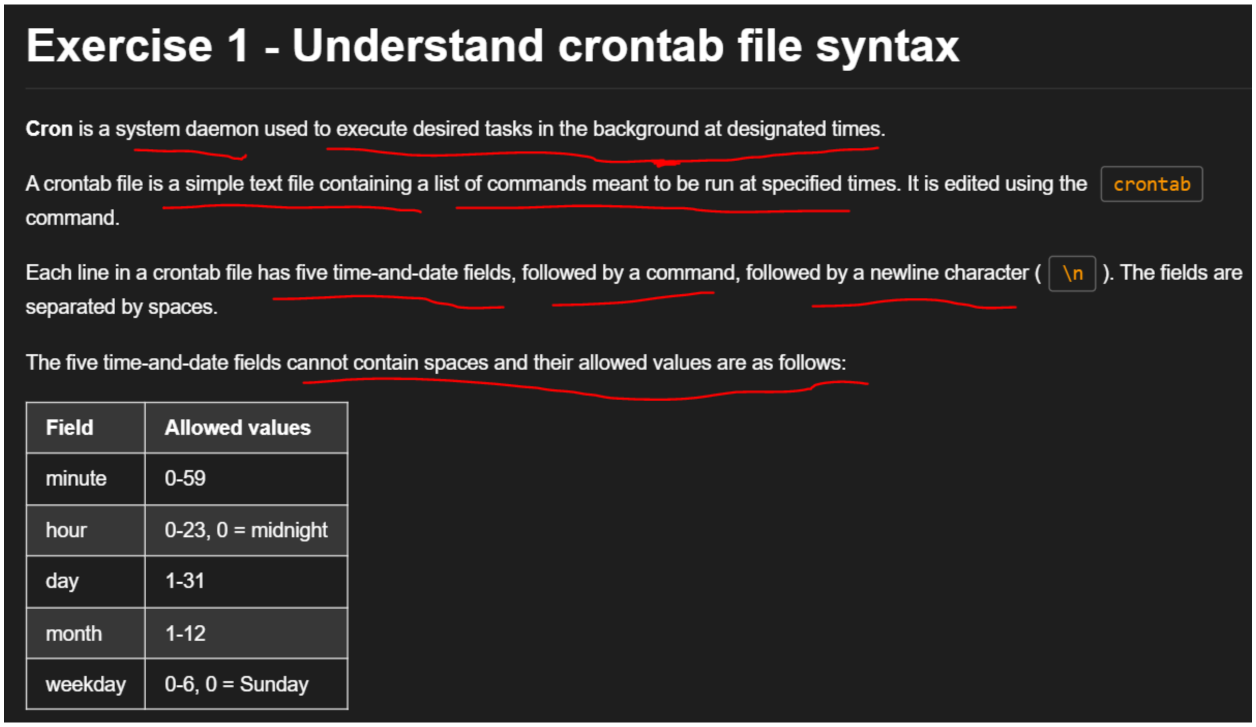
Hands-on Lab: Scheduling Jobs using crontab

Objectives

After completing this lab you will be able to:

- List existing cron jobs
- Add a cron job
- Remove cron jobs



Exercise 1 - Understand crontab file syntax

Cron is a system daemon used to execute desired tasks in the background at designated times.

A crontab file is a simple text file containing a list of commands meant to be run at specified times. It is edited using the `crontab` command.

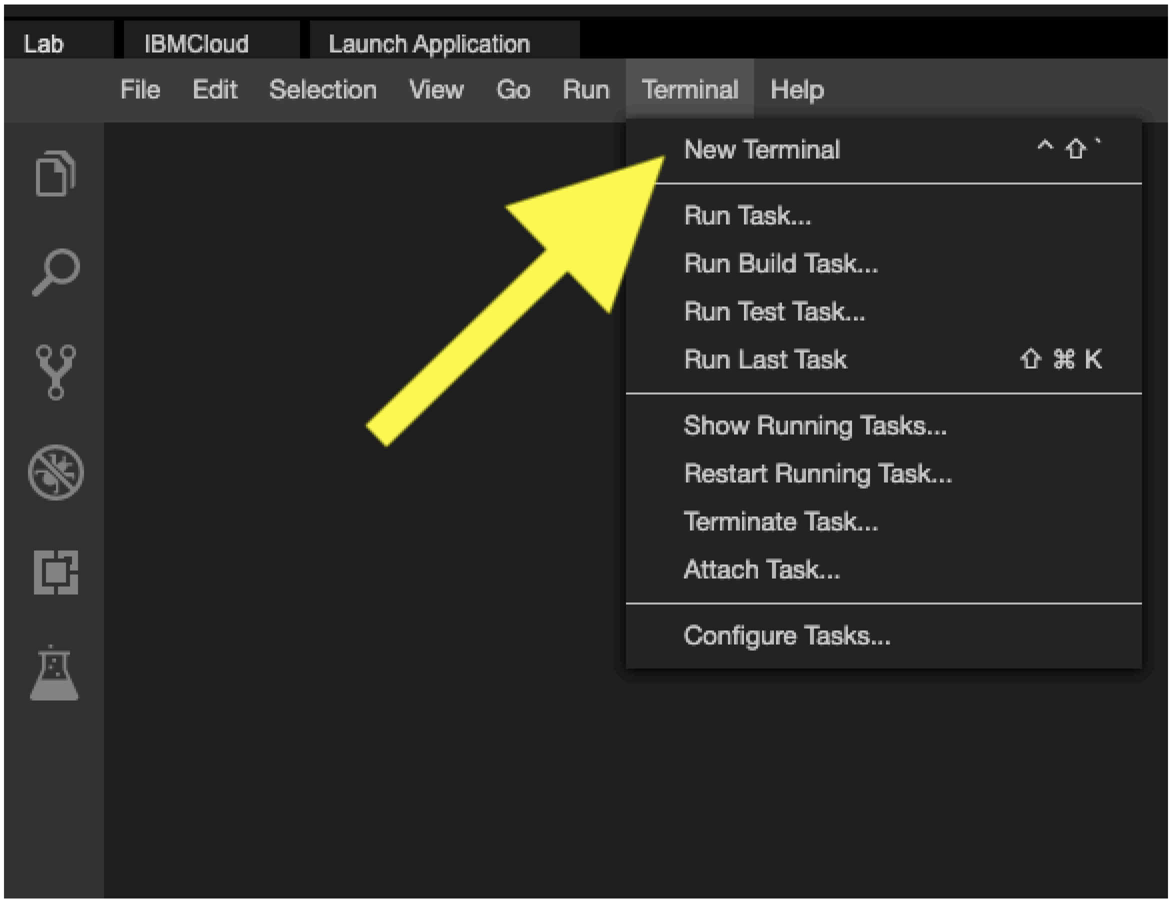
Each line in a crontab file has five time-and-date fields, followed by a command, followed by a newline character (`\n`). The fields are separated by spaces.

The five time-and-date fields cannot contain spaces and their allowed values are as follows:

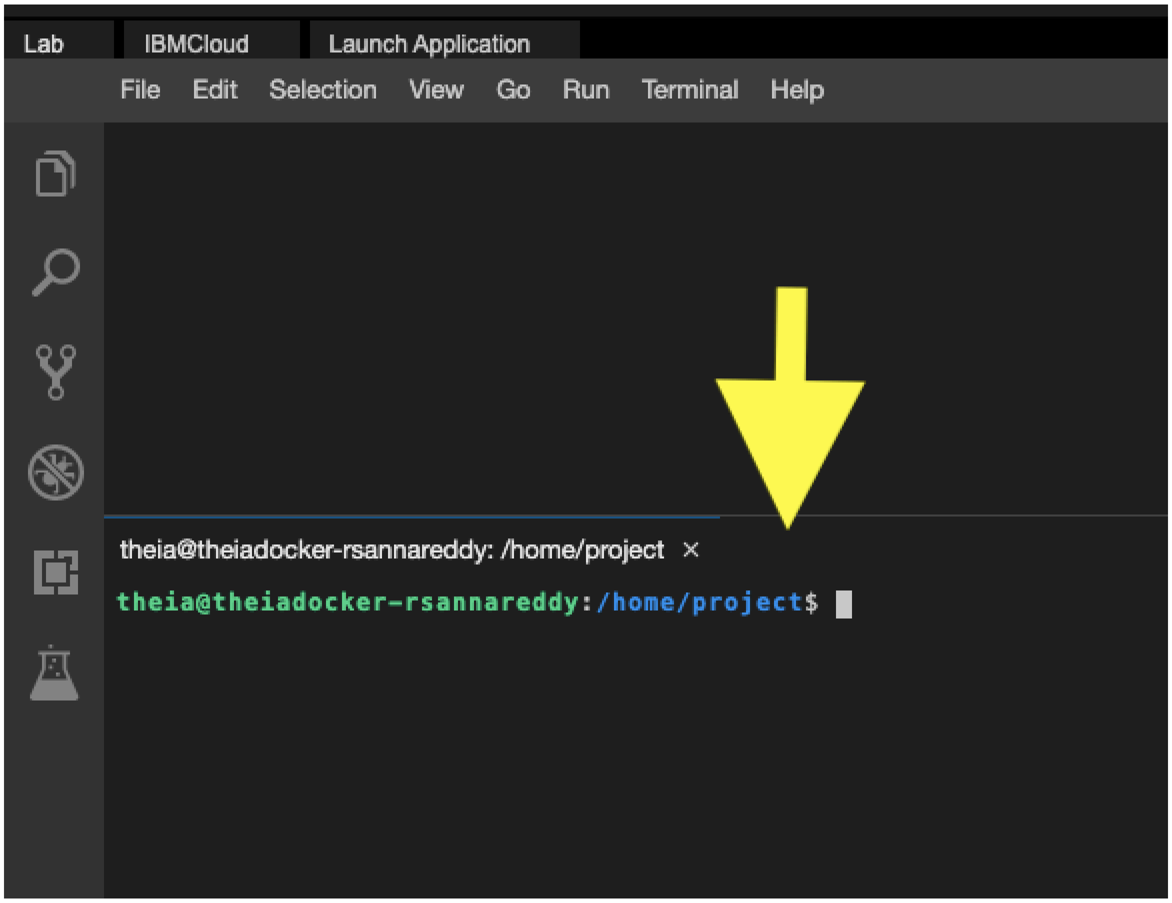
Field	Allowed values
minute	0-59
hour	0-23, 0 = midnight
day	1-31
month	1-12
weekday	0-6, 0 = Sunday

Exercise 2 - List cron jobs

Open a new terminal, by clicking on the menu bar and selecting **Terminal→New Terminal**, as in the image below.



This will open a new terminal at the bottom of the screen as in the image below.



```
theia@theiadocker-rsannareddy:/home/project$
```

Run the commands below on the newly opened terminal.

The `-l` option of the `crontab` command prints the current crontab.

1. `crontab -l`

You may get a message `no crontab for theia` if your crontab is empty.

Exercise 3 - Add a job in the crontab file

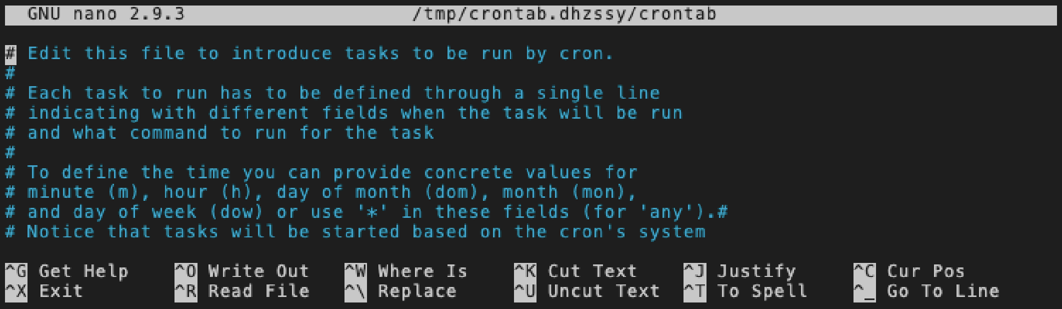
3.1. Add a job to crontab

To add a cron job, run the command below:

```
crontab -e
```

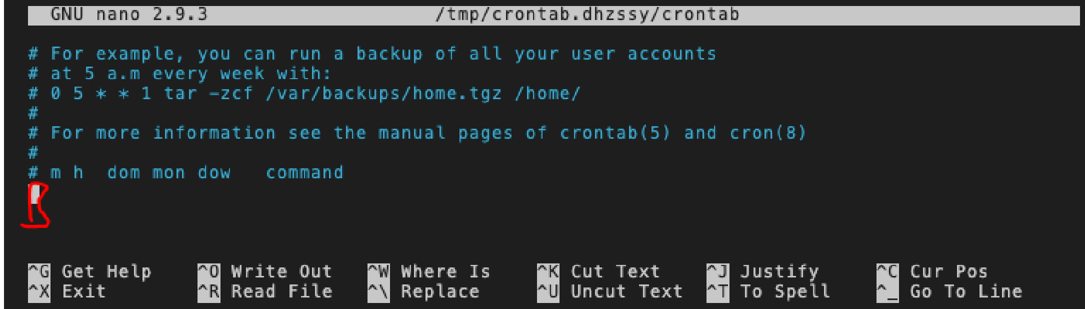
This will create a new crontab file for you (if you don't have one already). Now you are ready to add a new cron job.

Your crontab file will be opened in an editor as shown in the image below:

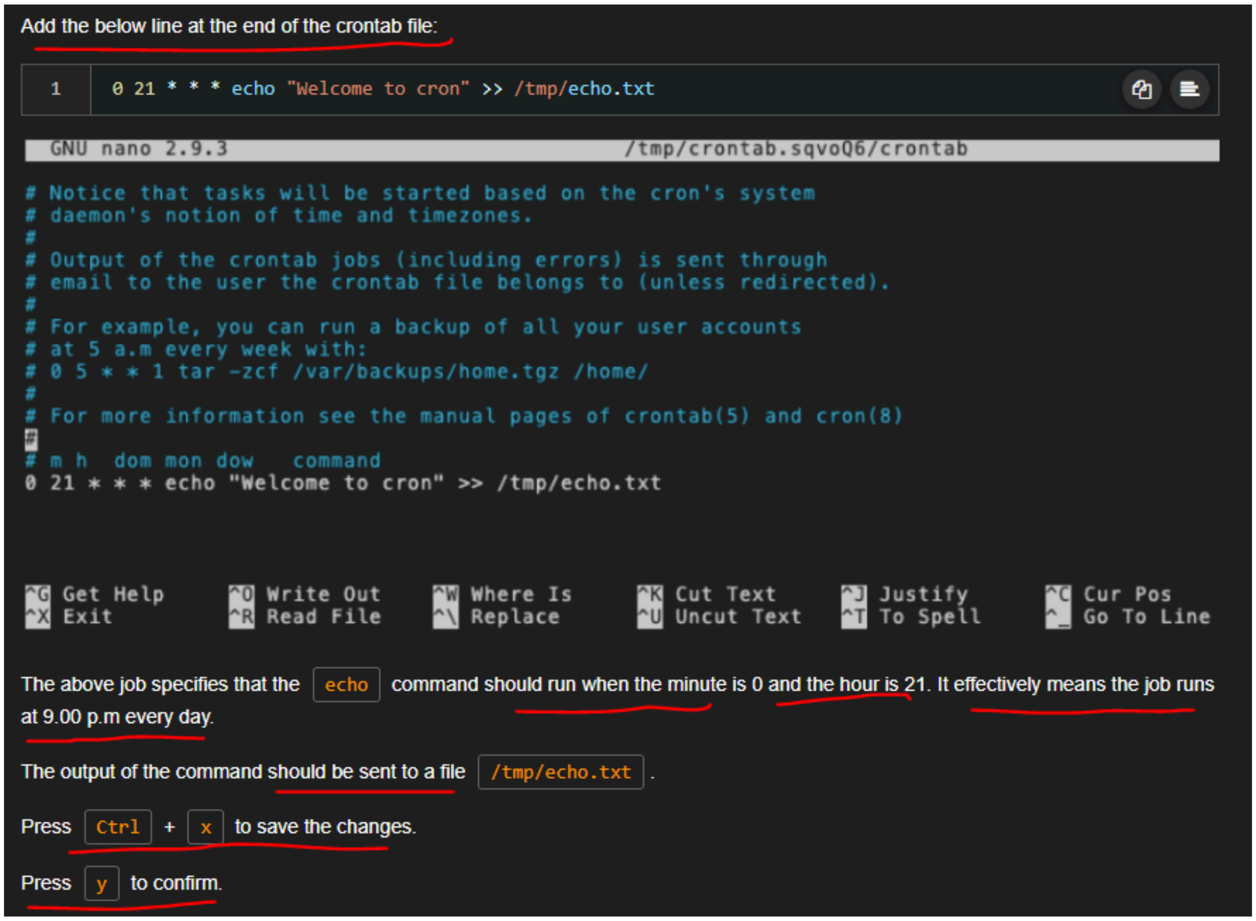


```
  GNU nano 2.9.3                    /tmp/crontab.dhzssy/crontab

# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').#
# Notice that tasks will be started based on the cron's system

^G Get Help    ^O Write Out   ^W Where Is    ^K Cut Text    ^J Justify     ^C Cur Pos
^X Exit        ^R Read File   ^\ Replace     ^U Uncut Text  ^T To Spell    ^_ Go To Line
```

Scroll down to the end of the file using the arrow keys:



```
  GNU nano 2.9.3                    /tmp/crontab.dhzssy/crontab

# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h  dom mon dow   command


^G Get Help    ^O Write Out   ^W Where Is    ^K Cut Text    ^J Justify     ^C Cur Pos
^X Exit        ^R Read File   ^\ Replace     ^U Uncut Text  ^T To Spell    ^_ Go To Line
```



Add the below line at the end of the crontab file:

```
0 21 * * * echo "Welcome to cron" >> /tmp/echo.txt
```

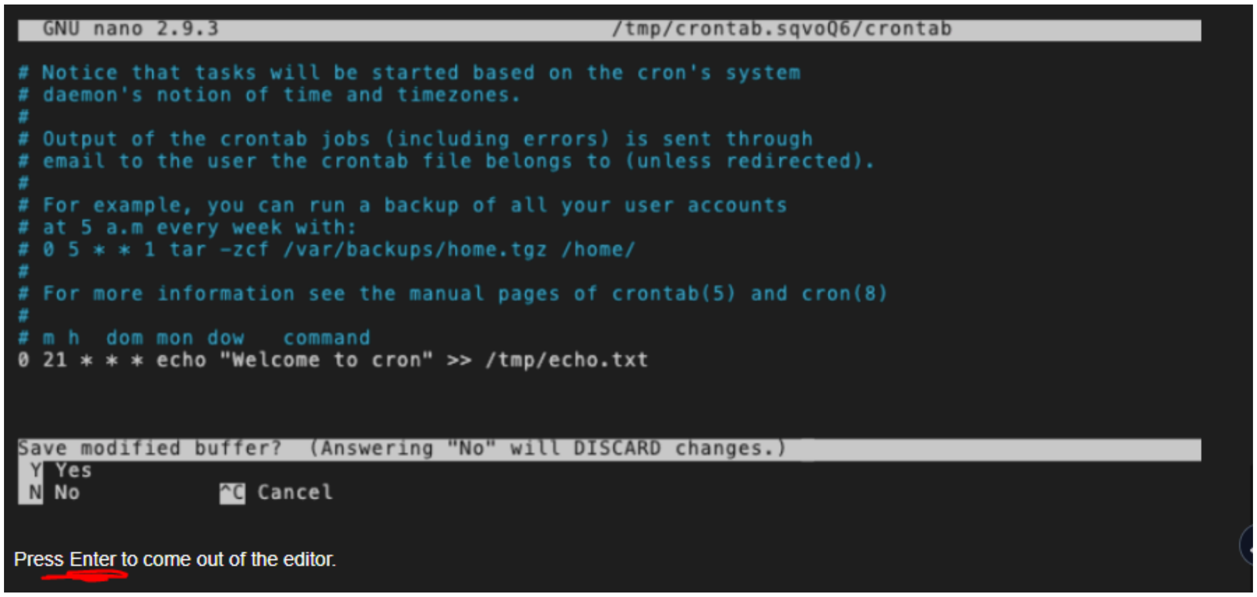
```
  GNU nano 2.9.3                            /tmp/crontab.sqvoQ6/crontab

# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h  dom mon dow   command
0 21 * * * echo "Welcome to cron" >> /tmp/echo.txt



^G Get Help    ^O Write Out   ^W Where Is    ^K Cut Text    ^J Justify     ^C Cur Pos
^X Exit        ^R Read File   ^\ Replace     ^U Uncut Text  ^T To Spell    ^_ Go To Line
```

The above job specifies that the `echo` command should run when the minute is 0 and the hour is 21. It effectively means the job runs at 9.00 p.m every day.

The output of the command should be sent to a file `/tmp/echo.txt`.

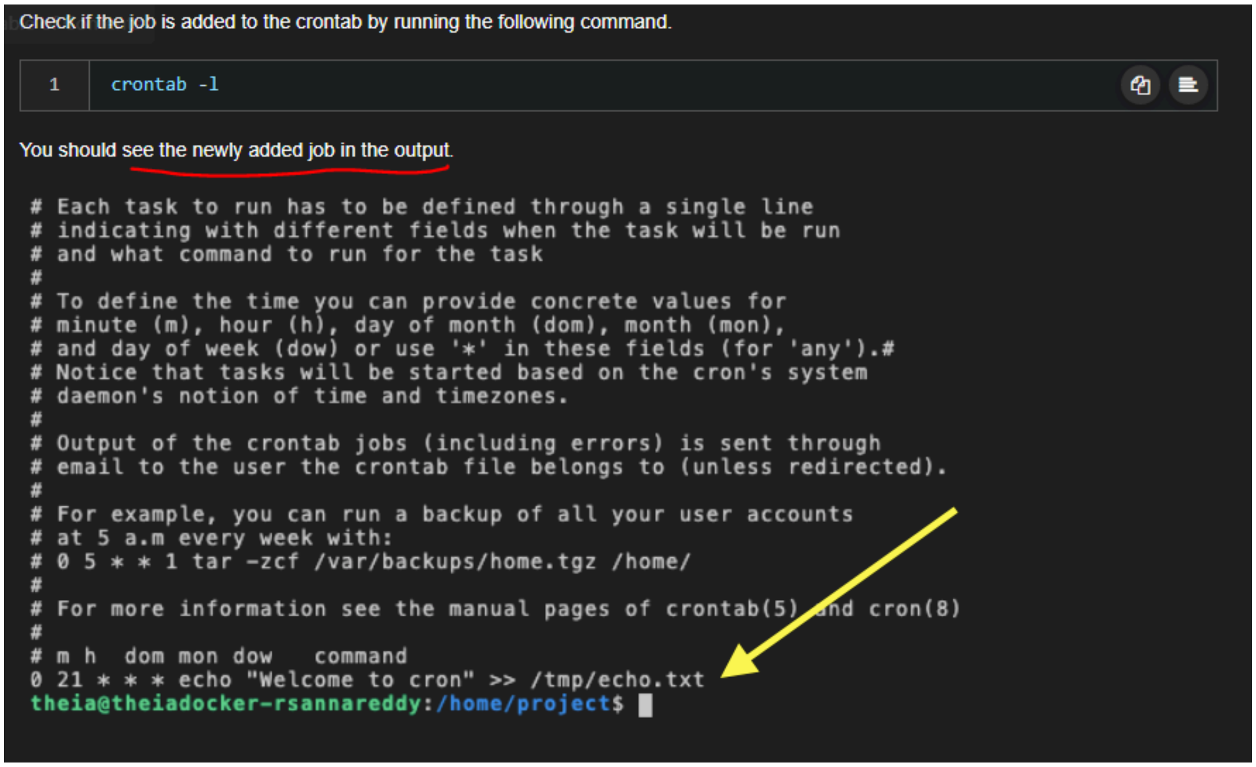
Press `Ctrl` + `x` to save the changes.

Press `y` to confirm.



```
  GNU nano 2.9.3                            /tmp/crontab.sqvoQ6/crontab

# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h  dom mon dow   command
0 21 * * * echo "Welcome to cron" >> /tmp/echo.txt


Save modified buffer?  (Answering "No" will DISCARD changes.)
 Y Yes
 N No             ^C Cancel
```

Press Enter to come out of the editor.

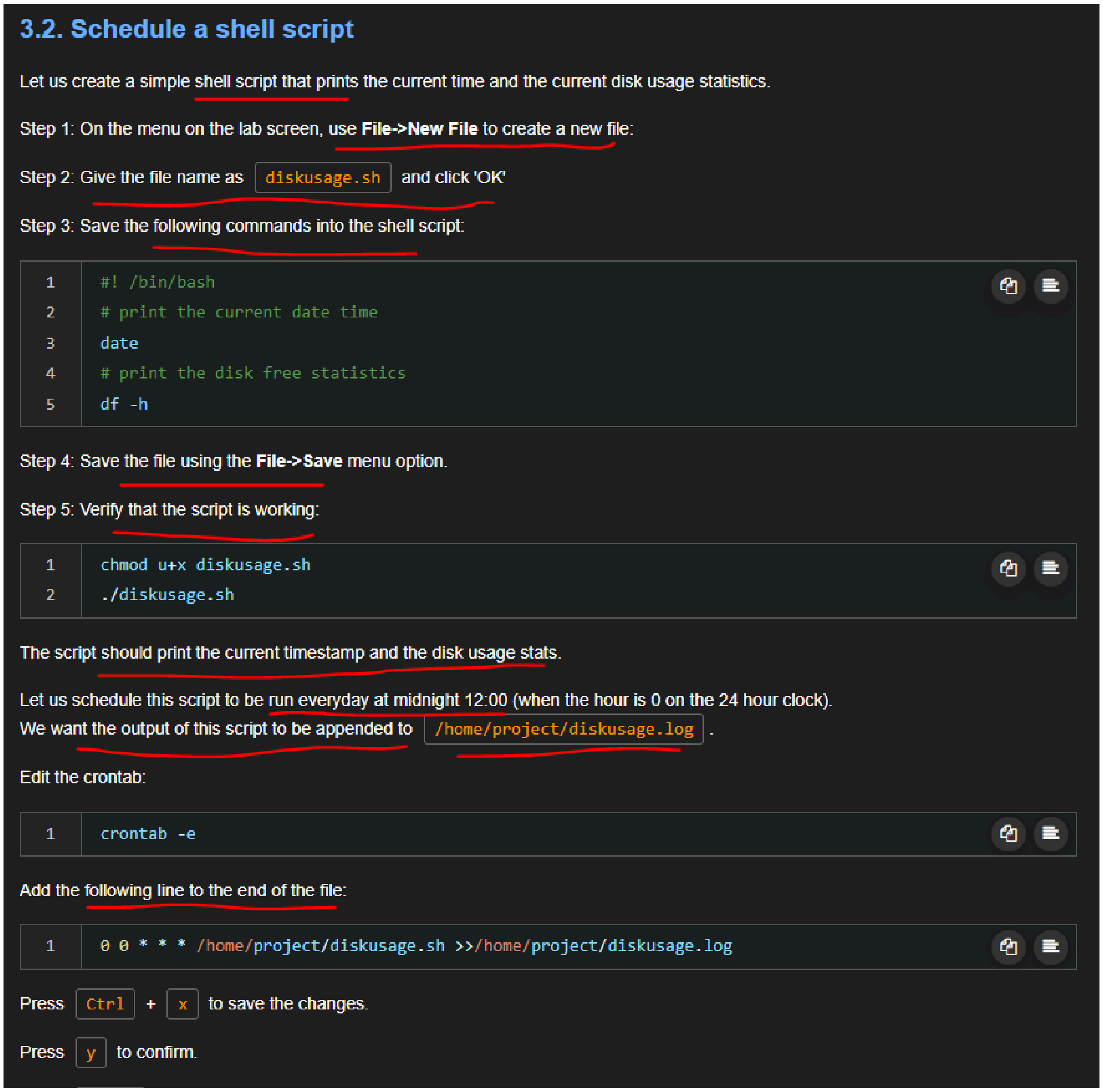


Check if the job is added to the crontab by running the following command.

```
crontab -l
```

You should see the newly added job in the output.

```
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h  dom mon dow   command
0 21 * * * echo "Welcome to cron" >> /tmp/echo.txt
theia@theiadocker-rsannareddy:/home/project$
```



3.2. Schedule a shell script

Let us create a simple shell script that prints the current time and the current disk usage statistics.

Step 1: On the menu on the lab screen, use **File->New File** to create a new file:

Step 2: Give the file name as `diskusage.sh` and click 'OK'

Step 3: Save the following commands into the shell script:

```
#! /bin/bash
# print the current date time
date
# print the disk free statistics
df -h
```

Step 4: Save the file using the **File->Save** menu option.

Step 5: Verify that the script is working:

```
chmod u+x diskusage.sh
./diskusage.sh
```

The script should print the current timestamp and the disk usage stats.

Let us schedule this script to be run everyday at midnight 12:00 (when the hour is 0 on the 24 hour clock).
We want the output of this script to be appended to `/home/project/diskusage.log` .

Edit the crontab:

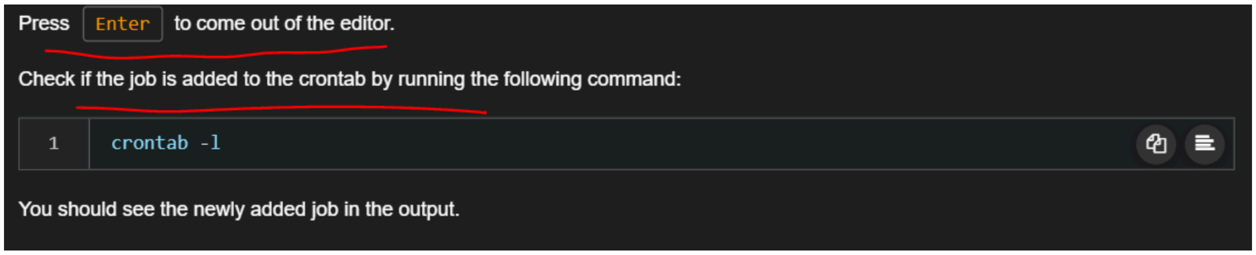
```
crontab -e
```

Add the following line to the end of the file:

```
0 0 * * * /home/project/diskusage.sh >>/home/project/diskusage.log
```

Press `Ctrl` + `x` to save the changes.

Press `y` to confirm.

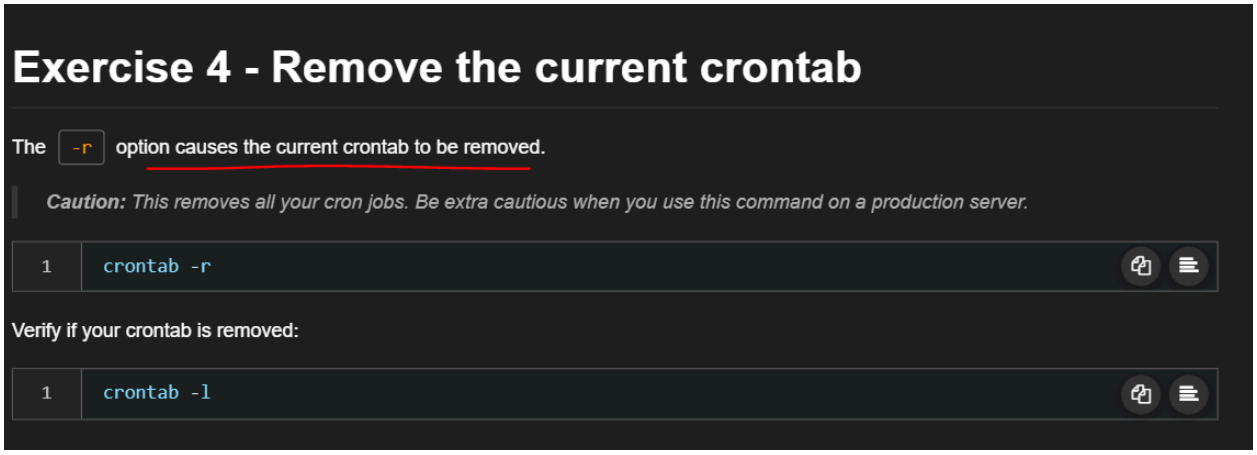


Press `Enter` to come out of the editor.

Check if the job is added to the crontab by running the following command:

```
crontab -l
```

You should see the newly added job in the output.


Exercise 4 - Remove the current crontab

The `-r` option causes the current crontab to be removed.

> ***Caution:*** *This removes all your cron jobs. Be extra cautious when you use this command on a production server.*

```
crontab -r
```

Verify if your crontab is removed:

```
crontab -l
```

Summary

In this lab, you learned how to:

- List cron jobs using `crontab -l`
- Add cron jobs using `crontab -e`
- Remove your current crontab using `crontab -r`