Docker Administration

Mohanraj Shanmugam

Auto Starting Docker Container

- Restart Policies are the built-in Docker mechanism for restarting containers when they exit.
- If set, restart policies will be used when the Docker daemon starts up, as typically happens after a system boot.
- Using the `--restart` flag on Docker run you can specify a restart policy for how a container should or should not be restarted on exit.
- When a restart policy is active on a container, it will be shown as either `Up` or `Restarting` in `docker ps`. It can also be useful to use `docker events` to see the restart policy in effect.

Restart Policy

Policy	Result
no	Do not automatically restart the container when it exits. This is the default.
on-failure[:max-retries]	Restart only if the container exits with a non-zero exit status. Optionally, limit the number of restart retries the Docker daemon attempts.
always	Always restart the container regardless of the exit status. When you specify always, the Docker daemon will try to restart the container indefinitely. The container will also always start on daemon startup, regardless of the current state of the container.
unless-stopped	Always restart the container regardless of the exit status, but do not start it on daemon startup if the container has been put to a stopped state before.

- This means the daemon will wait for 100 ms, then 200 ms, 400, 800, 1600, and so on until either the `on-failure` limit is hit, or when you `docker stop` or`docker rm -f` the container.
- If a container is successfully restarted (the container is started and runs for at least 10 seconds), the delay is reset to its default value of 100 ms.

- Number of Restart of the container
- $ docker inspect -f "{{ .RestartCount }}" my-container
- *# 2*
- Time of last restart
- $ docker inspect -f "{{ .State.StartedAt }}" my-container
- *# 2015-03-04T23:47:07.691840179Z*
- Restart with always set
- $ docker **run** --restart=always centos
- $ docker **run** --restart=on-failure:10 centos

Exit Status

- ***125*** if the error is with Docker daemon ***itself***
- ***126*** if the ***contained command*** cannot be invoked
- ***127*** if the ***contained command*** cannot be found
- ***Exit code*** of ***contained command*** otherwise

```
$ docker run ubuntu /bin/sh -c 'exit 3'
# 3
```

Using a process manager

- If restart policies don’t suit your needs (i.e., you have non-Docker processes that depend on Docker containers), you can use a process manager like upstart, systemd or supervisor instead.
- Docker does not set any restart policies by default, but be aware that they will conflict with most process managers.
- So don’t set restart policies if you are using a process manager.

Using a process manager

- When you run`docker start -a`, Docker will automatically attach to the running container, or start it if needed and forward all signals so that the process manager can detect when a container stops and correctly restart it.

Upstart on Debian

- In /etc/init.d create a script

description "Redis container"

author "Me"

start on filesystem **and** started docker

stop on runlevel [!2345]

respawn

script /usr/**bin**/docker **start** -a redis_server

end script

Systemd unit files in Redhat based

In /etc/systemd/system/
[Unit]
Description=Redis container
Requires=docker.service
After=docker.service
[Service]
Restart=always
ExecStart=/usr/bin/docker
start -a redis_server Exec
Stop=/usr/bin/docker stop -t 2 redis_server
[Install]
WantedBy=local.target

Control and configure Docker with systemd

- Starting the Docker daemon

sudo systemctl **start** docker

sudo systemctl enable docker

- Creating HTTP Proxy for Docker
- **mkdir** /etc/systemd/**system**/docker.service.d
- vi /etc/systemd/system/docker.service.d/http-proxy.conf
- **[Service]**
  Environment="HTTP_PROXY=http://<username>:<password>@proxy.example.com:80/”
  "NO_PROXY=localhost,127.0.0.1,docker-registry.somecorporation.com"
- sudo systemctl daemon-reload
- sudo systemctl **show** docker *--property Environment*
  Environment=HTTP_PROXY=**http**://proxy.example.com:80/
- sudo systemctl restart docker

Docker with Puppet

- The module is available on the Puppet Forge and can be installed using the built-in module tool
- $ puppet **module install garethr/docker**

A Puppet manifest

```
docker::image { 'ubuntu': ensure => 'absent', }
docker::run { 'helloworld':
  image => 'ubuntu',
  command => '/bin/sh -c "while true; do echo hello world; sleep 1; done"',
}
```

Docker stats

- You can use the `docker stats` command to live stream a container's runtime metrics. The command supports CPU, memory usage, memory limit, and network IO metrics.
- $ docker stats test
- CONTAINER CPU % MEM USAGE / LIMIT MEM % NET I/O BLOCK I/O test 1 0.07% 796 KB / 64 MB 1.21% 788 B / 648 B 3.568 MB / 512 KB