Using Docker Swarm mode on OpenPOWER servers

Pradipta Banerjee

December 21, 2016

This article explains how to set up a Docker swarm cluster using the newly introduced Swarm mode feature of Docker Engine.

Introduction

Docker Engine 1.12 and later versions include the *swarm* mode for natively managing a cluster of Docker Engine hosts (called *swarm*). You can use the Docker command-line interface (CLI) to create a swarm and deploy services to a swarm.

In this article, we'll see how to use the Docker Swarm mode on OpenPOWER servers.

You can find more details about the Docker Swarm mode at: [https://docs.docker.com/engine/swarm/](https://docs.docker.com/engine/swarm/).

What you'll need

For IBM® Power® platforms, Docker is already part of Ubuntu from Vivid version 15.04 and later, SUSE Linux Enterprise Server (SLES) 12 and later, and Fedora version 23 and later. You can download the relevant packages from distribution repositories. Additionally, if you want to try out the latest upstream Docker binary on IBM Power Systems™ servers, you can download it from the [Docker Master Binaries](https://github.com/docker/swarmkit) website. The binaries are prefixed with the `linux/ppc64le/` tag.

Binary packages for RHEL 7 LE is available at [Unicamp repository](https://github.com/docker/swarmkit) or [IBM yum repository](https://github.com/docker/swarmkit).

Some key terms

**SwarmKit**

SwarmKit is a toolkit for orchestrating distributed systems. It includes primitives for node discovery, raft-based consensus, task scheduling, and so on. You can find more details about SwarmKit at: [https://github.com/docker/swarmkit](https://github.com/docker/swarmkit).

**Swarm**

A swarm is a cluster of Docker Engines. The Docker Engine CLI includes the commands for swarm management (such as adding and removing nodes) and commands for deploying services onto the swarm.
Node

A node is where the Docker Engine is running and it is part of the swarm. Nodes can either be worker nodes or manager nodes.

Manager nodes perform the orchestration and cluster management functions required to maintain the required state of the swarm. Manager nodes elect a single leader to conduct orchestration tasks.

Worker nodes receive and run tasks dispatched from manager nodes.

Service and task

A service is the definition of the task that is run on the worker node. The service definition includes a specific Docker image to be used.

Setting up a swarm cluster

The following steps guide you through the process of setting up a swarm cluster.

Step 1. Set up swarm

On a node designated as the manager node, run the following command:

```
# docker swarm init --advertise-addr 192.168.122.173
```

Swarm initialized: current node (0gqevumsltdkz8bo8a4ybprii) is now a manager.

To add a worker to this swarm, run the following command:
```
docker swarm join --token SWMTKN-1-14dpd920dwirq5k6bqm3h4x1lwy4m1fiopdiw3ykorlxxkio9aq-do6misrpzzx77r4jwd4qfvu02 192.168.122.173:2377
```

To add a manager to this swarm, run the following command:
```
docker swarm join --token SWMTKN-1-14dpd920dwirq5k6bqm3h4x1lwy4m1fiopdiw3ykorlxxkio9aq-11mlah1d25fsxihj8sj4bmr
```

The output specifies the command to be used to add worker nodes to the swarm. Run the specified command on all the nodes that you want to be added as a worker node.

Run the `docker node ls` command on the manager node to view the nodes in a swarm.

```
# docker node ls
```

<table>
<thead>
<tr>
<th>ID</th>
<th>HOSTNAME</th>
<th>STATUS</th>
<th>AVAILABILITY</th>
<th>MANAGER STATUS</th>
</tr>
</thead>
<tbody>
<tr>
<td>0gqevumsltdkz8bo8a4ybprii*</td>
<td>pkb-rhel71-1</td>
<td>Ready</td>
<td>Active</td>
<td>Leader</td>
</tr>
<tr>
<td>5scn3k2kwtdgx20b1jv5f13yvq</td>
<td>pkb-rhel71-2</td>
<td>Ready</td>
<td>Active</td>
<td></td>
</tr>
<tr>
<td>9bck58xol1lxic1b188ae1kac</td>
<td>pkb-rhel71-3</td>
<td>Ready</td>
<td>Active</td>
<td></td>
</tr>
</tbody>
</table>

Step 2: Create a service for deploying to swarm

Run the `docker service create` command on the manager node to create a service.
Using Docker Swarm mode on OpenPOWER servers

```bash
# docker service create ppc64le/mysql
794lmhdqvj0d
# docker service ls

ID          NAME            REPLICAS IMAGE COMMAND
794lmhdqvj0d dreamy_euler 1/1  ppc64le/mysql
```

Run the `docker service inspect` command to view details about the service.

```bash
# docker service inspect 794lmhdqvj0d
[
  {
    "ID": "794lmhdqvj0dt17oj1sywgw7o",
    "Version": { "Index": 158 },
    "CreatedAt": "2016-12-06T08:55:18.622742Z",
    "UpdatedAt": "2016-12-06T08:55:18.622742Z",
    "Spec": {
      "Name": "dreamy_euler",
      "TaskTemplate": {
        "ContainerSpec": {
          "Image": "ppc64le/mysql"
        },
        "Resources": {
          "Limits": {},
          "Reservations": {}
        },
        "RestartPolicy": {
          "Condition": "any",
          "MaxAttempts": 0
        },
        "Placement": {}
      },
      "Mode": {
        "Replicated": {
          "Replicas": 1
        }
      },
      "UpdateConfig": {
        "Parallelism": 1,
        "FailureAction": "pause"
      },
      "EndpointSpec": {
        "Mode": "vip"
      }
    },
    "Endpoint": {
      "Spec": {"}
    },
    "UpdateStatus": {
      "StartedAt": "0001-01-01T00:00:00Z",
      "CompletedAt": "0001-01-01T00:00:00Z"
    }
  }
]
```

**Step 3: Swarm constraints**

It is possible to limit the nodes on which the tasks get scheduled by using `constraints`. For example, if you want to run a task on a specific node, you can run the following command:
Using Docker Swarm mode on OpenPOWER servers

```bash
# docker service create --constraint 'node.hostname == pkb-rhel71-2' ppc64le/mysql
eh1zbktxdrqypvwbu61dvst

# docker service ls
```

<table>
<thead>
<tr>
<th>ID</th>
<th>NAME</th>
<th>REPLICAS</th>
<th>IMAGE</th>
<th>COMMAND</th>
</tr>
</thead>
<tbody>
<tr>
<td>794lmhdqvj0d</td>
<td>dreamy_euler</td>
<td>1/1</td>
<td>ppc64le/mysql</td>
<td></td>
</tr>
<tr>
<td>eh1zbktxdrq</td>
<td>jovial_carson</td>
<td>1/1</td>
<td>ppc64le/mysql</td>
<td></td>
</tr>
</tbody>
</table>

```bash
# docker service ps eh1zbktxdrq
```

<table>
<thead>
<tr>
<th>ID</th>
<th>NAME</th>
<th>IMAGE</th>
<th>NODE</th>
<th>DESIRED STATE</th>
<th>CURRENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>79n3swv7zxr1h7l3arrz8fw</td>
<td>jovial_carson.1</td>
<td>ppc64le/mysql</td>
<td>pkb-rhel71-2</td>
<td>Running</td>
<td>Running</td>
</tr>
</tbody>
</table>

30 seconds ago

Similarly, you can add labels to a swarm node and use the labels for scheduling.

```bash
# docker node update --label-add role=frontend pkb-rhel71-3
# docker node inspect pkb-rhel71-3 --pretty
```

<table>
<thead>
<tr>
<th>ID:</th>
<th>9bck58xolilxic1b188ae1kac</th>
</tr>
</thead>
<tbody>
<tr>
<td>Labels:</td>
<td></td>
</tr>
<tr>
<td>- role = frontend</td>
<td></td>
</tr>
<tr>
<td>Hostname:</td>
<td>pkb-rhel71-3</td>
</tr>
<tr>
<td>Status:</td>
<td></td>
</tr>
<tr>
<td>State:</td>
<td>Ready</td>
</tr>
<tr>
<td>Availability:</td>
<td>Active</td>
</tr>
<tr>
<td>Platform:</td>
<td></td>
</tr>
<tr>
<td>Operating System:</td>
<td>linux</td>
</tr>
<tr>
<td>Architecture:</td>
<td>ppc64le</td>
</tr>
<tr>
<td>Resources:</td>
<td></td>
</tr>
<tr>
<td>CPUs:</td>
<td>2</td>
</tr>
<tr>
<td>Memory:</td>
<td>11.47 GiB</td>
</tr>
<tr>
<td>Plugins:</td>
<td></td>
</tr>
<tr>
<td>Network:</td>
<td>bridge, host, null, overlay</td>
</tr>
<tr>
<td>Volume:</td>
<td>local</td>
</tr>
<tr>
<td>Engine Version:</td>
<td>1.12.0</td>
</tr>
</tbody>
</table>

```bash
# docker service create --constraint 'node.labels.role==frontend' ppc64le/mysql
```

Conclusion

As you can see, setting up a Docker cluster using the new Docker Swarm mode is a breeze.