Setting up a Docker Swarm on the Personal Compute Cluster

September 2, 2019 (https://diybigdata.net/2019/09/setting-up-a-docker-swarm/)
michael (https://diybigdata.net/author/michael/)

The last time I set up a Spark cluster, I installed Spark manually and configured each node directly. For the small scale cluster I had, that was fine. This time time the cluster is still relatively small scale. However, I do want to take advantage of a prebuilt containers, if possible. The standard for that is Docker. So in this post I will set up a Docker Swarm on the Personal Compute Cluster (https://diybigdata.net/personal-compute-cluster-2019-edition/).

Installing Docker

Before we begin, if you set up SETI@Home as the end of the last post (https://diybigdata.net/2019/09/personal-cluster-node-build-out/), we need to stop it first (skip if you didn't set up SETI@Home):

```
Shell
1 parallel-ssh -i -h ~/cluster/all.txt -l root "service boinc-client sto
```

We need to ensure that swap is off, as various applications we will be working with later do not like it:

```
Shell
1 parallel-ssh -i -h ~/cluster/all.txt -l root "swapoff -a"
```

Now we install all the needed software:

Shell

```
parallel-ssh -i -h ~/cluster/all.txt -l root "apt-get update -y"
parallel-ssh -i -h ~/cluster/all.txt -l root "apt-get upgrade -y"
parallel-ssh -i -h ~/cluster/all.txt -l root "apt-get install apt-tran
parallel-ssh -i -h ~/cluster/all.txt -l root "wget https://download.do
parallel-ssh -i -h ~/cluster/all.txt -l root "echo 'deb [arch=amd64] h
parallel-ssh -i -h ~/cluster/all.txt -l root "sudo apt-get update -y"
parallel-ssh -i -h ~/cluster/all.txt -l root "apt-get install docker-c
```

The installation of the software should have created the user group `docker`. Now we need to add our user account to that group so we can run docker without `sudo` (change the user account name as needed):

Shell

```
parallel-ssh -i -h ~/cluster/all.txt -l root "usermod -aG docker micha
```

Now we need to configure Docker to use our nodes' `/mnt/data` partitions for all its storage. We also want to tell Docker that it is OK to use the local registry that we will be setting up later in this post, and force it to use the Google DNS. To do this, we need to create a configuration file for Docket. Start by creating or editing the `/etc/docker/daemon.json` file:

Shell

```
sudo vi /etc/docker/daemon.json
```

If this `daemon.json` file was new, make the contents look like this:

JSON

```
{
    "data-root":"/mnt/data/docker/",
    "insecure-registries": ["master:5000"],
    "dns" : ["8.8.8.8"]
    "metrics-addr" : "0.0.0.0:9323",
    "experimental" : true
}
```

If the file already existed, simply add the element shown above to the file. Now distribute this file to the slave nodes:

```
Shell
1 parallel-scp -h ~/cluster/slaves.txt -l root /etc/docker/daemon.json /
```

Finally, start docker on all machines:

```
Shell
1 parallel-ssh -i -h ~/cluster/all.txt -l root "systemctl restart docker
```

You need to log out of the master node and log back in for the group membership to take effect. Once doing so, try docker out:

```
Shell
1 docker run hello-world
```

You should get a pleasant message from Docker. Now lets set up the Docker swarm. On the master node, if you did not allow all traffic into the cluster via the `ufw` firewall, we first need to open the firewall for docker:

```
Shell
1 sudo ufw allow 2376/tcp
2 sudo ufw allow 7946/udp
3 sudo ufw allow 7946/tcp
4 sudo ufw allow 80/tcp
5 sudo ufw allow 2377/tcp
6 sudo ufw allow 4789/udp
7 sudo ufw reload
8 sudo ufw enable
9 sudo systemctl restart docker
```

Now we init the swarm:

```
Shell
1 docker swarm init --advertise-addr 10.1.1.1
```

That should give you a message which has a command to run on the slave nodes for them to join the swarm. You can do that easily with `pssh`. Update for the join command you got from the prior command:

```
Shell
1 parallel-ssh -i -h ~/cluster/slaves.txt -l michael "docker swarm join
```

Check that it worked and the nodes are all in the swarm:

```
Shell
1 docker node ls
```

Now let's create a simple service just to prove it all worked:

```
Shell
1 docker service create --name webserver -p 80:80 httpd
2 docker service ls
```

You can now go to a web browser on your development computer, and visit the URL of the public IP address for your cluster, which for me would be `http://192.168.1.60`. You should see a web page with the phrase "It Works!". Once you are satisfied with your accomplishment, we can remove the web service:

```
Shell
1 docker service rm webserver
2 docker service ls
```

The web page should no longer work if you try to refresh it in your browser.

Monitoring Docker Swarm

If you want to set up a nice lightweight management UI for your docker swarm, install Portainer:

```
Shell
1 sudo mkdir -p /mnt/data/portainer/
2 docker service create \
3     --name portainer \
4     --publish 9999:9000 \
5     --replicas=1 \
6     --limit-memory=128M \
7     --constraint 'node.role == manager' \
8     --mount=type=bind,src=/mnt/data/portainer/,dst=/data \
9     --mount=type=bind,src=/var/run/docker.sock,dst=/var/run/docker.soc
10    portainer/portainer
```

Now point a browser at `http://master-node-ip:9999` (replacing `master-node-ip` with the public WAN address of your master node), create an account, and enjoy!

Creating a Local Docker Registry

We will be running a Docker registry within the swarm in order to manage the images we will be creating later. The registry is needed to easily distribute your custom docker images to all nodes when a service is create. Normally registries are set up with security. However, because our cluster is private and we do not have a signed certificate for from a certificate authority, we need to set up the registry is insecure with a self-signed certificate. On the master node do the following:

```
Shell
1 cd
2 mkdir certs
3 cd certs/
4 openssl req -newkey rsa:4096 -nodes -sha256 -keyout registry.key -x509
```

The certificate generation will pose a few questions to you. Fill them in as you see fit, except for the question on "Common Name". For that, you should enter the IP address of the master node, which is `master`. Once you do that, we need to distribute the certificate to the docker engine on all nodes:

```
Shell
1 parallel-ssh -i -h ~/cluster/all.txt -l root "mkdir -p /etc/docker/cer
2 parallel-scp -h ~/cluster/all.txt -l root registry.crt /etc/docker/cer
3 parallel-ssh -i -h ~/cluster/all.txt -l root "mkdir -p /etc/docker/cer
4 parallel-scp -h ~/cluster/all.txt -l root registry.crt /etc/docker/cer
5 parallel-ssh -i -h ~/cluster/all.txt -l root "service docker restart"
```

Now, start the registry in the swarm with the following, but adjust the path `/home/michael/certs` to be where the certs folder is in your home directory is on the master node:

```
Shell
1 docker service create --name registry --publish=5000:5000 \
2     --constraint=node.role==manager \
3     --mount=type=bind,src=/home/michael/certs,dst=/certs \
4     --limit-memory=500m \
5     -e REGISTRY_HTTP_ADDR=0.0.0.0:5000 \
6     -e REGISTRY_HTTP_TLS_CERTIFICATE=/certs/registry.crt \
```

```
    -e REGISTRY_HTTP_TLS_KEY=/certs/registry.key \
    registry:latest
```

In order to track want's in the registry, we can load a simply registry visualizer:

Shell

```
docker service create --name registry-browser -p 5050:8080 \
    -e DOCKER_REGISTRY_URL=https://10.1.1.1:5000 \
    -e NO_SSL_VERIFICATION=true \
    --limit-memory=128m \
    klausmeyer/docker-registry-browser
```

Now point you browser at `http://master-node-ip:5050` (change the IP address to your cluster's WAN address), and you can see that your local docker registry is empty. But we will be changing that in the upcoming posts.

Fun Things to do on your Docker Swarm

Run SETI@Home on Docker Swarm

We still have some work to do before we are ready to run Spark. As with the last post, that will be another day. For now, we can now leverage our new Docker Swarm to run and manager a SETI@Home processes. A Docker image for BOINC, the software that enables SETI@Home, has been prebuilt and available in the registry, and you can use that. There is even a version optimized for Intel CPUs, which the Personal Cluster users. To launch the service into the Docker Swarm such that each node gets an instance to run, use this command:

Shell

```
docker network create -d overlay --attachable boinc
parallel-ssh -i -h ~/cluster/all.txt -l root "mkdir -p /mnt/data/boinc
parallel-ssh -i -h ~/cluster/all.txt -l root "chgrp docker /mnt/data/b
docker service create \
    --mode global \
    --name boinc \
    --network=boinc \
    --hostname="boinc-{{.Node.Hostname}}" \
    --mount type=bind,src=/mnt/data/boinc,dst=/var/lib/boinc \
    -p 31416:31416 \
    -e BOINC_GUI_RPC_PASSWORD="123" \
```

```
    -e BOINC_CMD_LINE_OPTIONS="--allow_remote_gui_rpc" \
    --limit-memory=1G \
    --limit-cpu=8 \
    boinc/client:intel
docker run \
    --rm \
    --network boinc \
    boinc/client \
    boinccmd_swarm \
        --passwd 123 \
        --project_attach http://setiathome.berkeley.edu \
        <*insert your account key here*>
```

Replace `<*insert your account key here*>` with your SETI@Home account key. One thing I will point out is the use of the `--limit-cpu` option in the service creation command. Without it, each instance of the service will use 100% of the CPUs. The option limits the effective number of CPUs each instance can utilize. Since each of our nodes has 12 virtual CPUs (6 cores, 2 threads per core), setting `--limit-cpu` to 6 has the effect of limiting the process to 50% of the CPU capacity. The reason I did this was at 100%, the nodes' cooling fans are running at full blast. While these EGLOBAL S200 computers (http://s.click.aliexpress.com/e/3rIO2ICs) are pretty quiet, at 100% CPU utilization and 100% fan speed, things do get noticeably noisy. So, I limited CPU usage to 50% and the nodes run without the fans turning on.

You can read more about the BOINC client and operating it within Docker here (https://github.com/BOINC/boinc-client-docker).

Run a Nethack Server

If you can recollect the days of crawling a dungeon made up of ASCII characters, you might enjoy this one: you can easily deploy a Nethack (http://www.nethack.org) server for anyone on your home network to play on. This really shows off the power of Docker in a fun way: it makes it easy for people to share whole software deployments. In this case, we can pull Matsuu Takuto's image for a Nethack server (https://github.com/matsuu/docker-nethack-server) and set it up to run on our swarm:

Shell

```
docker service create --replicas 1 --publish=23:23 --name nethack matsu
```

Then to the play the game, simply telnet into port 23 on the cluster's master node using its external IP address.

```
Shell
1 telnet 192.168.1.60 23
```

Best wishes on your ascension!

Personal Cluster (https://diybigdata.net/category/personal-cluster/)

Docker (https://diybigdata.net/tag/docker/), setup (https://diybigdata.net/tag/setup/)

« Building Out Nodes for Personal Cluster

Installing GlusterFS on the Personal Compute Cluster »

Leave a Reply

You must be logged in (https://diybigdata.net/wp-login.php?redirect_to=https%3A%2F%2Fdiybigdata.net%2F2019%2F09%2Fsetting-up-a-docker-swarm%2F) to post a comment.

Recent Posts

Improving the cooling of the EGLOBAL S200 computer (https://diybigdata.net/2020/01/improving-cpu-cooling-of-eglobal-s200/)

Archives

June 2020 (https://diybigdata.net/2020/06/)

March 2020 (https://diybigdata.net/2020/03/)

January 2020 (https://diybigdata.net/2020/01/)

December 2019 (https://diybigdata.net/2019/12/)

October 2019 (https://diybigdata.net/2019/10/)

September 2019 (https://diybigdata.net/2019/09/)

November 2017 (https://diybigdata.net/2017/11/)

January 2017 (https://diybigdata.net/2017/01/)

November 2016 (https://diybigdata.net/2016/11/)

October 2016 (https://diybigdata.net/2016/10/)

September 2016 (https://diybigdata.net/2016/09/)

August 2016 (https://diybigdata.net/2016/08/)

July 2016 (https://diybigdata.net/2016/07/)

June 2016 (https://diybigdata.net/2016/06/)

Categories

Computer Science (https://diybigdata.net/category/general/computer-science/)

Data Analysis (https://diybigdata.net/category/low-cost-cluster/data-analysis/)

Data Analysis (https://diybigdata.net/category/general/data-analysis-general/)

General (https://diybigdata.net/category/general/)

Hardware (https://diybigdata.net/category/low-cost-cluster/hardware/)

Low Cost Cluster (https://diybigdata.net/category/low-cost-cluster/)

Personal Cluster (https://diybigdata.net/category/personal-cluster/)

Set Up (https://diybigdata.net/category/low-cost-cluster/set-up/)

Spark Performance (https://diybigdata.net/category/general/spark-performance/)

Uncategorized (https://diybigdata.net/category/uncategorized/)

Meta

Log in (https://diybigdata.net/wp-login.php)

Entries feed (https://diybigdata.net/feed/)

Comments feed (https://diybigdata.net/comments/feed/)

WordPress.org (https://wordpress.org/)