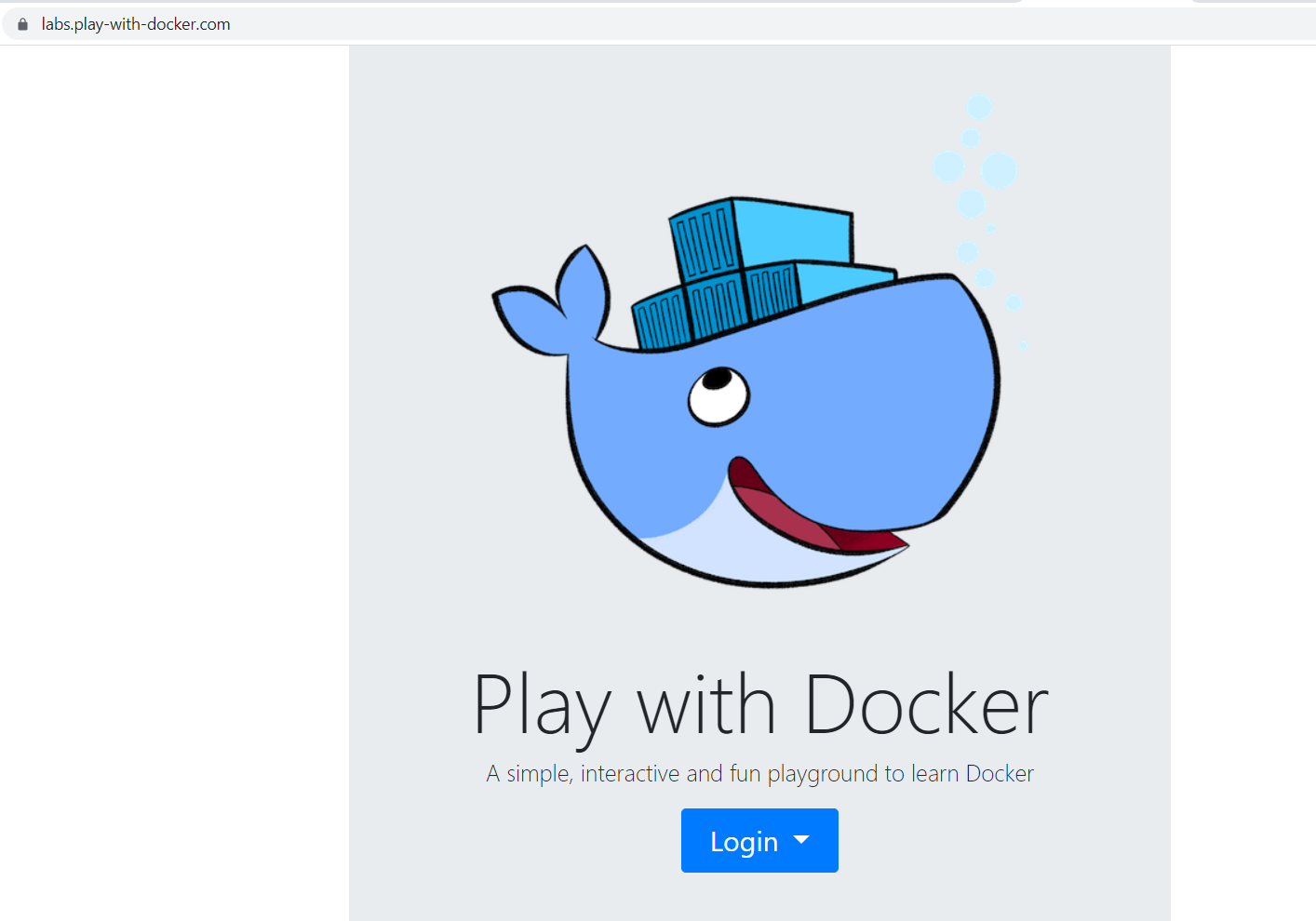
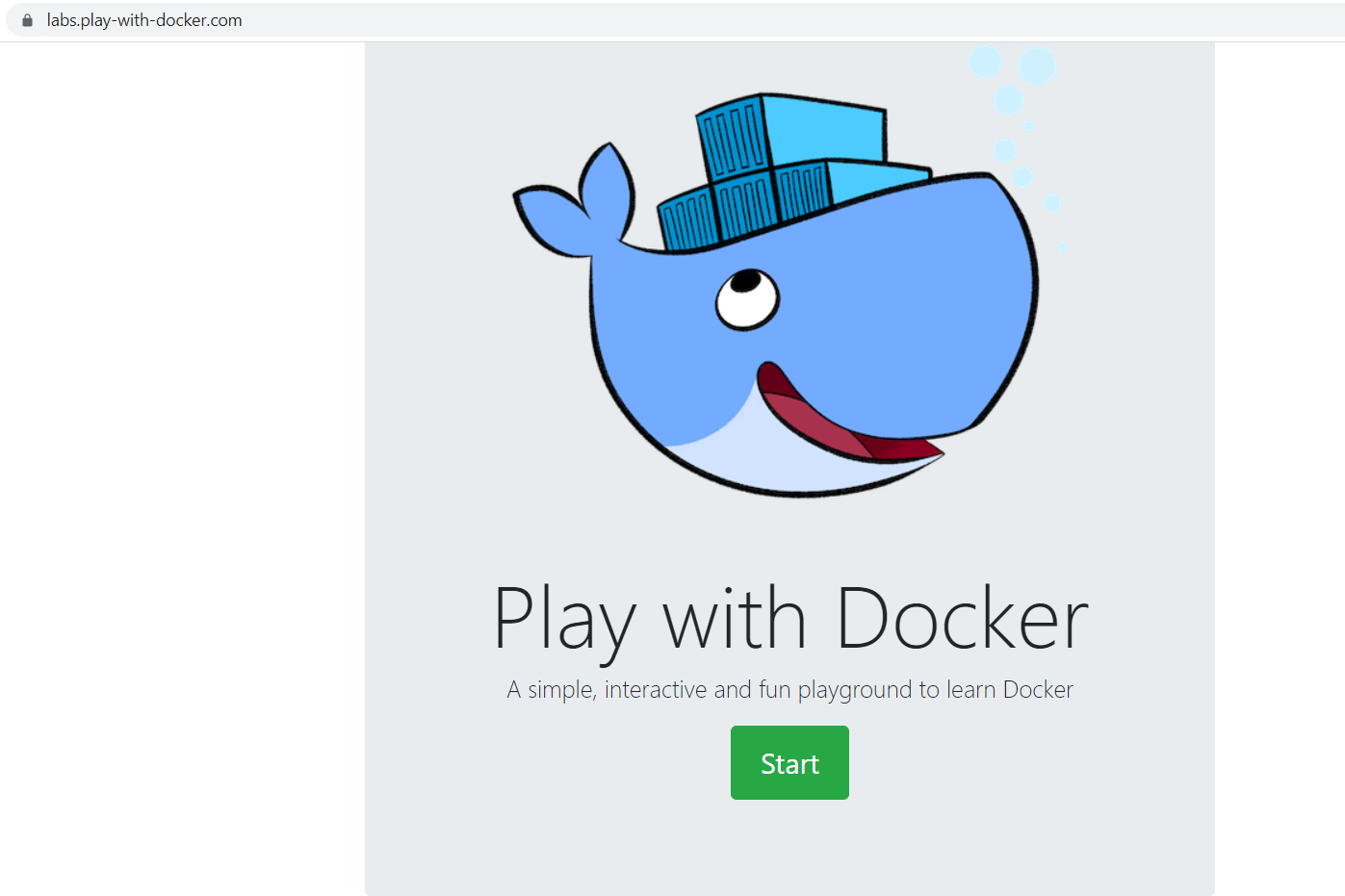
# Lab: Linux Console Commands

Lab for the ["Containers and Clouds"](https://softuni.bg/trainings/4359/containers-and-cloud-january-2024) course @ SoftUni

## Configure Connectivity of the VM

The first step is to open the link – <https://labs.play-with-docker.com/>. 

Then press the **[Login]** button and click on "**docker**". A new dialog box opens, which is for **docker registration**. If you don't have a registration click on **[Sign Up]**. You have to create an account with a username, password and email. Then sign in to your account. It takes you to a page to select the plan you want – click on "**Continue to Free**" (Personal plan). Log in to the email you registered with and **confirm your account**. You may need to reload the page until you see this:



Click on the **[Start]** button.

Картина, която съдържа текст

Описанието е генерирано автоматично

You should see this. Click on **[+ add new instance]**.

Картина, която съдържа текст

Описанието е генерирано автоматично

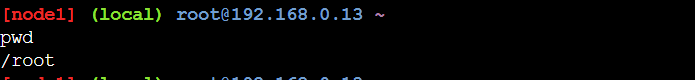
We are ready for work!

## Getting to Know the Console

**Let's start writing the command in the docker playground.**

On the prompt, we can **input the user and password** that we created during the installation. Of course, we can use the **root** **user**, but it is not considered a good practice.

Now, let’s check where (in which **folder**) **we are** with the pwd command:



It appears that we are in our **home folder** (/home/lsauser).

Check what we have here:



There is **nothing** or at least it appears to be this way. Of course, what we will see here on a **clean new installation** depends on what **distribution we chose** and **what settings** there are by default.

Even though we are not familiar with all options on the ls command yet, let's check if there are **any hidden files and** **folders** with:

Картина, която съдържа текст

Описанието е генерирано автоматично

Note that you can **autocomplete commands** by clicking the [Tab] **button** **twice** (Tab-Tab).

So, **there are some files** after all. Those are considered **hidden files**, because their name starts with the dot **symbol** (.). The same rule applies to **folders** as well.

Now, let's execute this:

Картина, която съдържа текст, телевизия, екран, тъмен

Описанието е генерирано автоматично

We can also **give arguments**, not only options. For example, we can check what we have in the **main** (**root**) **folder**. The "/"symbol is used to state that we want to **access the root of our file system**:

Картина, която съдържа текст

Описанието е генерирано автоматично

Картина, която съдържа текст

Описанието е генерирано автоматично

We can note that there is a **folder** named "**/root**". This is the **home folder for the root user**.

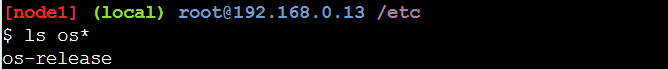
Let's **change the folder**. For example, go to the "/etc" **folder**. This is the place where most of the **configuration files** **are stored**. Then we can check if indeed we **changed the folder**:

Картина, която съдържа текст

Описанието е генерирано автоматично

As we can see, there is **no need** to execute pwd. The **prompt reflects or shows** where in the **file system tree** we are currently.

When we want to **address all files** **which name starts with something**, no matter what, and how many symbols their name contains, we can use the "\*" symbol. For example, ask for **all files** starting with "os\*":



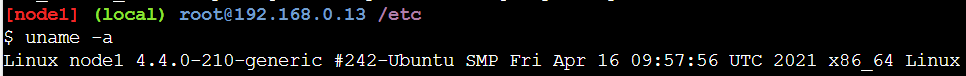
Okay, now that we know that this **file exists**, let's check **what it contains** (the actual output may be different):

Картина, която съдържа текст

Описанието е генерирано автоматично

It appears, that this **file** contains **detailed information** **about our** **distribution**.

**Similar or additional information** about the distribution we can get by executing:



Beside the information about the kernel, we can extract information about the **name of the host**:

Картина, която съдържа текст

Описанието е генерирано автоматично

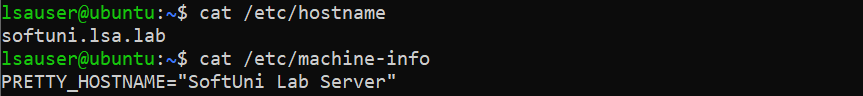
Now, let's **return to our home folder**:

Картина, която съдържа текст

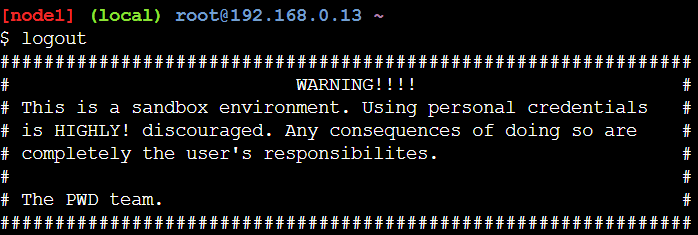
Описанието е генерирано автоматично

As we can see, if we execute the cd command **without any arguments**, the result is that we are "**back home**". There is also a **special symbol** that we can use – it is again the tilde symbol – "~".

Now, let's try a different approach. In general, no matter what distribution we use, there is a **common set of** **commands** that is always available.

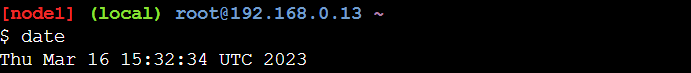


In order the **changes to be reflected in the prompt**, we must **close the session**, and **open a new one**. So, type logout and **log back in**:

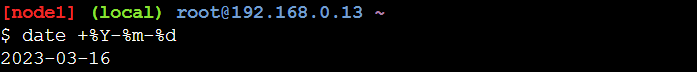


We can see that the **new name is applied**.

Now, we can check **what** **date is today** and **what is the time** **now**:

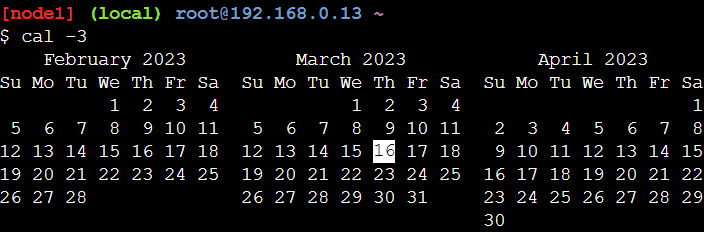


There is a way to **modify the output** of the date command:

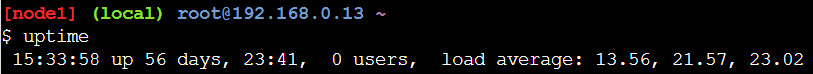


This way, we will receive the **current date**, represented in "YYYY-MM-DD" format.

Should we need a **calendar** on the command line, we can have it easily with:



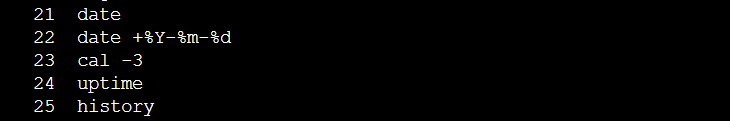
If we need to know **since when or how long our system is operating**, we can do it with:



At any point we can ask for the **history of executed commands**:

Картина, която съдържа текст

Описанието е генерирано автоматично



If we want to **end our session**, we can do it with either exit or logout. Issuing any of these will close our session but will leave the machine up and running. Let's type exit and press the [Enter] key. Our **session is closed** now:

Картина, която съдържа текст

Описанието е генерирано автоматично

Now **log in back again** and **ask for all files** (including hidden ones) in our **home folder**:

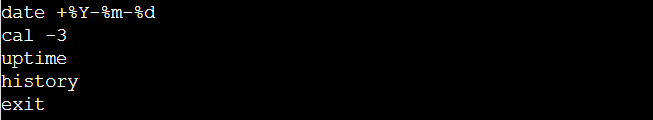
Картина, която съдържа текст

Описанието е генерирано автоматично

It appears that there is a **special file** that takes care for our **history** – .bash\_history. Let's check its **contents**:

Картина, която съдържа текст

Описанието е генерирано автоматично



As we can see the **last few commands are not here** and there is a perfect explanation for this. The reason is that they are **kept in a buffer** and are stored on the disk only when certain events occur, like session end.

Now let's continue with the console command explanations and transfer files.

## Getting to Know Environment Variables

Let's start with the **environment exploration process**. Try out the set command and look at its result:



↓

Картина, която съдържа текст

Описанието е генерирано автоматично

When executed **without parameters**, set returns **information about the environment** – **variables** and **functions**. Again, the information will differ from distribution to distribution and between versions of distribution.

We added | less to the set command to **display the command output**, **one page at a time**. It is **optional**, but it's a good idea to use it when the **output is longer**. You can go to the **next pages** with [Enter] or **quit** with [q].

The set command can be used to **modify the parameters** that are driving the environment. In order to see what **parameters** are there, we can execute:

Картина, която съдържа текст

Описанието е генерирано автоматично

We can **see the same information** but prepared for **re-use** with:

Картина, която съдържа текст

Описанието е генерирано автоматично

Let's **change the flag** that **controls the amount of information** shown during commands execution:

Картина, която съдържа текст

Описанието е генерирано автоматично

You should note that the minus ("-") is used to **turn on a flag**, while the plus ("+") is used to **deactivate a flag**. Now, we can execute few more commands:

Картина, която съдържа текст

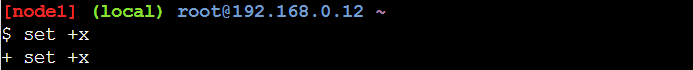
Описанието е генерирано автоматично

Картина, която съдържа маса

Описанието е генерирано автоматично

We can see that immediately **after each command** there is **information aboutse what is going to be executed**. This way we can see that instead ls, in fact something else is being executed – its **alias** or the statement it specifies.

It is time to **deactivate** the xtrace **mode**:



Now you will not be able to see what commands are executed.

## Getting Help

Now it is time to explore ways of **getting help** about different commands.

For the **shell built-in commands**, we can use the **help** command. If we execute it **without parameters**, it will return **all built-in commands**:



↓

Картина, която съдържа текст

Описанието е генерирано автоматично

In order to **ask for a command**, we must execute it like:



↓

Картина, която съдържа текст

Описанието е генерирано автоматично

Most **external commands** offer integrated help. The ways to **ask for this information** vary, but typically we can use:

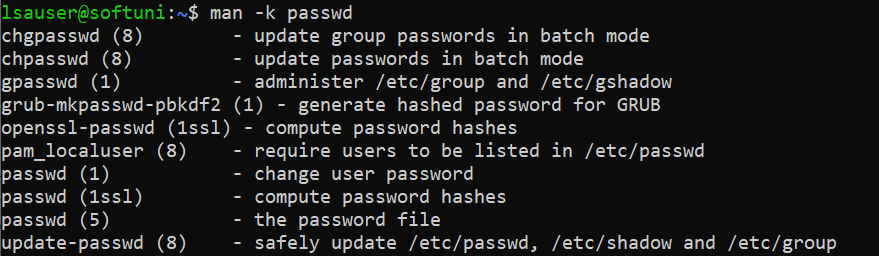
Картина, която съдържа текст

Описанието е генерирано автоматично

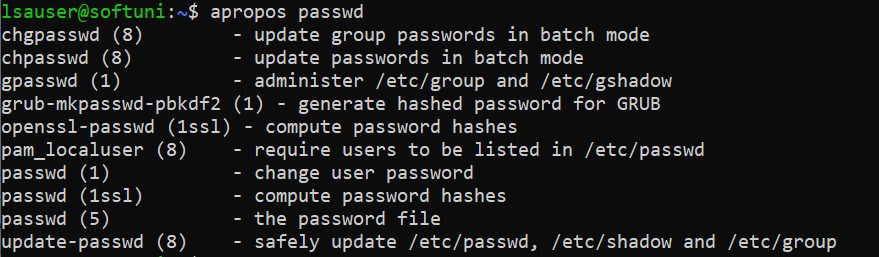
When using man, it is good to know the following **set of keys**:

* Key [h]shows **help information**
* Key [q] either **exits the help mode** or **the man**
* If we want to **search for something** from the current cursor position onwards, then we can press[/], then enter the string and then press the [Enter] key. Once in this mode, we can press [n]key to **move forward** over the matches, or the[N] key to **move backward**
* Searching backwards works the same, but it is initiated with the[?] key

A **quick search** in man can be done on the command line with:



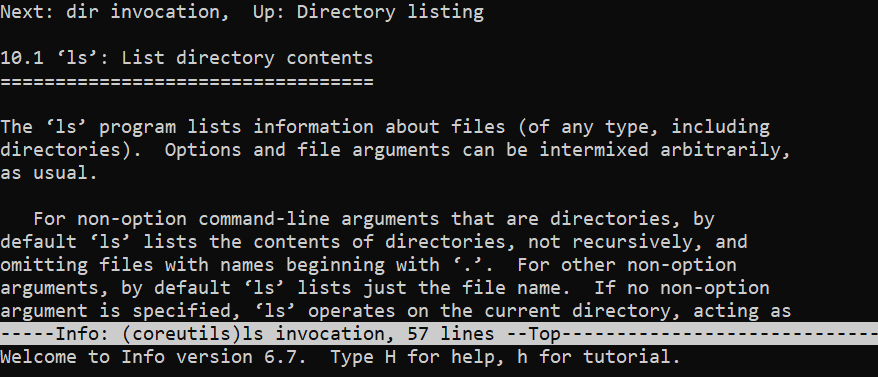
Similar effect can be achieved by using the apropos command:



Beside man, usually there is an alternative and modern help system available – info. We can use it the same way:



↓



For **help** inside the tool, we can press the [h] key. The help screen can be **closed** with the [x] key. No matter where we are in the tool, we can **exit** with the [q] key.