Integration Testing with Kubernetes & Openshift

using Arquillian Cube & Forge

Dipak Pawar

Why?

- `Advanced namespace management`
- `Manages Lifecycle of Containers`
- `Manages Deployment for Kubernetes & Openshift`
- `Make created services, pods, replica sets and kubernetes client available inside your test case (via @ArquillianResource)`

End to end test flow

```
Given:

- Kubernetes cluster should be up & running.
- Deployment config.
```

```
When:

- Deploy Resources:
 * Create temporary unique namespace per testsuite.
 * Deploy all resources to kubernetes cluster.
 * Wait for environment to setup on cluster.
```

```
Then:

- Inject all resources like KubernetesClient, Services, Replication Controller, Pods
into test & make it available to perform operations around it.
- Verify all required resources & integration around all resources.
- Clean up entire namespace after test.
```

Write a test using

Prerequisites

- Make sure that forge is installed
- Go to Forge console by running command `forge`
- Install arquillian addon for Forge using: `addon-install-from-git --url https://github.com/forge/arquillian-addon.git`

Write a test for K8s using Forge

```
    # Create a project.
    project-new --named demo-k8s

    # Add depenency & configuration for arquillian & junit.
    arquillian-setup --test-framework junit --standalone

    # Add depenency & configuration required for arquillian-cube
    arquillian-cube-setup --type kubernetes --file-path src/test/resources/k8s.json

    # Create a test.
    arquillian-create-test --named KubernetesTest --target-package org.arquillian.cube

    # Add Cube Test.
    arquillian-cube-add-test --test-class org.arquillian.cube.KubernetesTest --service
-name serviceName

    # Run test in Forge console.
    build

    # OR

    # Run test using maven.
    mvn test
```

Test Scaffolding for arquillian-cube

Integration Test

Problem: - Verify number of pods running in current namespace after deployment as a part of your build pipeline.

- Solution1: Write shell scripts for following logic:
    - Deploy resources
    - Switch to required namespace
    - Look for all pods
    - Verify it with expected number of pods
- Solution2: Let's look at how easy it is using arquillian-cube ?
    - Arquillian-cube create unique namespace for your tests
    - Manages deployment for you.
    - You have all resources available to you inside test. Look for following example.

```
    @RunWith(Arquillian.class)
    public class ExampleTest {

        @ArquillianResource KubernetesClient client;

        @ArquillianResource Session session;

        @Test
        public void testAtLeastOnePod() throws Exception {
            assertThat(client).pods().runningStatus().filterNamespace(session
.getNamespace()).hasSize(1);
        }
    }
```

We can verify almost every resource inside tests(rc, rs, po, svc), application endpoints & it is alive or not.

More examples

```
import io.fabric8.kubernetes.client.KubernetesClient;
import org.jboss.arquillian.junit.Arquillian;
import org.jboss.arquillian.test.api.ArquillianResource;
import org.junit.Test;
import org.junit.runner.RunWith;

import static io.fabric8.kubernetes.assertions.Assertions.assertThat;

@RunWith(Arquillian.class)
public class SystemTest {

    @ArquillianResource
    KubernetesClient client;

    @Test
    public void testRunningPodStaysUp() throws Exception {
        assertThat(client).deployments().pods().isPodReadyForPeriod();
    }
}
```

Ready within a time period (30 seconds by default), then that the pod keeps being Ready for a period (defaults to 10 seconds).

Resources

- https://github.com/arquillian/arquillian-cube
- http://arquillian.org/arquillian-cube/

Questions?

twitter

@dipakpawar231

github

github.com/dipak-pawar