[How to schedule sqoop job using crontab](https://stackoverflow.com/questions/46489326/how-to-schedule-sqoop-job-using-crontab)

I have created sqoop job which is working perfectly but when I am trying to schedule it using crontab it is not working.I have scheduled it as follows,

50 12 \* \* \* sqoop job --exec myjob\_direct\_append >> /home/training/my\_Local/direct\_append1.log

This looks fine to me. Where are you executing this from ? is it a system crontab (usually /etc/crontab) ?

Try scheduling

\* \* \* \* \* sqoop job --exec myjob\_direct\_append >> /home/training/my\_Local/direct\_append1.log

Ex: You can put the shell script in /etc/cron.daily and run

1. crontab -e
2. 0 1 \* \* \* <path to script> ----> run every day at 1 AM

You already have a script created ***sample.sh*** ,Now save it in some directory like ***/home/devi/sample.sh*** make sure the file is executable then the entry in the crontab should be like below

invoke the crontab in edit mode

1. *# crontab -e*

Paste your code

1. *0 1 \* \* \* /home/devi/sample.sh ----> run every day at 1 AM*

save the file

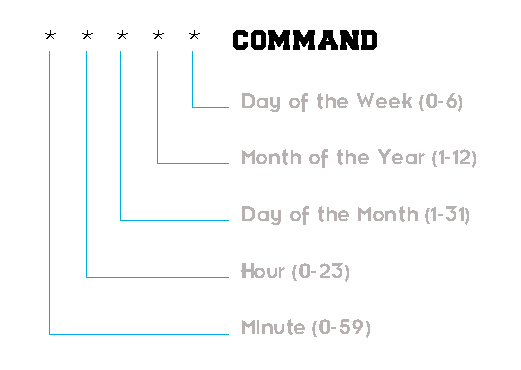
You can adjust the test by changing the time to every 2 minute

1. *\*/2 \* \* \* \* /home/devi/sample.sh ----> run every 2 minutes*

### **What is Crontab?**

In order to manipulate the job schedules, we use the Crontab program in Linux. Crontab, short for ‘cron table,‘ is a configuration file. Each line of the Crontab represents a job and contains information on what to run and when to run. The following is the format for the Linux Crontab:

M H DOM MON DOW Command



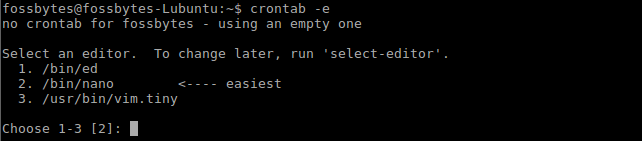
The command gets executed whenever all the time specification fields match the current date and time. More often than not, we use the asterisk (**\***) symbol in the time specification field to match any value in that field.

### **Opening and Editing Crontab**

Now, let us learn how to open and edit the Crontab file, which is an important step in Linux job scheduling. In order to do so, we use the following command:

crontab -e

After executing this command, you might be prompted to choose an editor. If you are a beginner, I would advise you to select Nano, otherwise feel free to go with any editor that you are comfortable with.



**Note:** The Hash (**#**) Symbol is used to denote comments. These comments will be ignored by Cron.

### **Scheduling Jobs**

Now in order to schedule jobs in Linux, all you need to do is enter all the necessary details while following the format mentioned above. Here is an example: Suppose I want to run the command usr/bin/backup at 2:30 AM on the first day of every month, then I will add the following line to the Crontab:

30 02 1 \* \* /usr/bin/backup

**30** : 30th Minute  
**02** : 2 AM  
**1**   : 1st Day  
**\***   : Every Month  
**\***   : Every Day of the Week

### **Specifying Multiple Value and Ranges**

It is also possible to schedule jobs in Linux to occur at multiple times. Just use a comma (**,**) to separate the required values. As an example, lets again consider the previous example. Now if i wish to execute the command at 2:30 PM as well, all i need to do is this:

30 02,14 1 \* \* /usr/bin/backup

**30**      : 30th Minute  
**02,14** : 2 AM and 2 PM  
**1**        : 1st Day  
**\***        : Every Month  
**\***        : Every Day of the Week

**Note:** We have specified 2 PM by 14 , as Crontab utilizes the 24-hour time format

What’s more, its even possible to specify a range of time within Crontab in Linux job scheduling. Just insert the values separated by a dash (**–**). Here, we will again consider our initial example for showing how it’s done. This time, suppose we wish to execute the command at every hour between 2 AM and 2 PM on the first day of every month. We will do this in the following fashion:

00 02-14 1 \* \* /usr/bin/backup

**00**: 0th Minute  
**02-14**: 2AM, 3AM, 4AM, 5AM, 6AM, 7AM, 8AM, 9AM, 10AM, 11AM, 12AM, 1PM, 2PM  
**1**        : 1st Day  
**\***        : Every Month  
**\***        : Every Day of the Week

### **Crontab Shortcuts**

As a bonus in this article on how to schedule jobs in Linux, consider the following shortcuts and keywords of the Crontab format which might come in handy to schedule jobs in Linux:

**@yearly**       : run once a year at midnight on the morning of January 1  
**@annually**   : same as @yearly  
**@monthly**   : run once a month at midnight on the morning of the first day of the month  
**@weekly**     : run once a week at midnight on the morning of Sunday  
**@daily**         : run everyday at midnight  
**@midnight**  : same as @daily  
**@hourly**      : run once a hour at the beginning of the hour

For example: Consider that I have to run the /usr/bin/backup command every month. The Crontab format for it will be:

@monthly /usr/bin/backup

Once you save the changes to the Crontab and exit, you shall see the following message indicating that you carried out the process successfully.

 used /etc/crontab. I used vi and entered in the PATHs I needed into this file and ran it as root. The normal crontab overwrites PATHs that you have set up. [A good tutorial on how to do this](https://web.archive.org/web/20150626074155/http:/unixhelp.ed.ac.uk/CGI/man-cgi?crontab+5).

The systemwide cron file looks like this:

This has the username field, as used by /etc/crontab.

# /etc/crontab: system-wide crontab

# Unlike any other crontab you don't have to run the `crontab'

# command to install the new version when you edit this file.

# This file also has a username field, that none of the other crontabs do.

SHELL=/bin/sh

PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# m h dom mon dow user command

42 6 \* \* \* root run-parts --report /etc/cron.daily

47 6 \* \* 7 root run-parts --report /etc/cron.weekly

52 6 1 \* \* root run-parts --report /etc/cron.monthly

01 01 \* \* 1-5 root python /path/to/file.py

The best way would be to get shell script that takes 2 arguments. 1 would be the name of the table and other would be hdfs path because those would be the only 2 factors that change in your scenarios. Below is sample script that you can put into .sh and run in bash.

!/bin/bash

TABLENAME=${^^1}

HDFSPATH=${^^2}

NOW=$(date +"%m-%d-%Y-%H-%M-%S")

sqoop --import --connect jdbc:db2://mystsrem:60000/SCHEMA \

--username username \

--password-file password \

--query "select \* from ${TABLENAME} \$CONDITIONS" \

-m 1 \

--delete-target-dir \

--target-dir ${HDFSPATH} \

--fetch-size 30000 \

--class-name ${TABLENAME} \

--fields-terminated-by '\01' \

--lines-terminated-by '\n' \

--escaped-by '\' \

--verbose &> logonly/${TABLENAME}\_import\_${NOW}.log