Installing Docker on FIWARE Cloud

Introduction

This project is part of FIWARE. Check also the FIWARE Catalogue entry for Docker.

This installation guide will explain how you can easily setup your Docker environment to develop and deploy your services on the FIWARE Lab. For more information about how to use docker on fiware please see the docker on fiware user guide.

FIWARE has developed many features with which you can compose your services;

Features related to: * data context management * integration of data and media content * IoT * Apps for data visualization and publication * Advanced Web UI * Software defined networking * Security * and more.

All of the features known as Generic Enablers or GEs for short have packaged as Docker containers. These FIWARE containers can be leveraged to compose your own FIWARE based services. You can develop and deploy your services on the FIWARE Cloud.

The FIWARE Cloud supports both VM hosting and Docker hosting. So why would you want to use Docker over VMs? Docker containers are much smaller than VMs. They take up a fraction of the space. Most organization could fit all of their required containers on a single Docker host. In the world dominated by VMs, modular design would call for one component per VM. With Docker you can pack the same components referred as Docker micro-services on a single Docker host. It is much easier to manage a single Docker host than multiple VMs. The start-up time for each micro-service is much faster than what would be required for a VM. Likewise, integrating the micro-services into a single application is much easier. Beyond all that the Docker eco-system makes it easier to reuse other Docker containers and tools for managing the docker lifecycle.

The primary elements of the Docker Ecosystem are Docker Hub, Docker Engine, Docker Compose, Docker Swarm and Docker Machine. * Docker Hub is a cloud service for managing and sharing Docker container images, including FIWARE services, known as Generic Enablers (GE); You can browse for the GEs in the fiware catalogue or search for their docker containers in the Docker Hub. * Docker Engine, or simply Docker, creates and runs Docker containers; It supports automatic pulling of docker images from Docker Hub. It then caches the images locally to speed up subsequent pulls. You can make run time changes to your images and then push them back on to Docker Hub to save them or share them with other developers. * Docker Compose allows you to define and run multi-container applications; Most GEs describe how they can be used to compose new applications. * Docker Swarm: manages a pool of Docker hosts using the full suite of Docker tools. Because Docker Swarm serves the standard Docker API, any tool that already communicates

with a Docker daemon, e.g. Docker-Compose, can use Swarm to transparently scale to multiple hosts. * Docker Machine: creates and manages Docker hosts locally or on cloud providers (including OpenStack). It can be used to create and manage Docker swam clusters. Since the FIWARE Cloud is based on OpenStack, docker machine can be used to create and manage docker hosts on the FIWARE cloud. * Docker containers, Docker hosts, and Docker Swarm clusters can be hosted on the FIWARE lab.

Not only can you host and manage docker on FIWARE, but you can do this remotely from your local Docker client;

The Docker hosts may be spread over multiple FIWARE regions and each region may contain multiple Docker hosts. Most of the communication between the client and FIWARE is through the Docker REST API. But Docker Machine also uses the OpenStack API to allocate resources and SSH to provision and configure the Docker hosts. All of this communication is done securely using a combination of Transport Layer Security (TLS), token based authentication and authorization with Openstack Keystone, and SSH encryption.

Once the Docker host have be created other tools like compose can be used to create and deploy complex Docker services.

Swarm can be used to manage a cluster of Docker hosts running on FIWARE.

Requirements

As we said earlier, most of FIWARE Docker hosting can be accomplish remotely from your docker client.

But there are a few set up steps that require you to use the FIWARE:

1. Cloud Portal. You must obtain a FIWARE account. Signing up for a FIWARE account is simple. Go to FIWARE. Submit the required information. You should be authorized within a day.
2. Optionaly request a community account which will give you more privaleges and a more stable environment.
3. Once you have an account you must create a security group.
   - Select Cloud->Security->Security Groups.
   - Edit Security Group open incoming ports:
   - The required ports are 2376 and 22. The Docker Daemon kistens on port 2376. Opening the docker daemon port allows docker clients to communicate with docker remotely. The ssh daemon listens on port 22. Docker Machine uses ssh to provision and configure the docker host. If necessary you can replace this port with another.
   - Optional you can open other ports to be used by the docker containers to interact with the outside world: Ports 32768-33768 are auto allocated by docker when creating containers. Of course you open other ports. For instance 8080 for a web service. We also have observed that the Docker Swarm Master uses 3376. Through out this document we refer to the security group docker-machine-sg that we created in this step.
4. Next you must allocate at least one floating IP to your project.

* Select Cloud->Security->Floating IPs->Allocate IP to Project. 5. You also might what to take a look a the VM images that are used to when creating a docker host. Through out this document we use the VM image "Ubuntu Server 14.04.1 (x64)". But the "base_ubuntu_14.04" also works. The difference between the two images are that "Ubuntu Server 14.04.1 (x64)" allows for root access with ssh. "base_ubuntu_14.04" is set up for ubuntu access with sudo privaledges. 6. Next you must install docker and docker machine on your local work station. We will use docker machine to create docker hosts and swarm clusters on FIWARE regions.

Details on how to accomplish all this follows.

Docker Machine

Overview

Machine lets you create Docker hosts on your computer, on cloud providers, and inside your own data center. It automatically creates hosts, installs Docker on them, then configures the docker client to talk to them. A "machine" is the combination of a Docker host and a configured client.

Once you create one or more Docker hosts, Docker Machine supplies a number of commands for managing them. Using these commands you can

- start, inspect, stop, and restart a host
- upgrade the Docker client and daemon
- configure a Docker client to talk to your host For details see the Docker Machine home page.

Create Docker Host on FIWARE

The first step in creating a docker host on the FIWARE is set up your local docker client environment.

First install docker and docker machine on your local host to create a docker client.

We now turn our attention to preparing the local docker client to remotely create and manage docker hosts that will run on the FIWARE cloud.

Prepare the environmental variables for Docker Machine by exporting the following environment variables: * OS_REGION_NAME specifies the region that will be used * OS_TENANT_NAME specifies your project name * OS_USERNAME is the email address that you specified when opening your FIWARE account * OS_PASSWORD is your fiware account password * OS_AUTH_URL is the FIWARE identity manager's URL * OS_AUTH_STRATEGY is the strategy for authentication. In this case "keystone"

For example:

```
>export OS_REGION_NAME='Spain2'
>export OS_TENANT_NAME='john-smith cloud'
>export OS_USERNAME='jsmith@gmail.com''
>export OS_PASSWORD='secret''
```

```
>export OS_AUTH_URL='http://cloud.lab.fi-ware.org:4730/v2.0/'
>export OS_AUTH_STRATEGY='keystone'
```

Now you can use Docker Machine to create a docker host on FIWARE. The docker machine takes 7 parameters: * -d specifies the openstack driver is used to interact with the cloud provider. Of course we specify the openstack driver since FIWARE is implemented as an openstack cloud. * --openstack-flavor-id: The flavor id species the size of the virtual machine in which to place the docker host * --openstack-image-name: The image name is the name of the virtual machine image. The image id could be used instead using the openstack-image-id flag. * --openstack-net-name: The network name as shown in FIWARE Cloud GUI. * --openstack-floatingip-pool: floating ip-pool as shown in FIWARE Cloud GUI.
* --openstack-sec-groups: security group as shown in FIWARE Cloud GUI.
* The last parameter is the name of the docker host, in this cast docker-host.

When it completes it tells you to run the docker-machine env command to learn how to interact with the docker host that was created. It tells you the TLS with be used secure the communication between the client and docker host.

For instance to create the docker host called docker-host you issue this command:

```
>docker-machine create -d openstack
--openstack-flavor-id="2"
--openstack-image-name="Ubuntu Server 14.04.1 (x64)"
--openstack-net-name="node-int-net-01"
--openstack-floatingip-pool="public-ext-net-01"
--openstack-sec-groups="docker-machine-sg" docker-host
```

When it finishes it says:

```
To see how to connect Docker to this machine run: docker-machine env docker-host
```

```
>docker-machine env docker-host
```

supplies the information to interact remotely with docker-host with your docker client.

```
>eval "$(docker-machine env docker-host)"
```

transforms the environment so that the local docker client manages the remote fiware docker host. Once the eval command is run all docker commands acts as if they are run on the remote docker host.
Actually the commands are executed as docker REST apis securely transferred over the internet within the TLS envelop.

For example:

```
>docker run hello-world
```

launches hello-world on the docker-host running of FIWARE.

```
>docker run -d -P training/webapp python app-py
```

launches the webapp service on the docker-host running of FIWARE. Use docker ps to see which port the webapp is listening to and the external port to which it is paired. Then you can interact with the service using the docker-host URL and the assigned external port.

Docker Compose

Docker compose can also be used to run multi-service applications.

For instance given you can run the FIWARE context broker using this docker-compose.yml:

```
mongo:
  image: mongo:2.6
  command: --smallfiles

orion:
  image: fiware/orion
  links:
    - mongo
  ports:
    - ":1026"
  command: -dbhost mongo
```

The yml file describes two containers mongo db and orion. Orion listens on port 1026. We could assign an external port to pair with 1026, but in this case we allow docker to auto define the port.

```
>docker-compose up -d
```

brings up the orion service and run it in the background as a daemon. We the use the docker-compose ps command to see the service instance and its external port.

Docker Swarm

We can also create swarm clusers on the FIWARE cloud.

The first task is to create a token that is used to bind the different hosts on the cluster. The 'docker run swarm create` command is used for the task. It returns a token that is used to identify and bind the cluster.

Next we create the Swarm master using the same docker-machine create command that was described above and add the following flags: * --swarm * --swarm-master * --swarm-discovery discovery requires the token as input

For example:

```
>docker-machine create -d openstack
--openstack-flavor-id="2"
--openstack-image-name="Ubuntu Server 14.04.1 (x64)"
--openstack-net-name="node-int-net-01"
--openstack-floatingip-pool="public-ext-net-01"
--openstack-sec-groups="docker-machine-sg" docker-host
--swarm --swarm-master
--swarm-discovery token://$TOKEN Swarm-Master
```

Creates the docker host "Swarm-Master" which is a swarm master.

To create a swarm slave do not include the --swarm-master flag.

The docker-machine ls command will indicate which host is a master and which host is a slave.