How to run a cron job inside a docker container?

Asked 8 years, 3 months ago Modified 2 months ago Viewed 762k times

543

I am trying to run a cronjob inside a docker container that invokes a shell script.

How can I do this?

docker cron containers sh

Share Follow

edited Jan 23 at 6:35
Omer Dagan
15.7k ● 16 ● 46 ● 61

asked May 26, 2016 at 10:32
C Heyer
5,433 ● 3 ● 11 ● 8

I would not recommand this at all. There are a few cases like "docker-registry" should clean up itself when its running as a container BUT! The setup is complicated and a job inside a container gives bad maintainability and eats resources. Instead: simply let your ROOT system handle the cronjobs instead. Go on your root-machine, and put your desired command into crontab using docker exec -it <container-name> <your shell cmd or script inside container> – Steini May 3 at 7:25

32 Answers

Sorted by: Highest score (default)

1 2 Next

You can copy your `crontab` into an image, in order for the container launched from said image to

CRLF for your `cron` file.

his **`Ekito/docker-cron`**:

ribe our job.

```
ot "CRLF" (Windows)
 2>&1
le for a valid cron file.
```

lains.

> The following Dockerfile describes all the steps to build your image

```
FROM ubuntu:latest
MAINTAINER docker@ekito.fr

RUN apt-get update && apt-get -y install cron

# Copy hello-cron file to the cron.d directory
COPY hello-cron /etc/cron.d/hello-cron

# Give execution rights on the cron job
RUN chmod 0644 /etc/cron.d/hello-cron

# Apply cron job
RUN crontab /etc/cron.d/hello-cron

# Create the log file to be able to run tail
RUN touch /var/log/cron.log

# Run the command on container startup
CMD cron && tail -f /var/log/cron.log
```

But: if `cron` dies, the container keeps running.

(see Gaafar's comment and How do I make `apt-get` install less noisy?: `apt-get -y install -qq --force-yes cron` can work too)

As noted by Nathan Lloyd in the comments:

> Quick note about a gotcha:
> If you're adding a script file and telling cron to run it, remember to
> `RUN chmod 0744 /the_script`

ng your tasks twice

ockerfile:

- By placing the `hello-cron` file in the `/etc/cron.d` directory, you automatically schedule the cron jobs contained in this file. The cron daemon checks this directory for any files containing cron schedules and automatically loads them.
- The `crontab` command with `/etc/cron.d/hello-cron` takes the contents of the `hello-cron` file and loads them into the main crontab. This means the same jobs are now scheduled directly in the crontab as well, effectively duplicating them.

you should choose one method to manage your cron jobs, depending on your specific needs:

- **If you prefer using `/etc/cron.d`** (often easier for managing multiple separate cron job files):

```
COPY hello-cron /etc/cron.d/hello-cron
RUN chmod 0644 /etc/cron.d/hello-cron
```

- **If you prefer using `crontab`** (gives you a consolidated view of all cron jobs and can be easier for a single or a few jobs):

```
ADD hello-cron /etc/cronjob
RUN crontab /etc/cronjob
```

OR, make sure your job itself redirect directly to stdout/stderr instead of a log file, as described in hugoShaka's answer:

```
* * * * * root echo hello > /proc/1/fd/1 2>/proc/1/fd/2
```

Replace the last Dockerfile line with

```
```

root.

und") "docker ubuntu `cron -f` is not

```
```

ne should display:

```
Hello world
Hello world
```

Eric adds in the comments:

> Do note that `tail` may not display the correct file if it is created during image build.
> If that is the case, you need to create or touch the file during container runtime in order for tail to pick up the correct file.

See "Output of `tail -f` at the end of a docker `CMD` is not showing".

See more in "**Running Cron in Docker**" (Apr. 2021) from **Jason Kulatunga**, as he commented below

See Jason's image `AnalogJ/docker-cron` based on:

- Dockerfile installing `cronie`/`crond`, depending on distribution.
- an entrypoint initializing `/etc/environment` and then calling

```
cron -f -l 2
```

Share Follow

edited Apr 18 at 13:29

answered May 26, 2016 at 10:42

VonC
1.3m ● 554 ● 4.6k ● 5.5k

ker build exiting – gafi Oct 16, 2016 at 7:41

ven, when I log into the container as root and so, my screen remains blank. However, when I more surprisingly), when I check ile content is being appended with `Hello` `sion/baseimage:0.10.0`. Any ideas about 2, 2018 at 16:30

n able to get their cronjob to work with Ubuntu hich comes with cron running out of the box

telling cron to run it, remember to `RUN chmod` an Lloyd Mar 18, 2020 at 0:42

sues I found running cron in docker) into centos): blog.thesparktree.com/cron-in-docker

272

The accepted answer ***may be dangerous in a production environment***.

> In docker you should only execute one process per container because if you don't, the process that forked and went background is not monitored and may stop without you knowing it.

When you use `CMD cron && tail -f /var/log/cron.log` the cron process basically fork in order to execute `cron` in background, the main process exits and let you execute `tailf` in foreground. The background cron process could stop or fail you won't notice, your container will still run silently and your orchestration tool will not restart it.

> You can avoid such a thing by redirecting directly the cron's commands output into your docker `stdout` and `stderr` which are located respectively in `/proc/1/fd/1` and `/proc/1/fd/2`.

Using basic shell redirects you may want to do something like this :

```
* * * * * root echo hello > /proc/1/fd/1 2>/proc/1/fd/2
```

And your CMD will be : `CMD ["cron", "-f"]`

But: this doesn't work if you want to run tasks as a non-root.

Share Follow

edited Feb 5, 2023 at 14:33

x-yuri

18.3k ● 15 ● 121 ● 172

answered Sep 14, 2017 at 13:11

hugoShaka

5,337 ● 3 ● 18 ● 31

ur answer in mine above, for more visibility. +1

e this method and be sure my process isn't
at 19:34

ground, thats why it does not fails silently.
imple. If you want to have a running
o monitor the multiple processes you run in the
ontainer next to the main one called 'sidecar'.
tainer. – hugoShaka May 22, 2018 at 14:14

issue. When the container receives a
rocess to finish and gracefully shutdown,
mes Hulse Jan 21, 2020 at 11:03

ainers. This is the most "portable" solution.

231

For those who wants to use a simple and lightweight image:

```
FROM alpine:3.6

# copy crontabs for root user
COPY config/cronjobs /etc/crontabs/root

# start crond with log level 8 in foreground, output to stderr
CMD ["crond", "-f", "-d", "8"]
```

Where *cronjobs* is the file that contains your cronjobs, in this form:

```
* * * * * echo "hello stackoverflow" >> /test_file 2>&1
# remember to end this file with an empty new line
```

But apparently you won't see `hello stackoverflow` in `docker logs`.

Share Follow

edited Feb 5, 2023 at 15:09
x-yuri
18.3k ● 15 ● 121 ● 172

answered Dec 24, 2017 at 11:22

Oscar Fanelli
3,567 ● 3 ● 31 ● 40

26 Simple, light and standard image based. This should be the accepted answer. Also use the `> /proc/1/fd/1 2> /proc/1/fd/2` redirection to access cronjobs output directly from the docker logs. – HenriTel Apr 24, 2018 at 8:24

10 For people not using alpine: The crond supporting the `-d 8` parameter is not the standard cron, it is the crond command from busybox. For example from ubuntu, you can run this as `busybox crond -f -d 8`. For older versions you have to use `-L /dev/stdout/`. – Trendfischer Apr 25, 2018 at 13:44

to run cron jobs in a Docker environment.

`mage:alpine`? – Groostav Jun 1, 2019 at

020 at 13:08

ll cron job configuration in one line. This
and you don't need a separate cron file.

```
# Setup cron job
RUN (crontab -l ; echo "* * * * * echo "Hello world" >> /var/log/cron.log") |
crontab

# Run the command on container startup
CMD cron && tail -f /var/log/cron.log
```

After running your docker container, you can make sure if cron service is working by:

```
# To check if the job is scheduled
docker exec -ti <your-container-id> bash -c "crontab -l"
# To check if the cron service is running
docker exec -ti <your-container-id> bash -c "pgrep cron"
```

If you prefer to have ENTRYPOINT instead of CMD, then you can substitute the CMD above with

```
ENTRYPOINT cron start && tail -f /var/log/cron.log
```

But: if `cron` dies, the container keeps running.

Share Follow

edited Feb 5, 2023 at 15:11
x-yuri
18.3k ● 15 ● 121 ● 172

answered Jul 6, 2017 at 20:15

Youness
2,050 ● 24 ● 29

8 `RUN apt-get update && apt-get -y install cron` or else it will not be able to find package `cron` – alphabetasoup Aug 29, 2017 at 23:08

3 Thanks Youness, you gave me the idea of doing the following, which worked in my case where each cron is specified in a different file: `RUN cat $APP_HOME/crons/* | crontab` Like a charm :) – marcostvz Nov 30, 2017 at 15:51

…tion: ENTRYPOINT ["entrypoint.sh"] – bozdoz

…elieve the first one (`cron`) forks to the …If `cron` stops, you'll never know it. If `tail` …23:43

…e monitoring/logging around it (with another …the health status of the cron service – Youness

…nner that has cron (a scheduler) support.

…our base image (python, java, nodejs, …ntain. Tasker avoid that by decoupling the …ge that you want to execute your

Here an `docker-compose.yml` file, that will run some tasks for you

```
version: "2"

services:
    tasker:
        image: strm/tasker
        volumes:
            - "/var/run/docker.sock:/var/run/docker.sock"
        environment:
            configuration: |
                logging:
                    level:
                        ROOT: WARN
                        org.springframework.web: WARN
                        sh.strm: DEBUG
                schedule:
                    - every: minute
                      task: hello
                    - every: minute
                      task: helloFromPython
                    - every: minute
                      task: helloFromNode
                tasks:
                    docker:
                        - name: hello
                          image: debian:jessie
                          script:
                              - echo Hello world from Tasker
                        - name: helloFromPython
                          image: python:3-slim
                          script:
                              - python -c 'print("Hello world from python")'
                        - name: helloFromNode
                          image: node:8
                          script:
                              - node -e 'console.log("Hello from node")'
```

e (`every: minute`), and each of them will
`image` section.

s the Tasker repo with the full

answered Sep 16, 2017 at 14:33

OPSXCQ

556 ● 5 ● 6

tainer) is a bad practice and should be limited
`cker exec` on specified containers.

1 Tasker doesn't use docker in docker (Dind/Dockerception), note that is passed the docker socket as a mapping, all containers spawned are are in spawned in the daemon that tasker runs. And if you don't want to run tasker inside docker, you can just deploy it as any other application. – OPSXCQ Apr 27, 2018 at 22:17

2 I don't get the advantages of using tasker. Seems really a overkill to me using java and sh*** just to run a cron job. – Karl Adler Oct 30, 2018 at 14:59

Mixing cron and the base image that you need (python/node for example) create a extra dependency that need to be maintained and deployed, in this scenario all jobs share the same container, it means that you have to worry about cleaning up everything after every job runs. Jobs running on tasker are idempotent, so you have less things to worry about. – OPSXCQ Jan 10, 2019 at 13:59

20

Though this aims to run jobs beside a running process in a container via Docker's `exec` interface, this may be of interest for you.

I've written a daemon that observes containers and schedules jobs, defined in their metadata, on them. Example:

```
version: '2'

services:
  wordpress:
    image: wordpress
  mysql:
    image: mariadb
    volumes:
      - ./database_dumps:/dumps
    labels:
      deck-chores.dump.command: sh -c "mysqldump --all-databases >
/dumps/dump-$$(date -Idate)"
      deck-chores.dump.interval: daily
```


answered Dec 16, 2016 at 14:52

funky-future

3,917 ● 1 ● 33 ● 43

ers environment. No any changes in Docker
works like command `docker exec`
HLOP Oct 6, 2019 at 10:25

paradigm of containers. Getting rid of crutches.

15

If you're using docker for windows, remember that you have to change your line-ending format from CRLF to LF (i.e. from dos to unix) if you intend on importing your crontab file from windows to your ubuntu container. If not, your cron-job won't work. Here's a working example:

```
FROM ubuntu:latest

RUN apt-get update && apt-get -y install cron
RUN apt-get update && apt-get install -y dos2unix

# Add crontab file (from your windows host) to the cron directory
ADD cron/hello-cron /etc/cron.d/hello-cron

# Change line ending format to LF
RUN dos2unix /etc/cron.d/hello-cron

# Give execution rights on the cron job
RUN chmod 0644 /etc/cron.d/hello-cron

# Apply cron job
RUN crontab /etc/cron.d/hello-cron

# Create the log file to be able to run tail
RUN touch /var/log/hello-cron.log

# Run the command on container startup
CMD cron && tail -f /var/log/hello-cron.log
```

This actually took me hours to figure out, as debugging cron jobs in docker containers is a tedious task. Hope it helps anyone else out there that can't get their code to work!

But: if `cron` dies, the container keeps running.

Share Follow

edited Feb 5, 2023 at 15:19

121 • 172

answered Dec 19, 2019 at 16:28

Andre

Andreas Forslöw

2,588 • 1 • 27 • 37

tion to work. A command like `cat`
rom crond stating `crond: USER root pid`
`nexistent directory/proc/1/fd/1`.
mmand successfully. Thanks, this took me
2020 at 13:06

dd one thing that helped me. If you just
pted to just remove the `&& tail -f`

rtly after running because when the cron
as exited and hence kills the container.
This can be avoided by running cron in the foreground via `cron -f`.

Share Follow

edited May 24, 2019 at 11:24

halfer

20.3k ● 19 ● 106 ● 197

answered Jun 20, 2017 at 1:54

vanugrah

131 ● 1 ● 4

13

Unfortunately, none of the above answers worked for me, although all answers lead to the solution and eventually to my solution, here is the snippet if it helps someone. Thanks

This can be solved with the bash file, due to the layered architecture of the Docker, cron service doesn't get initiated with RUN/CMD/ENTRYPOINT commands.

Simply add a bash file which will initiate the cron and other services (if required)

DockerFile

```
FROM gradle:6.5.1-jdk11 AS build
# apt
RUN apt-get update
RUN apt-get -y install cron
# Setup cron to run every minute to print (you can add/update your cron here)
RUN touch /var/log/cron-1.log
RUN (crontab -l ; echo "* * * * * echo testing cron.... >> /var/log/cron-1.log
2>&1") | crontab
# entrypoint.sh
RUN chmod +x entrypoint.sh
CMD ["bash","entrypoint.sh"]
```

entrypoint.sh

```
#!/bin/sh
service cron start & tail -f /var/log/cron-2.log
```


…n then add that service with `&` in the

…alone.sh & service cron start & tail -f

…n see that `testing cron....` will be

…g

…15:21

…121 ● 172

answered Feb 23, 2021 at 17:41

Gaurav Tyagi

679 ● 7 ● 8

2 Shouldn't it be doing `tail -f /var/log/cron-1.log` instead of `/var/log/cron-2.log`, since `cron-1.log` is where the STDOUT/STDERR is being directed? (Unless I'm missing something) – Peter Jul 4, 2021 at 5:05

Yes, correct, that was a typo, `/var/log/cron-1.log` should be at every place – Gaurav Tyagi Jul 5, 2021 at 8:06

10

I created a Docker image based on the other answers, which can be used like

```
docker run -v "/path/to/cron:/etc/cron.d/crontab" gaafar/cron
```

where `/path/to/cron`: absolute path to crontab file, or you can use it as a base in a Dockerfile:

```
FROM gaafar/cron

# COPY crontab file in the cron directory
COPY crontab /etc/cron.d/crontab

# Add your commands here
```

For reference, the image is here.

Share Follow

edited Feb 10, 2022 at 15:30
vvvvv
30k 19 57 97

answered Oct 19, 2016 at 12:48

gafi
12.6k 2 31 33

I occasionally tried to find a `docker`-friendly `cron` implementation. And this last time I tried, I've

seen in `docker logs` w/o resorting to

. It can be fed a crontab file, all while

```
FROM alpine:3.17
RUN set -x \
    && apk add --no-cache supercronic shadow \
    && useradd -m app
USER app
COPY crontab .
```

`crontab`:

```
* * * * * date
```

A gist with a bit more info.

Another good one is `yacron`, but it uses YAML.

`ofelia` can be used, but they seem to focus on running tasks in separate containers. Which is probably not a downside, but I'm not sure why I'd want to do that.

And there's also a number of traditional `cron` implementations: `dcron`, `fcron`, `cronie`. But they come with "no easy way to see output of the tasks."

Share Follow

answered Feb 5, 2023 at 16:05

x-yuri
18.3k ● 15 ● 121 ● 172

A good alternative to my answer indeed. – VonC Feb 5, 2023 at 16:55

i just tested supercronic and it works as expected. i will now test yacron as the additional configuration options are a very nice addon in my usecase! – Stefan Krüger s-light Jun 25 at 8:01

ages.

tderr (-d), I didn't choose to change the
ool/cron/crontabs

```
-c DIR]
```

```
8
```

```
ntabs
```

```
CMD [ "crond", "-f", "-d" ]
```

But output of the tasks apparently can't be seen in `docker logs`.

Share Follow

edited Feb 5, 2023 at 15:23
x-yuri
18.3k ● 15 ● 121 ● 172

answered Jun 18, 2021 at 14:57
Iljanne
71 ● 1 ● 2

The `-d` parameter requires the log level as argument. You should change your `CMD` line to: `CMD [ "crond", "-f", "-d", "8" ]` – Daniel Jan 14, 2023 at 12:45

6

Define the cronjob in a dedicated container which runs the command via docker exec to your service.

This is higher cohesion and the running script will have access to the environment variables you have defined for your service.

```
#docker-compose.yml
version: "3.3"
services:
    myservice:
      environment:
        MSG: i'm being cronjobbed, every minute!
      image: alpine
      container_name: myservice
      command: tail -f /dev/null

    cronjobber:
     image: docker:edge
     volumes:
```

```
rintenv | grep MSG' >
```


answered Feb 4, 2018 at 17:40
Jakob Eriksson
18.7k ● 1 ● 27 ● 34

`myservice unknown` errors. – Mark Grimes

ounting a docker socket has: lvh.io/posts/…

5

When you deploy your container on another host, just note that it won't start any processes automatically. You need to make sure that 'cron' service is running inside your container. In our case, I am using Supervisord with other services to start cron service.

```
[program:misc]
command=/etc/init.d/cron restart
user=root
autostart=true
autorestart=true
stderr_logfile=/var/log/misc-cron.err.log
stdout_logfile=/var/log/misc-cron.out.log
priority=998
```

Share Follow

edited Dec 16, 2016 at 7:08

Priya

1,357 ● 6 ● 21 ● 41

answered Dec 16, 2016 at 5:48

Sagar Ghuge

231 ● 3 ● 9

I get an error in supervisor.log that the cron service stopped multiple times and entered a FATAL state. However cron seems to be running in top and executing cronjobs normally. Thanks for this! – lephleg Jan 14, 2017 at 23:13

Yes, same thing happened to me as well, but it works as normal, so don't need to bother. – Sagar Ghuge Feb 6, 2017 at 10:42

5

From above examples I created this combination:

Alpine Image & Edit Using Crontab in Nano (I hate vi)

```
FROM alpine

RUN apk update
RUN apk add curl nano

ENV EDITOR=/usr/bin/nano

# start crond with log level 8 in foreground, output to stderr
CMD ["crond", "-f", "-d", "8"]

# Shell Access
# docker exec -it <CONTAINERID> /bin/sh

# Example Cron Entry
# crontab -e
# * * * * * echo hello > /proc/1/fd/1 2>/proc/1/fd/2
# DATE/TIME WILL BE IN UTC
```

Share Follow

answered Apr 22, 2020 at 1:54

Dan Watts

109 ● 1 ● 3

5

Setup a cron in parallel to a one-time job

Create a script file, say run.sh, with the job that is supposed to run periodically.

```
#!/bin/bash
```


erization, use the entrypoint file to run

a docker run command is issued. So, all

e.

d”

Run periodic job: run.sh

Create entrypoint.sh

```
#!/bin/bash

# Start the run once job.
echo "Docker container has been started"

# Setup a cron schedule
echo "* * * * * /run.sh >> /var/log/cron.log 2>&1
# This extra line makes it a valid cron" > scheduler.txt

crontab scheduler.txt
cron -f
```

Let's understand the crontab that has been set up in the file

`* * * * *`: Cron schedule; the job must run every minute. You can update the schedule based on your requirement.

`/run.sh`: The path to the script file which is to be run periodically

`/var/log/cron.log`: The filename to save the output of the scheduled cron job.

`2>&1`: The error logs(if any) also will be redirected to the same output file used above.

Note: Do not forget to add an extra new line, as it makes it a valid cron. `Scheduler.txt`: the complete cron setup will be redirected to a file.

Using System/User specific environment variables in cron

s as the environment variables passed to
to use any of the environment variables

```
)|PPID|SHELLOPTS|UID' >
```

At last, your `entrypoint.sh` should look like

```
#!/bin/bash

# Start the run once job.
echo "Docker container has been started"

declare -p | grep -Ev 'BASHOPTS|BASH_VERSINFO|EUID|PPID|SHELLOPTS|UID' >
/container.env

# Setup a cron schedule
echo "SHELL=/bin/bash
BASH_ENV=/container.env
* * * * * /run.sh >> /var/log/cron.log 2>&1
# This extra line makes it a valid cron" > scheduler.txt

crontab scheduler.txt
cron -f
```

Last but not the least: Create a Dockerfile

```
FROM ubuntu:16.04
MAINTAINER Himanshu Gupta

# Install cron
RUN apt-get update && apt-get install -y cron

# Add files
ADD run.sh /run.sh
ADD entrypoint.sh /entrypoint.sh

RUN chmod +x /run.sh /entrypoint.sh

ENTRYPOINT /entrypoint.sh
```


t 10:27

:96 ● 344

answered May 16, 2020 at 11:05

himanshuIIITian

6,075 ● 6 ● 52 ● 70

ne "run once job" is never returning and also
y ideas? thanks – Doron Levi May 19, 2020 at

he issue? Or you can check the whole code
nshuIIITian May 19, 2020 at 10:28

nd generally the description/solution seems too

5

Here's my `docker-compose` based solution:

```
cron:
  image: alpine:3.10
  command: crond -f -d 8
  depends_on:
    - servicename
  volumes:
    - './conf/cron:/etc/crontabs/root:z'
  restart: unless-stopped
```

the lines with cron entries are on the `./conf/cron` file.

Note: this won't run commands that aren't in the `alpine` image.

Also, output of the tasks apparently won't appear in `docker logs`.

Share Follow

edited Feb 5, 2023 at 15:28

x-yuri

18.3k ● 15 ● 121 ● 172

answered Jul 13, 2020 at 14:12

That Brazilian Guy

3,491 ● 6 ● 32 ● 51

4

This question have a lot of answers, but some are complicated and another has some drawbacks. I try to explain the problems and try to deliver a solution.

`cron-entrypoint.sh`:

```
#!/bin/bash

# copy machine environment variables to cron environment
printenv | cat - /etc/crontab > temp && mv temp /etc/crontab
```


vironment (like env vars or kubernetes

- stop gracefully cron jobs when machine receive an SIGTERM signal

For context, I use previous script on Kubernetes with Laravel app.

Share Follow

answered Sep 20, 2021 at 14:10

pablorsk
4,198 ● 1 ● 33 ● 38

If I run `docker stop` with this setup, nothing happens, i.e. `service cron stop` doesn't get executed. If I run the latter manually from within the container, the `cron` process stops immediately instead of waiting for the cronjobs. cronjobs will still complete their run, so that may be fine. When they are done, the container does not stop either though. What am I missing? – Johnson_145 Jun 17, 2022 at 14:19

Got it working now. I think the trap handler wasn't triggered, because I defined my entryscript as `CMD "/bin/sh" ENTRYPOINT /entrypoint.sh` instead of `ENTRYPOINT ["/entrypoint.sh"]`. That way, it got wrapped in another shell which didn't pass the signals through. I had to do some further steps to actually wait for running cronjobs to finish. Elaborating on your answer over here. – Johnson_145 Jun 23, 2022 at 19:14

3

This code has successfully worked for me. I placed the script.sh file inside the project folder, and both the script.sh and main.py Docker files are located at the same directory level. I made the following modifications to the Docker file in order to execute the Cron-job within the Docker container.

Script.sh file

```
#!/bin/bash
# script.sh

sleep 0
```

```
RUN chmod 0644 /root/script.sh
RUN apt-get update
RUN apt-get -y install cron
RUN crontab -l | { cat; echo "30 2 * * * bash /root/script.sh"; } | crontab -

# Tell uvicorn to start spin up our code, which will be running inside the
container now
CMD ["bash", "-c", "cron && uvicorn main:app --host 0.0.0.0 --port 8081"]
```

Share Follow

answered Sep 2, 2023 at 7:39

Buddhika Lakshan

320 ● 4 ● 14

2

With multiple jobs and various dependencies like `zsh` and `curl`, this is a good approach while also combining the best practices from other answers. Bonus: This does NOT require you to set `+x` execution permissions on `myScript.sh`, which can be easy to miss in a new environment.

`cron.dockerfile`

```
FROM ubuntu:latest

# Install dependencies
RUN apt-get update && apt-get -y install \
  cron \
  zsh \
  curl;

# Setup multiple jobs with zsh and redirect outputs to docker logs
RUN (echo "\
* * * * * zsh -c 'echo "Hello World"' 1> /proc/1/fd/1 2>/proc/1/fd/2 \n\
* * * * * zsh /myScript.sh 1> /proc/1/fd/1 2>/proc/1/fd/2 \n\
") | crontab

                                                    is running
```


`ron` when you change contents of the
eflected right away as it's mounted in
compose.

Share Follow

answered Dec 15, 2022 at 22:42

Cwista

359 ● 1 ● 11

2

this line was the one that helped me run my pre-scheduled task.

```
ADD mycron/root /etc/cron.d/root

RUN chmod 0644 /etc/cron.d/root

RUN crontab /etc/cron.d/root

RUN touch /var/log/cron.log

CMD ( cron -f -l 8 & ) && apache2-foreground # <-- run cron
```

--> My project run inside: **FROM php:7.2-apache**

But: if `cron` dies, the container keeps running.

Share Follow

edited Feb 5, 2023 at 15:34

x-yuri

18.3k ● 15 ● 121 ● 172

answered Jul 20, 2020 at 17:30

Santiago Vasquez

147 ● 1 ● 10

1

So, my problem was the same. The fix was to change the command section in the `docker-compose.yml`.

From

command: crontab /etc/crontab && tail -f /etc/crontab

To

command: crontab /etc/crontab

command: tail -f /etc/crontab

The **problem** was the '&&' between the commands. After deleting this, it was all fine.

Share Follow

edited Feb 21, 2019 at 11:28

answered Feb 20, 2019 at 19:36

random2137

138 ● 1 ● 2 ● 15

The second command overrides the first one. That is, having 2 command keys equals to having only the last one (the last one wins). – x-yuri Feb 5, 2023 at 15:40

1

Focusing on *gracefully* stopping the cronjobs when receiving `SIGTERM` or `SIGQUIT` signals (e.g. when running `docker stop`).

That's not too easy. By default, the cron process just got killed without paying attention to running cronjobs. I'm elaborating on pablorsk's answer:


```
_without_log.sh

RUN crontab /etc/cron.d/cronjobs
```

```
# to gain access to environment variables, we need this additional entrypoint
script
COPY entrypoint.sh /entrypoint.sh
RUN chmod +x /entrypoint.sh

# optionally, change received signal from SIGTERM TO SIGQUIT
#STOPSIGNAL SIGQUIT

# Run the command on container startup
ENTRYPOINT ["/entrypoint.sh"]
```

`entrypoint.sh`:

```
#!/bin/bash
# make global environment variables available within crond, too
printenv | grep -v "no_proxy" >> /etc/environment

# SIGQUIT/SIGTERM-handler
term_handler() {
  echo 'stopping cron'
  service cron stop
  echo 'stopped'
  echo 'waiting'
  x=$(($(ps u -C run_cronjob.sh | wc -l)-1))
  xold=0
  while [ "$x" -gt 0 ]
  do
    if [ "$x" != "$xold" ]; then
      echo "Waiting for $x running cronjob(s):"
      ps u -C run_cronjob.sh
      xold=$x
      sleep 1
    fi
    x=$(($(ps u -C run_cronjob.sh | wc -l)-1))
  done
  echo 'done waiting'
```

Assuming you wrap all your cronjobs in a `run_cronjob.sh` script. That way, you can execute arbitrary code for which shutdown will wait gracefully.

`run_cronjobs.sh` (optional helper script to keep cronjob definitions clean)

```
#!/bin/bash

DIR_INCL="${BASH_SOURCE%/*}"
if [[ ! -d "$DIR_INCL" ]]; then DIR_INCL="$PWD"; fi
cd "$DIR_INCL"

# redirect all cronjob output to docker
./run_cronjob_without_log.sh "$@" > /proc/1/fd/1 2>/proc/1/fd/2
```

`run_cronjob_without_log.sh`

```
your_actual_cronjob_src()
```

Btw, when receiving a `SIGKILL` the container still shut downs immediately. That way you can use a command like `docker-compose stop -t 60 cron-container` to wait 60s for cronjobs to finish gracefully, but still terminate them for sure after the timeout.

Share Follow

edited Jun 23, 2022 at 19:54

answered Jun 23, 2022 at 19:10

Johnson_145

2,024 ● 1 ● 20 ● 28

All the answers require root access inside the container because 'cron' itself requests for UID 0. To request root acces (e.g. via sudo) is against docker best practices. I used https://github.com/gicarneiro/yacron to manage scheduled tasks.

answered Dec 19, 2022 at 17:37

Andreas Wittig

176 ● 8

ict root access, I had to add my user to

```
# Allow node user to start cron daemon with sudo
```

```
RUN echo 'node ALL=NOPASSWD: /usr/sbin/cron' >>/etc/sudoers

ENTRYPOINT sudo cron && tail -f /var/log/cron.log
```

Maybe that helps someone

But: if `cron` dies, the container keeps running.

Share Follow

edited Feb 5, 2023 at 15:35
x-yuri
18.3k ● 15 ● 121 ● 172

answered Nov 28, 2017 at 15:05

Senica Gonzalez
8,154 ● 16 ● 68 ● 110

I believe the node image uses the node user; so maybe you needed to add permissions for that user – bozdoz May 8, 2020 at 0:44

1

As a quick workaround for tasks that simply need to be executed in regular intervals you could also use the `HEALTHCHECK` instruction. The health status then shows the most recent result.

```
HEALTHCHECK --interval=60m --timeout=5s command || exit 1
```

- Documentation
- syntax for docker-compose.yml

Share Follow

answered Jul 4 at 7:33
chris_cm
144 ● 13

he container (under root user) alongside
rfile with `start.sh` script what includes

answered Sep 6, 2022 at 16:16


Vitali Khvatkov
1

0

If your image doesn't contain any daemon (so it's only the short-running script or process), you may also consider starting your cron from **outside**, by simply defining a LABEL with the cron information, plus the scheduler itself. This way, your default container state is "exited". If you have multiple scripts, this may result in a lower footprint on your system than having multiple parallel-running cron instances.

See: https://github.com/JaciBrunning/docker-cron-label

Example docker-compose.yaml:

```
version: '3.8'

# Example application of the cron image
services:
  cron:
    image: jaci/cron-label:latest
    volumes:
      - "/var/run/docker.sock:/var/run/docker.sock"
      - "/etc/localtime:/etc/localtime:ro"

  hello:
    image: hello-world
    restart: "no"
    labels:
      - "cron.schedule=* * * * * "
```

Share Follow

edited Sep 13, 2022 at 9:51

answered Sep 13, 2022 at 9:29


Daniel Alder
5,265 ● 2 ● 49 ● 56

e of these other suggestions that I found
with an environment variable and ended
oint.sh, but before the call to cron -f

```
fd/2" >> /etc/cron.d/restart-
```

This removes any existing cron files, creates a new cronfile using an ENV variable of crondef, and then loads it.

Share Follow

answered Nov 3, 2022 at 6:09

Christopher Richmond
646 ● 5 ● 12

0

Our's was a nodejs application to be run as cron job and it was also dependent on environment variables.

The below solution worked for us.

Docker file:

```
# syntax=docker/dockerfile:1
FROM node:12.18.1
ENV NODE_ENV=production

COPY ["startup.sh", "./"]

# Removed steps to build the node js application

#--------------- Setup cron -----------------
# Install Cron
RUN apt-get update
RUN apt-get -y install cron

# Run every day at 1AM
#/proc/1/fd/1 2>/proc/1/fd/2 is used to redirect cron logs to standard output
and standard error
RUN (crontab -l ; echo "0 1 * * * /usr/local/bin/node /app/dist/index.js  >
/proc/1/fd/1 2>/proc/1/fd/2") | crontab
```



```
o that it is available for
```

answered Nov 5, 2022 at 17:03

Baga
1,422 ● 15 ● 24

1 2 Next

Highly active question. Earn 10 reputation (not counting the association bonus) in order to answer this question. The reputation requirement helps protect this question from spam and non-answer activity.