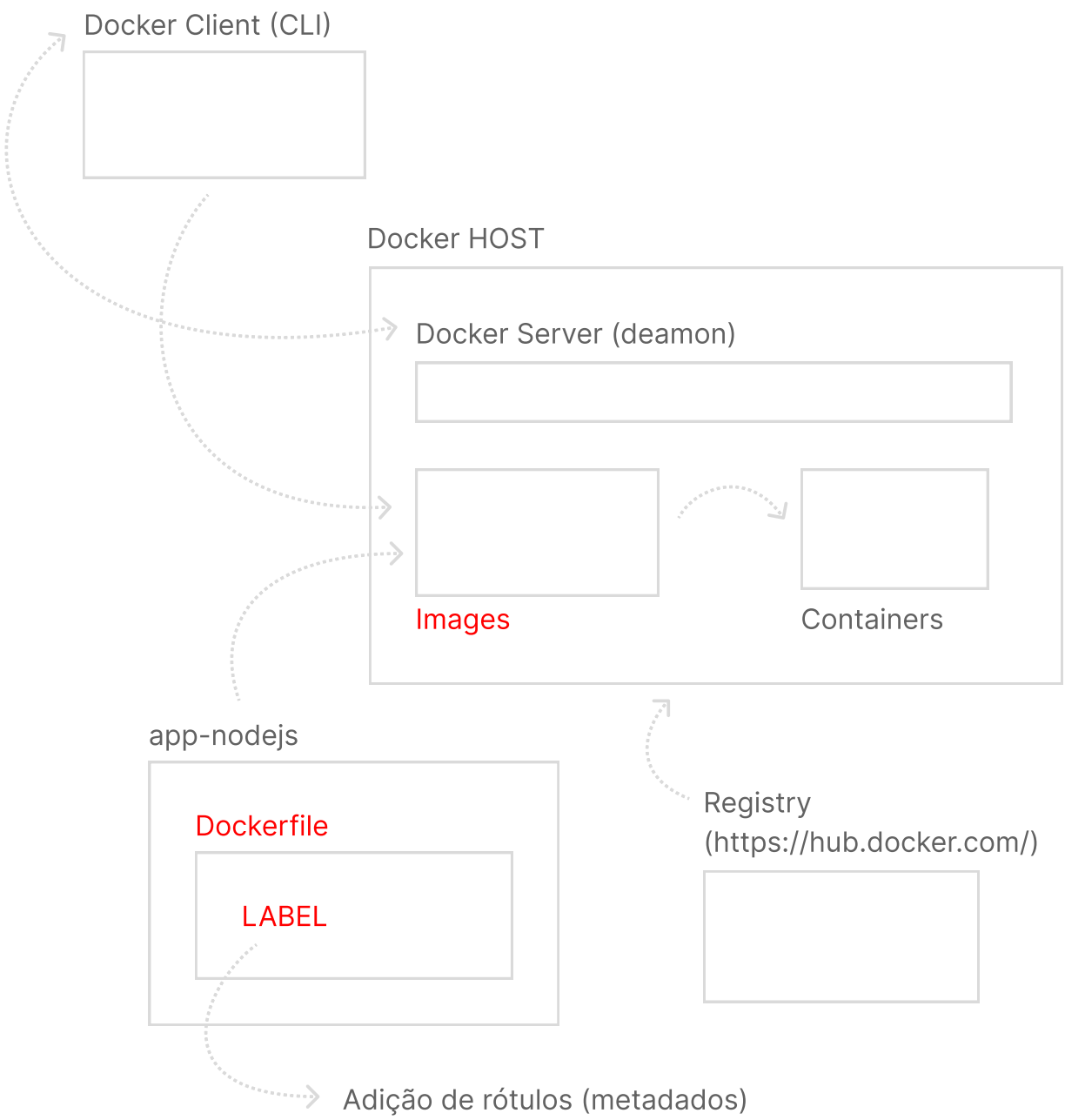
Dockerfile

```
RUN comando1 \
    && comando2 \
    && comando3
```

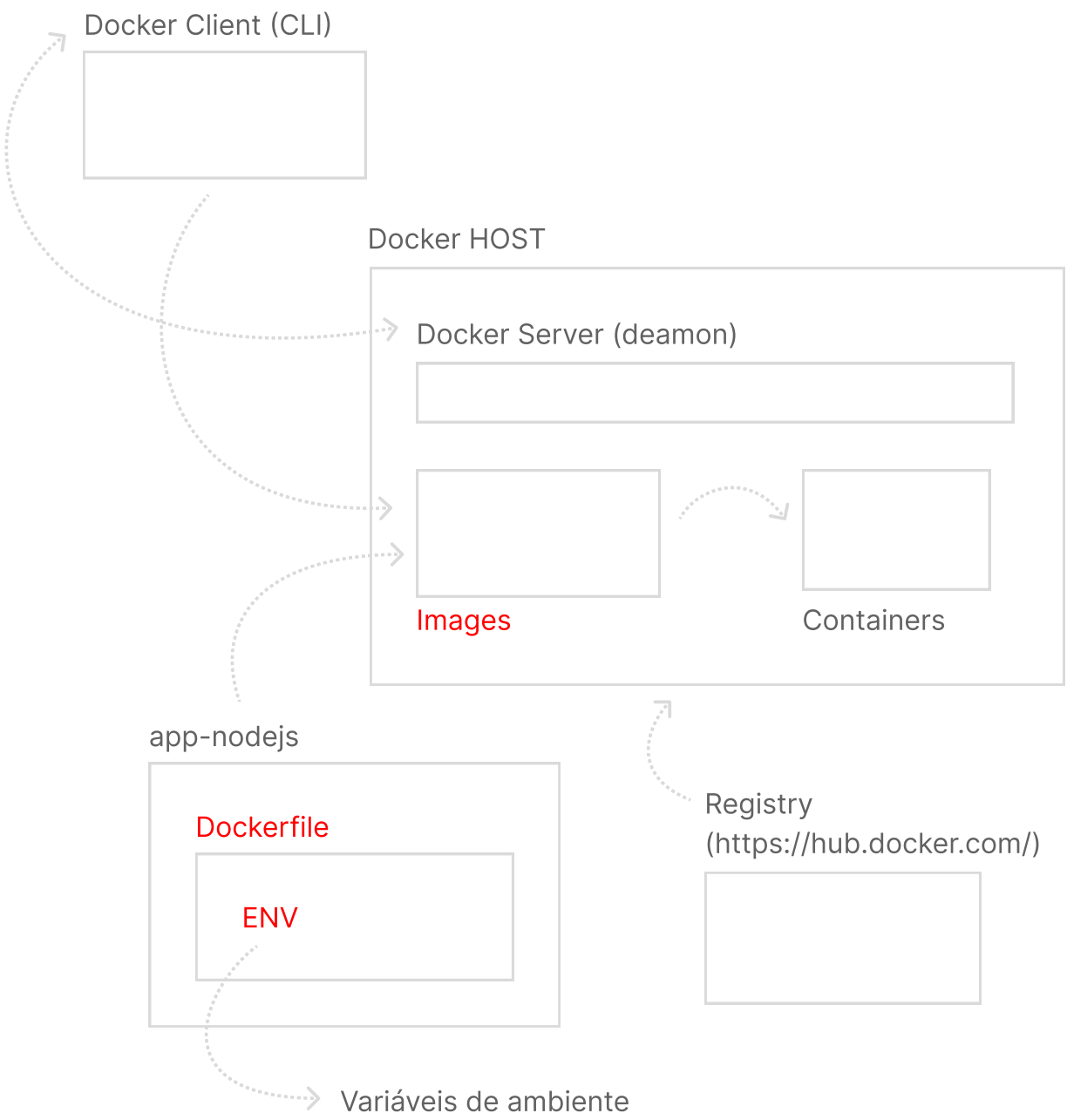
Dockerfile

```
RUN <<EOF
    comando1
    comando2
    comando3
EOF
```

Dockerfile - LABEL

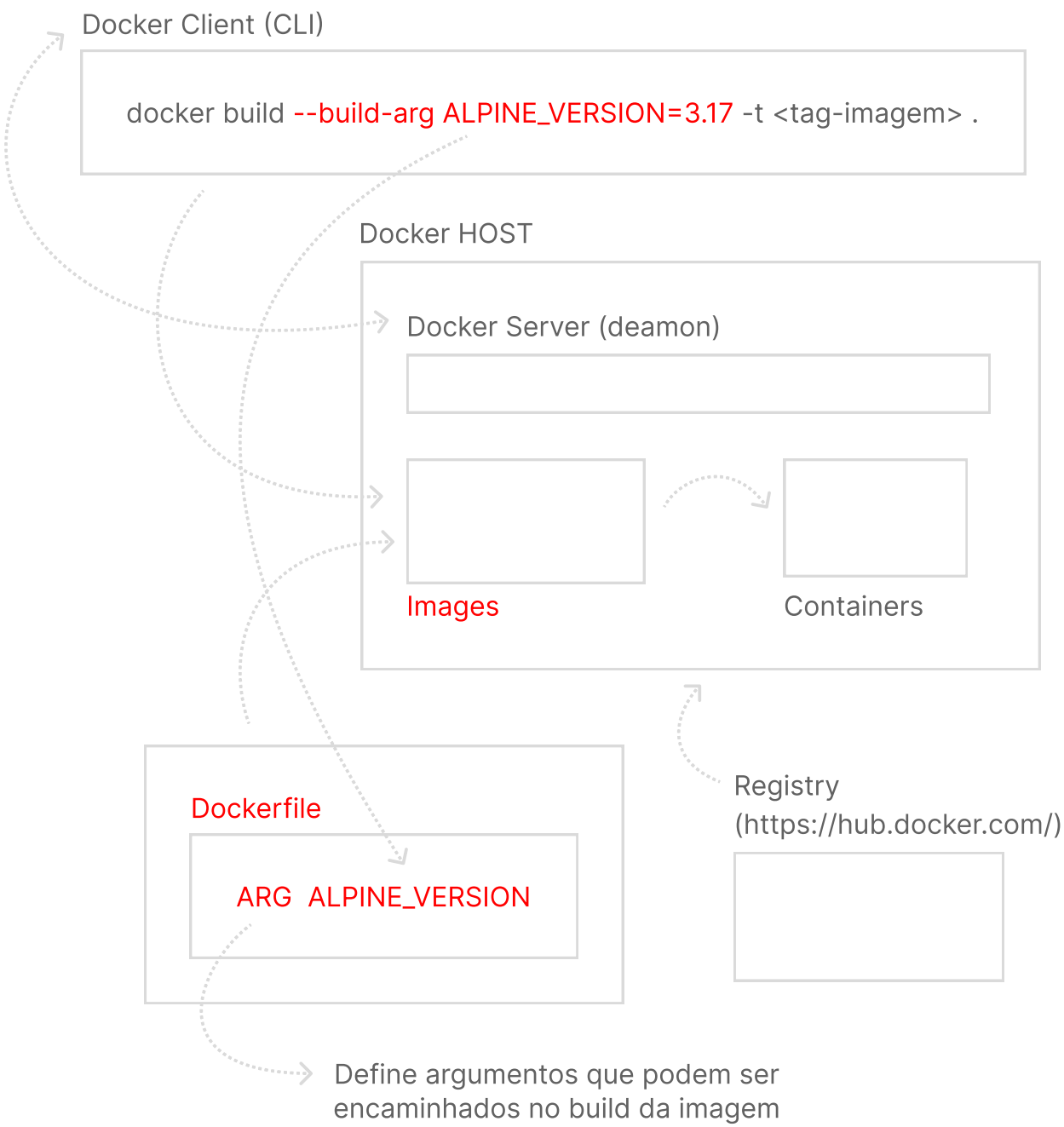


Dockerfile - ENV



Dockerfile - ENV reutilizando variáveis de ambiente no Dockerfile

Dockerfile - ARG



Dockerfile - ARG múltiplos argumentos

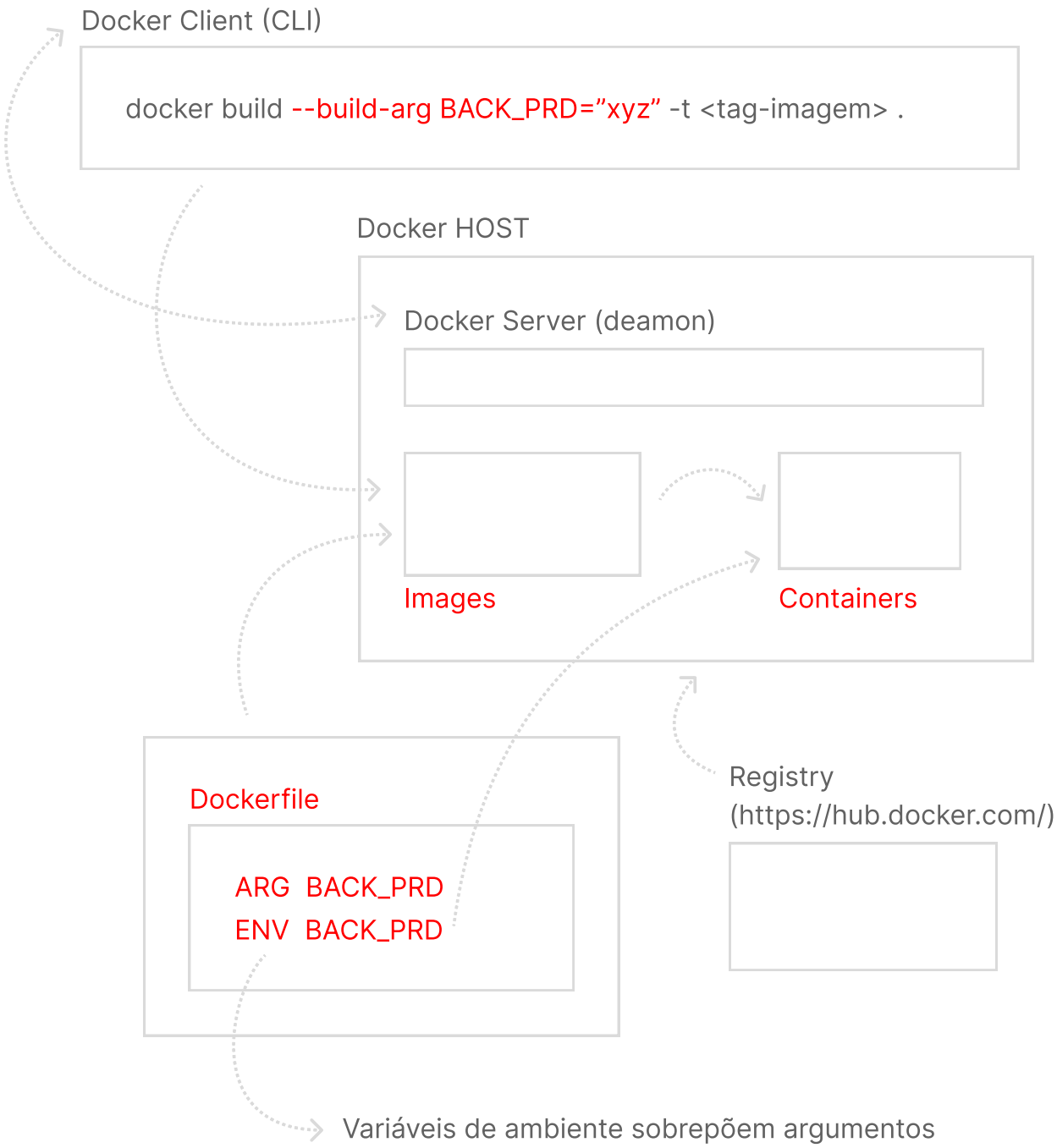
Docker Client (CLI)

```
docker build --build-arg ALPINE_VERSION=3.17 \
--build-arg FRONT_PRD=https://meuapp.com.br \
--build-arg BACK_PRD=https://meuapp.com.br:3002 \
-t <tag-imagem> .
```

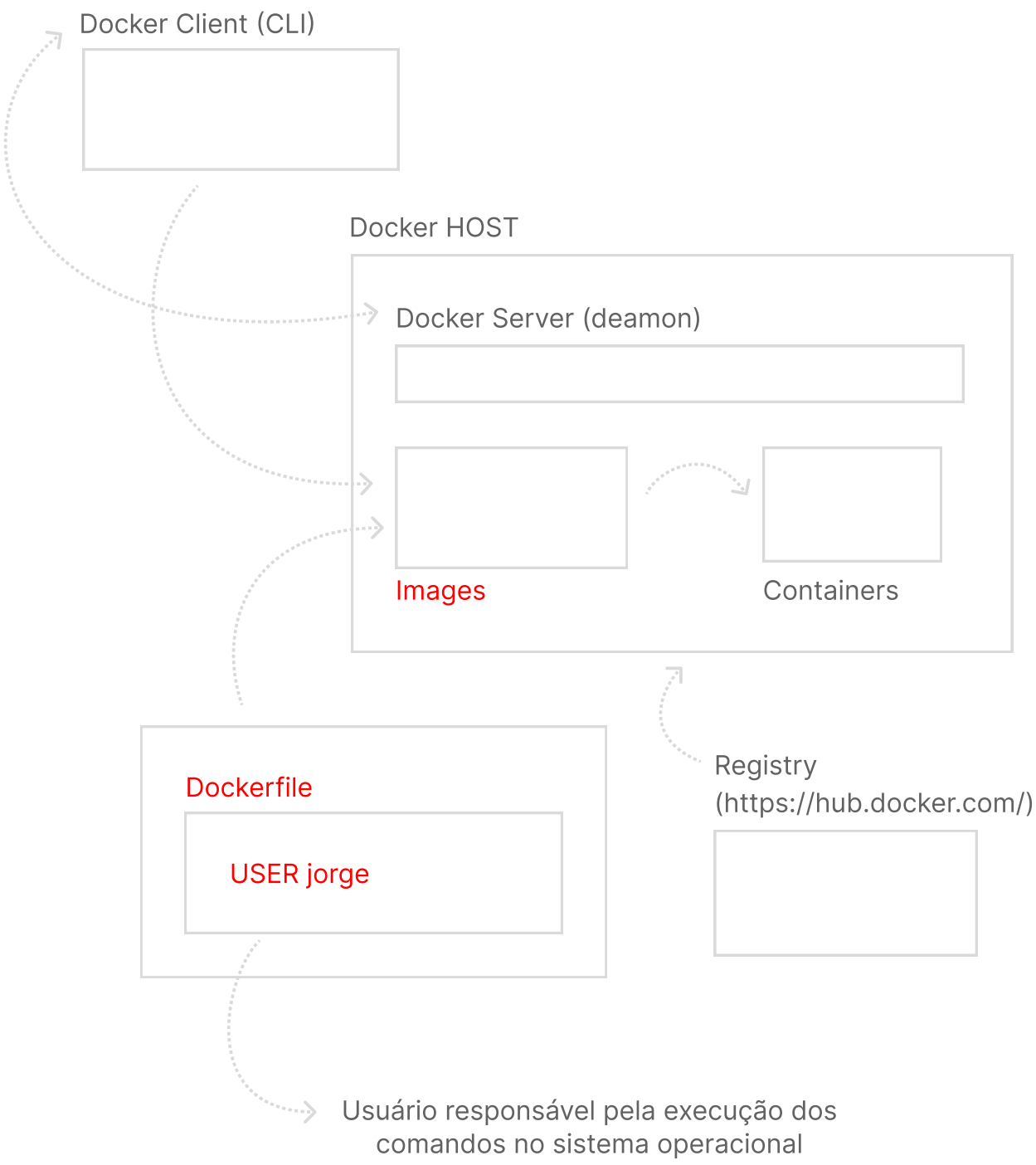
Dockerfile

```
ARG  ALPINE_VERSION
ARG  FRONT_PRD
ARG  BACK_PRD
```

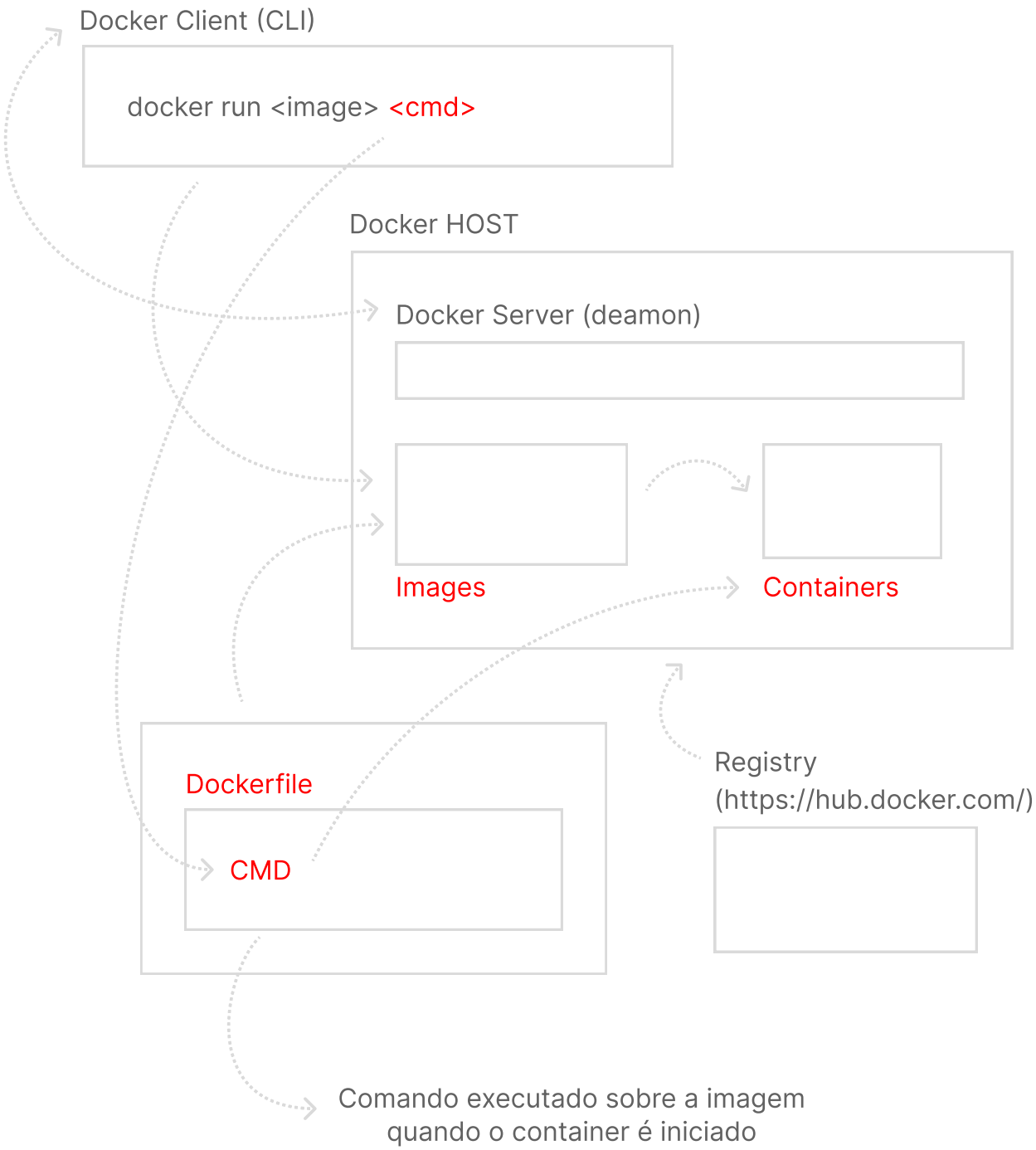
Dockerfile - ARG vs ENV



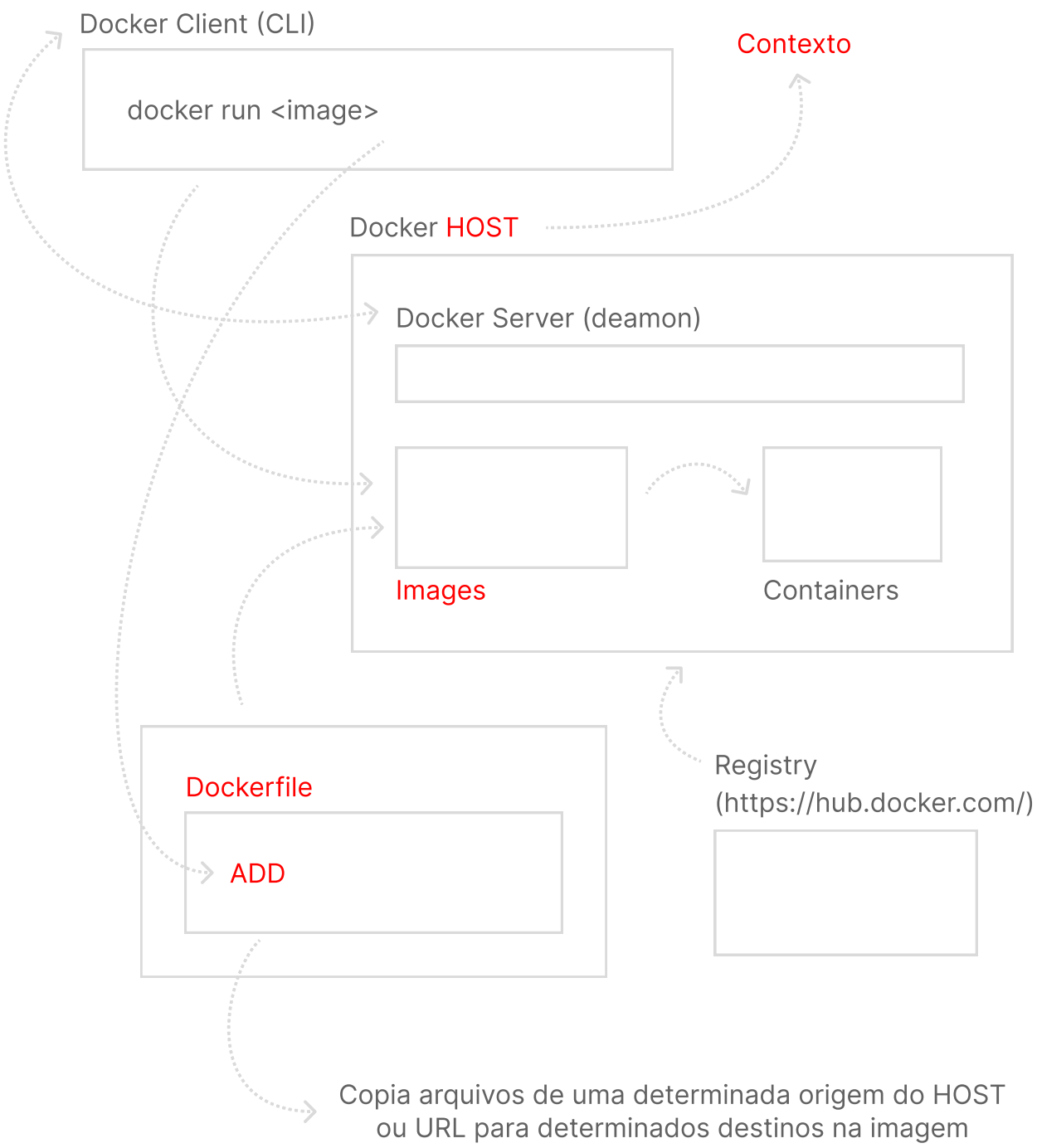
Dockerfile - USER



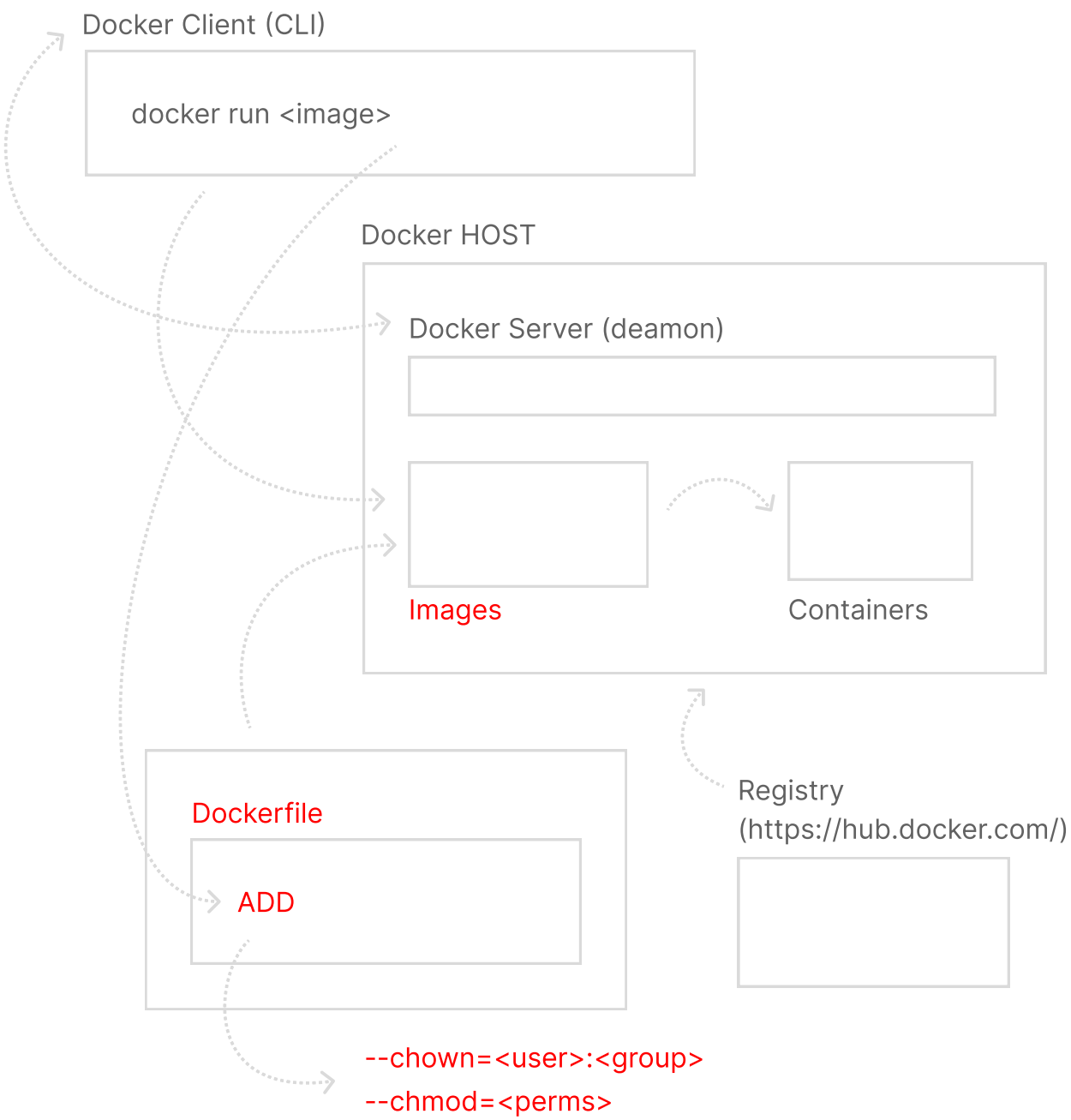
Dockerfile - CMD



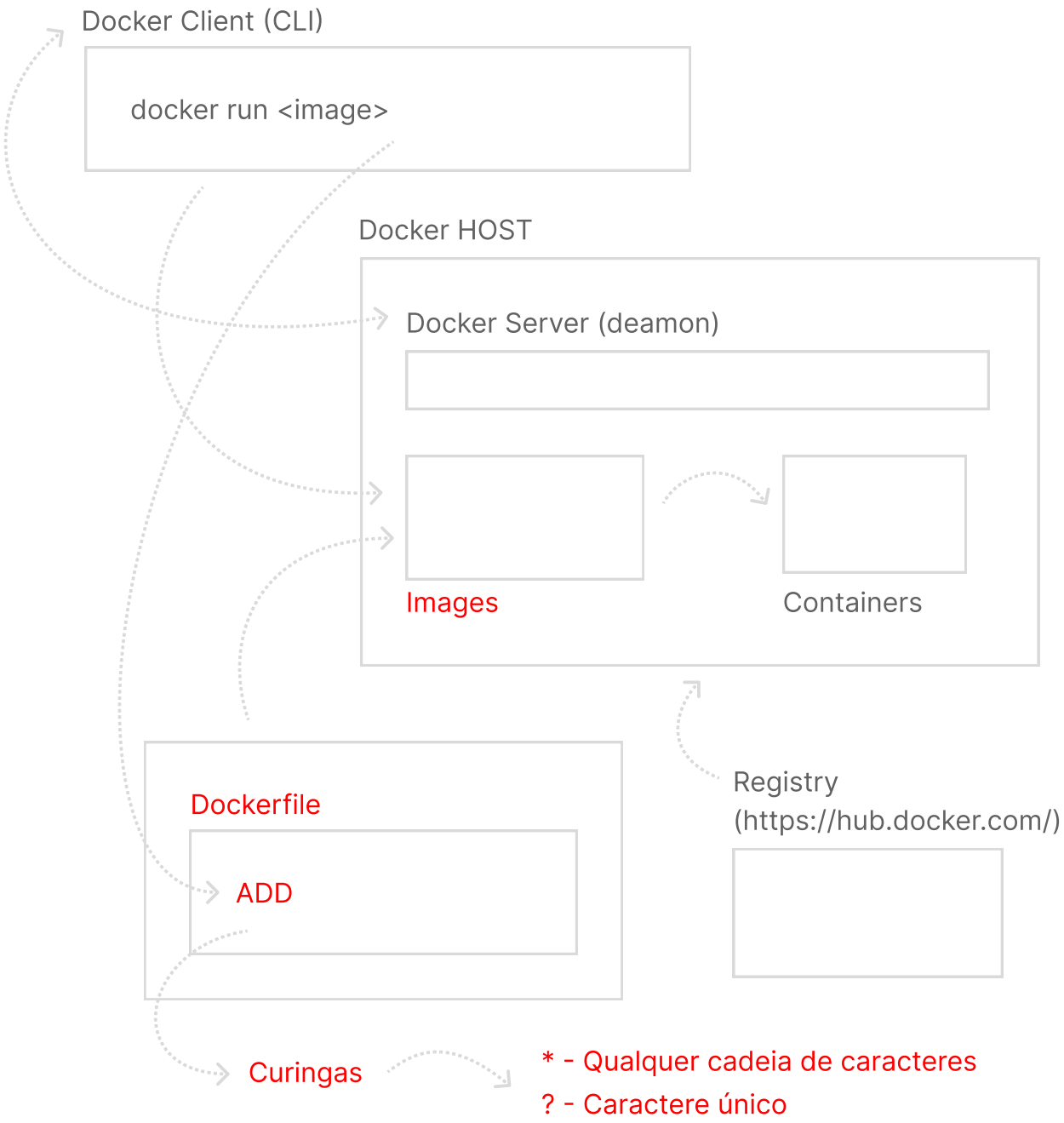
Dockerfile - ADD



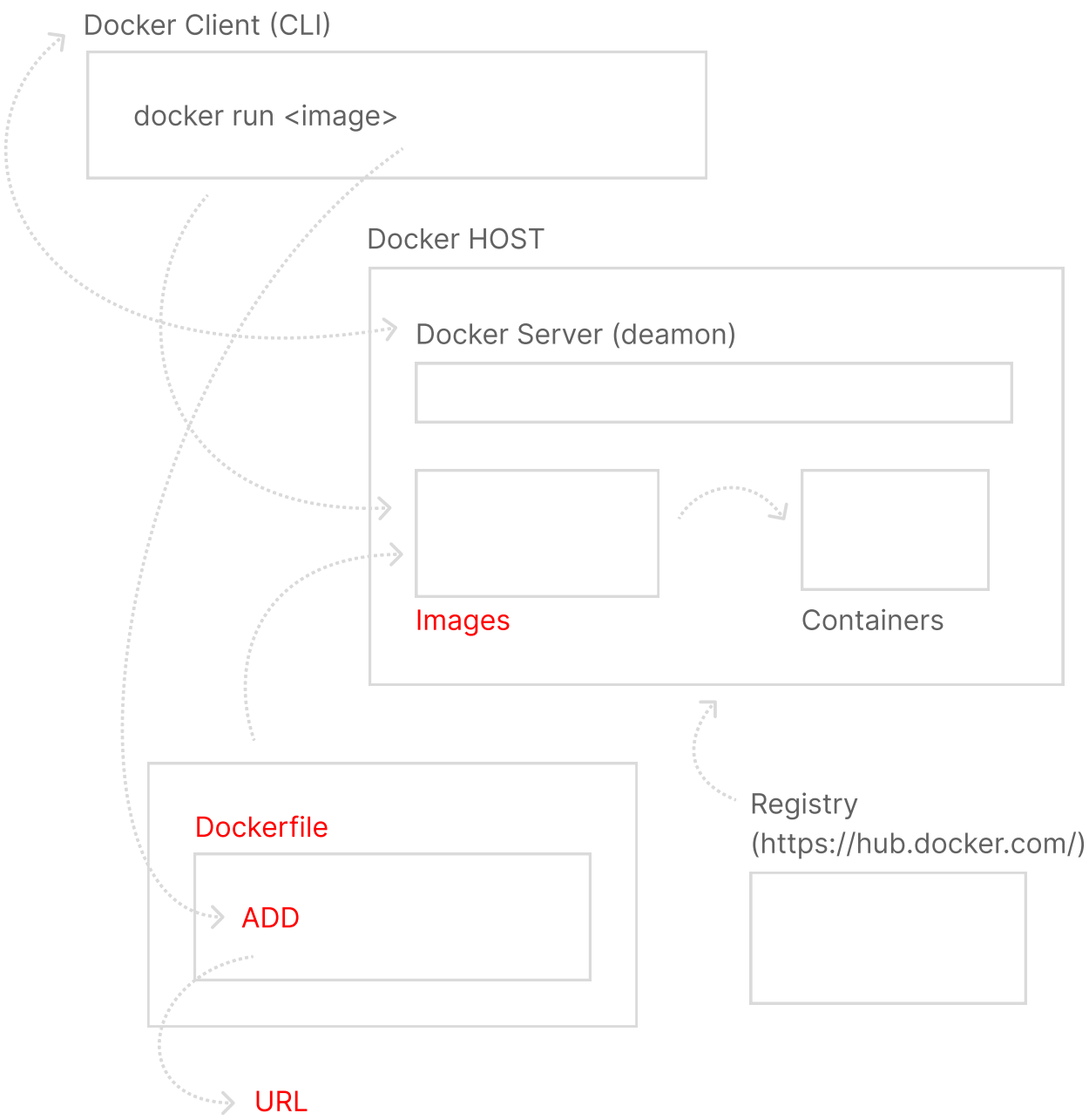
Dockerfile - ADD ajustando permissões



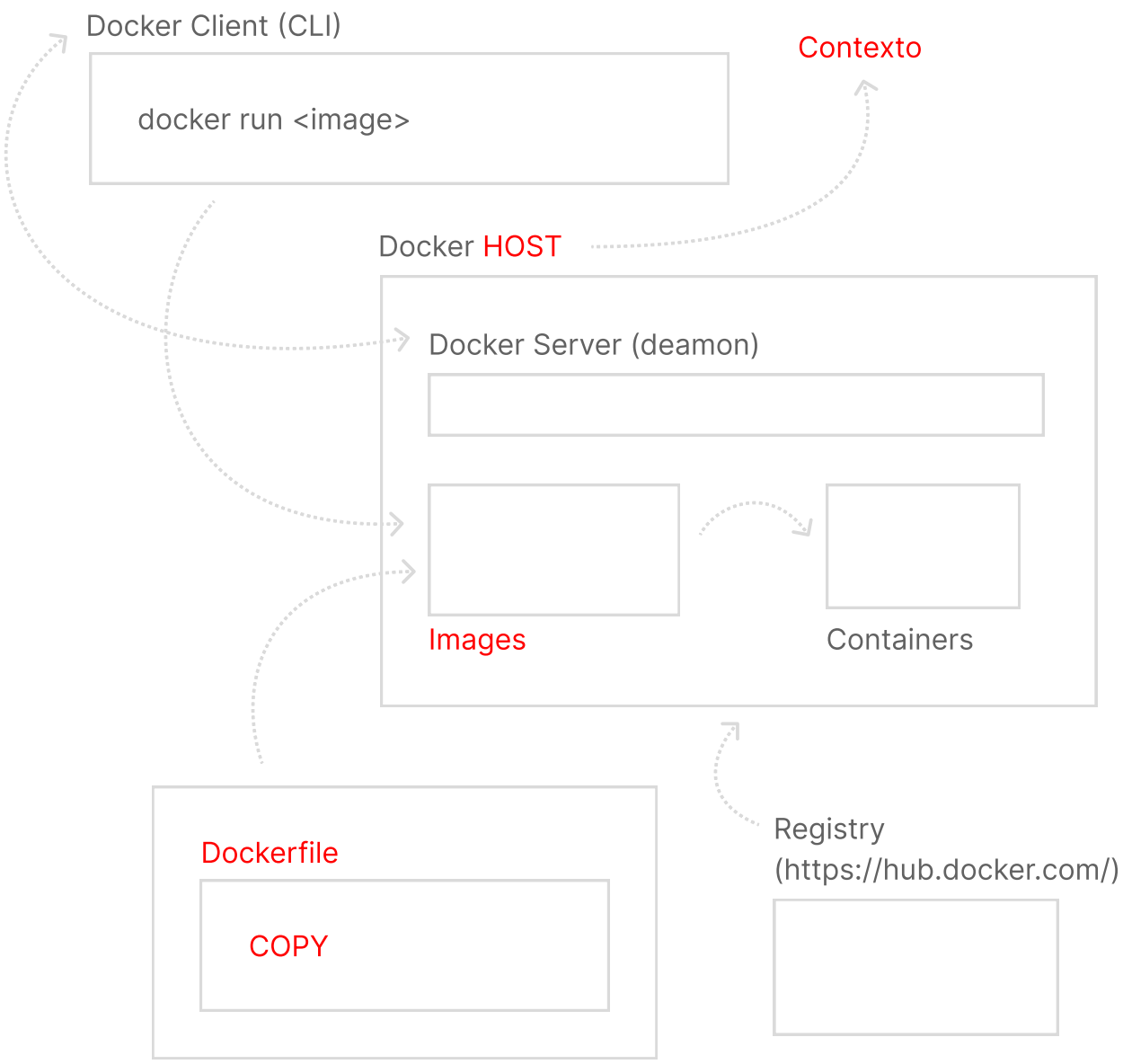
Dockerfile - ADD curingas e arquivos zipados



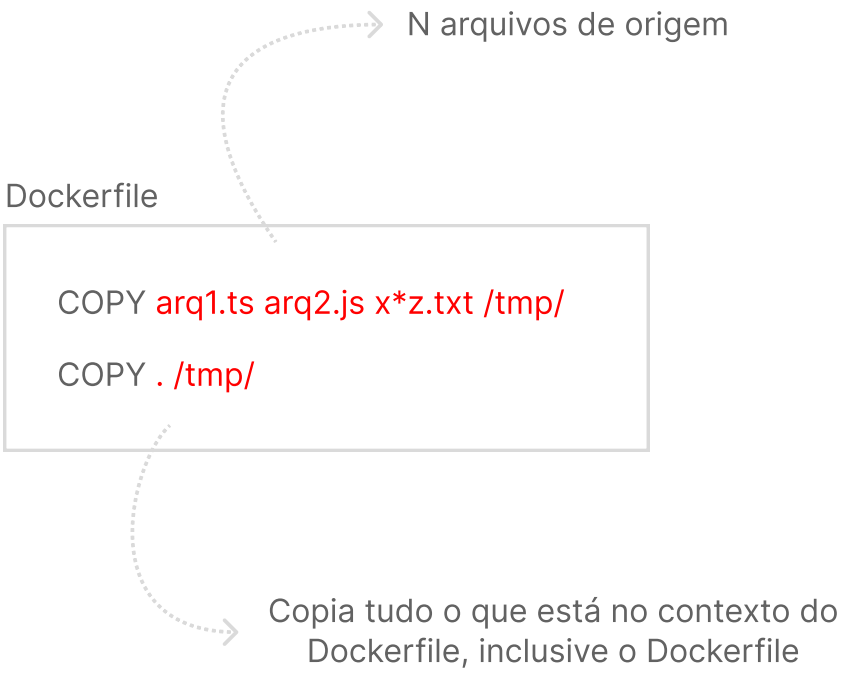
Dockerfile - ADD copiando arquivos via URL



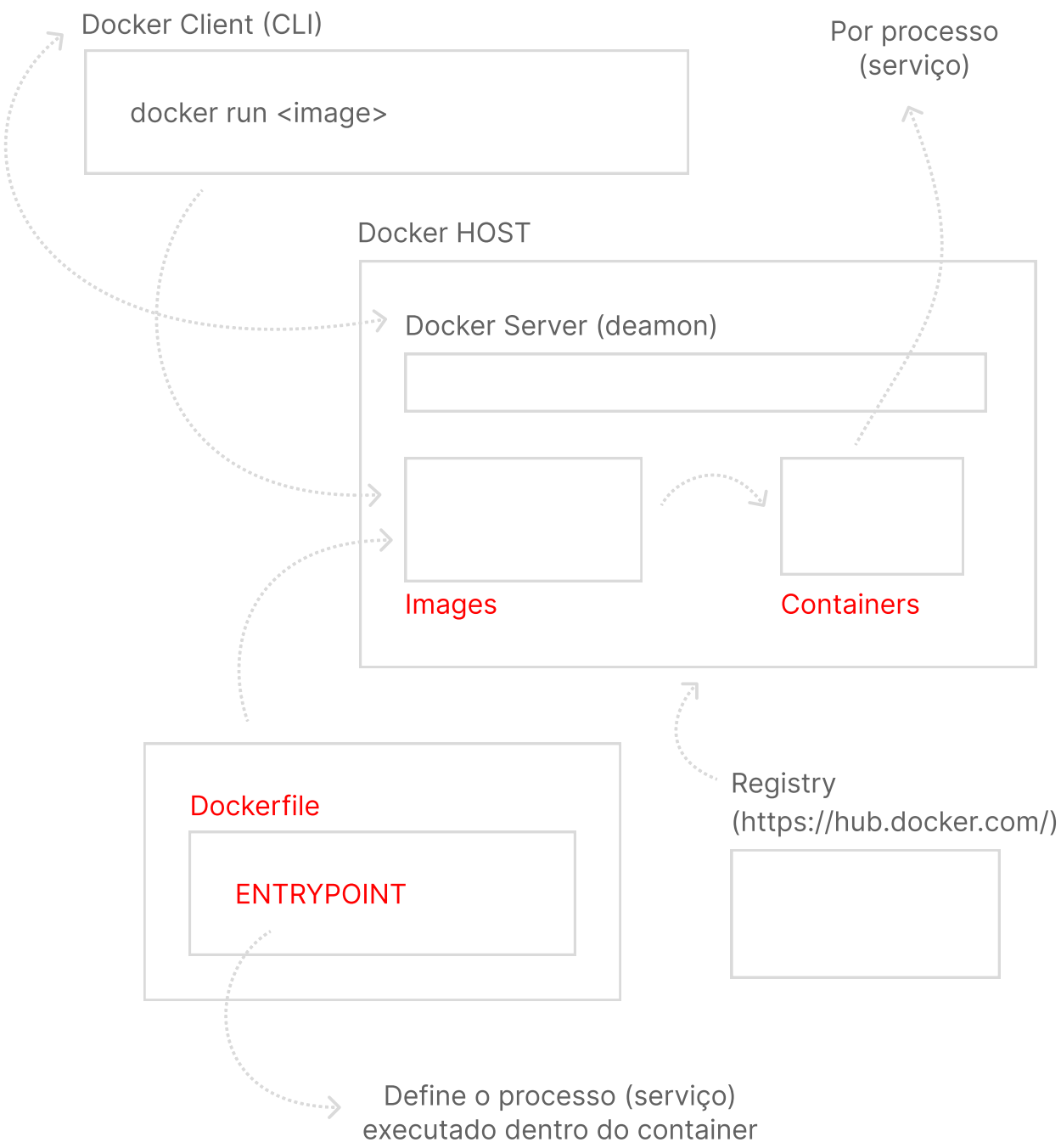
Dockerfile - COPY



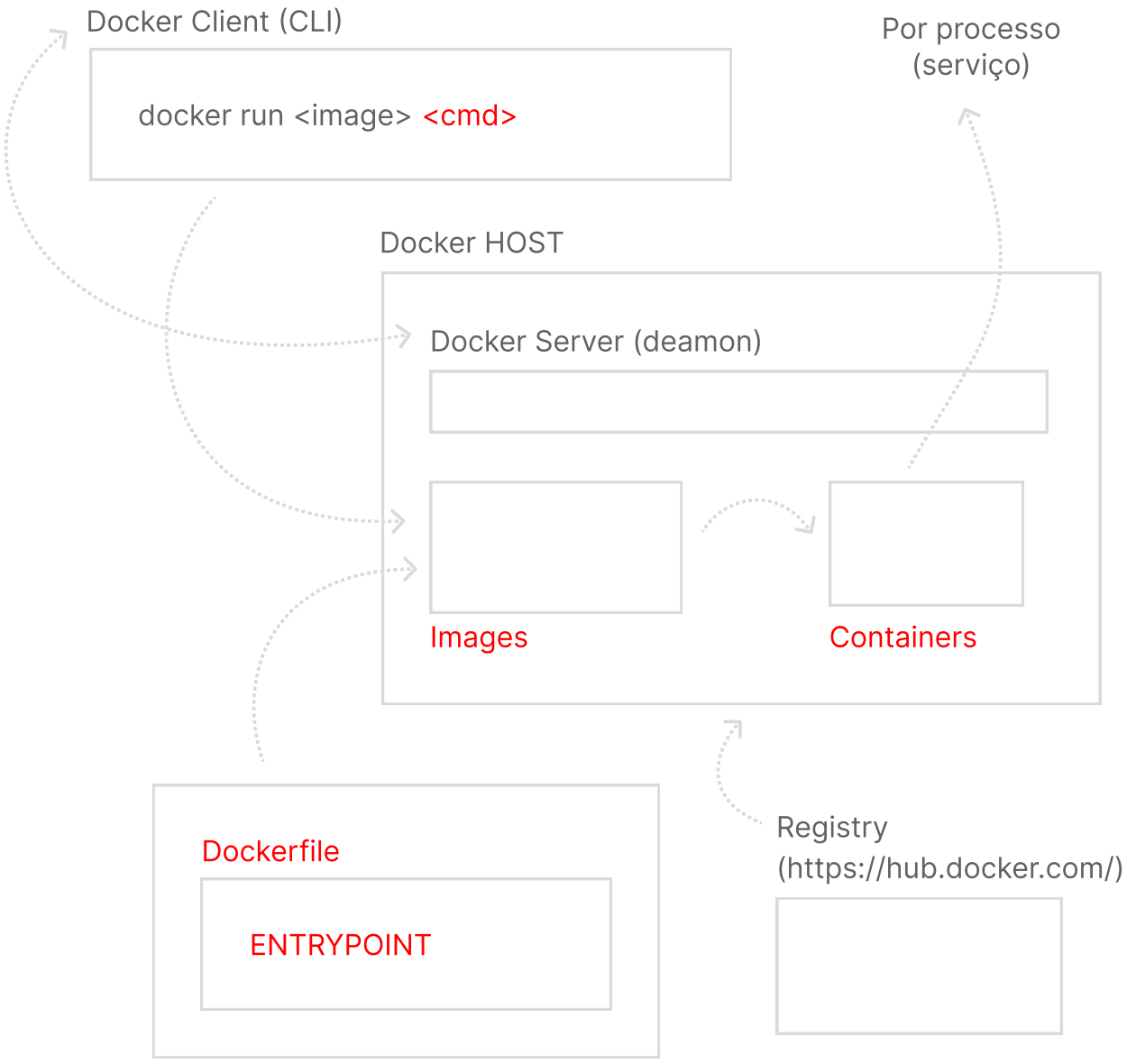
Dockerfile - COPY múltiplos arquivos



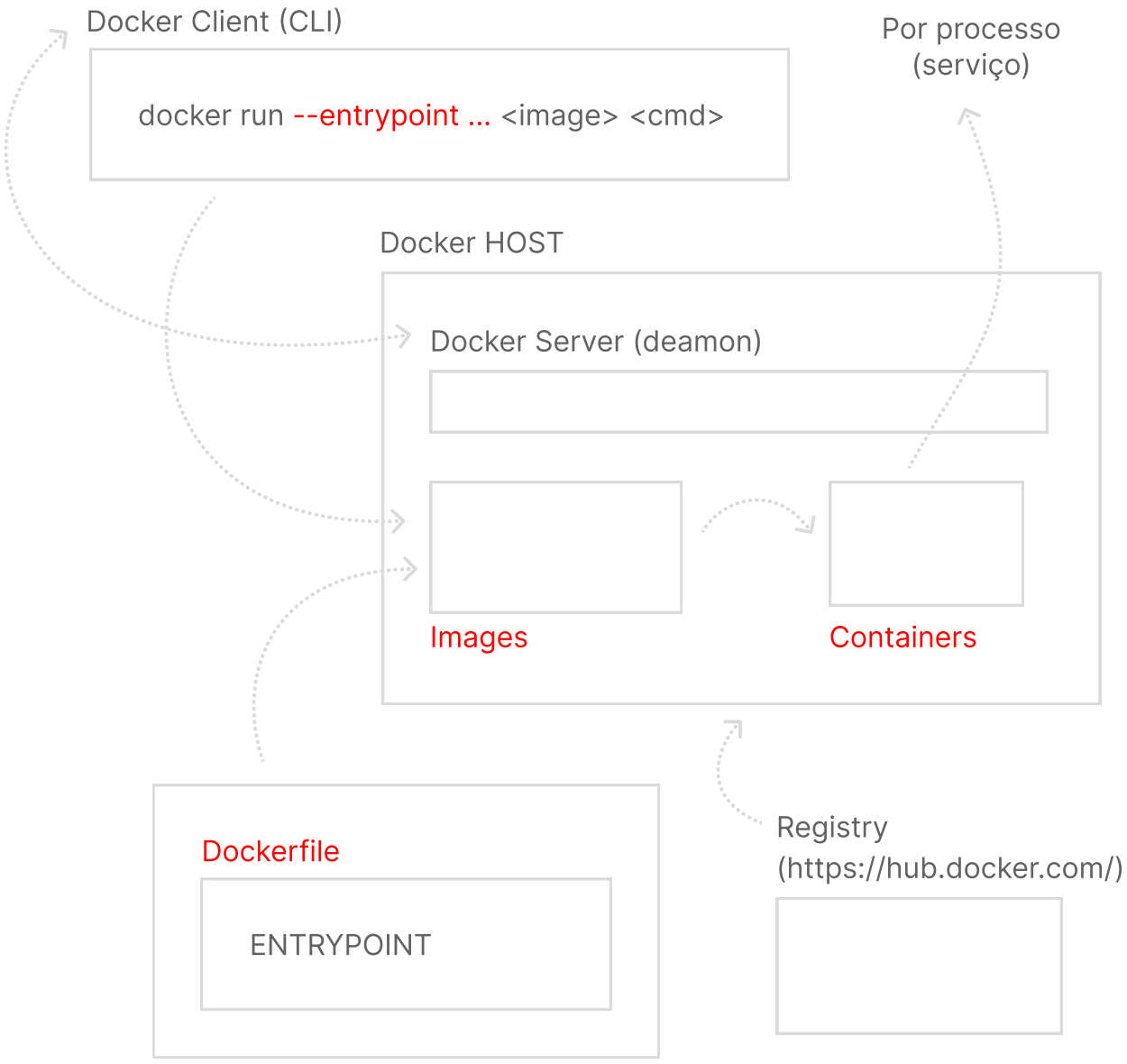
Dockerfile - ENTRYPOINT



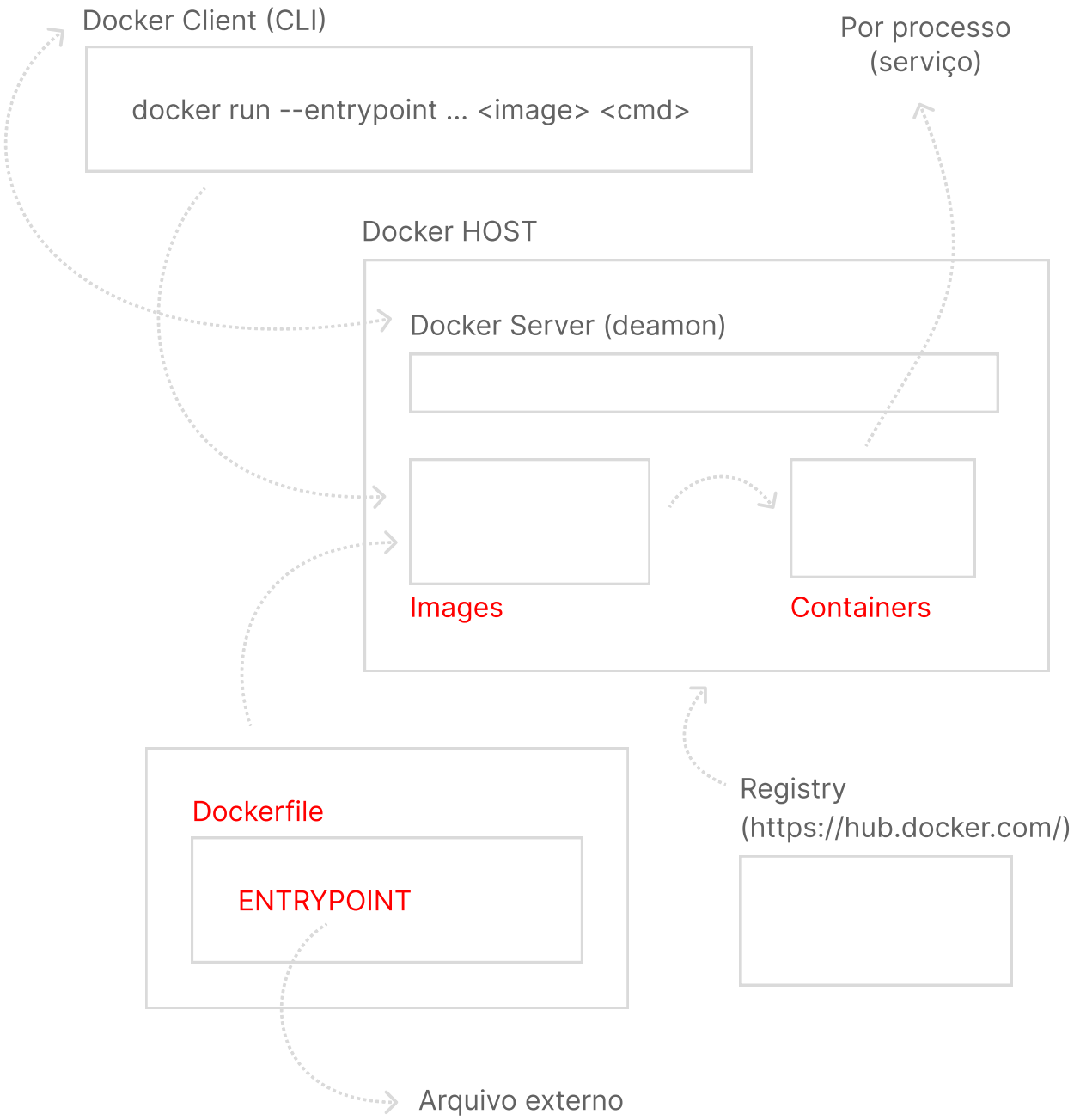
Dockerfile - ENTRYPOINT combinado com CMD



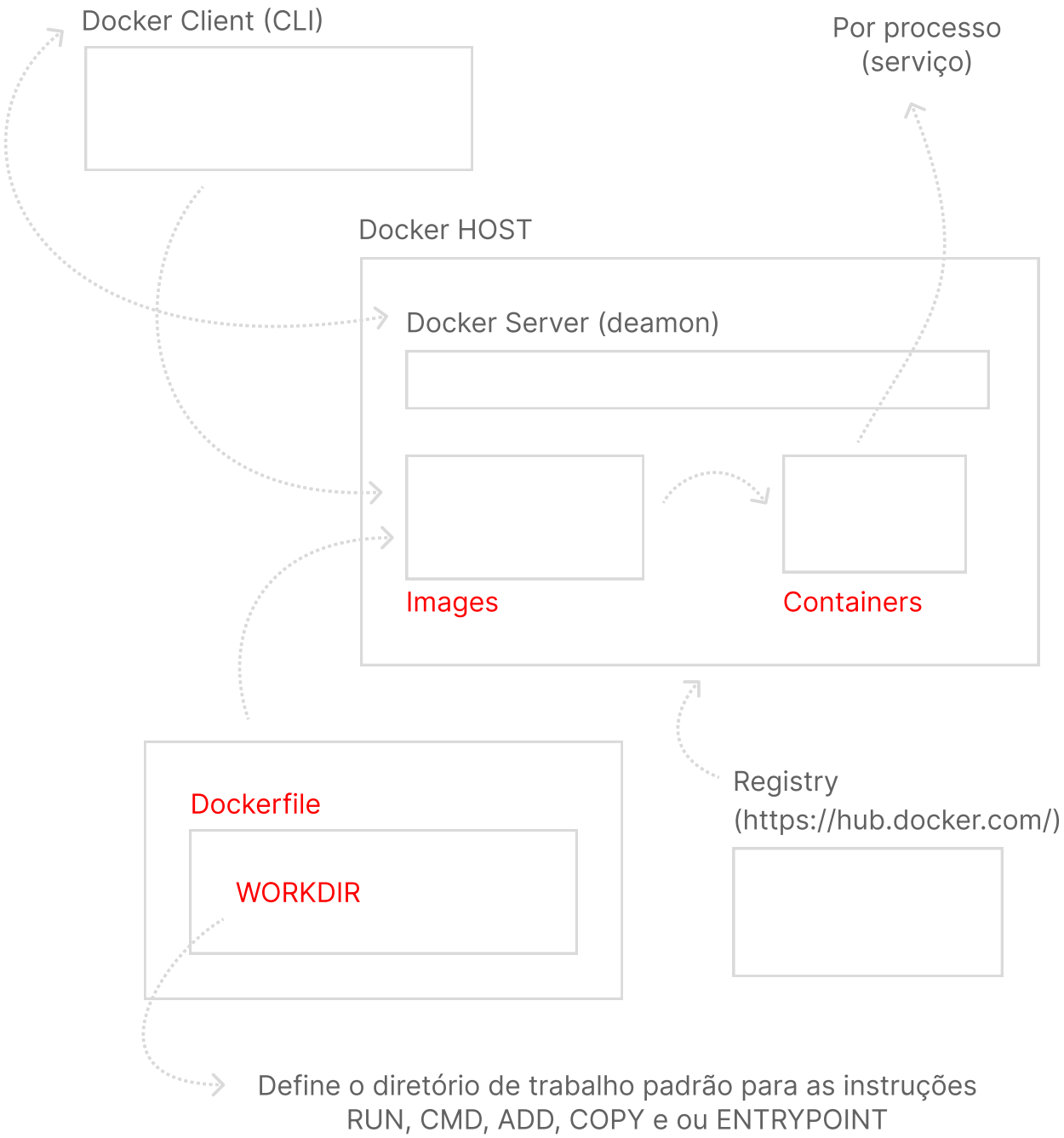
Dockerfile - ENTRYPOINT via terminal



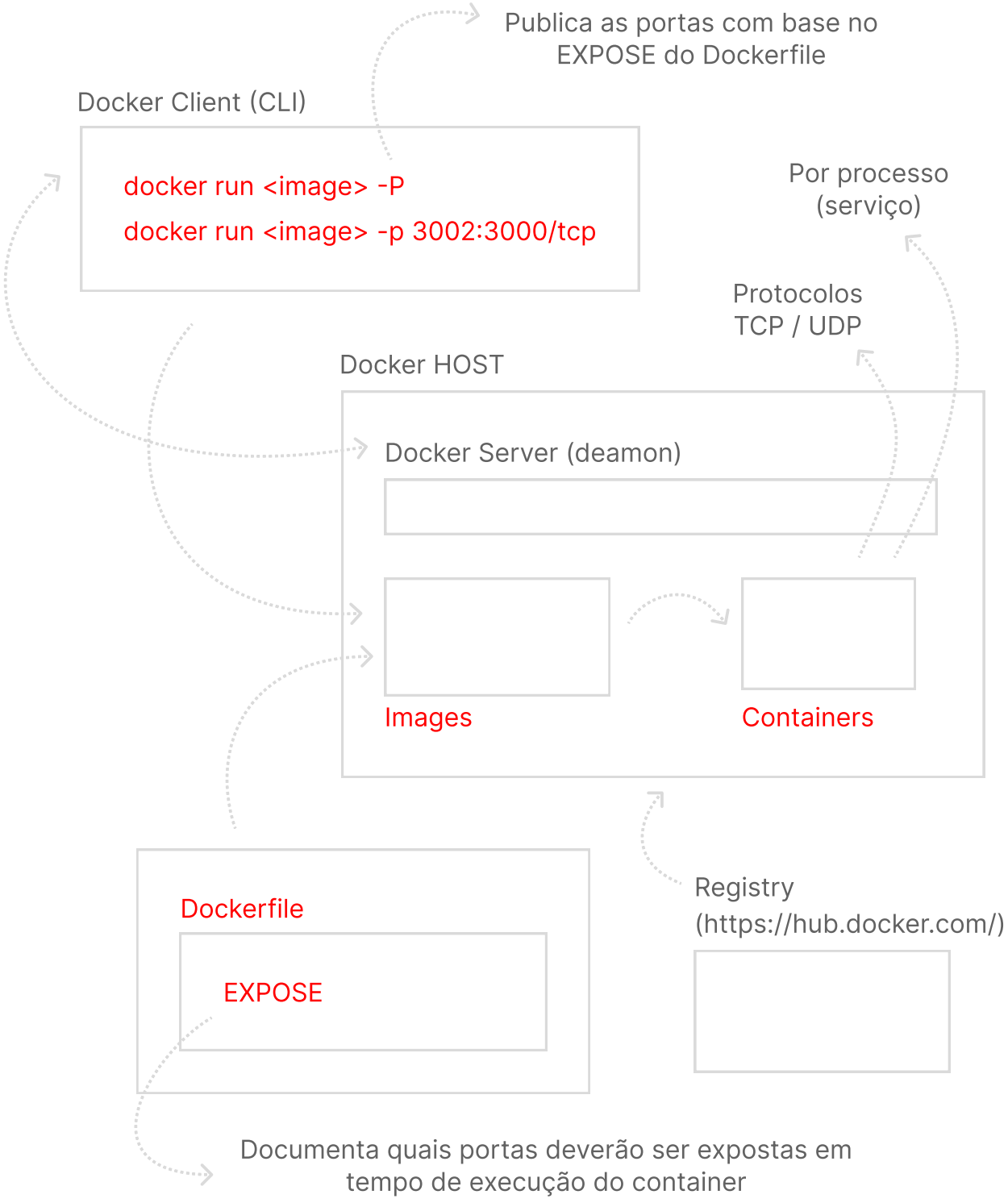
Dockerfile - ENTRYPOINT via arquivo externo



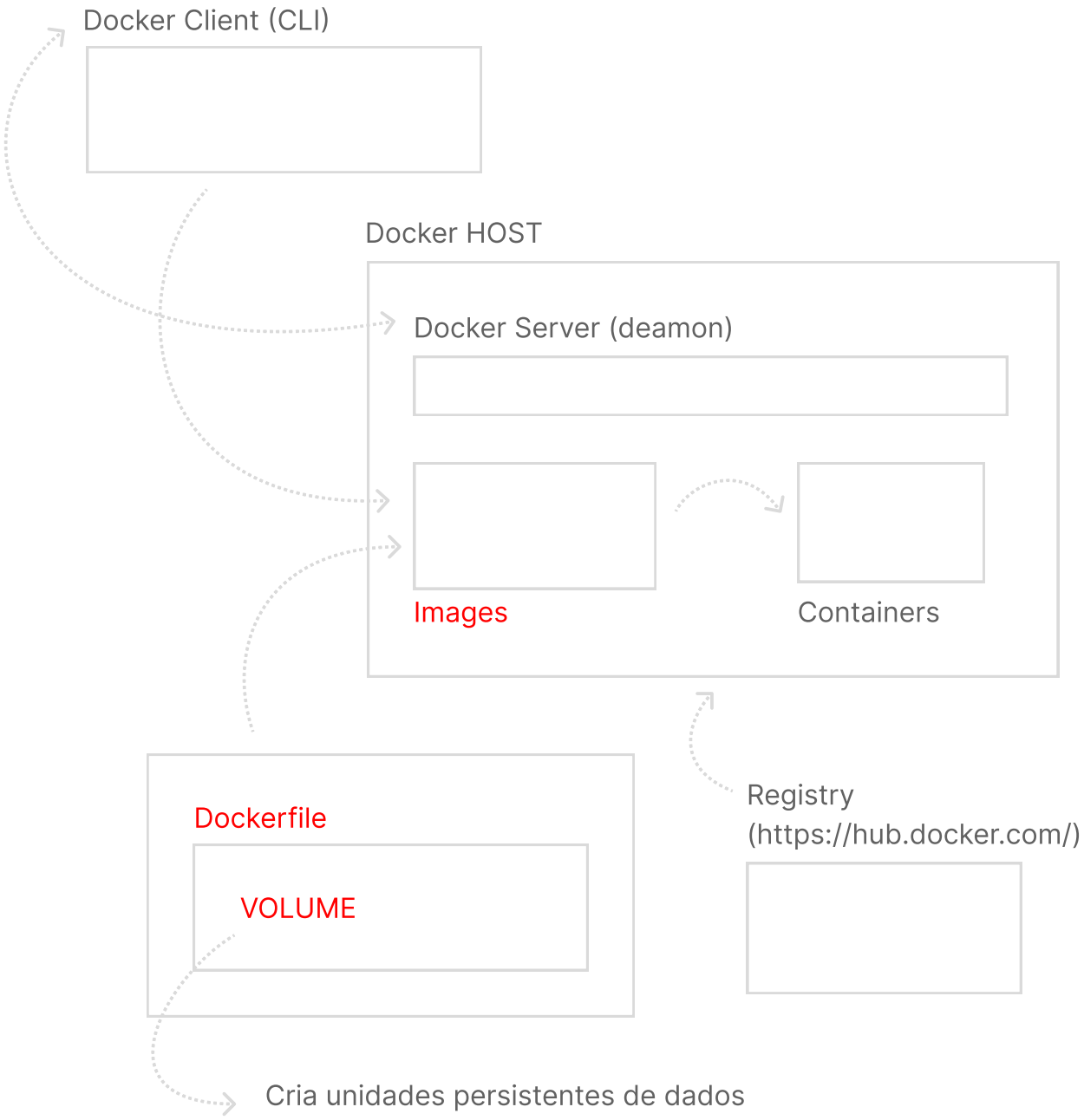
Dockerfile - WORKDIR



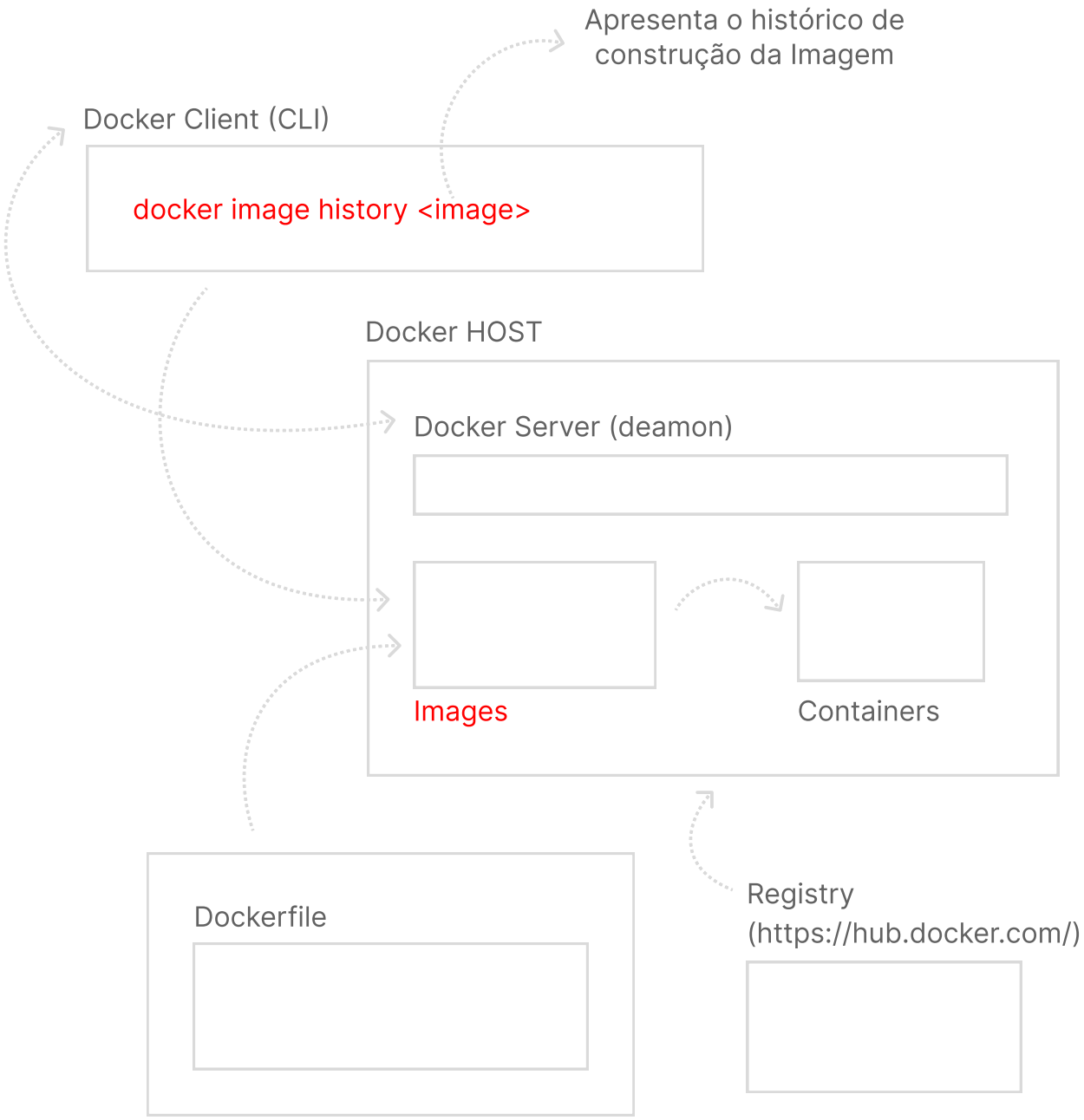
Dockerfile - EXPOSE



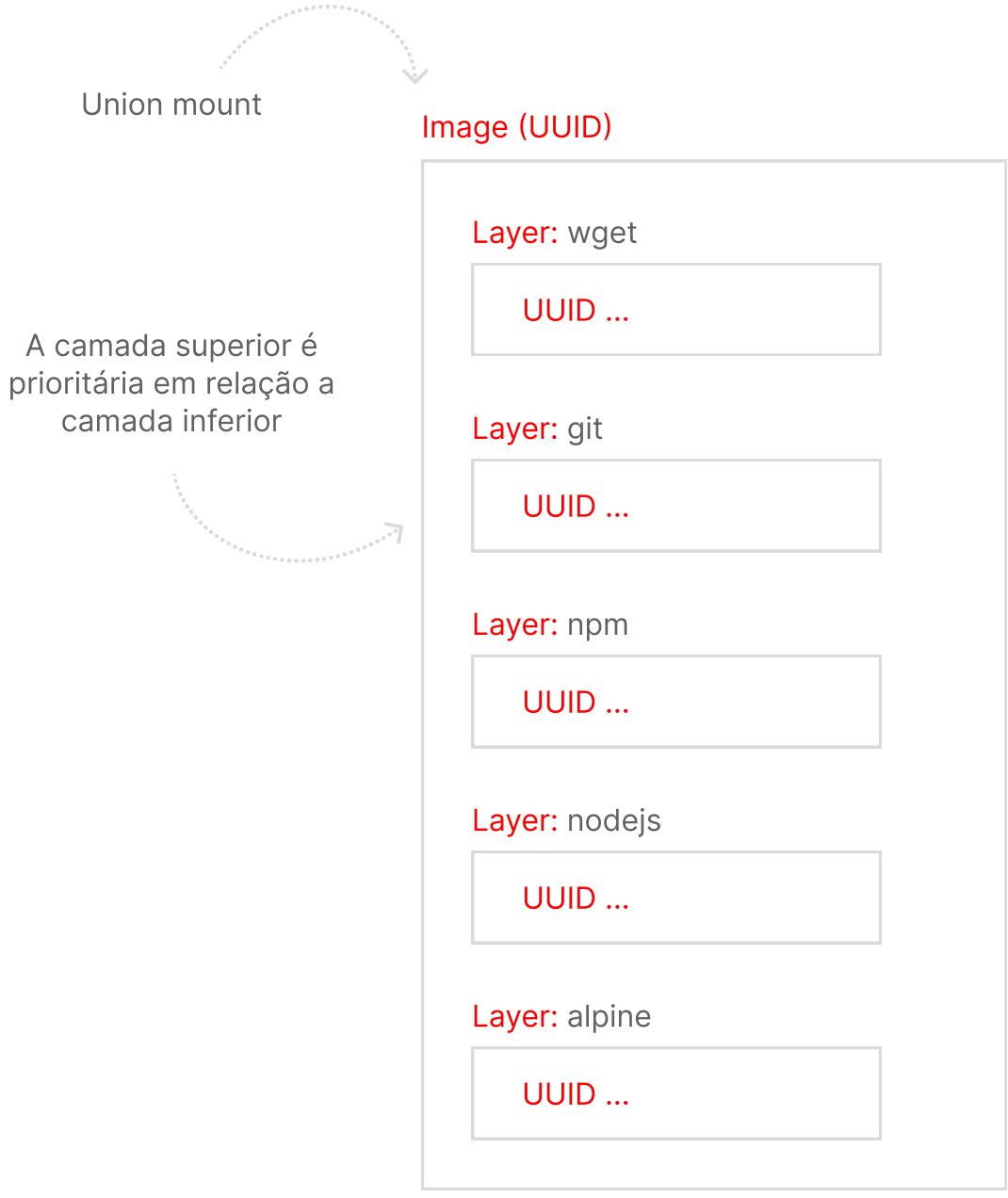
Dockerfile - VOLUME



Consultando o histórico de construção da imagem



Camadas e cache de camadas

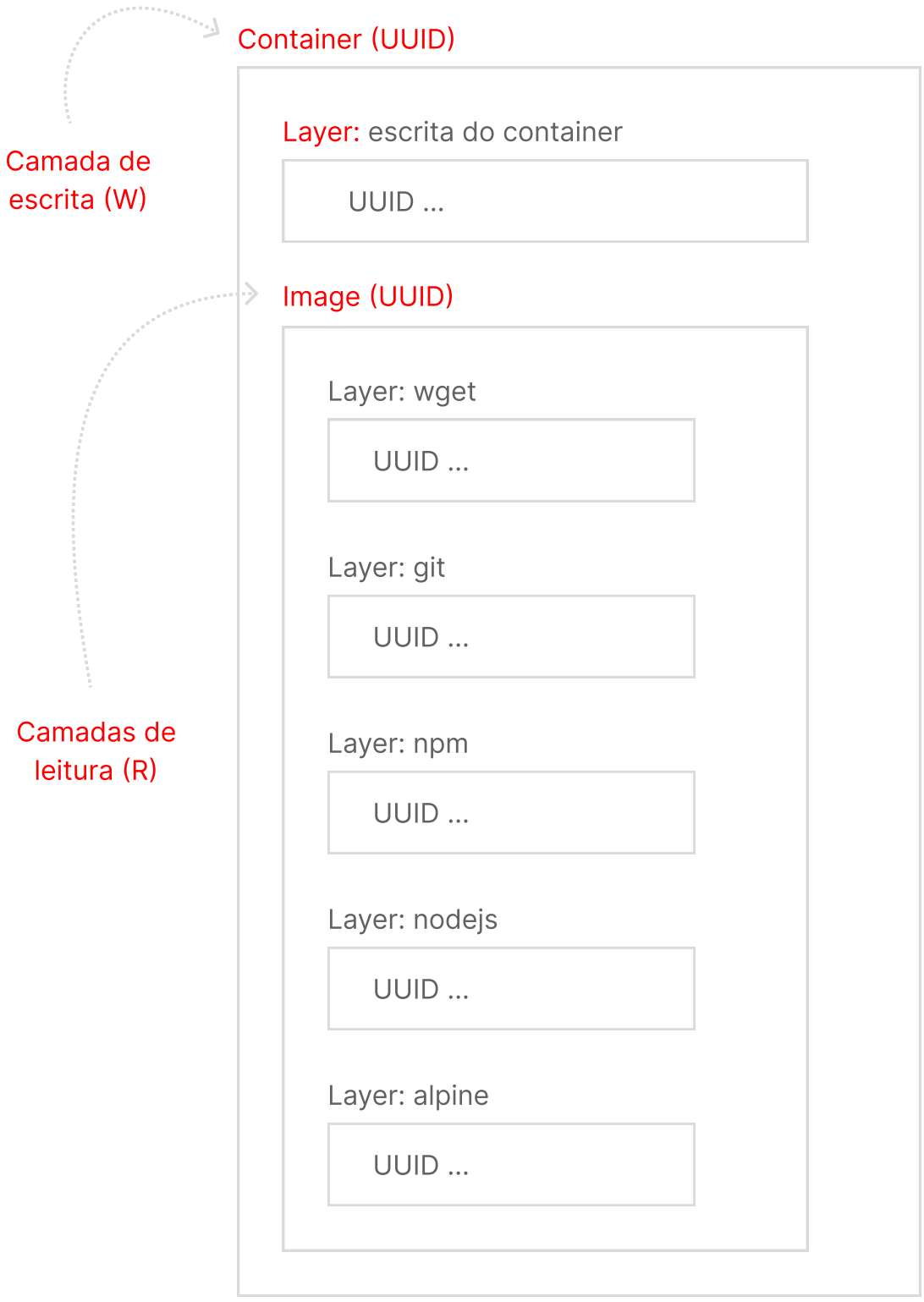


Dockerfile e as instruções que geram camadas

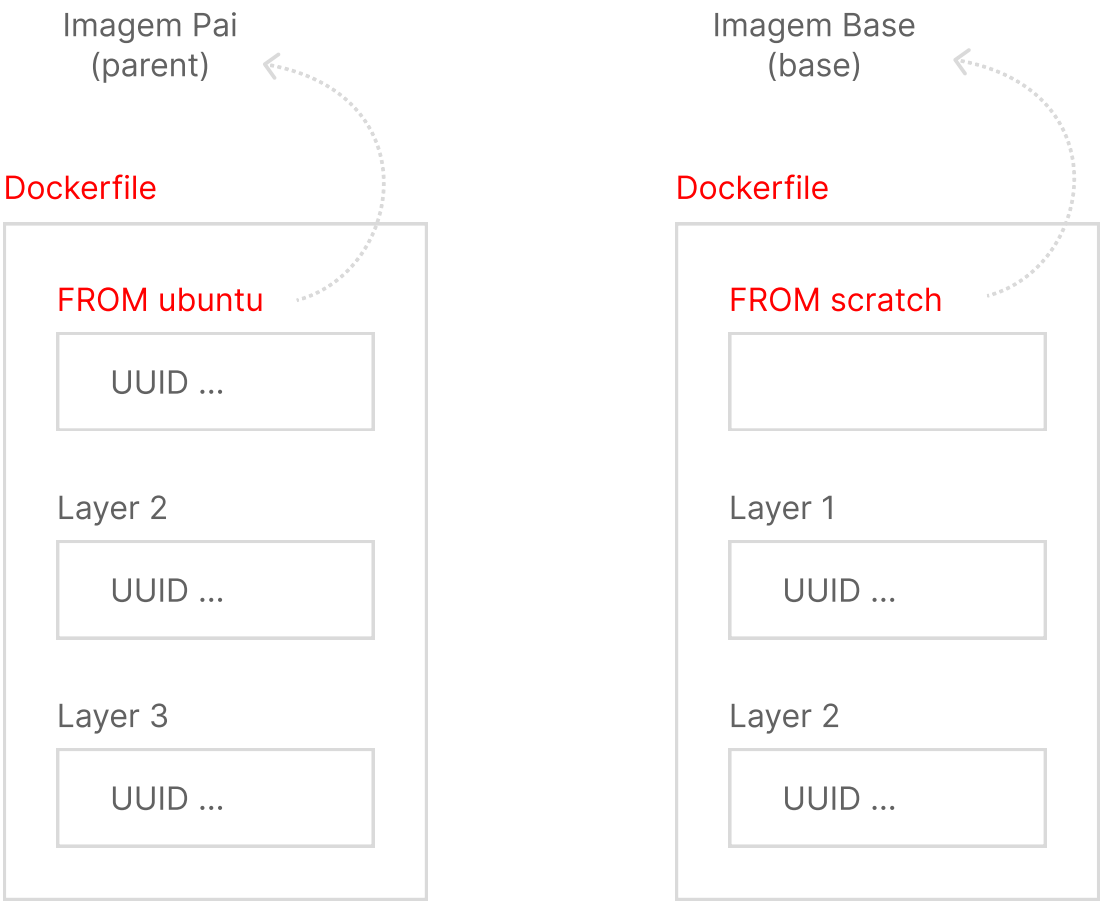
Geram camadas	Não geram camadas
FROM	LABEL
RUN	ENV
ADD	ARG
COPY	USER
WORKDIR	CMD
VOLUME	ENTRYPOINT
	EXPOSE

Dockerfile e a herança de instruções entre camadas

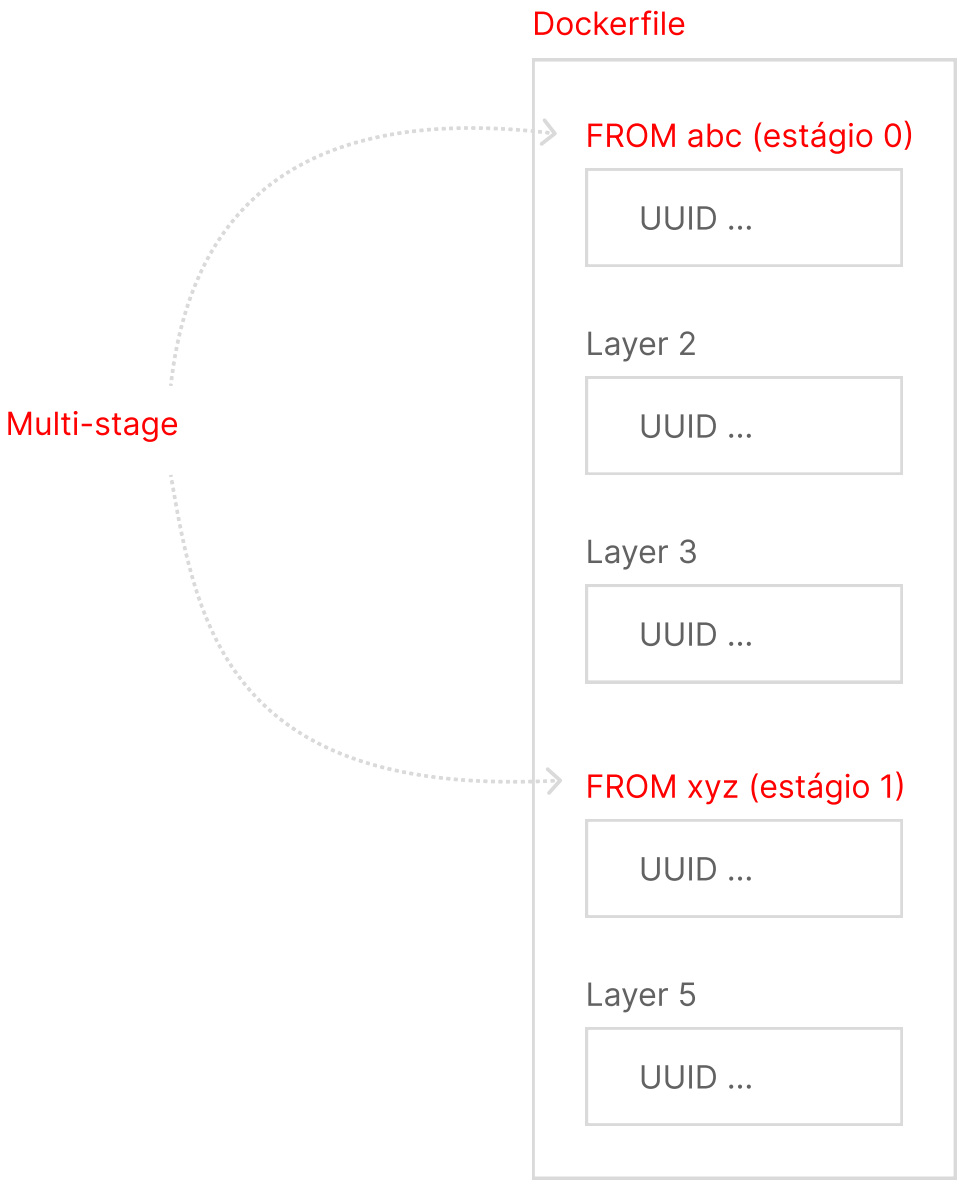
Camadas de leitura da Imagem e escrita do Container



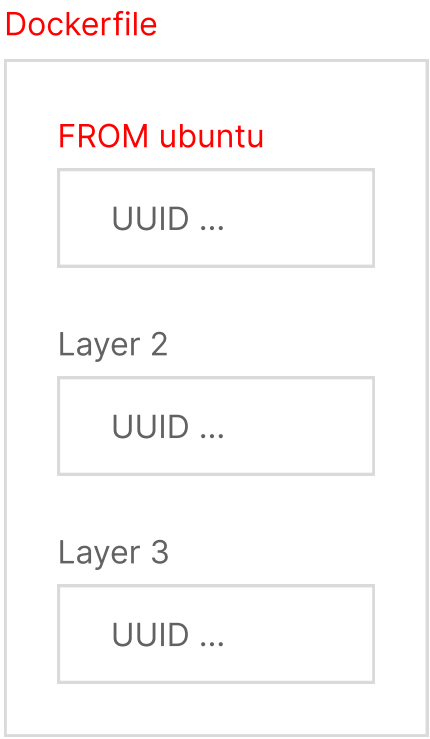
Criando Imagens com FROM scratch parte 1



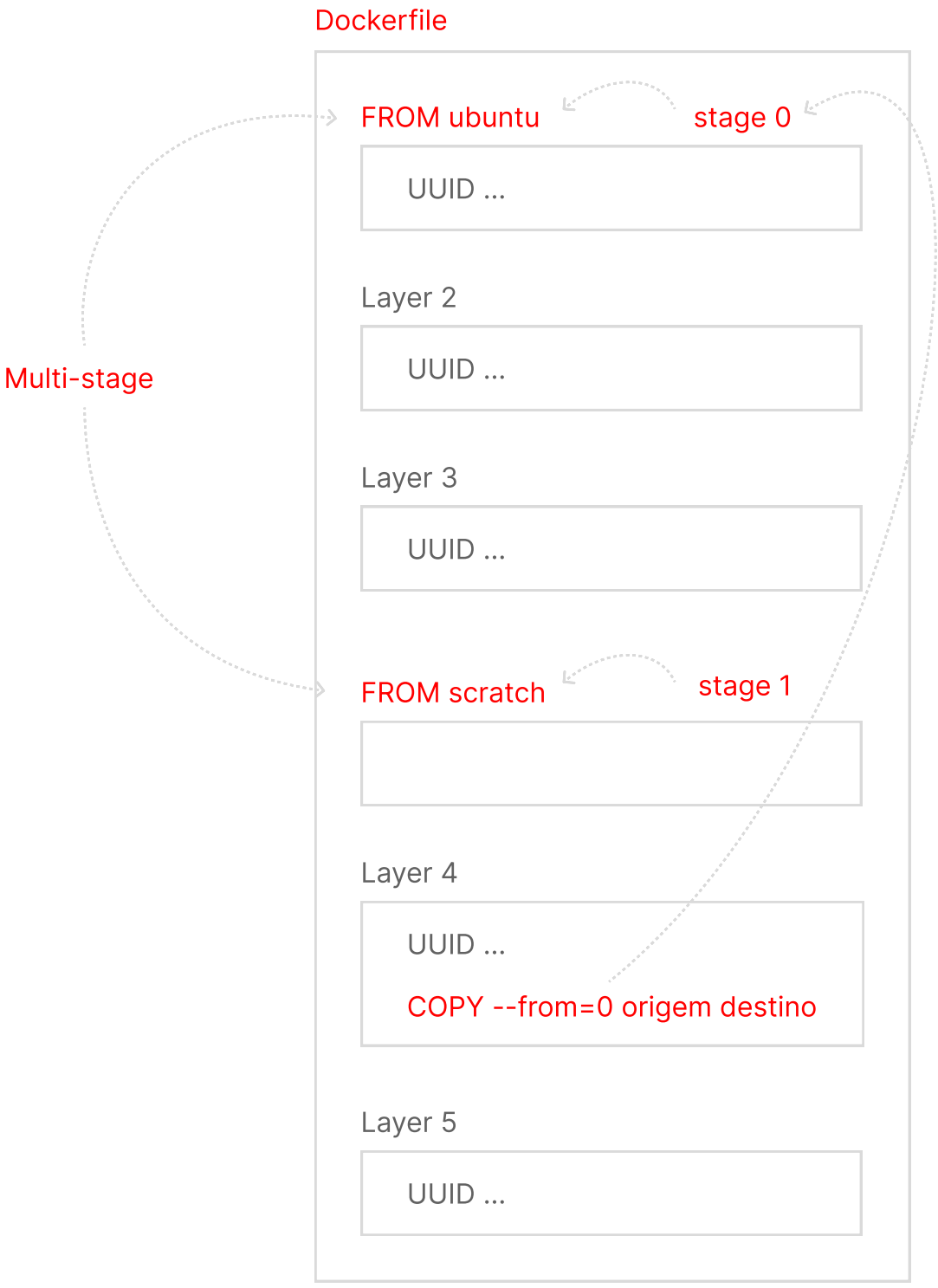
Multi-stage Build - Introdução



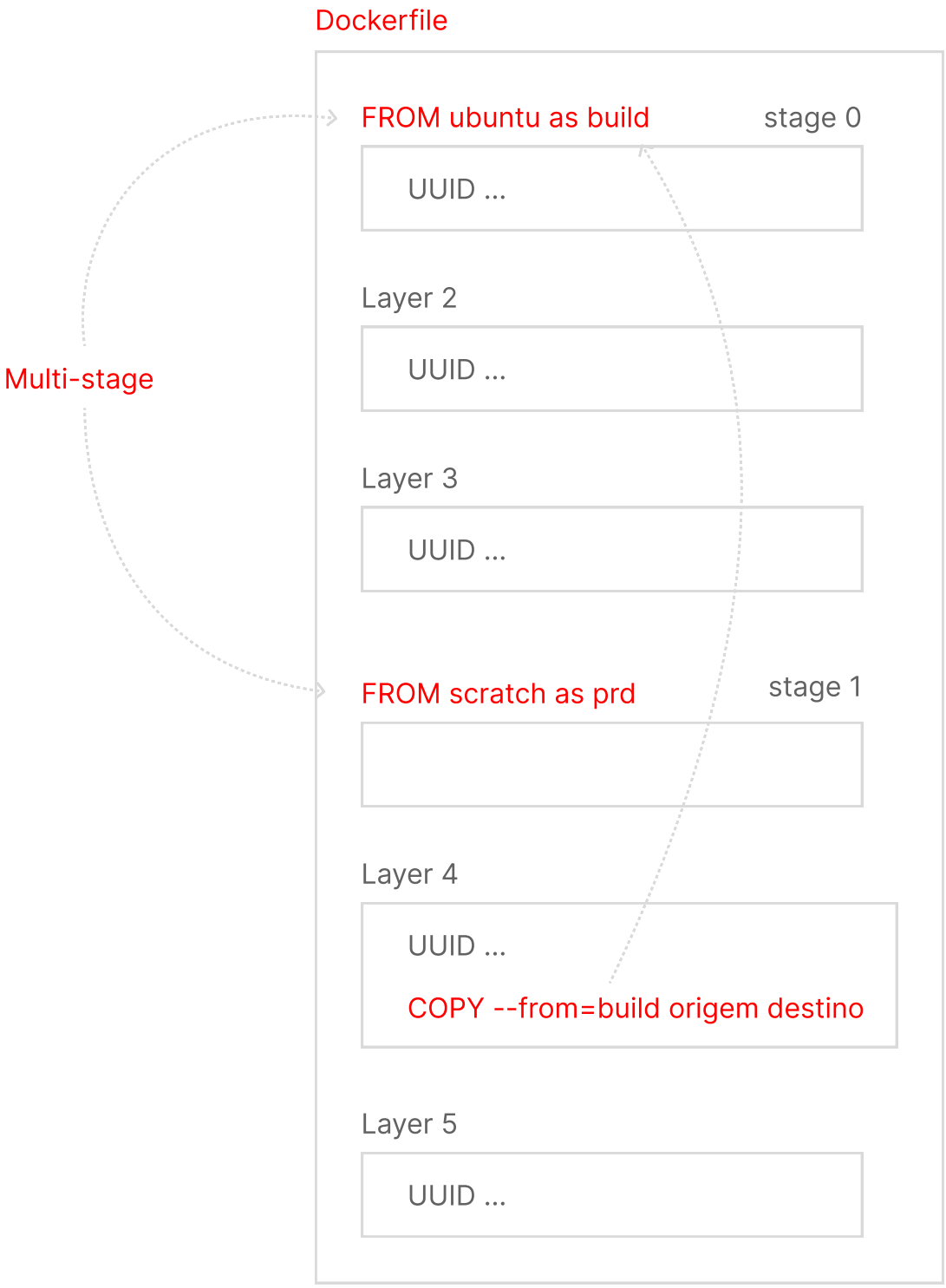
Multi-stage Build - Criando uma Imagem Multi-stage parte 1



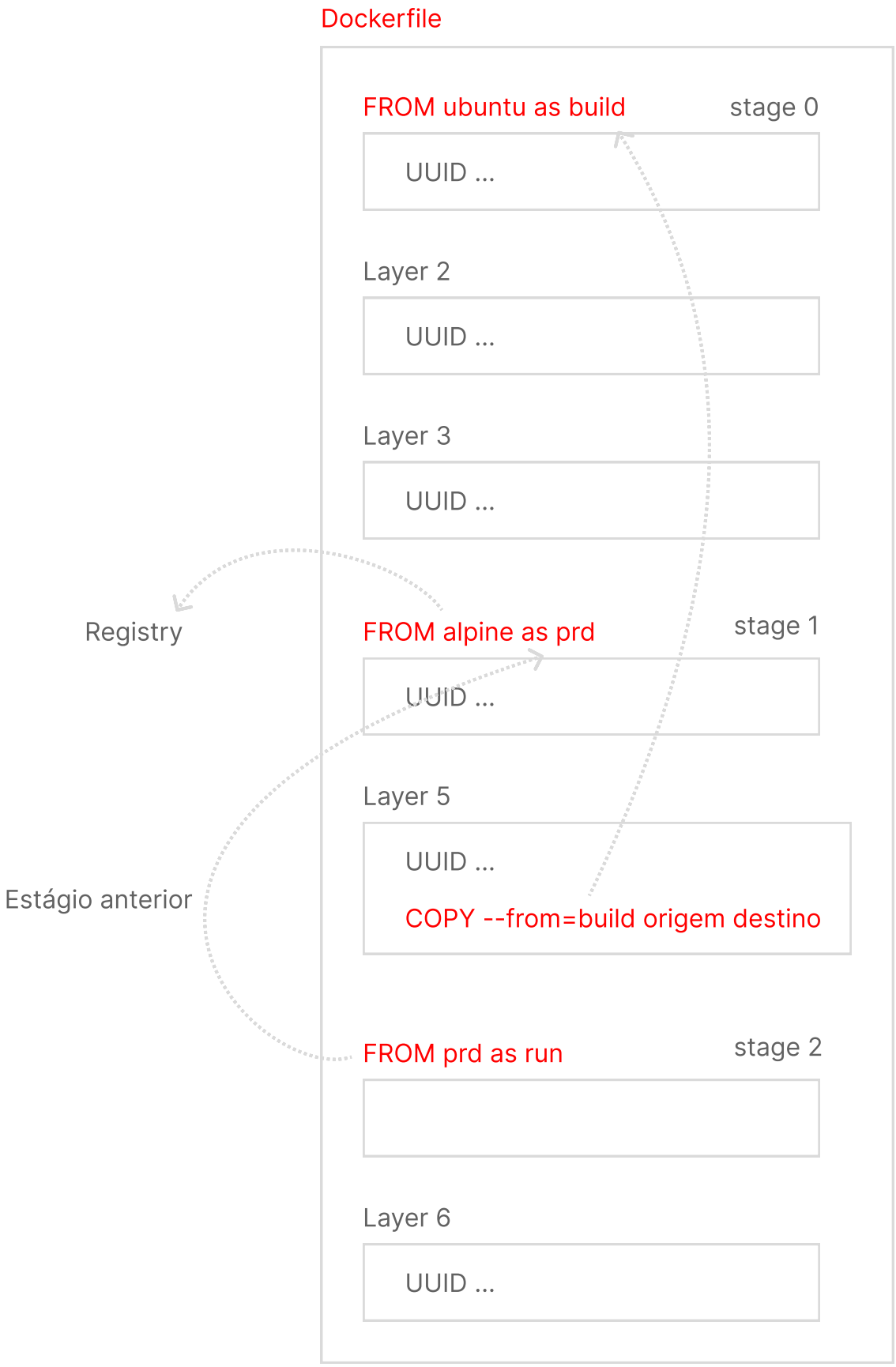
Multi-stage Build - Criando uma Imagem Multi-stage parte 2



Multi-stage Build - Nomeando estágios

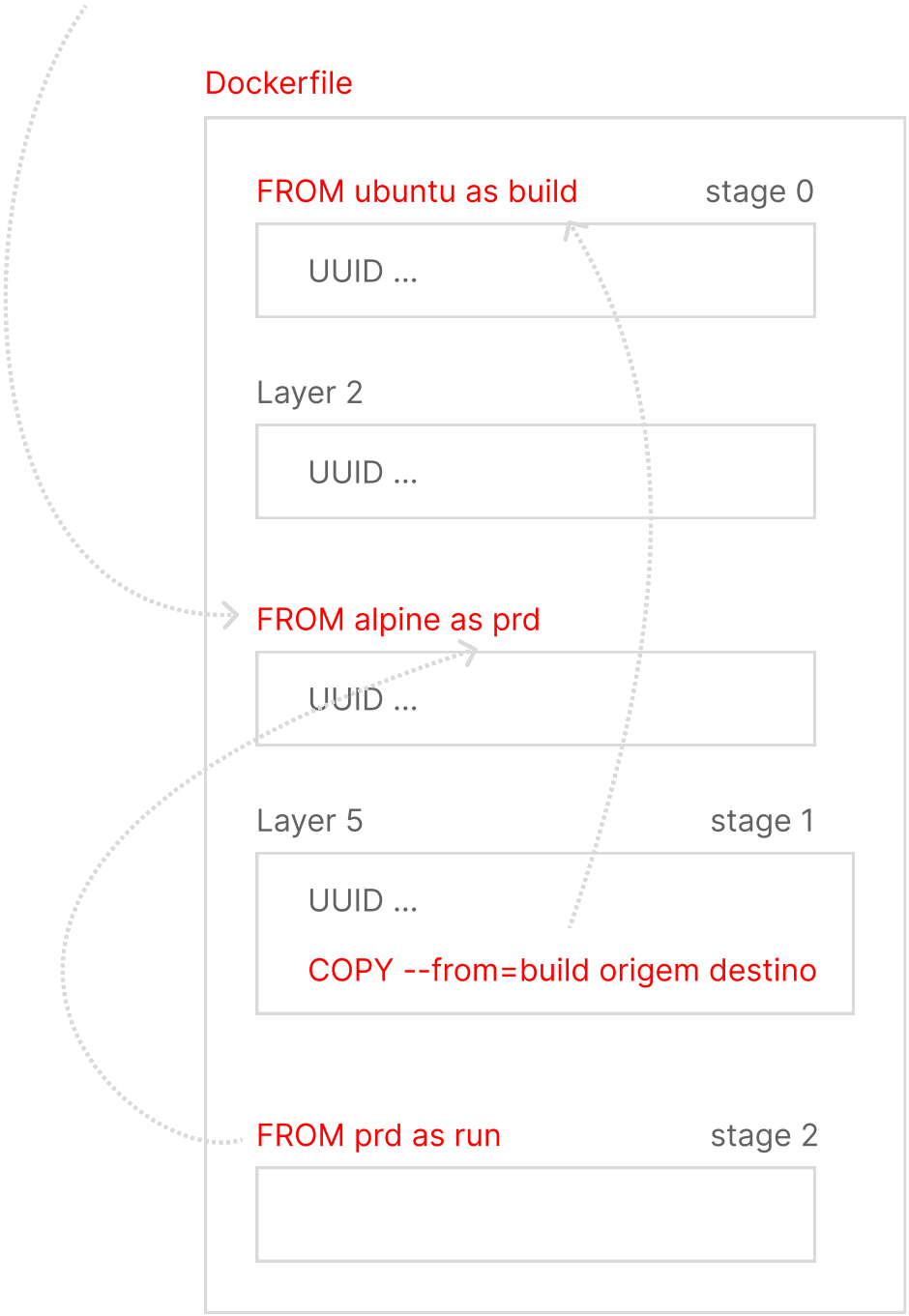


Multi-stage Build - Estágios baseados em estágios anteriores ou imagens

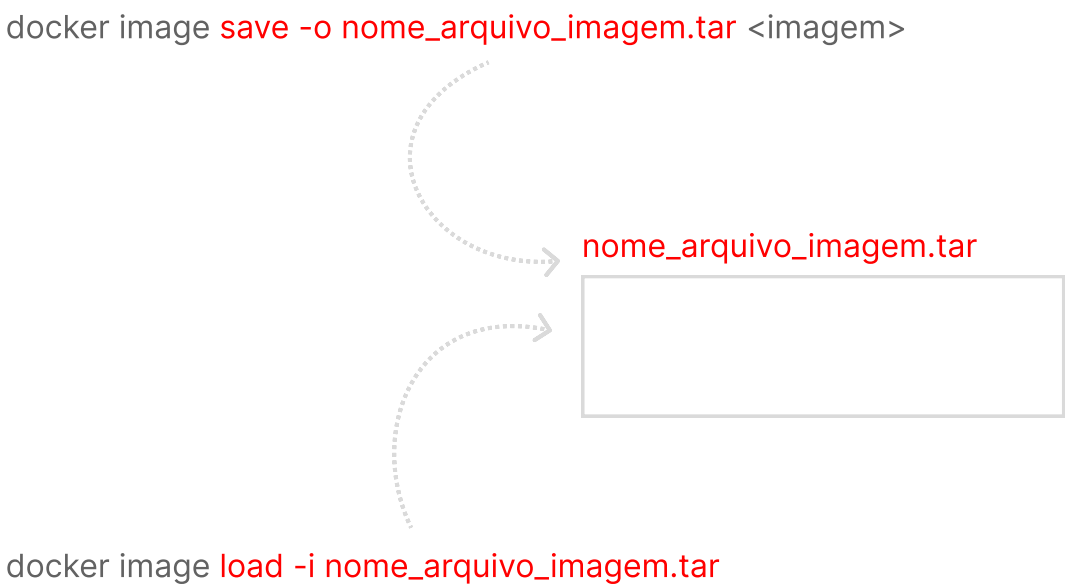


Multi-stage Build - Build até estágio específico

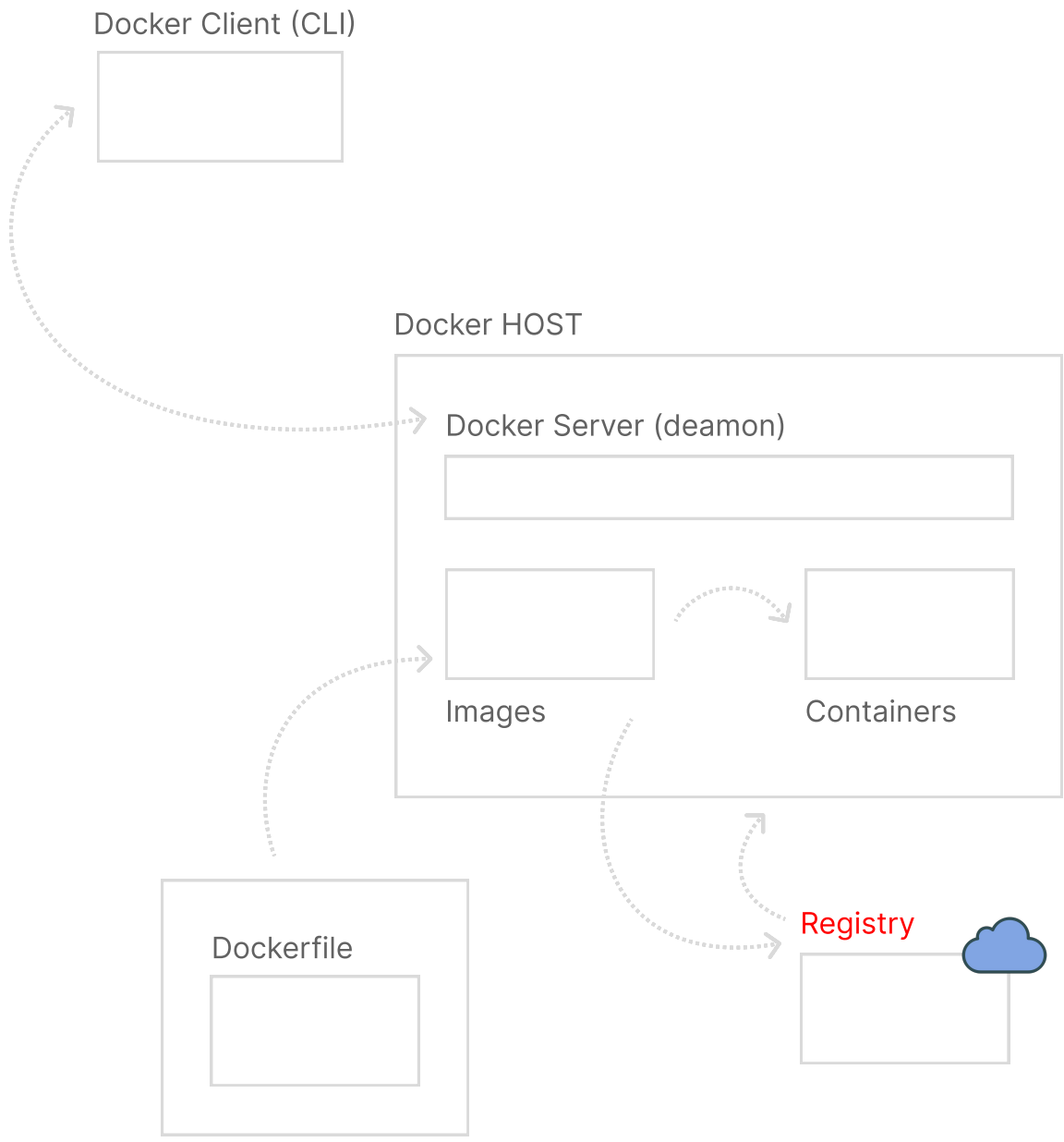
docker build --target prd -t <imagem> .



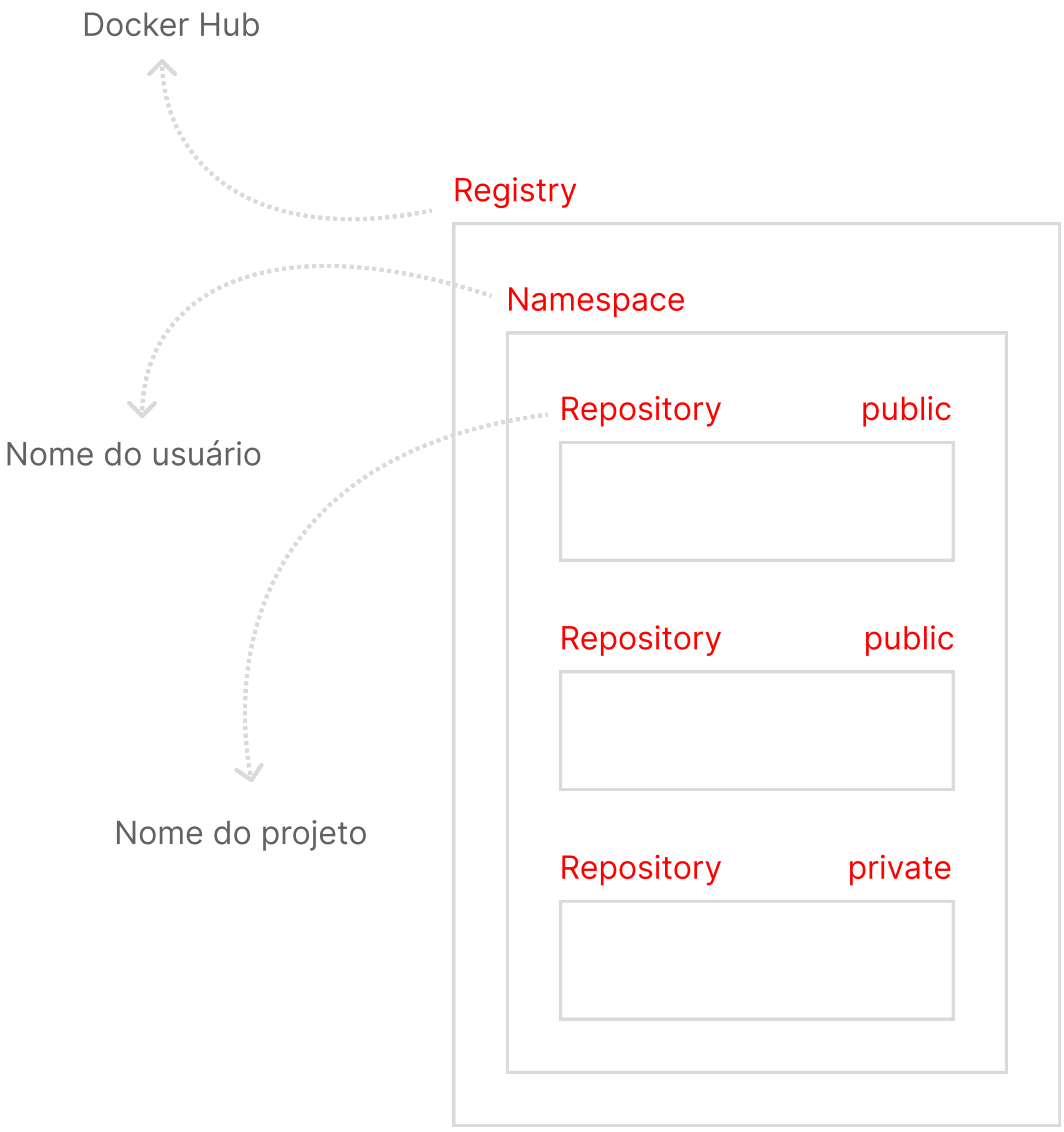
Exportar, transferir e importar Imagens manulamente



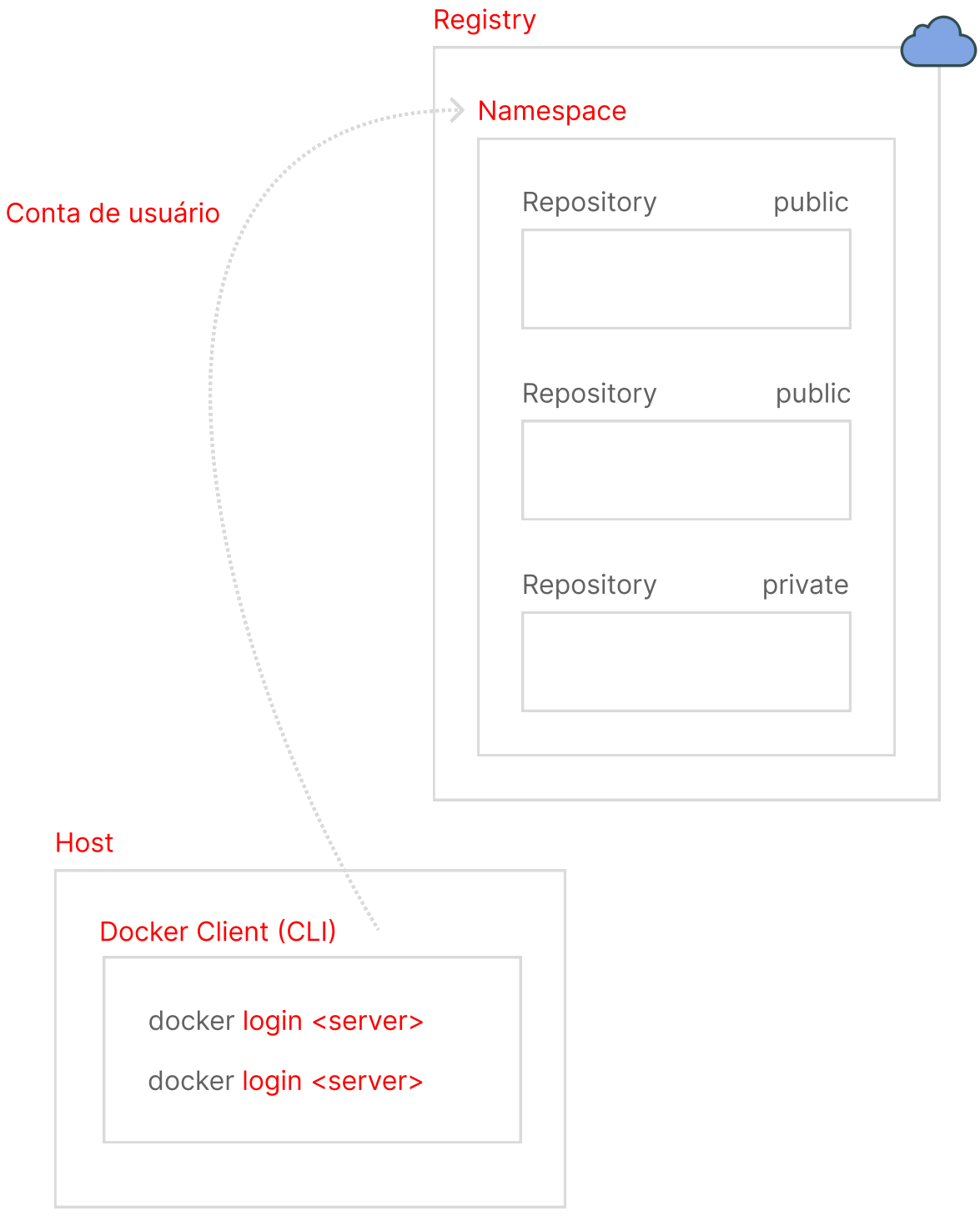
Criando uma conta no Docker Hub



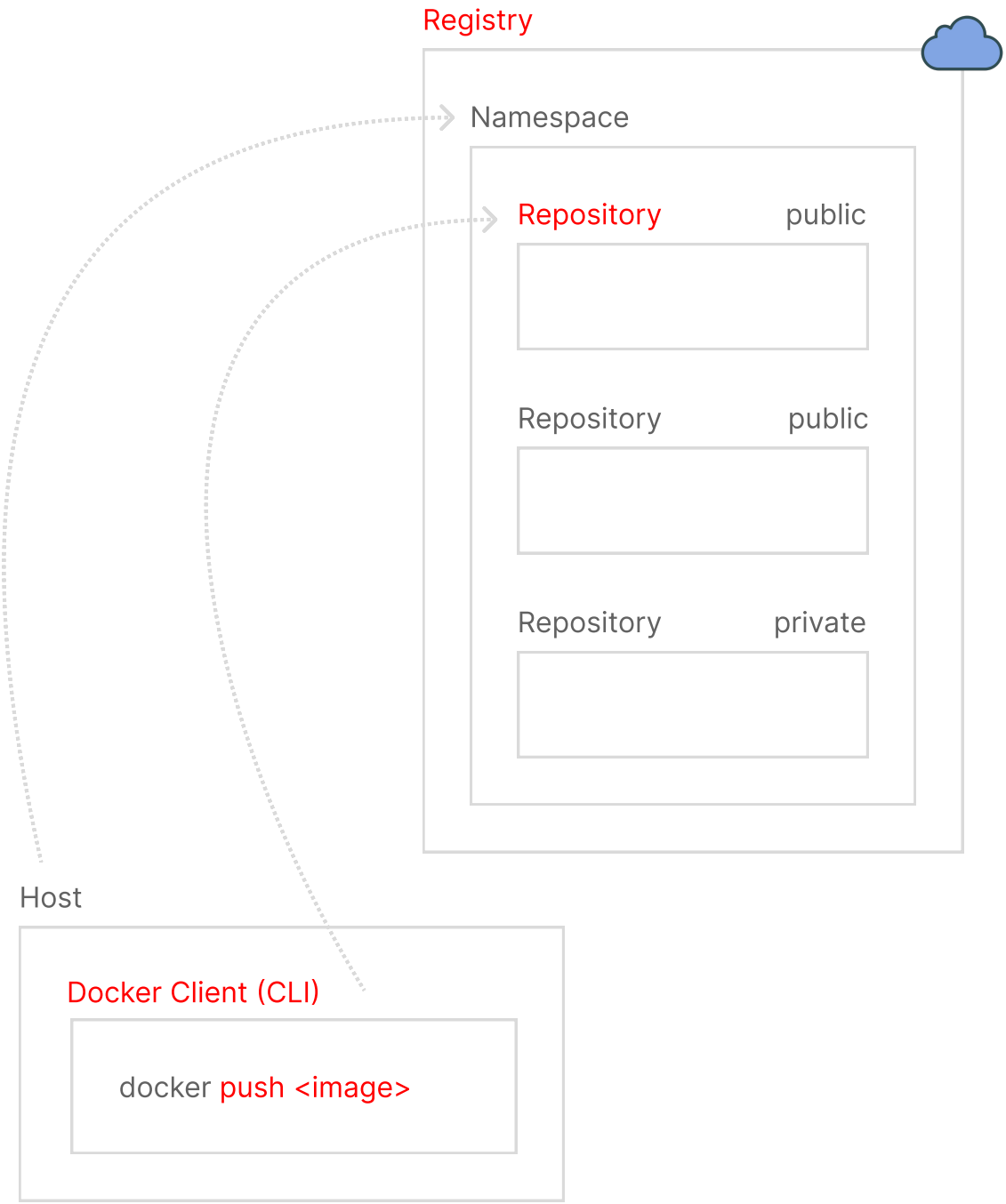
Criando um repositório (público e privado)



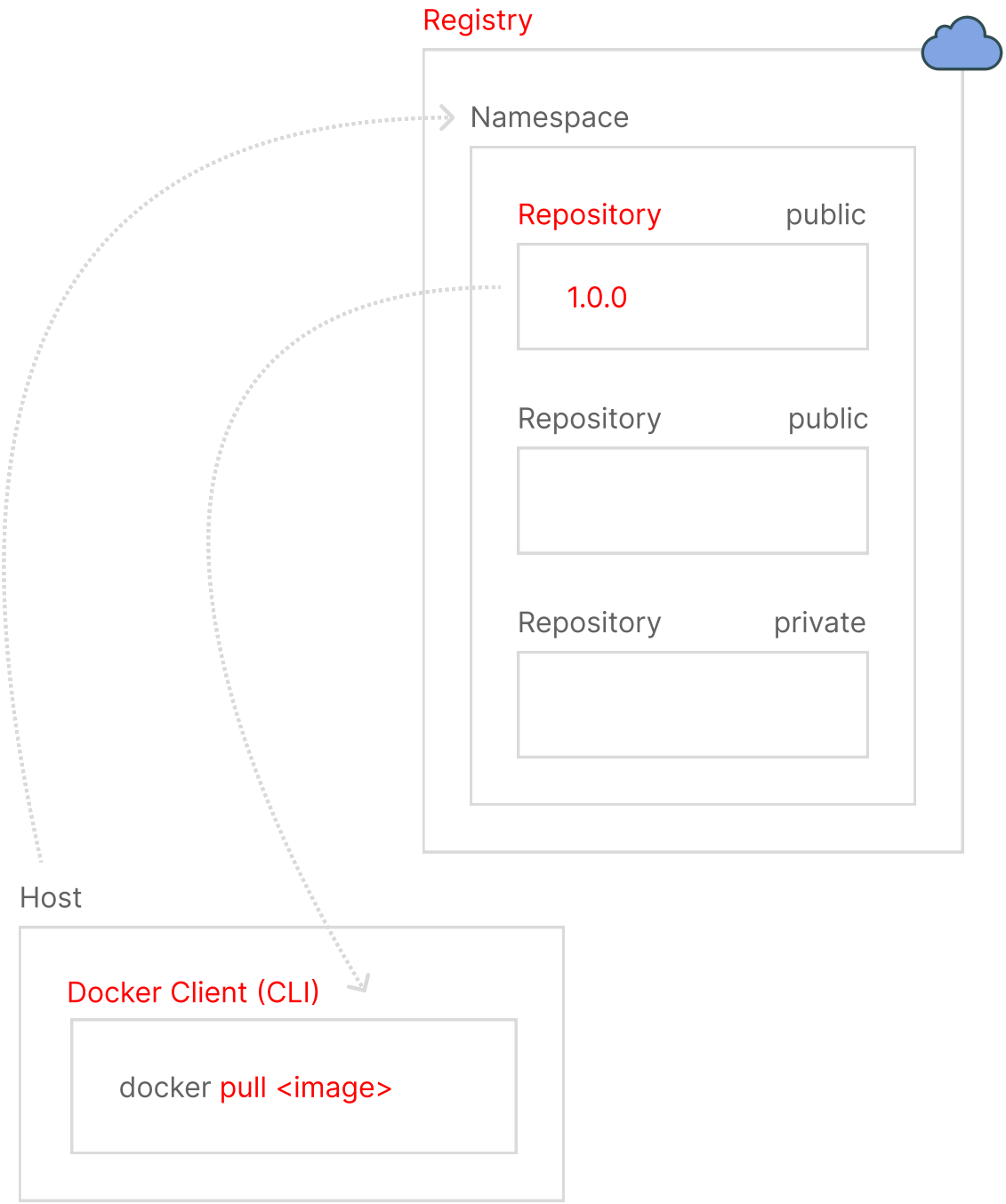
Login e logout no Docker Hub via terminal



Enviando imagem para o registry (push)

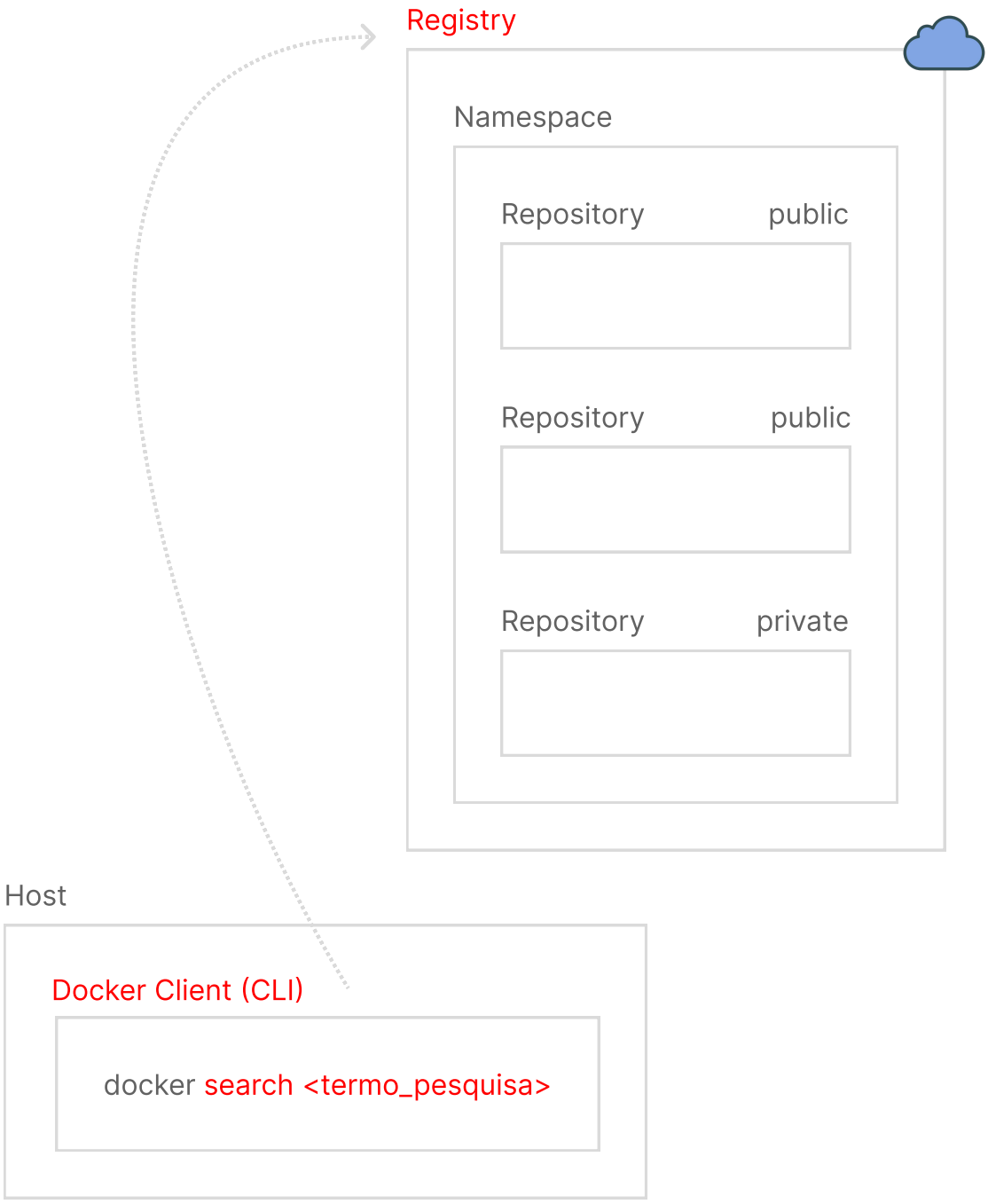


Baixando imagem do registry (pull)

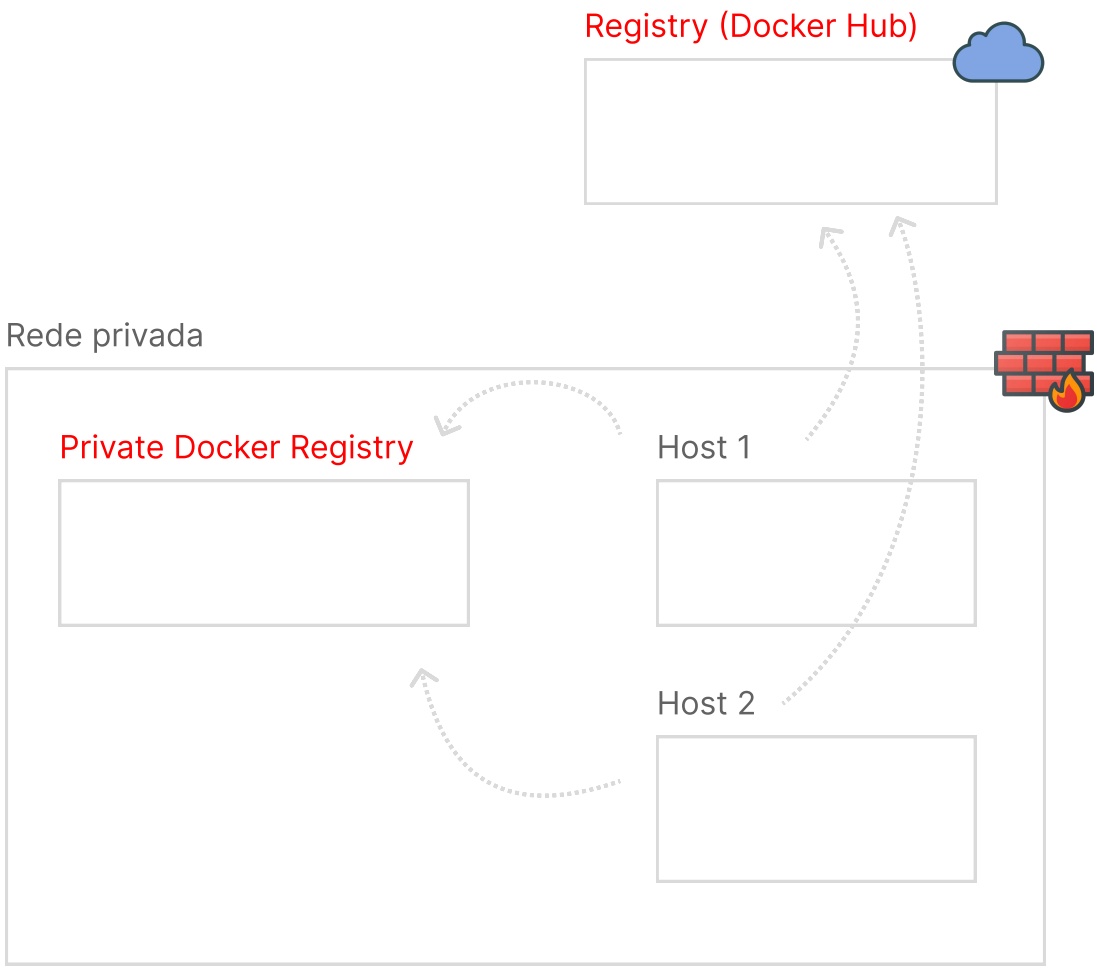


Removendo imagem do registry

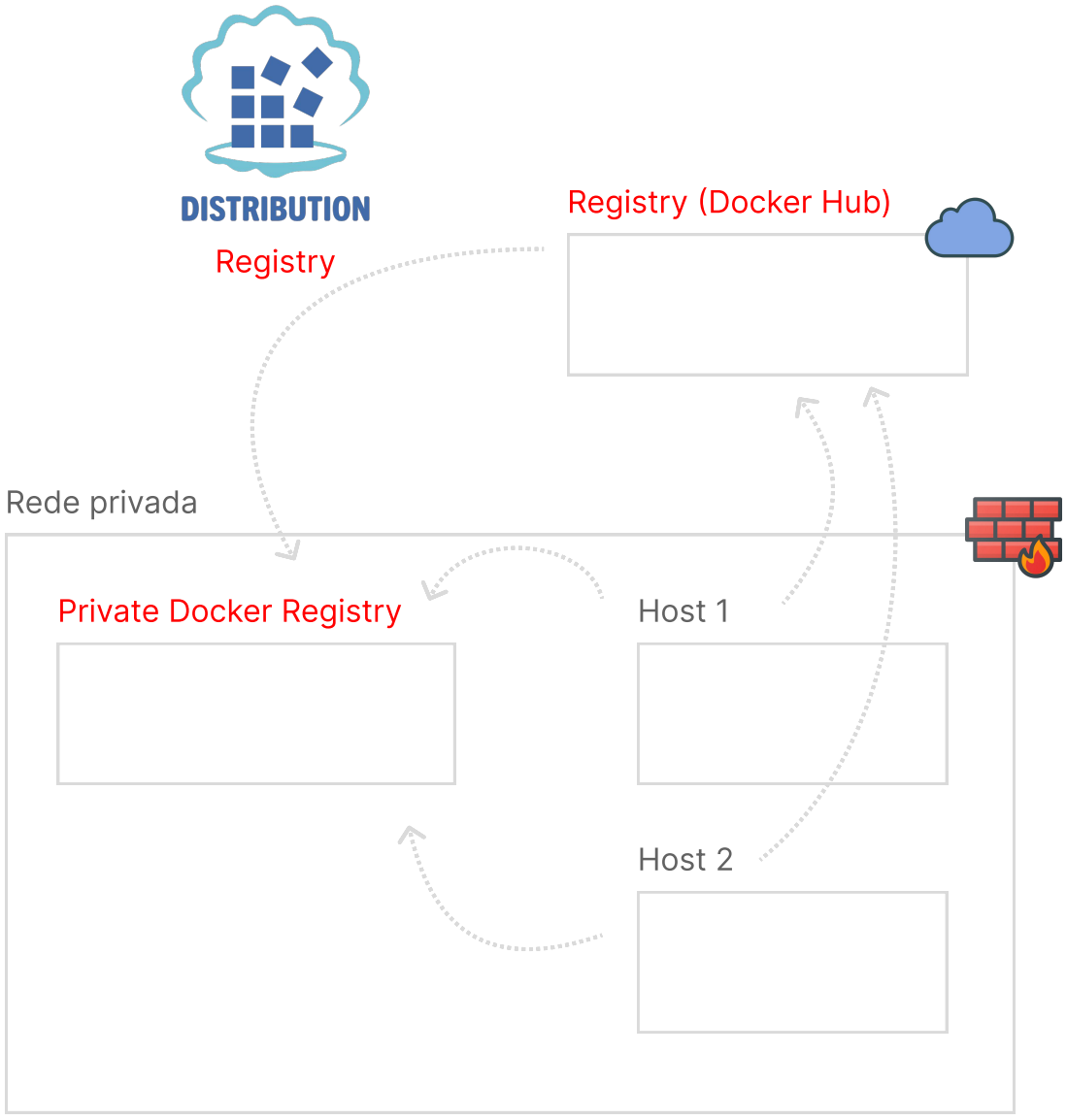
Pesquisando por imagens no registry via CLI



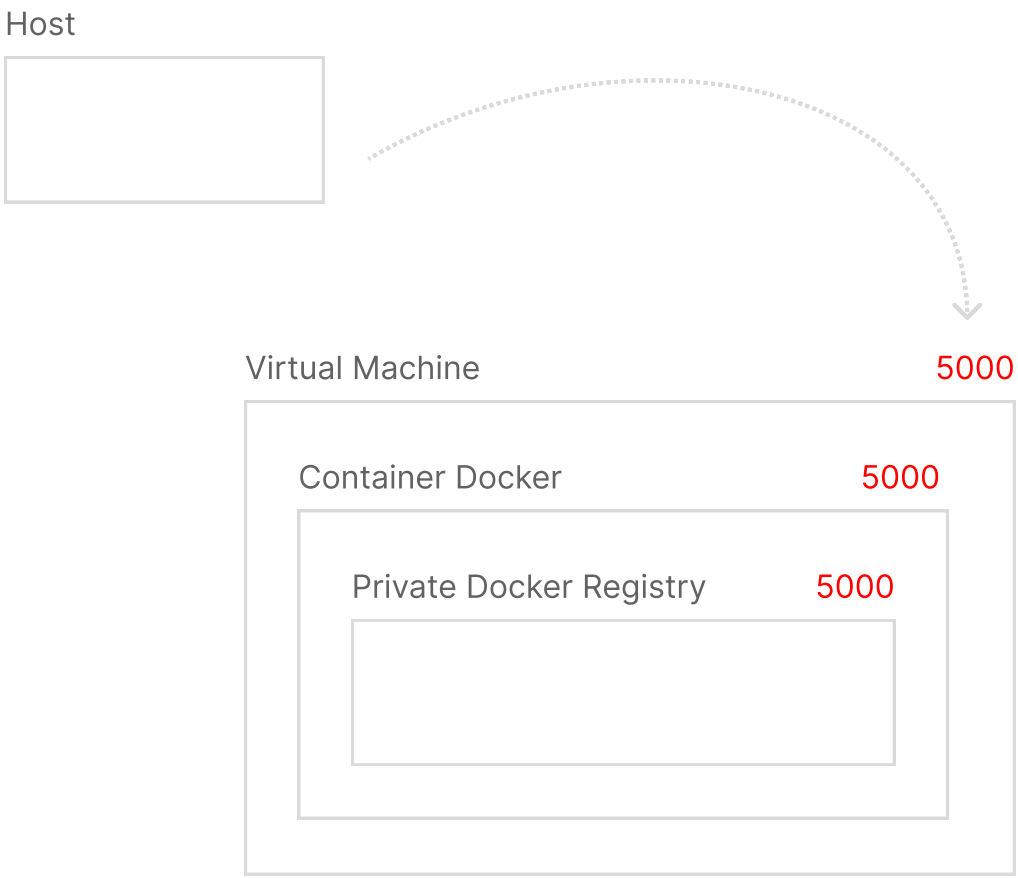
Criando um Registry Privado



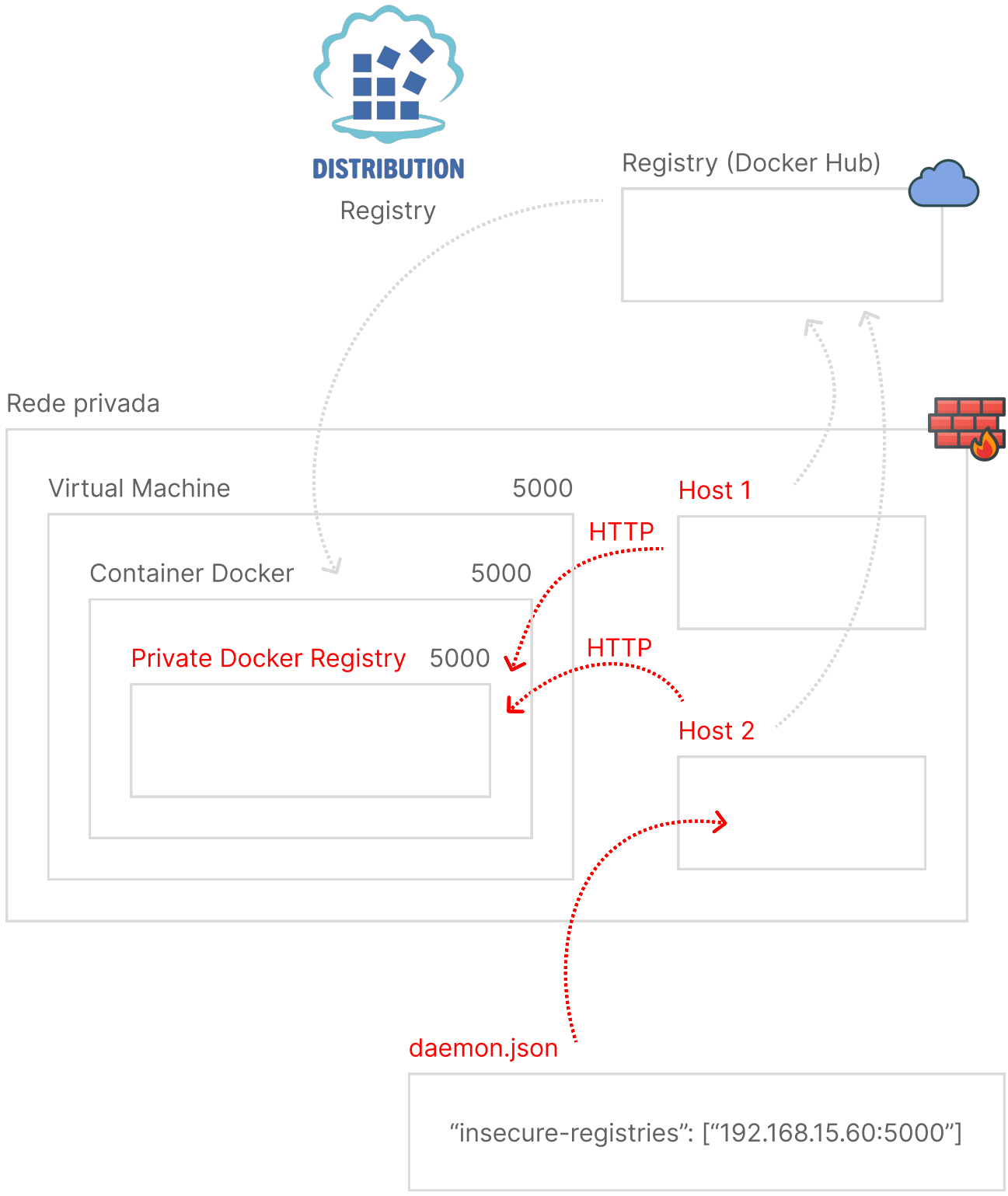
Subindo o serviço de Registry Privado



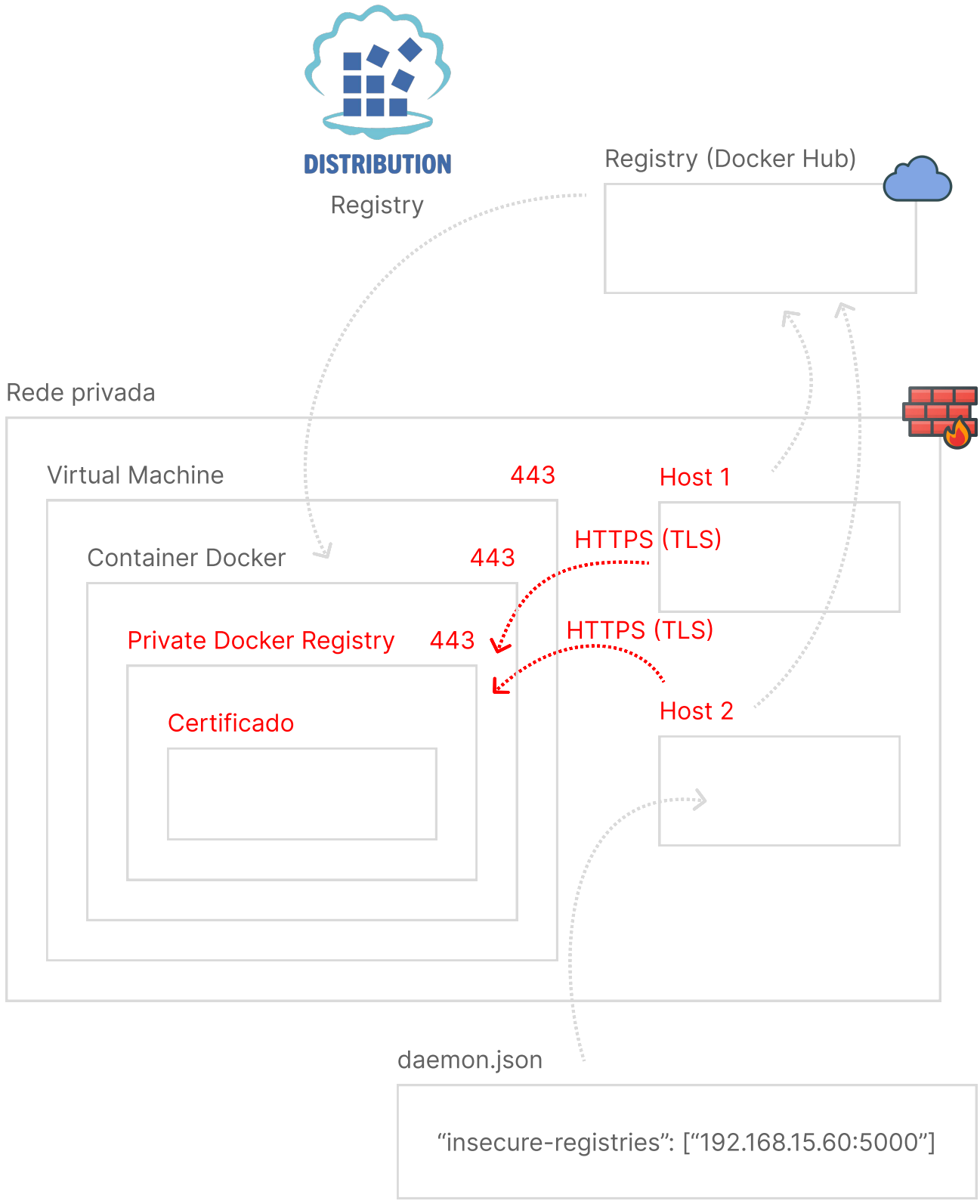
Acessando o IP da VM por meio do Host



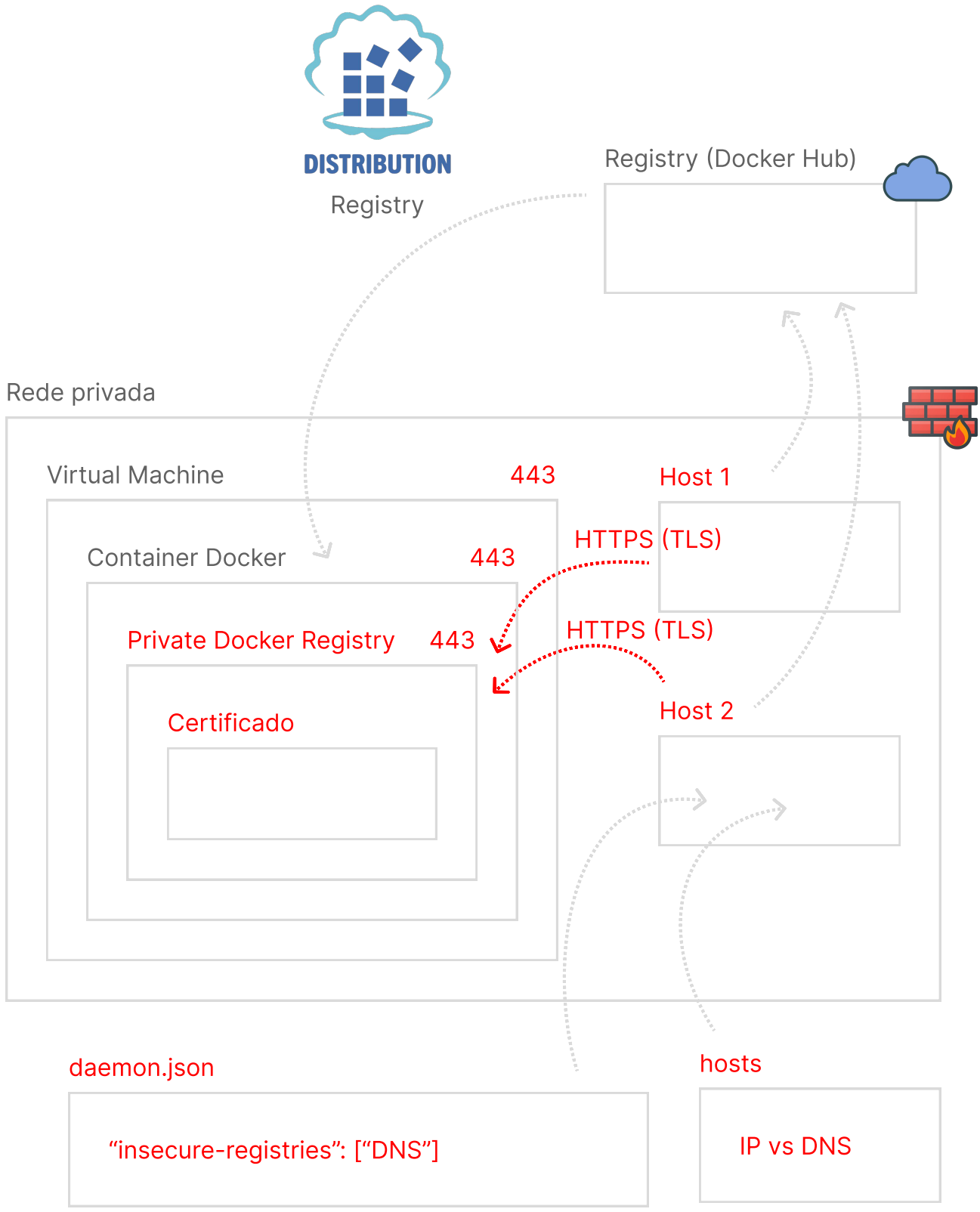
Enviando imagens para o Registry de modo inseguro



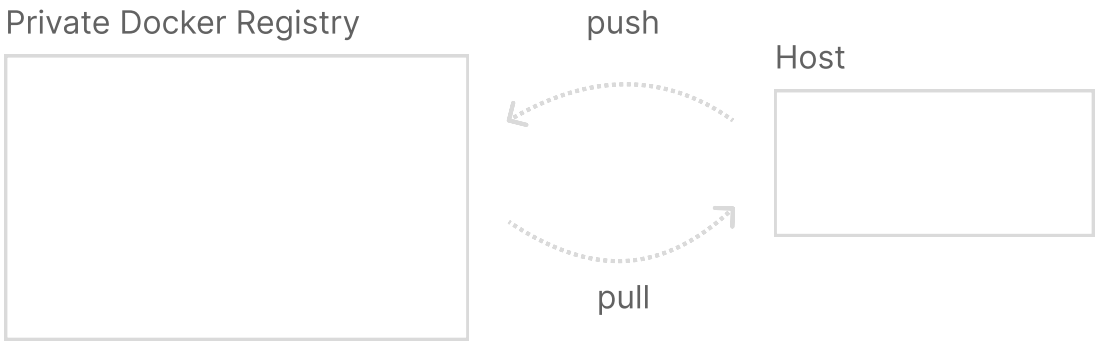
Enviando imagens para o Registry de modo seguro



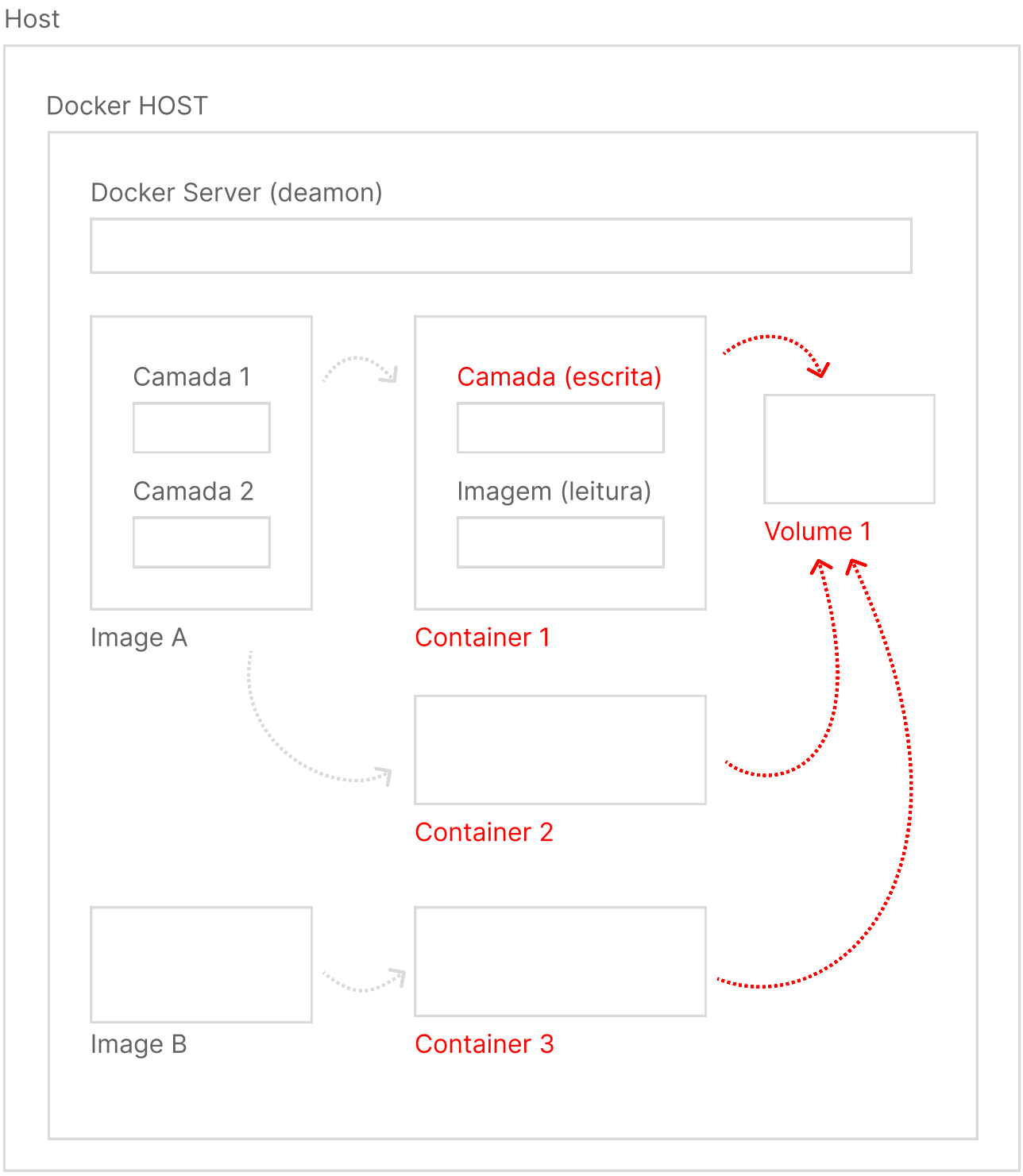
Enviando imagens para o Registry de modo seguro parte 2



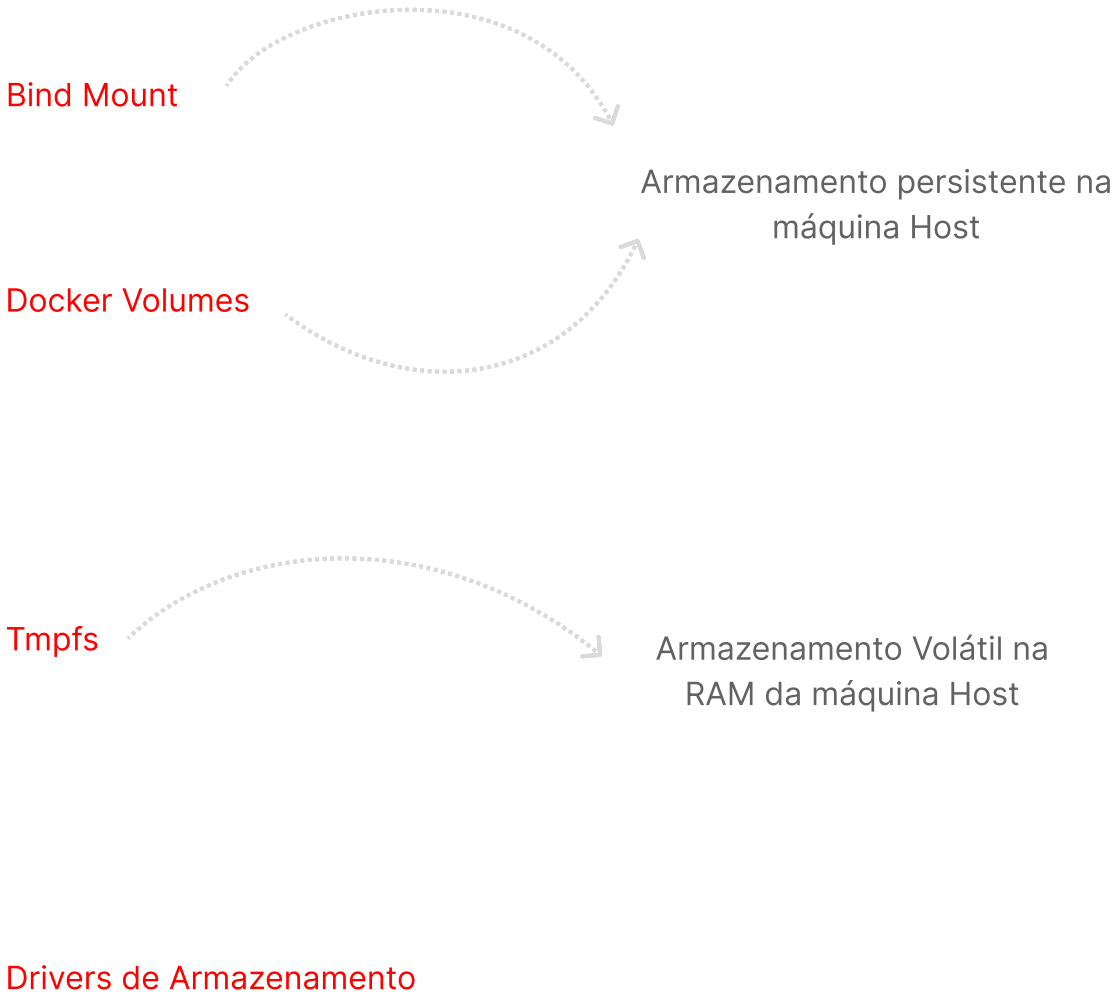
Baixando uma imagem do Registro Privado



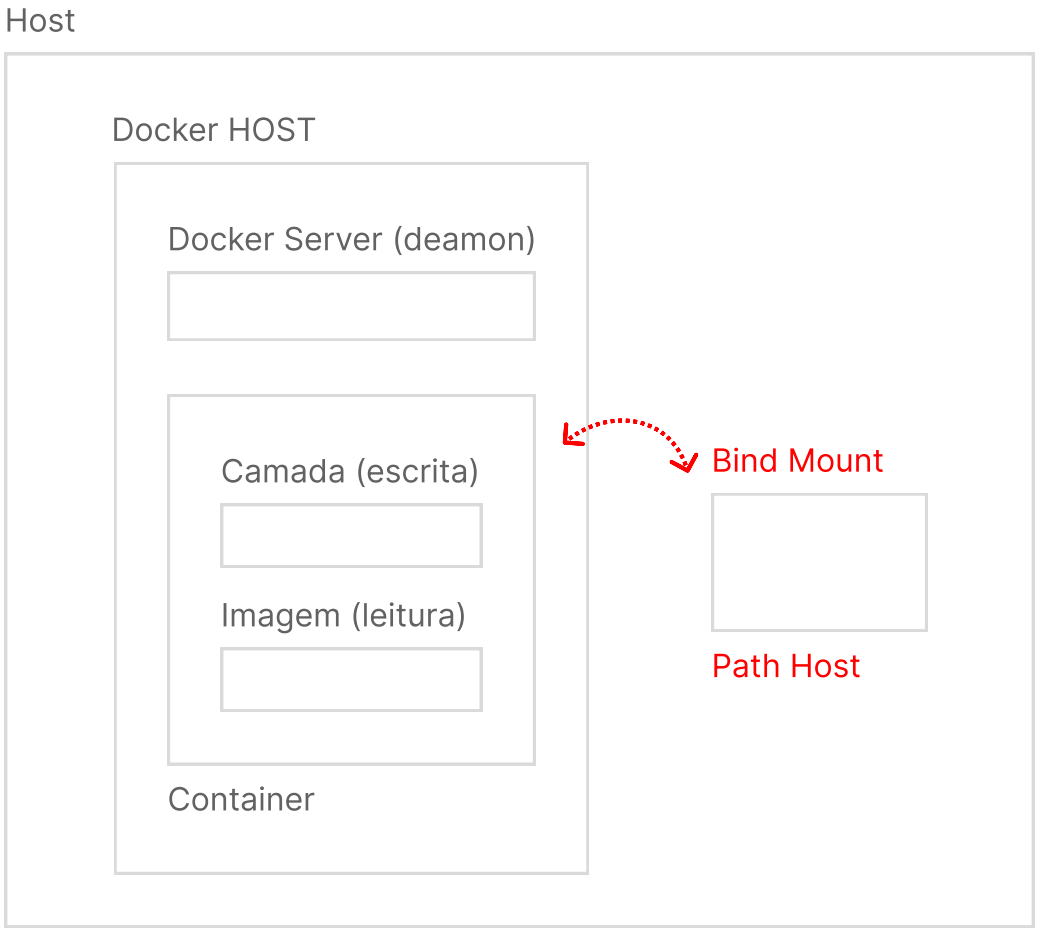
O que são volumes



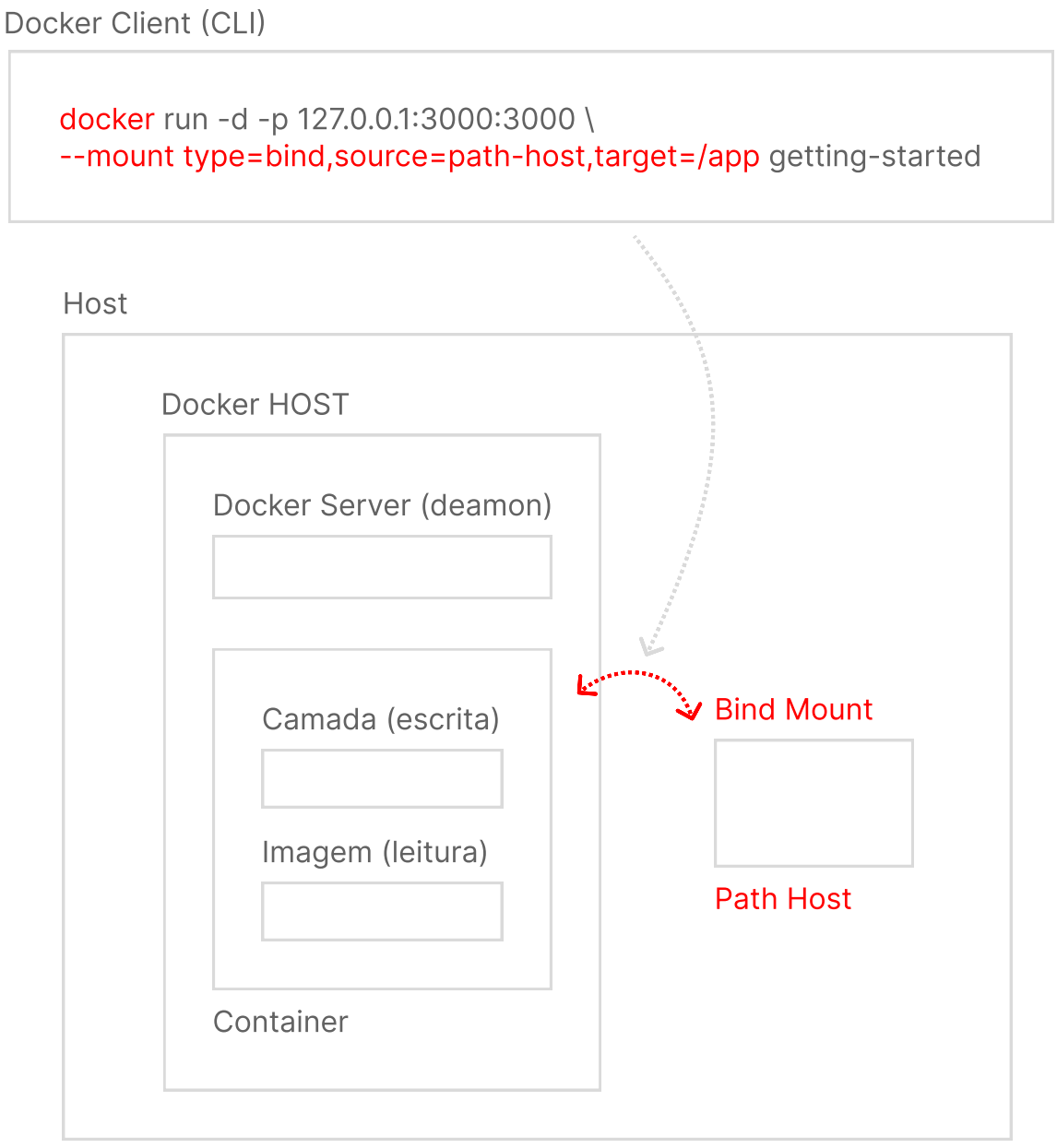
Tipos de montagens



Bind Mount - Casos de Uso



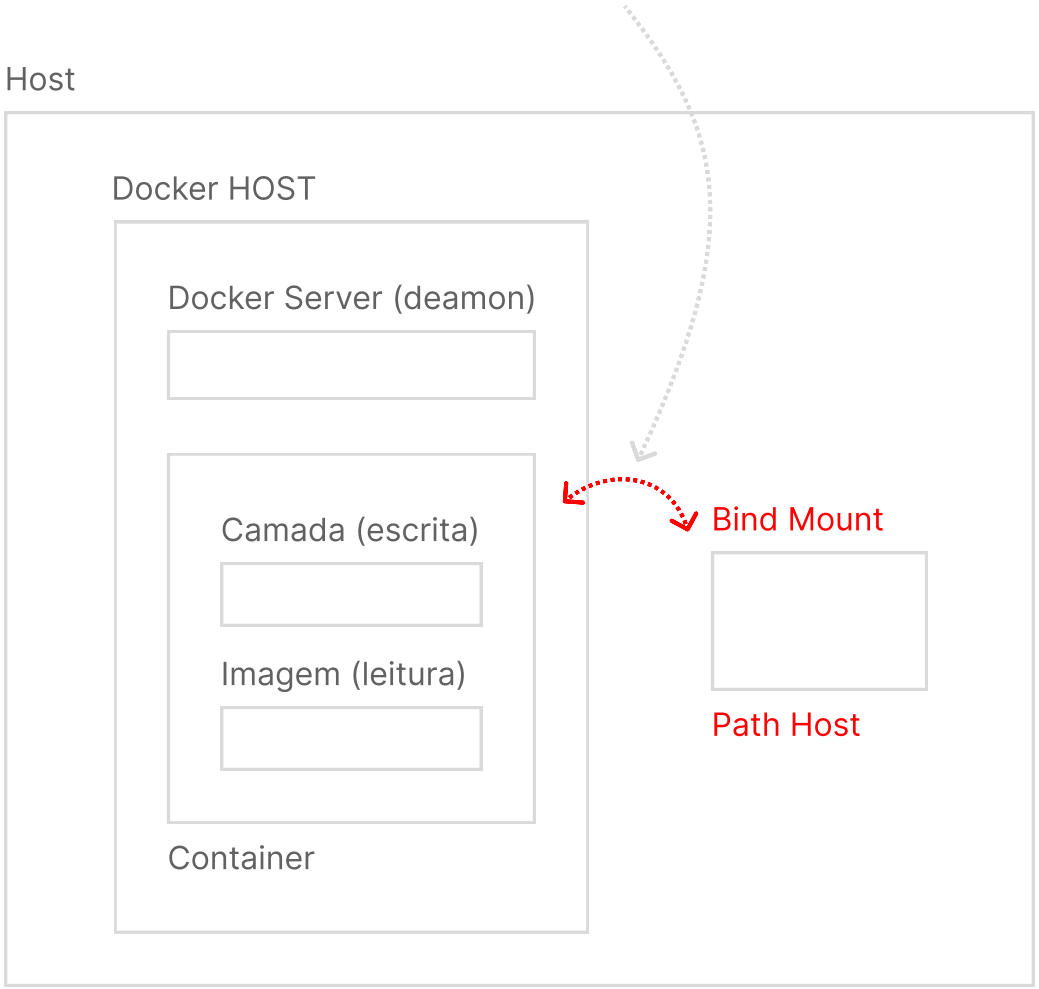
Bind Mount - Solucionando o problema



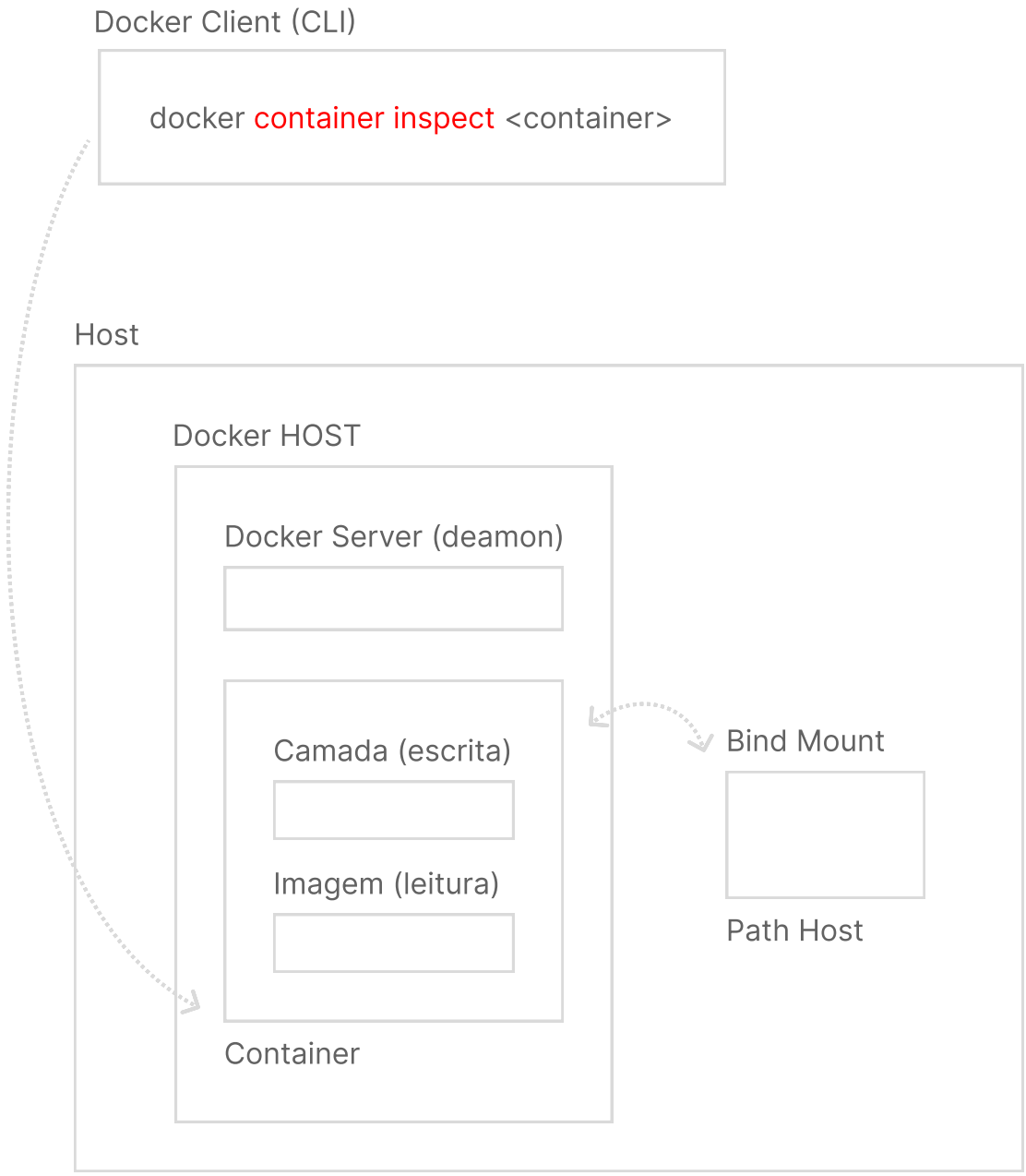
Bind Mount - Sintaxe alternativa

Docker Client (CLI)

```
docker run -d -p 127.0.0.1:3000:3000 \
--mount type=bind,source=path-host,target=/app getting-started

docker run -d -p 127.0.0.1:3000:3000 \
-v path-host:path-container getting-started
```



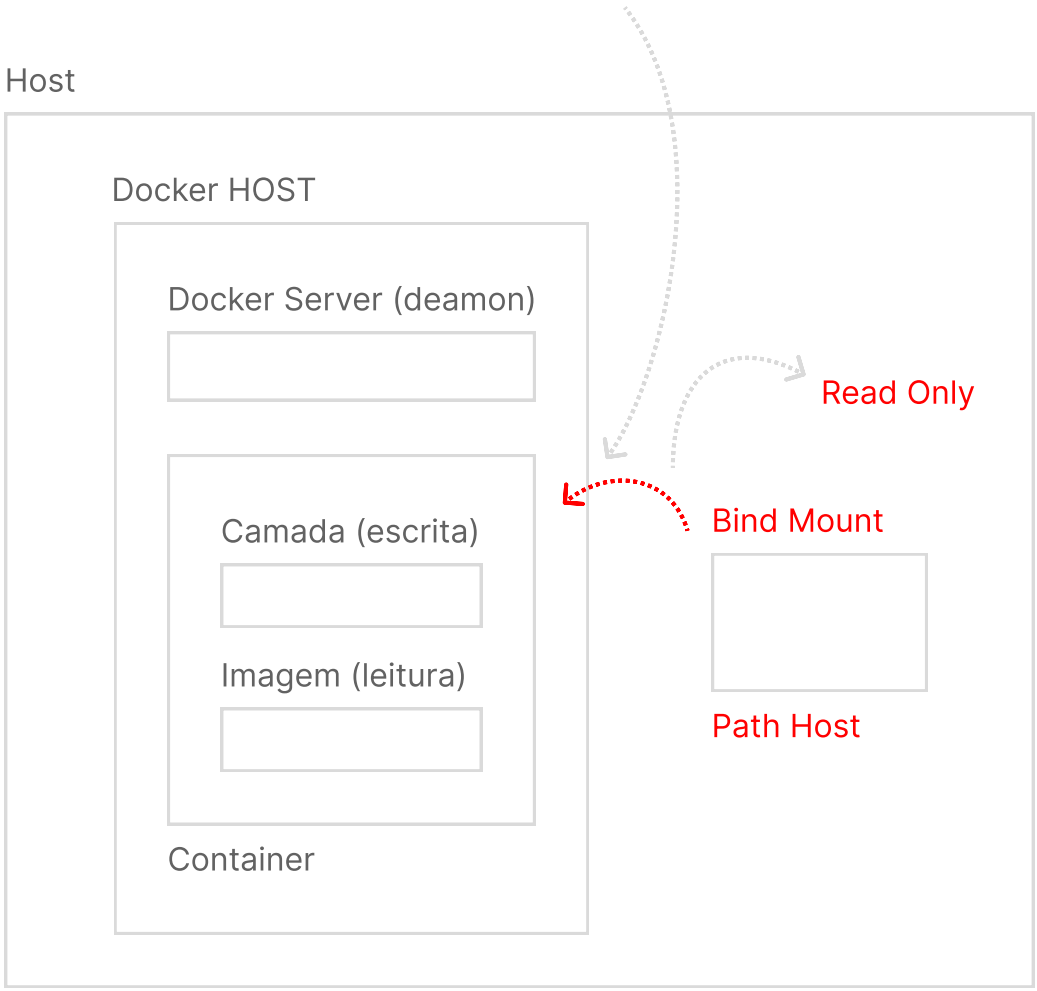
Inspecionando o container para verificar os pontos de montagem



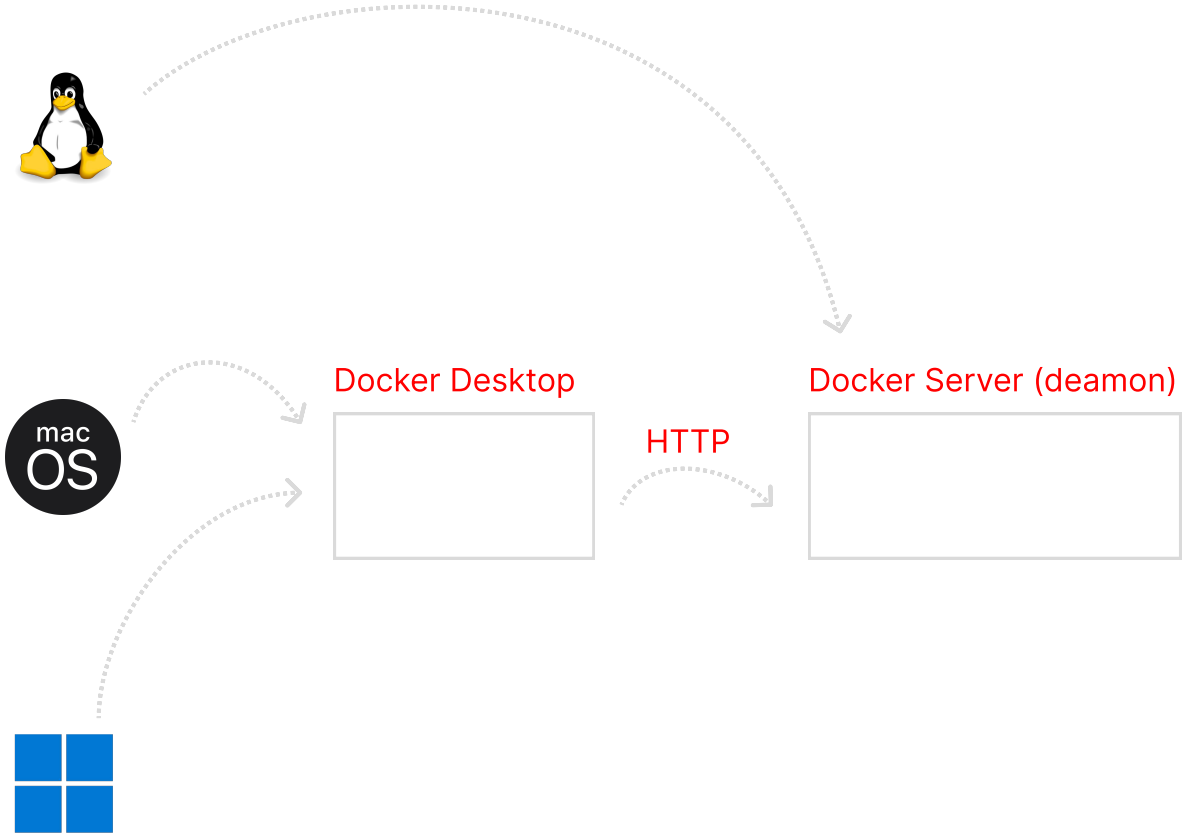
Bind Mount - Somente leitura

Docker Client (CLI)

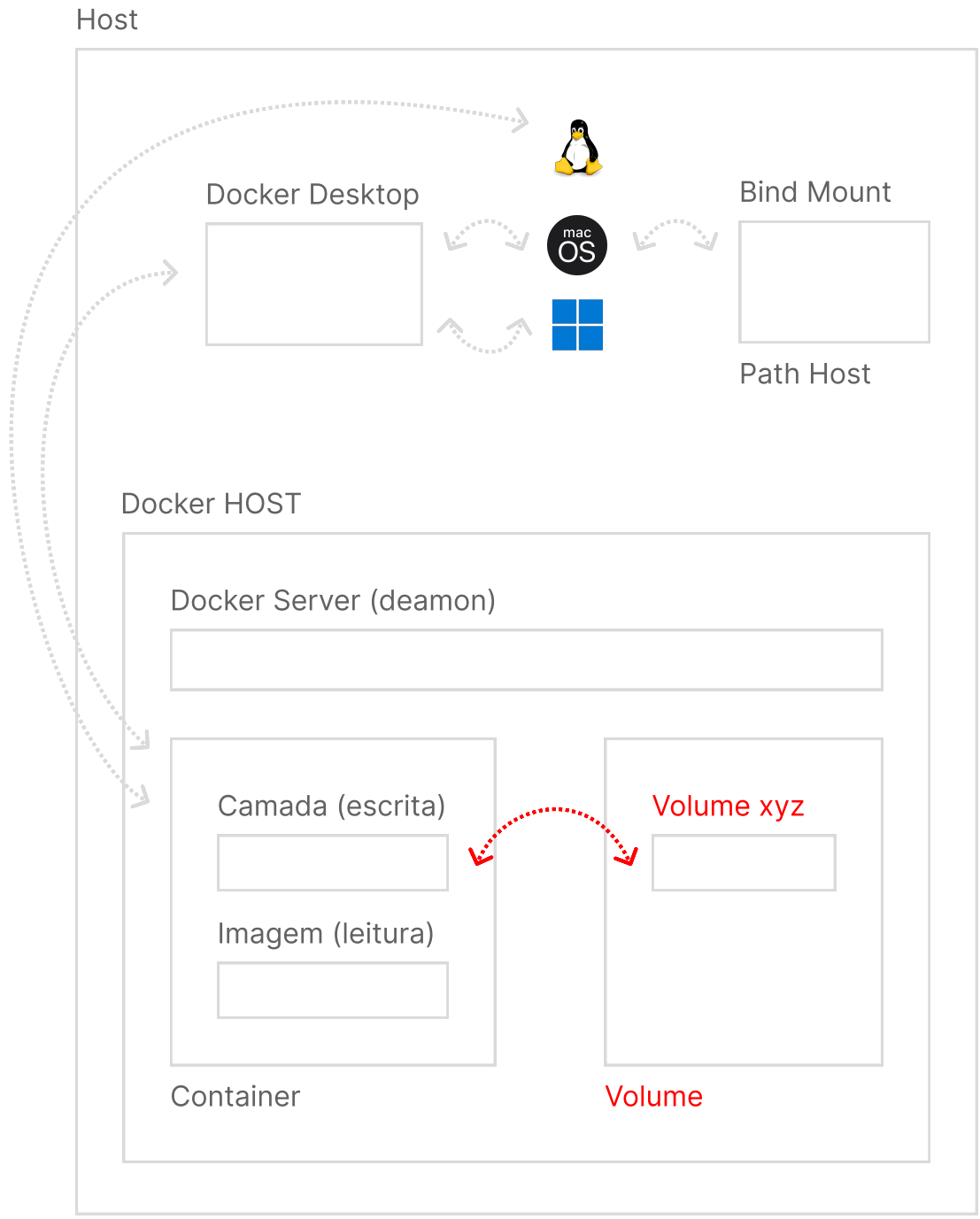
```
docker run -d -p 127.0.0.1:3000:3000 \
--mount type=bind,source=path-host,target=/app,readonly \
getting-started

docker run -d -p 127.0.0.1:3000:3000 \
-v path-host:path-container:ro getting-started
```



Bind Mount - Considerações sobre o funcionamento e performance



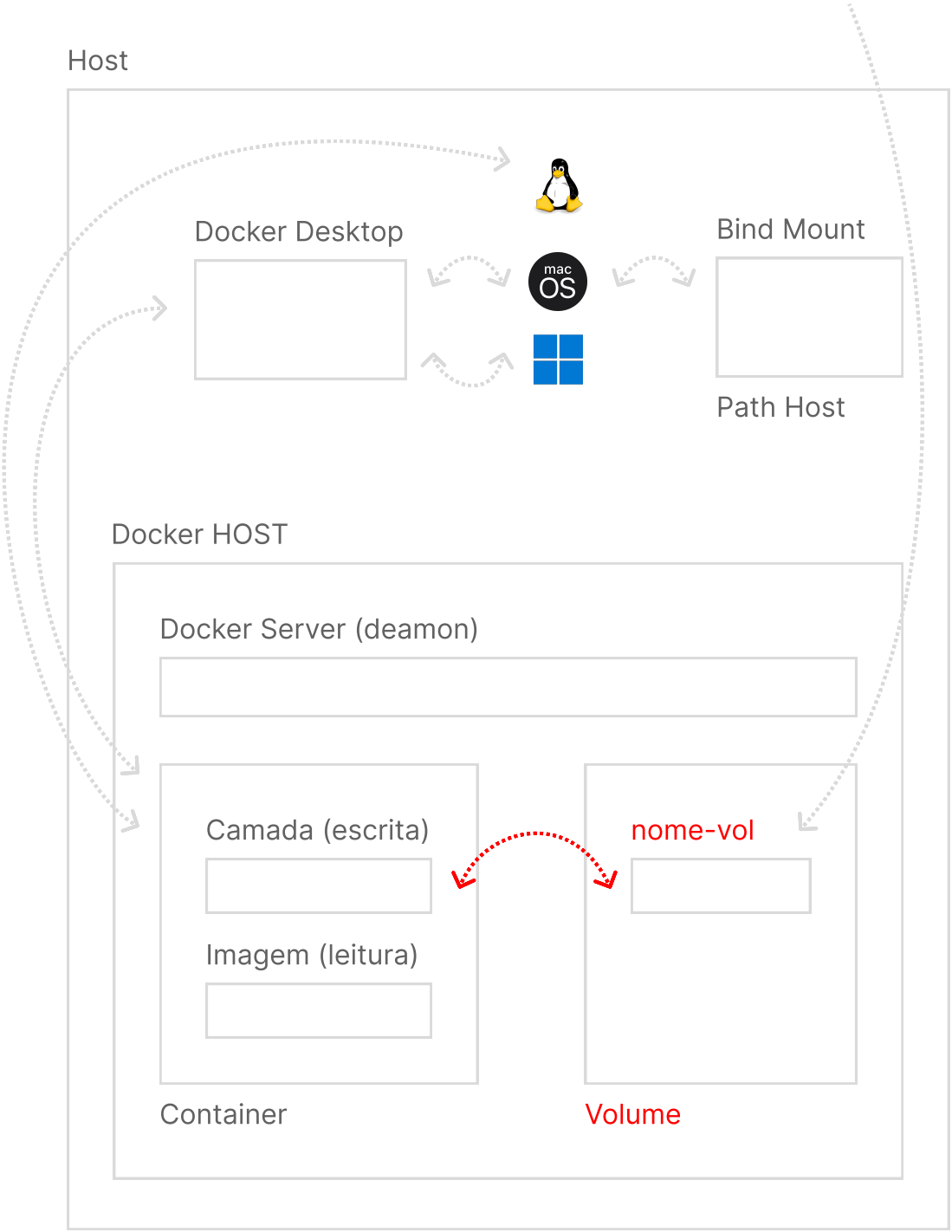
Docker Volume - Casos de Uso



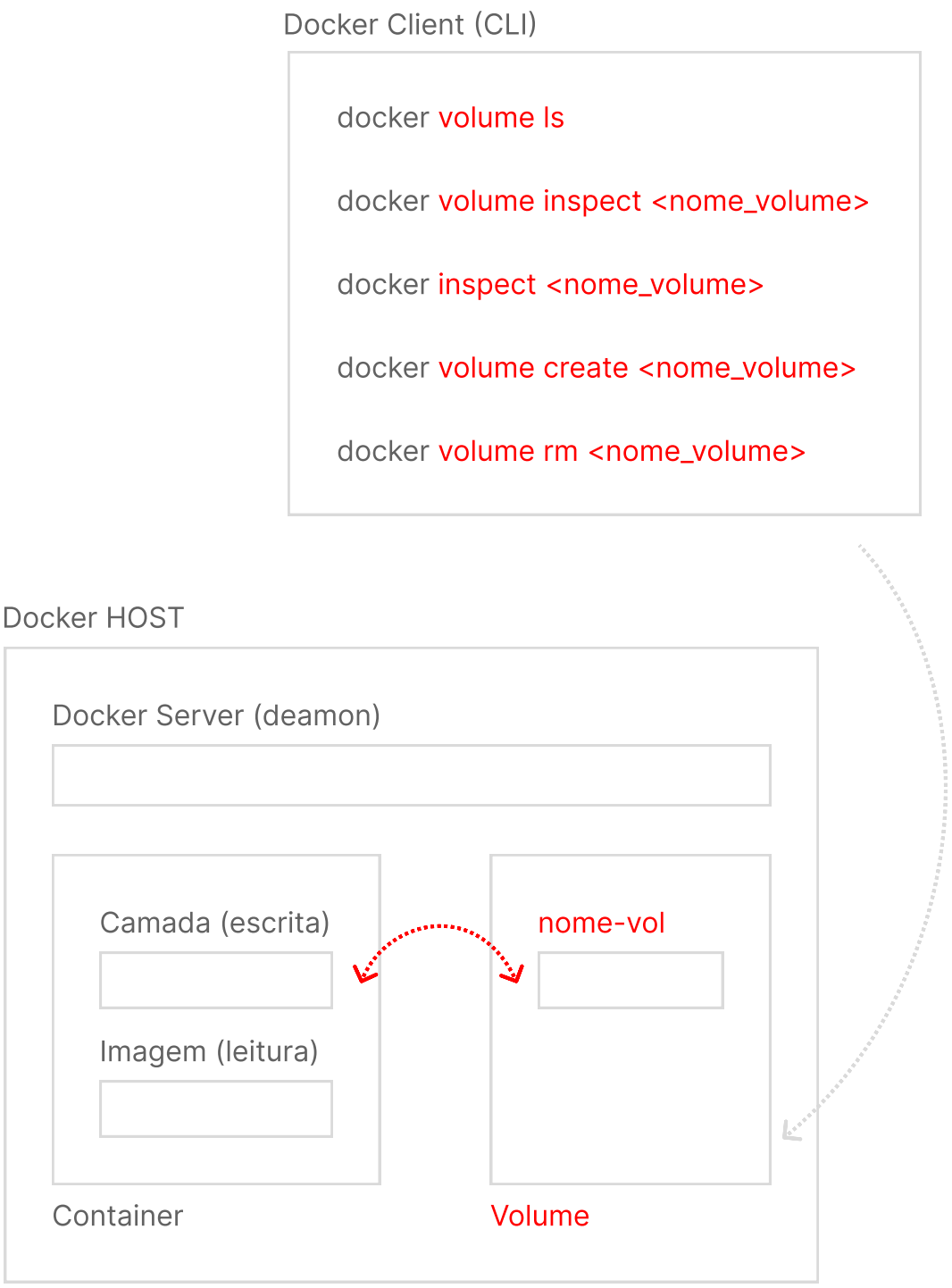
Docker Volume - Solucionando o problema

Docker Client (CLI)

```
docker run -d -p 5432:5432 \
--mount type=volume,source=nome-vol,target=/data postgresql
```



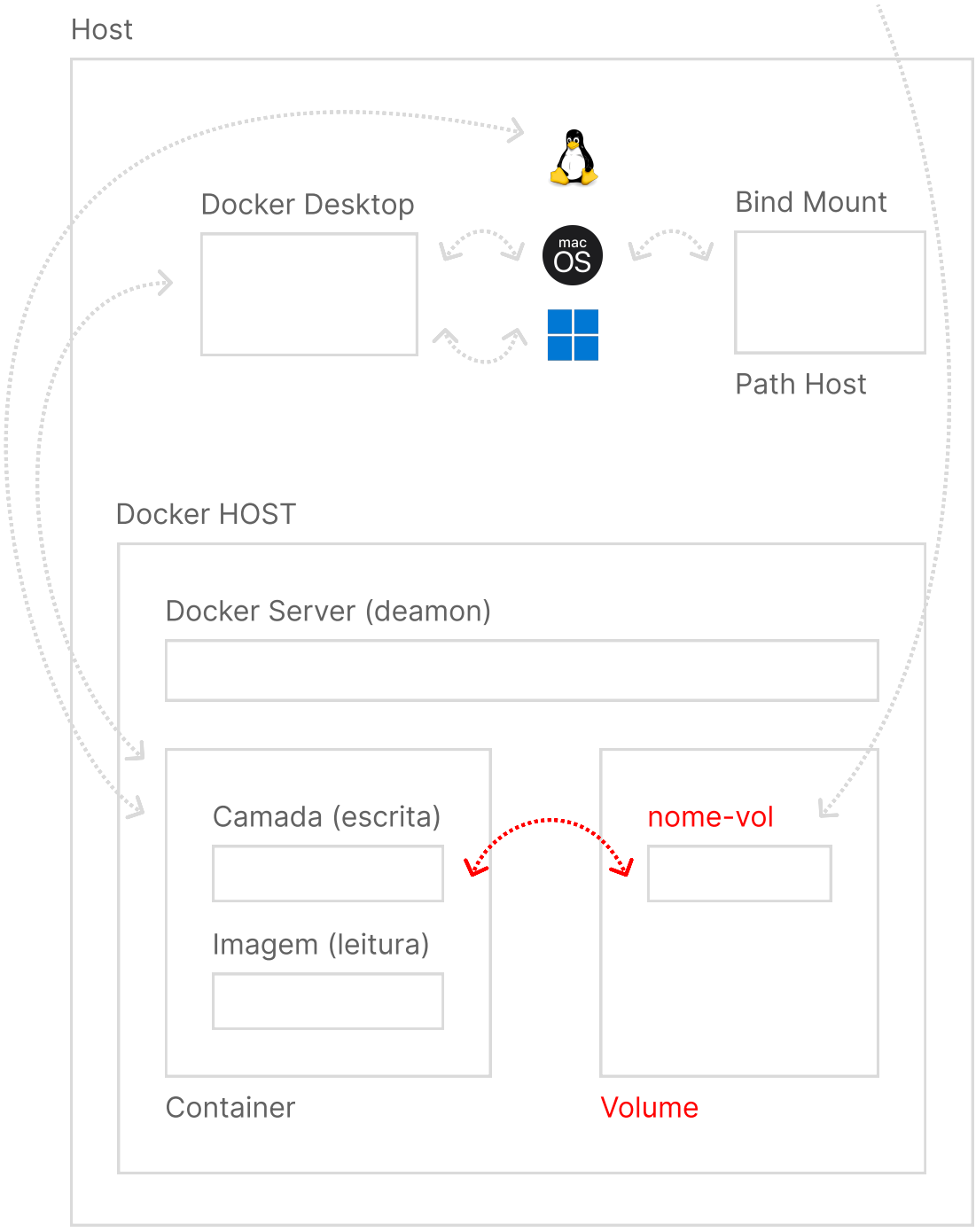
Docker Volume - Localizando, inspecionando e manipulando os volumes



Docker Volume - Sintaxe alternativa

Docker Client (CLI)

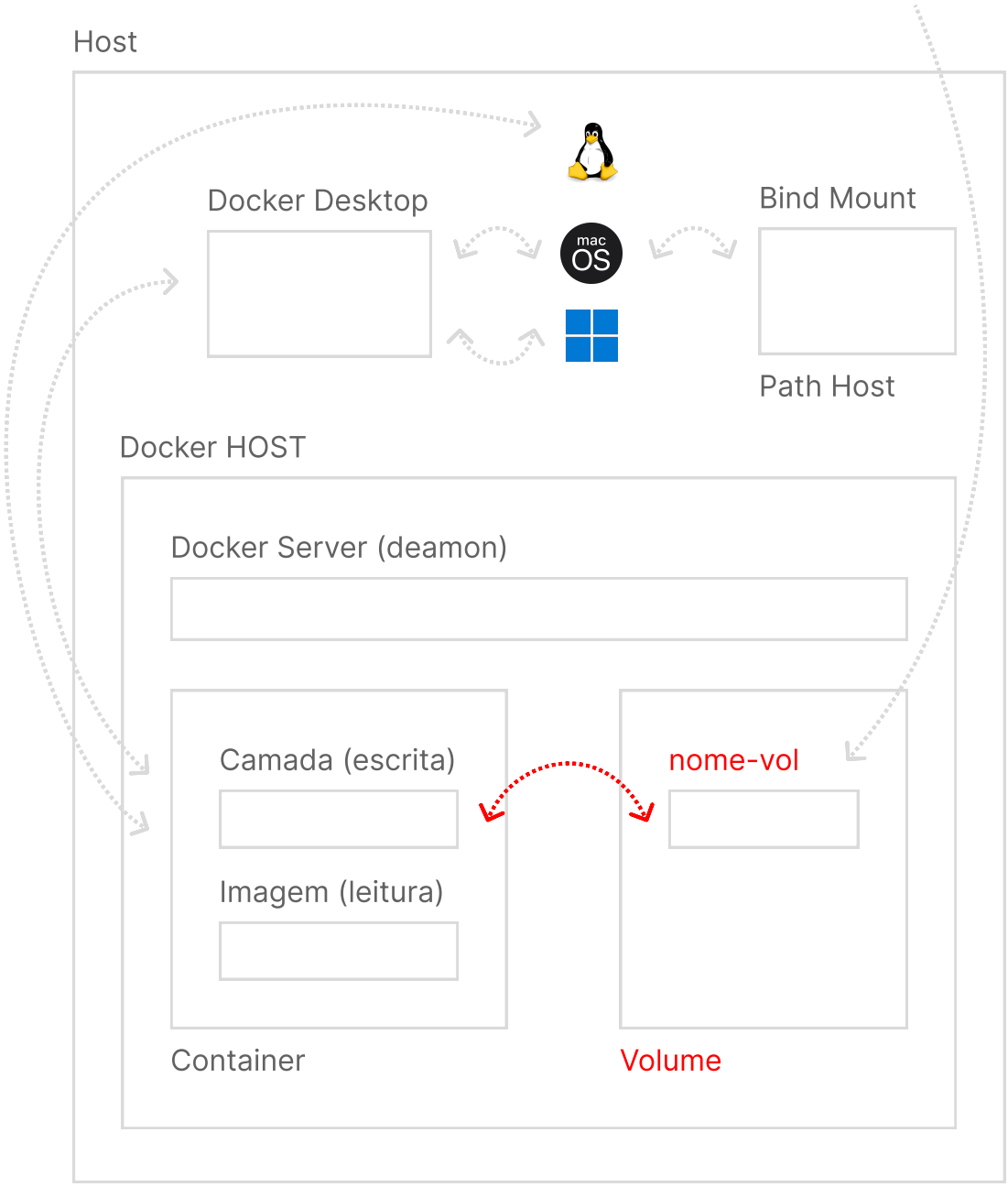
```
docker run -d -p 5432:5432 \
--mount type=volume,source=nome-vol,target=/data postgresql

docker run -d -p 5432:5432 -v nome-vol:/data postgresql
```



Docker Volume - Sintaxe alternativa parte 2

Docker Client (CLI)

```
docker run -d -p 5432:5432 \
--mount type=volume,source=nome-vol,target=/data postgresql

docker run -d -p 5432:5432 -v nome-vol:/data postgresql

docker run -d -p 5432:5432 -v /data postgresql
```

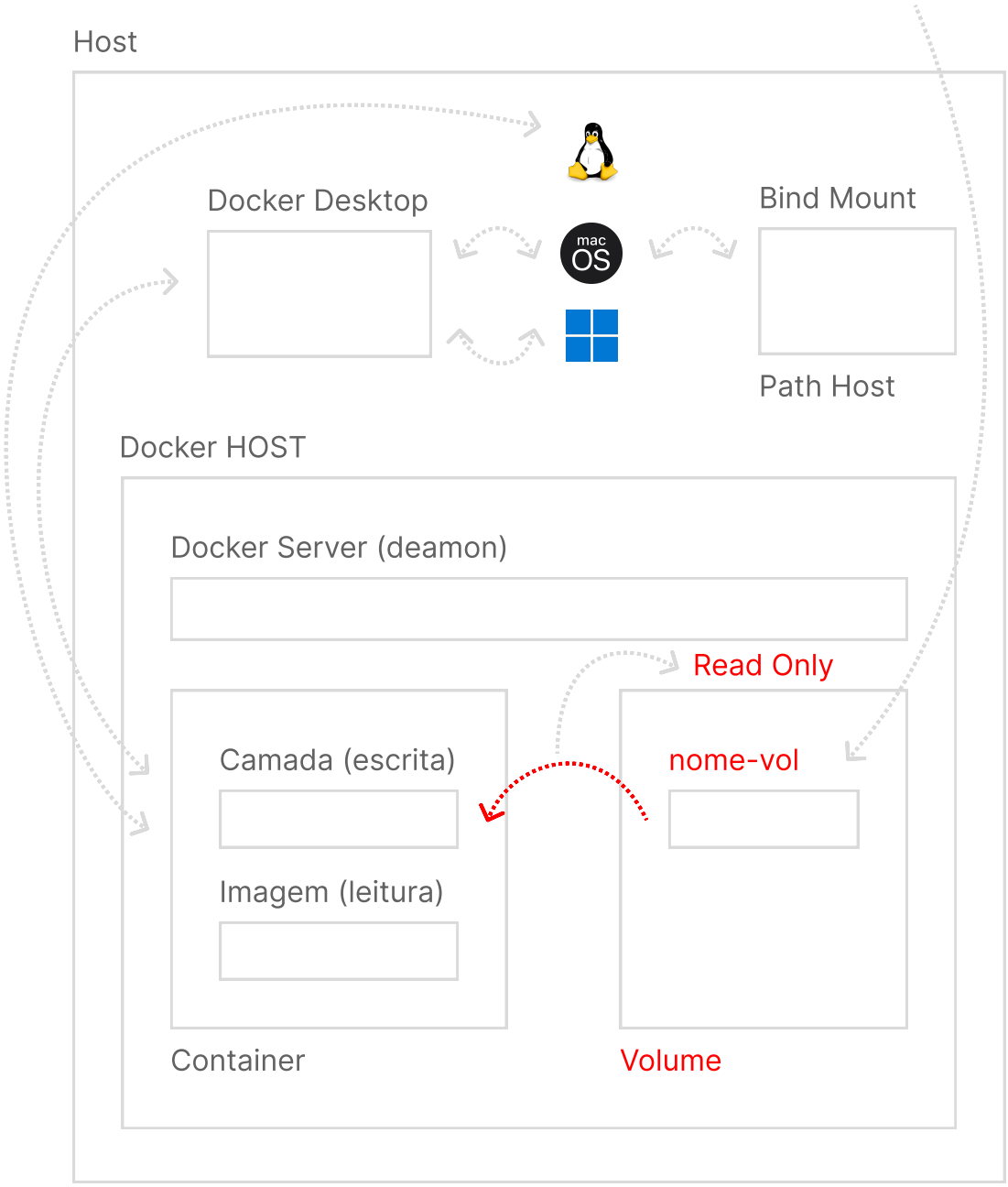


Docker Volume - Somente leitura

Docker Client (CLI)

```
docker run -d -p 5432:5432 \
--mount type=volume,source=nome-vol,target=/data,readonly postgresql

docker run -d -p 5432:5432 -v nome-vol:/data:ro postgresql

docker run -d -p 5432:5432 -v /data:ro postgresql
```

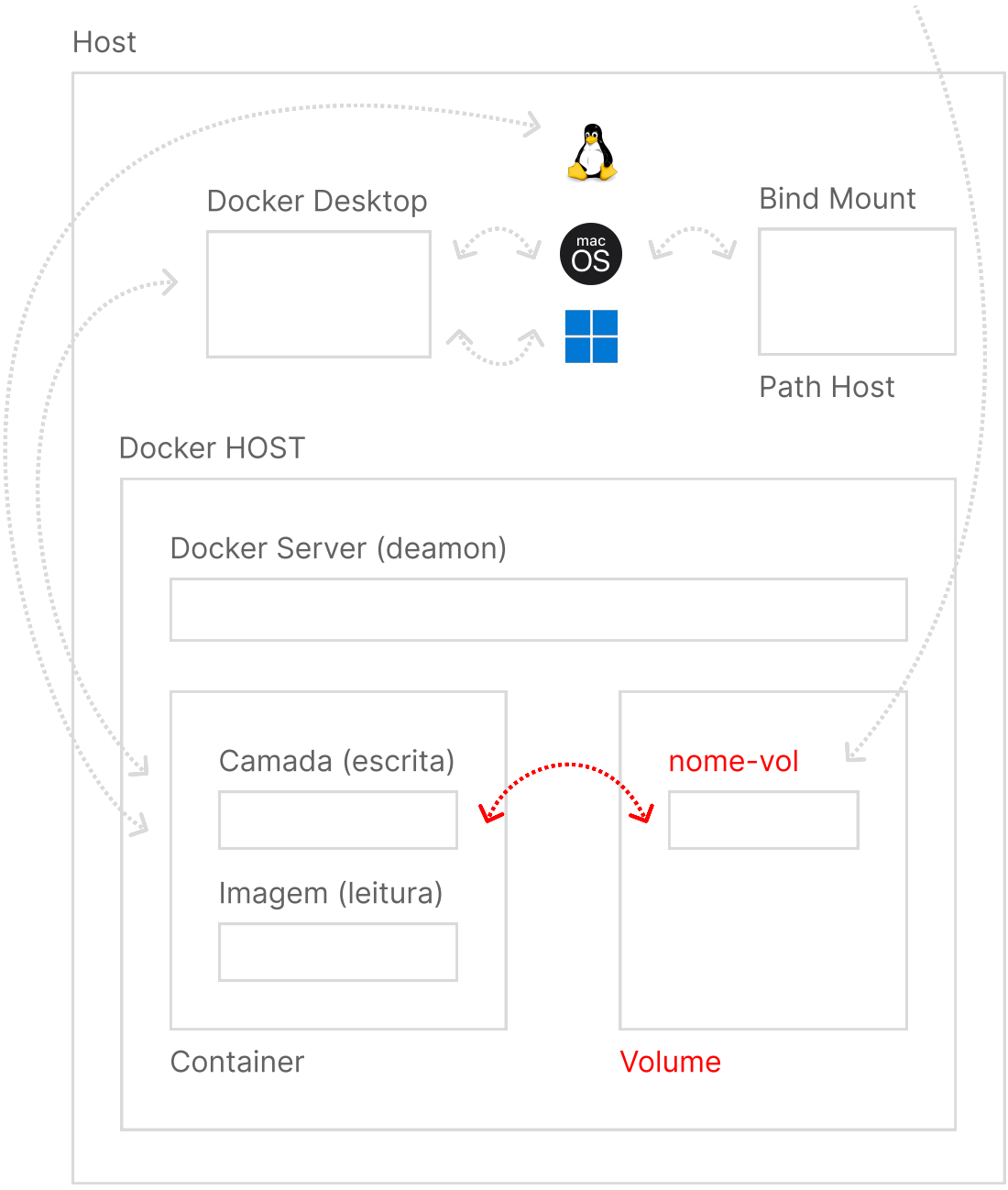
Erro de sintaxe



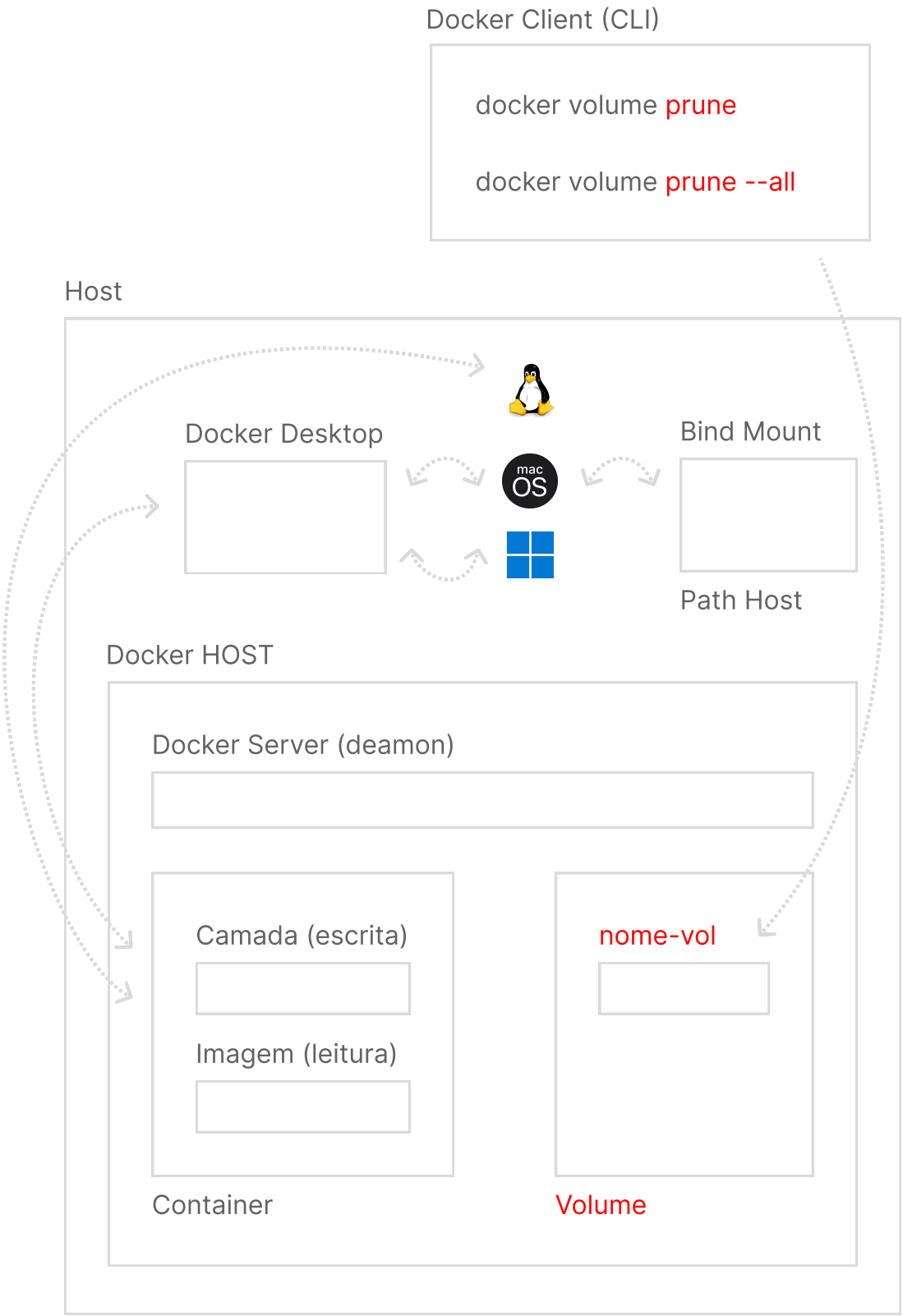
Docker Volume - Definição no Dockerfile

Dockerfile

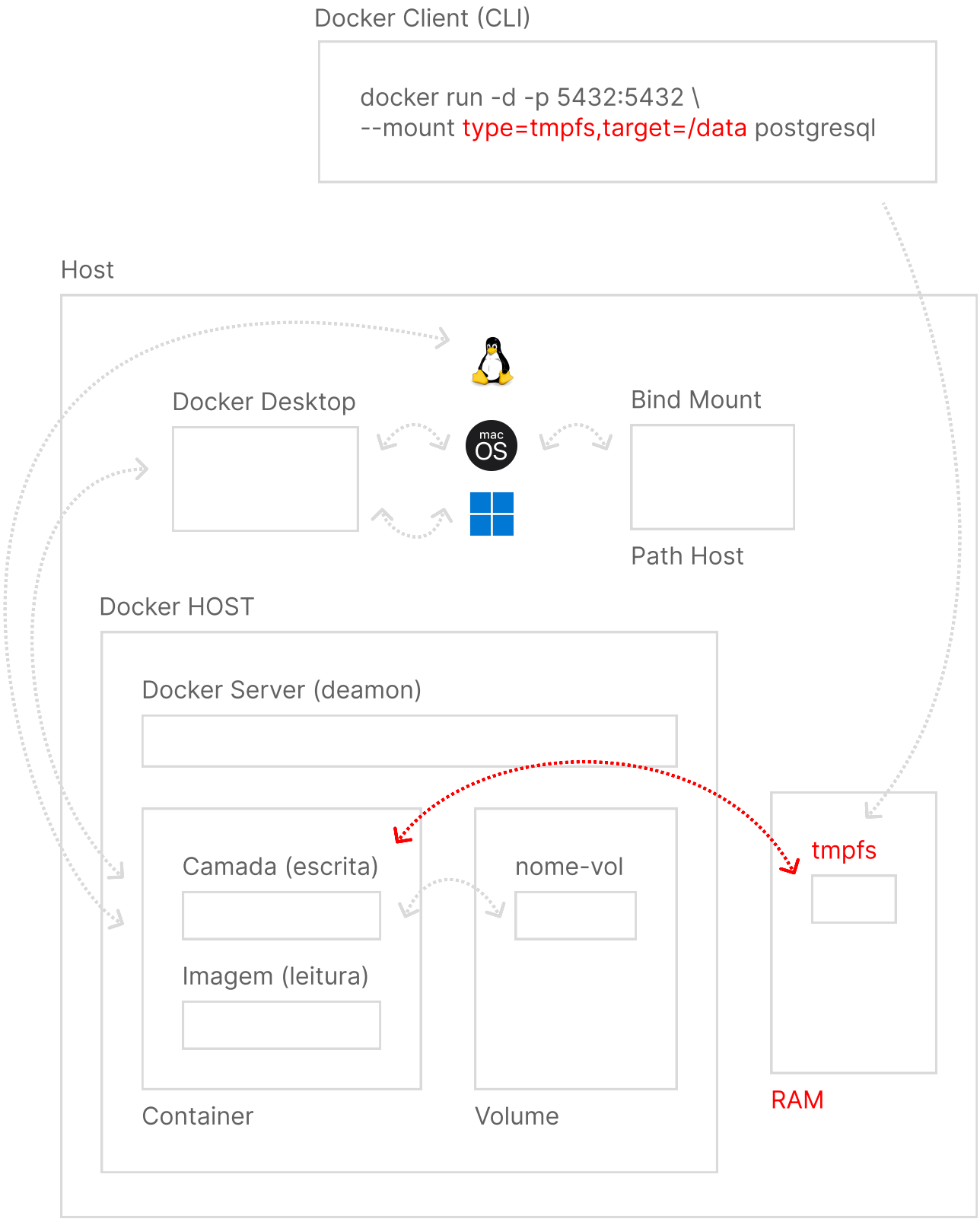
```
VOLUME ["/var/log"]

VOLUME /var/log

VOLUME /var/log /var/db
```



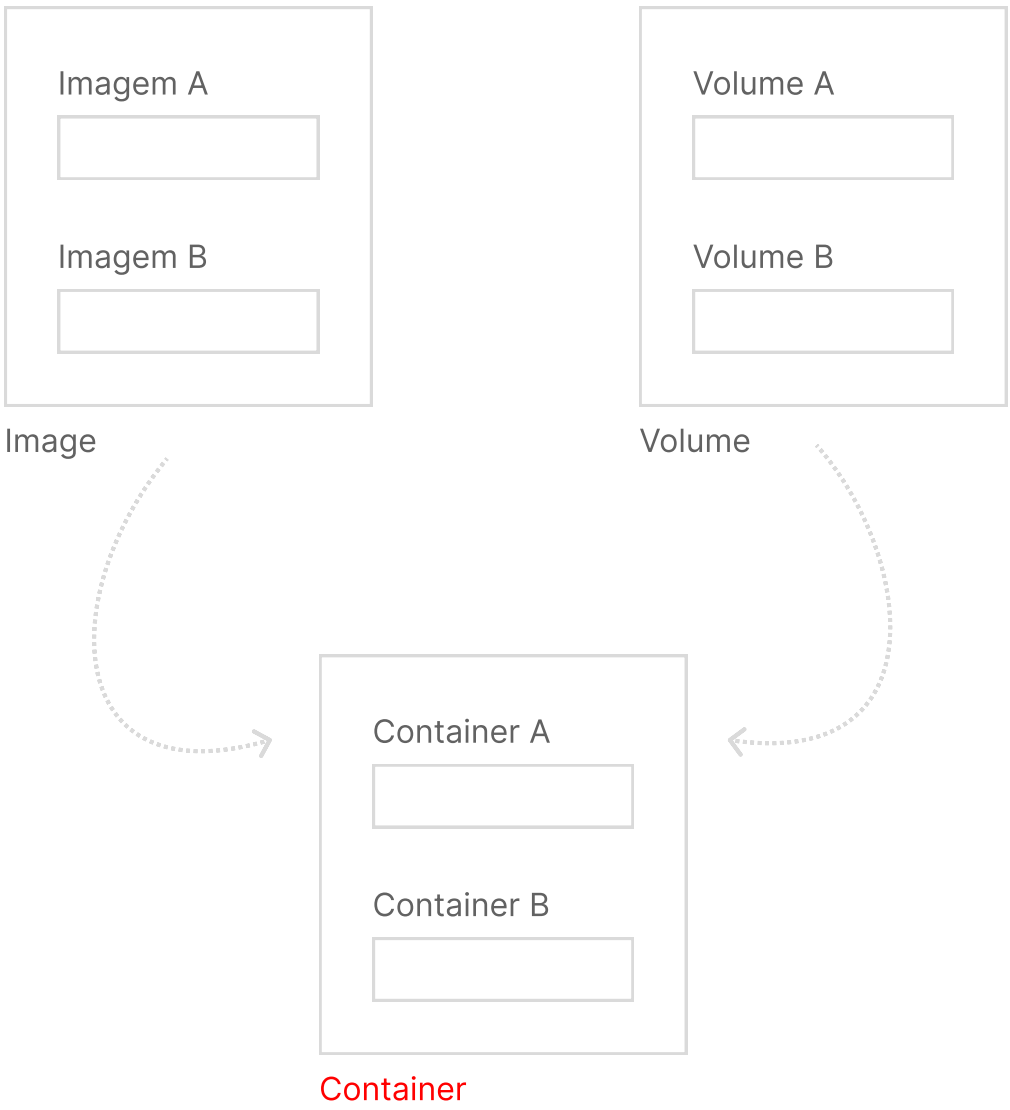
Removendo volumes em massa



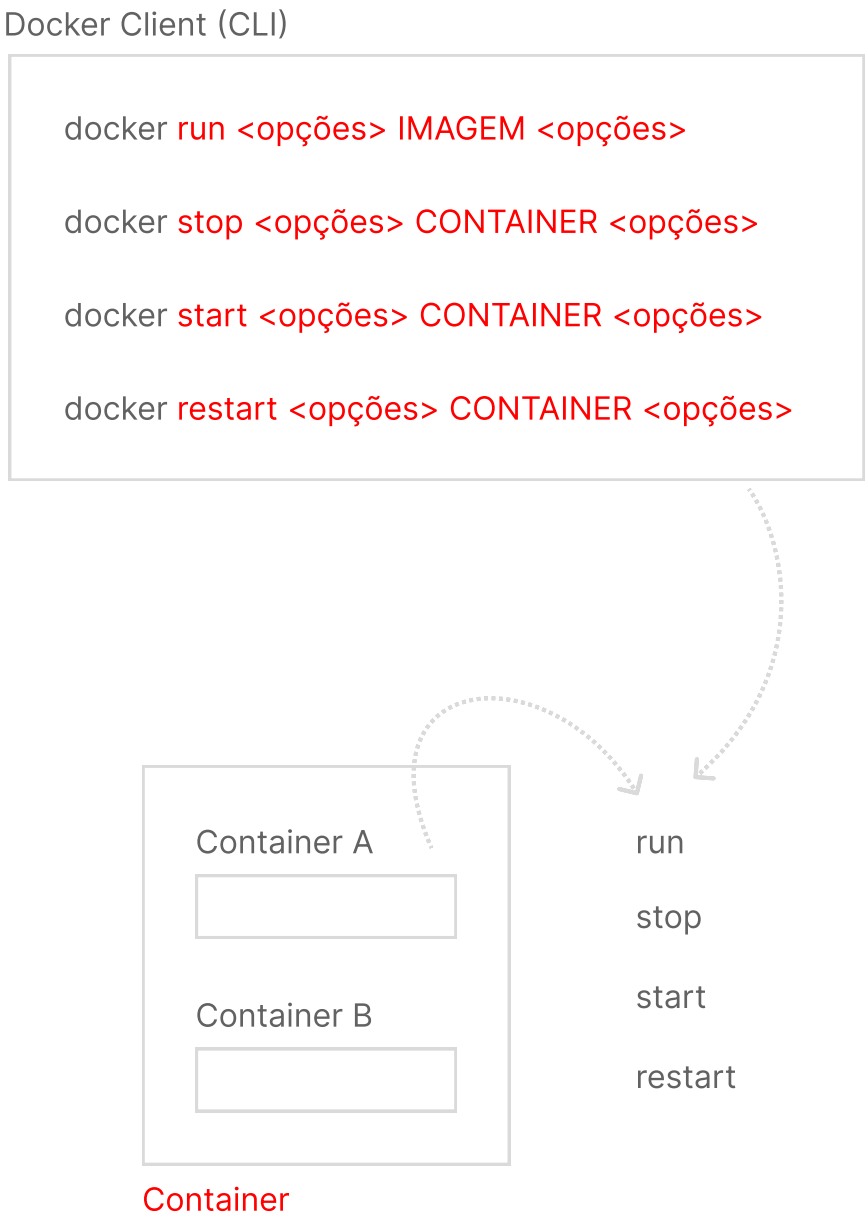
TMPFS - Armazenamento de dados não persistentes



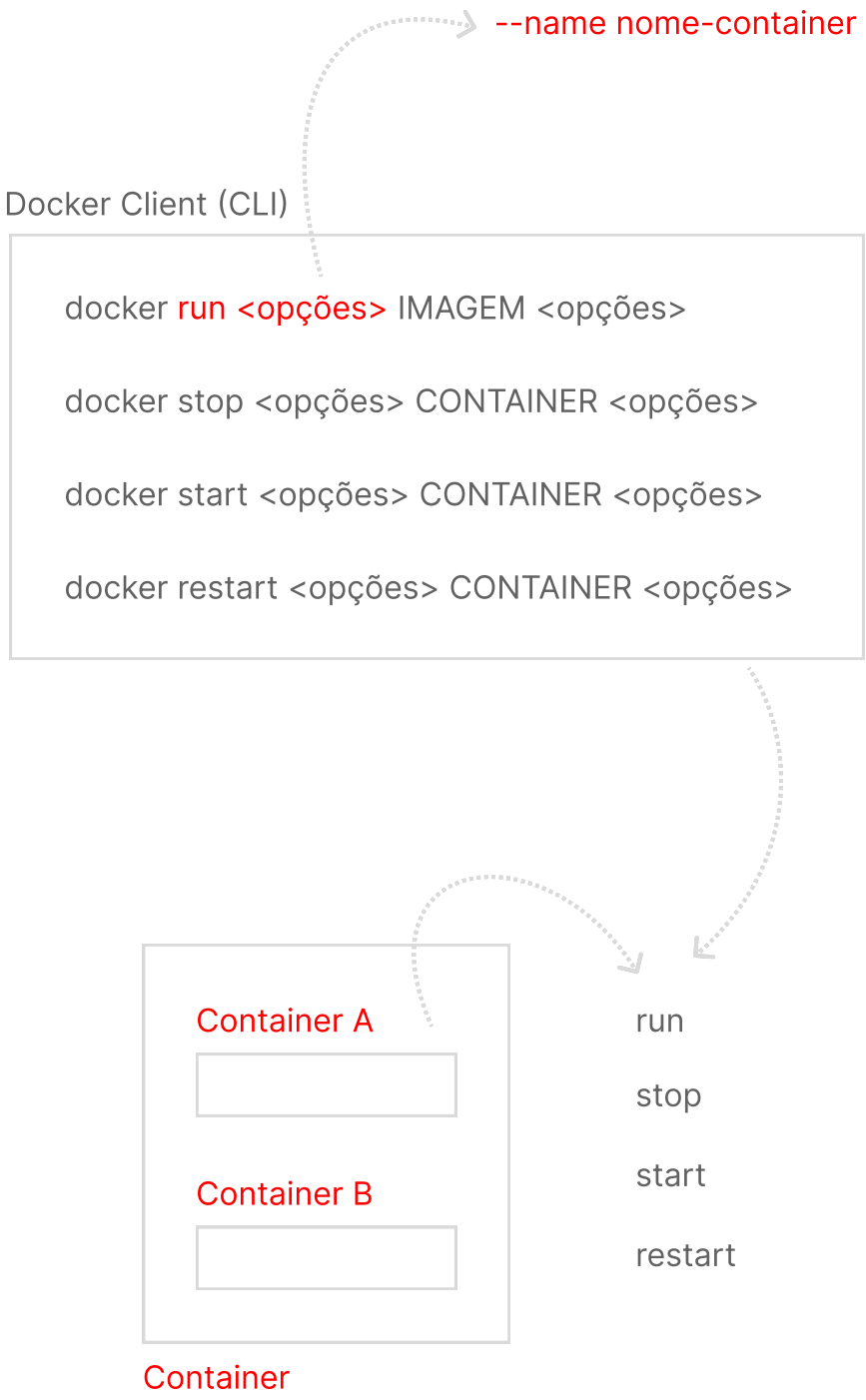
O que são containers



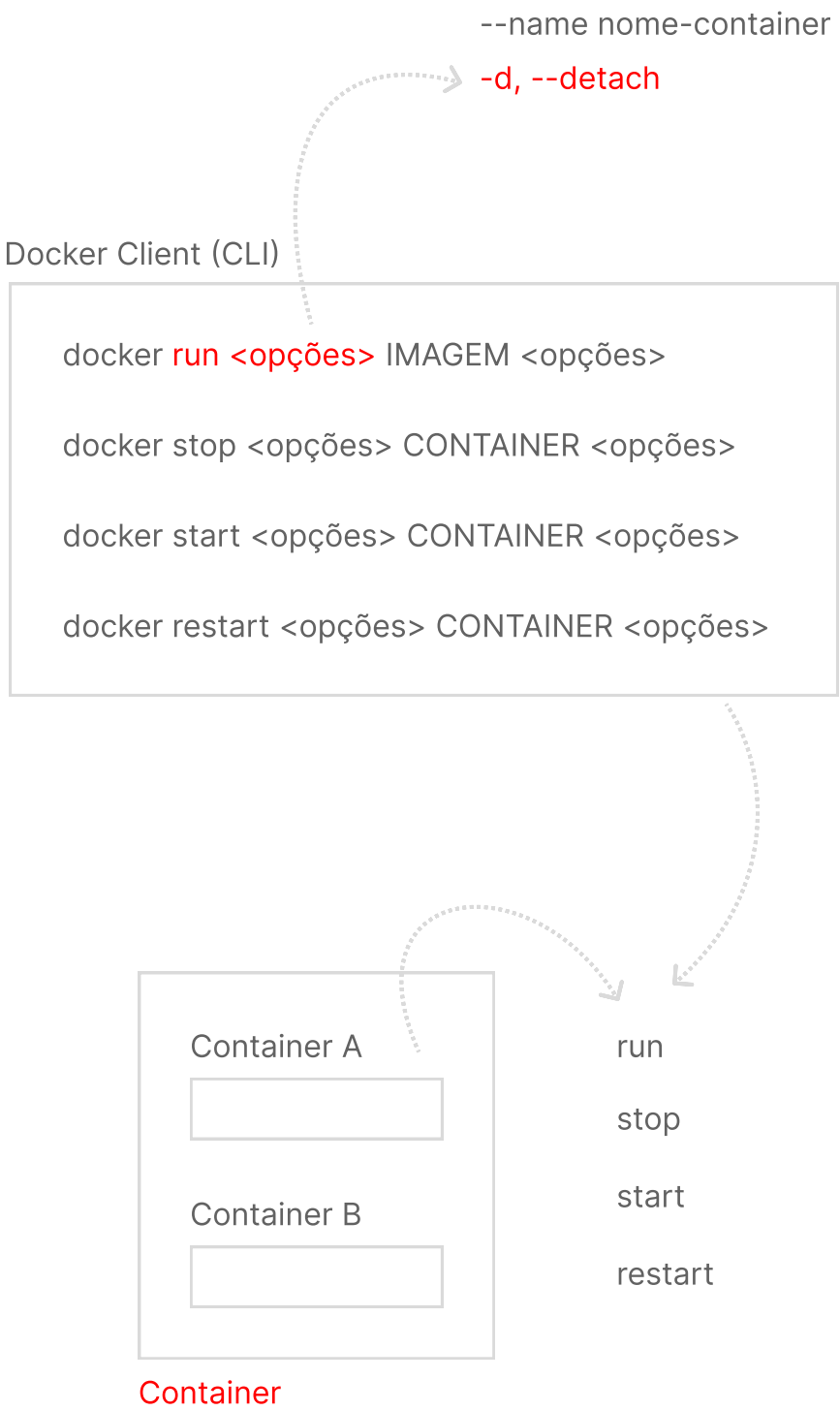
Executando, parando, iniciando e reiniciando containers



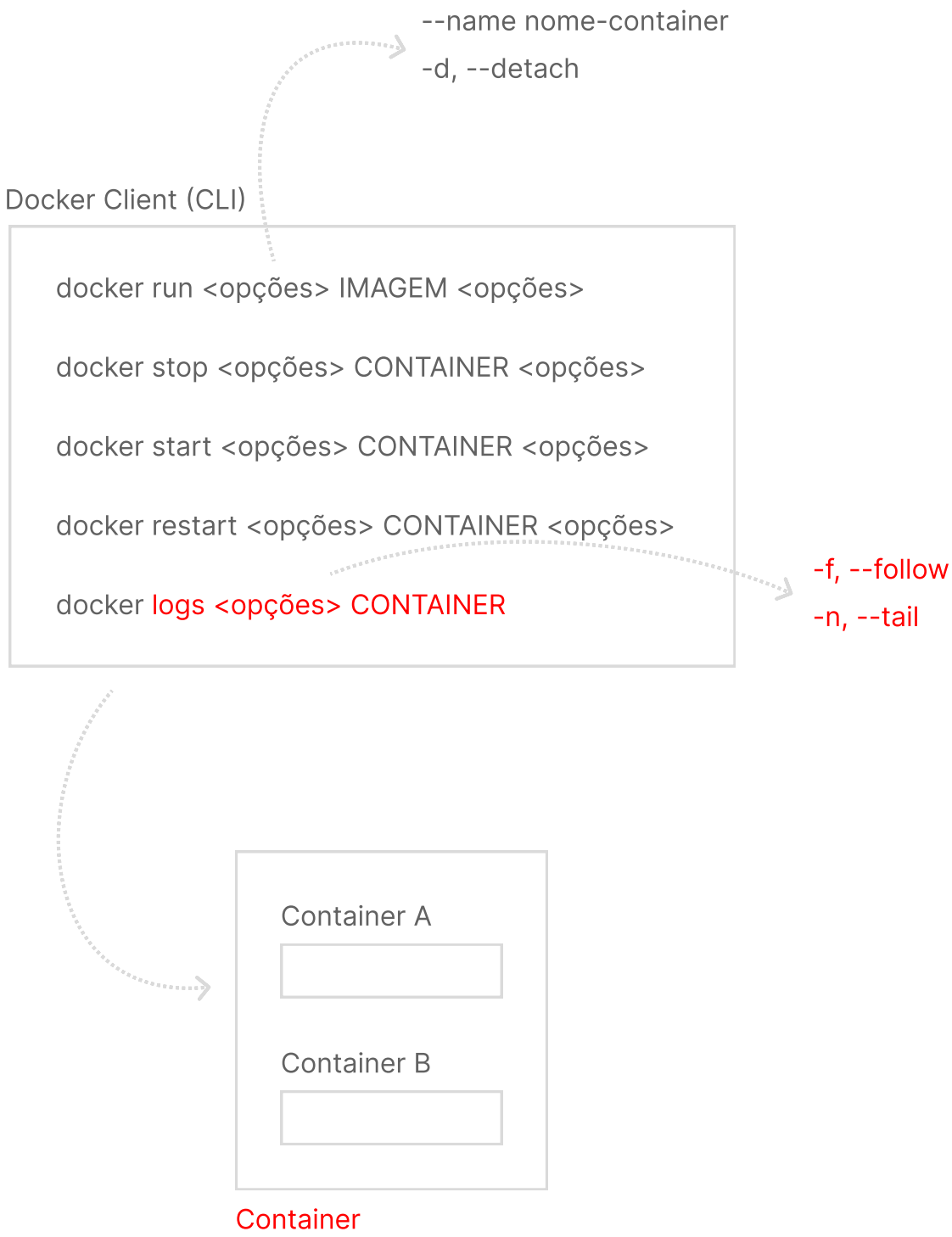
Nomeando o container



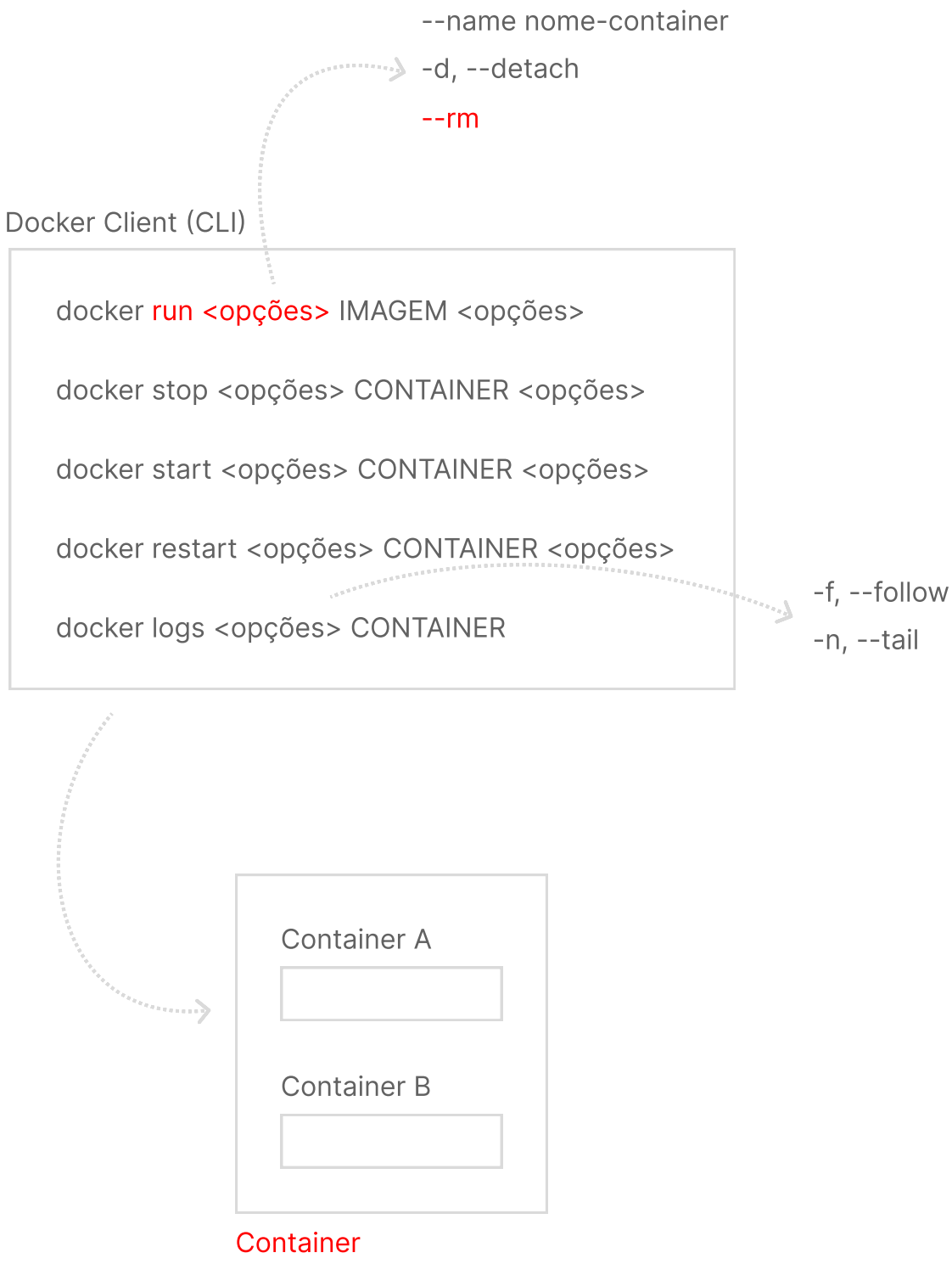
Execução do container em segundo plano



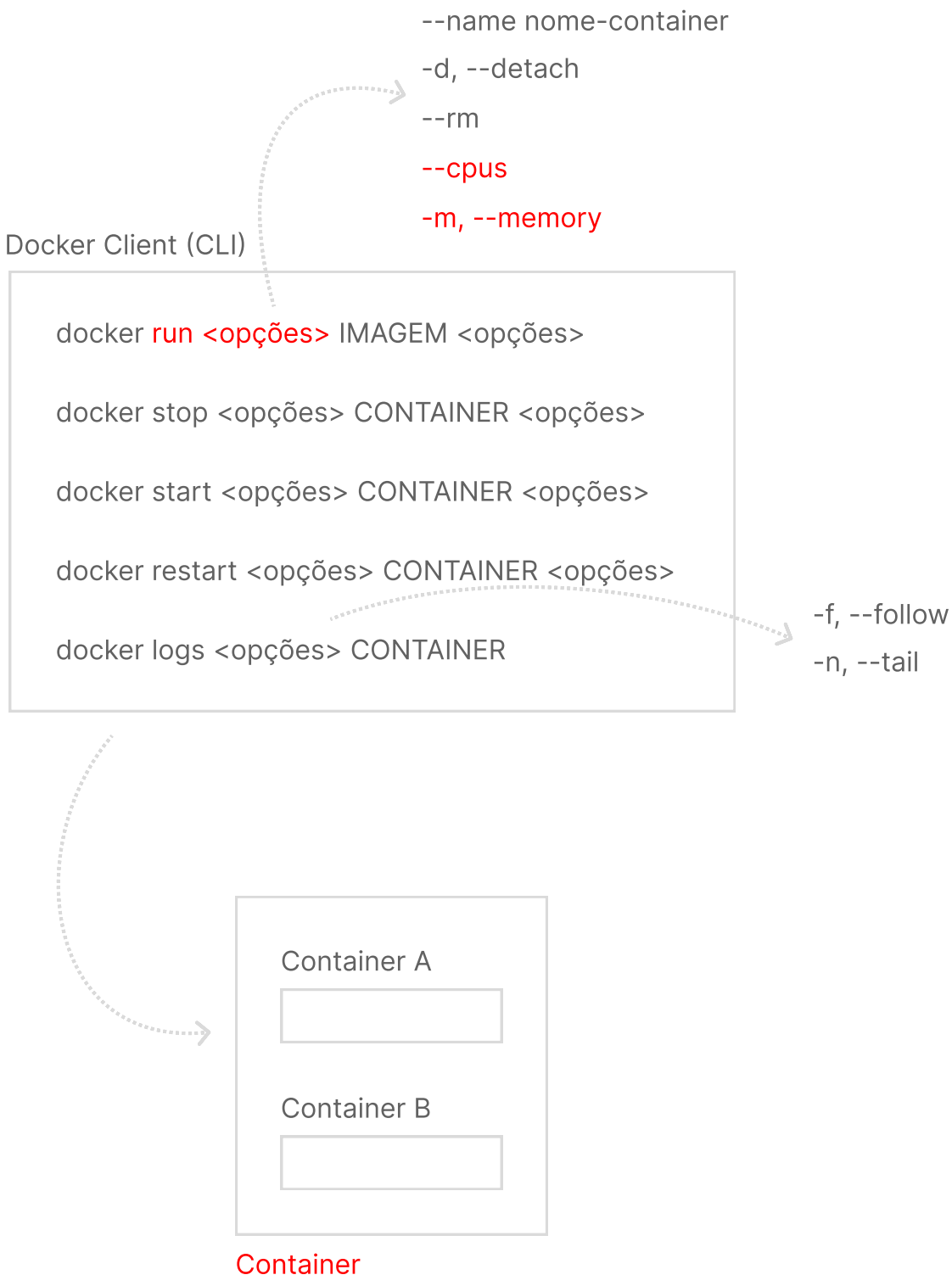
Docker logs


--name nome-container
-d, --detach
Docker Client (CLI)
docker run <opções> IMAGEM <opções>
docker stop <opções> CONTAINER <opções>
docker start <opções> CONTAINER <opções>
docker restart <opções> CONTAINER <opções>
docker logs <opções> CONTAINER
-f, --follow
-n, --tail
Container A
Container B
Container

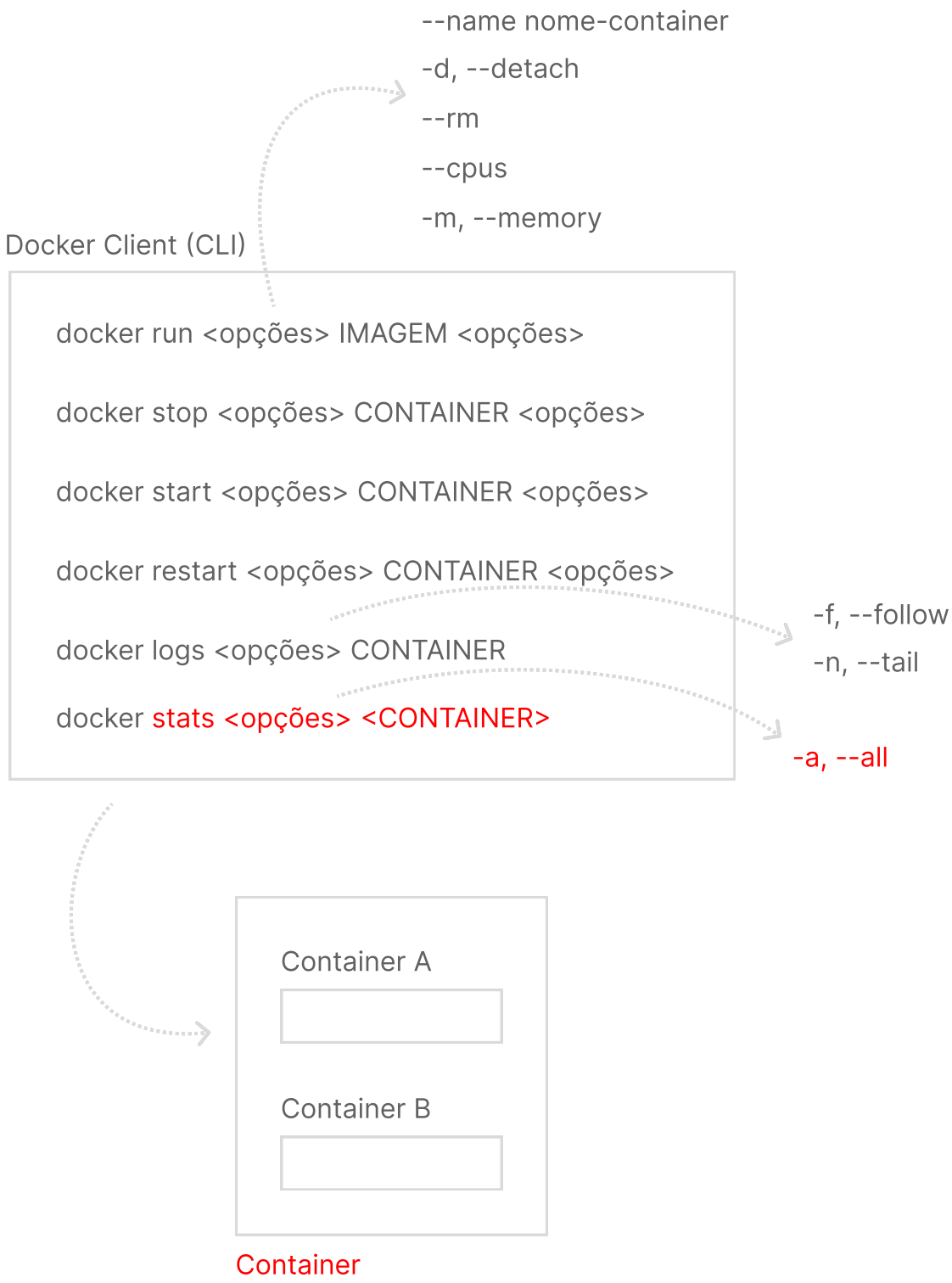
Removendo automaticamente o container após a execução



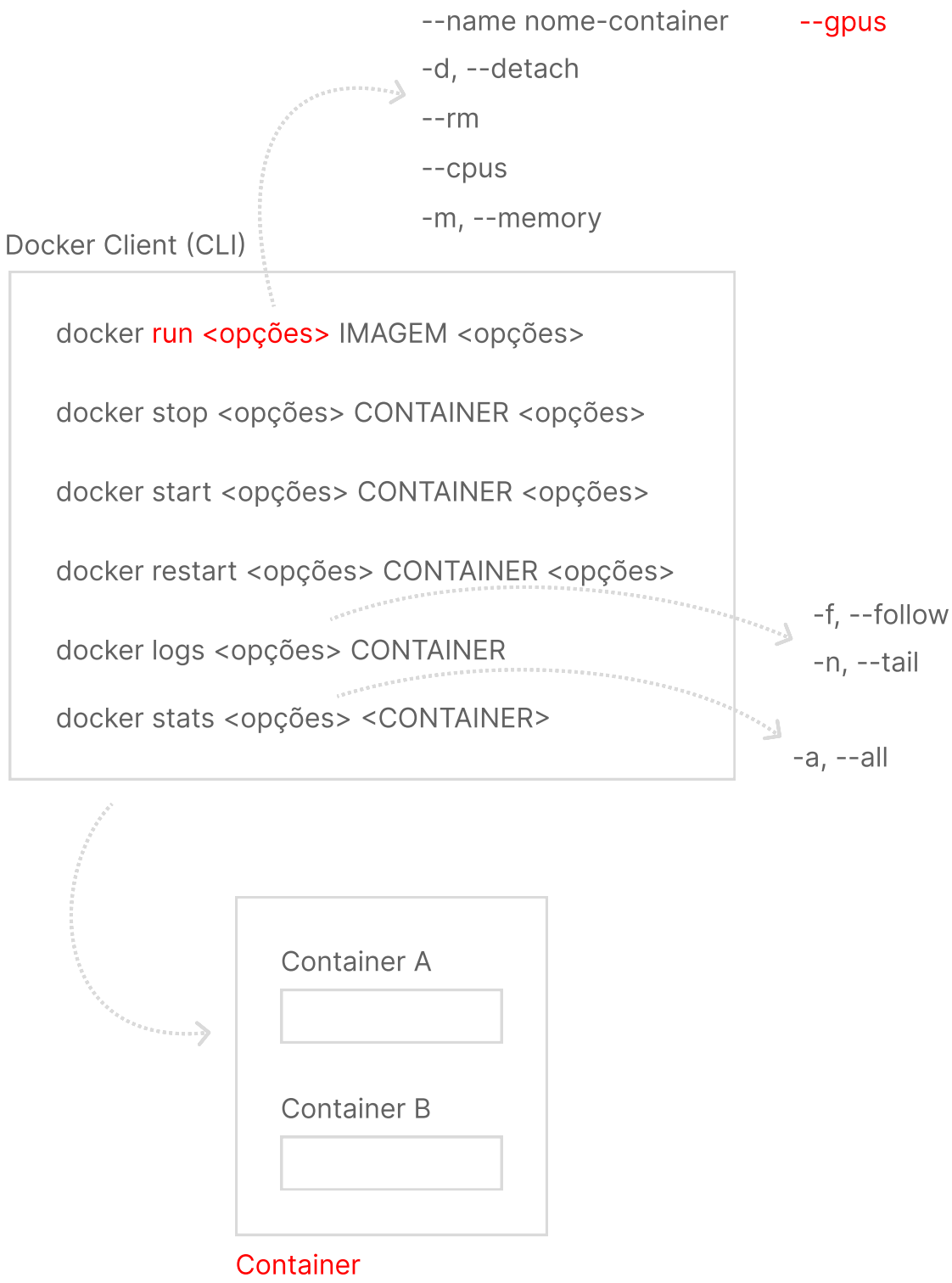
Definindo capacidade computacional (memória e cpu)



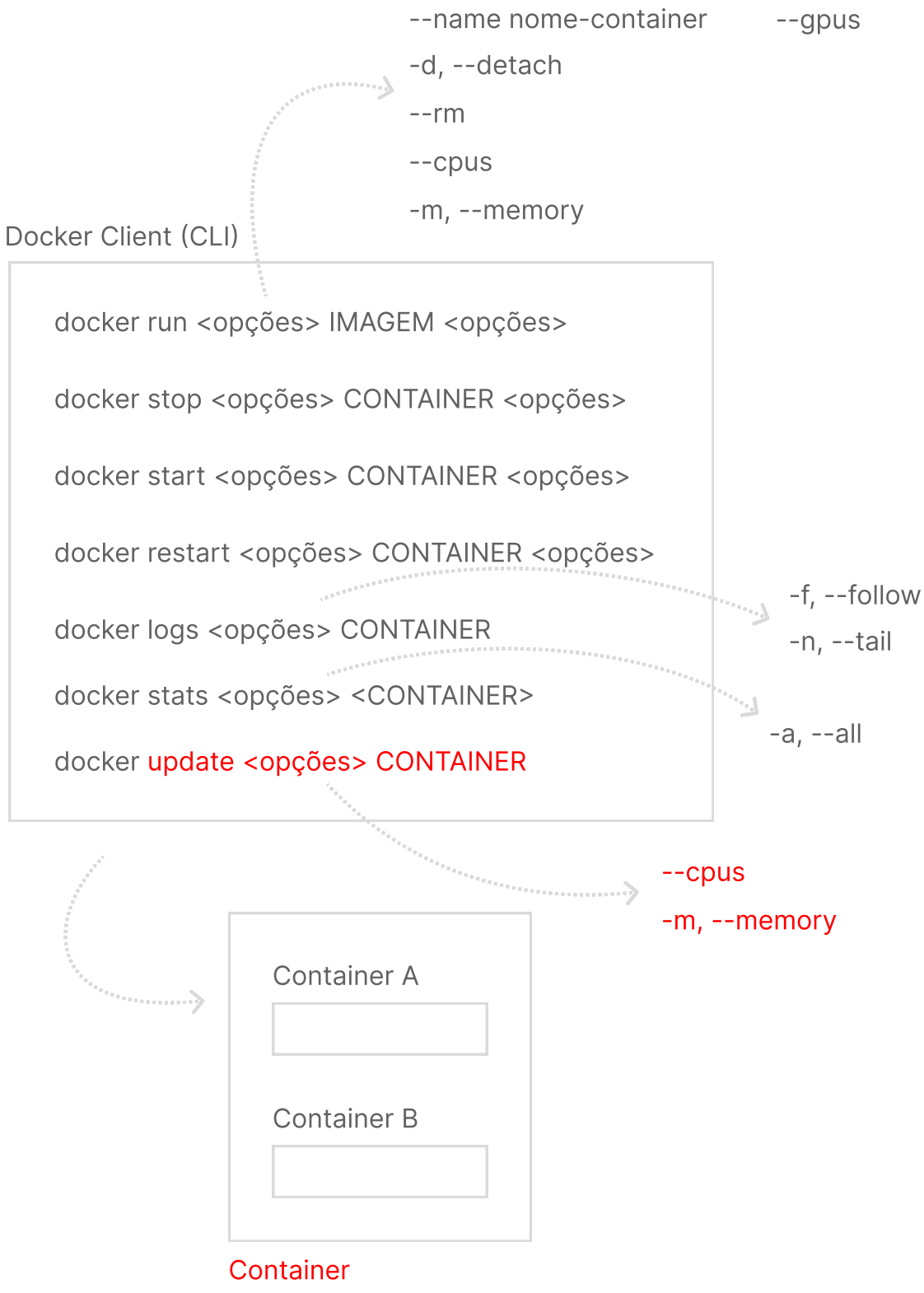
Verificando as estatísticas de um container



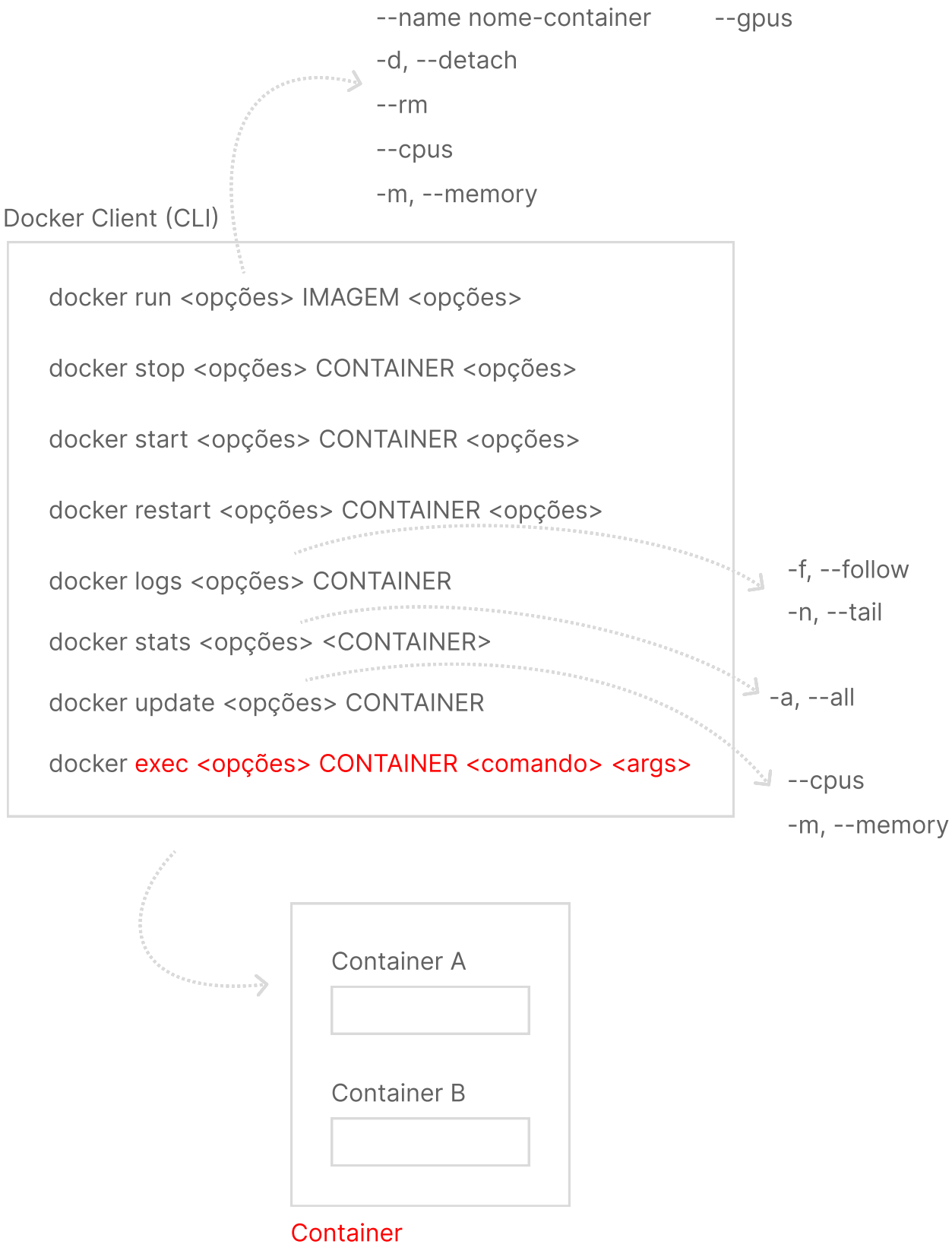
Habilitando o uso a GPUs



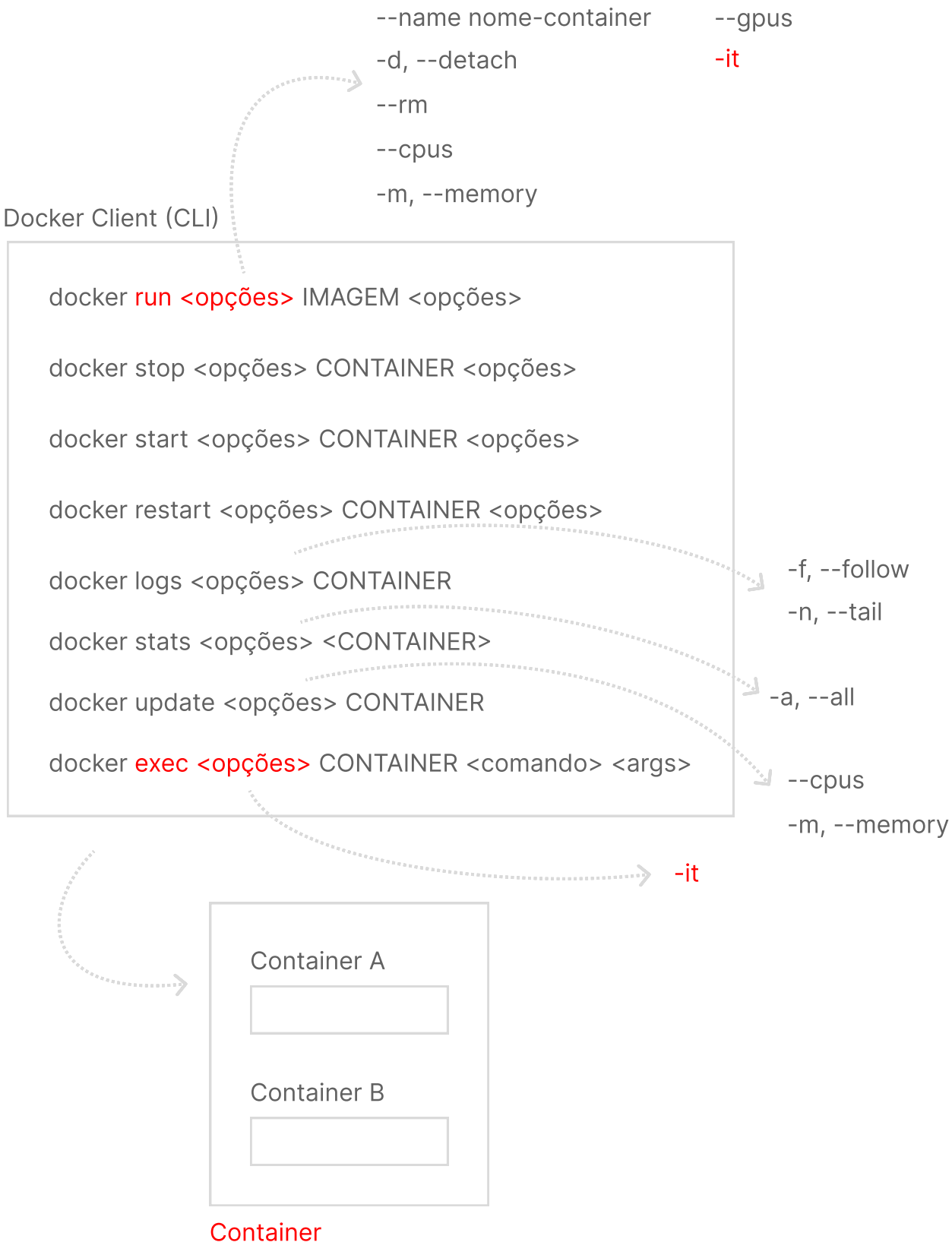
Atualizando a capacidade computacional de um container



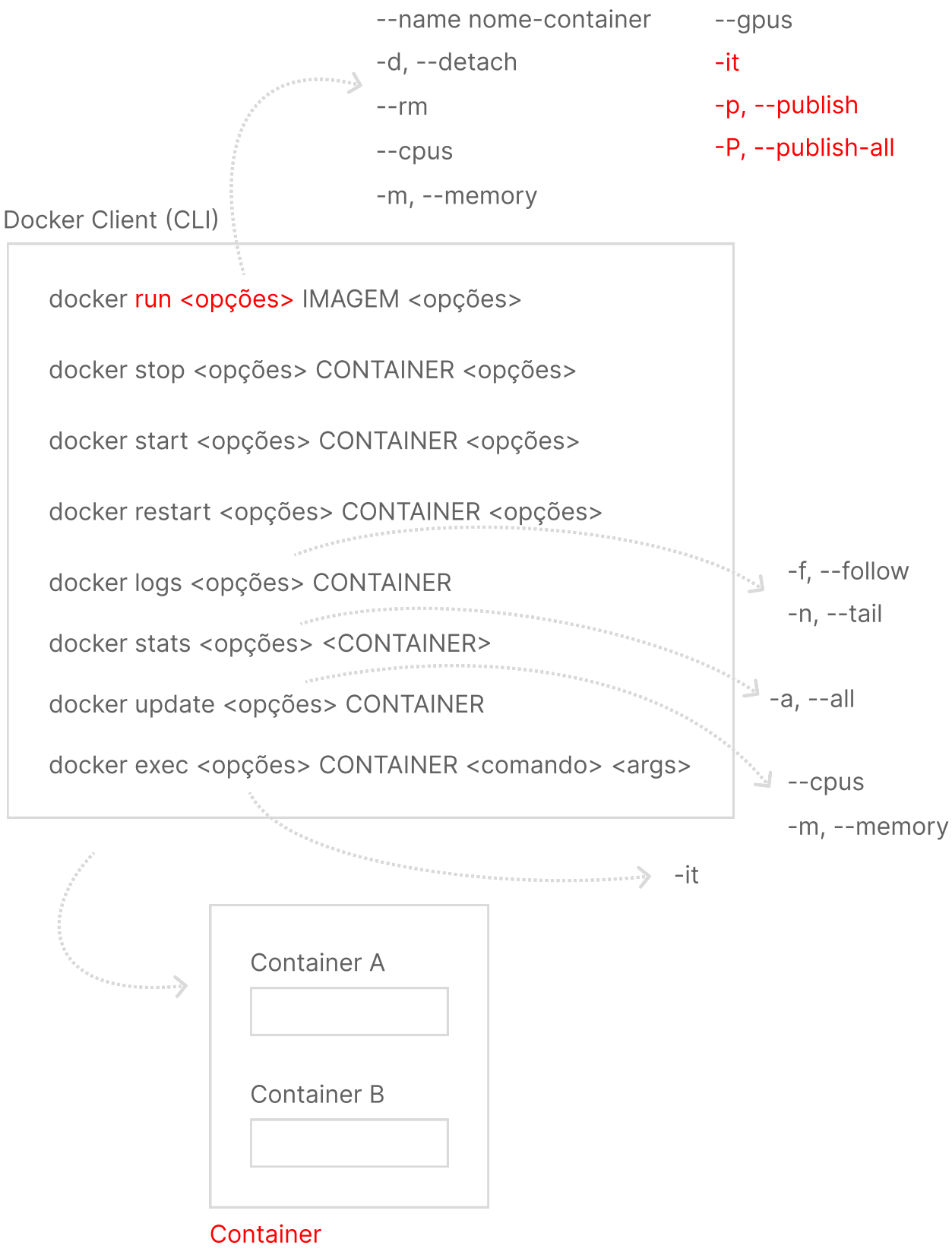
Executando comandos em um container



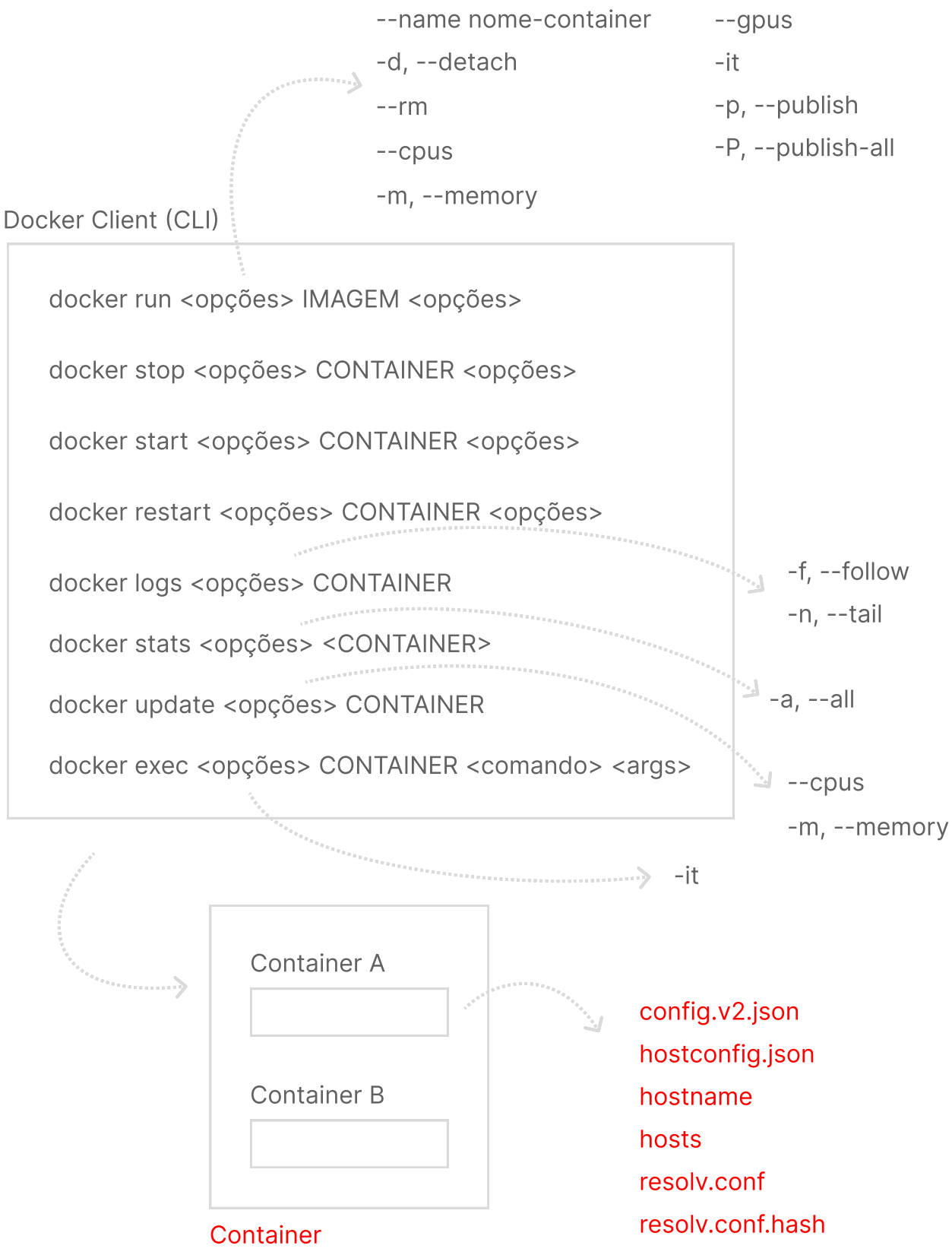
Acessando o terminal interativo



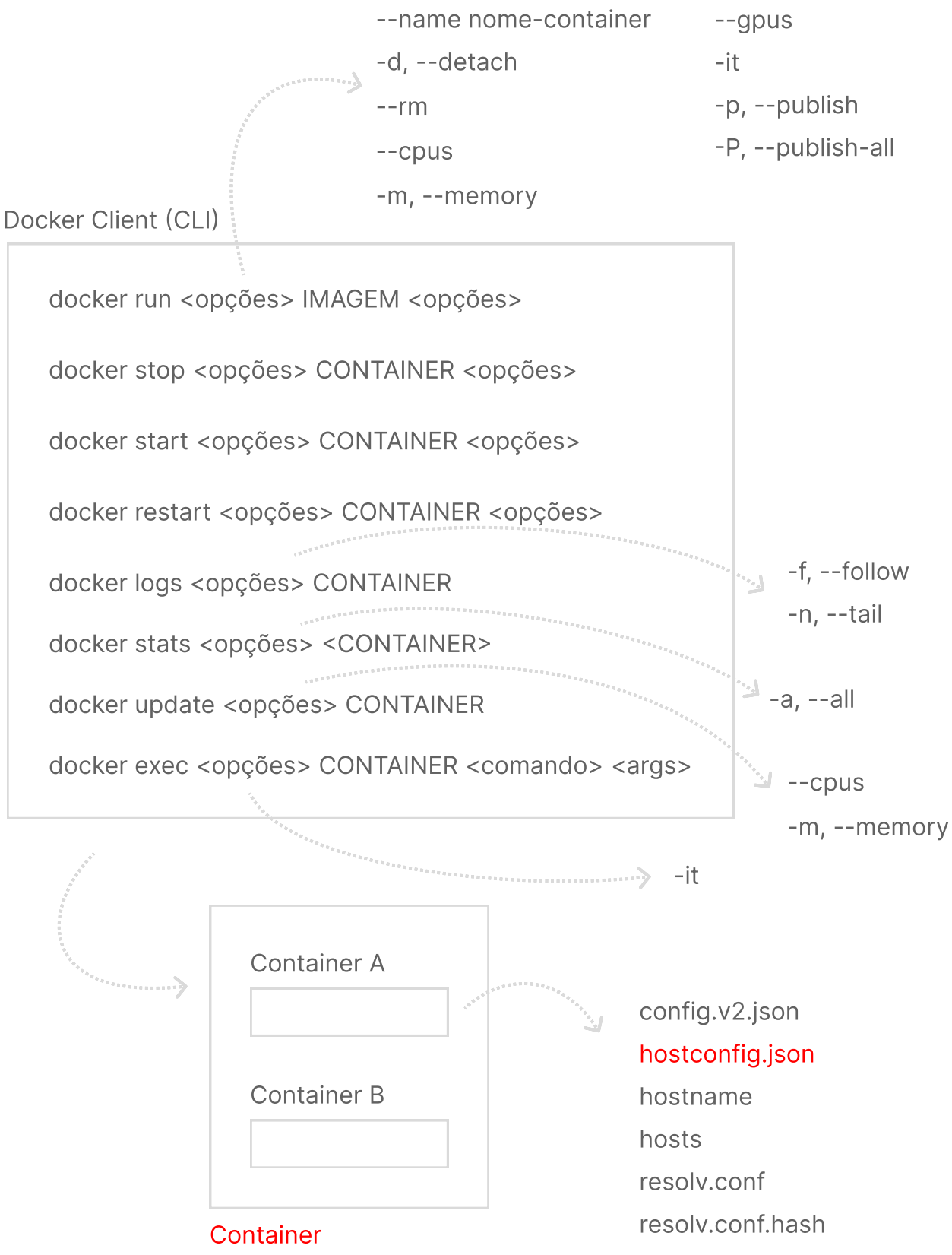
Expondo portas



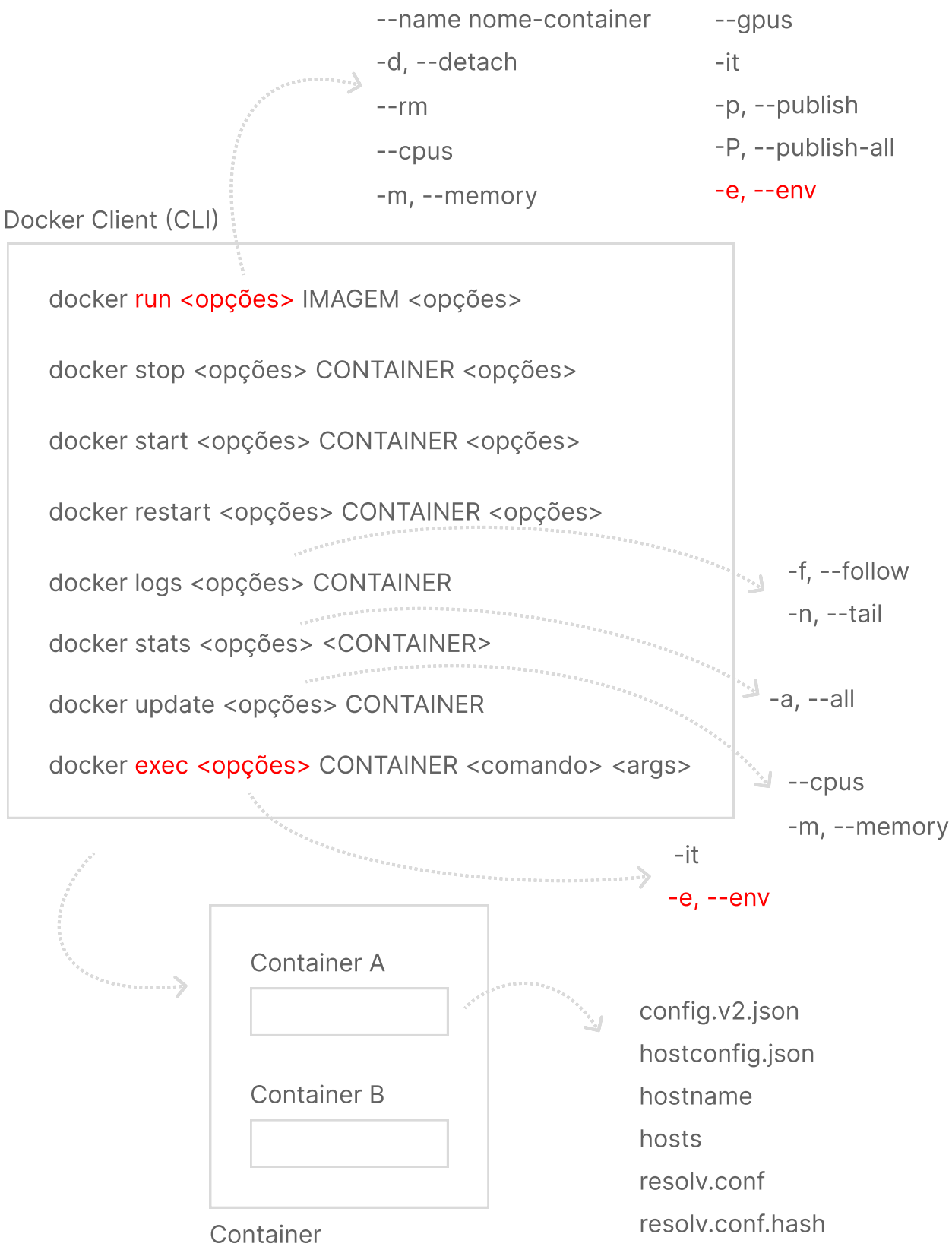
Localização dos arquivos de configuração do container



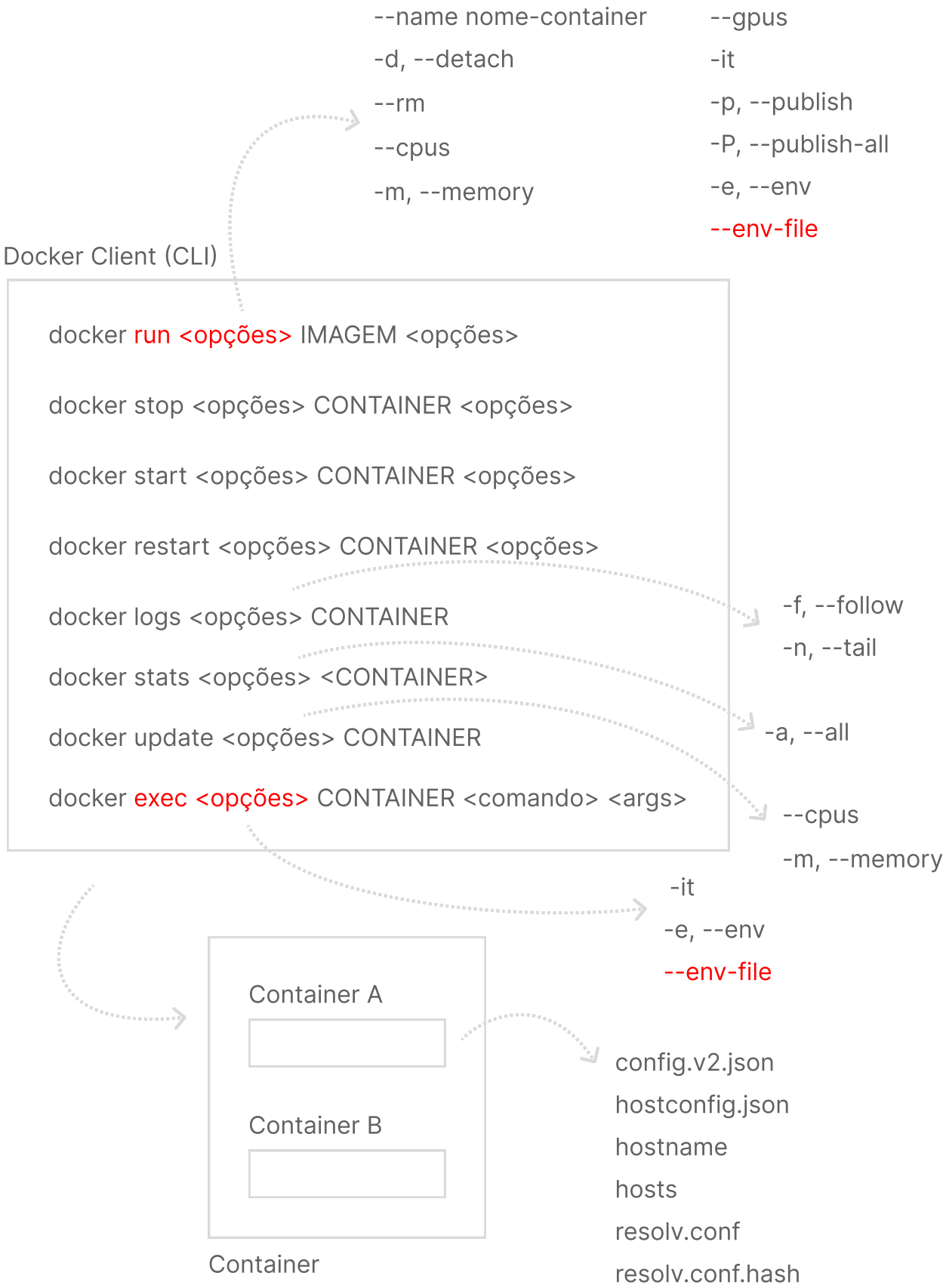
Alterando configurações do container via hostconfig



Definindo variáveis de ambiente (via linha de comando)



Definindo variáveis de ambiente (via arquivo externo)



Transferindo arquivos do host para o container

--name nome-container
-d, --detach
--rm
--cpus

-m, --memory
--gpus
-it
-p, --publish

-P, --publish-all
-e, --env
--env-file

```
docker run <opções> IMAGEM <opções>
docker stop <opções> CONTAINER <opções>
docker start <opções> CONTAINER <opções>
docker restart <opções> CONTAINER <opções>
docker logs <opções> CONTAINER
docker stats <opções> <CONTAINER>
docker update <opções> CONTAINER
docker exec <opções> CONTAINER <comando> <args>
docker cp arq_host CONTAINER:dest_container
```

-f, --follow
-n, --tail

-a, --all

--cpus
-m, --memory

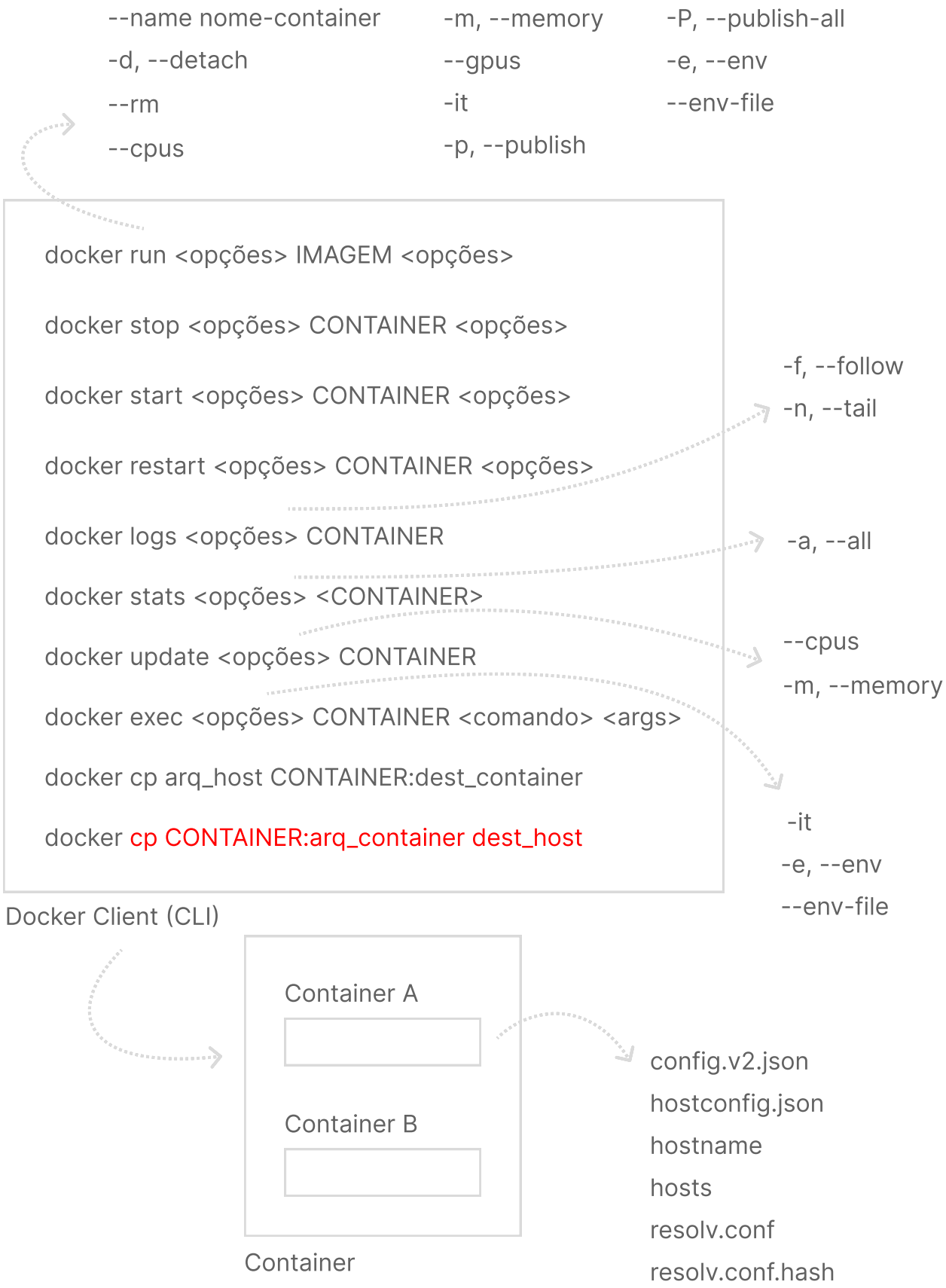
-it
-e, --env
--env-file

Docker Client (CLI)

Container

config.v2.json
hostconfig.json
hostname
hosts
resolv.conf
resolv.conf.hash

Transferindo arquivos do container para o host



Definindo o diretório de trabalho

--name nome-container
-d, --detach
--rm
--cpus
-m, --memory
--gpus
-it
-p, --publish
-P, --publish-all
-e, --env
--env-file
-w, --workdir

```
docker run <opções> IMAGEM <opções>

docker stop <opções> CONTAINER <opções>

docker start <opções> CONTAINER <opções>

docker restart <opções> CONTAINER <opções>

docker logs <opções> CONTAINER

docker stats <opções> <CONTAINER>

docker update <opções> CONTAINER

docker exec <opções> CONTAINER <comando> <args>

docker cp arq_host CONTAINER:dest_container

docker cp CONTAINER:arq_container dest_host
```

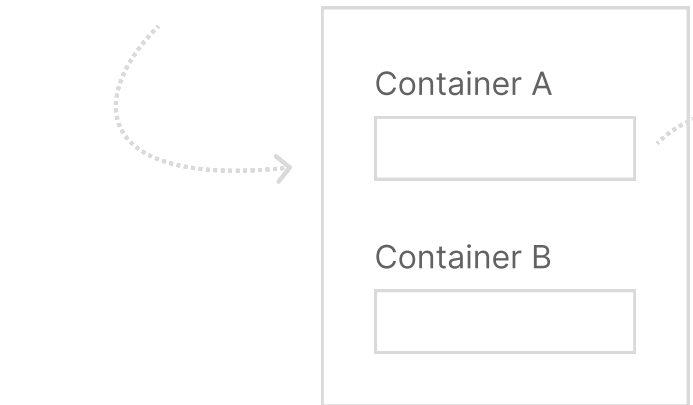
-f, --follow
-n, --tail

-a, --all

--cpus
-m, --memory

-it
-e, --env
--env-file

Docker Client (CLI)



Container

config.v2.json
hostconfig.json
hostname
hosts
resolv.conf
resolv.conf.hash

Mapeando volumes parte 1

--name nome-container
-d, --detach
--rm
--cpus
-m, --memory
--gpus
-it
-p, --publish
-P, --publish-all
-e, --env
--env-file
-w, --workdir
-v

docker run <opções> IMAGEM <opções>

docker stop <opções> CONTAINER <opções>

docker start <opções> CONTAINER <opções>

docker restart <opções> CONTAINER <opções>

docker logs <opções> CONTAINER

docker stats <opções> <CONTAINER>

docker update <opções> CONTAINER

docker exec <opções> CONTAINER <comando> <args>

docker cp arq_host CONTAINER:dest_container

docker cp CONTAINER:arq_container dest_host

-f, --follow
-n, --tail

-a, --all

--cpus
-m, --memory

-it
-e, --env
--env-file

Docker Client (CLI)

Container A

Container B

Container

config.v2.json
hostconfig.json
hostname
hosts
resolv.conf
resolv.conf.hash

Mapeando volumes parte 2

--name nome-container
-d, --detach
--rm
--cpus
-m, --memory
--gpus
-it
-p, --publish
-P, --publish-all
-e, --env
--env-file
-w, --workdir
-v
--mount

```
docker run <opções> IMAGEM <opções>

docker stop <opções> CONTAINER <opções>

docker start <opções> CONTAINER <opções>

docker restart <opções> CONTAINER <opções>

docker logs <opções> CONTAINER

docker stats <opções> <CONTAINER>

docker update <opções> CONTAINER

docker exec <opções> CONTAINER <comando> <args>

docker cp arq_host CONTAINER:dest_container

docker cp CONTAINER:arq_container dest_host
```

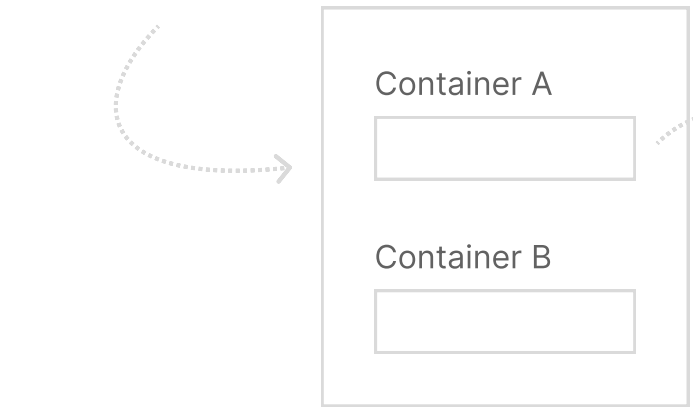
-f, --follow
-n, --tail

-a, --all

--cpus
-m, --memory

-it
-e, --env
--env-file

Docker Client (CLI)



Container

config.v2.json
hostconfig.json
hostname
hosts
resolv.conf
resolv.conf.hash

Inspecione diferenças do file system de um container em relação a imagem

--name nome-container	-m, --memory	-P, --publish-all	-v
-d, --detach	--gpus	-e, --env	--mount
--rm	-it	--env-file	
--cpus	-p, --publish	-w, --workdir	

```
docker run <opções> IMAGEM <opções>

docker stop <opções> CONTAINER <opções>

docker start <opções> CONTAINER <opções>

docker restart <opções> CONTAINER <opções>

docker logs <opções> CONTAINER

docker stats <opções> <CONTAINER>

docker update <opções> CONTAINER

docker exec <opções> CONTAINER <comando> <args>

docker cp arq_host CONTAINER:dest_container

docker cp CONTAINER:arq_container dest_host

docker diff CONTAINER
```

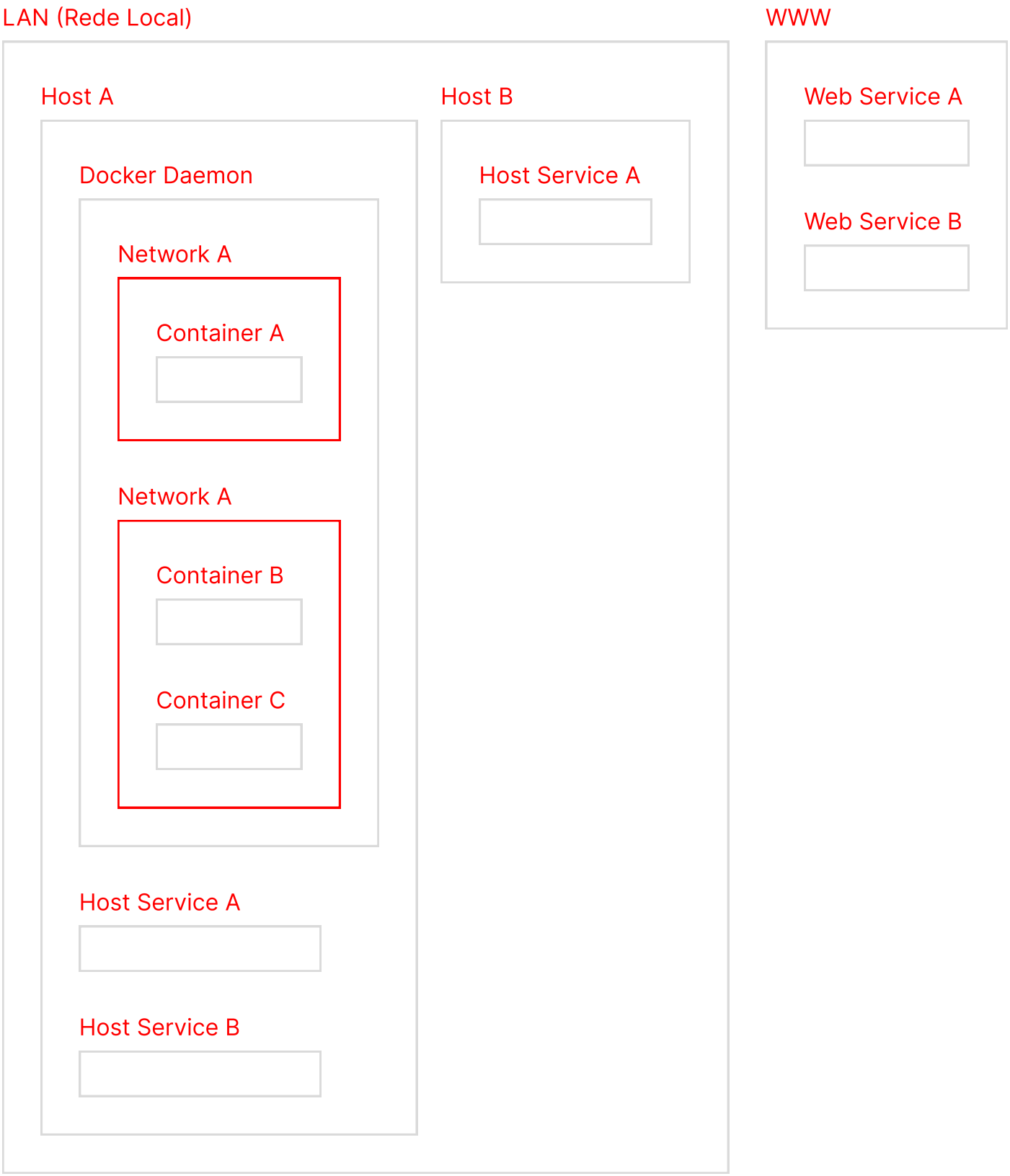
-f, --follow
-n, --tail

-a, --all

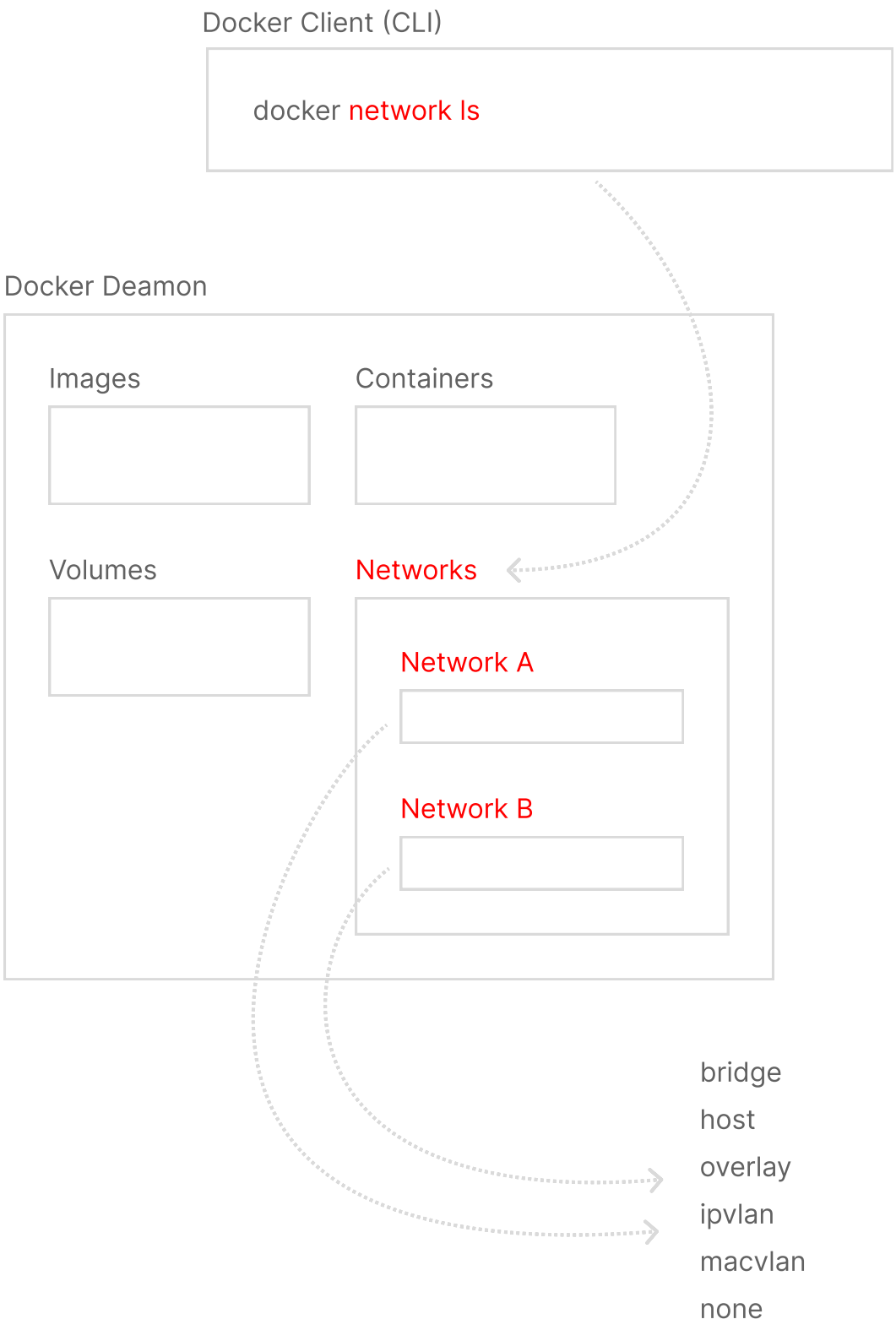
--cpus
-m, --memory

-it
-e, --env
--env-file

O que são networks

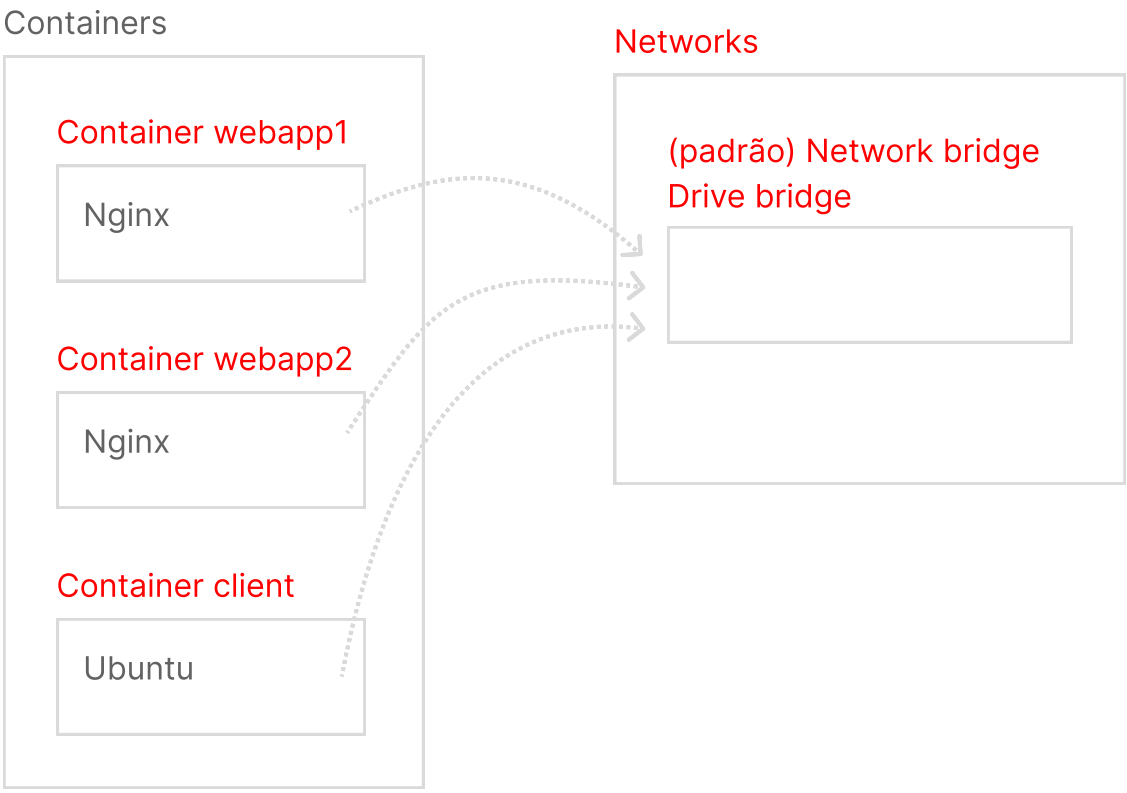


Networks e tipos de drivers

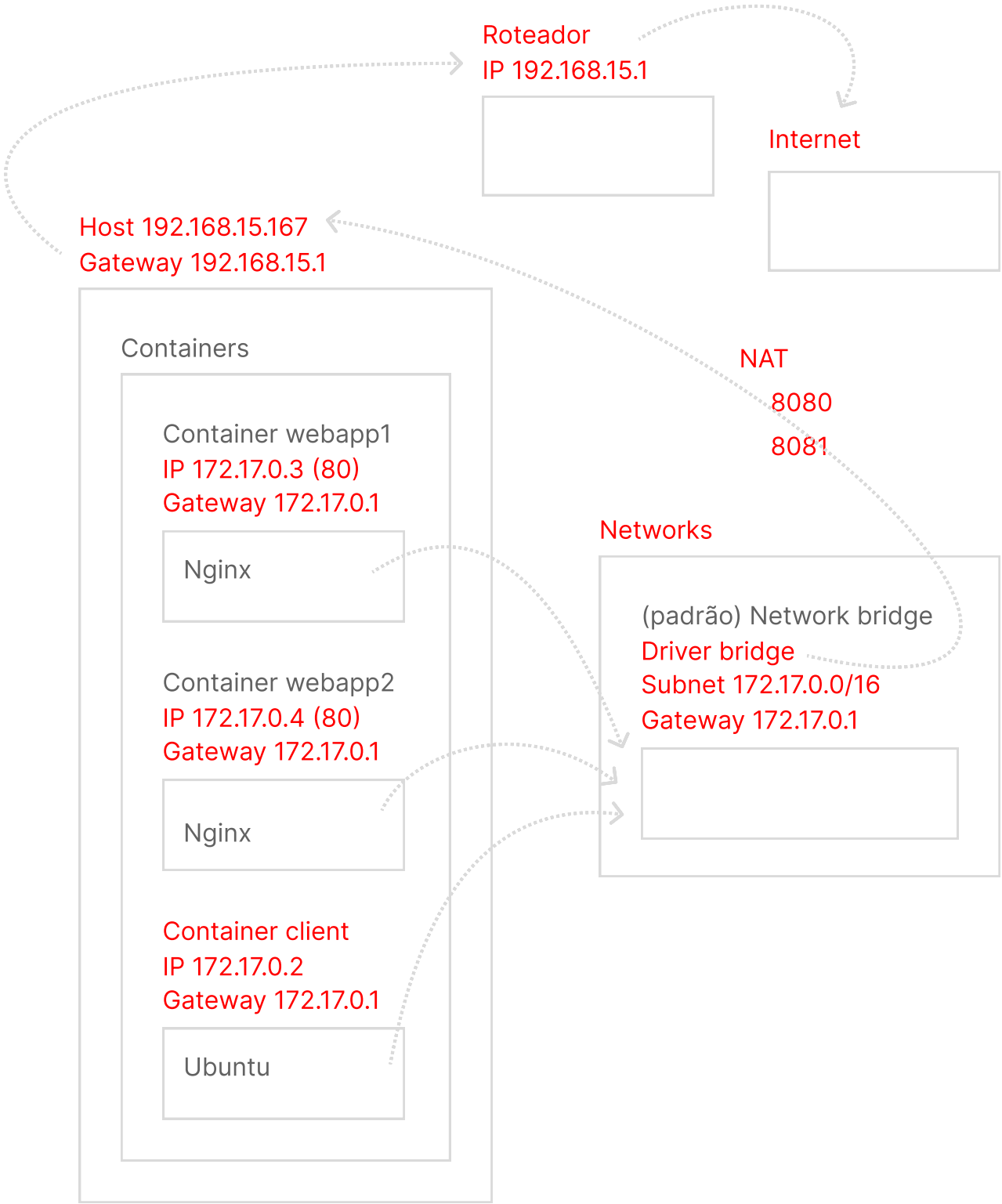


Analisando o comportamento padrão de rede do Docker (bridge) parte 1

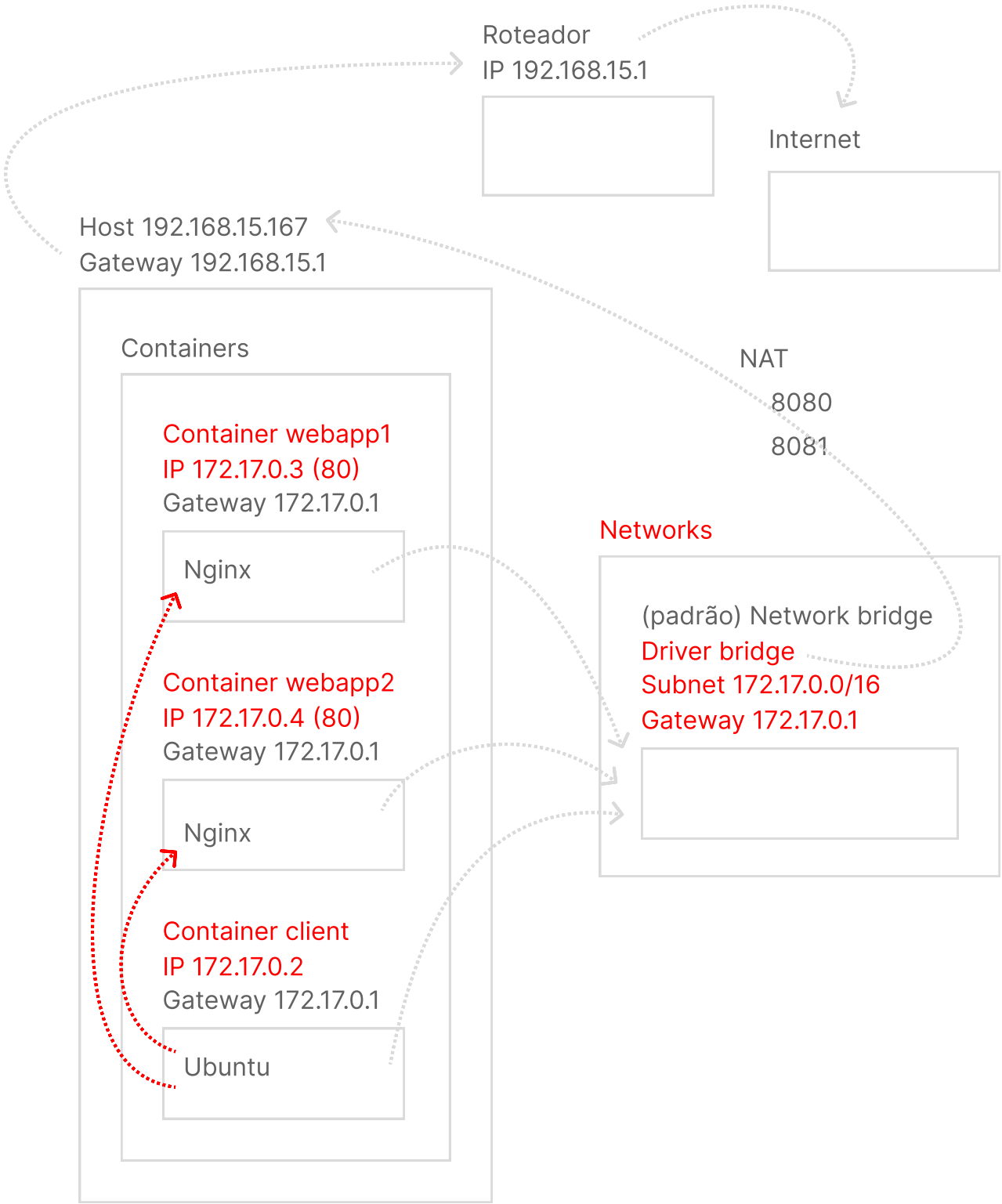
Por padrão, quando um Container é criado, ele é automaticamente adicionado a Network bridge padrão do Docker.



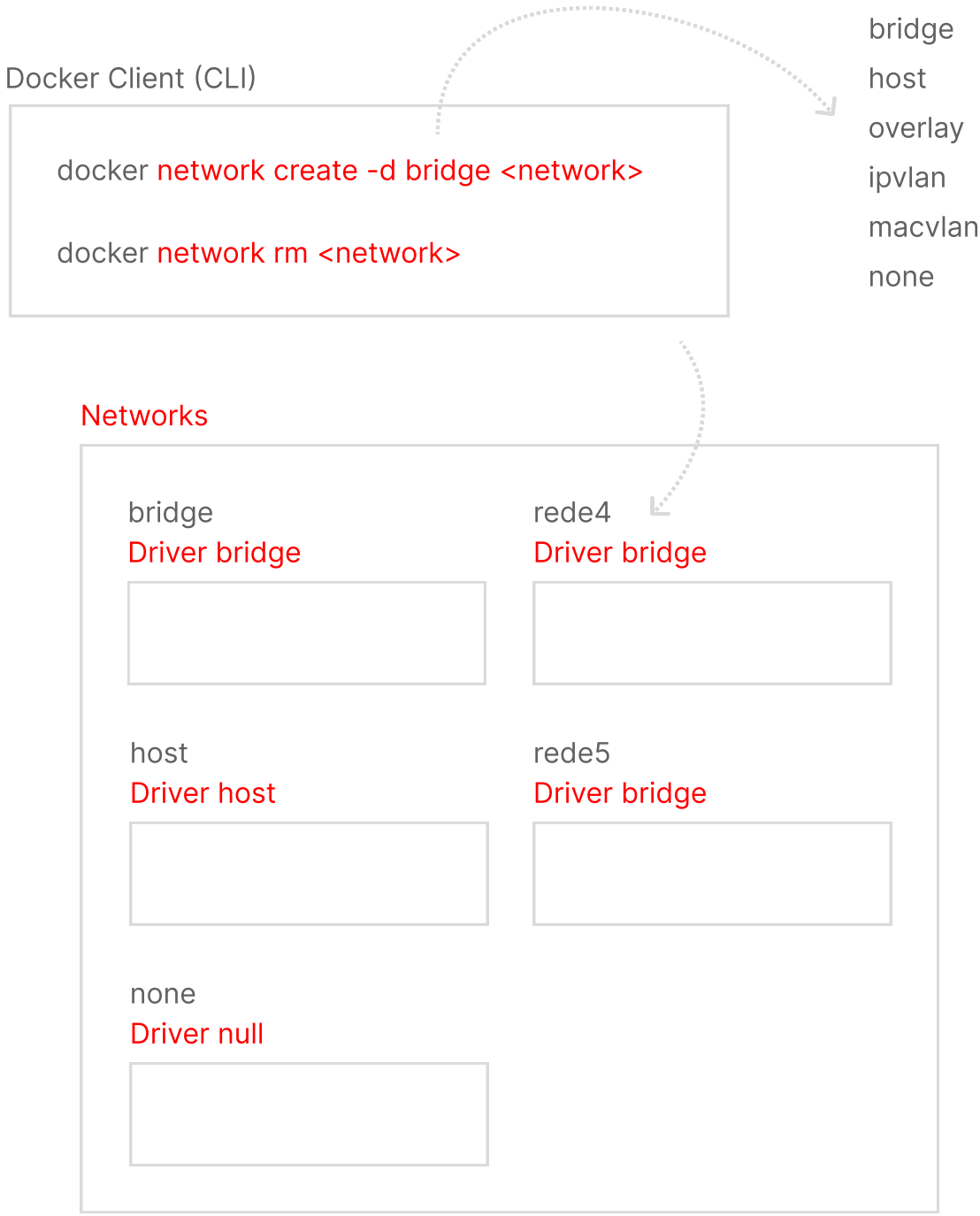
Analisando o comportamento padrão de rede do Docker (bridge) parte 2



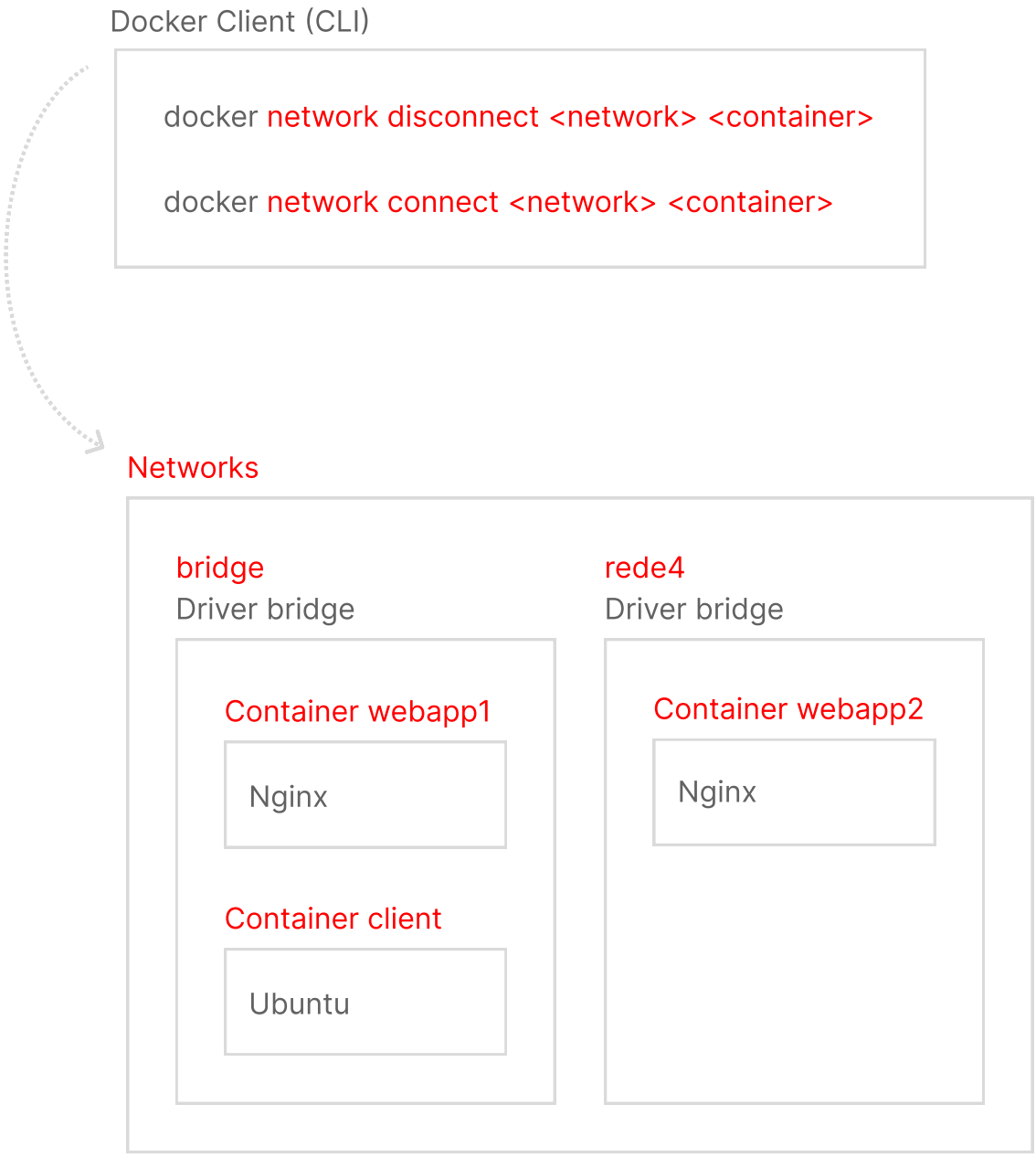
Analisando o comportamento padrão de rede do Docker (bridge) parte 3



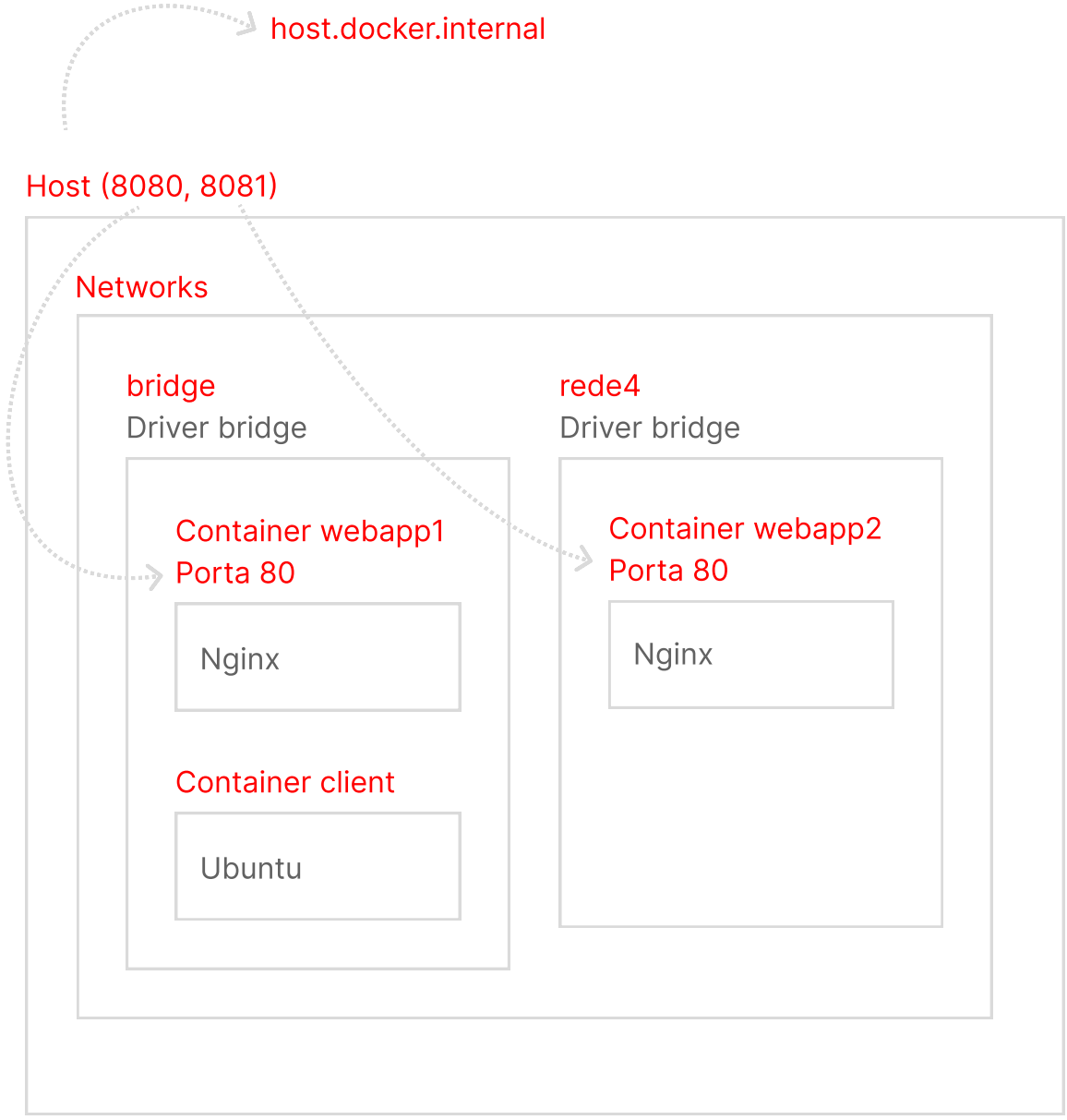
Criando e removendo networks



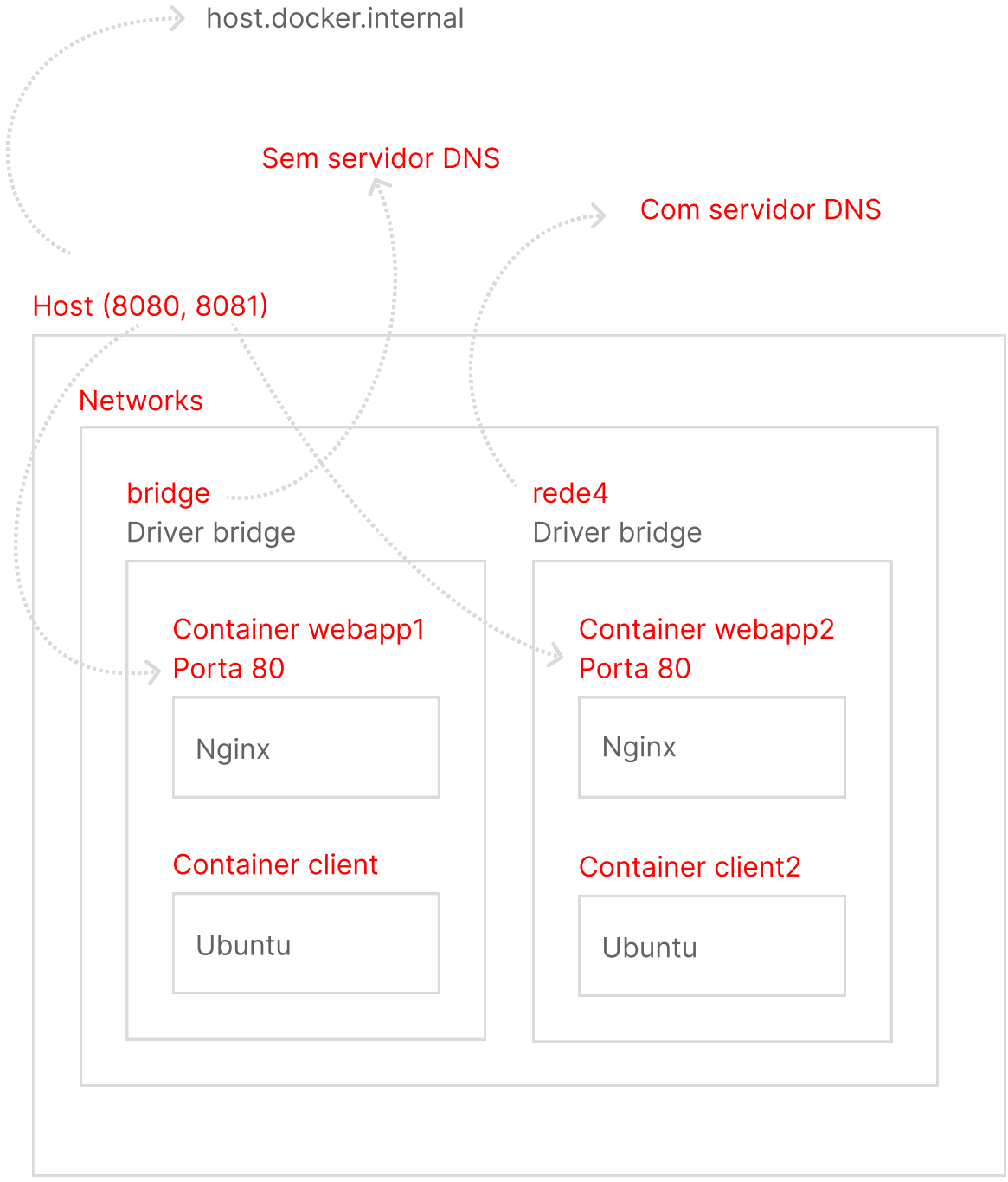
Desconectando e conectando um container a uma network



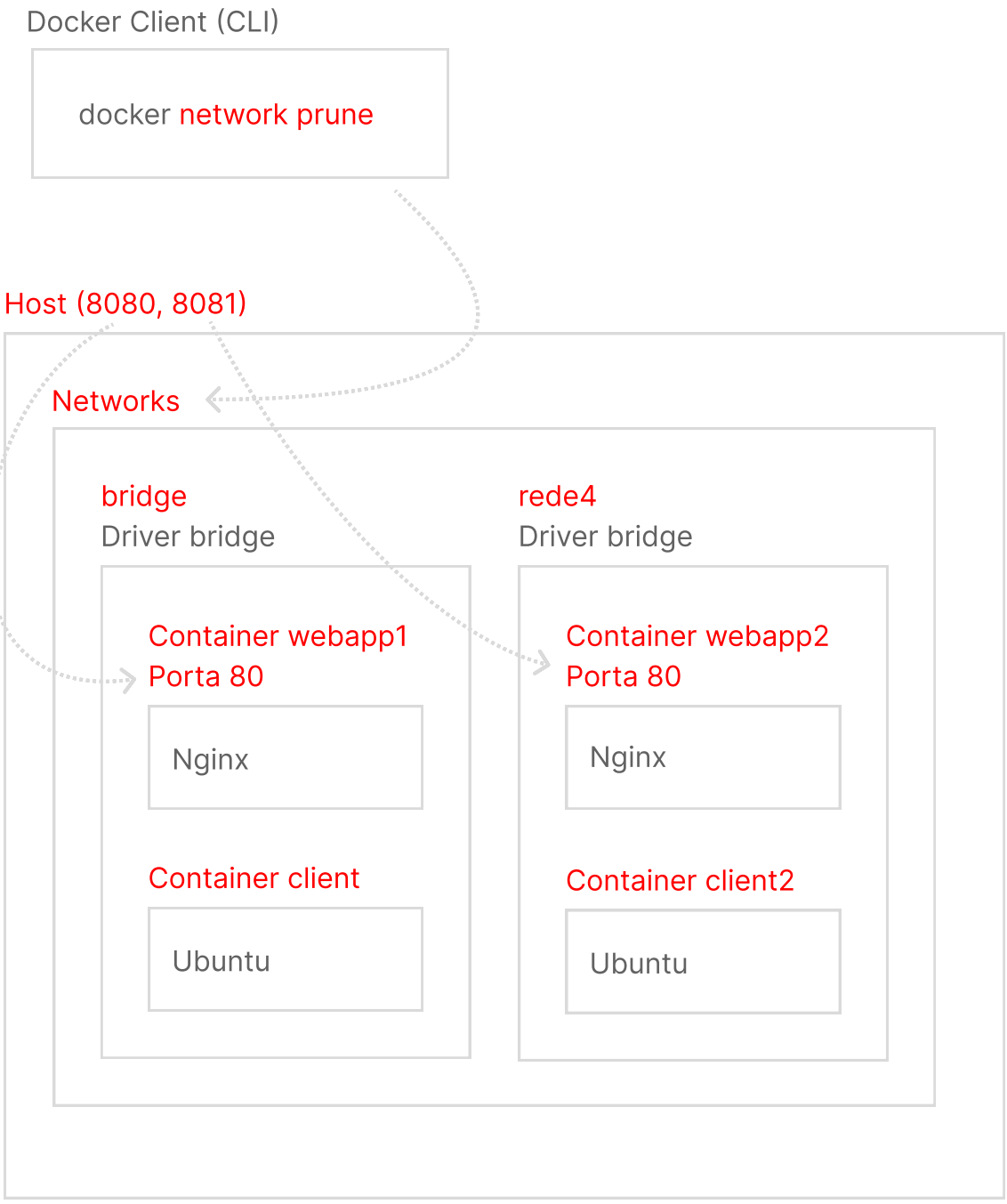
Comunicação entre containers de redes bridges diferentes via host



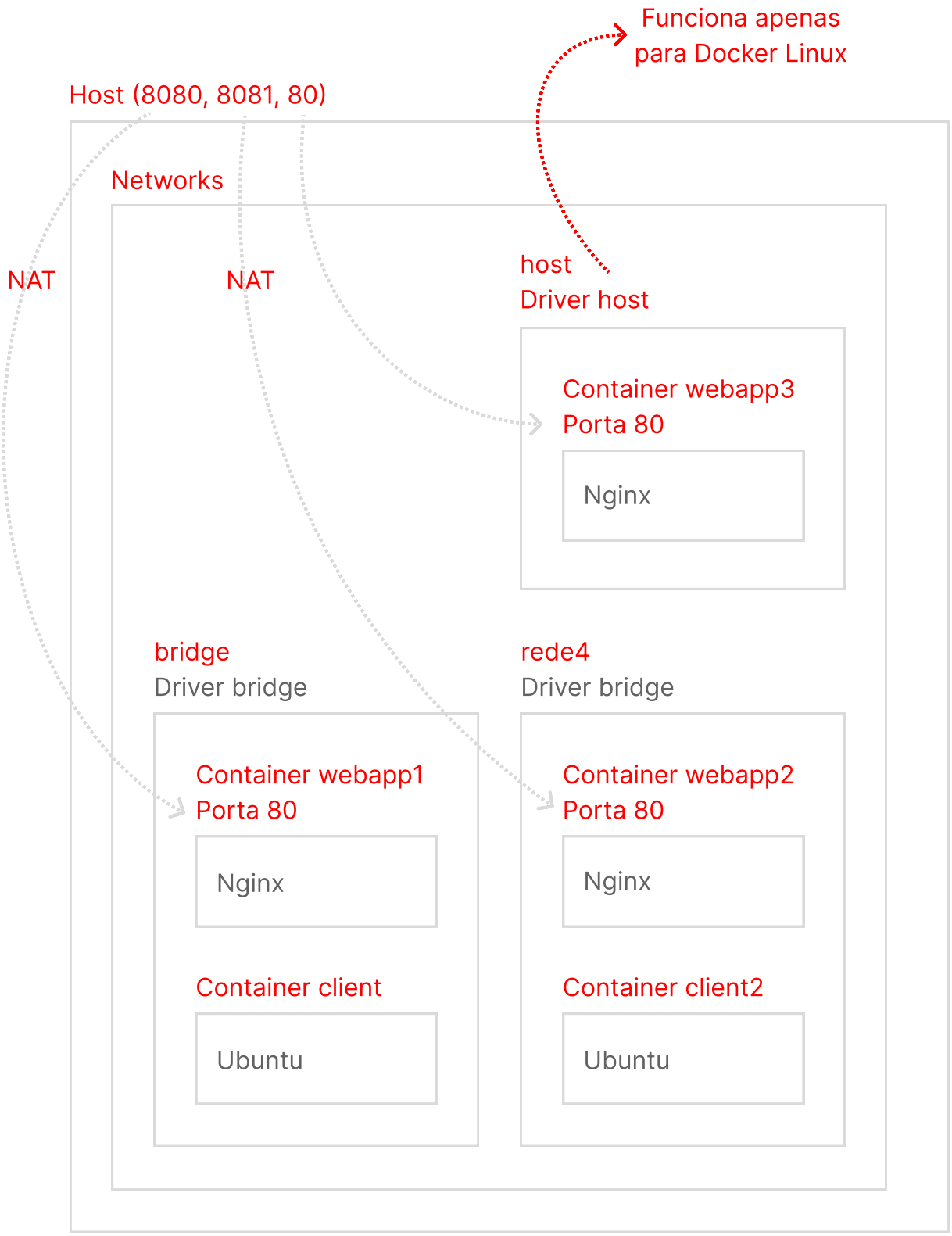
Analisando o comportamento padrão de rede do Docker (bridge) parte 4



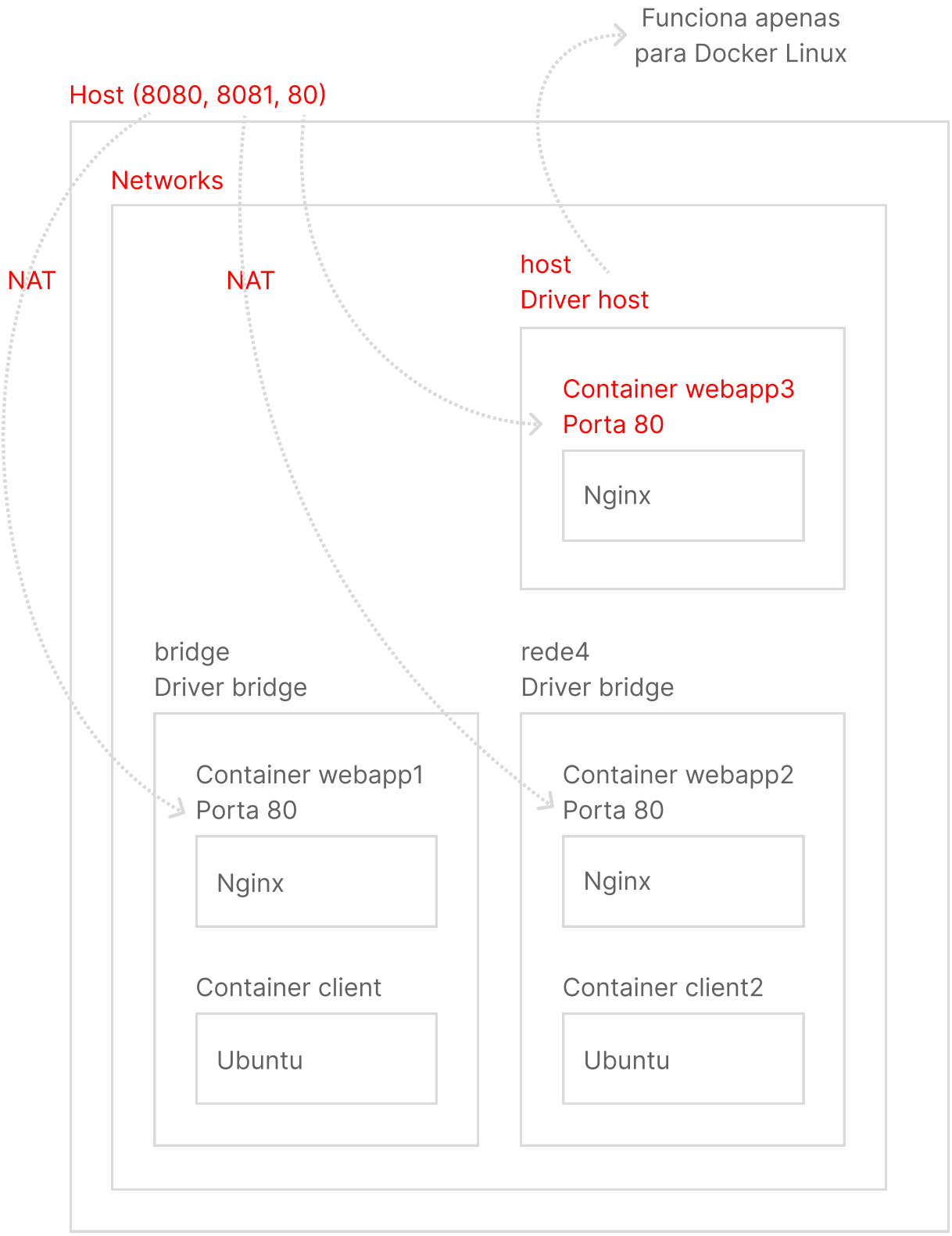
Removendo networks não utilizadas



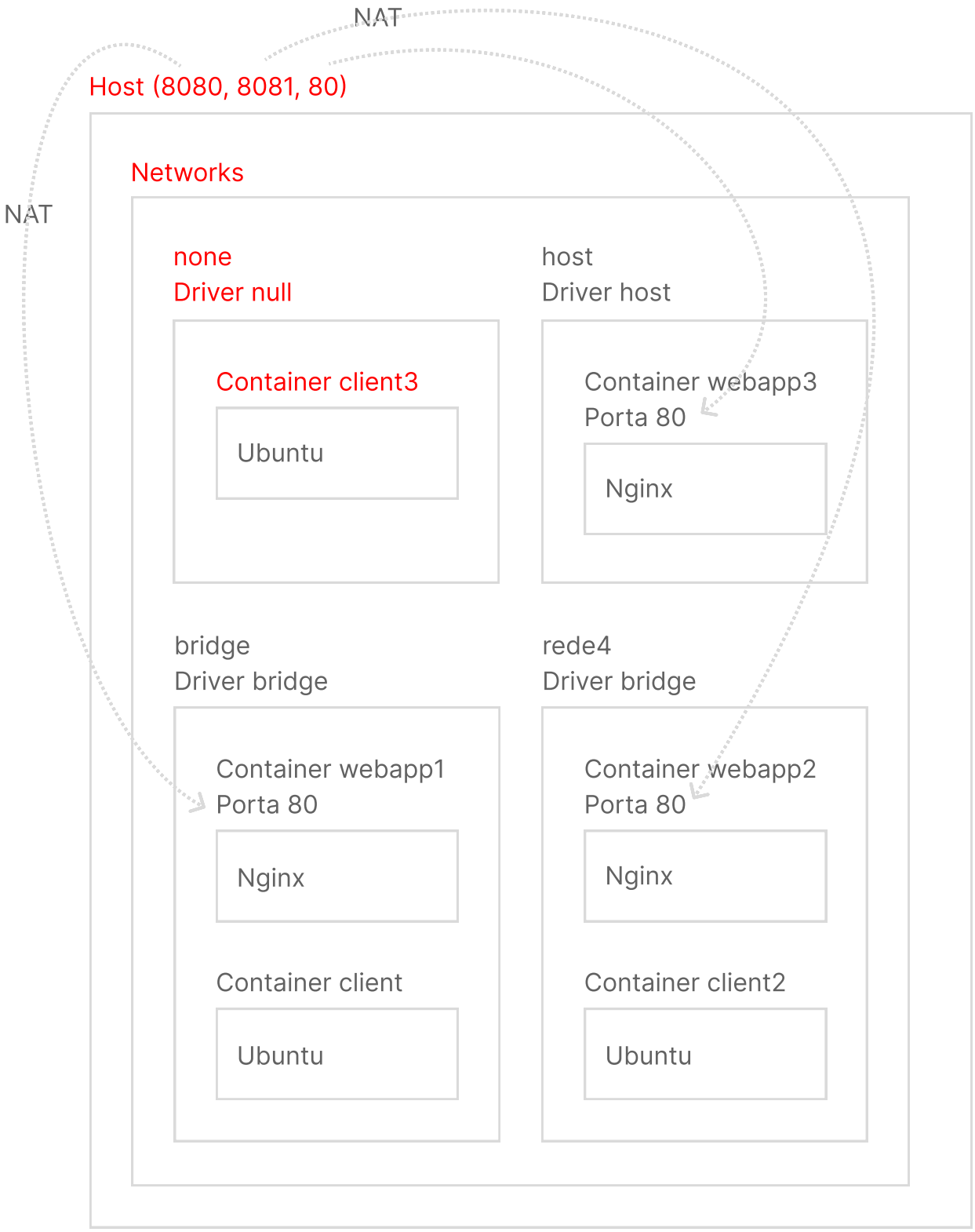
Comportamento do driver host



Vantagens na utilização de uma network com driver host



Comportamento do driver null



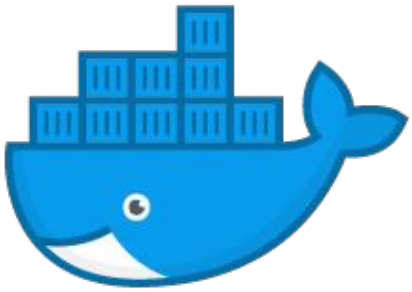
O que é o Docker Compose?

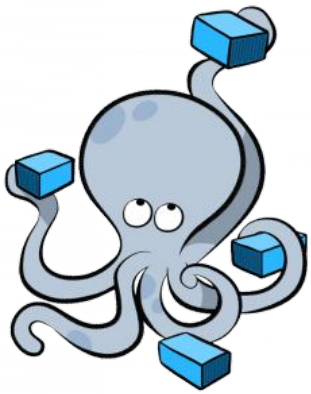
Docker Engine/Daemon

Plugin (Interface)

Docker

Docker Compose





Images | Containers

Volumes | Networks

Gerenciamento de múltiplos containers

Configurando o ambiente com Nginx, Next, Nest, PostgreSQL, Mongo, Redis e Kafka

Servidor HTTP

Nginx, Apache

Front-end

Next, React, Vue, Angular

Back-end

Nest, PHP, GoLang

Banco de Dados Relacional

PostgreSQL, MySQL, MariaDB, SQL Server, Oracle

Banco de Dados Não Relacional (NoSQL)

MongoDB, DynamoDB, Couchbase, ElasticSearch

Cache de Memória

Redis, Memcached

Mensageria

Kafka, RabbitMQ

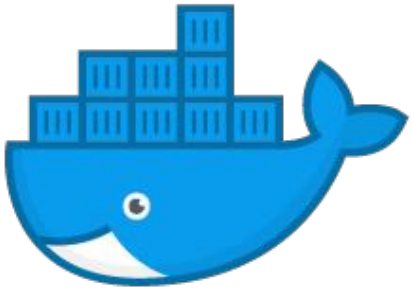
Conhecendo o arquivo docker-compose e a sintaxe YAML

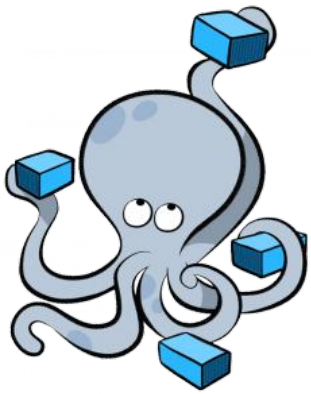
Docker Engine/Daemon

Plugin (Interface)

Docker

Docker Compose





Images

Containers

Volumes

Networks

Gerenciamento de múltiplos containers

docker-compose.yaml

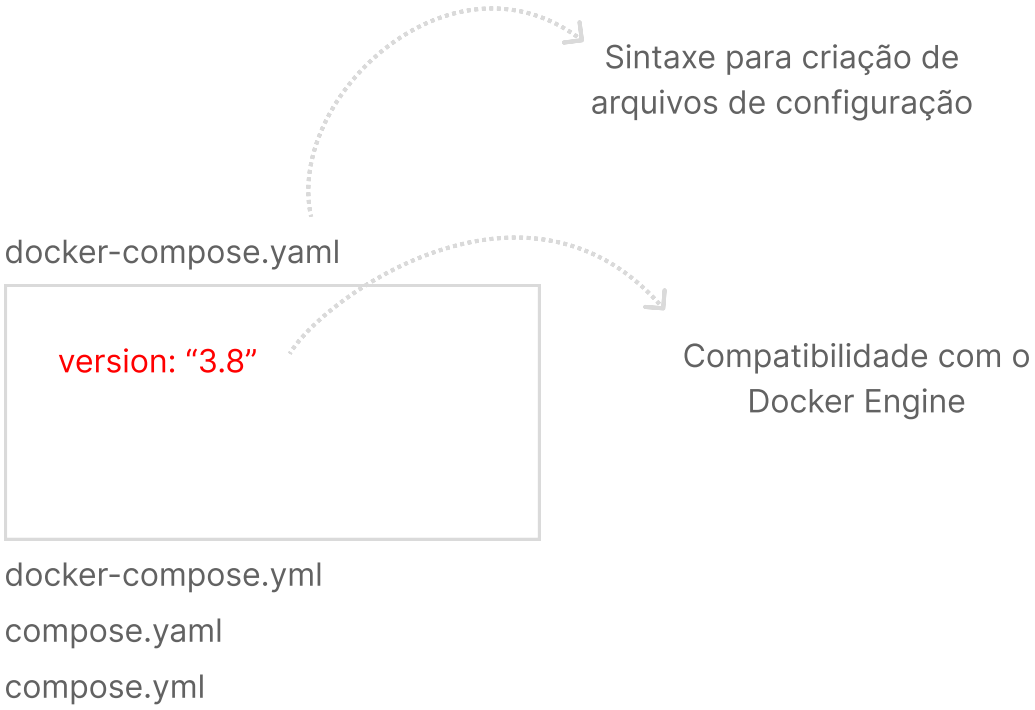
Sintaxe para criação de arquivos de configuração

docker-compose.yml

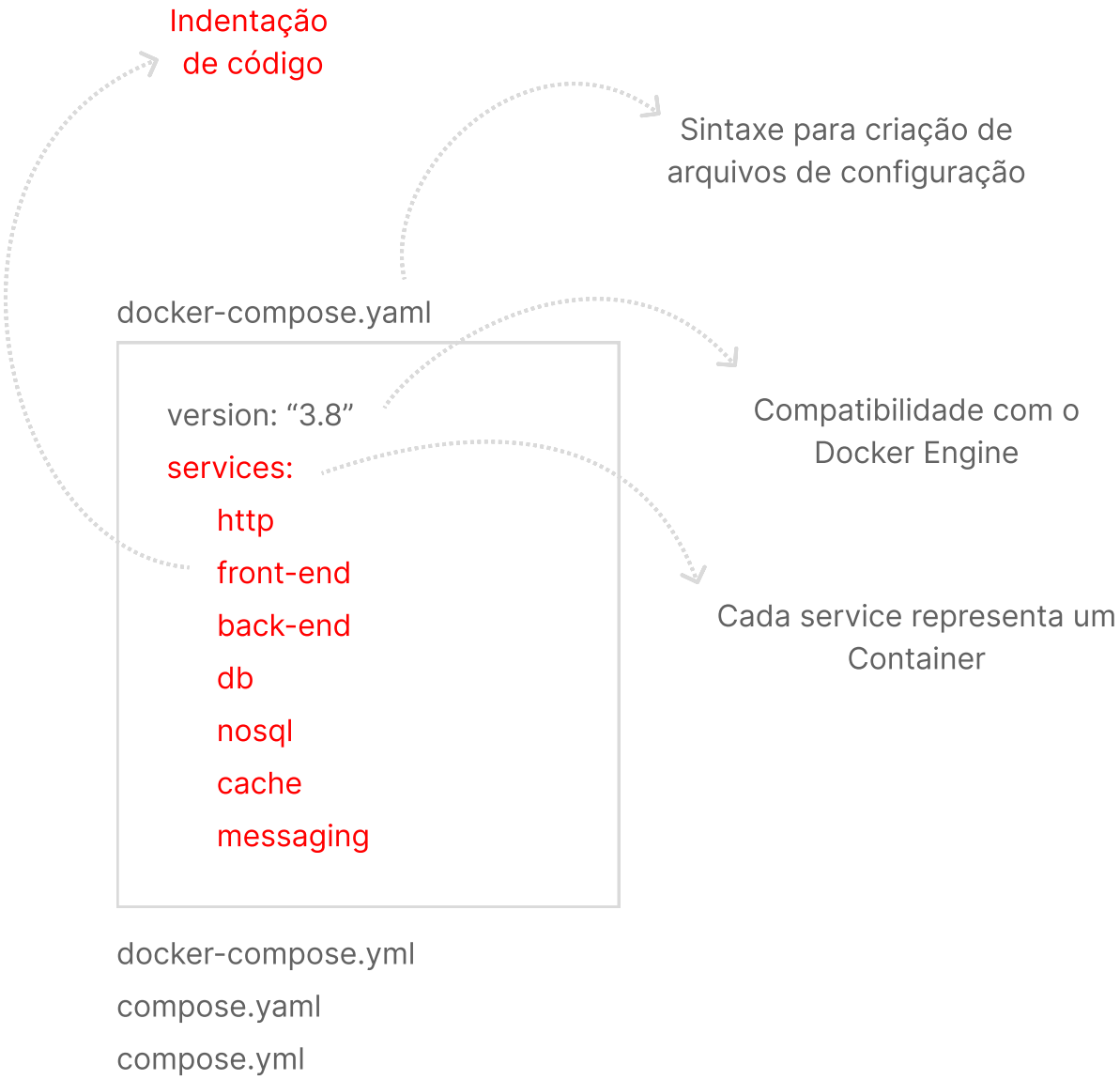
compose.yaml

compose.yml

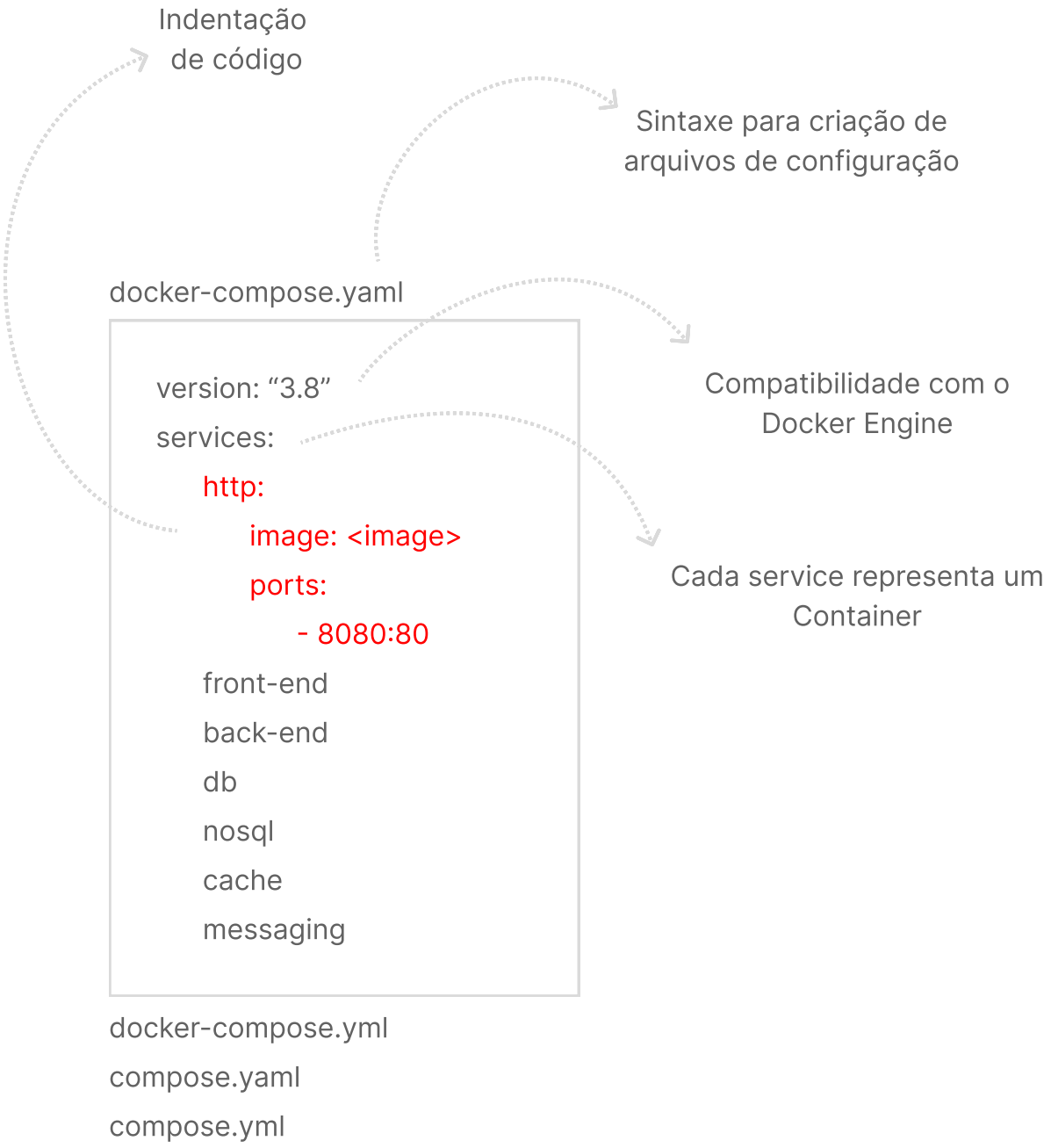
Definindo a versão da sintaxe utilizada no arquivo



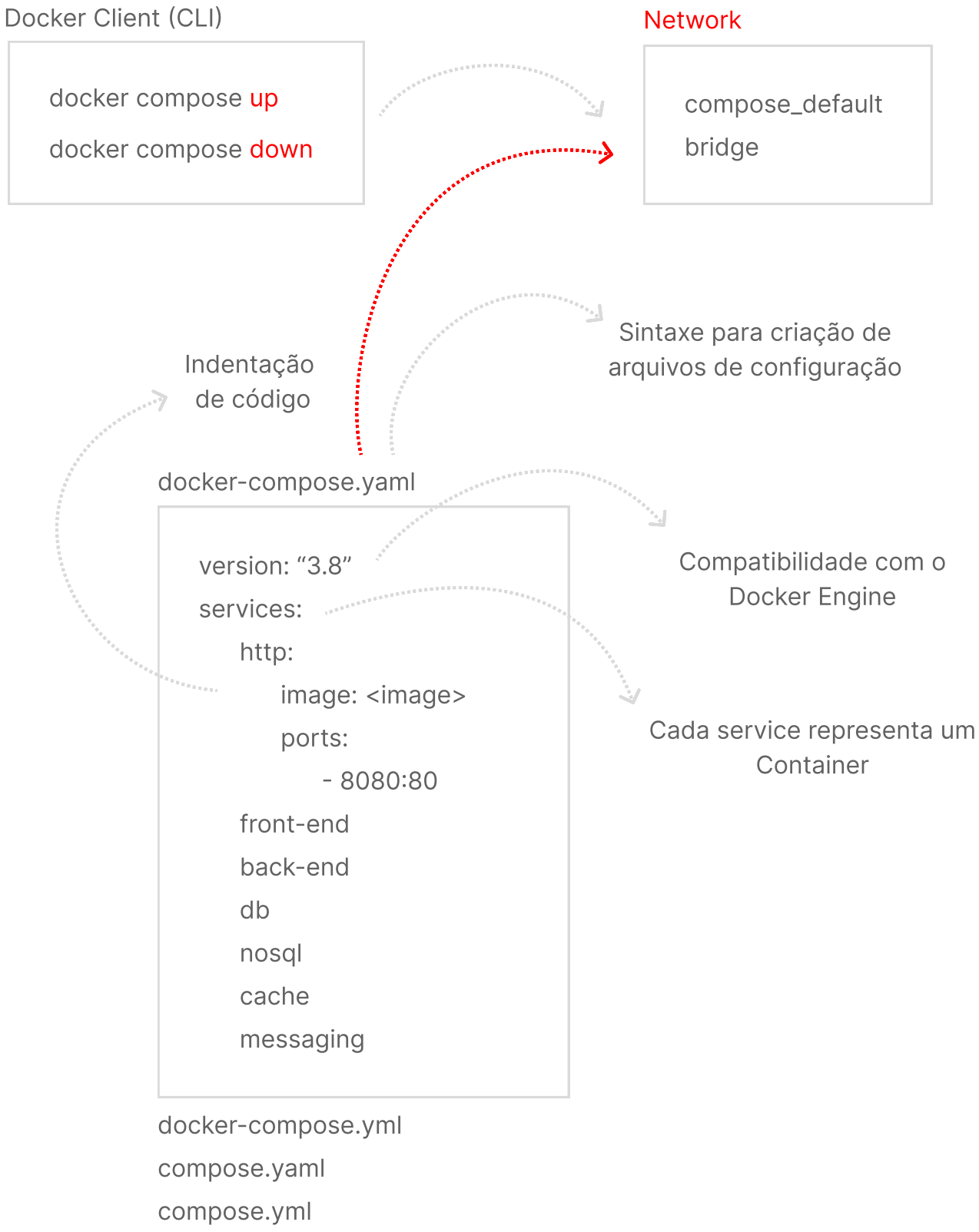
Definindo os serviços



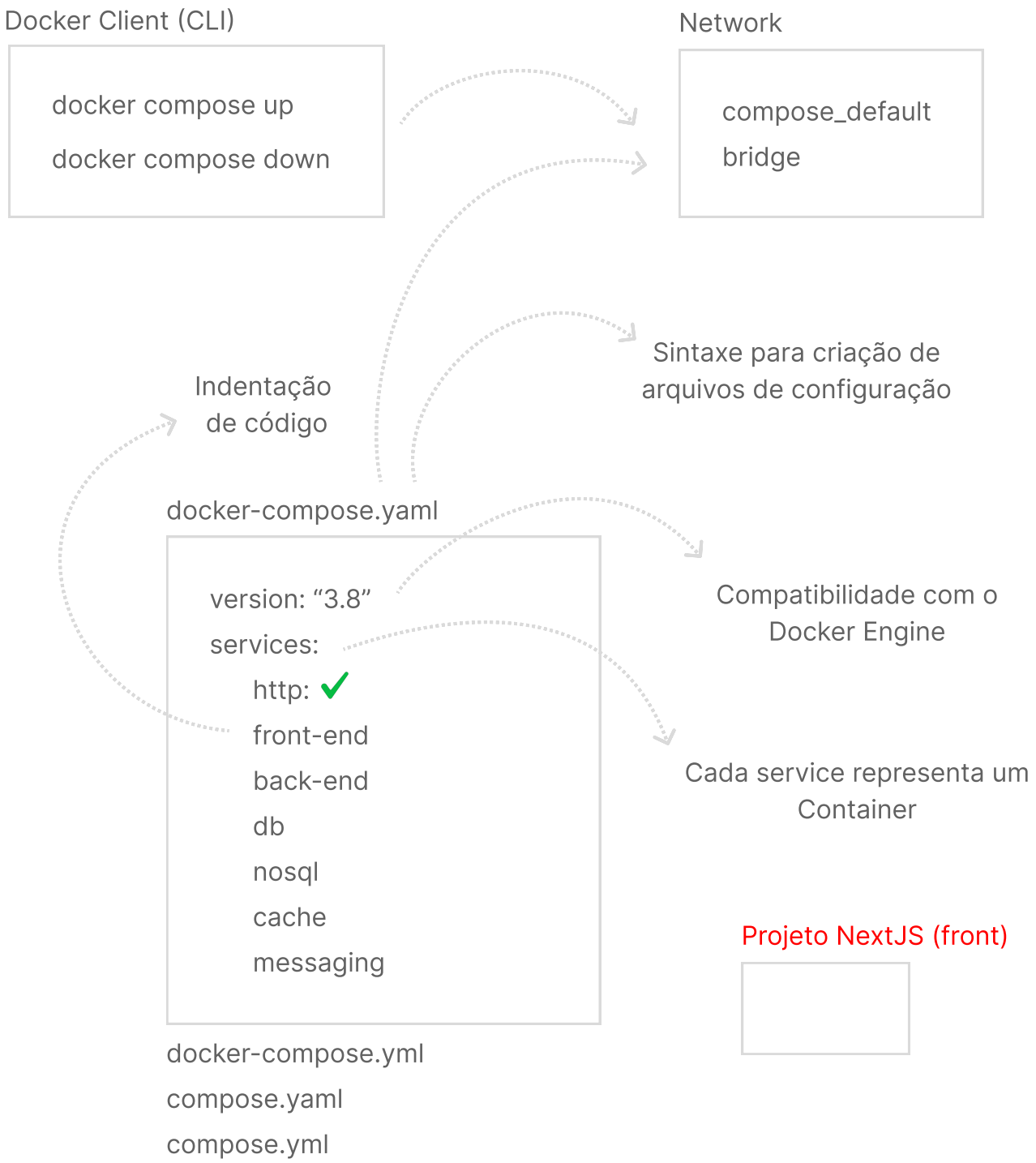
Serviço HTTP (Nginx) - Definindo as instruções image e ports



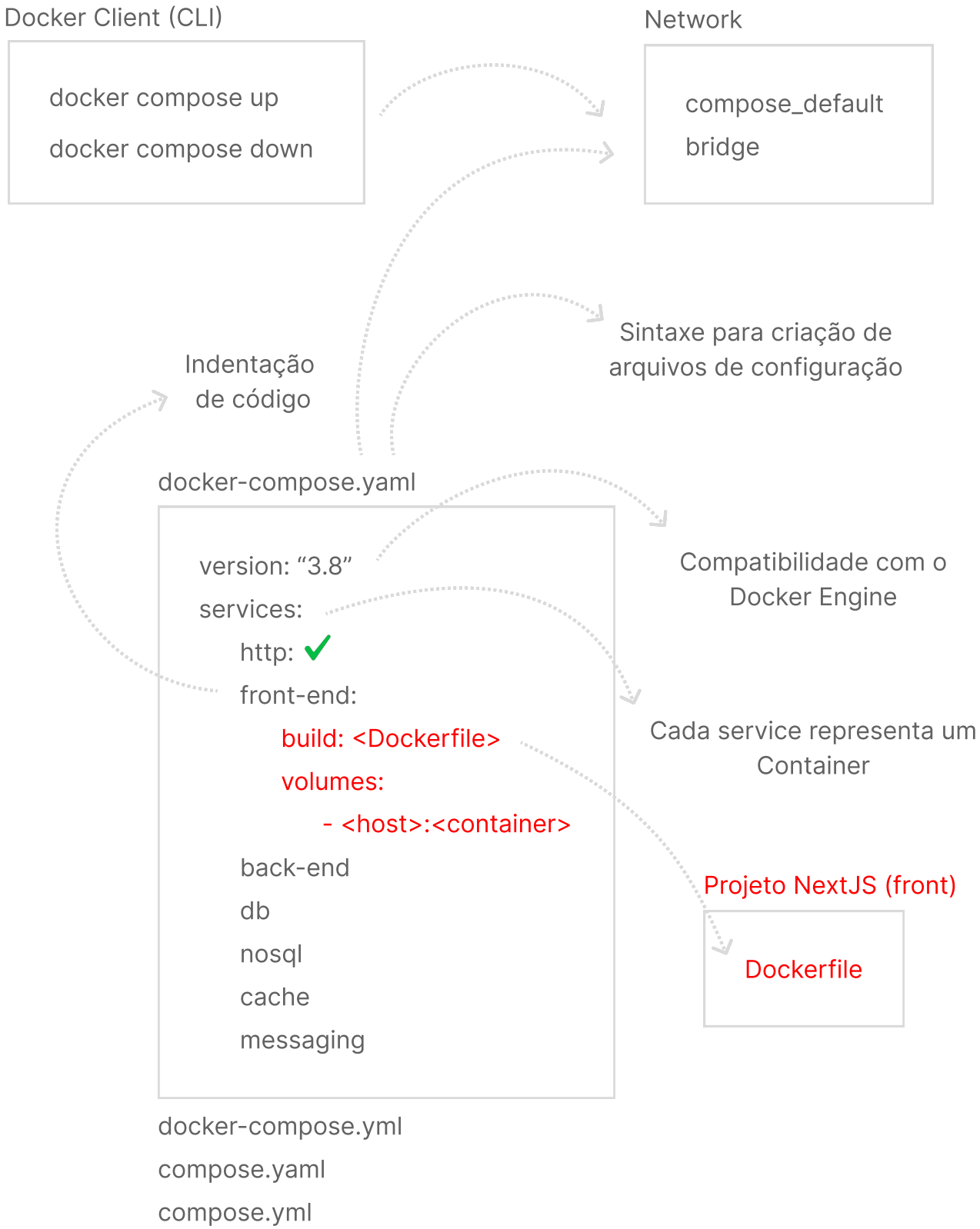
A rede padrão criada pelo Docker Compose



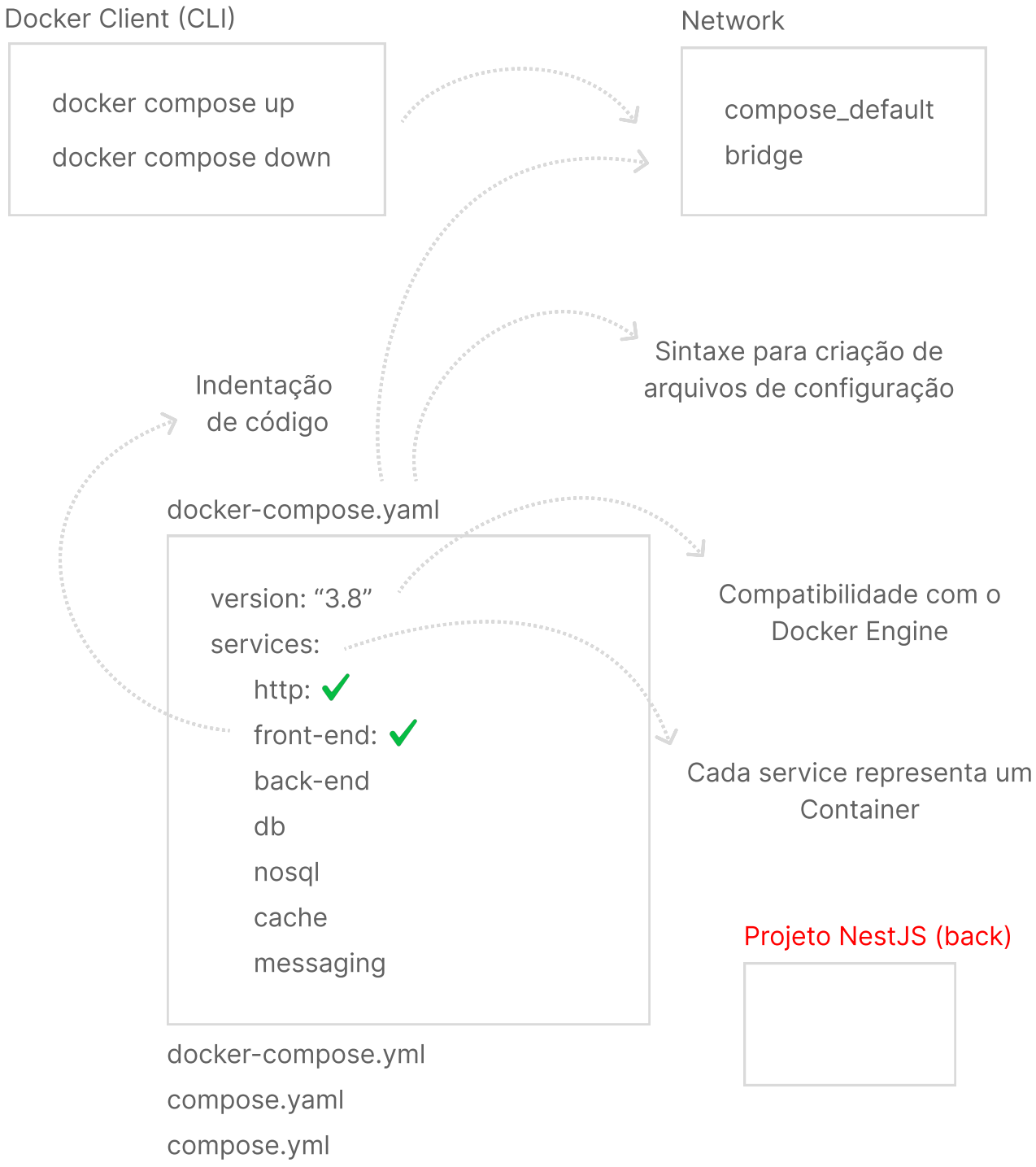
Serviço Front-end (NextJS) parte 1 - Organizando o projeto



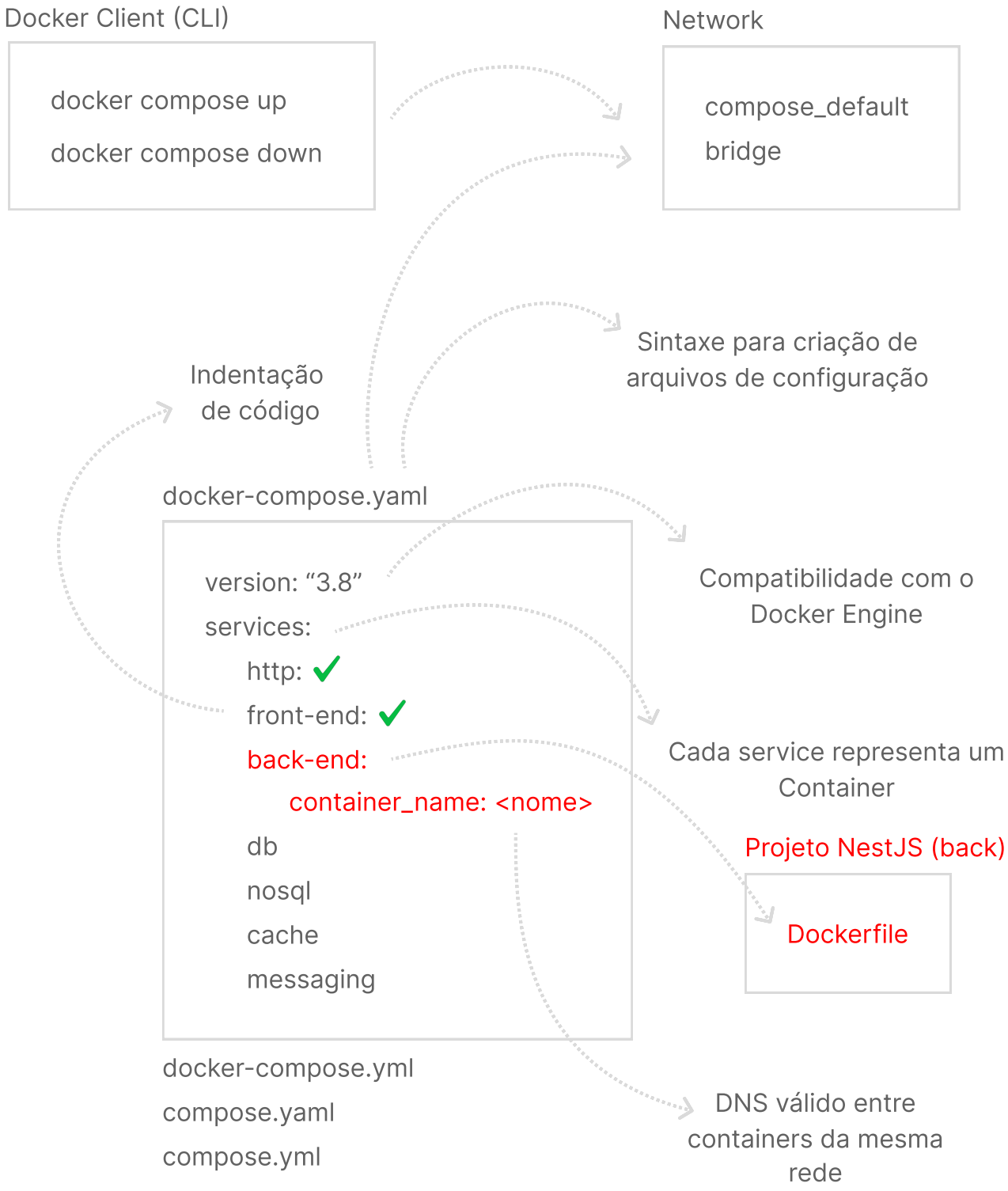
Serviço Front-end (NextJS) parte 2 - Definindo as instruções build e volumes



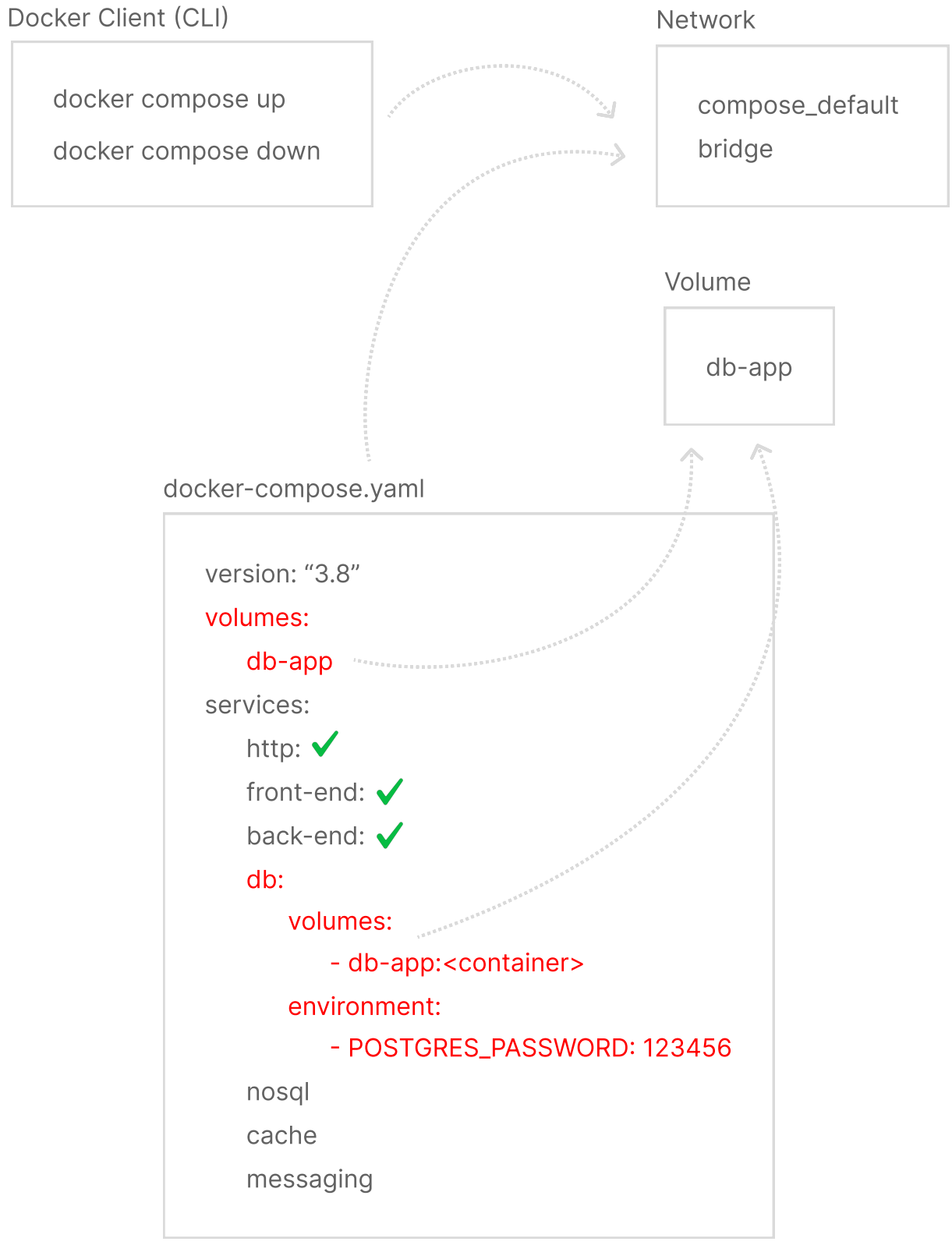
Serviço Back-end (NestJS) parte 1 - Organizando o projeto



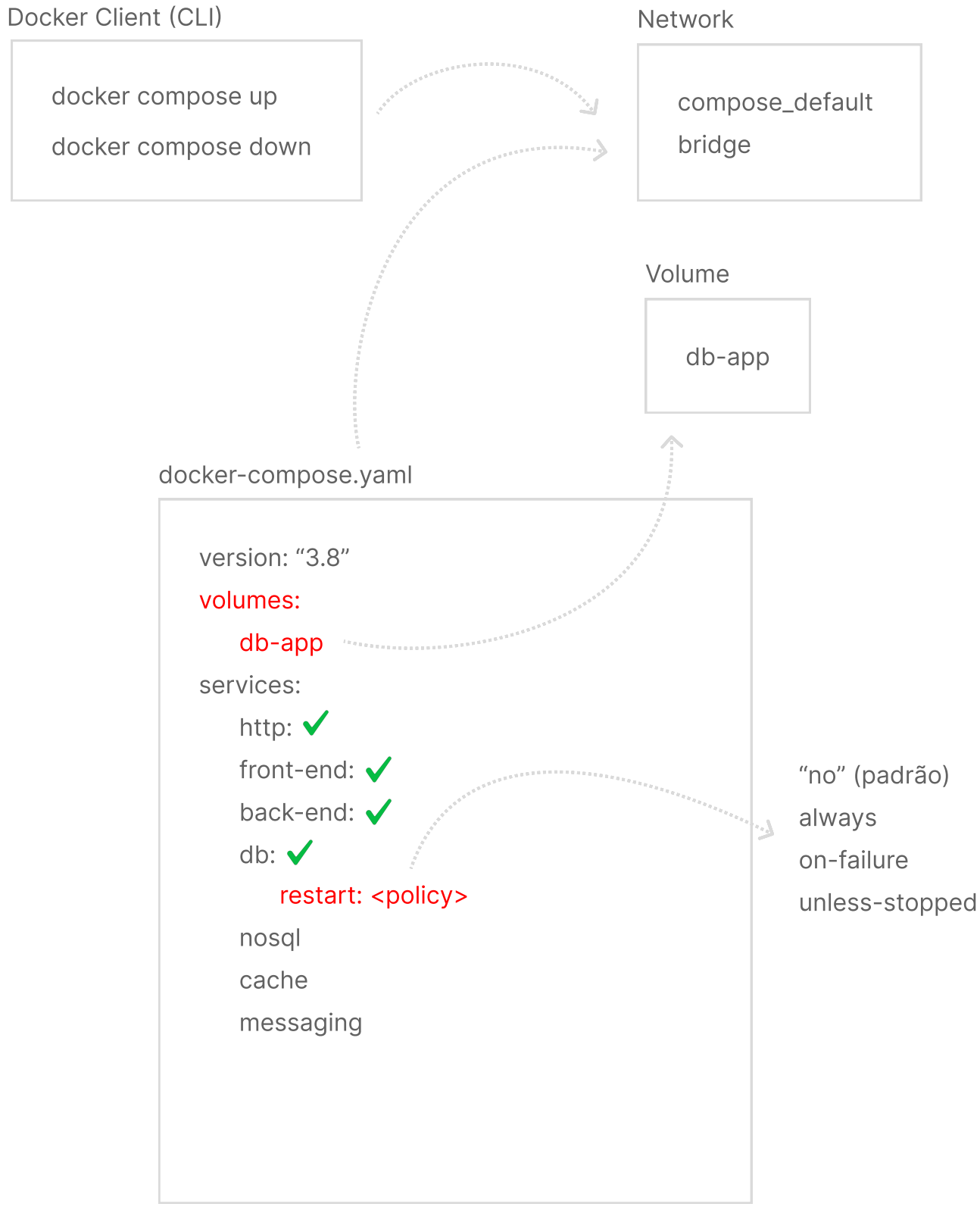
Serviço Back-end (NestJS) parte 2 - Definindo a instrução container_name



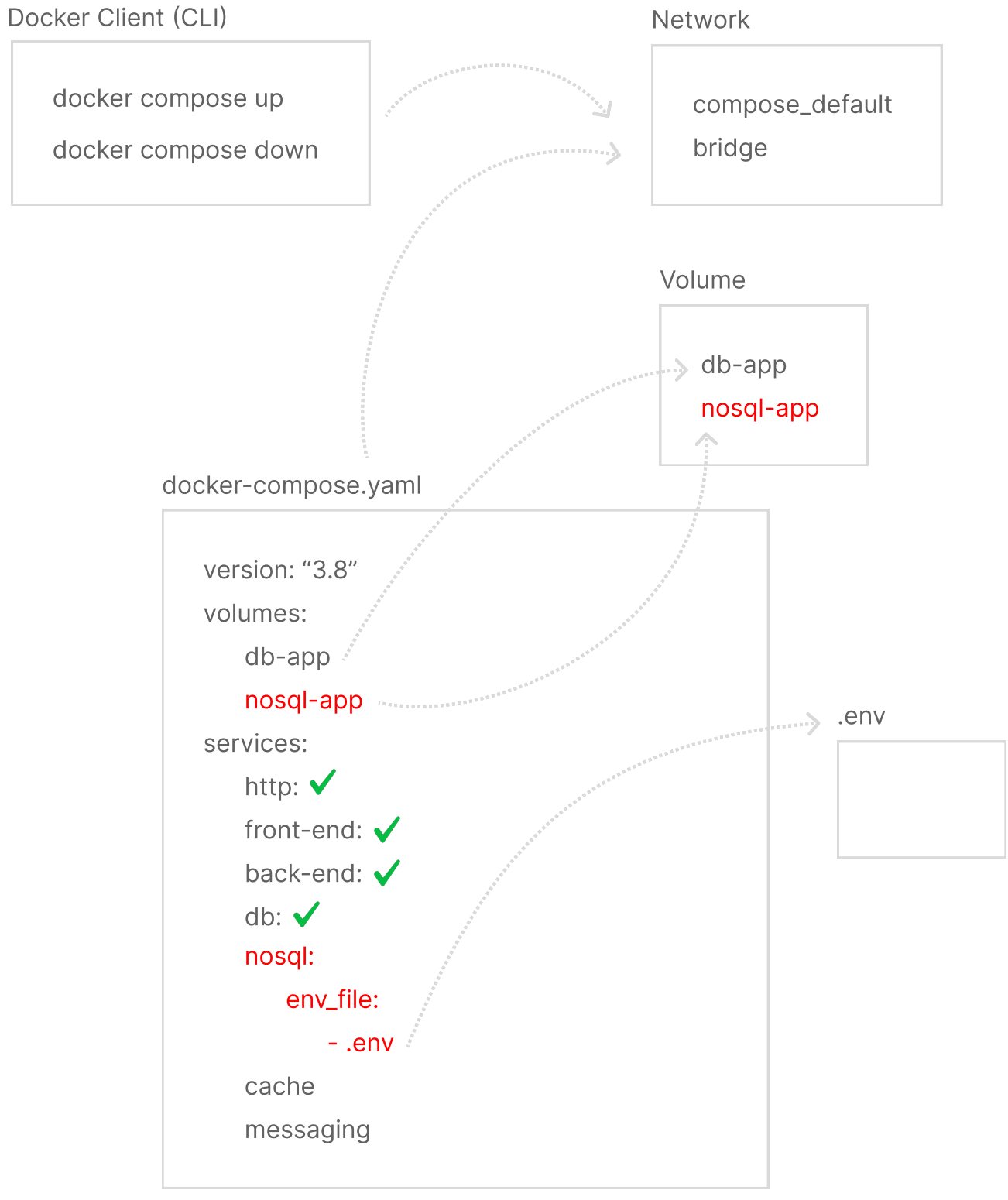
Serviço DB (PostgreSQL) - Definindo variáveis de ambiente e volumes



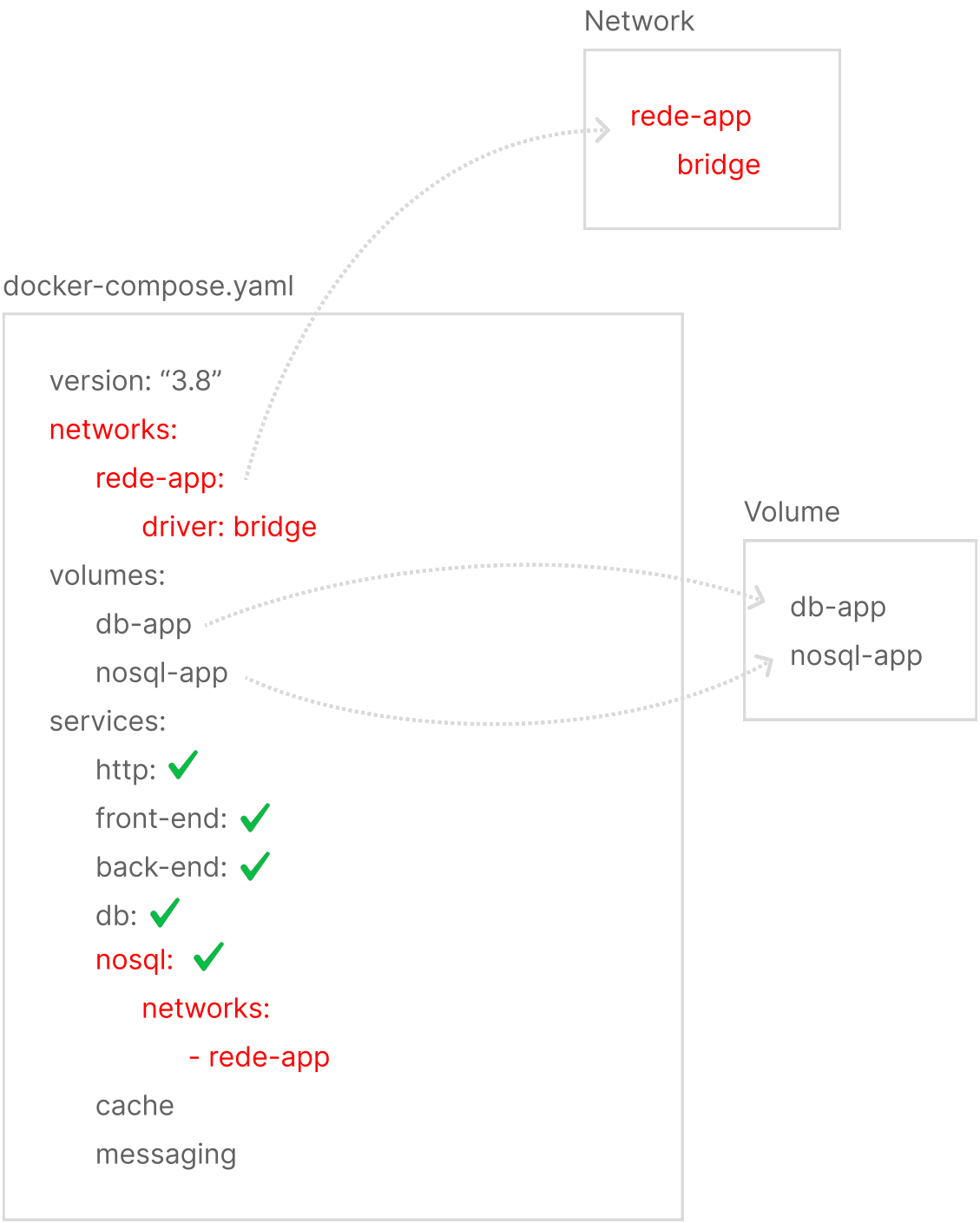
Reiniciando os serviços



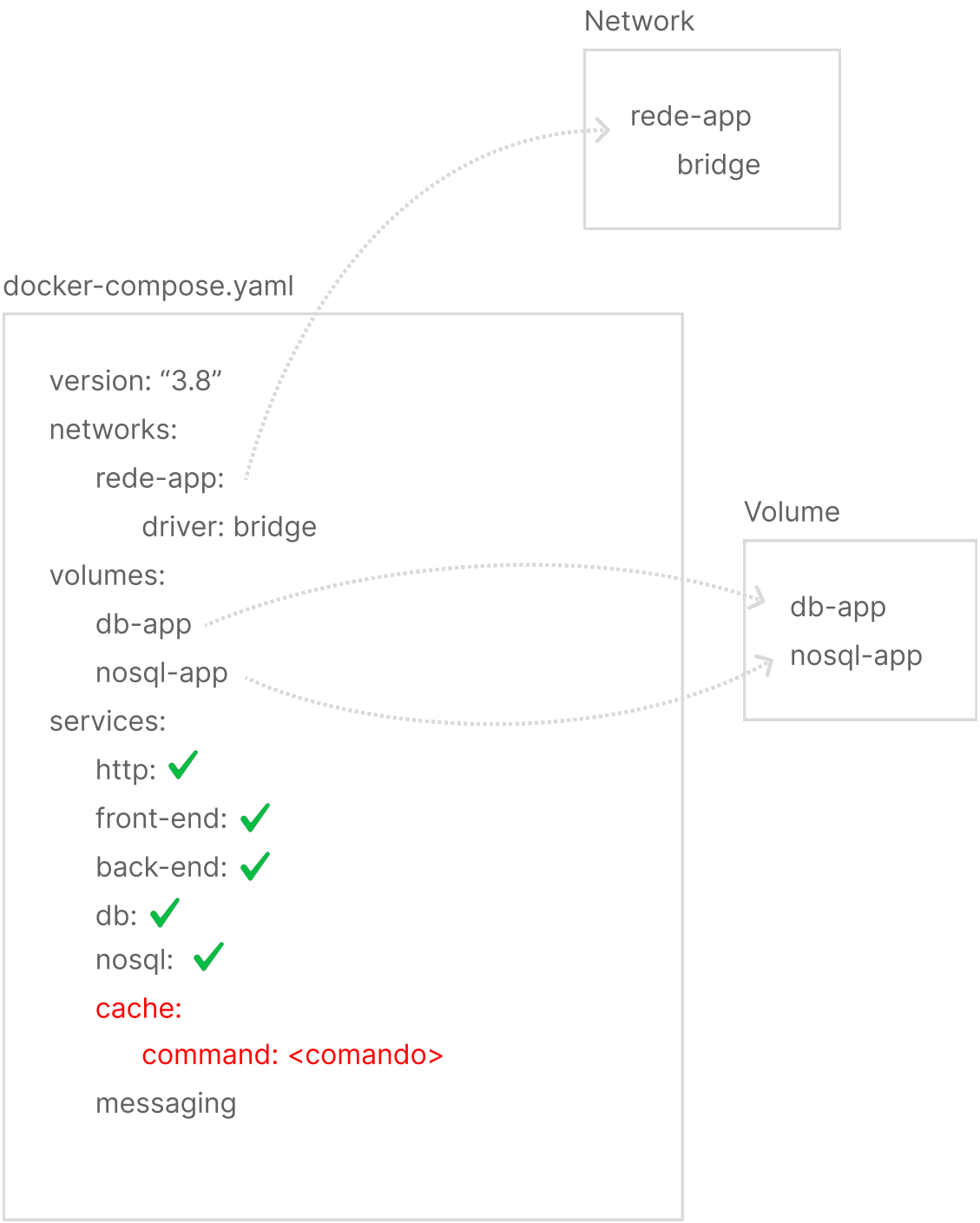
Serviço NoSQL (MongoDB) - Definindo variáveis de ambiente via arquivo externo



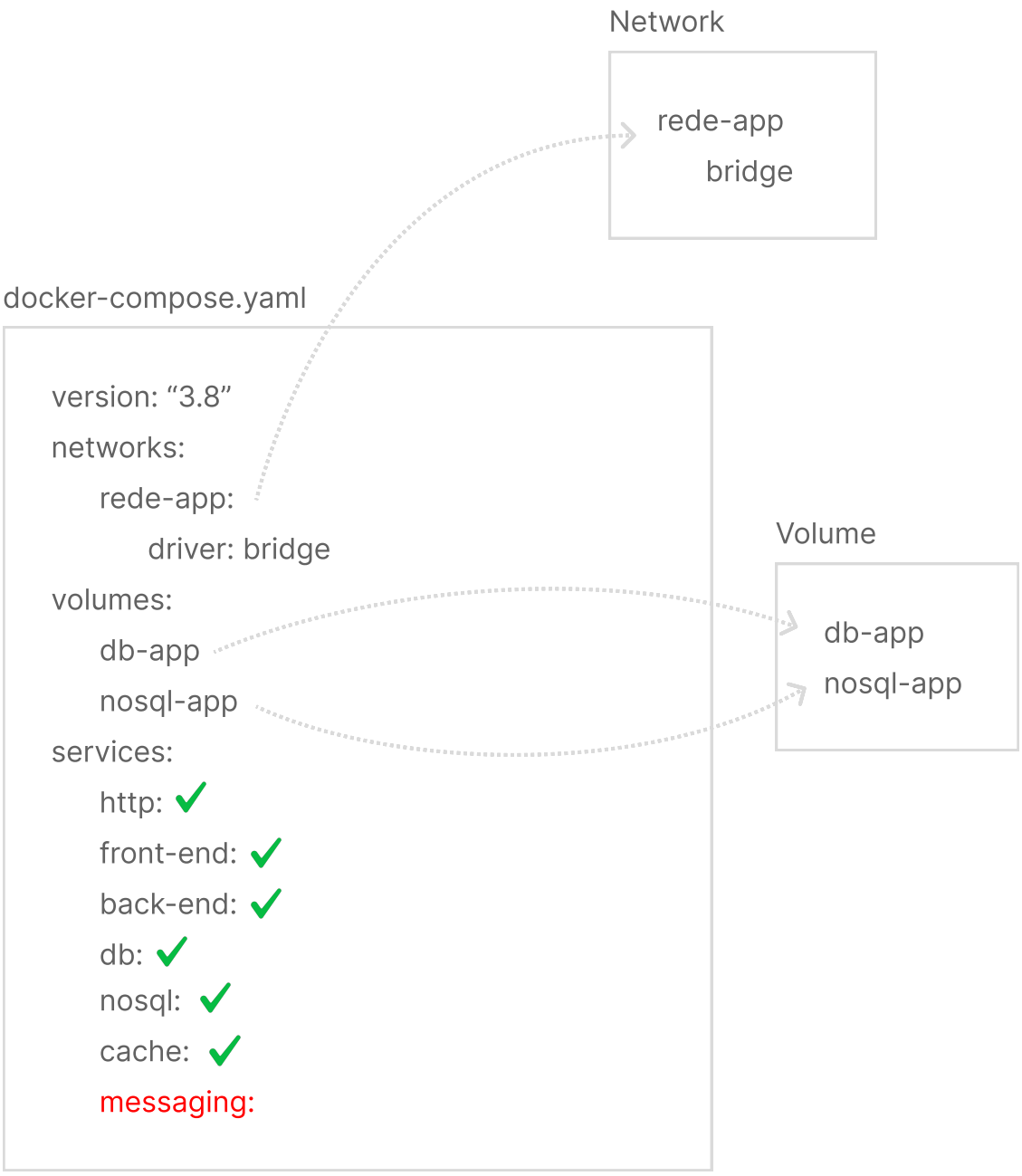
Criando uma nova rede



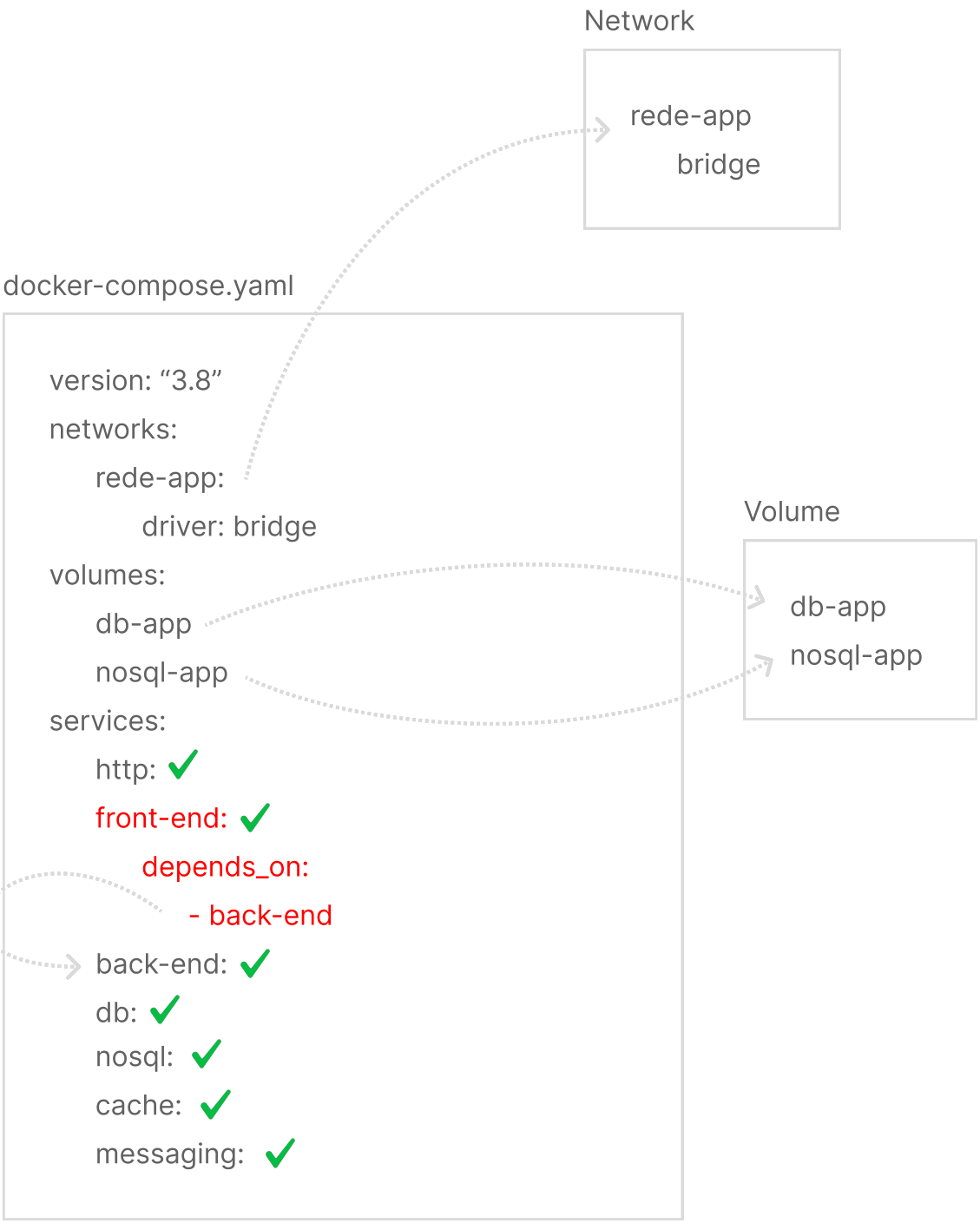
Serviço de Cache (Redis) - Definindo a instrução command



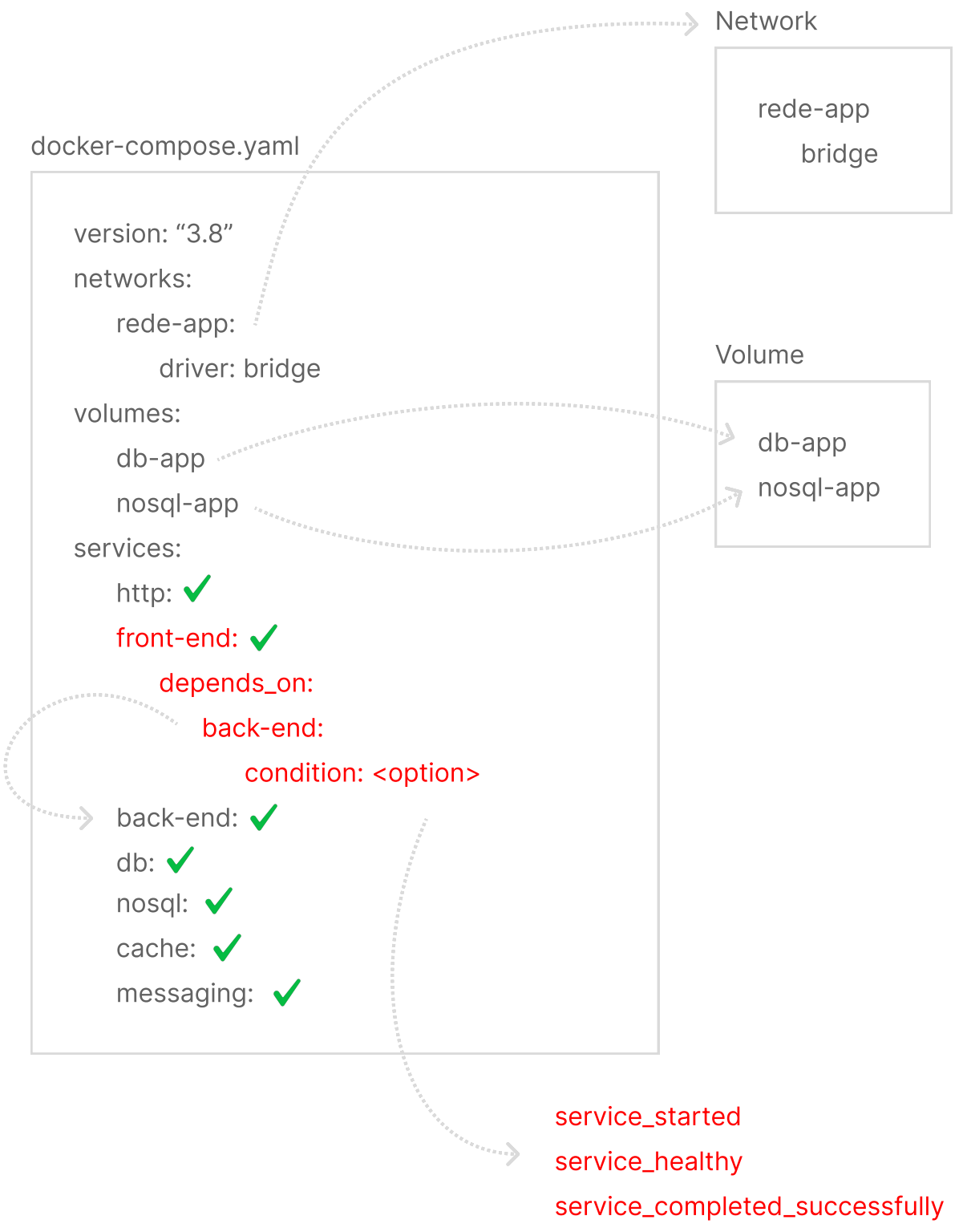
Serviço de Messaging (Kafka) - Consolidando os conhecimentos



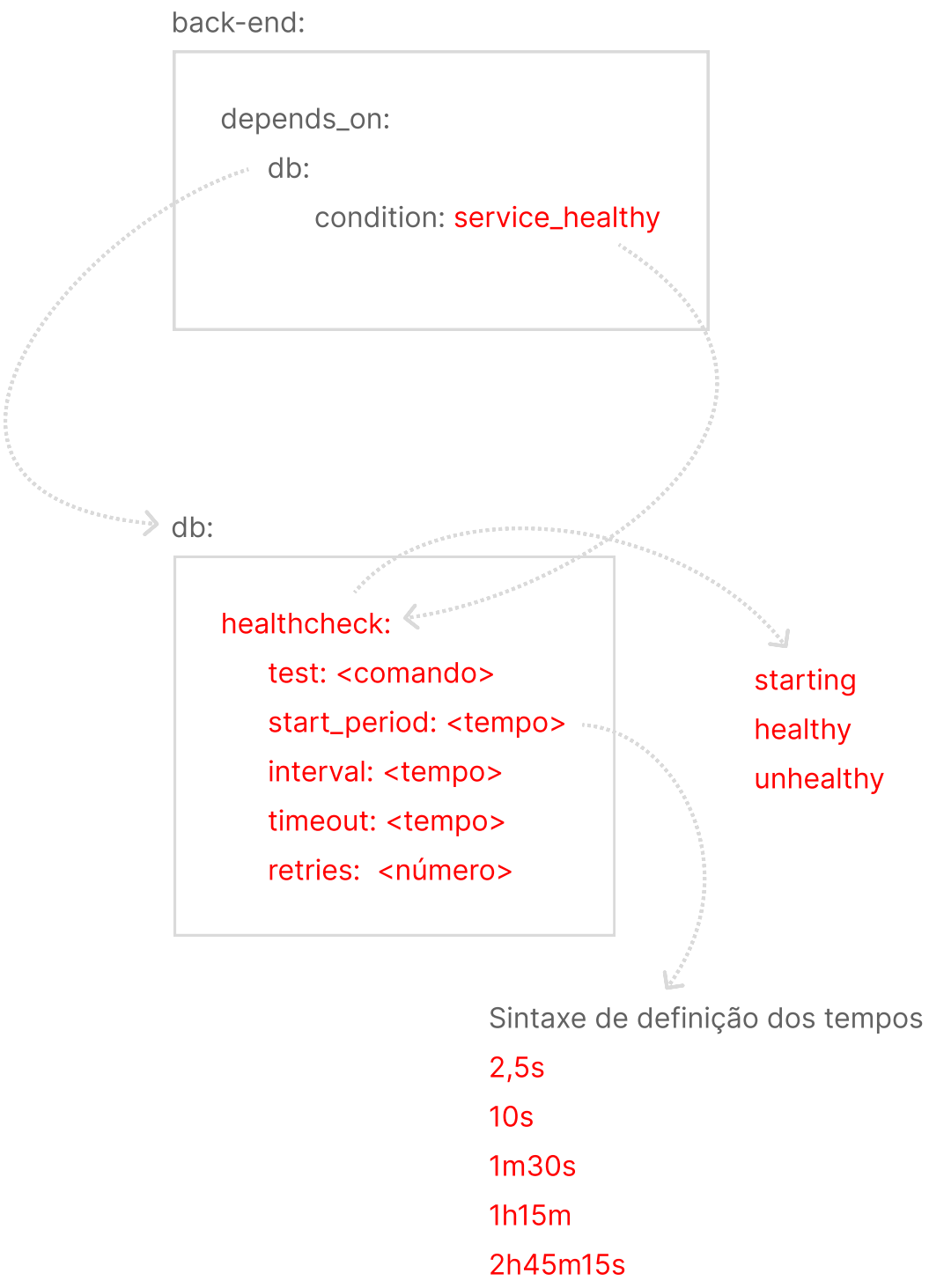
Depends On - Relação de dependências entre serviços



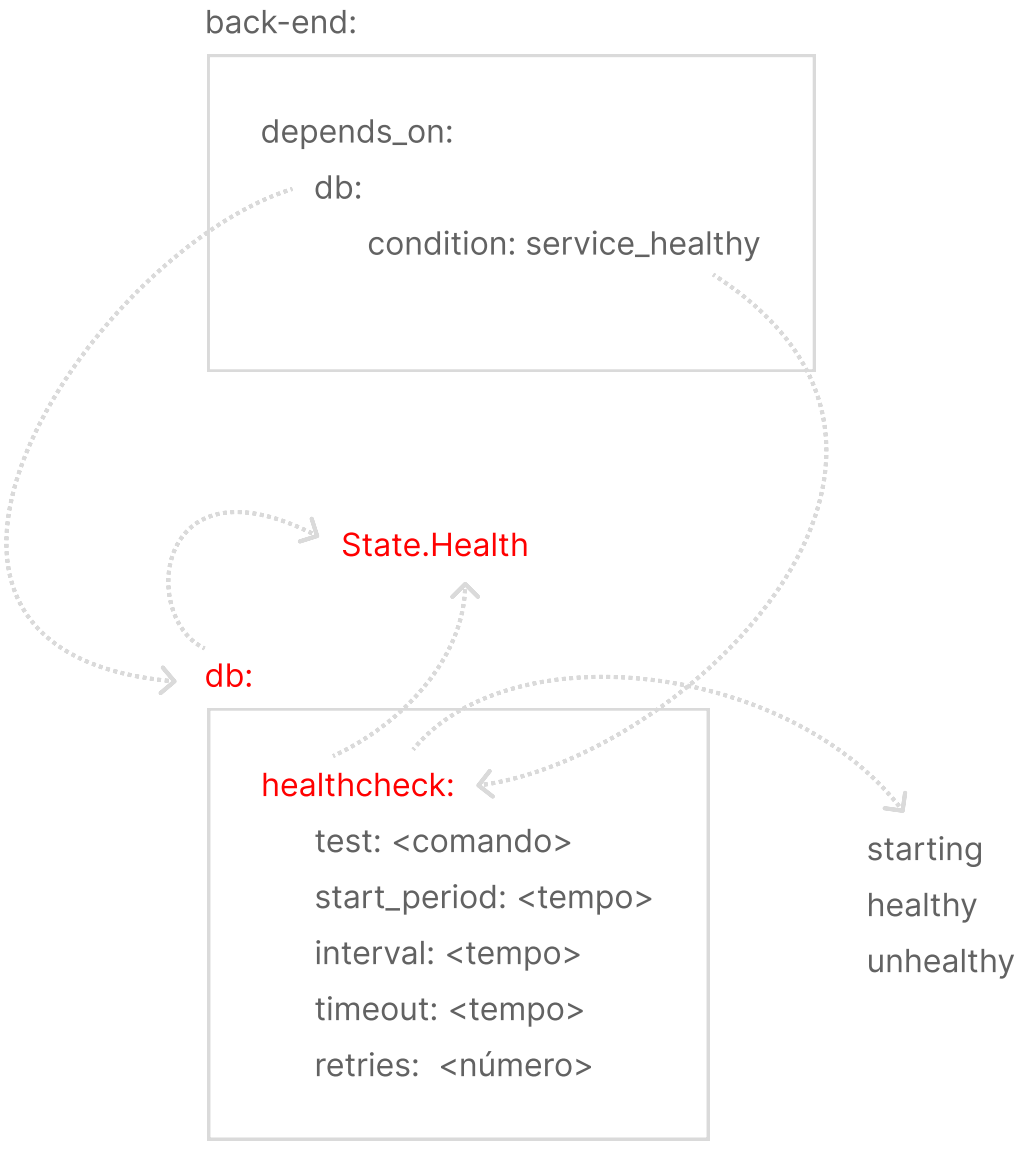
Depends On - Condições de dependência



Depends On - Avaliando a integridade do serviço



Healthcheck - Logs de estado da integridade do serviço



Especificando o arquivo docker compose a ser interpretado

docker compose up

docker-compose.yaml
docker-compose.yml
compose.yaml
compose.yml

docker compose -f <nome_arquivo> up

docker-compose.dev.yaml
docker-compose.hml.yaml
docker-compose.prd.yaml

ARGUS ACADEMY

www.argus-academy.com