System & Integration Testing with Kubernetes & Openshift

using Arquillian Cube & Forge

Dipak Pawar

Arquillian?

- Testing Platform
- Features
  - Container Agnostic
  - IDE friendly
  - Extensible Platform
  - Test Enrichment
  - Strong tooling for getting started
  - Integration available with Junit, TestNG, Spock, JBehave, Cucumber

Why?

- 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)
- Advanced namespace management

Integration Testing

- Integration testing is the phase in software testing in which individual software modules are combined and tested as a group

System Testing

- System Testing is a level of the software testing where a complete and integrated software is tested.

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

Integration Test

```
    @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);
        }
    }
```

System Test

```
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).

Questions?

twitter

@dipakpawar231

github

github.com/dipak-pawar