

UNITS 3 AND 4 ASSIGNMENT

The aim of the assignment is to “containerize” Git using Docker. We will do this in two ways.

- ✓ Running a script non-interactively
- ✓ Running a script interactively

Below you will find the commands you need to use for Git:

- **apt-get -y update**: To update repositories before installing Git.
- **apt-get -y install git**: To install git after updating the corresponding repository.
- **git config global user.name "Your name and surnames"**: Once Git installed, you must set the username. Replace the red text with your name, surnames or a different identifier.
- **git config --global user.email "Your e-mail address"**: Once Git installed, you must set the user e-mail. Replace the red text with your e-mail address (or you can use one invented).
- **git init**: We create a repository in the current folder. For example, if the current folder is /home/user, the new repository will be automatically created right there. This command can work without parameters referring to the current directory.
- **git add ***: Unlike on GitHub Desktop or through the website, using Git we can decide which files we want to commit. The command “git add *” allows you to move all the files to the staging area so that they can be committed. The parameter * refers to the files or directories in the current repository folder.
- **git commit -m "Text for the commit"**: A commit similar to GitHub, but you first need to run “git add”. Replace the red text with the commit message you want.
- **git log**: It shows all the commits made in the current repository.

Taking all the described commands into account, we will create the required containers to complete the following parts:

a) Running a script non-interactively

For this part, you have to create a script without parameters which will take the next steps:

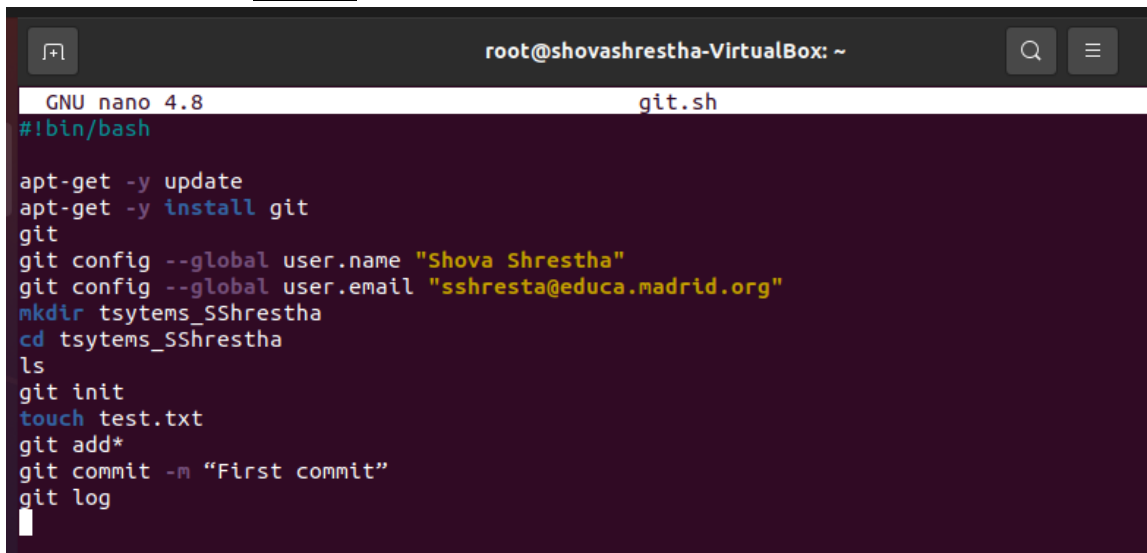
- Install Git.
- Configure Git with your name and e-mail.
- It creates a new folder called **systems_initials** into /root (if your name is John Doe, the folder will be /root/systems_JD as a result). The folder

should not be empty and will contain something which can be a blank file or subfolder. This new content of **/root/systems_initials** will be used later to create a repository

(you can create sample files or subfolders from the script with touch or mkdir respectively).

- Then, we change the current directory to **/root/systems_initials** (remember that you must replace “initials” with the corresponding ones to your name and surnames). Once there, we will create a repository (git init) and sample files or subfolders created from the script (remember that we first need to run “git add” and then “git commit”).
- The script output will be the current state running **git log**.

FIRST WE **CREATE OUR SCRIPT NAMED GIT.SH**, WE CREATED WITH THE COMMAND **NANO**.



```
GNU nano 4.8 git.sh
#!/bin/bash

apt-get -y update
apt-get -y install git
git
git config --global user.name "Shova Shrestha"
git config --global user.email "sshresta@educa.madrid.org"
mkdir tsystems_SShrestha
cd tsystems_SShrestha
ls
git init
touch test.txt
git add*
git commit -m "First commit"
git log
```

Finally, the script will be executed in a **container** based on the **Ubuntu** image. For this purpose, you must create a **Docker volume**. **The script will be manually copied to the volume folder** (run “docker inspect volume-name” to check the real folder). Afterwards, the Ubuntu container will be run with the command **bash** which executes the script from the volume. You can associate the volume with the Linux folder you want, for example `/root` (the command would be “`bash /root/script.sh`”) in this case.

AFTER CREATING THE SCRIPT WE CREATE A **Docker volume** WITH THE COMMAND: **DOCKER VOLUME CREATE GIT1-VOL**, then we inspect it, WITH THE COMMAND: **DOCKER INSPECT GIT1-VOL** W WE CAN SEE THAT OUR VOLUME IS IN `/var/lib/docker/volumes/git1-vol/_data`.

```
root@shovashrestha-VirtualBox:~# nano git.sh
root@shovashrestha-VirtualBox:~# docker volume create git1-vol
git1-vol
root@shovashrestha-VirtualBox:~# docker inspect git1-vol
[
  {
    "CreatedAt": "2021-12-16T17:52:56+01:00",
    "Driver": "local",
    "Labels": {},
    "Mountpoint": "/var/lib/docker/volumes/git1-vol/_data",
    "Name": "git1-vol",
    "Options": {},
    "Scope": "local"
  }
]
```

The script will be manually copied to the **volume folder** (run “docker inspect volume-name” to check the real folder).

TO COPY THE SCRIPT TO THE VOLUME FOLDER WE NEED THE COMMAND CP THE NAME OF THE SCRIPT AND THE PATH OF THE VOLUME FOLDER. IN MY CASE:

cp git.sh /var/lib/docker/volumes/git1-vol/_data.

```
root@shovashrestha-VirtualBox:~# cp git.sh /var/lib/docker/volumes/git1-vol/_data
root@shovashrestha-VirtualBox:~# ls
git.sh  Main.java  snap
root@shovashrestha-VirtualBox:~# cat git.sh
#!/bin/bash

apt-get -y update
apt-get -y install git
git
git config --global user.name "Shova Shrestha"
git config --global user.email "sshresta@educa.madrid.org"
mkdir tsytems_SShrestha
cd tsytems_SShrestha
ls
git init
touch test.txt
git add*
git commit -m "First commit"
git log
root@shovashrestha-VirtualBox:~# cd /var/lib/docker/volumes/git1-vol/_data
root@shovashrestha-VirtualBox:/var/lib/docker/volumes/git1-vol/_data# ls
git.sh
```

Afterwards, the Ubuntu container will be run with the command **bash** which executes the script from the volume. You can associate the volume with the Linux folder you want, for example `/root` (the command would be “`bash /root/script.sh`”) in this case.

Use a volume in a container

TO EXECUTE THE SCRIPT WE NEED TO **CREATE A CONTAINER THEN INSERT THE VOLUME, WHERE IS THE SCRIPT, IN THAT CONTAINER AND EXECUTE IT WITH BASH** TO DO THAT WE NEED THE COMMAND:

MY CONTAINER NAME IS UBUNTU1, THE VOLUME IS GIT1-VOL, DESTINATION /USR/SRC/ THEN WE PUT THE IMAGE THAT WE ARE GOING TO USE, IN MY CASE UBUNTU, THEN THE BASH COMMAND TO EXECUTE THE SCRIPT AND THE PATH WHERE IS THE SCRIPT WHICH IS /USR/SRC/GIT.SH

```
docker run --rm --name ubuntu2 --mount source=git1-vol,destination=/usr/src/ ubuntu bash /usr/src/git.sh
root@shovashrestha-VirtualBox:~# docker run --rm --name ubuntu2 --mount source=git1-vol,destination=
/usr/src/ ubuntu bash /usr/src/git.sh
Get:1 http://security.ubuntu.com/ubuntu focal-security InRelease [114 kB]
Get:2 http://archive.ubuntu.com/ubuntu focal InRelease [265 kB]
Get:3 http://archive.ubuntu.com/ubuntu focal-updates InRelease [114 kB]
Get:4 http://archive.ubuntu.com/ubuntu focal-backports InRelease [108 kB]
Get:5 http://archive.ubuntu.com/ubuntu focal/restricted amd64 Packages [33.4 kB]
Get:6 http://archive.ubuntu.com/ubuntu focal/main amd64 Packages [1275 kB]
Get:7 http://security.ubuntu.com/ubuntu focal-security/main amd64 Packages [1468 kB]
Get:8 http://archive.ubuntu.com/ubuntu focal/universe amd64 Packages [11.3 MB]
Get:9 http://security.ubuntu.com/ubuntu focal-security/restricted amd64 Packages [889 kB]
Get:10 http://archive.ubuntu.com/ubuntu focal/multiverse amd64 Packages [177 kB]
Get:11 http://archive.ubuntu.com/ubuntu focal-updates/universe amd64 Packages [1119 kB]
Get:12 http://security.ubuntu.com/ubuntu focal-security/universe amd64 Packages [837 kB]
Get:13 http://security.ubuntu.com/ubuntu focal-security/multiverse amd64 Packages [30.1 kB]
Get:14 http://archive.ubuntu.com/ubuntu focal-updates/restricted amd64 Packages [952 kB]
Get:15 http://archive.ubuntu.com/ubuntu focal-updates/multiverse amd64 Packages [33.7 kB]
```

What happens? At this point, we have executed the script and we have the output of **git log**. But we cannot interact with the container. So, let's move on the next part, which makes more sense.

b) Running a script interactively

We will use the same script as in the previous part. Although this time we will run an interactive **Ubuntu container** with a shell bash. The script will be in a volume too. But we must manually run the script from the terminal.

WE ALREADY HAVE CREATED THE SCRIPT, THE VOLUME, MOVED THE SCRIPT IN THE VOLUME FILE SO IN THIS CASE WE JUST NEED THE COMMAND, IN THIS CASE **WE NEED THE -IT INTERACTIVE COMMAND**, THE **CONTAINER NAME IS UBUNTU3**, THE **DESTINATION IS THE SAME ONE HAS BEFORE /USR/SRC/**, THEN WE USE THE **IMAGE** THAT WE ARE GOING TO USE, **UBUNTU** AND THE **BASH** COMMAND.

```
docker run -it --rm --name ubuntu3 --mount source=git1-vol,destination=/usr/src/ ubuntu bash
```

```
root@shovashrestha-VirtualBox:~# docker run -it --rm --name ubuntu3 --mount source=git1-vol,destination=/usr/src/ ubuntu bash
root@db25dbc6b27d:/# ls
bin  dev  home  lib32  libx32  mnt  proc  run  srv  tmp  var
boot  etc  lib  lib64  media  opt  root /sbin  sys  usr
root@db25dbc6b27d:/# docker pwd
bash: docker: command not found
root@db25dbc6b27d:/# cd /usr/src
root@db25dbc6b27d:/usr/src# ls
git.sh
root@db25dbc6b27d:/usr/src# bash git.sh
```

What happens? This alternative requires to manually execute the script from the container. If you “exit” from the interactive bash, the container will stop.

WE ARE LOGGED IN THE CONTAINER.

TO EXECUTE THE SCRIPT WE NEED TO **DO IT MANUALLY** WITH THE COMMAND **bash git.sh**

IT HAS EXECUTED CORRECTLY

```
root@06042c8fe31b:/# bash git.sh
Get:1 http://security.ubuntu.com/ubuntu focal-security InRelease [114 kB]
Get:2 http://archive.ubuntu.com/ubuntu focal InRelease [265 kB]
Get:3 http://archive.ubuntu.com/ubuntu focal-updates InRelease [114 kB]
Get:4 http://archive.ubuntu.com/ubuntu focal-backports InRelease [108 kB]
Get:5 http://security.ubuntu.com/ubuntu focal-security/restricted amd64 Packages [833 kB]
Get:6 http://archive.ubuntu.com/ubuntu focal/multiverse amd64 Packages [177 kB]
Get:7 http://archive.ubuntu.com/ubuntu focal/main amd64 Packages [1275 kB]
Get:8 http://security.ubuntu.com/ubuntu focal-security/multiverse amd64 Packages [30.1 kB]
Get:9 http://security.ubuntu.com/ubuntu focal-security/universe amd64 Packages [837 kB]
Get:10 http://security.ubuntu.com/ubuntu focal-security/main amd64 Packages [1417 kB]
Get:11 http://archive.ubuntu.com/ubuntu focal/restricted amd64 Packages [33.4 kB]
Get:12 http://archive.ubuntu.com/ubuntu focal/universe amd64 Packages [11.3 MB]
```

AS WE CAN SEE IF WE DO LS WE CAN SEE THE FOLDER **tsystems_SShrestha** AND INSIDE THE FOLDER IS THE **test.txt** DOCUMENT WHICH WE CREATED IN THE SCRIPT

```
root@db25dbc6b27d:/usr/src# ls
git.sh  tsystems_SShrestha
root@db25dbc6b27d:/usr/src# cd tsystems_SShrestha
root@db25dbc6b27d:/usr/src/tsystems_SShrestha# ls
test.txt
root@db25dbc6b27d:/usr/src/tsystems_SShrestha# cat test.txt
root@db25dbc6b27d:/usr/src/tsystems_SShrestha#
```

You can restart the container with **docker start**. You **must also explain how to run the same container again** with an interactive terminal.

1. List all dockers by using this command and note the container id of the container you want to restart: `docker ps -a`.
2. Start your container using container id: `docker start <container_id>`
3. Attach and run your container: `docker attach <container_id>`

WE CAN SEE IN THE IMAGE THAT WE HAVE THE **CONTAINER UBUNTU2** WHICH IS **SHUT DOWN**, TO USE IT AGAIN WE CAN DO IT WITH THE COMMAND **docker start name_of_container**. **docker start ubuntu2**

IF WE USE THE COMMAND **docker ps** WE CAN SEE THAT THE **CONTAINER IS UP**.

```
root@shovashrestha-VirtualBox:~# docker ps -a
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS   NAMES
7956d76857c0   ubuntu   "bash"    13 seconds ago   Exited (0) 4 seconds ago   ubuntu2
root@shovashrestha-VirtualBox:~# docker start 7956d76857c0
7956d76857c0
root@shovashrestha-VirtualBox:~# docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS   NAMES
7956d76857c0   ubuntu   "bash"    49 seconds ago   Up 5 seconds                ubuntu2
root@shovashrestha-VirtualBox:~#
```

You **must also explain how to run the same container again** with an interactive terminal.

TO RUN THE SAME CONTAINER WITH AN INTERACTIVE TERMINAL WE USE THE COMMAND `docker attach name_container`, **docker attach ubuntu2**

```
root@shovashrestha-VirtualBox:~# docker attach 7956d76857c0
root@7956d76857c0:/# ls
bin boot dev etc home lib lib32 lib64 libx32 media mnt opt proc root run sbin srv sys tmp usr var
root@7956d76857c0:/# exit
exit
root@shovashrestha-VirtualBox:~# docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS   NAMES
root@shovashrestha-VirtualBox:~#
```


c) Remove all the containers and images created. Explain the commands you need.

WITH `docker ps -a` WE CAN SEE THE SHUTDOWN CONTAINER.

TO REMOVE ALL THE CONTAINERS WITH ONE COMMAND WE USE `docker rm -fv `docker ps -aq``, WE USE IT AND AS YOU CAN SEE IT'S ALL REMOVED.

```
root@shovashrestha-VirtualBox:/home/shovashrestha# cd $home
root@shovashrestha-VirtualBox:~# docker ps -a
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS   NAMES
95b901d2017b   ubuntu   "bash"    2 hours ago   Exited (0) 2 hours ago   ubuntu1
root@shovashrestha-VirtualBox:~# docker rm -fv `docker ps -aq`
95b901d2017b
root@shovashrestha-VirtualBox:~# docker ps -a
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS   NAMES
root@shovashrestha-VirtualBox:~#
```

WE CAN SEE ALL THE IMAGES THAT I HAVE IN THIS MOMENT WITH THE COMMAND `docker images`, NOW TO REMOVE ALL THE IMAGES WE USE THE COMMAND `docker rmi `docker images -q`` IF WE PUT AGAIN `docker images` WE CAN SEE THAT ALL OF THEM HAVE BEEN REMOVED

```
root@shovashrestha-VirtualBox:~# docker images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
bash          latest   5557e073f11c   12 days ago   13MB
openjdk       latest   1b3756d6df61   2 months ago  471MB
ubuntu        latest   ba6accedd29    3 months ago  72.8MB
hello-world   latest   feb5d9fea6a5   3 months ago  13.3kB
root@shovashrestha-VirtualBox:~# docker rmi `docker images -q`
Untagged: bash:latest
Untagged: bash@sha256:f34654f94d3d227b7a19e2e8156c8ac7e68f45639d67ba8acee87c18f54261ca
Deleted: sha256:5557e073f11c1ffdd418c40394b79c736b492fd7ddf95124d022290985e09c55
Deleted: sha256:a1479d0e989665fbd3be58effba33ac682442d6c4eeffbf782afa396a0c77c4
Deleted: sha256:cf727866ee06acfad178a0fe3159a093b3dc9224cbb463280eccc8d26ed4e709
Deleted: sha256:1a058d5342cc722ad5439cacae4b2b4eedde51d8fe8800fcf28444302355c16d
Untagged: openjdk:latest
Untagged: openjdk@sha256:0e5ae79482731eef1526afb4e3a42e62b38142681c5752a944ff4236da979648
Deleted: sha256:1b3756d6df61476ba3a7a060e4f0b9cd2a4ff7ac7777520da2ac4c0e9b8caa71
Deleted: sha256:7f80312d296c20460ad9bf4dda03a184524e9b6a3e8b0eea53bebed40ea3c70
Deleted: sha256:391032f15bc8d9f319caf9575f703f32f9e5946128788cf1732ca4e4bce45f99
Deleted: sha256:32ac9dd9610b7b8c69248fb0454d9aad8082c327a5a7a46c7527a7fc0c15be37
Untagged: ubuntu:latest
Untagged: ubuntu@sha256:626ffe58f6e7566e00254b638eb7e0f3b11d4da9675088f4781a50ae288f3322
Deleted: sha256:ba6accedd2923aee4c2acc6a23780b14ed4b8a5fa4e14e252a23b846df9b6c1
Deleted: sha256:9f54eef412758095c8079ac465d494a2872e02e90bf1fb5f12a1641c0d1bb78b
Untagged: hello-world:latest
Untagged: hello-world@sha256:cc15c5b292d8525effc0f89cb299f1804f3a725c8d05e158653a563f15e4f685
Deleted: sha256:feb5d9fea6a5e9606aa995e879d862b825965ba48de054caab5ef356dc6b3412
Deleted: sha256:e07ee1baac5fae6a26f30cabfe54a36d3402f96afda318fe0a96cec4ca393359
root@shovashrestha-VirtualBox:~# docker images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
root@shovashrestha-VirtualBox:~#
```


INSTRUCTIONS

- You must create a document with screenshots to explain step by step that the two parts work. The filename format will be Units3-4_Name_Surname.pdf.
- You must also submit the script required in a and b. The filename format will be Units3-4_Name_Surname.sh.
- So, the submission will be a ZIP file containing: the document with screenshots and the script. The filename format will be Units3-4_Name_Surname.zip.

ALL THE EXPLANATIONS AND SCREENSHOTS MUST BE IN ENGLISH. If found at least one in Spanish, you will get a mark of 0.

