

## Tutorial – Usando Docker e RancherServer para aprendizado de container e deploy de uma aplicação de chat e Banco de dados.

O Rancher Server é um tipo de aplicação instalado sobre o Docker para monitoramento de container's, serviços posteriores instalados e fornece um meio de gerenciamento, monitoramento, bem como instanciação de container's ao Docker mais facilitada. É possível [ pós-instalação, óbvio ] - gerenciar os container's via linha de comando [ usando Rancher CLI ], ou através de UI.

[ Obs: parecia óbvio, por isso coloquei o pós-instalação ]: Mas o docker permite também gerenciar “manualmente” os container's [ digo de: criação, deleção, e requisições para criação de container's, leia-se imagens].

Então, neste tutorial iremos instalar e configurar o Rancher Server, aprender alguns conceitos, fazer o deploy de um chat em 2 container's e 1 banco de dados do mongoDB em um terceiro container, supervisionados por um load balance, que fará o balanceamento das requisições web.

### Referências: Quick Started Guide - RancherServer

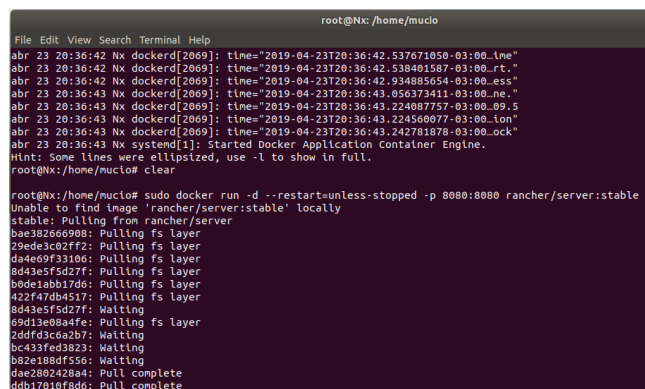
[ link ]: <https://rancher.com/docs/rancher/latest/en/quick-start-guide/#prepare-a-linux-host>

Este tutorial foi feito seguindo os passos do guia acima, assim alguns pré-requisitos são necessários, segue:

- Máquina Linux x64 v. 16,04 , que tenha kernel igual ou superior a 3.04;
  - Docker instalado, caso não tenha pode seguir estes passos [ <https://docs.docker.com/engine/installation/linux/ubuntu/linux/> ];
  - Conhecimento sobre Docker e container's;
- [ Observações ]: Você pode instalar o Rancher Server em sua máquina [ laptop, notebook, desktop ], em uma máquina virtual, ou um servidor físico; Porém, certifique-se que tenha disponível ao menos 1GB de memória RAM;

1º - Iremos rodar um comando no Docker para baixar e instalar uma imagem do Rancher Server [ lembrando: o mesmo roda sobre o docker ]:

[ Comando, terminal, install ]: `sudo docker run -d --restart=unless-stopped -p 8080:8080 rancher/server:stable`



```
root@Nx:/home/mucio
File Edit View Search Terminal Help
abr 23 20:36:42 Nx dockerd[2069]: time="2019-04-23T20:36:42.537671059-03:00_tne"
abr 23 20:36:42 Nx dockerd[2069]: time="2019-04-23T20:36:42.538401587-03:00_rft."
abr 23 20:36:42 Nx dockerd[2069]: time="2019-04-23T20:36:42.934885654-03:00_ess"
abr 23 20:36:43 Nx dockerd[2069]: time="2019-04-23T20:36:43.056373411-03:00_ne."
abr 23 20:36:43 Nx dockerd[2069]: time="2019-04-23T20:36:43.224087757-03:00_09.5"
abr 23 20:36:43 Nx dockerd[2069]: time="2019-04-23T20:36:43.224560877-03:00_ton"
abr 23 20:36:43 Nx dockerd[2069]: time="2019-04-23T20:36:43.242781878-03:00_lock"
abr 23 20:36:43 Nx systemd[1]: Started Docker Application Container Engine.
Hint: Some lines were ellipsized, use -l to show in full.
root@Nx:/home/mucio# clear

root@Nx:/home/mucio# sudo docker run -d --restart=unless-stopped -p 8080:8080 rancher/server:stable
Unable to find image 'rancher/server:stable' locally
stable: Pulling from rancher/server
bae382666908: Pulling fs layer
29ede3c02ff2: Pulling fs layer
d4e69f33106: Pulling fs layer
8d43e5f5d27f: Pulling fs layer
b0de1abb17d6: Pulling fs layer
422f47db4517: Pulling fs layer
8d43e5f5d27f: Waiting
69d13e08a4fe: Pulling fs layer
2ddf3c0a2b7: Waiting
bc433fed3823: Waiting
b82e188df556: Waiting
dae2802428a4: Pull complete
ddb17610f8d6: Pull complete
```

2º - Se achar necessário, pode executar o comando de logs do docker, assim:

[ Comando, terminal, logs ]: `sudo docker logs -f <CONTAINER_ID>`

```

root@Nx: /home/mucio
File Edit View Search Terminal Help
priceless_knuth
18cabefa6291 hello-world "/hello"
8 days ago Exited (0) 8 days ago
adoring_heyrovsky
root@Nx:/home/mucio# sudo docker logs -f accb6db17176
190424 0:39:10 [Note] /usr/sbin/mysqld (mysqld 5.5.62-0ubuntu0.14.04.1) starting as process 26 ...
Uptime: 2 Threads: 1 Questions: 2 Slow queries: 0 Opens: 33 Flush tables: 1
Open tables: 26 Queries per second avg: 1.000
Setting up database
Importing schema
CATTLE_AGENT_PACKAGE_HOST_API_URL=/usr/share/cattle/artifacts/host-api.tar.gz
CATTLE_AGENT_PACKAGE_PER_HOST_SUBNET_URL=/usr/share/cattle/artifacts/rancher-per-host-subnet.zip
CATTLE_AGENT_PACKAGE_PYTHON_AGENT_URL=/usr/share/cattle/artifacts/go-agent.tar.gz
CATTLE_AGENT_PACKAGE_WINDOWS_AGENT_URL=/usr/share/cattle/artifacts/go-agent.zip
CATTLE_API_UI_URL=//releases.rancher.com/api-ui/1.0.8
CATTLE_CATTLE_VERSION=v0.103.75
CATTLE_DB_CATTLE_DATABASE=mysql
CATTLE_DB_CATTLE_MYSQL_HOST=localhost
CATTLE_DB_CATTLE_MYSQL_NAME=cattle
CATTLE_DB_CATTLE_MYSQL_PORT=3306
CATTLE_DB_CATTLE_USERNAME=cattle
CATTLE_GRAPHITE_HOST=
CATTLE_GRAPHITE_PORT=
CATTLE_HOME=/var/lib/cattle

```

→ para verificar o container\_id:

**[ Comando, terminal, container ]: docker container ls -a**

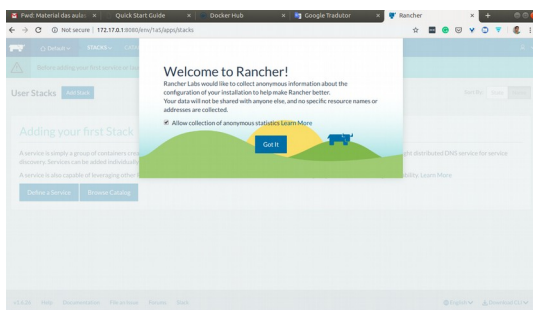
```

root@Nx:/home/mucio
File Edit View Search Terminal Help
root@Nx:/home/mucio# docker container ls -a
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS
accb6db17176       rancher/server:stable /usr/sbin/entry /usr... 3 minutes ago      Up 3 minutes       3306/tcp, 0.0.0.0:8080->8080/tcp
38c793be2ff2       ubuntu:16.04       stoic_diffie            7 days ago         Exited (127) 7 days ago
ec7ced414692       ubuntu:16.04       blissful_shannon       7 days ago         Exited (1) 7 days ago
788b520b9a2       ubuntu:16.04       ps -ef /bin/bash       7 days ago         Exited (1) 7 days ago
8e742b71d90       ubuntu:16.04       silly_mccarthy         7 days ago         Exited (1) 7 days ago
1e87239828eb       ubuntu:16.04       clever_einstein         7 days ago         Exited (1) 7 days ago
3f071ce299ee       ubuntu:16.04       elegant_goodall        7 days ago         Exited (1) 7 days ago
c735925a7691       ubuntu:16.04       bash                   8 days ago         Exited (0) 8 days ago
67b93f56d13f       ubuntu:16.04       hardcore_ebonyshv     8 days ago         Exited (0) 8 days ago
8af4533aa9e       ubuntu:16.04       gallant_cohen         8 days ago         Exited (0) 8 days ago
ed9697ee84d9       hello-world        upbeat_snyder          8 days ago         Exited (0) 8 days ago
18cabefa6291       hello-world        priceless_knuth        8 days ago         Exited (0) 8 days ago

```

3º – A partir deste passo, a maioria do gerenciamento será via Interface.

3.1 → Acesse o Rancher Server, via navegador usando seu <seuEndeçoIP:8080>;



→ Não é possível acessar usando: http://localhost:8080 ou <http://127.0.0.1>;

→ Será preciso, saber o IP da máquina;

→ Para verificar seu IP, execute:

**[ comando, IP, dica ]: hostname -I ou ifconfig**

4º – Vamos adicionar um Host:

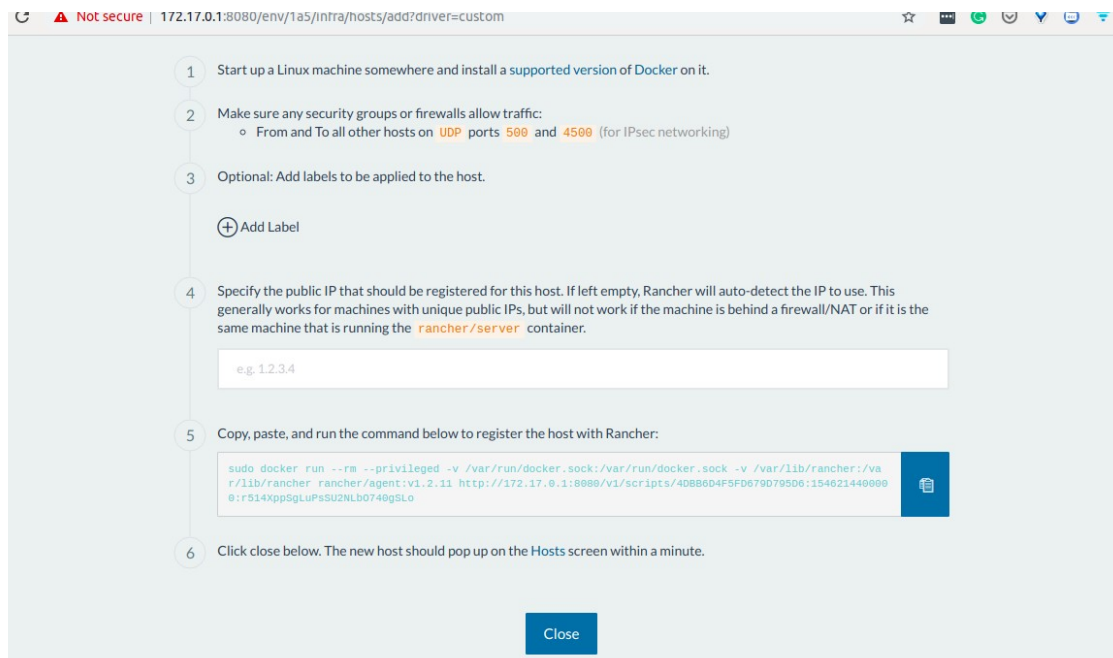
Vamos adicionar um host ao Rancher Server; Em aplicações reais, recomenda-se o uso de um host dedicado ao Rancher Server.

4.1 – Acesse: **Infraestructure** [ Você vai ser direcionado para a página de **Hosts** ];

4.2 → Clique em **Add Hosts** [por Padrão há uma opção pré-selecionada chamada - **Custom**];

4.3 → Abaixo, haverá um comando [passo 5 da imagem], **copie-o**,

4.4 → Abra o terminal e **cole** [ use **ctrl+shift+v**];

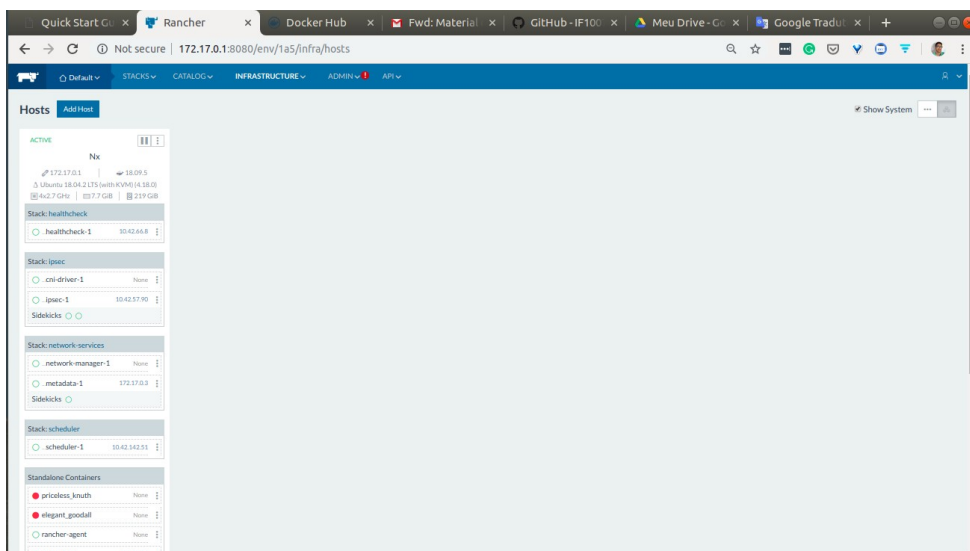


```
root@Nx: /home/mucio
File Edit View Search Terminal Help
command 'iwconfig' from deb wireless-tools
command 'ifconfig' from deb ipmiutil
command 'ifconfig' from deb net-tools

Try: apt install <deb name>

root@Nx:/home/mucio# sudo docker run --rm --privileged -v /var/run/docker.sock:/var/run/docker.sock
-v /var/lib/rancher:/var/lib/rancher rancher/agent:v1.2.11 http://172.17.0.1:8080/v1/scripts/408B6
D4F5FD679D795D6:1546214400000:r514XppSgLUpsSU2NLb0740gSL0
Unable to find image 'rancher/agent:v1.2.11' locally
v1.2.11: Pulling from rancher/agent
b3e1c725a85f: Pull complete
6a710864a9fc: Pull complete
d0ac3b234321: Pull complete
87f567b5cf58: Pull complete
063e24b217c4: Pull complete
d0a3f58caef0: Pull complete
16914729cfd3: Pull complete
bbad862633b9: Pull complete
3cf9849d7f3c: Pull complete
Digest: sha256:0fba3fb10108f7821596dc5ad4bfa30e93426d034cd3471f6ccd3afb5f87a963
Status: Downloaded newer image for rancher/agent:v1.2.11
INFO: Running Agent Registration Process, CATTLE_URL=http://172.17.0.1:8080/v1
```

4.5 → Volte ao navegador, e verá uma tela como a da imagem abaixo. Do lado esquerdo é possível ver o host já sendo monitorado, bem como os serviços do mesmo;



[ Observação ]: Ao logar, você estará no ambiente padrão [ interface ] do Rancher Server, sendo possível selecionar outros. É preciso executar serviços de infraestrutura para que seja possível fazer uso de recursos avançados, como dns, metadados, rede e verificações de integridade. Essas pilhas de infraestrutura pode ser acessada através do menu **Stacks > Infrastructure**.

Essas pilhas poderão estar em 2 estados, o primeiro estado **unhealthy** - até que um host seja adicionado ao Rancher. Depois de adicionar um host, é recomendável aguardar até que todas as pilhas de infraestrutura estejam, no segundo estado, **active** antes de adicionar outros serviços.

Pronto! Até aqui, já temos o Rancher Server rodando sobre o docker e com alguns serviços sendo monitorados por ele, no entanto, ainda não fizemos nenhum deploy.

## Criando um Container via Interface do Usuário

Você pode adicionar um serviço a um **Stack** de serviços, ou criar um stack novo personalizado;

5º - Vamos criar, um stack de serviços, para isso vá no menu **Stack > All > Add Stack**;

5.1 - Nomeie, seu stack e clique em **Create**, você terá algo desse tipo [ imagem ];

Aguarde enquanto o processo de criação termina [ e demora um pouco, é normal ];

6º Agora vamos adicionar Serviços:

6.1 → Vá em **Stacks > all**;

6.2 → Clique sobre o criado anteriormente, procure pela opção **Add Service** [ Tela abaixo ]

6.3 → Nomeie como **first-container**;

6.4 → Clique em **Create**;

[Observação ]: É possível criar um container dentro do Rancher Server via terminal, desta forma:

**\$ docker run -d -it --name=second-container ubuntu:14.04.2**

**onde:**

**name:** nome do container;

**ubuntu:** nome do sistema que servirá de host e sua respectiva **Versão**;

**-it:** significa --interactive --tty. Usado para fixar a linha de comando com o contêiner, assim, todos os comandos são executados pelo bash de dentro do contêiner;

**-d:** executa o container em background;

**Pronto! Até agora aprendemos a criar e trabalhar com o Rancher Server. Vamos fazer uma implantação na prática!.**

### Criando uma aplicação multi-container

É possível implantar aplicações em diversos container's e tê-las comunicando-se entre si. Iremos implantar desta forma,

- Um load balancer para redirecionar o tráfego de internet para o 'letschat';
- Um web service que utilizará dois container's;
- Uma base de dados – mongo – que utilizará um container;

1 → Crie um Stack e nomei como three-container;

2 → Clique em **Add Service**,

2.1 → Nomei o serviço como **database**;

2.2 → No campo image, escreva **mongo**;

2.3 → Clique em **Create**;

The screenshot shows the 'Add Service' interface in Rancher. At the top, there's a 'Scale' section with a slider set to 1 and two radio buttons: 'Run 1 container' (selected) and 'Always run one instance of this container on every host'. Below this is a button labeled 'database' and a link '+ Add Sidekick Container'. The main form has two columns: 'Name' and 'Description'. The 'Name' field contains 'database'. The 'Description' field contains 'e.g. My Application'. Below the 'Name' field is a 'Select Image\*' dropdown menu with 'mongo' selected. To the right of this dropdown is a checkbox labeled 'Always pull image before creating' which is checked. At the bottom, there are two expandable sections: 'Port Map' and 'Service Links', both with a '+' icon.

3 → Clique em **Add Service**,

3.1 → [ ATENÇÃO! ] **Deslize o slide**, fazendo com que modifique o **número de container's para 2**;

3.2 → Nomei o serviço como **web**;

3.3 → No campo image, escreva **sdelements/lets-chat**;

3.4 → No campo abaixo chamado **Service Links** [ expando-o ],

3.5 → Em **Destination Name**, escolha **database** [container que tem o seu banco de dados **mongo**];

3.6 → Nomei como **mongo**, no campo **Description**;

3.7 → Clique em **Create**;

**Scale**

☒ Run 1 container
 ☐ Always run one instance of this container on every host

web [+ Add Sidekick Container](#)

---

**Name**

**Description**

**Select Image\***

☒ Always pull image before creating

[+ Port Map](#)

[+ Service Links](#)

Destination Service  > As Name

Stack: <span>Three-Container</span> <span>Add Service</span> <span>Degraded</span>					
Description: MyStackAppLetsChat					
Active	database	Image: mongo	Service	1 Container	
Activating	web (Container should have been running but is in error state. Check logs for more information.: Error response from daemon: Get https://registry-1.docker.io/v2/: dial tcp: lookup registry-1.docker.io: no such host)	Image: sdelements/lets-chat	Service	2 Containers	

**[ Observação ]:** Pode aparecer uma mensagem como na imagem acima, na hora da implantação da aplicação, mas é só aguardar, enquanto o processo termina;

E também, um alerta no seu Stack, conforme imagem abaixo, porém é preciso aguardar um pouco enquanto o processo é finalizado.

All Stacks <span>Add Stack</span> <span>Add from Catalog</span> <span>Sort By: State Name</span>					
+	healthcheck	Up to date	1 Service	1 Container	
+	ipsec	Up to date	2 Services	4 Containers	
+	network-services	Up to date	2 Services	3 Containers	
+	scheduler	Up to date	1 Service	1 Container	
+	ServicosTutorialVCG Este stack servirá como base de aprendizado para a dis...	Add Service	1 Service	1 Container	
+	Three-Container MyStackAppLetsChat	Add Service	2 Services	3 Containers	

Por fim, criaremos o load balancer, para isso:  
4 → Clique em **Add Service**,



- 4.1 → Clique em **Add Load Balancer**;
- 4.2 → Nomeie o serviço como **letschatapplb**;
- 4.3 → No campo **Port**, digite **80**;
- 4.4 → Em **Target**, selecione **container/web**;

**Add Load Balancer** ⓘ

**Scale**

☒ Run 1 container

☐ Always run one instance of this container on every host

**Name**

**Description**

**Port Rules** ⊕ Add Service Rule ⊕ Add Selector Rule

Access*	Protocol*	Request Host	Port*	Path	Target*	Port*
<input type="button" value="⌵"/>	<input type="button" value="Public"/>	<input type="button" value="HTTP"/>	<input type="text" value="e.g. example.com"/>	<input type="text" value="e.g. 80"/>	<input type="text" value="e.g. /foo"/>	<input type="button" value="Choose a Service..."/>
						<input type="text" value="e.g. 80"/>

Host and Path rules are matched top-to-bottom in the order shown. Backends will be named randomly by default; to customize the generated backends, provide a name and then refer to that in the custom haproxy.cfg. [Show custom backend names.](#) [Show host IP address options.](#)

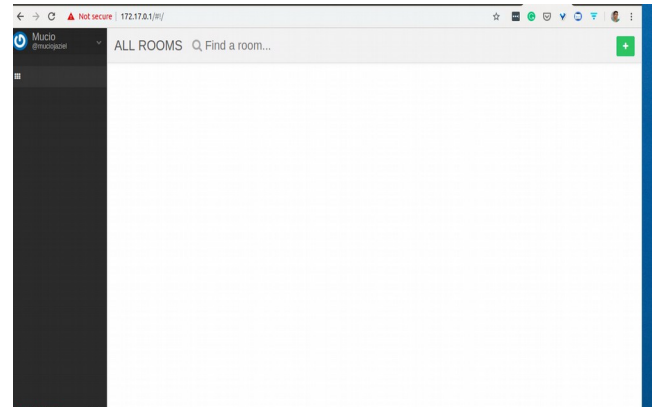
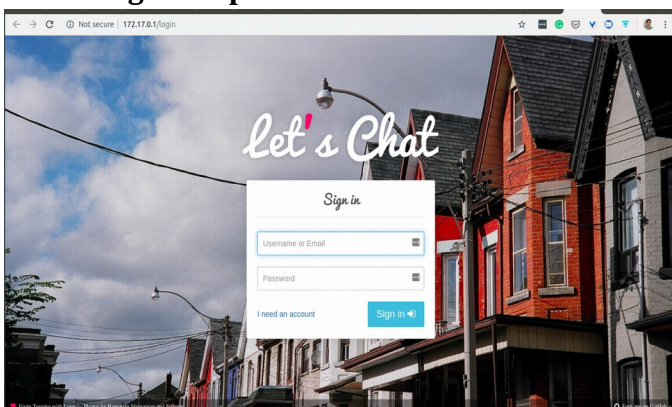
**Stack:** ⚠ Three-Container Add Service ⌵ ⌵ ⌵ ⚠ Degraded ⚙ ⋮

**Description:** MyStackAppLetsChat

<span>🟢 Active</span>	database ⓘ	Image: mongo	Service	1 Container	<span>⌵</span> <span>⌵</span> <span>⌵</span>
<span>🟡 Activating</span>	letschatapplb (In Progress) ⓘ	To: web Ports: 80/tcp	Load Balancer	1 Container	<span>⌵</span> <span>⌵</span> <span>⌵</span>
<span>🟢 Active</span>	web ⓘ	Image: sdelements/lets-chat	Service	2 Containers	<span>⌵</span> <span>⌵</span> <span>⌵</span>

4.5 → No campo **Port [a frente de target]**, digite **8080**, o web service escuta nessa porta;

Pronto, agora que terminamos o tutorial, você pode acessar a aplicação através de:  
**endereçoIP:80** ou somente **endereçoIP**;  
**E terá algo do tipo:**



É possível criar os container's, usando o Rancher CLI. Na documentação, é possível seguir o passo-a-passo;