https://doi.org/10.20378/irbo-51198

Hypriot Cluster Lab: An ARM-Powered Cloud Solution Utilizing Docker

Marcel Großmann, Andreas Eiermann, Mathias Renner
Faculty of Information Systems and Applied Computer Science
Otto-Friedrich University,
Bamberg, Germany
Email: marcel.grossmann@uni-bamberg.de, {andreas | mathias}@hypriot.com

***Abstract*—Following the establishment of virtualization approaches, cloud services within data center environments have become easily manageable. Modern infrastructures use virtual machines as a platform for service delivery, requiring powerful servers. Conjointly, the uprising of the Internet of Things implies new challenges to provide applications that can successfully manage data and communicate with a large number of connected devices. The standards of entry have resulted in extreme difficulties for small enterprises and educational institutions trying to provide their own virtualized services. The *Hypriot Cluster Lab (HCL)* - made publicly available on Github[1] - offers cloud functionality while running on ARM processors, thereby minimizing costs. Due to the fact that such processors offer less computational power, services are packaged into lightweight containers built using the Docker framework, which avoid the overhead associated with virtual machines.**

I. Fog Computing Clusters

In recent years *single board computers (SBC)* have gained tremendous computing power, resulting in a need for rethinking the structures of the Internet, especially the placement of powerful and energy-hungry data centers. As proposed by Azam and Huh [1] the *Internet of Things (IoT)* endorses unnecessary data communications that burden core networks and data centers in the cloud. They propose fog computing as one possible solution for relocating virtualized services from the cloud to the network edge. At this point HCL could provide the basis for a virtualized edge because it runs on ARM architecture, which behaves like a small data center and ships energy efficient features by design.

The survey of the Linux Foundation [2], which covered 53 different SBCs, demonstrated that the Raspberry Pi (RPi) 2 Model B surpasses others. However, full virtualization that depends on *virtual machines (VM)* outperforms the capacities of small SBCs. Therefore, we only consider Linux container virtualization approaches such as *Docker*. As Felter *et al.* [3] explain, this is a great benefit for small SBCs because containers "offer the control and isolation of VMs with the performance of bare metal" [3].

Compared to similar platforms (e.g. Apache Mesos[2] or Google's Kubernetes[3]) that offer containerized cluster computing (mainly for data centers), HCL enables the same functionality on inexpensive and energy efficient hardware, like the RPi.

[1]https://github.com/hypriot/cluster-lab
[2]http://mesos.apache.org/
[3]http://kubernetes.io/

II. Hypriot Cluster Lab

The activities required to initialize HCL are depicted in Figure 1. First an IEEE 802.1Q VLAN with a predefined ID is set up on every node so that the existing network is not polluted by broadcast messages of the cluster. Subsequently, a fixed IP address is used temporarily to listen for a specific zeroconf service that announces whether a cluster is already running in the VLAN. When the service exists, the temporary IP address is deleted and a new one is requested via DHCP (cf. the yellow activities in Figure 1). When the service is unavailable, the IP address is changed to the first subnet and the node publishes a zeroconf service. Additionally, a DHCP server is configured, such that other nodes allocate their IP addresses dynamically (cf. the blue activities in Figure 1).

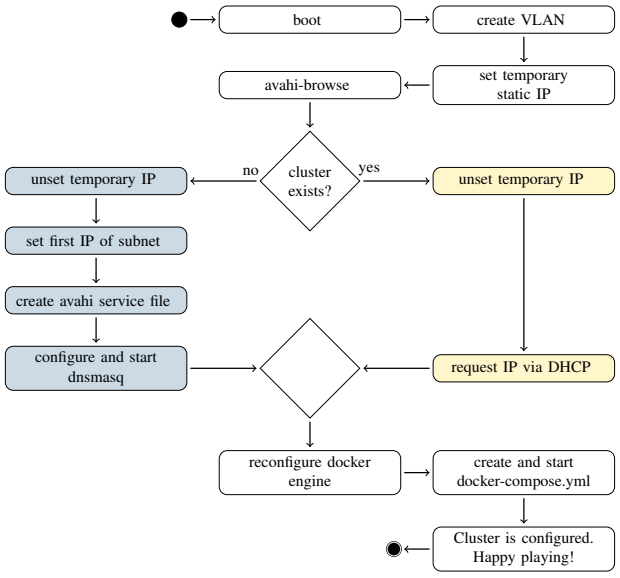

Figure 1. Activity Diagram of the node configuration of HCL. White activities occur on all nodes, while blue are associated with master nodes and yellow ones with slave nodes.

Subsequently, this first node initiates a Consul[4] container, which is a key-value store with a built-in DNS server. Other

[4]https://github.com/hashicorp/consul

nodes join the existing Consul service. As with Consul, all nodes also interconnect with Docker Swarm[5].

After the entire cluster is configured, one is able to use Docker-Compose[5], Consul[4], Docker Swarm[5] and other suitable tools. Therefore, one must be aware of the architecture of Docker images in order to run them on ARM.

III. Use Cases for HCL

We reveal how HCL works in two use cases that involve setting it up by flashing HCL's image[6] on three SD cards. Subsequently, one RPi is booted from them and is referred to as the *master* node. The hardware of HCL is configured like depicted in Figure 2. After approximately two minutes, we point a browser to the IP address or hostname of the master and append port 8500, e.g., *http://master:8500*. The web interface of Consul is displayed, listing one node up and running. Next, we boot up two more RPis and the web interface shows three nodes as running two minutes later. To configure Docker, we log into the master via SSH. Finally, we set an environment variable that maps all Docker commands to the Docker Swarm API, which is useful for easily managing containers on different hosts.

```
export DOCKER_HOST=tcp://master:2378
```

A. Starting a Webserver in an Overlay Network

A very basic use case involves spinning up a webserver on one node and displaying it from a second node through an overlay network. In doing so, we create an overlay network *demo* with the webserver placed on the master node.

```
docker network create -d overlay demo && \
docker run -d --name=webserver \
 --net=demo -e "constraint:node==master" \
 hypriot/rpi-nano-httpd
```

Because of *name=webserver*, nodes inside the *demo* network reach this container at *http://webserver*. Thereafter, a second container is started on a slave node that displays the contents of the webserver residing on the master node.

```
docker run --net=demo \
 -e "constraint:node!=master" \
 hypriot/armhf-busybox \
 wget -O- http://webserver/index.html
```

In summary, Docker containers can perform multi-host communication with the help of overlay networks. Under the hood, VXLAN is used to setup overlay networks.

B. Load Balancing of Webservers in the Cluster

This use case implements HAProxy as a load balancer on the master, which distributes incoming requests to other slave nodes in the cluster that eventually serve content. It is useful to set up at least three RPis for this use case. Docker Compose allows the setup to be automated to a large extent. First, prepared setup files[7] are downloaded to the master of the cluster for this demo. Within the downloaded folder, we change to the subfolder *loadbalancing_with_haproxy*, allowing all components to be initiated with a single command:

```
docker-compose -p loadbalancing up -d
```

This command sets up three services with one container each: HAProxy, which receives incoming HTTP requests that are forwarded to slave nodes, which are listed in the configuration of HAProxy (regularly updated by Consul-template). Consul-template has instructions for maintaining a list of available webservers in HAProxie's configuration. Consul-template is triggered by Registrator, which monitors the cluster for new containers and reports them to Consul-template. Thus, when pointing a browser from a device outside the cluster (but from within the same network) to the IP address of the master node, a website with the ID of the website's container appears.

Finally, we increase the number of webservers to be distributed on all slave nodes with this single command:

```
docker-compose -p loadbalancing scale \
 demo-hostname=10
```

When reloading the website in the browser, a round-robin strategy routes requests one by one to ten different web server containers.

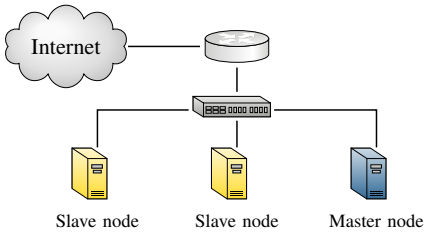

Figure 2. Example RPi configuration for example use cases of HCL

IV. Conclusion & Future Work

This demonstration illustrated the ease of setting up a small energy-efficient virtualized cloud on affordable hardware. However, the single point of failure of HCL is the *master* node, which manages the swarm. To overcome this, a future extension of HCL could include failover mechanisms for the master node based on high availability features[5] recently published for Docker Swarm.

References

[1] M. Aazam and E.-N. Huh, "Fog computing and smart gateway based communication for cloud of things," in *Future Internet of Things and Cloud (FiCloud), 2014 International Conference on*, Aug 2014, pp. 464–470.

[2] E. Brown, "Top 10 linux and android hacker sbcs of 2015," 2015, (accessed 2015-Sept-2). [Online]. Available: https://www.linux.com/news/embedded-mobile/mobile-linux/834861-top-10-linux-and-android-based-hacker-sbcs-of-2015

[3] W. Felter, A. Ferreira, R. Rajamony, and J. Rubio, "An updated performance comparison of virtual machines and linux containers," *technology*, vol. 28, p. 32, 2014.

[5] https://docs.docker.com/

[6] http://blog.hypriot.com/post/introducing-hypriot-cluster-lab-docker-clustering-as-easy-as-it-gets/

[7] https://github.com/hypriot/rpi-cluster-demo