Terminal Unix

- Comandos e produtividade no terminal
- Contentores vs Máquinas Virtuais
- Docker vs Podman vs Kubernetes
- Comandos básicos de Docker/Podman

1

Aumentar produtividade

- **wget**: permite o download de dados
- Ficheiro ~/.bashrc contém as configurações básicas
- **alias**: permite a definição de novos momes para os comandos

```
# download data in the command line
$ wget https://nextcloud.bio.di.uminho.pt/s/2BHzz5EtbiRJsiM/download -O datasets.zip

# hidden configuration file of the bash ~/.bashrc

# aliases allow to create shortcuts for different commands
# edit .bashrc and add some alias
alias la='ls -la'

# ask before removing or overwriting files
alias mv='mv -i'
alias dirsize='du -sh */'
```

2

Atalhos

- *Symbolic links* ou atalhos permitem o acesso a ficheiros e directorias de forma rápida sem replicação.
- Para criar um atalho: mover para a diretoria onde se quer o atalho e executar o comando:

```
# Get the full path of the folder to be linked
$ pwd
# move (cd) to folder where you want the link stored
$ ln -s fullPathToFileOrFolderToLink linkName
```

- Atalhos são listados como ficheiro/diretorias com o comando ls
- Para remover um atalho usar o comando **unlink**

Comandos

- Unix tem um conjunto de comandos que permitem a manipulação de grandes ficheiros de texto de forma eficiente. Estas ferramentas são muito úteis em estudos de bioinformática.
- Ferramentas:
  - Contagens : **wc**
  - Extração e filtragem de informação: **head, tail, grep, uniq, awk, cut**
  - Manipulação: **dos2unix, sort, tr, sed**
  - Comparação: **diff**

Comandos *wc* e *tr*

- **wc**: permite a contagem de linhas, palavras e caracteres
- Opções:
  - **c**: number of characters
  - **w**: number of words
  - **l**: number lines
- **tr**: susbtituição de caracteres
  - Sintaxe: tr 'find' 'replace'

```
# number of lines in a file
$ wc -l genes.gtf
# number of lines that match TP53
$ grep TP53 genes.gtf | wc -l
```

```
#echo: shows the text in terminal window

# convert lowercase to uppercase
$ echo "toupper" | tr '[a-z]' '[A-Z]'

# tr can also be used to delete characters: tr -d 'del'
$ tr -d 'chr' < input.txt > out.txt
```


Comando *cut*

- **cut**: remove ou seleciona uma secção de texto de cada linha de um ficheiro.
- Sintaxe:
  - cut [-d delim] -f <fields> [file(s)]
- Opções:
  - delim: delimitador
  - fields: index do campo que será extraído

```
manager@bl8vbox[Documents] head gene.regions.bed

chr8    134249414  134309547  NDRG1    gene    -
chr4    1795039    1810598    FGFR3    gene    +
chr19   56459198   56499995   NLRP8    gene    +
chr1    224804179  224928249  CNIH3    gene    +
chr1    171283486  171311223  FMO4     gene    +
```

```
# extract gene names and strand
$ cut -d \t -f4,6 gene.regions.bed

# get the coordinates of genes
$ cut -f 1-3 gene.regions.bed

# extract the first three characters of each line
$ cut -c 1-3 gene.regions.bed
```

Comandos

- **uniq**: filtra linhas repetidas num ficheiro
- Sintaxe:
  - uniq [options] [files(s)]
  - Opção: **-c** conta o numero de ocorrências de cada linha

```
# how many different chromossomes
cut -f1 gene.regions.bed | sort | uniq | wc -l

# count number of repeated occurrences per item
cut -f1 gene.regions.bed | sort | uniq -c
```


Permissões e controlo de ficheiros e pastas

```
-rwxrw-r-- 1 root root 2048 Jan 13 07:11 afile.exe
```

- file permissions
- number of links,
- owner name,
- owner group,
- file size,
- time of last modification, and
- file/directory name

- Por vezes há a necessidade de manipular a permissão que um utilizador detém sobre um ficheiro ou pasta. O Unix disponibiliza as seguintes ferramentas:
  - Alteração de permissões de leitura e escrita: **chmod**
  - Alteração de Autor e Grupo: **chown**
- Exemplos:
  - Alterar Autor e Grupo de "root" para "aluno" de uma pasta e subpastas:
    - **chown -R aluno : aluno /home/aluno/pasta**
  - Alterar as permissões da pasta para somente em modo de leitura:
    - **chmod -R 444 /home/aluno/pasta**

https://pplware.sapo.pt/linux/linux-permissoes-em-ficheiros-e-directorios/

https://www.howtoforge.com/linux-chown-command/

diff

- **diff**: compara ficheiros linha a linha
  - Sintaxe: diff [options] [files(s)]
- **dos2unix**: converte ficheiro de texto DOS/MAC para format Unix.
  - Sintaxe: dos2unix [options] [-c convmode] [-n infile outfile]
  - Comando não é nativo. Necessário instalar o package: ***sudo apt-get install tofrodos***

```
sudo ln -s /usr/bin/fromdos /usr/bin/dos2unix
sudo ln -s /usr/bin/todos /usr/bin/unix2dos
```

O que fazem estes commandos ??

Expressões regulares

- Expressões usadas para identificar um padrão de sequência de caracteres de forma flexivel.
- Alternância:
  - **a|b** : caractere a ou b
- Padrões:
  - **.** : qualquer caracter
  - **[a-z]** : qualquer caracter de 'a' a 'z'. **[0-9]** : qualquer algarismo.
  - **\s** : espaço em branco, tabulação ou mudança de linha
  - **\w**: letras, digitos e _.
  - **\d**: digitos de 0 a 9.
- Quantificadores:
  - **?** : zero ou uma ocorrência. Ex: ac?ção match com "acção" e "ação".
  - ***** : zero ou mais ocorrências. Ex: ab*c match com "ac", "abc", "abbc", "abbbc"…
  - **+** :uma ou mais ocorrências. Ex:, ab+c match com "abc", "abbc", "abbbc",.., mas não "ac".
  - **{n}** : ocorrência de exactamente n vezes.

Expressões regulares

- Agrupamento:
  - **(….)** usado para definir a área onde um operador é aplicado. Ex: **con(s|c)elho** faz match apenas com as palavras *conselho* e *concelho.*
- Posição:
  - **^** : início da linha
  - **$** : final da linha
- Algums exemplos:
  - **^\s*casa** : match com linhas que comecem com a palavra *casa* podendo conter ou não espaços antes da palavra.
  - **\d{4}-\d{3}** : match com o código postal (formato português)
  - **\w+@[a-z]+\.((com)|(org))** : match com endereços de email no formato *nome@servidor.com* ou *nome@servidor.org*
- Para "brincar": https://regex101.com/

sed

- **sed** (stream editor). Lê a informação do canal de input (ficheiro ou stdin), modifica-a com base nos comandos especificados e imprime o resultado no stdout.
- Ferramenta ponderosa para seleção e substituição de texto.
- Sintaxe:
  - sed [options] [script] [inputfile]
- Opções:
  - **-e**: executa os comandos enquanto o input está a ser processado.
  - **-f**: usa um ficheiro (script) onde uma lista de comandos está definida
  - **-i**: as alterações são efectuadas no ficheiro de input e não redirecionadas para o output.

sed

- Sed usado na substituição de texto:

sed –e 's/regExpToMatch/replaceText/' fileName

```
manager@bl8vbox[Documents] head gene.regions.bed

chr8    134249414  134309547  NDRG1   gene   -
chr4    1795039    1810598    FGFR3   gene   +
chr19   56459198   56499995   NLRP8   gene   +
chr1    224804179  224928249  CNIH3   gene   +
chr1    171283486  171311223  FMO4    gene   +
```

```
# convert to upper chromossomes
$ sed -e 's/chr/CHR/' gene.regions.bed

# Note that sed will only replace the first match. option g is
used to do a global match

# remove the chr prefix
$ sed 's/chr//g' gene.regions.bed

# forcing matches at the start of the line

$ sed 's/^chr//' gene.regions.bed

# Trim whitespaces and tabulations at start and end of file
$ sed 's/^[ \t]*//;s/[ \t]*$//' file.txt
```


sed

- Exemplo de seleção de linhas:

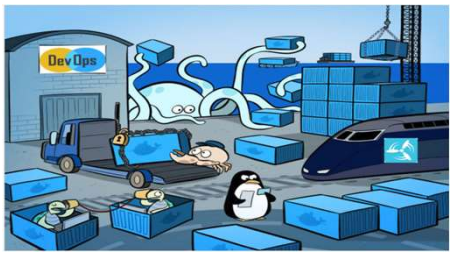
```
# delete blank line
$ sed '/^$/d' file.txt

# delete the fourth line
$ sed '4d' file.txt

# delete the first to the third line
$ sed '1,3d' file.txt

# return only the 4° line of the file
$ sed '4!d' file.txt

# return all lines between the 3 and 5 lines
$ sed '3,5!d' file.txt
$ sed –n '3,5p' file.txt

# return lines 3 and 5
$ sed –n –e '3p' –e '5p' file.txt
```

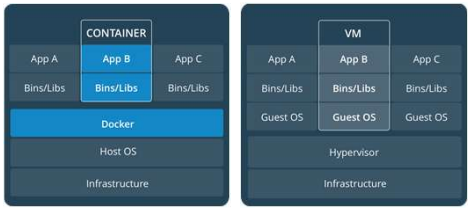
-n : suppress the default output
-e: sets the following command to be run

p: print
d: delete
!: not

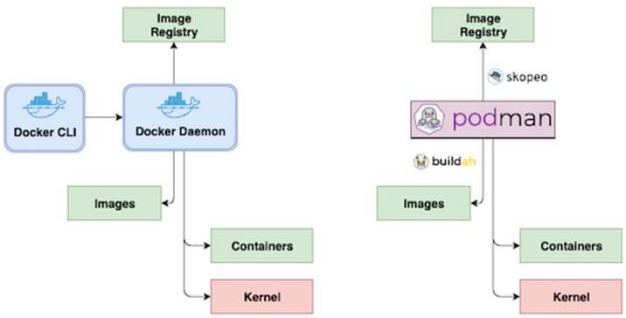
Contentores vs Máquinas virtuais

- O uso crescente de contentores (Docker) na comunidade de bioinformática deve-se principalmente à possibilidade de **usar aplicações num formato portátil** com as **mesmas configurações e especificações** utilizadas durante o desenvolvimento da aplicação.
- Facilita o uso de recursos computacionais com maior poder computacional devido a **facilidade de instalação** e **escalabilidade garantida por via de replicação**.
- São **mais leves** que máquinas virtuais e **não estão restritas ao uso de Hardware** devido à **inexistência de drivers** do sistema operativo integrado.
- https://docs.docker.com/get-started/





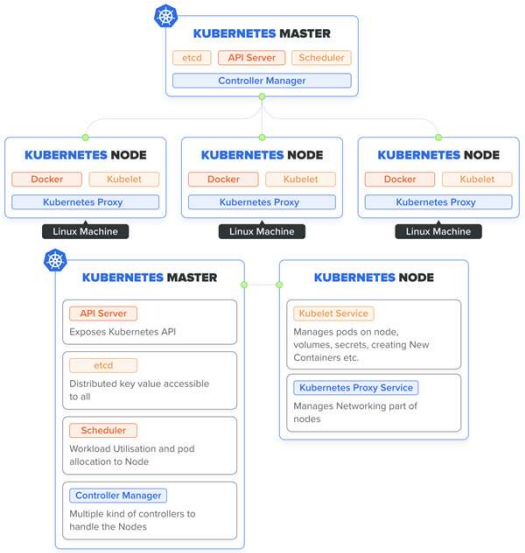
Docker vs Podman



- Apesar de o **Docker** ser o gestor de contentores mais popular, grande parte dos servidores têm adotado a utilização do **Podman** ou o **Kubernetes** como gestor nativo. A razão deve-se ao facto de o Docker recorrer a um **serviço daemon em vez de interagir diretamente com a kernel do SO**, criando assim uma camada de aplicação que pode ser vulnerável a ataques.

Docker/Podman vs Kubernetes

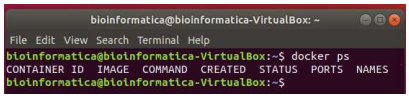
- O Kubernetes, por outro lado, **requer diversas máquinas**, uma vez que foi desenvolvido para ser aplicado **em sistemas distribuídos de larga escala** (recorrendo a serviços como o **AWS, Azure** e o **Google cloud**) de forma a possibilitar a replicação e distribuição de aplicações de uma forma eficiente e com alta escalabilidade.
- Os gestores de contentores Docker e Podman detêm as mesmas funcionalidades entre si, ao ponto em que é possível utilizar o Podman como se fosse o Docker recorrendo ao comando: **alias docker=podman**
- Além disso, o **Podman** possibilita a **migração de contentores, configurações de serviços e pods** do Docker para Kubernetes.
- Devido às razões mencionadas acima, os servidores do grupo BIOSYSTEMS utilizam o **Podman** como gestor de contentores nativo.

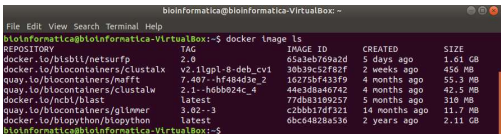


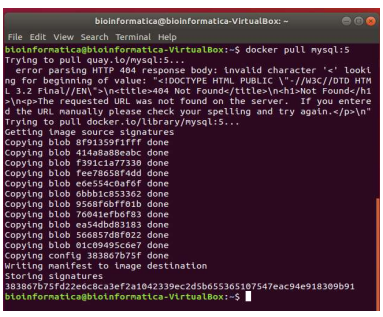
Comandos básicos para gerir imagens de contentores

- Identificar contentores a correr:
  - docker ps
- Identificar imagens de docker instaladas:
  - docker image ls
- Obter novas imagens com aplicações diferentes:
  - docker pull nome_da_imagem:versão/TAG

  Exemplo: docker pull mysql:5







Comandos básicos para criar imagens de contentores

- Criar imagens de contentores requer um script *Dockerfile* e todos os ficheiros auxiliares para serem guardados dentro da imagem:
- Estando numa diretoria que contenha todos os ficheiros necessários é possível gerar uma imagem com o comando:
  - **docker build --tag=nome_da_imagem .**

```
# Example of DockerFile content
# Use an official Python runtime as a parent image
FROM python:2.7-slim
# Set the working directory to /app
WORKDIR /app
# Copy the current directory contents into the container at /app
COPY . /app
# Install any needed packages specified in requirements.txt
RUN pip install --trusted-host pypi.python.org -r requirements.txt
# Make port 80 available to the world outside this container
EXPOSE 80
# Define environment variable ENV NAME World
# Run app.py when the container launches
CMD ["python", "app.py"]
```

(**o ponto no final do comando** indica que todos os ficheiros para a montagem da imagem estão dentro da diretoria onde o terminal esta a utilizar -- para saber quais, basta escrever o comando '*ls*' no terminal)

- https://docs.docker.com/get-started/part2/

Comandos básicos para executar programas em contentores

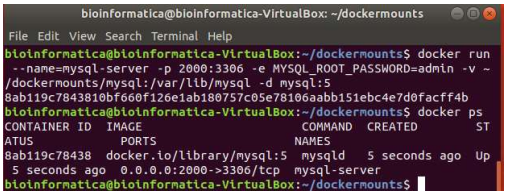
- Executar uma aplicação dentro do contentor:
  - docker run nome_da_imagem:versão comando [parâmetros opcionais]

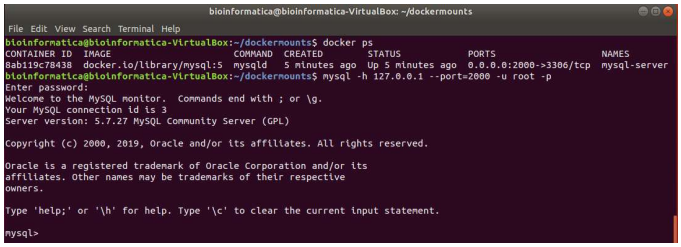
  Exemplo: docker run ncbi/blast:latest blastn –help
- Comandos opcionais para adicionar funcionalidades à execução:
  - Associar uma porta de rede do Hardware ao contentor:
    - docker run … –p porta_do_hardware:porta_do_container …
  - Associar uma diretoria do Hardware ao contentor (volumes):
    - docker run … -v diretoria_do_hardware:diretoria_dentro_container …
  - Definir variáveis de ambiente Linux ao contentor:
    - docker run … -e nome_da_variavel=definicao_da_variável …
  - Atribuir um nome ao contentor:
    - docker run … --name=nome_do_container …

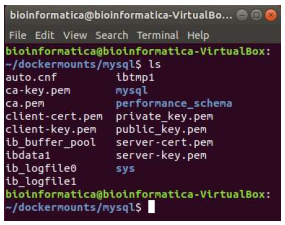
Comandos básicos para executar serviços a partir de contentores

- Exemplo utilizando mysql em modo serviço (execução *detached*):
- docker run --name=mysql-server -p 2000:3306 -e MYSQL_ROOT_PASSWORD=admin -v ~/dockermounts/mysql:/var/lib/mysql -d mysql:5

Execução em background/serviço







Comandos básicos para executar serviços a partir de contentores

- Entrar nos contentores em modo de serviço:
  - docker exec –it nome_do_container_ou_id sh ou bash

  Exemplo: docker exec –it mysql-server bash
- Parar contentores em modo de serviço:
  - docker stop nome_do_container_ou_id

  Exemplo: docker stop mysql-server
- Apagar contentores:
  - docker rm nome_do_container_ou_id

  Exemplo: docker rm mysql-server