**HOME**

For these exercises, use a Docker container running the latest Ubuntu image (docker run --rm -it ubuntu 'bash') [**TASK 0**](#task0)

1. Set the bash prompt to include the time and shell name. [**TASK 1**](#task1)

2. Install HTTPD and configure it to listen on port 8081. How can you do these two steps in one bash line? [**TASK 2**](#task2)

3. Change the default landing page and display something of your choosing. [**TASK 3**](#task3)

4. Secure Apache by installing an SSL certificate. [**TASK 4**](#task4)

5. Follow the HTTPD logs from the exercise above, but only print log lines from 4xx and 5xx status codes/responses. [**TASK 5**](#task5)

6. Create a script that sends the status, PID and any lines containing error messages related to Apache service via email. [**TASK 6**](#task6)

7. Write a script that uses sh (not bash) and that prints the size of each user's home directory. [**TASK 7**](#task7)

8. Script a guessing game. The range of numbers is 0-10. If the user guess is outside this range, or under or above the number to-be-guessed, print a useful message (e.g., "your guess is higher than the number!"). Allow the user to make 3 guesses before exiting with status code 27. All STDOUT messages are to be printed with cowsay. [**TASK 8**](#task8)

9. Create a script that shows the CPU, memory and disk usage every 30 minutes and writes the output to a file. [**TASK 9**](#task9)

10. Commit the scripts in different branches in your git repository using git commit messages via the command line. [**TASK 10**](#task10)

**TASK 0** [**-HOME**](#home)

For these exercises, use a Docker container running the latest Ubuntu image (docker run --rm -it ubuntu 'bash')

Script:

docker run -td -p 80:8081 --name 3pillar ubuntu

docker ps -a

docker exec -it 3pillar bash

apt-get update && apt install vim -y && apt install git -y

git clone https://github.com/erwin8989/3pillar.git

git config --global user.name "erwin8989"

git config --global user.email k\_erwin1@yahoo.com

cd 3pillar

git branch newbranch0

git checkout newbranch0

git branch

mkdir 0.dockerContainer && cd 0.dockerContainer

vim 1.task.txt

vim 2.script.txt

cd ..

git status

git add .

git commit -m "Setting up docker container + Addig Task + Script"

git push origin newbranch0

**TASK 1** [**-HOME**](#home)

Set the bash prompt to include the time and shell name.

Script:

git branch newbranch1

git checkout newbranch1

git branch

mkdir 1.promptBash && cd 1.promptBash

vim 1.task.txt

vim 2.script.txt

vim prompt.bash

cd ~

echo "PS1=['\d - $SHELL] > \W $ '" >> .bashrc

bash --login

chmod +x prompt.bash

bash prompt.bash

cd /3pillar

git status

git add .

git commit -m "Committing bash prompt + Addig Task + Script"

git push origin newbranch1

**TASK 2** [**-HOME**](#home)

Install HTTPD and configure it to listen on port 8081. How can you do these two steps in one bash line?

Script:

git branch newbranch2

git checkout newbranch2

git branch

mkdir 2.port8081 && cd 2.port8081

vim 1.task.txt

vim 2.script.txt

apt install apache2 -y

vim port8081.bash

sed -i 's/80/8081/g' /etc/apache2/ports.conf && sed -i 's/80/8081/g' /etc/apache2/sites-enabled/000-default.conf && service apache2 restart && service apache2 status

chmod +x port8081.bash

bash port8081.bash

cp /etc/apache2/ports.conf /3pillar/2.port8081 && cp /etc/apache2/sites-enabled/000-default.conf /3pillar/2.port8081

cd ..

git status

git add .

git commit -m "Committing listen on port 8081 + Addig Task + Script"

git push origin newbranch2

**TASK 3** [**-HOME**](#home)

Change the default landing page and display something of your choosing

Script:

git branch newbranch3

git checkout newbranch3

git branch

mkdir 3.landingPage && cd 3.landingPage

vim 1.task.txt

vim 2.script.txt

cd /var/www/html/ && > index.html && vim index.html

*<html>*

*<body>*

*<h1>My first heading</h1>*

*<p>My first paragraph</p>*

*</body>*

*</html>*

cd /3pillar/3.landingPage

cp /var/www/html/index.html /3pillar/3.landingPage

cd ..

git status

git add .

git commit -m "Change default landing page + Addig Task + Script"

git push origin newbranch3

**TASK 4** [**-HOME**](#home)

Secure Apache by installing an SSL certificate

Script:

git branch newbranch4

git checkout newbranch4

git branch

mkdir 4.sslCertificate && cd 4.sslCertificate

vim 1.task.txt

vim 2.script.txt

cd /etc/apache2/ && mkdir ssl

openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout /etc/apache2/ssl/apache.key -out /etc/apache2/ssl/apache.crt

vim /etc/apache2/apache2.conf

ServerName localhost

vim /etc/apache2/sites-available/mysite-ssl.conf

<VirtualHost \*:443>

SSLEngine on

SSLCertificateFile /etc/apache2/ssl/apache.crt

SSLCertificateKeyFile /etc/apache2/ssl/apache.key

ServerName localhost

DocumentRoot /var/www/html

</VirtualHost>

a2ensite mysite-ssl.conf

a2dissite default-ssl.conf

a2enmod ssl

service apache2 restart && service apache2 status

cd /3pillar

git status

git add .

git commit -m "Installing SSL Cerficates + Addig Task + Script"

git push origin newbranch4

**TASK 5** [**-HOME**](#home)

Follow the HTTPD logs from the exercise above, but only print log lines from 4xx and 5xx status codes/responses.

Script:

git branch newbranch5

git checkout newbranch5

git branch

mkdir 5.httpdLogs && cd 5.httpdLogs

vim 1.task.txt

vim 2.script.txt

cat /var/log/apache2/access.log | grep " 4.. " && grep " 5.. "

ctrl + c

cp /var/log/apache2/access.log /3pillar/5.httpdLogs

cd ..

git status

git add .

git commit -m "Print HTTPD logs + Addig Task + Script"

git push origin newbranch5

**TASK 6** [**-HOME**](#home)

Create a script that sends the status, PID and any lines containing error messages related to Apache service via email.

Script:

git branch newbranch6

git checkout newbranch6

git branch

mkdir 6.errorSending && cd 6.errorSending

vim 1.task.txt

vim 2.script.txt

cd ..

git status

git add .

git commit -m "Adding Task + Script"

git push origin newbranch6

**TASK 7** [**-HOME**](#home)

Write a script that uses sh (not bash) and that prints the size of each user's home directory.

Script:

git branch newbranch7

git checkout newbranch7

git branch

mkdir 7.homeSize && cd 7.homeSize

vim 1.task.txt

vim 2.script.txt

vim usershome.sh

awk -F: '{print $1,$6}' /etc/passwd

awk -F: '{print $6}' /etc/passwd

chmod +x usershome.sh

sh usershome.sh

du -sh (for each home directory)

cd ..

git status

git add .

git commit -m "Committing the users home directory + Adding Task + Script"

git push origin newbranch7

**TASK 8** [**-HOME**](#home)

Script a guessing game. The range of numbers is 0-10. If the user guess is outside this range, or under or above the number to-be-guessed, print a useful message (e.g., "your guess is higher than the number!"). Allow the user to make 3 guesses before exiting with status code 27. All STDOUT messages are to be printed with cowsay.

Script:

git branch newbranch8

git checkout newbranch8

git branch

mkdir 8.guessingGame && cd 8.guessingGame

vim 1.task.txt

vim 2.script.txt

apt install python3 python3-pip build-essential python3-dev -y && pip install cowsay

vim game.py

*import sys*

*import random*

*import cowsay*

*number = random.randint(0, 10)*

*cowsay.cow('Try to guess the number between 0 to 10 in 3 guesses!')*

*guess1 = int(input('\n Enter your first guess: '))*

*if guess1 == number:*

*cowsay.cow('You guest in on the first try! Congratulation!')*

*sys.exit()*

*elif guess1 > number:*

*cowsay.cow('Your guess is higher than the number!')*

*else:*

*cowsay.cow("Your guess is lower than the number!")*

*guess2 = int(input('\n Enter your second guess: '))*

*if guess1 == number:*

*cowsay.cow('You guest in on the second try! Great job!')*

*sys.exit()*

*elif guess2 > number:*

*cowsay.cow('Your guess is higher than the number!')*

*else:*

*cowsay.cow("Your guess is lower than the number!")*

*guess3 = int(input('\n Enter your third guess: '))*

*if guess3 == number:*

*cowsay.cow('You guest in on the third try! Nice!')*

*elif guess3 > number:*

*cowsay.cow('Your guess is higher than the number: ' + str(number))*

*cowsay.cow('Sorry, you just lost!')*

*else:*

*cowsay.cow('Your guess is lower than the number: ' + str(number))*

*cowsay.cow('Sorry, you just lost!')*

python3 game.py

cd ..

git status

git add .

git commit -m "Committing the game + Adding Task + Script"

git push origin newbranch8

**TASK 9** [**-HOME**](#home)

Create a script that shows the CPU, memory and disk usage every 30 minutes and writes the output to a file.

Script:

git branch newbranch9

git checkout newbranch9

git branch

mkdir 9.usage && cd 9.usage

vim 1.task.txt

vim 2.script.txt

apt-get install -y cron && cron

vim system\_stats.sh

echo TIME = $(date)

echo MEMORY = $(free -m | awk 'NR==2{printf "%.2f%%\t\t", $3\*100/$2 }')

echo DISK = $(df -h | awk '$NF=="/"{printf "%s\t\t", $5}')

echo CPU = $(top -bn1 | grep load | awk '{printf "%.2f%%\t\t\n", $(NF-2)}')chmod +x system\_stats.sh

echo

chmod +x system\_stats.sh

vim /etc/crontab

\*/30 \* \* \* \* root date && /bin/sh /3pillar/9.usage/system\_stats.sh >> /3pillar/9.usage/timelog.txt

cp /etc/crontab /3pillar/9.usage

cd ..  
git status

git add .

git commit -m "Committing script and rontab + Adding Task + Script"

git push origin newbranch9

**TASK 10** [**-HOME**](#home)

Commit the scripts in different branches in your git repository using git commit messages via the command line.

Script:

git branch newbranch10

git checkout newbranch10

git branch

mkdir 10.gitRepo && cd 10.gitRepo

vim 1.task.txt

vim 2.script.txt

cd ..

git status

git add .

git commit -m "Adding Task + Script"

git push origin newbranch10