

Docker & Qt

Igor Pires dos Santos

igor.pires@ice.ufjf.br

Professor: Bernardo M. Rocha



Programa de Pós-Graduação em Modelagem Computacional
Universidade Federal de Juiz de Fora

2 de setembro de 2020

Docker - Introdução

Docker - Instalação

Docker - Contêineres

Docker - Imagens

Docker - Compose & Builder

Qt - Introdução

Qt - Instalação

Qt - Interface Gráfica

Docker & Qt - Padrões de Projeto

- ▶ **O que é o Docker?**
- ▶ O Docker foi criado como um projeto *open source*, em 2013, pela então nomeada DotCloud. Pouco mais de um ano após o lançamento do projeto a empresa passou por uma reestruturação.
- ▶ Isso se deu pela grandiosidade que o Docker tomou, englobando a empresa original e criando-se a Docker TM.

- ▶ Desde então diversas empresas mudaram seus sistemas de *software management* para a estrutura Docker. Isso se dá pela praticidade em migrar e manter o sistema utilizando o Docker
- ▶ Especialistas da área dizem que grandes revoluções assim no meio da computação não inevitáveis, que as grandes empresas precisam se adaptar para correr atrás desta tecnologia ou serão devoradas.

Docker Momentum



450+

Docker EE
commercial
customers



37B

Container
downloads



15K

Job listings on
LinkedIn



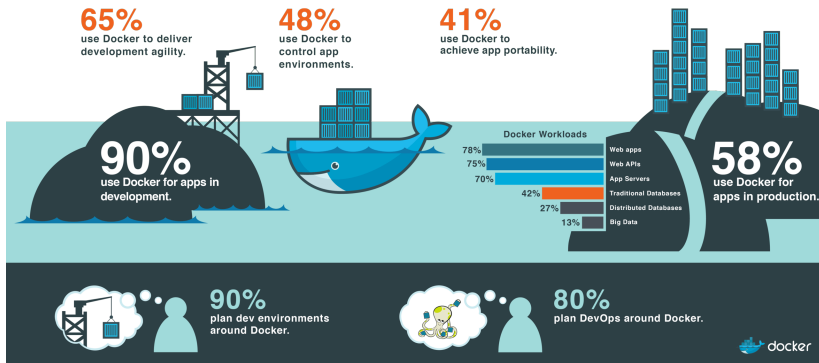
3.5M

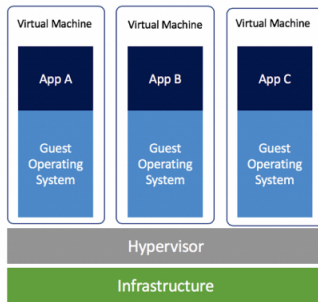
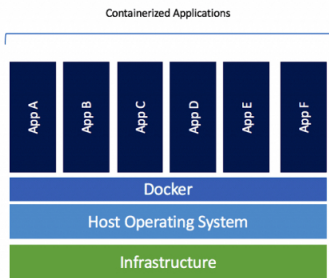
Dockerized
apps



200+

Active Docker
user groups





- ▶ Há algum tempo o Docker não é mais referenciado como uma alternativa à máquinas virtuais, ou contêineres em tempo de execução
- ▶ O Docker assim como a maioria das tecnologias atuais está em constante evolução. Portanto, os materiais de referência ao Docker também estão em constante atualização. Existem diferentes versões da ferramenta Docker, aqui fala-se exclusivamente do Docker CE (Community Edition).

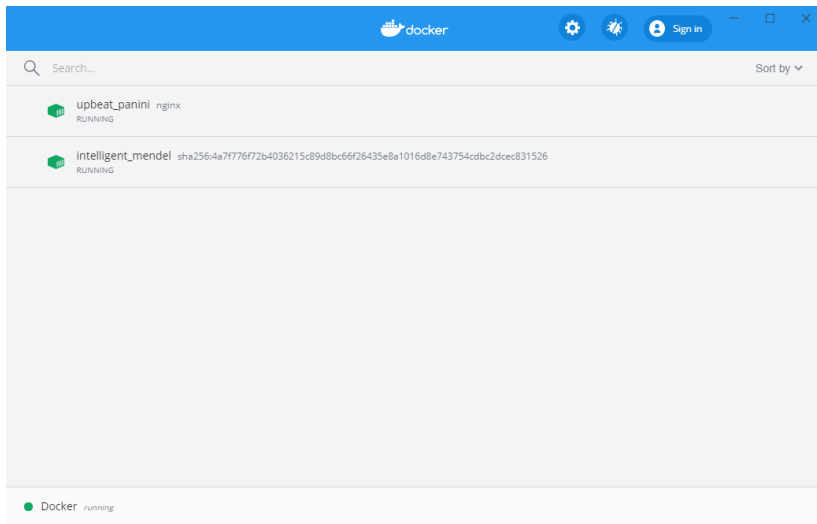
- ▶ Existem três tipos de instalação do Docker:
- ▶ **Direta:** A instalação direta significa que o Docker estará rodando diretamente no Sistema Operacional (SO) do ambiente utilizado. Instalando-se o Docker no Linux e suas distribuições os contêineres estão rodando "*nativamente*" e podem ser acessados diretamente pela linha de comando. Versões do Mac OSX acima de Yosemite 10.10.3 e o Windows 10 **Pro** possui uma tecnologia (*Hyper-V*) que permite a utilização destes contêineres "*nativos*".
- ▶ **Mac/Windows:** Em outras versões do Windows e Mac o Docker utiliza de uma máquina virtual dentro dos SOs com Linux para poder rodar os contêineres.
- ▶ **Cloud:** Existem versões do Docker que são feitas especificamente para sistemas "nas nuvens", como Azure, AWS e Google. Com isto as empresas conseguem utilizar o Docker em Larga Escala.

- ▶ **Docker CE vs Docker EE:** Basicamente ao se adquirir a versão Enterprise do Docker o usuário é incluído no ciclo de vida do Docker e tem suporte para sua aplicação em específico além de produtos extras. Atualmente em 5 US\$ por mês.
- ▶ **Stable vs Edge:** Ao instalar o docker você pode selecionar dois tipos de versionamento. As funcionalidades inseridas na versão *Stable* possuem suporte por maior tempo.



► Docker para Windows 10 Pro

- Nesta versão do Windows existe a ferramenta "*Docker Desktop*" que traz a melhor experiência do Docker no Windows, pois com a tecnologia Hyper-V que é uma virtualização nativa do Windows, sendo mais rápido que uma Máquina Virtual (MV) de fato.
- Nesta versão é possível acessar os comandos do Docker através do Powershell.



- ▶ **Docker para Windows 7, 8, 10 Home**
- ▶ Infelizmente para estas versões do Windows não há tecnologia que permita os contêineres rodem "nativamente". Para estas versões é necessário instalar o Docker Toolbox, juntamente com algum gerenciador de MVs como o Oracle VirtualBox.
- ▶ Nesta versão é possível acessar os comandos do Docker através de SSH com a máquina virtual criada.

- ▶ **Docker para Mac**
- ▶ Assim como no caso do Windows, existe a ferramenta "*Docker Desktop*" com a versão nativa do Docker para OSX com versão superior à Yosemite 10.10.3 e *Docker Toolbox* para as demais.

- ▶ **Docker para Linux**
- ▶ **Não** utilizar versões pré compiladas.
- ▶ **Ubuntu e Distro gratuitos:** O comando abaixo utiliza de um script automatizado do Docker que instala todas as dependências juntamente:
- ▶ *`curl -sSL https://get.docker.com/ | sh`*
- ▶ Todas as Distro Unix pagas requerem a versão EE do Docker.
- ▶ Todos os sistemas Unix com docker requerem a instalação separada dos pacotes "docker-compose" e "docker-machine".

► Docker Compose:

- *sudo curl -L*
"https://github.com/docker/compose/releases/download/1.26.2/docker-
compose-\$(uname -s)-\$(uname -m)-o
/usr/local/bin/docker-compose
- *sudo chmod +x /usr/local/bin/docker-compose*

► Docker Machine:



```
base=https://github.com/docker/machine/releases/download/v0.16.0  
&& curl -L $base/docker-machine-$(uname -s)-$(uname -m)  
>/tmp/docker-machine && sudo mv /tmp/docker-machine  
/usr/local/bin/docker-machine && chmod +x  
/usr/local/bin/docker-machine
```

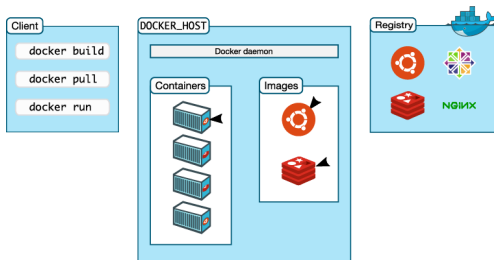
- ▶ Com o Docker instalado, o seguinte comando pode ser executado:
- ▶ *docker version*

```
C:\Program Files (x86)\cmdr
λ docker version
Client: Docker Engine - Community
Version:      19.03.12
API version:  1.40
Go version:   go1.13.10
Git commit:   48a66213fe
Built:        Mon Jun 22 15:43:18 2020
OS/Arch:      windows/amd64
Experimental: false

Server: Docker Engine - Community
Engine:
Version:      19.03.12
API version:  1.40 (minimum version 1.12)
Go version:   go1.13.10
Git commit:   48a66213fe
Built:        Mon Jun 22 15:49:27 2020
OS/Arch:      linux/amd64
Experimental: false
containerd:
Version:      v1.2.13
GitCommit:    7ad184331fa3e55e52b890ea95e65ba581ae3429
runc:
Version:      1.0.0-rc10
GitCommit:    dc9208a3303feef5b3839f4323d9beb36df0a9dd
docker-init:
Version:      0.18.0
GitCommit:    fec3683
C:\Program Files (x86)\cmdr
λ
```



docker



- ▶ Com a estrutura do Docker instalada no ambiente é possível então iniciar diretamente contêineres, como por exemplo um servidor Nginx:
- ▶ *docker container run nginx*
- ▶ Automaticamente ao se instalar o Docker se ganha acesso ao *hub.docker.com* que é um repositório de imagens além das outras funcionalidades. Portanto, ao executar o comando a imagem "nginx" é "puxada" do repositório e executada no Docker

- ▶ Quando o nome do contêiner é somente o seu nome como *nginx* significa que ele é gerido por uma empresa e tem o selo do Docker.
- ▶ Quando o nome do contêiner é *user/container* significa que algum usuário é o dono daquela imagem e ela não possui selo do Docker.


```
λ docker container run nginx
Unable to find image 'nginx:latest' locally
latest: Pulling from library/nginx
bf5952930446: Pull complete
cb9a6de05e5a: Pull complete
9513ea0afb93: Pull complete
b49ea07d2e93: Pull complete
a5e4a503d449: Pull complete
Digest: sha256:b0ad43f7ee5edbc0effbc14645ae7055e21bc1973aee5150745632a24a75266
1
Status: Downloaded newer image for nginx:latest
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to per
form configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-def
ault.sh
10-listen-on-ipv6-by-default.sh: Getting the checksum of /etc/nginx/conf.d/def
ault.conf
10-listen-on-ipv6-by-default.sh: Enabled listen on IPv6 in /etc/nginx/conf.d/d
efault.conf
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates
.sh
/docker-entrypoint.sh: Configuration complete; ready for start up
C:\Program Files (x86)\cmd.exe
λ
```

- ▶ **Estrutura dos comandos**
- ▶ Todos os comandos do Docker seguem esta estrutura (*fora os comandos mais antigos*):
- ▶ *docker <escopo> <comando>*

```
A docker --help

Usage: docker [OPTIONS] COMMAND

A self-sufficient runtime for containers

Options:
  --config string      Location of client config files (default
                        "C:\\Users\\igor6\\.docker")
  -c, --context string  Name of the context to use to connect to the
                        daemon (overrides DOCKER_HOST env var and
                        default context set with "docker context use")
  -D, --debug           Enable debug mode
  -H, --host list       Daemon socket(s) to connect to
  -l, --log-level string Set the logging level
                        ("debug"|"info"|"warn"|"error"|"fatal")
                        (default "info")
  --tls                Use TLS; implied by --tlsverify
  --tlscacert string    Trust certs signed only by this CA (default
                        "C:\\Users\\igor6\\.docker\\ca.pem")
  --tlscert string      Path to TLS certificate file (default
                        "C:\\Users\\igor6\\.docker\\cert.pem")
  --tlskey string       Path to TLS key file (default
                        "C:\\Users\\igor6\\.docker\\key.pem")
  --tlsverify           Use TLS and verify the remote
  -v, --version         Print version information and quit

Management Commands:
  builder      Manage builds
  config        Manage Docker configs
  container     Manage containers
  context       Manage contexts
  image         Manage images
  network       Manage networks
  node          Manage Swarm nodes
  plugin        Manage plugins
  secret        Manage Docker secrets
  service       Manage services
  stack         Manage Docker stacks
  swarm         Manage Swarm
  system        Manage Docker
  trust         Manage trust on Docker images
  volume        Manage volumes
```

```
A docker --help

Usage: docker [OPTIONS] COMMAND

A self-sufficient runtime for containers

Options:
  --config string      Location of client config files (default
                        "C:\\Users\\igor6\\.docker")
  -c, --context string  Name of the context to use to connect to the
                        daemon (overrides DOCKER_HOST env var and
                        default context set with "docker context use")
  -D, --debug           Enable debug mode
  -H, --host list       Daemon socket(s) to connect to
  -l, --log-level string Set the logging level
                        ("debug"|"info"|"warn"|"error"|"fatal")
                        (default "info")
  --tls                Use TLS; implied by --tlsverify
  --tlscacert string    Trust certs signed only by this CA (default
                        "C:\\Users\\igor6\\.docker\\ca.pem")
  --tlscert string      Path to TLS certificate file (default
                        "C:\\Users\\igor6\\.docker\\cert.pem")
  --tlskey string       Path to TLS key file (default
                        "C:\\Users\\igor6\\.docker\\key.pem")
  --tlsverify           Use TLS and verify the remote
  -v, --version         Print version information and quit

Management Commands:
  builder      Manage builds
  config        Manage Docker configs
  container     Manage containers
  context       Manage contexts
  image        Manage images
  network      Manage networks
  node         Manage Swarm nodes
  plugin       Manage plugins
  secret       Manage Docker secrets
  service      Manage services
  stack        Manage Docker stacks
  swarm        Manage Swarm
  system       Manage Docker
  trust        Manage trust on Docker images
  volume       Manage volumes
```

- ▶ Após a execução do comando *docker container run nginx* um contêiner é inicializado com a imagem inicial de um servidor HTTP Nginx. Ao executar este comando o *shell* fica preso à execução, mostrando o log de execução do contêiner até que *Ctrl+C* seja pressionado ou o contêiner pare espontaneamente.
- ▶ Portanto para gerir bem o ambiente Docker e os recursos computacionais é necessário o acesso às imagens e contêineres armazenados, devido ao seu tamanho e custo computacional. O comando abaixo lista todos os contêineres:
- ▶ *docker container ls -a*

```
G:\UDEMY\compose\docker-compose-for-alexandriaPS>docker container ls -a
```

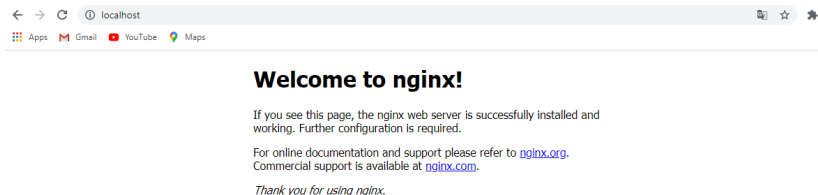
CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
7c8884162656	mrblackpower/ubuntuqt	"/usr/sbin/sshd -D"	6 hours ago	Up 6 hours	22/tcp	ubuntuqt
8bf0eef41475	atlascian/bitbucket-server	"/sbin/tini -- /entr "	8 hours ago	Up 8 hours	0.0.0.0:7998->7998/tcp, 0.0.0.0:7999->7999/tcp	bitbucket
49f18c3edff2	jenkins/jenkins	"/sbin/tini -- /usr/..."	8 hours ago	Up 8 hours	50000/tcp, 0.0.0.0:8081->8080/tcp	jenkins
a6599736bf4d	postgres	"docker-entrypoint.s..."	9 hours ago	Up 8 hours	0.0.0.0:5432->5432/tcp	db
cda7db3d391b	mrblackpower/centosqt	"/bin/sh -c '/usr/sb..."	46 hours ago	Up 11 hours		centosqt

```
G:\UDEMY\compose\docker-compose-for-alexandriaPS>
```

- ▶ Com o comando anterior é possível ver todos os contêineres, inclusive os parados. Para parar a execução de um contêiner utilizamos:
- ▶ *docker container stop <container-id-list>*
- ▶ Analogamente, *docker container start <container-id-list>* para reiniciar-los.
- ▶ E para removê-los:
- ▶ *docker container remove <container-id-list>*

- ▶ Além disso, portas do contêiner podem ser expostas localmente para que possamos acessar os dados através dos protocolos disponíveis.
- ▶ Pode-se iniciar o contêiner com a *flag* -p:
- ▶ *docker container run -d -p 80:80 nginx*

- Desta forma, ao acessar o endereço <http://localhost:80> a seguinte imagem deve aparecer.



- ▶ Nesta seção analisaremos como o sistema de imagens do Docker e o repositório de imagens Docker Hub funcionam.
- ▶ Como dito anteriormente as imagens são os dados de todo o contêiner, contendo o SO bem como os comandos realizados posteriormente.
- ▶ Portanto é possível fazer o download de uma imagem alterar os seus arquivos e subir novamente sua imagem.
- ▶ Para visualizar as imagens já armazenadas no cache do Docker, se utiliza o comando:
- ▶ *docker image ls -a*

```
C:\Program Files (x86)\cmd
A docker image ls -a
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
<none>	<none>	3d9d76675d29	18 hours ago	32.6GB
<none>	<none>	967ad9f2682f	18 hours ago	32.6GB
mrblackpower/ubuntuqt	latest	8cbecccc4719	18 hours ago	32.6GB
<none>	<none>	4de113814e5d	18 hours ago	32.6GB
<none>	<none>	cbfe6c8b0f30	18 hours ago	32.6GB
<none>	<none>	3ec1b5b3414f	18 hours ago	32.6GB
<none>	<none>	36afc52eae26	18 hours ago	32.6GB
<none>	<none>	1ed3d628bd70	18 hours ago	32.6GB
mrblackpower/centosqt	latest	9d994ffe53cf	18 hours ago	298MB
<none>	<none>	55d5d23b2b65	18 hours ago	298MB
<none>	<none>	314c6b549496	18 hours ago	298MB
<none>	<none>	3f99b1aac835	18 hours ago	298MB
<none>	<none>	a89c4136700a	19 hours ago	32.6GB
<none>	<none>	129a39f08eac	19 hours ago	32.6GB
<none>	<none>	11989ceb01d4	19 hours ago	32.6GB
<none>	<none>	cb65f8c25de9	19 hours ago	32.6GB
<none>	<none>	cced22ffc441	28 hours ago	32.6GB
<none>	<none>	5b1236773820	28 hours ago	32.6GB
<none>	<none>	736bce8d522f	28 hours ago	32.6GB
<none>	<none>	753f513d1a64	28 hours ago	32.5GB
<none>	<none>	a35c8c9e39ff	28 hours ago	32.5GB
<none>	<none>	e22ee29b175e	28 hours ago	32.5GB
<none>	<none>	63bcf8596ceb	28 hours ago	32.5GB
<none>	<none>	532a8089e479	28 hours ago	32.5GB
<none>	<none>	eb939d5118b2	28 hours ago	32.5GB
<none>	<none>	6bd7cf667a77	28 hours ago	32.5GB
<none>	<none>	373fbb321cd0	28 hours ago	28.3GB
<none>	<none>	b36b172c3550	31 hours ago	3.26GB
<none>	<none>	d019a27b1a4a	31 hours ago	3.14GB
<none>	<none>	f205e83a9b2b	31 hours ago	3.14GB
<none>	<none>	ec3a9ea84bd0	31 hours ago	3.14GB
<none>	<none>	5c54d0e5f497	31 hours ago	858MB
<none>	<none>	437c28f98594	31 hours ago	858MB
<none>	<none>	c906f8df46b4	31 hours ago	858MB
<none>	<none>	d314dbb3fdfd	31 hours ago	858MB
<none>	<none>	4d9da3accf89	31 hours ago	857MB
<none>	<none>	fab5b1586c44	31 hours ago	857MB
<none>	<none>	fa2fd3291e9c	31 hours ago	817MB
<none>	<none>	b176ee262ccb	31 hours ago	156MB
atlassian/bitbucket-server	6.6.0-ubuntu	8ef30ba80257	2 days ago	873MB
<none>	<none>	1d04c4025d70	3 days ago	298MB
<none>	<none>	8594df351c6	3 days ago	298MB
<none>	<none>	04d2a6436ead	3 days ago	127MB
<none>	<none>	5bf19ecd4d50	3 days ago	127MB
atlassian/bitbucket-server	7.2.1-ubuntu	8a7677be62b6	6 days ago	958MB
atlassian/bitbucket-server	latest	b8867be097fa	6 days ago	971MB
jenkins/jenkins	latest	0edc8be124ad	6 days ago	707MB
ubuntu	16.04	4b22027ede29	12 days ago	127MB
postgres	latest	62473370e7ee	2 weeks ago	314MB
nginx	latest	4hh46517rac3	2 weeks ago	133MB
centos	7	7e6257c9f8d8	3 weeks ago	203MB

```
C:\Program Files (x86)\cmd
```

- É possível ainda atualizar estas imagens atualizadas com o mais novo conteúdo do contêiner através do comando *docker container commit <container> <image:tag>*

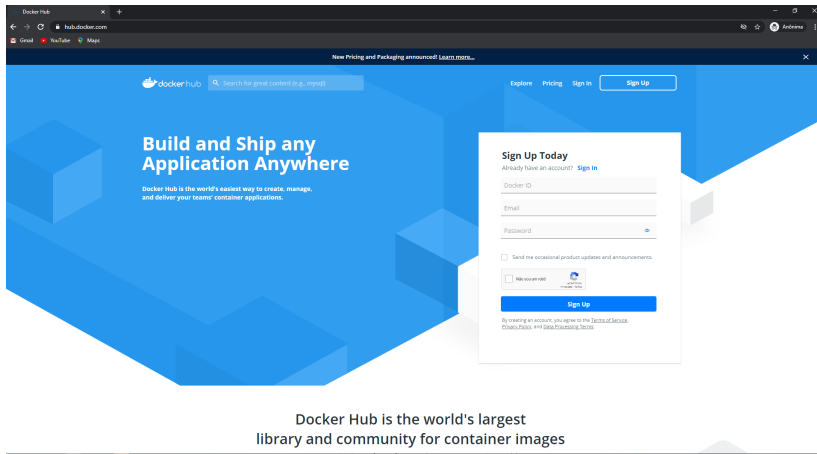
```
C:\Program Files (x86)\cmdr
> docker container commit --help

Usage: docker container commit [OPTIONS] CONTAINER [REPOSITORY[:TAG]]

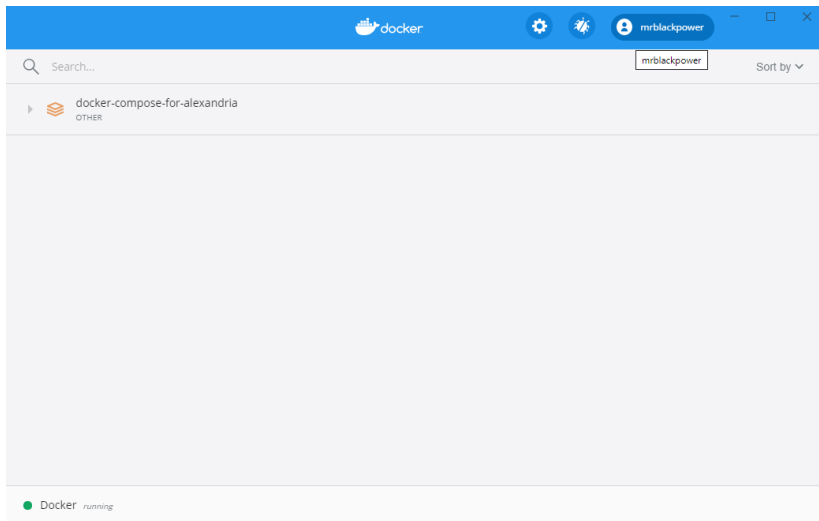
Create a new image from a container's changes

Options:
  -a, --author string      Author (e.g., "John Hannibal Smith
                           <hannibal@a-team.com>")
  -c, --change list        Apply Dockerfile instruction to the created image
  -m, --message string     Commit message
  -p, --pause              Pause container during commit (default true)
```

- ▶ O Docker Hub é um repositório de imagens que funciona em conjunto com o Docker. Assim como um repositório *git* ele é responsável pelo gerenciamento de arquivos e o versionamento, entretanto lida apenas com imagens Docker. Desta forma o repositório permite que rapidamente as imagens sejam salvas nas nuvens e utilizadas remotamente.
- ▶ Para que o repositório de imagens funcione corretamente é necessário criar uma conta no portal hub.docker.com
- ▶ <https://hub.docker.com/signup>



- ▶ Uma vez criada a conta é necessário que se faça o login antes de ter acesso ao repositório pessoal e ser possível "*subir*" imagens.
- ▶ Nas versões gráficas do Docker é possível realizar o login pela própria ferramenta gráfica.
- ▶ Para versões sem estes auxílio é necessário rodar o comando:



- ▶ Com isto é possível enviar a imagem guardada no cache do Docker para o repositório, desta forma a imagem pode ser utilizada em qualquer Docker remoto (*caso seja um repositório público*).
- ▶ *docker push <image:tag>*

```
C:\Program Files (x86)\cmdr
λ docker image push mrblackpower/ubuntuqt
The push refers to repository [docker.io/mrblackpower/ubuntuqt]
ea41f3fa3c4a: Pushed
81d0b0ff181c: Pushed
5f2a89a5851f: Pushed
1cad860f3650: Pushed
aae7d47d9088: Pushed
bbff90c8c8b2: Pushed
b42d19e2fe48: Layer already exists
5f3346391bac: Layer already exists
4345aef52875: Layer already exists
9ba3c8c1b5e5: Layer already exists
429a5a282586: Layer already exists
eeda694586b4: Layer already exists
3193a5bfe62c: Layer already exists
fee46be9d85f: Layer already exists
cdad805d6472: Layer already exists
b8024a34e571: Layer already exists
d25bccaecaef: Layer already exists
17d5f1141f29: Layer already exists
ac1d8fac4f4b: Layer already exists
024cbb7c61d7: Layer already exists
7f8406715c26: Layer already exists
7a22f42dde7: Layer already exists
dcc0cc99372e: Layer already exists
87c128261339: Layer already exists
41a253a417e6: Layer already exists
e06660e80cf4: Layer already exists
```

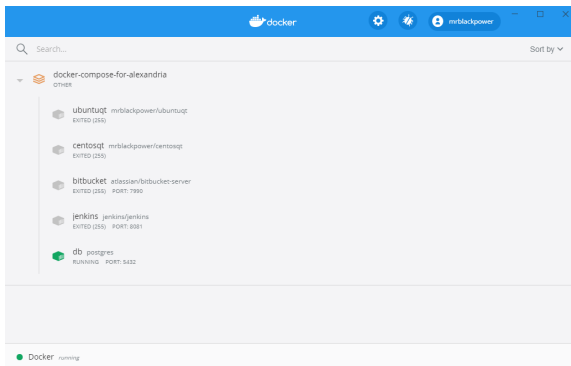
```
C:\Program Files (x86)\cmdr
λ docker image push mrblackpower/ubuntuqt
The push refers to repository [docker.io/mrblackpower/ubuntuqt]
ea41f3fa3c4a: Pushed
81d0b0ff181c: Pushed
5f2a89a5851f: Pushed
1cad860f3650: Pushed
aae7d47d9088: Pushed
bbff90c8c8b2: Pushed
b42d19e2fe48: Layer already exists
5f3346391bac: Layer already exists
4345aef52875: Layer already exists
9ba3c8c1b5e5: Layer already exists
429a5a282586: Layer already exists
eeda694586b4: Layer already exists
3193a5bfe62c: Layer already exists
fee46be9d85f: Layer already exists
cdad805d6472: Layer already exists
b8024a34e571: Layer already exists
d25bccaecaef: Layer already exists
17d5f1141f29: Layer already exists
ac1d8fac4f4b: Layer already exists
024cbb7c61d7: Layer already exists
7f8406715c26: Layer already exists
7a22f42ddee7: Layer already exists
dcc0cc99372e: Layer already exists
87c128261339: Layer already exists
41a253a417e6: Layer already exists
e0660e80cf4: Layer already exists
latest: digest: sha256:92b56fcd997ae2c89e4a60044e756d765f3794ddb9535f3c54010e6419e7273 size: 5757
C:\Program Files (x86)\cmdr
λ
```

- ▶ O docker builder é responsável por construir contêineres através de instruções básicas.
- ▶ O comando *docker builder build* procura no diretório atual um arquivo *Dockerfile* e tenta construir o contêiner.
- ▶ O comando *docker builder prune* remove as camadas "sobrando".

```
1 ARG CENTOS_VERSION=7
2 FROM centos:${CENTOS_VERSION}
3
4 RUN yum -y install openssh-server
5
6 RUN useradd remote_user && \
7     echo "123456" | passwd remote_user --stdin && \
8     mkdir /home/remote_user/.ssh && \
9     chmod 700 /home/remote_user/.ssh
10
11 COPY remote-key.pub /home/remote_user/.ssh/authorized_keys
12
13
14 RUN chown remote_user:remote_user -R /home/remote_user/.ssh && \
15     chmod 600 /home/remote_user/.ssh/authorized_keys
16
17 RUN /usr/sbin/sshd-keygen
18
19 CMD /usr/sbin/sshd -D
```

```
PS>docker-compose builder compose-for-alexandri@190: Missing git support, install posh-git with "Install-Module posh-git" and restart cmd.exe.
PS>docker builder build centos
Sending build context to Docker daemon 3.072KB
Step 1/8 : ARG CENTOS_VERSION=7
Step 2/8 : FROM centos:${CENTOS_VERSION}
----> 76b257c0f5de
Step 3/8 : RUN yum -y install openssh-server
----> Using cache
----> 8594afe3516e
Step 4/8 : RUN useradd remote_user && echo "123456" | passwd remote_user --stdin && mkdir /home/remote_user/.ssh && chmod 700 /home/remote_user/.ssh
----> Using cache
----> 51b0c4e2d5f9
Step 5/8 : COPY remote-key.pub /home/remote_user/.ssh/authorized_keys
----> f8b61746dd58
Step 6/8 : RUN chown remote_user:remote_user -R /home/remote_user/.ssh && chmod 600 /home/remote_user/.ssh/authorized_keys
----> Running in ea59ebac608
Removing intermediate container ea59ebac608
----> 48f362a0373
Step 7/8 : RUN /usr/sbin/sshd-keygen
----> Running in 3ba25718a4b
/usr/sbin/sshd-keygen: line 18: /etc/rc.d/init.d/functions: No such file or directory
Generating SSH2 RSA host key: /usr/sbin/sshd-keygen: line 61: success: command not found
Generating SSH2 ECDSA host key: /usr/sbin/sshd-keygen: line 105: success: command not found
Generating SSH2 ED25519 host key: /usr/sbin/sshd-keygen: line 126: success: command not found
Removing intermediate container 3ba25718a4b
----> eec7620475df
Step 8/8 : CMD /usr/sbin/sshd -D
----> Running in 4887586c9147
Removing intermediate container 4887586c9147
----> 56183a50bfc7
Successfully built 56183a50bfc7
SECURITY WARNING: You are building a Docker image from Windows against a non-Windows Docker host. All files and directories added to build context will have '-rwxr-xr-x' permissions.
   See https://docs.docker.com/build/building/other-platforms/#for-sensitive-files-and-directories
D:\docker\compose\docker-compose-for-alexandri\PS1
```

- ▶ Já o docker-compose é responsável por criar múltiplos contêineres, podendo contruí-los também no processo. É uma forma de arquitetar diversos contêineres de forma conjunta.
- ▶ O interessante é o compartilhamento de volumes e contêineres parametrizados que permitem a rápida criação de um sistema complexo de contêineres.



- ▶ Para utilizar o visualizar a ajuda do compose usa-se:
- ▶ *`docker-compose --help`*
- ▶ Para executar-se a construção dos contêineres é utilizado o comando abaixo somado à um arquivo YAML no mesmo diretório, usualmente *`docker-compose.yml`*.
- ▶ *`docker-compose up -d --build`*

```
1 version: '3'
2 services:
3   jenkins:
4     container_name: jenkins
5     image: jenkins/jenkins
6     ports:
7       - "8081:8080"
8     volumes:
9       - "./jenkins:/var/jenkins_home"
10    networks:
11      - net
12  db:
13    image: postgres
14    restart: always
15    container_name: db
16    environment:
17      - POSTGRES_PASSWORD=drTmJgbUgkHULbnuBqeQ2NrLtxP6bx
18      - POSTGRES_USER=admin
19      - POSTGRES_DB=bitbucket
20    volumes:
21      - "./postgres:/var/lib/postgresql/data"
22    networks:
23      - net
24    ports:
25      - "5432:5432"
26  bitbucket:
27    container_name: bitbucket
28    image: atlassian/bitbucket-server
29    ports:
30      - "7990:7990"
31      - "7999:7999"
32    networks:
33      - net
```

```
G:\UDEMY\compose\docker-compose-for-alexandriaPS>docker-compose up -d --build
Building ubuntuqt
Step 1/40 : ARG UBUNTU_VERSION=16
Step 2/40 : ARG UBUNTU_SUBVERSION=04
Step 3/40 : FROM ubuntu:${UBUNTU_VERSION}.${UBUNTU_SUBVERSION}
----> 4b22027ede29
Step 4/40 : RUN useradd remote_user
----> Using cache
----> 04d2a6436ead
Step 5/40 : RUN echo remote_user:123456 | chpasswd
----> Using cache
----> 5bf19ecdd450
Step 6/40 : RUN apt-get update && apt-get install -y redis-server && apt-get clean
----> Using cache
----> b176ee262ccb
Step 7/40 : RUN apt-get update -q && DEBIAN_FRONTEND=noninteractive apt-get install -q -y
      curl      python      gperf      bison      flex      build-essential      pkg-config      libgl
      1          libx11-dev      libxext-dev      libxfixes-dev      libxi-dev      libx
lean
----> Using cache
----> fa2fd3291e9c
Step 8/40 : RUN apt-get install openssh-server -y
----> Using cache
----> fab5b1586c44
Step 9/40 : RUN mkdir /var/run/ssh
----> Using cache
----> 4d9da3acdf89
Step 10/40 : RUN apt-get update
----> Using cache
----> 437c28f98594
Step 11/40 : ARG QT_VERSION_A=5.12
----> Using cache
----> c906f8df46b4
Step 12/40 : ARG QT_VERSION_B=5.12.2
----> Using cache
```

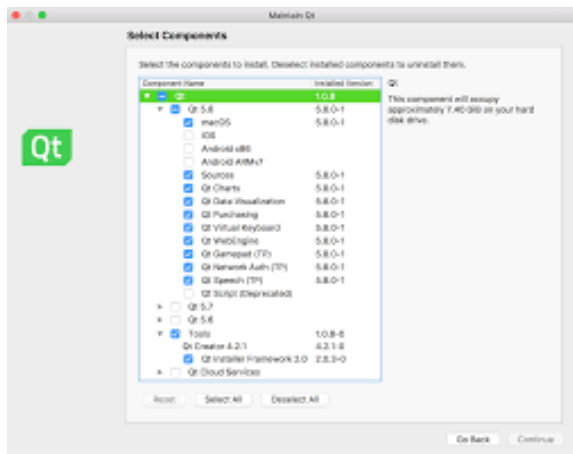
```
Step 36/40 : RUN chown remote_user:remote_user -R /home/remote_user/.ssh && chmod 600 /home/remote_user/.ssh/authorized_keys
----> Using cache
----> 3baf52bae28
Step 37/40 : RUN usermod -s sudo remote_user
----> Using cache
----> 4d5138f4e5d
Step 38/40 : RUN apt-get clean&& rm -rf /var/lib/apt/lists/* /tmp/* /var/tmp/*
----> Using cache
----> 967a9f2622f
Step 39/40 : WORKDIR /
----> Using cache
----> 3db76675c29
Step 40/40 : CMD ["usr/sbin/sshd", "-D"]
----> Using cache
----> 8c8eacc4718
Successfully tagged mblackpower/ubuntuqt:latest
Building centosqt
Step 1/8 : ARG CENTOS_VERSION=7
Step 2/8 : FROM centos:${CENTOS_VERSION}
----> 7a6237c6f6a
Step 3/8 : RUN yum -y install openssh-server
----> Using cache
----> 8204efaf51c6
Step 4/8 : RUN useradd remote_user && echo "123456" | passed remote_user --stdin && mkdir /home/remote_user/.ssh && chmod 700 /home/remote_user/.ssh
----> Using cache
----> 3db4c821c78
Step 5/8 : COPY remote-key.pub /home/remote_user/.ssh/authorized_keys
----> Using cache
----> 3f99b1aac95
Step 6/8 : RUN chown remote_user:remote_user -R /home/remote_user/.ssh && chmod 600 /home/remote_user/.ssh/authorized_keys
----> Using cache
----> 55d5d23b265
Step 7/8 : RUN /usr/sbin/sshd-keygen
----> Using cache
----> 314c0854046
Step 8/8 : CMD /usr/sbin/sshd -D
----> Using cache
----> 9d904ffa53cf
Successfully built sd904ffa53cf
Successfully tagged mblackpower/centosqt:latest
Starting bitbucket ...
Starting bitbucket ... done
Starting jenkins ... done
Starting centosqt ... done
Starting ubuntuqt ... done
$ docker-compose up docker-compose-for-alexandrip5
```



- ▶ O Qt é uma IDE (Integrated Development Environment) que possibilita a execução de programas C++ em diversos sistemas, inclusive em sistemas embarcados. O Qt era de propriedade da Nokia mas foi vendido para outra empresa (Digia) que transformou o Qt no que ele é hoje.
- ▶ Através do Qt é possível criar aplicações gráficas para diversos sistemas operacionais e diferentes sistemas embarcados. Sendo extremamente para unificação do código de projetos que utilizam de tais ambiente em um único contexto.

- ▶ Dentre os programas modernos que utilizam o Qt, estão:
- ▶ Age of Wonders III (Jogo)
- ▶ Battle.net
- ▶ GNU Octave
- ▶ GMount & Blade: Warband (Jogo)
- ▶ Spotify
- ▶ Oracle Virtualbox
- ▶ Wolfram Mathematica

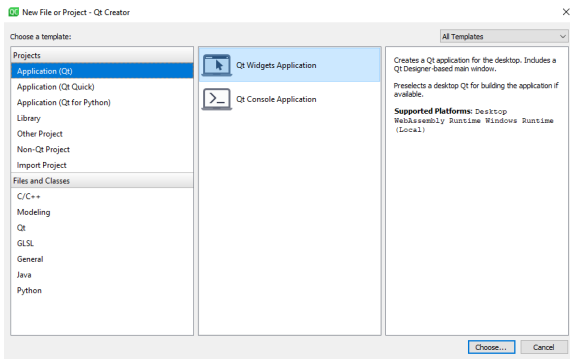
- ▶ É possível se instalar as bibliotecas do Qt compilando-as, entretanto seria necessário linkar o resultado compilado manualmente ao compilador, o que é no mínimo trabalhoso.
- ▶ É recomendado que se instale o Qt utilizando o instalador do Qt (em qualquer SO). Um fato interessante é que o próprio instalador do Qt é desenvolvido em cima das bibliotecas do Qt.
- ▶ O instalador ainda pode ser utilizado sem a interface gráfica, o que facilita sua utilização dentro de contêineres do Docker.



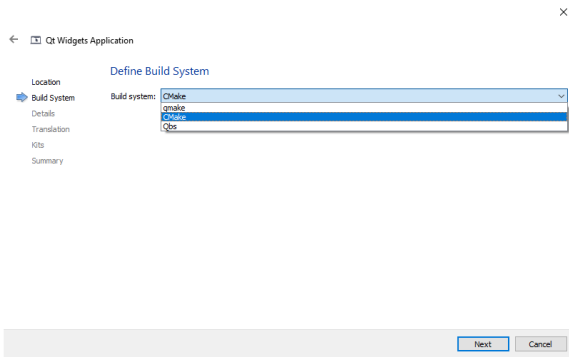
- ▶ Uma vez instaladas corretamente, as bibliotecas serão acessíveis e o software de programação QtCreator é instalado juntamente.
- ▶ O QtCreator é atualmente uma das grandes ferramentas utilizadas no desenvolvimento de aplicações em C++ e funciona também para projetos que não incluem as bibliotecas do Qt.
- ▶ Outro comando instalado juntamente é o *qmake*, este comando funciona de modo similar ao *cmake*, configurando os arquivos para serem compilados e linka a interface gráfica.
- ▶ A grande vantagem destas bibliotecas está no fato de que podem ser instaladas em quase qualquer ambiente computacional e funciona utilizando os mesmos passos. Uma vez executado o *qmake* é possível compilar todo o projeto com qualquer compilador C++. Para projetos com interface gráfica isto é uma grande vantagem pois cada SO lida com protocolos diferentes para exibição de imagens ou mesmo os componentes da interface gráfica e o Qt unifica todos estes comandos.



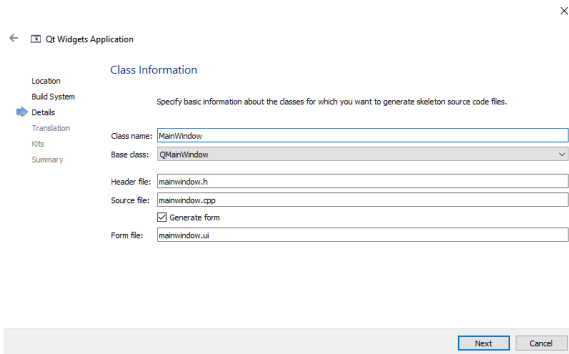
- ▶ Através do próprio QtCreator é possível se criar projetos com interface de usuário gráficas utilizando os chamados QtWidgets.



- ▶ É possível ainda selecionar a "máquina" construtora (qmake ou cmake).



- ▶ É necessário inserir o nome da janela principal e o seu tipo. O tipo QMainWindow representa uma janela principal, passível de ser utilizada na maioria das aplicações. Sendo composta por três arquivos *.h* *.cpp* *.ui*, sendo o último responsável por conter os elementos da interface gráfica.



Qt Widgets Application

Class Information

Specify basic information about the classes for which you want to generate skeleton source code files.

Location
Build System
Details
Translation
Kits
Summary

Class name:

Base class:

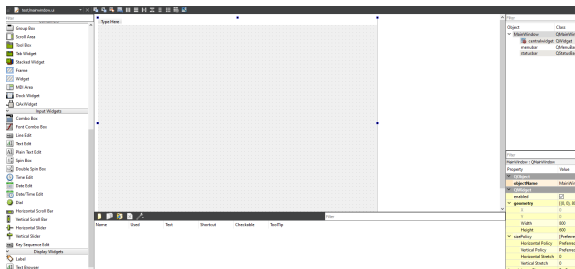
Header file:

Source file:

☒ Generate form

Form file:

Next Cancel

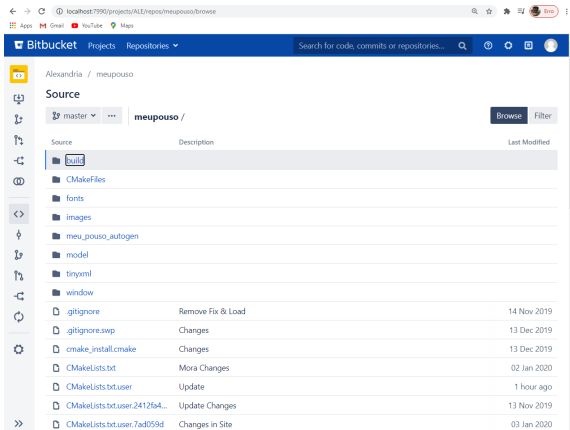


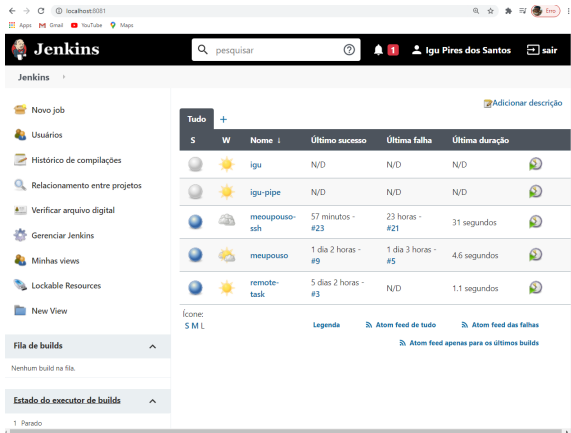
- ▶ É possível ver a estrutura principal do programa acessando o arquivo main.cpp que continua sendo o arquivo principal do projeto. Aqui vemos Um objeto que representa a janela principal sendo criado e exibido, enquanto uma aplicação recebe os argumentos de entrada e é enviado no retorno.
- ▶ Na realidade esta aplicação representa o loop do programa, que eventualmente passa pela janela por ser um loop especial, mas não fica travado nele.

```
1  #include "mainwindow.h"
2
3  #include <QApplication>
4
5  int main(int argc, char *argv[])
6  {
7      QApplication a(argc, argv);
8      MainWindow w;
9      w.show();
10     return a.exec();
11 }
12
```


- ▶ Ao executar a ferramenta de configuração (cmake ou qmake) as bibliotecas do Qt são acionadas e os três arquivos (.cpp .h e .ui) são linkados
- ▶ Na realidade, por debaixo dos panos o Qt cria outras classes de interface baseadas no arquivo .ui e as compila. O Qt é responsável por essa tradução e por este motivo que ao modificar a interface é possível que erros C++ apareçam na compilação.

- ▶ O Qt pode ser compilado para diferentes plataformas e possui diferentes versões com diferentes funcionalidades.
- ▶ Portanto, o Docker fornece um ambiente estável para que o programa seja compilado e lançado(deployed).
- ▶ Ainda dentro do Docker é possível manter um repositório local e um automatizador.
- ▶ Para estas funções foram escolhidos, respectivamente o Bitbucket Server e o Jenkins.

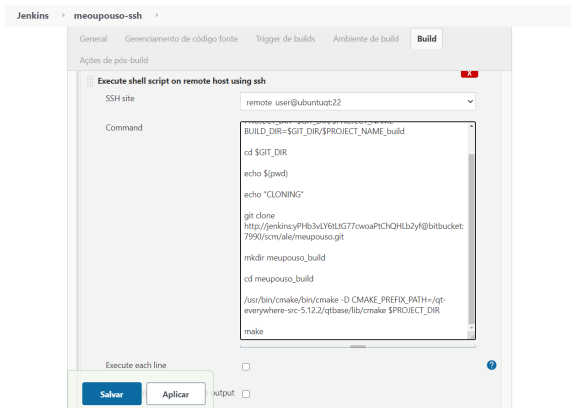


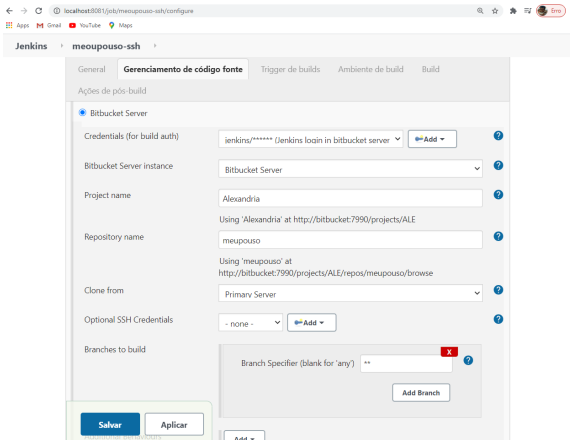


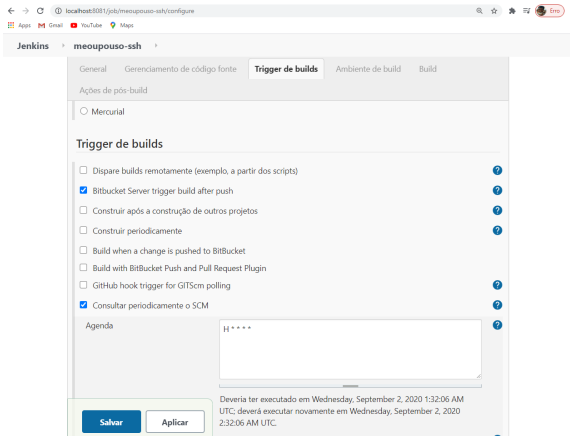
The screenshot shows the Jenkins web interface. The top navigation bar includes the Jenkins logo, a search bar, and user information for 'Igu Pires dos Santos'. The left sidebar contains links to various Jenkins features like 'Novo job', 'Usuários', 'Histórico de compilações', etc. The main content area displays a table of jobs with columns for status, name, last success, last failure, and last duration.

S	W	Nome	Último sucesso	Última falha	Última duração
		igu	N/D	N/D	N/D
		igu-pipe	N/D	N/D	N/D
		meupouso-ssh	57 minutos - #23	23 horas - #21	31 segundos
		meupouso	1 dia 2 horas - #9	1 dia 3 horas - #5	4.6 segundos
		remote-task	5 dias 2 horas - #3	N/D	1.1 segundos

Below the table, there are links for 'Legenda', 'Atom feed de tudo', and 'Atom feed das falhas'. The bottom status bar shows '1 Parado'.









- ▶ Sumarizando, são 3 contêineres principais, um com o SO de lançamento (deploy), um com o repositório (Bitbucket) e um com o automatizador (Jenkins)
- ▶ (Demonstrar no ambiente)

Obrigado!