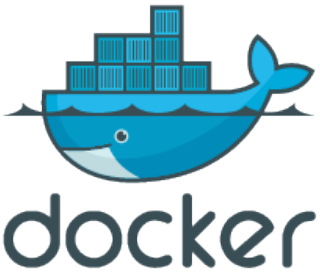
Getting Started

- Quickstart Guide

Quickstart Guide



Overview

A Docker swarm service for automatically updating your services whenever their base image is refreshed.
This Wiki Post will guide you through setting up Shepherd for Docker Swarm.

How does it work?

Shepherd triggers updates by updating the image specification for each service, removing the current digest. Docker resolves the image tag, checks the registry for a newer version, and updates running container tasks as needed. Docker handles rolling updates, minimizing downtime for replicated services.

Prerequisite

Having a running Docker Swarm Enviroment (https://wiki.aeoneros.com/books/docker-guide/chapter/swarm-mode)

Usage

Docker CLI

```
docker service create --name shepherd \
  --constraint "node.role==manager" \
  --mount type=bind,source=/var/run/docker.sock,target=/var/run/docker.sock,ro \
  containrrr/shepherd
```

Docker Compose

```
version: "3"
services:
  shepherd:
    image: containrrr/shepherd
    environment:
      TZ: 'US/Eastern'
      SLEEP_TIME: '5m'
      FILTER_SERVICES: ''
      VERBOSE: 'true'
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock
    deploy:
      placement:
        constraints:
          - node.role == manager
```

Compose Examples:

docker-compose.apprise.yml

```
version: "3"
```

```
services:
  app:
    image: containrrr/shepherd
    environment:
      APPRISE_SIDECAR_URL: notify:5000
      TZ: 'US/Eastern'
      SLEEP_TIME: '5m'
      FILTER_SERVICES: ''
      VERBOSE: 'true'
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock
    networks:
      - notification
    deploy:
      placement:
        constraints:
          - node.role == manager

  notify:
    image: mazzolino/apprise-microservice:0.1
    environment:
      NOTIFICATION_URLS: mailtos://user:pass@domain/?to=target@example.com
    networks:
      - notification

networks:
  notification:
```

docker-compose.labels.yml

```
version: "3"

services:
  shepherd:
    image: containrrr/shepherd
    environment:
```

```
    # Beware YAML gotchas regarding quoting:
    # With KEY: 'VALUE', quotes are part of yaml syntax and thus get stripped
    # but with KEY='VALUE', they are part of the value and stay there,
    # causing problems!
    SLEEP_TIME: "1d"
    TZ: "US/Eastern"
    VERBOSE: "true"
    IGNORELIST_SERVICES: "label=shepherd.autodeploy=false"
    FILTER_SERVICES: "label=shepherd.autodeploy=true"
  volumes:
    - /var/run/docker.sock:/var/run/docker.sock
  deploy:
    placement:
      constraints:
        - node.role == manager

# Explicitly enable shepherd for this service
updating-app:
  image: hello-world
  deploy:
    labels:
      - shepherd.autodeploy=true

# Explicitly disable shepherd for this service
not-updating-app:
  image: hello-world
  deploy:
    labels:
      - shepherd.autodeploy=false

# Implicitly disable shepherd for this service
# because of FILTER_SERVICES above
another-not-updating-app:
  image: hello-world
```

docker-compose.swarm-cronjob.yml

```
version: "3"

services:
  app:
    image: containrrr/shepherd
    environment:
      TZ: 'US/Eastern'
      FILTER_SERVICES: ''
      IGNORELIST_SERVICES: "test"
      RUN_ONCE_AND_EXIT: "true"
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock:ro
    deploy:
      replicas: 0
      restart_policy:
        condition: none
      labels:
        - swarm.cronjob.enable=true
        # Start service every day at midnight
        - swarm.cronjob.schedule=0 0 0 * * *
        - swarm.cronjob.skip-running=true
      placement:
        constraints:
          - node.role == manager

  scheduler:
    image: crazymax/swarm-cronjob:latest
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock:ro
    environment:
      - "TZ=Europe/Berlin"
      - "LOG_LEVEL=info"
      - "LOG_JSON=false"
    deploy:
      placement:
        constraints:
          - node.role == manager
```

Running on a swarm-cron schedule

When running Shepherd as described with a `SLEEP_TIME`, the de facto running times will drift the longer the container is running. If you want to run Shepherd on a fixed schedule instead, it is recommended to pair it with swarm-cronjob:

1. Create a **swarm-cronjob** service as described here.
2. Set `RUN_ONCE_AND_EXIT` to `true`, `replicas` to `0`, and `restart_policy` to `condition: none`. Add Docker labels for the schedule.

See **docker-compose.swarm-cronjob.yml** above for a full stack example that includes both Shepherd and Swarm-Cronjob.

Configuration / Environment Variables

Below is a table of environment variables used for configuring Shepherd:

Environment Variable	Description	Example Value
SLEEP_TIME	Time interval between update checks (default: 5 minutes)	5m
IGNORELIST_SERVICES	Space-separated list of services to ignore during updates	shepherd my-other-service
FILTER_SERVICES	Filter for specifying services to update (matches docker service ls filter)	label=com.mydomain.autodeploy
ROLLBACK_ON_FAILURE	Roll back a service to the previous version if an update fails	true
UPDATE_OPTIONS	Additional options for the docker service update command	--update-delay=30s
TIMEOUT	Timeout for the docker service update process (default: 5 minutes)	300

Environment Variable	Description	Example Value
APPRISE_SIDECAR_URL	URL for the Apprise sidecar service to enable update notifications	apprise-microservice:5000
IMAGE_AUTOCLEAN_LIMIT	Enable old image autocleaning on service update	5
RUN_ONCE_AND_EXIT	Run Shepherd once and then exit	true
WITH_REGISTRY_AUTH	Enable private registry authentication	true
REGISTRY_USER / REGISTRY_PASSWORD	Credentials for private registry authentication	user123 / secret_password
REGISTRIES_FILE	Path to the secret file containing multiple registry authentication entries	/var/run/secrets/shepherd-registries-auth
WITH_INSECURE_REGISTRY	Enable connection to an insecure private registry	true
WITH_NO_RESOLVE_IMAGE	Prevent pulling images from the registry	true
TZ	Set the timezone for log entries	Europe/Zurich

Use Private Registry Authentication

Use Private Registry Authentication

Use Private Registry Authentication

You can enable private registry authentication by setting the **WITH_REGISTRY_AUTH** variable. Use **REGISTRY_USER** and **REGISTRY_PASSWORD** for a single registry. If using multiple accounts, create a secret file with the following format:

```
idregistryloginpassword
```

Example:

```
blog    registry.gitlab.com    gitlab+deploy-token-5123674    ssw2Nrd2
```

Create the Docker secret:

```
docker secret create shepherd-registries-auth private/shepherd-registries-auth
```

Then use the secret in your `docker-compose.yml`:

```
services:
  app:
    image: containrrr/shepherd
    environment:
      REGISTRIES_FILE: /var/run/secrets/shepherd-registries-auth
    secrets:
      - shepherd-registries-auth
secrets:
  shepherd-registries-auth:
    external: true
```

Add a label to specify the correct line from the secret file:

```
deploy:
  labels:
    - shepherd.enable=true
    - shepherd.auth.config=blog
```

Set **WITH_INSECURE_REGISTRY** to `true` to connect to an insecure private registry.

Set **WITH_NO_RESOLVE_IMAGE** to `true` to prevent pulling images from the