Your supermarket company is in the process of improving its Docker-based applications. They have built a set of three RESTful data services that communicate with each other as part of a larger infrastructure. You have been given the task of designing a Docker application stack so that these three services can be easily managed and scaled as a unit. A Docker Swarm cluster has already been set up by you to use in previous exercises.

Here is some background information on the three services:

1. Fruit Service

a. Provides a list of fruits sold in the company's stores.

b. You can use the Docker image tag linuxacademycontent/fruit-service:1.0.1 to run this service.

c. Listens on port 80.

d. The service should be named fruit inside the stack.

2. Vegetable Service

a. Provides a list of vegetables sold in the company's stores.

b. You can use the Docker image tag linuxacademycontent/vegetable-service:1.0.0 to run this service.

c. Listens on port 80.

d. The service should be named vegetables inside the stack.

3. All Products Service

a. Queries the other two services, combining their data into a single list of all produce.

b. You can use the Docker image tag linuxacademycontent/all-products:1.0.0 to run this service.

c. Listens on port 80.

d. Use the environment variables FRUIT\_HOST and FRUIT\_PORT to set the host and port which will be used to query the fruit service.

e. Use the environment variables VEGETABLE\_HOST and VEGETABLE\_PORT to set the host and port which will be used to query the vegetable service.

Step 1

Deploy a Docker application stack that meets the following specifications:

1. The stack is called produce.

2. The stack runs the Fruit, Vegetable, and All Products services.

3. The All Products service is able to query the Fruit and Vegetable services.

4. The All Products service is published on port 8080.

One you have deployed the stack, you can verify whether it is working by querying the All Products service:

curl localhost:8080

If the stack is set up correctly, you should get a combined list of fruits and vegetables.

Step 2

Once you have deployed the stack and verified that it is working, modify the stack by scaling both the Fruit and Vegetable services up to 3 replicas.

Good luck!

Build and deploy the application stack.

Create an empty project directory with a Docker compose YAML file inside.

cd ~/

mkdir produce

cd produce

vi produce.yml

Build a stack definition in produce.yml to meet the provided specifications.

version: '3'

services:

fruit:

image: linuxacademycontent/fruit-service:1.0.1

vegetables:

image: linuxacademycontent/vegetable-service:1.0.0

all\_products:

image: linuxacademycontent/all-products:1.0.0

ports:

- "8080:80"

environment:

- FRUIT\_HOST=fruit

- FRUIT\_PORT=80

- VEGETABLE\_HOST=vegetables

- VEGETABLE\_PORT=80

Deploy the stack using the compose file.

docker stack deploy -c produce.yml produce

Verify that the stack is working.

curl localhost:8080

Scale the Fruit and Vegetable services in the stack.

Set the number of replicas to 3 for the Fruit and Vegetable services in the compose file.

vi produce.yml

version: '3'

services:

fruit:

image: linuxacademycontent/fruit-service:1.0.1

deploy:

replicas: 3

vegetables:

image: linuxacademycontent/vegetable-service:1.0.0

deploy:

replicas: 3

all\_products:

image: linuxacademycontent/all-products:1.0.0

ports:

- "8080:80"

environment:

- FRUIT\_HOST=fruit

- FRUIT\_PORT=80

- VEGETABLE\_HOST=vegetables

- VEGETABLE\_PORT=80

Redeploy the stack using the compose file.

docker stack deploy -c produce.yml produce

Verify that the stack is still working.

curl localhost:8080

Use docker stack services produce to see that the number of replicas for the Fruit and Vegetable services is now 3.