[SLIDER-906] Support for Docker based application packaging with first class YARN support

Thomas (Yu) Liu, Billie Rinaldi, Gour Saha

In the tech preview version (SLIDER-780) Slider was responsible for performing docker pull and docker run of the application docker images. In this first-cut implementation of *Phase 2 of docker support*, Slider is no longer responsible for these operations. It integrates with YARN's **DockerLinuxContainerRuntime** (YARN-3611), which is responsible for docker pull (if required) and docker run to create a docker container. In the first patch it relies on the existence of Slider Agent in the docker image, which requires minor modification of application docker images, including ensuring that a compatible version of python is installed. Slider AM is responsible to take necessary steps to maintain the application in a stable state. For example it will request new containers to replace failed ones. It will also respond to application owner's requests to scale specific components of the application up/down. In subsequent commits (in the works), YARN will completely take over the responsibility of managing and monitoring the docker containers (YARN-4692). YARN will trigger appropriate callbacks in Slider AM to report the state/health of docker containers. This will enable application owners to run their docker based applications as is, without the need to modify their docker images.

Components

Slider Client

Interface that application owners use to launch and manage native docker on YARN

Slider AM

Coordinates with YARN RM to launch and manage containers

Slider Agent

Process running in user application docker containers for client side management

Slider appConfig.json, metainfo.json, resource.json

Configuration files for applications

YARN Resource Manager

Schedule and allocate resources for docker containers

YARN Node Manager

Launch and monitor user application docker containers

YARN Utility Scripts

Set up networking for user application docker containers

Workflow

Creation of docker based applications

- App owner creates appConfig.json, metainfo.json, resource.json. In metainfo.json, they need to specify docker images, start command for each container and number of containers per image, together with application specific configuration.
- Then the app owner launches the application using Slider client, providing the configuration files created in the previous step
- Slider client works with YARN resource manager to submit the application
- YARN launches Slider AM if sufficient resources are available in the cluster
- Slider AM, once launched, parses the configuration files provided by the user and gets information like docker images and number of containers per image. Then it requests for those resources/containers from YARN RM.
- YARN RM allocates containers, just like any other YARN application, on hosts with available resources. Node manager performs docker pull (if required) and docker run of the specified docker image of the application to create the docker container. For now node manager expects the docker images to be available on all hosts where NM is running. The command executed with docker run is launch_container.sh, a script created by NM that contains the command to start the Slider Agent process.
- Slider Agent registers itself to the Slider AM on behalf of the docker container.
- Slider AM then instructs the Agent to invoke the start command specified in metainfo.json. Slider Agent invokes the start command and the application process starts running inside the docker container.

Monitoring and managing of running docker based applications

- Slider Agent periodically reports the health of the docker container to AM. It also reports the networking information set up by the utility scripts to the AM.
- Slider AM listens to events from YARN on health of the docker containers and takes appropriate actions on failures.

Stopping of docker based applications

- The application owner uses Slider Client to issue stop command.
- Slider AM issues stop signal to the Slider Agent of all the running containers. Slider Agent stops the application process within the docker container.
- Then Slider AM requests YARN RM to kill the running containers.

Example App

Start command:

```
slider create myredis --metainfo metainfo.json --template appConfig.json
--resources resources.json
```

metainfo.json:

```
{
      "schemaVersion": "2.1",
      "application": {
            "name": "REDIS",
            "components": [{
                  "name": "REDIS",
                  "type": "yarn_docker",
                  "dockerContainers": [{
                        "name": "redis",
                        "image": "redis",
                        "startCommand": "redis-server --port ${REDIS_PORT}",
                        "ports": [{
                              "containerPort": "6379",
                              "hostPort": "6379"
                        }]
                  }]
            }]
      }
}
```

appConfig.json:

```
{
    "schema": "http://example.org/specification/v2.0.0",
    "metadata": {},
    "global": {},
    "components": {
        "REDIS": {
            "redis.lifetime": 12,
            "redis.statusCommand": "ls",
            "env.REDIS_PORT": "6379"
        }
```

```
    }
}
```

resources.json:

```
{
    "schema" : "http://example.org/specification/v2.0.0",
    "metadata" : {},
    "global" : {},
    "components": {
        "slider-appmaster": {
            "yarn.memory": "384"
        },
        "REDIS": {
            "yarn.role.priority": "1",
            "yarn.component.instances": "1",
            "yarn.memory": "512"
        }
    }
}
```