zero-downtime tests

Perform zero-downtime update tests

Hugo + Jenkins

The following guide describes zero-downtime update tests and provides instructions for running them.

Overview

Providing zero downtime is one of the main objectives of [redacted] as a project.

With zero downtime you can update an app or revert changes without a service's interruption. An end user never notices that you alter running software. Zero-downtime tests are recommended to verify that the containerized version of an app runs as planned, and that the Jenkins pipeline is correctly set. Before testing zero-downtime upgrades of the app, make sure you have deployed your app to a cluster. For more information, see Deploy your app to a [redacted] cluster.

Perform test

To test zero-downtime upgrades:

1. Make changes to your app's source code.
2. Prepare and run the following script that calls a certain endpoint of your app in the test realm.

```
for i in {1..1000}; do
  curl --write-out "%{http_code}\n" --silent --output /dev/null 'http://demo.crowd.ixs.wgcrowd.io' #enter your service's URL
done
```

3. While the script is running, build the Docker image of the app and assign a new Docker tag to it. Here, the new tag is `version1`.

```
$ docker build -t crowd-multiversion-test-app:version1 .

$ docker tag crowd-multiversion-test-app:version1 registry.wdo.io/crowd/crowd-multiversion-test-app:version1
```

4. Push the image to the registry.

```
docker push registry.wdo.io/crowd/crowd-multiversion-test-app:version1
```

5. In **Crowdfile**, in the `image` parameter, specify the updated registry address with the new tag.

```
tasks:
  web: #task name
        type: marathon
        container:
            image: registry.wdo.io/crowd/crowd-multiversion-test-app:version1
        health_checks:
        -
            protocol: "MESOS_HTTP"
            path: /
            port: 80
            gracePeriodSeconds: 40
            intervalSeconds: 10
            timeoutSeconds: 10
            maxConsecutiveFailures: 6
...
```

   The **health checks are critical** for the zero-downtime feature. Remember to have valid health checks in your **Crowdfile**.

6. Save **Crowdfile** and update it in the cluster by running consecutive commands `conf push` and `apply`.

```
$ corch -R demo-multiapp -c http://cras.ixs.wgcrowd.io conf push -c <path-to-realm-config> \
-s <path-to-Crowdfile> -j <path-to-JSON-schema>

$ corch -R demo-multiapp -c http://cras.ixs.wgcrowd.io apply -u
```

   The updates are applied on the cluster in a matter of seconds.

7. Verify that zero-downtime upgrade is achieved. For that:
   - Check that the running script returns status 200 all the time.

```
for i in {1..1000}; do
  curl --write-out "%{http_code}\n" --silent --output /dev/null 'http://demo.crowd.ixs.wgcrowd.io' #enter your service's URL
done
200
200
...
```

   - Ensure that no error responses from your app are displayed in [redacted] cluster's Grafana: http://grafana.[redacted]_cluster>.wgcrowd.io, when the new app's version is delivered and deployed.

A Project Manager can skip this step at the sole risk, and leave the corresponding remark in the epic's task. Remember that skipping this test may cause errors in production environments in future.