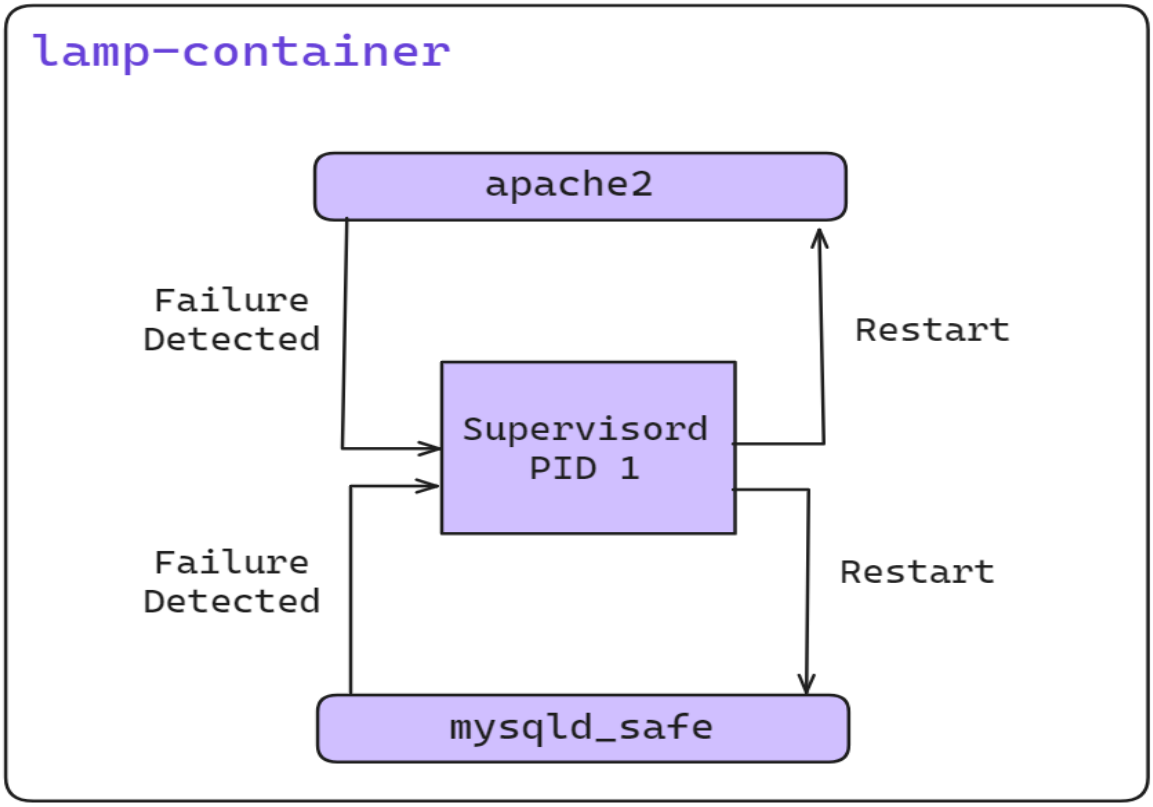
Keeping Containers Running with Supervisor

A supervisor process, or init process, is a program that's used to launch and maintain the state of other programs. On a Linux system, PID #1 is an init process. It starts all the other system processes and restarts them in the event that they fail unexpectedly. This concept can be effectively applied inside containers to start and manage processes.

Using a supervisor process inside your container ensures that the container remains operational even if the main process—such as a web server—fails and needs to be restarted. Several programs can serve as supervisor processes inside a container. The most popular ones include init, systemd, runit, upstart, and supervisord.

Example: Using Supervisord in a Container

Suppose a company provides software that produces a full LAMP (Linux, Apache, MySQL, PHP) stack inside a single container. These containers use supervisord to ensure that all the related processes are kept running. Below is an example of how to use such a container.



Create a LAMP Docker Image

Creating a LAMP (Linux, Apache, MySQL, PHP) stack image using supervisord allows you to manage multiple processes within a single Docker container. Here's a step-by-step guide to create such an image:

Step 1: Dockerfile Setup

Create a Dockerfile to define your LAMP stack image. Here, we'll use supervisord to manage Apache and MySQL processes.

[Docker file] [docker code ase ai folder]

Step 2: Supervisord Configuration

Create a supervisord.conf file in the same directory as your Dockerfile:

[supervisord]

nodaemon=true

[program:apache2]

command=/usr/sbin/apache2ctl -D FOREGROUND

[program:mysql]

command=/usr/bin/mysqld_safe

Step 3: Build and Run the Docker Image

Now, build your Docker image using the Dockerfile:

docker build -t my-lamp-image .

Starting the Container

And finally, run a container using your newly created image:

docker run -d -p 80:80 -p 3306:3306 --name lamp-container my-lamp-image

```
term@ubuntu-yl3ptw-d755bd6f5-ghgkj:~$ docker images
REPOSITORY      TAG       IMAGE ID       CREATED          SIZE
my-lamp-image   latest    eed7e197b80f   23 minutes ago   774MB
term@ubuntu-yl3ptw-d755bd6f5-ghgkj:~$ docker run -d -p 80:80 -p 3306:3306 --name lamp-container my-lamp-image
91ccee7cd127cd70bfeceddbfbb58399c45fd6fc10b63cdba94a3fedaed83349
term@ubuntu-yl3ptw-d755bd6f5-ghgkj:~$ docker ps
CONTAINER ID   IMAGE           COMMAND                  CREATED         STATUS         PORTS
                                                NAMES
91ccee7cd127   my-lamp-image   "/usr/bin/supervisor…"   9 seconds ago   Up 6 seconds   0.0.0.0:80->80/tcp, :::80->80/
tcp, 0.0.0.0:3306->3306/tcp, :::3306->3306/tcp   lamp-container
term@ubuntu-yl3ptw-d755bd6f5-ghgkj:~$
```

Checking Running Processes

You can see what processes are running inside this container by using the docker top command:

docker top lamp-container

The top subcommand will show the host PID for each of the processes in the container. You'll see supervisord, mysql, and apache included in the list of running programs.

```
term@ubuntu-yl3ptw-d755bd6f5-ghgkj:~$ docker top lamp-container
UID                 PID                 PPID                C                   STIME               TTY
   TIME                CMD
root                15775               15756               0                   16:11               ?
   00:00:00            /usr/bin/python3 /usr/bin/supervisord -c /etc/supervisor/conf.d/supervisord.conf
root                15805               15775               0                   16:11               ?
   00:00:00            /bin/sh /usr/sbin/apache2ctl -D FOREGROUND
root                15806               15775               0                   16:11               ?
   00:00:00            /bin/sh /usr/bin/mysqld_safe
root                15811               15805               0                   16:11               ?
   00:00:00            /usr/sbin/apache2 -D FOREGROUND
www-data            15904               15811               0                   16:11               ?
   00:00:00            /usr/sbin/apache2 -D FOREGROUND
www-data            15907               15811               0                   16:11               ?
   00:00:00            /usr/sbin/apache2 -D FOREGROUND
www-data            15909               15811               0                   16:11               ?
   00:00:00            /usr/sbin/apache2 -D FOREGROUND
www-data            15910               15811               0                   16:11               ?
   00:00:00            /usr/sbin/apache2 -D FOREGROUND
www-data            15912               15811               0                   16:11               ?
   00:00:00            /usr/sbin/apache2 -D FOREGROUND
_apt                15955               15806               0                   16:11               ?
   00:00:02            /usr/sbin/mysqld --basedir=/usr --datadir=/var/lib/mysql --plugin-dir=/usr/lib/mysql/plugin --
user=mysql --log-error=/var/log/mysql/error.log --pid-file=91ccee7cd127.pid
term@ubuntu-yl3ptw-d755bd6f5-ghgkj:~$
```

Stopping a Process Inside the Container

Now that the container is running, you can test the supervisord restart functionality by manually stopping one of the processes inside the container.

To kill a process inside a container from within that container, you need to know the PID in the container's PID namespace. To get that list, run the following exec subcommand:

docker exec lamp-container ps

The process list generated will have apache2 listed in the CMD column:

PID TTY TIME CMD

1 ? 00:00:00 supervisord

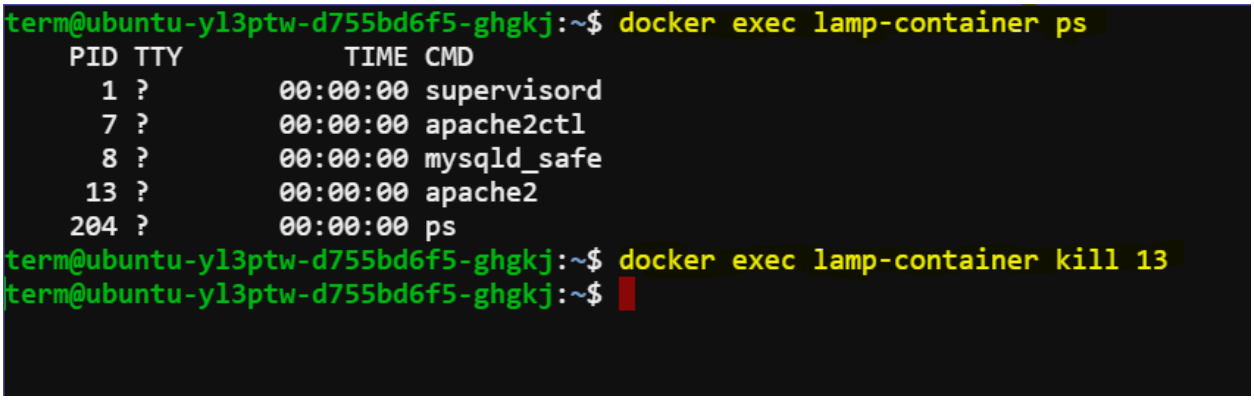
433 ? 00:00:00 mysqld_safe

835 ? 00:00:00 apache2

842 ? 00:00:00 ps

The values in the PID column will be different when you run the command. Find the PID on the row for apache2 and then insert that for <PID> in the following command:

docker exec lamp-container kill <PID>



Running this command will execute the Linux kill program inside the lamp-container container and tell the apache2 process to shut down. When apache2 stops, the supervisord process will log the event and restart the process. The container logs will clearly show these events:

... exited: apache2 (exit status 0; expected)

... spawned: 'apache2' with pid 820

... success: apache2 entered RUNNING state, process has stayed up for > than 1 seconds (startsecs)

```
term@ubuntu-yl3ptw-d755bd6f5-ghgkj:~$ docker logs lamp-container
2024-06-25 16:11:17,000 CRIT Supervisor is running as root.  Privileges were not dropped because no user is specified
 in the config file.  If you intend to run as root, you can set user=root in the config file to avoid this message.
2024-06-25 16:11:17,003 INFO supervisord started with pid 1
2024-06-25 16:11:18,006 INFO spawned: 'apache2' with pid 7
2024-06-25 16:11:18,008 INFO spawned: 'mysql' with pid 8
2024-06-25 16:11:19,156 INFO success: apache2 entered RUNNING state, process has stayed up for > than 1 seconds (star
tsecs)
2024-06-25 16:11:19,156 INFO success: mysql entered RUNNING state, process has stayed up for > than 1 seconds (starts
ecs)
2024-06-25 16:17:56,603 WARN exited: apache2 (terminated by SIGTERM; not expected)
2024-06-25 16:17:56,606 INFO spawned: 'apache2' with pid 216
2024-06-25 16:17:56,636 INFO reaped unknown pid 13 (exit status 0)
2024-06-25 16:17:57,638 INFO success: apache2 entered RUNNING state, process has stayed up for > than 1 seconds (star
tsecs)
term@ubuntu-yl3ptw-d755bd6f5-ghgkj:~$
```

This ensures that the apache2 process is restarted by supervisord, thereby maintaining the container's functionality.

By using a supervisor process inside containers, you can manage and monitor the state of your applications, ensuring high availability and reliability of your services.