

COMO INSTALAR E CONFIGURAR UM PASS **EM MINUTOS**



Douglas Nassif Roma Junior

 /douglasjunior

 /in/douglasjunior

 smarppy.com

 douglas@smarppy.com



Slides: <https://git.io/JGoeM>

ANTES DE INICIAR...

- Criar uma conta no **Docker Hub**:
 - <https://hub.docker.com/signup>
- Entrar no **Play With Docker**:
 - <https://labs.play-with-docker.com>



MÓDULO 1/3

- Conceitos do Docker
- Imagens e Containers
- Containers e Máquinas Virtuais
- Preparando o Ambiente Docker
- Recapitulando



MÓDULO 2/3

- Containers em detalhes
- Seu novo ambiente de desenvolvimento
- Definindo uma imagem com **Dockerfile**
- Construindo a imagem
- Iniciando o container



MÓDULO 3/3

- O que é uma “paas”?
- CapRover
- Pré-requisitos
- Instalação
- Explorando o ambiente
- Criando um app
- Bônus

MÓDULO 1

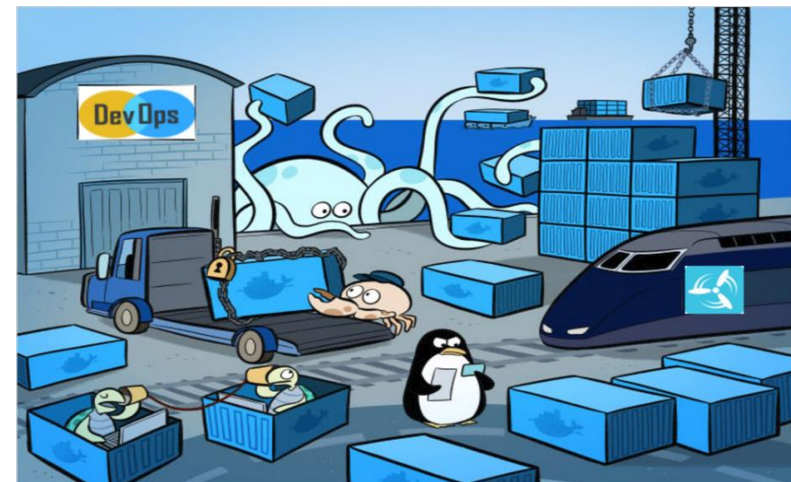
CONCEITOS DO DOCKER

CONCEITOS DO DOCKER

- Docker é uma plataforma para desenvolvedores e administradores de sistemas para auxiliar no desenvolvimento, implantação e execução de aplicações utilizando containers.
- O uso de containers Linux para implantar aplicações é chamado de “containerização”.
- Containers não são um conceito novo, mas a facilidade em implantar aplicações é.

CONCEITOS DO DOCKER

- A containerização está ficando cada vez mais popular, pois containers são:
 - **Flexíveis:** Até mesmo as aplicações mais complexas podem ser containerizadas.
 - **Leves:** Containers aproveitam e compartilham o mesmo kernel do host.
 - **Intercambiáveis:** Você pode atualizar um container *on-the-fly**.
 - **Portáveis:** Você pode construir localmente, implantar na nuvem e rodar em qualquer lugar.
 - **Escaláveis:** Você pode aumentar e distribuir automaticamente réplicas de seus containers.
 - **Empilháveis:** Você pode empilhar serviços verticalmente e *on-the-fly**.



* <https://www.maiovergara.com/on-the-fly-o-que-significa-esta-expressao/>

IMAGENS E CONTAINERS

IMAGENS E CONTAINERS

- Uma **imagem** é um pacote executável que inclui tudo o que é necessário para executar o código da aplicação:
 - *runtime* (PHP, Node, Java, etc)
 - bibliotecas
 - variáveis de ambiente
 - arquivos de configuração
- Um **container** é iniciado quando você executa uma imagem, ou seja, **container** é uma instância de uma imagem – é o que a imagem se torna quando é executada.
- Você pode ver a lista de containers que estão sendo executados com o comando:

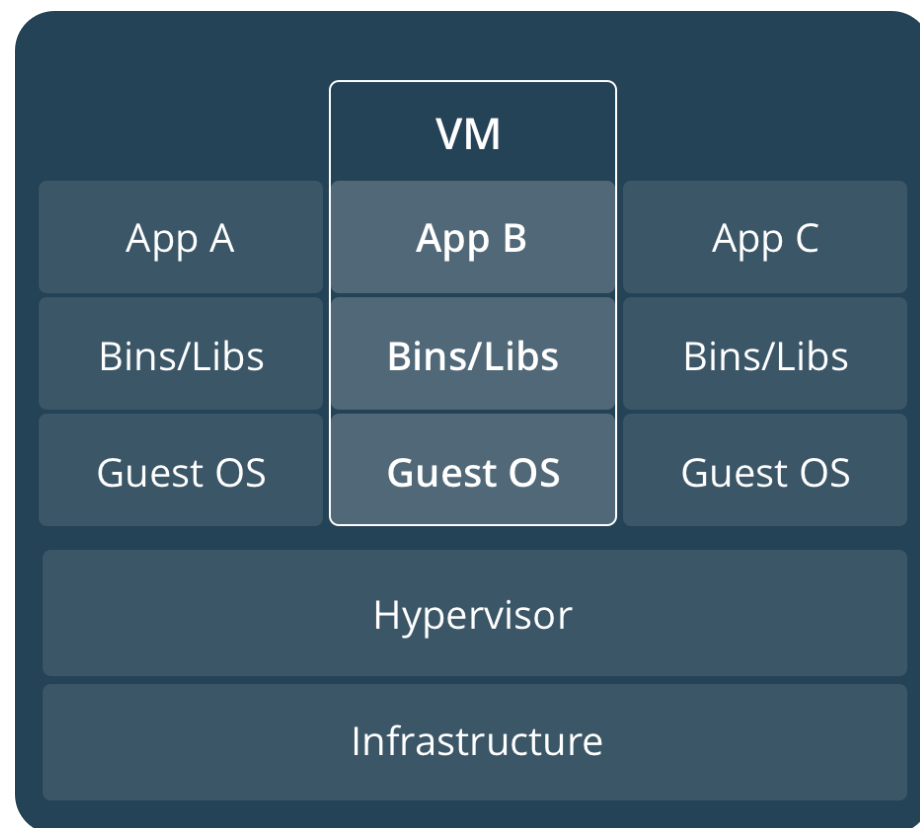
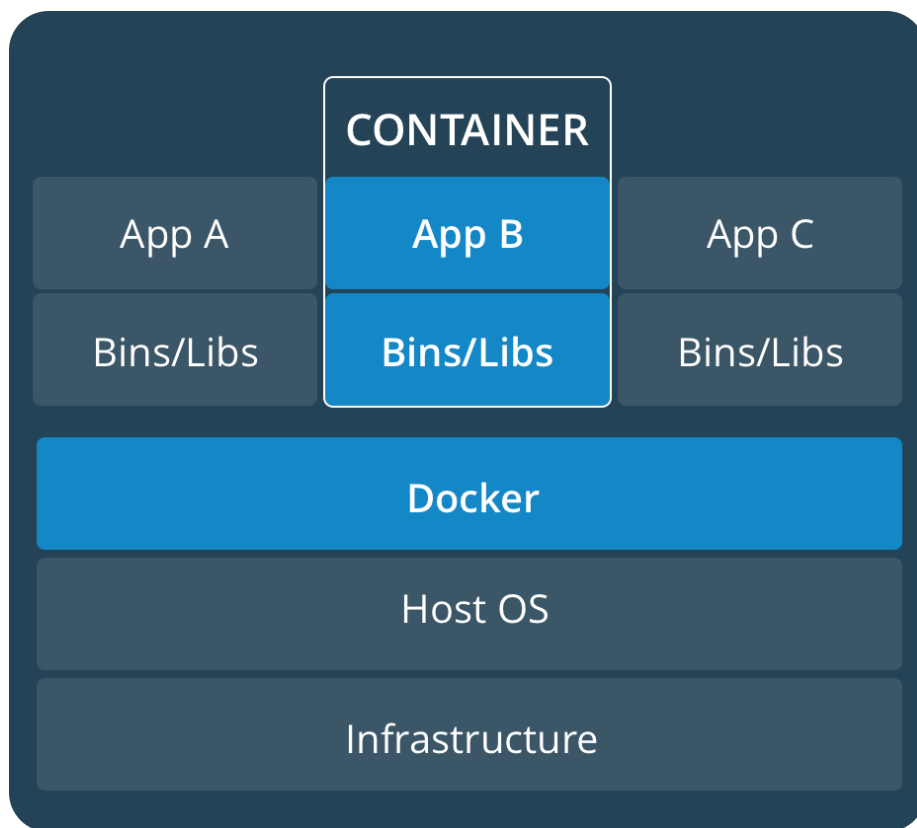
```
$ docker ps
```

CONTAINERS E MÁQUINAS VIRTUAIS

CONTAINERS E MÁQUINAS VIRTUAIS

- Um **container** roda “nativamente” em Linux e compartilha o *kernel* da máquina host com outros containers.
- Ele roda como um processo discreto, sem ocupar mais memória do que qualquer outro executável, fazendo com que seja leve.
- Em contraste, uma **máquina virtual (VM)** executa um novo sistema operacional, com acessos virtuais para a máquina host através de um *hypervisor*.
- Em geral, VMs fornecem um ambiente com mais recursos do que uma aplicação realmente precisa.

CONTAINERS E MÁQUINAS VIRTUAIS



PREPARANDO O AMBIENTE DOCKER

PREPARANDO O AMBIENTE DOCKER

- Instale a versão atual do **Docker Community Edition (CE)** ou **Enterprise Edition (EE)** através do site oficial, para a plataforma suportada.
- Mac OSX: <https://docs.docker.com/docker-for-mac/install/>
- Windows 10: <https://docs.docker.com/docker-for-windows/install/>
- Ubuntu: <https://docs.docker.com/engine/install/ubuntu/>
- CentOS: <https://docs.docker.com/engine/install/centos/>
- Outros: <https://docs.docker.com/desktop/>

PREPARANDO O AMBIENTE DOCKER

- Verificando a versão do docker:

```
$ docker --version
```

- Veja mais detalhes sobre a sua instalação do docker:

```
$ docker info
```

- No Linux, para evitar erros com permissões (e a necessidade do `sudo`), adicione o seu usuário no grupo docker: <https://docs.docker.com/install/linux/linux-postinstall/>

PREPARANDO O AMBIENTE DOCKER

- **Teste se a sua instalação funciona, rodando uma simples imagem `hello-world`.**

```
$ docker run hello-world
```

- **Liste as imagens que foram baixadas em sua máquina:**

```
$ docker image ls (ou docker images)
```

- **Liste os containers que foram criados em sua máquina. (para exibir os containers que estão parados utilize o `--all`)**

```
$ docker container ls --all (ou docker ps -a)
```

RECAPTULANDO

RECAPTULANDO

- **Containerização** torna o desenvolvimento e implantação de aplicações mais simples.
 - Sua aplicação não depende do sistema operacional da máquina
 - Atualizações podem ser enviadas para qualquer parte de uma aplicação distribuída
 - A densidade dos recursos pode ser otimizada
- Com **Docker**, escalar sua aplicação é uma questão de rodar novos executáveis, e não rodar pesadas máquinas virtuais inteiras.

MÓDULO 2

CONTAINERS EM DETALHES

CONTAINERS

- Para construir uma aplicação com Docker, é preciso começar da raiz de toda a hierarquia da aplicação: **o container**.
- Acima do container, temos o serviço (*service*), que define como os containers irão se comportar em produção.
- Finalmente, no topo da lista, temos a pilha (*stack*), que define as interações entre todos os serviços.



Stack
Service
Container (estamos aqui)

SEU NOVO AMBIENTE DE DESENVOLVIMENTO

SEU NOVO AMBIENTE DE DESENVOLVIMENTO

- Normalmente, quando você inicia um novo projeto, primeiro você precisa preparar todas as ferramentas de *runtime* (PHP, Java, Python, Node) em sua máquina.
- Porém, o ambiente de sua máquina precisa ser perfeito para sua aplicação rodar como o esperado. Além de ser idêntico ao ambiente de **produção**.



SEU NOVO AMBIENTE DE DESENVOLVIMENTO

- Com **Docker**, você precisa apenas obter uma imagem contendo o *runtime* desejado, nenhuma instalação é necessária.
- Então, sua construção pode conter uma imagem do *runtime* desejado com todas as bibliotecas e dependências necessárias, juntas e organizadas.



DEFININDO UMA IMAGEM COM DOCKERFILE

IMAGEM COM DOCKERFILE

- O **Dockerfile** define como a imagem deve ser criada, e o passos necessários para ele se tornar um container.
- Acesso a recursos como interface de rede e drivers de disco, são virtualizados dentro do ambiente, de forma isolada do restante do sistema.
- Sendo assim, é preciso mapear portas e diretórios para fora do container, além de especificar quais arquivos devem ser copiados para dentro do container.
- Deste modo, temos a garantia que nossa aplicação terá o mesmo comportamento, onde quer que seja executada.

IMAGEM COM DOCKERFILE

```
1  # Seleciona uma imagem base contendo o runtime desejado
2  FROM node:12-alpine
3
4  # Instala dependências necessárias para rodar a aplicação
5  RUN apk add --no-cache python g++ make
6
7  # Define o diretório padrão de trabalho
8  WORKDIR /app
9
10 # Copia o conteúdo do diretório local (host) para dentro da imagem
11 COPY . .
12
13 # Instala as dependências do projeto Node JS
14 RUN yarn install --production
15
16 # Executa o arquivo index.js com o Node JS
17 CMD ["node", "src/index.js"]
```

CONSTRUINDO A IMAGEM

CONSTRUINDO A IMAGEM

- Faça o clone do projeto, contendo os arquivos da aplicação e o **Dockerfile**

```
$ git clone https://github.com/douglasjunior/node-app-with-docker.git
```

```
$ cd node-app-with-docker
```

- Crie a imagem: perceba que o parâmetro `-t` é usado para dar um nome amigável para a imagem.

```
$ docker build -t meuapp .
```

- Para listar a imagem criada:

```
$ docker image ls (ou docker images)
```

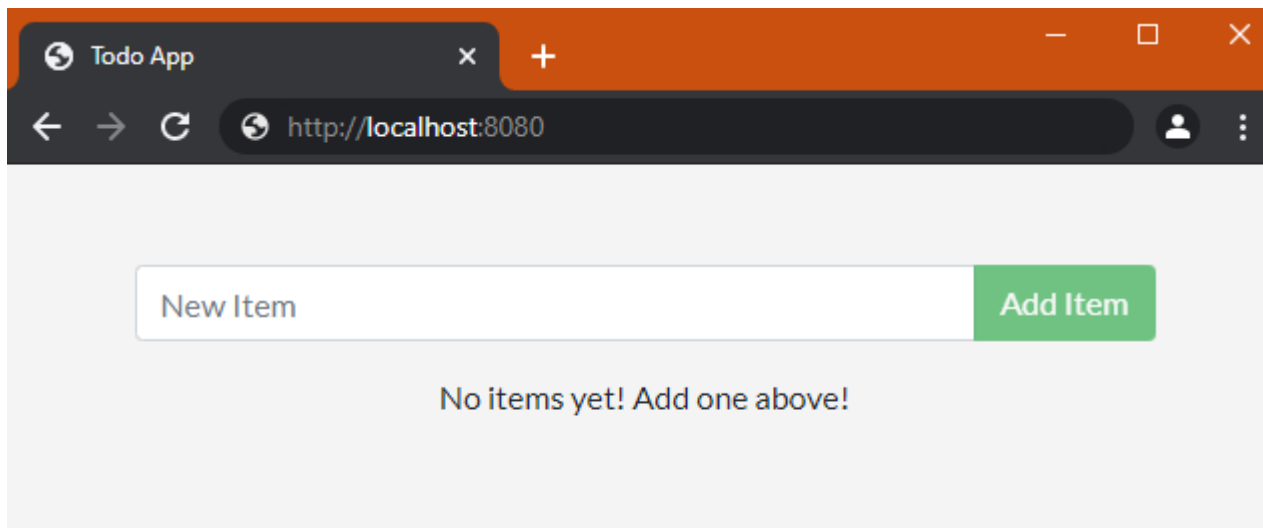
INICIANDO O CONTAINER

INICIANDO O CONTAINER

- Execute sua aplicação, mapeando a **porta 3000 do container** para a porta **8080 da sua máquina** (apenas para evitar possível conflitos).

```
$ docker run -p 8080:3000 meuapp
```

- Ao acessar o endereço em seu navegador, você deve visualizar algo assim:



INICIANDO O CONTAINER

- Você também pode rodar a aplicação em segundo plano, sem que o terminal fique preso:

```
$ docker run -d -p 8080:3000 meuapp
```

- O comando anterior lhe retornará o ID do container em execução. Agora seu container está executando em *background*.

- Você pode consultar os containers em execução com o comando:

```
$ docker container ls (ou docker ps)
```

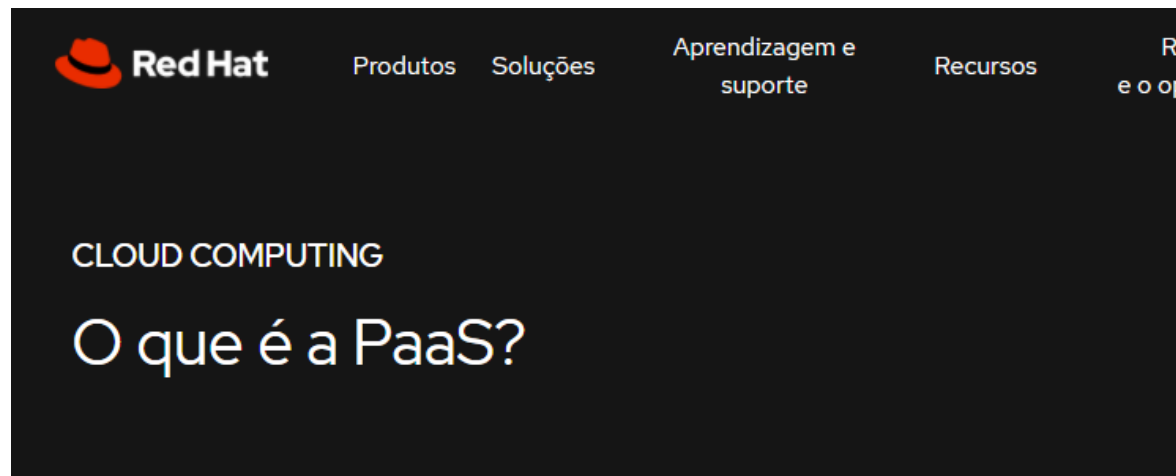
- Para interromper o container, execute o comando:

```
$ docker container stop <container-id>
```


MÓDULO 3

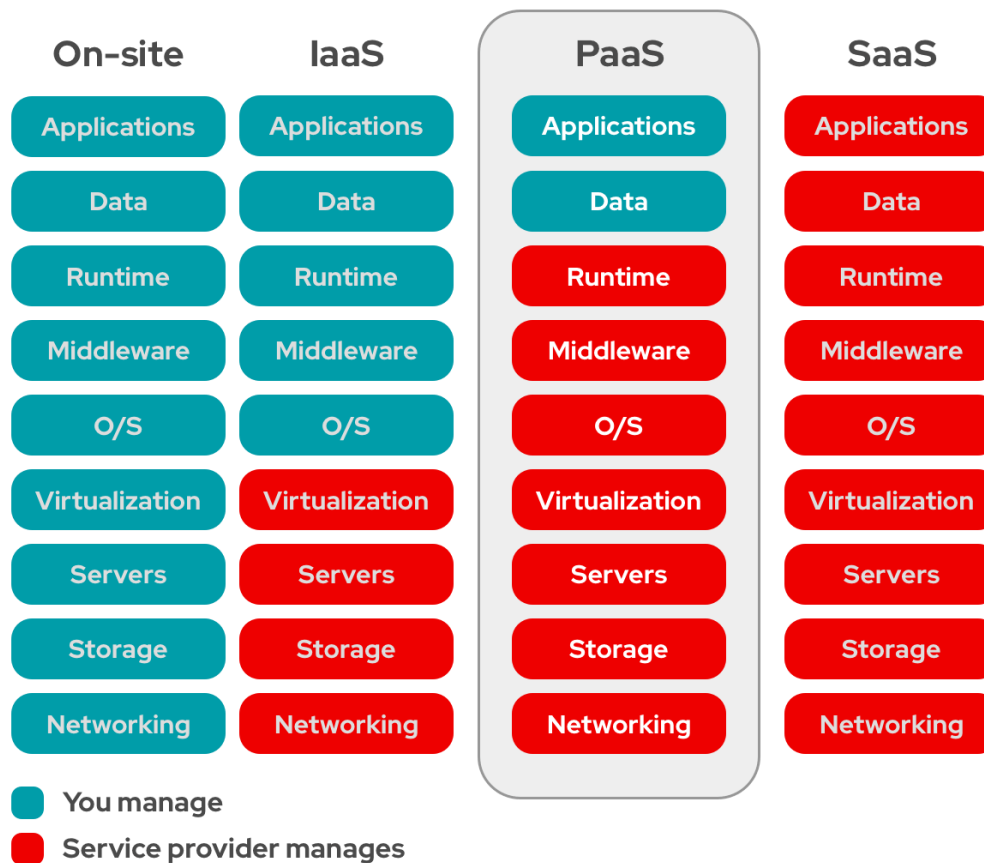
O QUE É UMA “PAAS”?

O QUE É UMA “PAAS”?



Plataforma como serviço (PaaS) é uma forma de cloud computing em que as plataformas de hardware e software de aplicações são fornecidas por terceiros. Direcionada principalmente para desenvolvedores e programadores, a solução de PaaS permite ao usuário desenvolver, executar e gerenciar aplicações sem ter o trabalho de criar e manter a infraestrutura ou plataforma que normalmente está associada a esses processos.

O QUE É UMA “PAAS”?



CAPROVER

CAPROVER

What's this?

CapRover is an extremely easy to use app/database deployment & web server manager for your **NodeJS, Python, PHP, ASP.NET, Ruby, MySQL, MongoDB, Postgres, WordPress** (and etc...) applications!

It's blazingly fast and very robust as it uses Docker, nginx, LetsEncrypt and NetData under the hood behind its simple-to-use interface.

- ✓ CLI for automation and scripting
- ✓ Web GUI for ease of access and convenience
- ✓ No lock-in! Remove CapRover and your apps keep working!
- ✓ Docker Swarm under the hood for containerization and clustering
- ✓ Nginx (fully customizable template) under the hood for load-balancing
- ✓ Let's Encrypt under the hood for free SSL (HTTPS)

CAPROVER

What's this?

CapRover is an extremely easy to use app/database deployment & web server manager for your **NodeJS, Python, PHP, ASP.NET, Ruby, MySQL, MongoDB, Postgres, WordPress** (and etc...) applications!

It's blazingly fast and very robust as it uses Docker, nginx, LetsEncrypt and NetData under the hood behind its simple-to-use interface.


- ✓ CLI for automation and scripting
- ✓ Web GUI for ease of access and convenience
- ✓ No lock-in! Remove CapRover and your apps keep working!
- ✓ Docker Swarm under the hood for containerization and clustering
- ✓ Nginx (fully customizable template) under the hood for load-balancing
- ✓ Let's Encrypt under the hood for free SSL (HTTPS)

CAPROVER

What's this?

CapRover is an extremely easy to use app/database deployment & web server manager for your **NodeJS, Python, PHP, ASP.NET, Ruby, MySQL, MongoDB, Postgres, WordPress** (and etc...) applications!

It's blazingly fast and very robust as it uses Docker, nginx, LetsEncrypt and NetData under the hood behind its simple-to-use interface.

- 
- ✓ CLI for automation and scripting
 - ✓ Web GUI for ease of access and convenience
 - ✓ No lock-in! Remove CapRover and your apps keep working!
 - ✓ Docker Swarm under the hood for containerization and clustering
 - ✓ Nginx (fully customizable template) under the hood for load-balancing
 - ✓ Let's Encrypt under the hood for free SSL (HTTPS)

CAPROVER

What's this?

CapRover is an extremely easy to use app/database deployment & web server manager for your **NodeJS, Python, PHP, ASP.NET, Ruby, MySQL, MongoDB, Postgres, WordPress** (and etc...) applications!

It's blazingly fast and very robust as it uses Docker, nginx, LetsEncrypt and NetData under the hood behind its simple-to-use interface.

- ✓ CLI for automation and scripting
- ✓ **Web GUI** for ease of access and convenience
- ✓ No lock-in! Remove CapRover and your apps keep working!
- ✓ Docker Swarm under the hood for containerization and clustering
- ✓ Nginx (fully customizable template) under the hood for load-balancing
- ✓ Let's Encrypt under the hood for free SSL (HTTPS)

CAPROVER

What's this?

CapRover is an extremely easy to use app/database deployment & web server manager for your **NodeJS, Python, PHP, ASP.NET, Ruby, MySQL, MongoDB, Postgres, WordPress** (and etc...) applications!

It's blazingly fast and very robust as it uses Docker, nginx, LetsEncrypt and NetData under the hood behind its simple-to-use interface.

- ✓ CLI for automation and scripting
- ✓ Web GUI for ease of access and convenience
- ✓ No lock-in! Remove CapRover and your apps keep working!
- ✓ Docker Swarm under the hood for containerization and clustering
- ✓ Nginx (fully customizable template) under the hood for load-balancing
- ✓ Let's Encrypt under the hood for free SSL (HTTPS)

CAPROVER

What's this?

CapRover is an extremely easy to use app/database deployment & web server manager for your **NodeJS, Python, PHP, ASP.NET, Ruby, MySQL, MongoDB, Postgres, WordPress** (and etc...) applications!

It's blazingly fast and very robust as it uses Docker, nginx, LetsEncrypt and NetData under the hood behind its simple-to-use interface.

✓ CLI for automation and scripting

✓ Web GUI for ease of access and convenience

✓ No lock-in! Remove CapRover and your apps keep working!

→ ✓ Docker Swarm under the hood for containerization and clustering

✓ Nginx (fully customizable template) under the hood for load-balancing

✓ Let's Encrypt under the hood for free SSL (HTTPS)

CAPROVER

What's this?

CapRover is an extremely easy to use app/database deployment & web server manager for your **NodeJS, Python, PHP, ASP.NET, Ruby, MySQL, MongoDB, Postgres, WordPress** (and etc...) applications!

It's blazingly fast and very robust as it uses Docker, nginx, LetsEncrypt and NetData under the hood behind its simple-to-use interface.

- ✓ CLI for automation and scripting
- ✓ Web GUI for ease of access and convenience
- ✓ No lock-in! Remove CapRover and your apps keep working!
- ✓ Docker Swarm under the hood for containerization and clustering
- ✓ Nginx (fully customizable template) under the hood for load-balancing
- ✓ Let's Encrypt under the hood for free SSL (HTTPS)

CAPROVER

What's this?

CapRover is an extremely easy to use app/database deployment & web server manager for your **NodeJS, Python, PHP, ASP.NET, Ruby, MySQL, MongoDB, Postgres, WordPress** (and etc...) applications!

It's blazingly fast and very robust as it uses Docker, nginx, LetsEncrypt and NetData under the hood behind its simple-to-use interface.

- ✓ CLI for automation and scripting
- ✓ Web GUI for ease of access and convenience
- ✓ No lock-in! Remove CapRover and your apps keep working!
- ✓ Docker Swarm under the hood for containerization and clustering
- ✓ Nginx (fully customizable template) under the hood for load-balancing
- ✓ Let's Encrypt under the hood for free SSL (HTTPS)

PRÉ-REQUISITOS

PRÉ-REQUISITOS

- Servidor (de preferência Linux)
 - Pode ser sua **máquina local**, um **VPS (Digital Ocean, AWS, etc)** ou testar com **Play With Docker**.
- Endereço de domínio (ex. meusite.com.br)
 - **Máquina local:** utiliza domínio local
 - **VPS/Cloud:** apontar o DNS para o IP da máquina
 - **Play With Docker:** utiliza domínio temporário
- Docker

INSTALAÇÃO

INSTALAÇÃO

▪ Máquina local

```
$ docker run -p 80:80 -p 443:443 -p 3000:3000 -v /var/run/docker.sock:/var/run/docker.sock -v /captain:/captain -e  
MAIN_NODE_IP_ADDRESS='127.0.0.1' caprover/caprover
```

▪ Servidor VPS

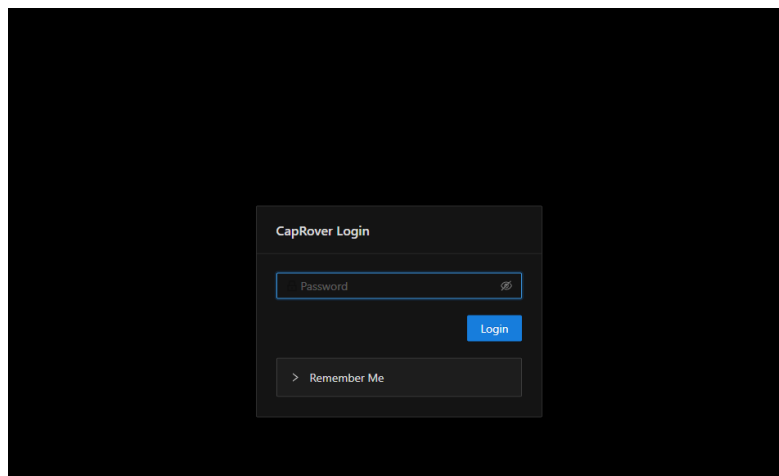
```
$ docker run -p 80:80 -p 443:443 -p 3000:3000 -v /var/run/docker.sock:/var/run/docker.sock -v /captain:/captain caprover/caprover
```

▪ Play With Docker

```
$ curl -L https://pwd.caprover.com | bash
```


INSTALAÇÃO

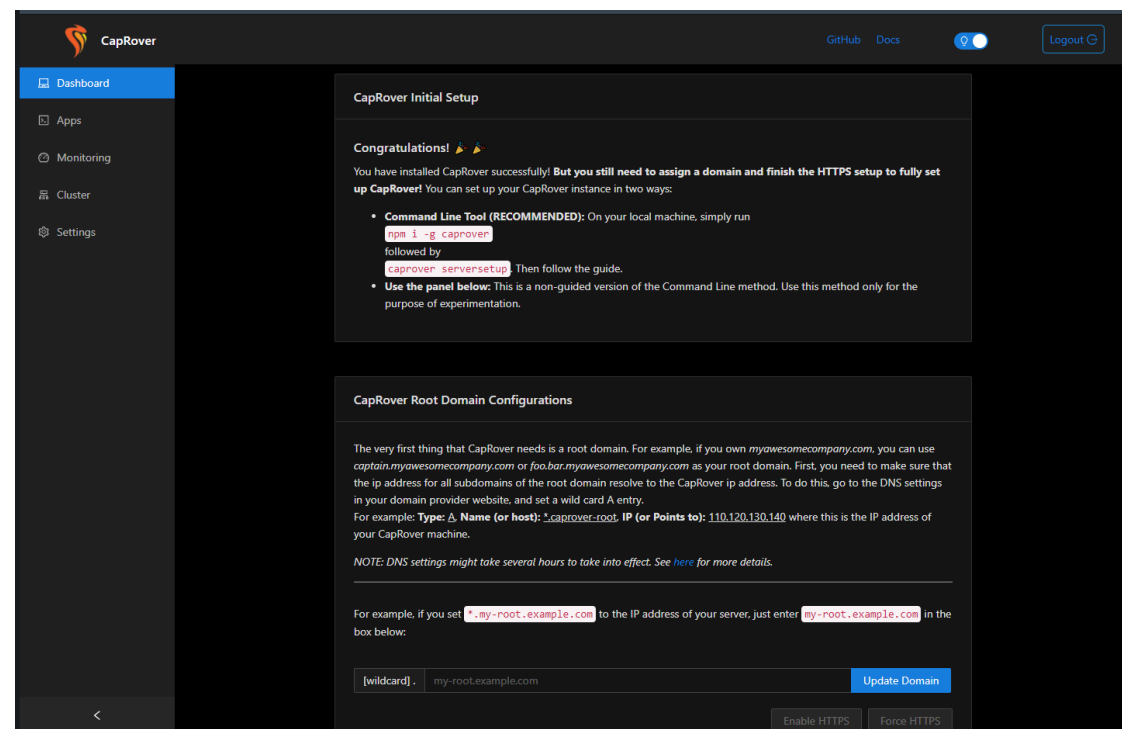
- Aguarde a inicialização dos containers e acesse:
 - **Máquina local:** <http://captain.captain.localhost>
 - **Servidor VPS:** <http://<ip-do-servidor>:3000>
 - **Play With Docker:** clicar na porta 3000 no topo da tela
- Entrar com a senha: `captain42`



EXPLORANDO O AMBIENTE

EXPLORANDO O AMBIENTE

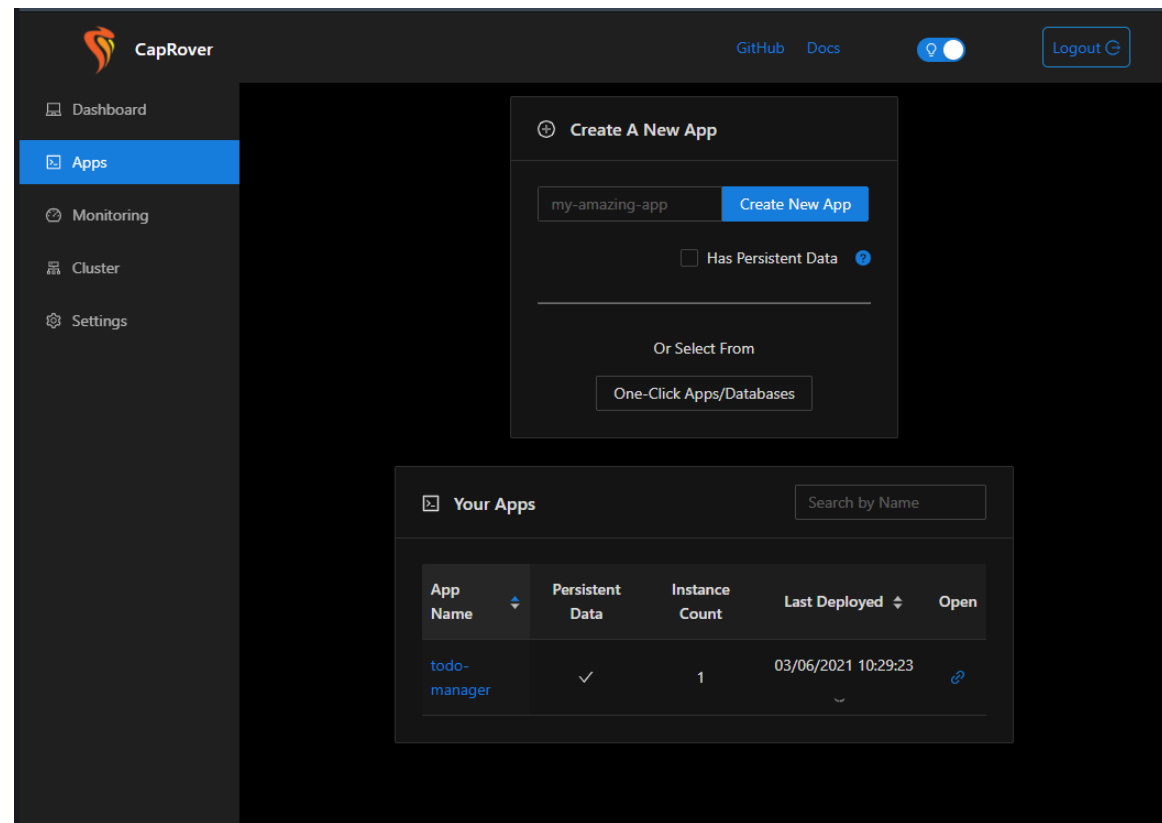
- Menu Dashboard
 - Contem instruções para iniciar
 - Permite configurar o domínio padrão



EXPLORANDO O AMBIENTE

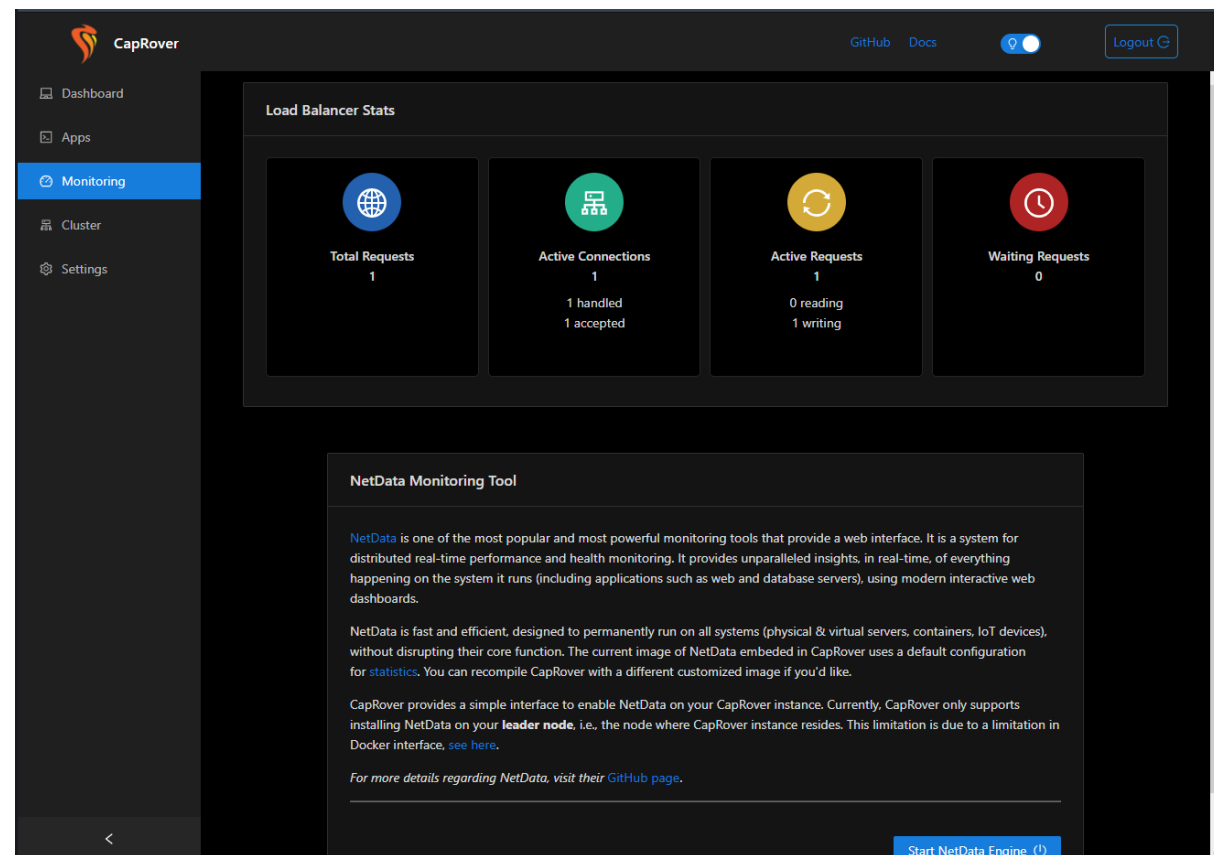
■ Menu Apps

- Exibe as aplicações criadas
- Permite criar novas aplicações
- Permite explorar as aplicações pré-configuradas (One-click apps)



EXPLORANDO O AMBIENTE

- Menu Monitoring
 - Exibe informações sobre a saúde da máquina
 - Permite ativação do NetData <https://www.netdata.cloud/>



EXPLORANDO O AMBIENTE

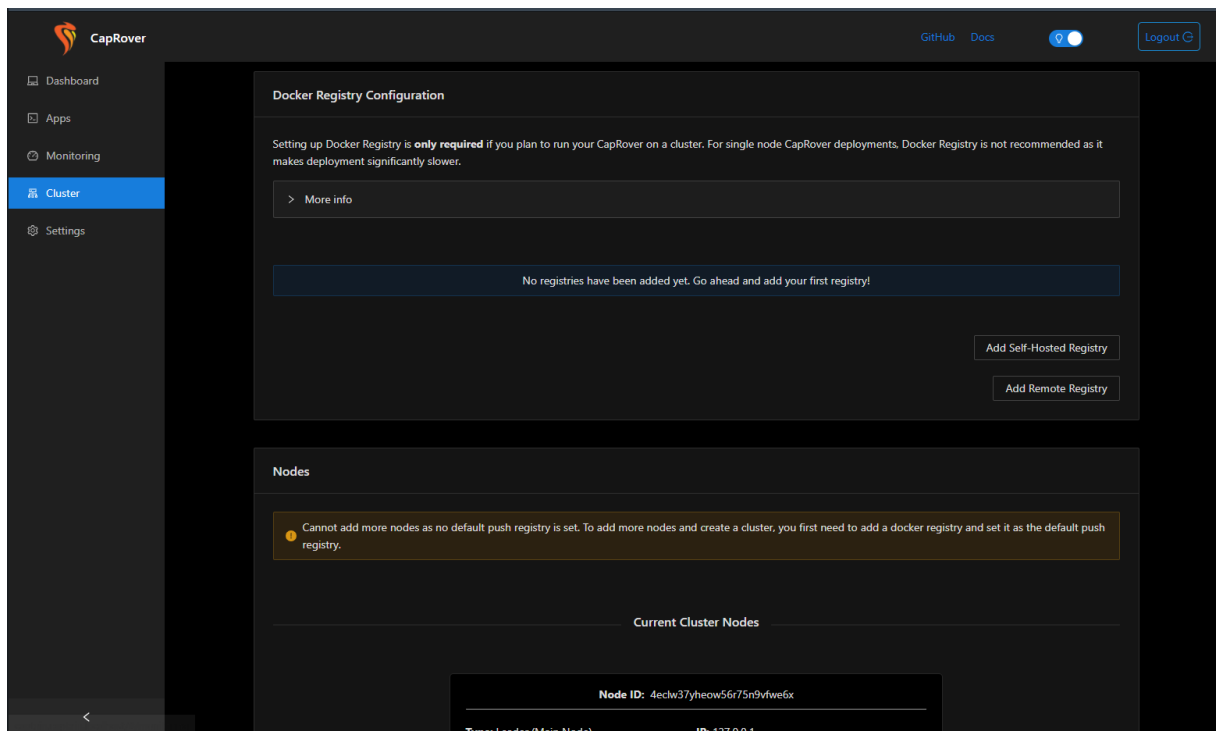
■ NetData



EXPLORANDO O AMBIENTE

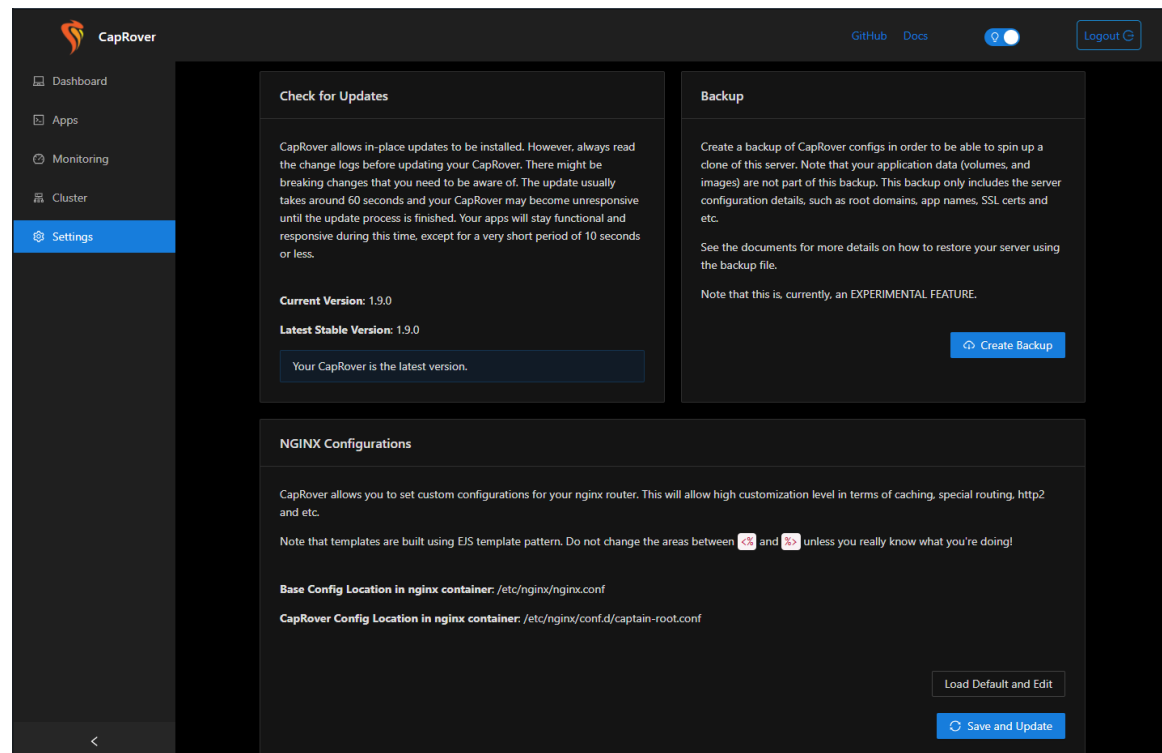
■ Menu Cluster

- Permite configuração de um Docker Registry
- Gerenciamento de nós do cluster
<https://docs.docker.com/engine/swarm/manage-nodes/>



EXPLORANDO O AMBIENTE

- Menu Settings
 - Atualizar o CapRover
 - Criar backups manuais
 - Customizar a configuração base do Nginx
 - Alterar a senha de acesso
 - Limpeza de disco: remover imagens Docker não utilizadas

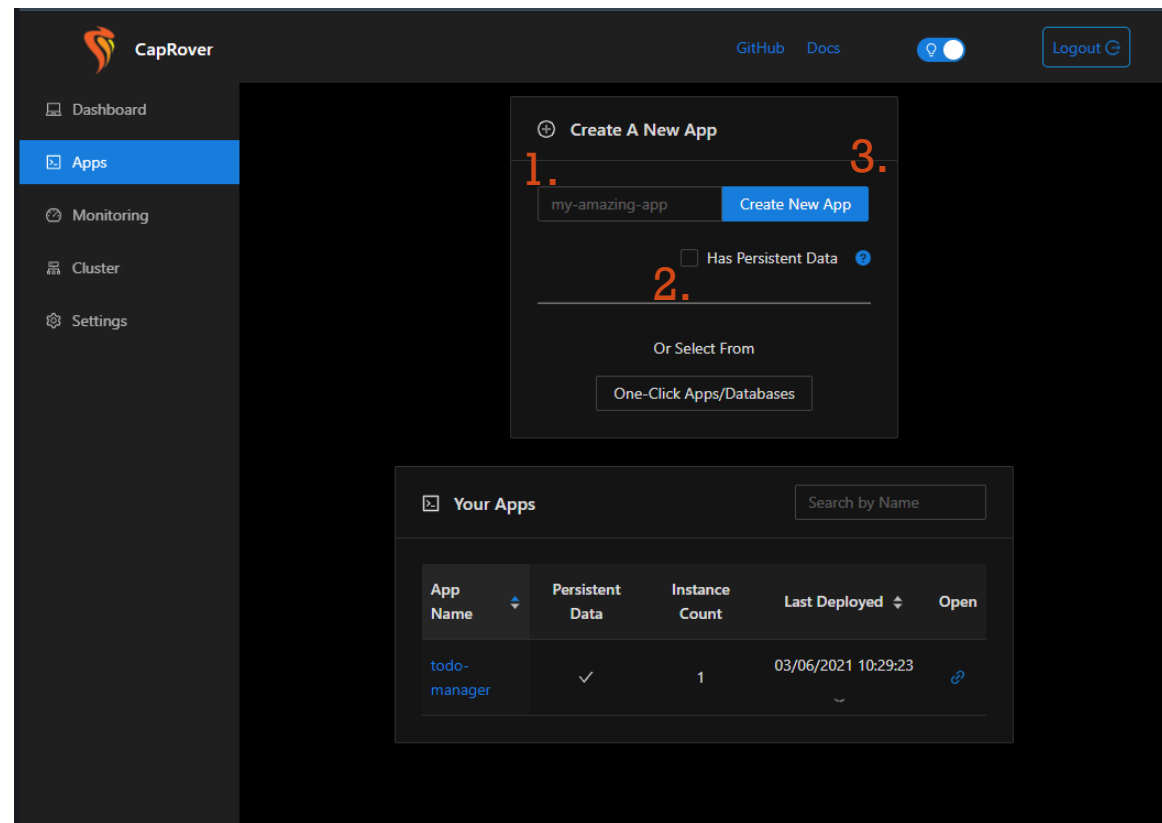


CRIANDO UM APP

CRIANDO UM APP

▪ Menu Apps

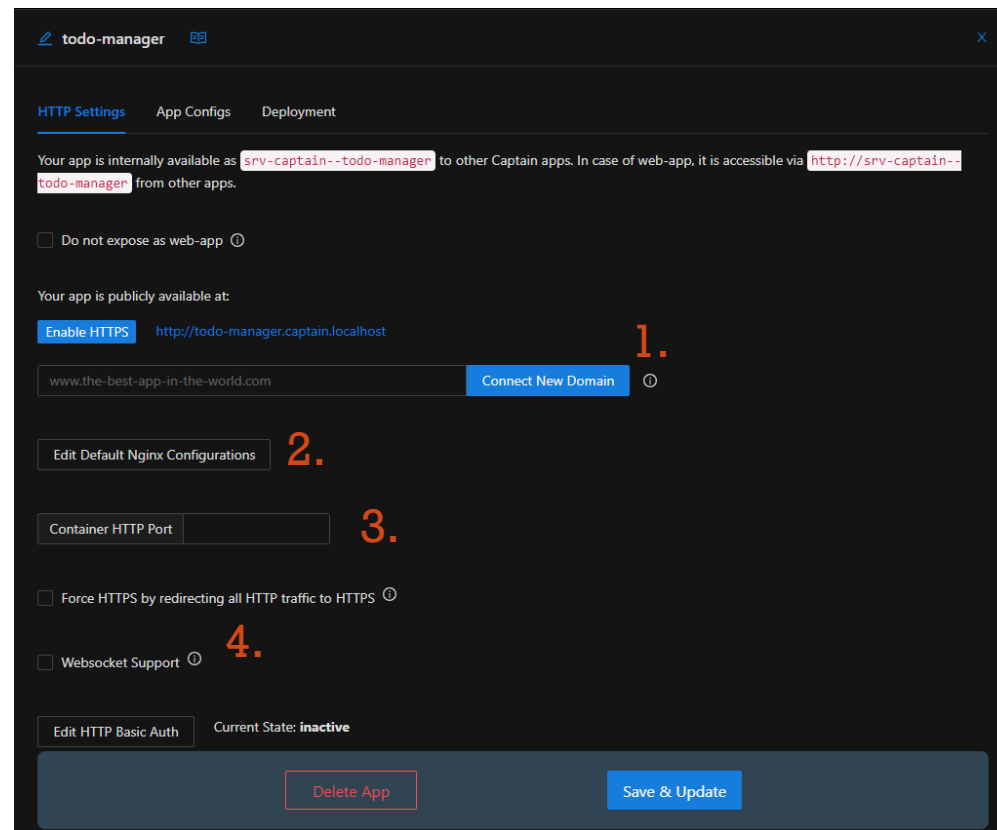
1. Preencha no nome do app:
Ex: todo-manager
(sem espaços e sem caracteres especiais)
2. Se o seu app precisa armazenar arquivos no disco, marque a opção “Has Persistent Data”.
3. Clique em “Create New App”



CRIANDO UM APP

■ Na aba Http Settings

1. Configurar domínio customizado.
(Apenas se usando em VPS com o DNS devidamente configurado)
2. Customizar a configuração do Nginx
(se necessário)
3. Definir a porta em que aplicação
estará rodando dentro do container.
Ex: 3000
4. Forçar uso de HTTPS ou habilitar
WebSockets



CRIANDO UM APP

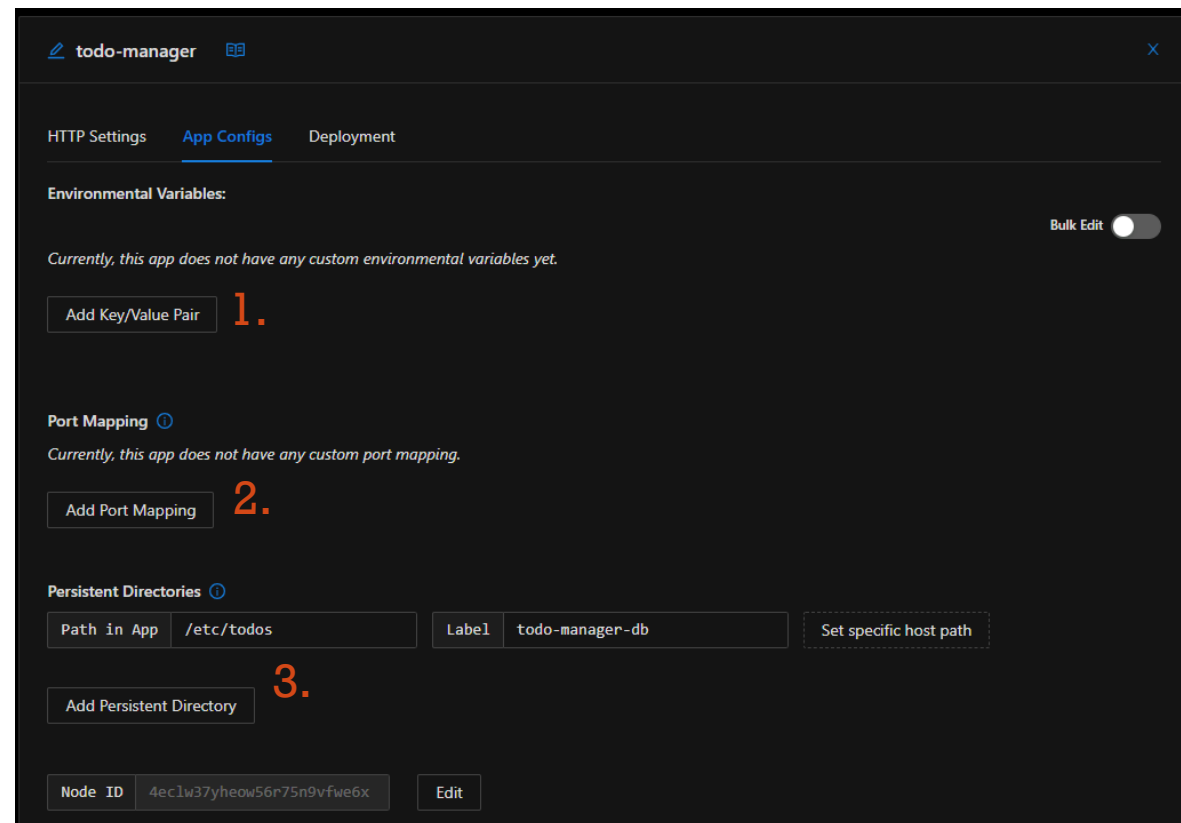
- Na aba App Configs

1. Configurar variáveis de ambiente
2. Mapear portas do container para a máquina
3. Configurar volumes para mapear diretórios da máquina para o container.

Ex:

Path in App: /etc/todos

Label: todo-manager-db



todo-manager

HTTP Settings App Configs Deployment

Environmental Variables: Bulk Edit

Currently, this app does not have any custom environmental variables yet.

Add Key/Value Pair 1.

Port Mapping

Currently, this app does not have any custom port mapping.

Add Port Mapping 2.

Persistent Directories

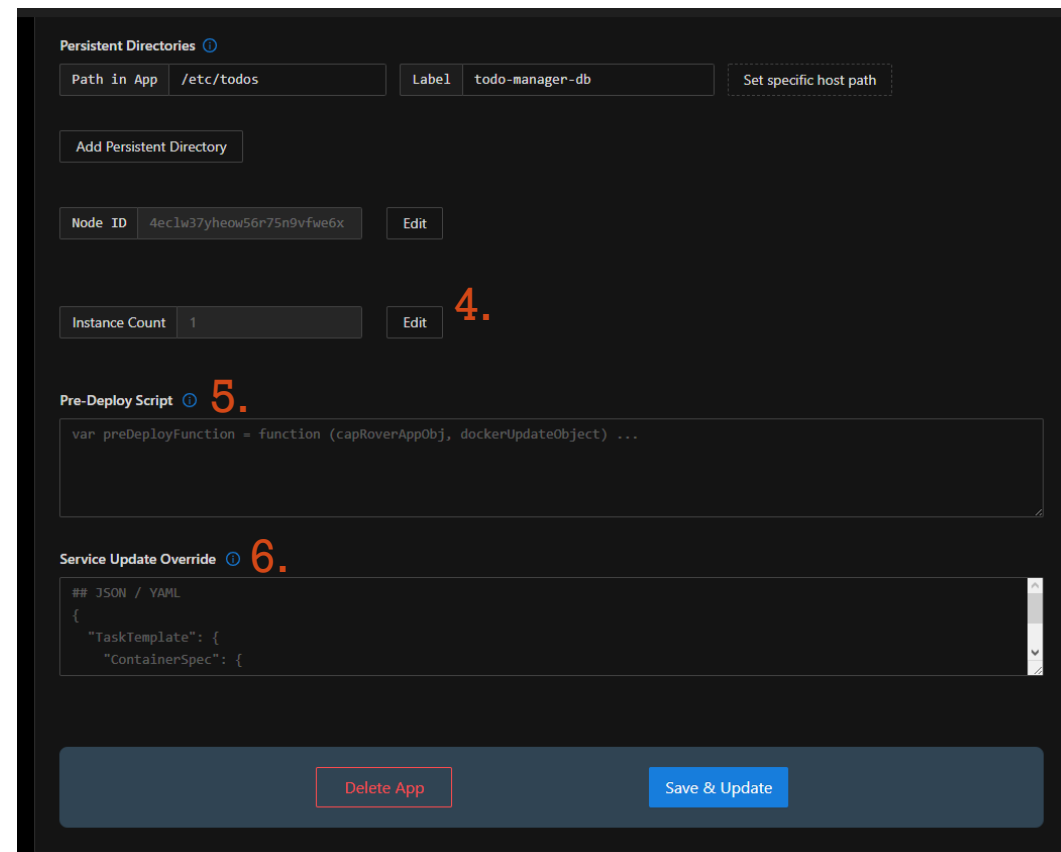
Path in App	/etc/todos	Label	todo-manager-db	Set specific host path
-------------	------------	-------	-----------------	------------------------

Add Persistent Directory 3.

Node ID	4ec1w37yheow56r75n9vfwe6x	Edit
---------	---------------------------	------

CRIANDO UM APP

- Na aba App Configs
- 4. Definir a quantidade de instâncias da aplicação.
- 5. Configurar script de deploys (em JavaScript)
- 6. Sobrescrever propriedade do Docker Swarm



The screenshot shows the 'App Configs' interface in CapRover. It includes sections for 'Persistent Directories', 'Node ID', 'Instance Count', 'Pre-Deploy Script', and 'Service Update Override'. The 'Instance Count' is set to 1, and the 'Pre-Deploy Script' contains a JavaScript function. The 'Service Update Override' section shows a YAML configuration for the task template and container spec. At the bottom, there are 'Delete App' and 'Save & Update' buttons.

Persistent Directories ⓘ

Path in App /etc/todos Label todo-manager-db Set specific host path

Add Persistent Directory

Node ID 4ec1w37yheow56r75n9vfw6x Edit

Instance Count 1 Edit 4.

Pre-Deploy Script ⓘ 5.

```
var preDeployFunction = function (capRoverAppObj, dockerUpdateObject) ...
```

Service Update Override ⓘ 6.

```
## JSON / YAML
{
  "TaskTemplate": {
    "ContainerSpec": {
```

Delete App Save & Update

CRIANDO UM APP

- Na aba Deployment

1. Log de construção da imagem

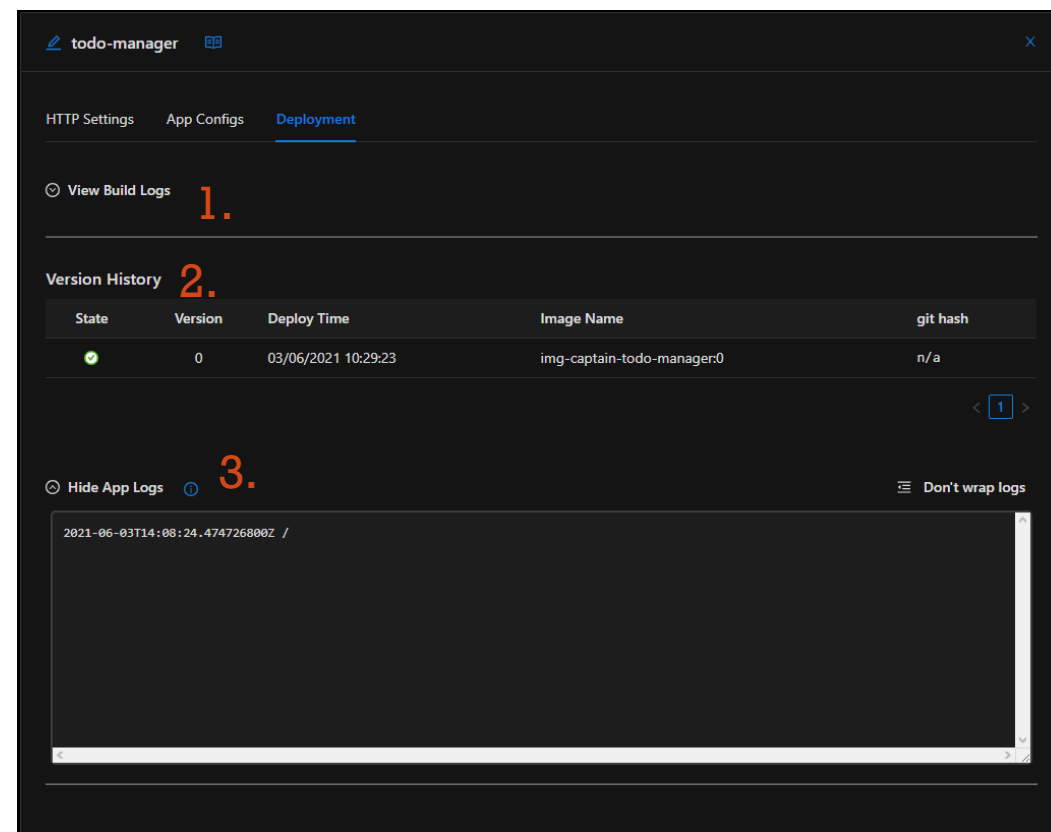
2. Histórico de imagens já construídas neste app.

- Permite voltar para uma versão específica

3. Log da aplicação.

Ex.

- JavaScript: console.log
- Java: System.out.print



CRIANDO UM APP

- Na aba Deployment

4. Deploy usando linha de comando

5. Deploy subindo um ZIP/TAR

6. Deploy com git

- Repositório:

<https://github.com/douglasjunior/node-app-with-docker.git>

- Branch: master

- Username: <seu usuário git>

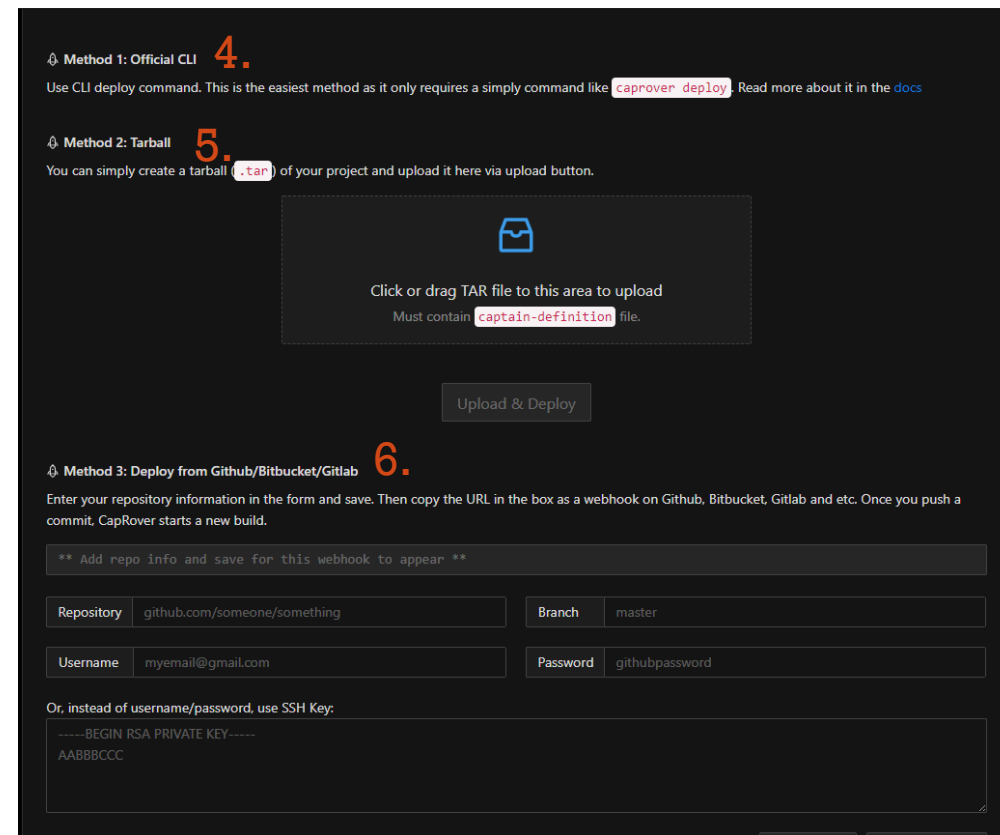
- Password: <sua senha git>

- Clicar Save & Update

- Clicar Force Build

🔗 Method 1: Official CLI **4.**
Use CLI deploy command. This is the easiest method as it only requires a simply command like `caprover deploy`. Read more about it in the [docs](#)

🔗 Method 2: Tarball **5.**
You can simply create a tarball (`.tar`) of your project and upload it here via upload button.


Click or drag TAR file to this area to upload
Must contain `captain-definition` file.

Upload & Deploy

🔗 Method 3: Deploy from Github/Bitbucket/Gitlab **6.**
Enter your repository information in the form and save. Then copy the URL in the box as a webhook on Github, Bitbucket, Gitlab and etc. Once you push a commit, CapRover starts a new build.

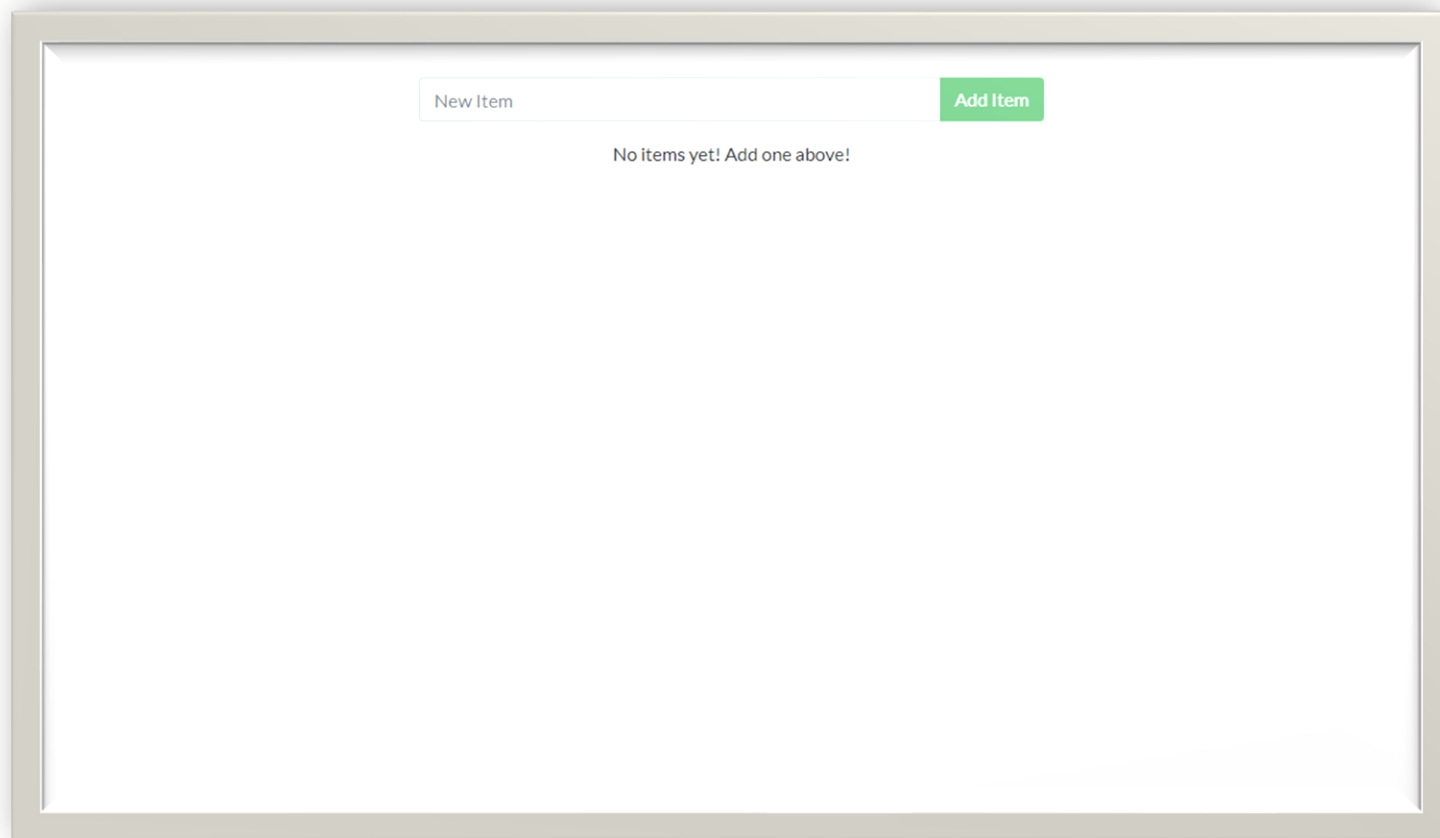
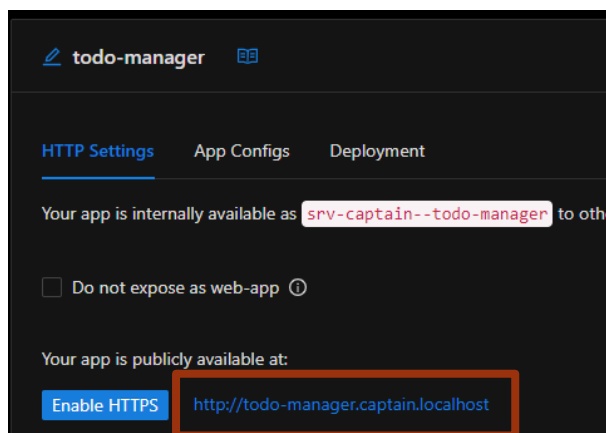
**** Add repo info and save for this webhook to appear ****

Repository	github.com/someone/something	Branch	master
Username	myemail@gmail.com	Password	githubpassword

Or, instead of username/password, use SSH Key:

-----BEGIN RSA PRIVATE KEY-----
AABBBCCC

CRIANDO UM APP

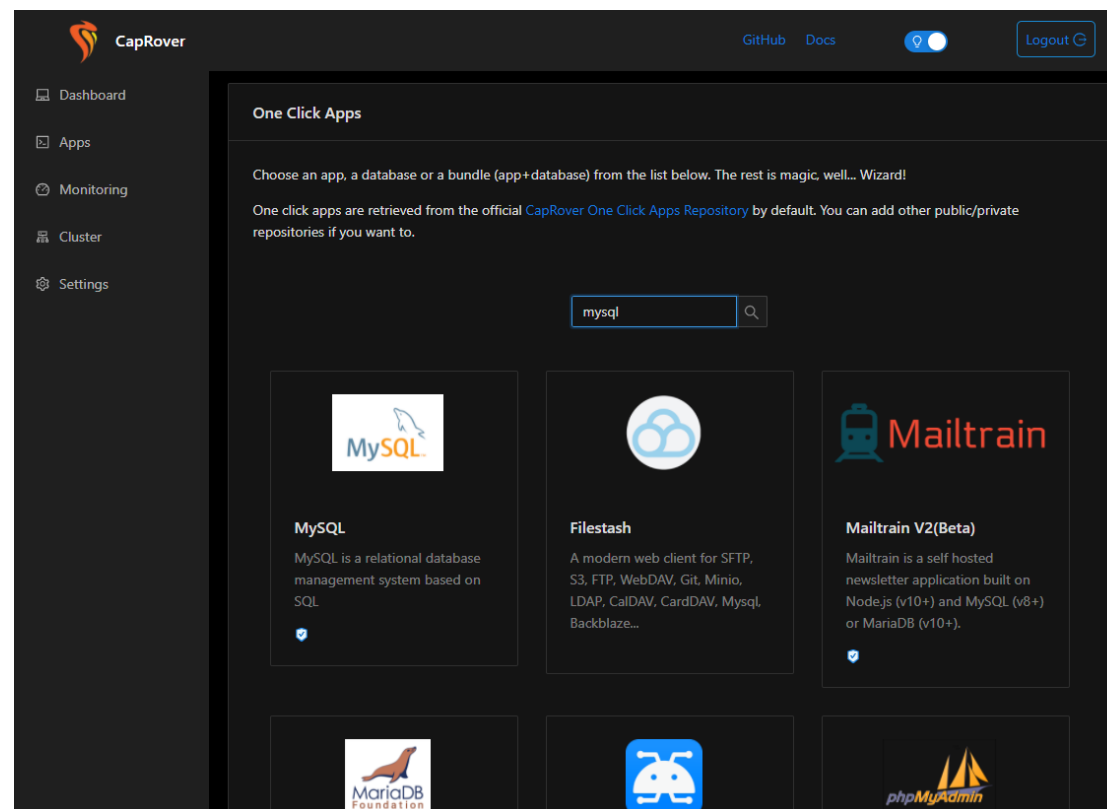


BÔNUS

(CONFIGURANDO MYSQL)

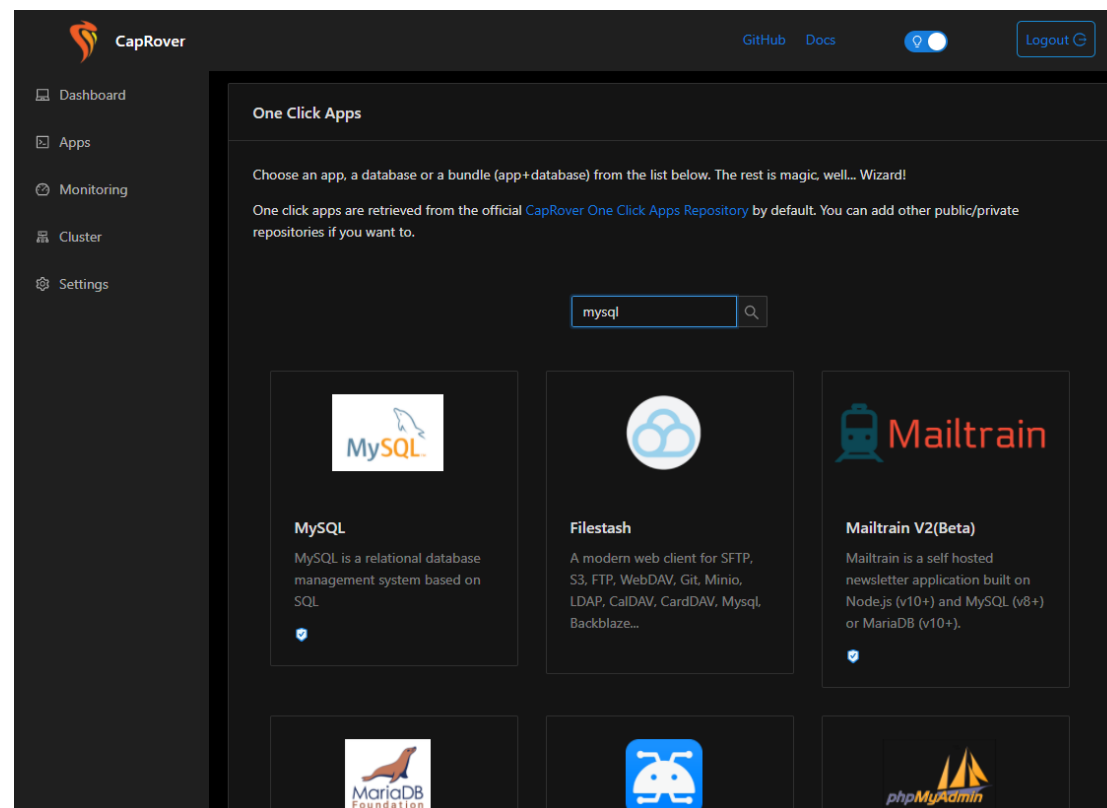
CONFIGURANDO MYSQL

- Em One-click Apps, procure por MySQL



CONFIGURANDO MYSQL

- Em One-click Apps, procure por MySQL



CONFIGURANDO MYSQL

- Definir o nome do “app” e a senha do usuário “root”

Setup your MySQL

MySQL

MySQL is the world's most popular open source database. With its proven performance, reliability and ease-of-use, MySQL has become the leading database choice for web-based applications, covering the entire range from personal projects and websites, via e-commerce and information services, all the way to high profile web properties including Facebook, Twitter, YouTube, Yahoo! and many more.

After installation on CapRover, it will be available as `srv-captain--YOUR_CONTAINER_NAME` at port 3306 to other CapRover apps.

Enter your MySQL Configuration parameters and click on next. It will take about a minute for the process to finish.

App Name

This is your app name. Pick a name such as `my-first-1-click-app`

MySQL Version

Check out their Docker page for the valid tags <https://hub.docker.com/r/library/mysql/tags/>

MySQL Root password

Deploy



Deploying your mysql

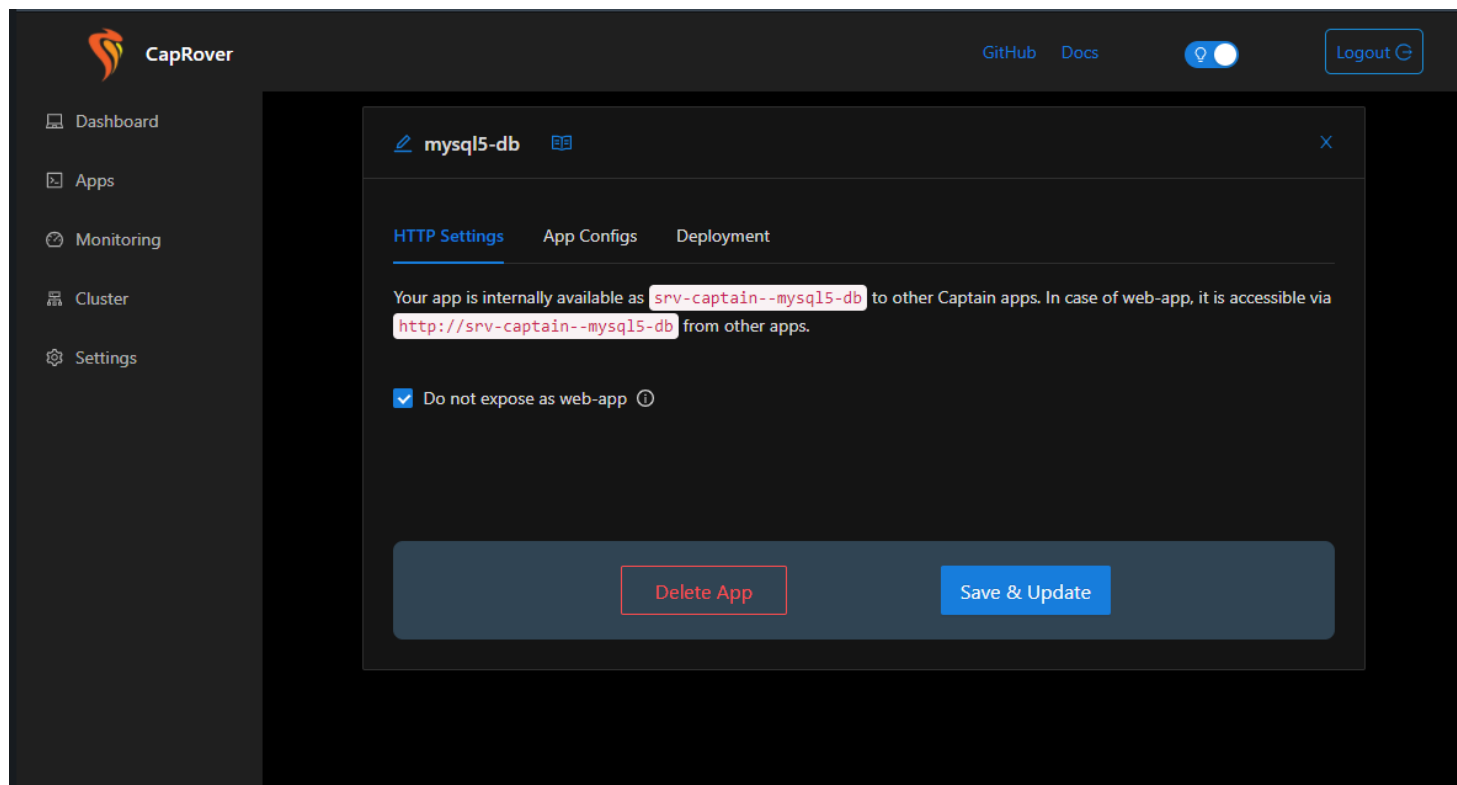
This process takes a few minutes to complete. DO NOT refresh this page and DO NOT navigate away!!!

Progress:

- ✓ Parsing the template
- ✓ Registering mysql5-db
- ✓ Configuring mysql5-db (volumes, ports, environmental variables)
- 4 Deploying mysql5-db (might take up to a minute)

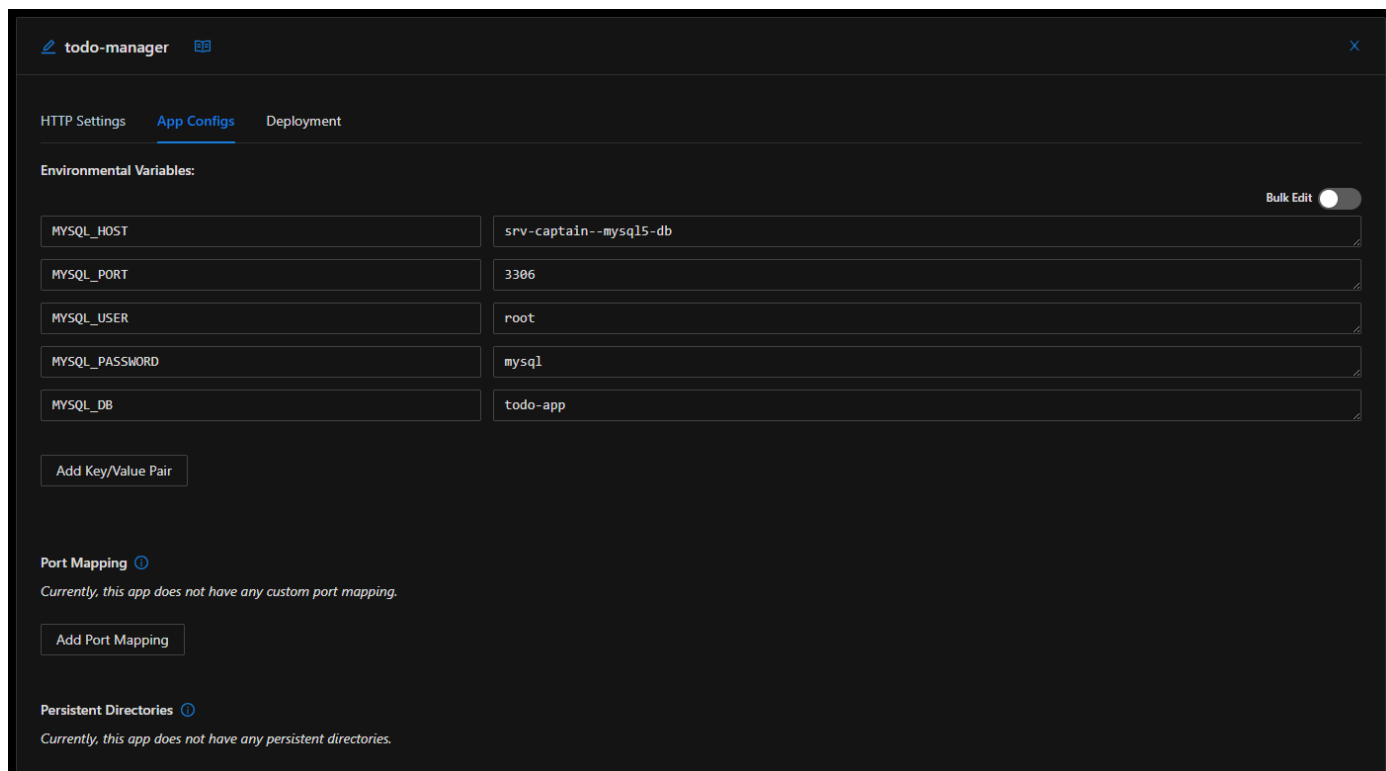
CONFIGURANDO MYSQL

- O MySQL já está rodando em um container Docker



CONFIGURANDO MYSQL

- Por fim, configurar as variáveis de ambiente na aplicação Node JS



The screenshot shows the 'App Configs' tab for an application named 'todo-manager'. Under the 'Environmental Variables' section, there is a table with the following data:

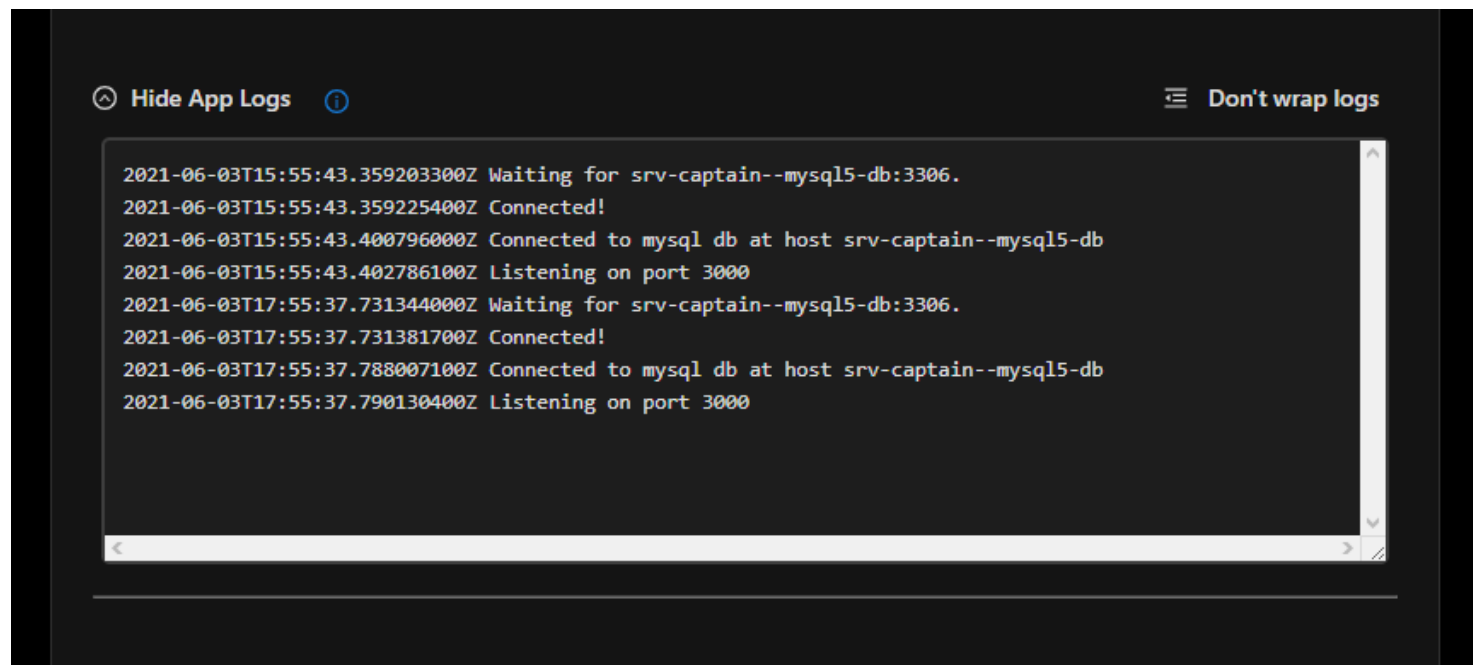
Variable	Value
MYSQL_HOST	srv-captain-mysql5-db
MYSQL_PORT	3306
MYSQL_USER	root
MYSQL_PASSWORD	mysql
MYSQL_DB	todo-app

Below the table is a button labeled 'Add Key/Value Pair'. Further down, there are sections for 'Port Mapping' and 'Persistent Directories', both indicating that no custom configurations are currently set.

(Obs: O volume usado para o SQLite pode ser removido.)

CONFIGURANDO MYSQL

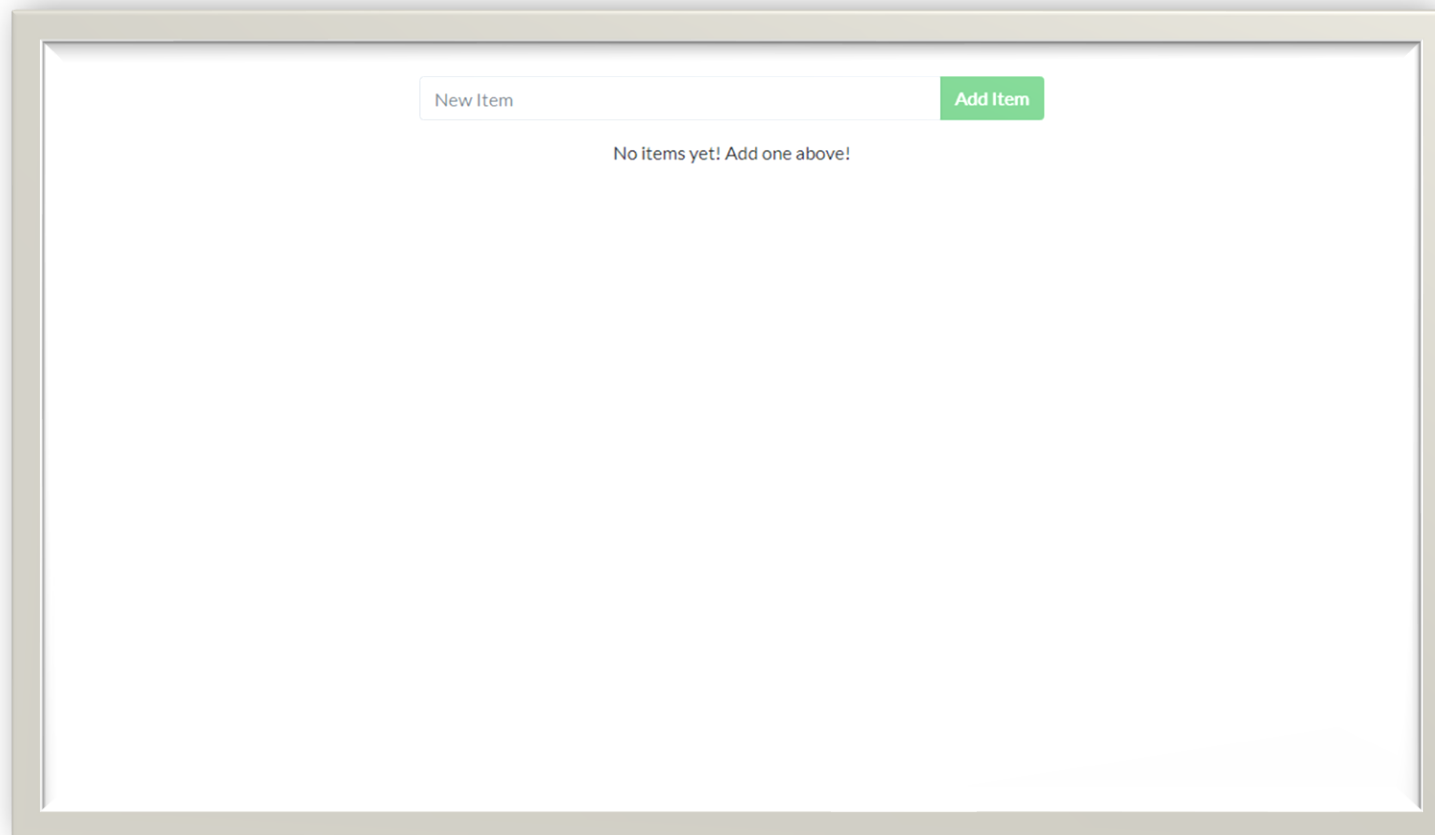
- Ao conferir nos Logs na aplicação, você deve ver a mensagem:
Connected to mysql db at host srv-captain--mysql5-db



```
2021-06-03T15:55:43.359203300Z Waiting for srv-captain--mysql5-db:3306.  
2021-06-03T15:55:43.359225400Z Connected!  
2021-06-03T15:55:43.400796000Z Connected to mysql db at host srv-captain--mysql5-db  
2021-06-03T15:55:43.402786100Z Listening on port 3000  
2021-06-03T17:55:37.731344000Z Waiting for srv-captain--mysql5-db:3306.  
2021-06-03T17:55:37.731381700Z Connected!  
2021-06-03T17:55:37.788007100Z Connected to mysql db at host srv-captain--mysql5-db  
2021-06-03T17:55:37.790130400Z Listening on port 3000
```

CONFIGURANDO MYSQL

- Acesse o app novamente, e desta vez estará rodando com MySQL:





OBRIGADO!