OSS
Test **A**utomation **F**ramework
User Guide

Introduction

The objective of the OSS Test Automation Framework is to automate the process of building/installing and testing a spinnaker instance. The main goal is to have this as an automated and repeatable process, potentially for different environments.

It has four major modules:

1. **Build Infra and Configure OSS:** This is achieved through Github actions.
    a. Provision EKS cluster (if required)
    b. Deploy Spinnaker
2. **Test Automation framework:** This is developed using Java and an easily extensible framework. The framework needs to be compiled into a docker image and configured before running the Github actions. Please see the Section "How to compile TestFramework into a docker image" on how to create the framework docker image.
3. **Execute Test Suite**: The above docker image is deployed and executed as a K8s job in the same namespace where OSS is installed through Github action:
    a. Run Test Cases
4. **Clean up Infra:** This is achieved through Github actions, if required.
    a. Destroy Spinnaker
    b. Destroy Infrastructure

This user guide explains how to use GitHub Actions to run OSS Automation Test Framework. The framework consists of several modules, given above, that includes creating a cluster, installing Spinnaker, running test cases, and destroying the cluster. Follow these steps to set up and use this test framework effectively.

How to Configure the OSS Test Automation Framework

Prerequisites

1. If provisioning the EKS K8s cluster is required:
    - Create Github secrets AWS_ACCESS_KEY_ID, AWS_SECRET_ACCESS_KEY and AWS_REGION with appropriate permissions for the access Id.
    - Create a Github secret AWS_BUCKET with the AWS S3 bucket name where Terraform state files are stored during K8s cluster creation and deletion.
2. If the K8s cluster is pre-defined (i.e., not created in the step-1)
    - Store kubeconfig file in AWS Secret Manager

3. Store values.yaml in AWS Secret Manager to be used by spinnaker installer
4. Store config-overide.properties in the AWS Secret Manager which has the required configurations to be overridden. The default config.properties is part of oss-test-framework image. (sample config-overide.properties and config.properities are given below).
5. Create and store the halconfig that has all the Spinnaker configurations (cloud accounts, integrations etc) in AWS S3 bucket. The AWS S3 bucket that has the halconfig must be given in values.yml.

How to Execute Test Suite & Check Test Results

1. Navigate to the Git repository where GitHub Actions are configured (e.g.,https://github.com/OpsMx/oss-test-automation-framework).
2. Run the Github action "**Provision EKS Cluster**" if the creation of the EKS cluster is required (see the Section below: Creating a cluster)
3. Run the Github action "**Deploy Spinnaker**" (see the Section below: Deploy Spinnaker)
4. Configure the Test Suite (see the Section below: How to configure the Test Suite)
5. Run the Github action "**Run Test Cases**" (see the Section below: Running Test Cases)
6. Run the Github action "**Destroy Spinnaker**" (see the Section below: Destroy the Spinnaker)
7. Run the Github action "**Destroy EKS cluster**" (see the Section below: Destroy the Infrastructure)
8. The Test Results are pushed to S3 and also available as a HTML link in Github actions.

Creating a Cluster

1. Navigate to the Git repository where GitHub Actions are configured (e.g.,https://github.com/OpsMx/oss-test-automation-framework).
2. Navigate to the "**Actions**" tab.
3. Choose "**Provision EKS Cluster"**
4. Click on the "**Run workflow**" button on the right side.
5. Fill in the appropriate details:
    a. **EKS cluster name in AWS:** Specify the name of the EKS cluster in AWS to be created
    b. **Kubeconfig Secret Name in AWS:** Specify the name of your Kubeconfig secret (AWS secret will be created with this name).
    c. **KMS_KEY alias name:** Enter the alias name of the AWS Key Management Service (KMS) key/Encryption key.
6. Click on the "**Run workflow**" button, represented in green colour.

7. Monitor the running workflow to check its status.

Deploy Spinnaker

1. Go to your Git repository.
2. Navigate to the "Actions" tab.
3. Choose "Deploy Spinnaker s3ops."
4. Click on the "Run workflow" button.
5. Fill in the appropriate details:
    a. Secret name of kubeconfig: Provide the AWS secret name which has cluster kubeconfig.
    b. Secret name for values file: Specify the AWS secret name which has values.yml file in it.
    c. Spinnaker namespace: Provide the namespace for deploying Spinnaker (if not exist it will create a namespace)
    d. Helm Release: Specify the release name for Spinnaker.
6. Click on the "Run workflow" button.
7. Check the status of the workflow by clicking on the running workflow and then OpsMx-Spinnaker. Spinnaker will be up and running once all checks are completed.

Running Test Cases

The config.properties and overrideconfig.properties must be configured appropriately before we create the docker image of the test framework. See the below section “How to configure the Test Suite” on how to configure various parameters required for the test cases.

1. Navigate to the Git repository where GitHub Actions are configured (e.g.,https://github.com/OpsMx/oss-test-automation-framework).
2. Navigate to the "**Actions**" tab.
3. Choose "**Run Test Cases**".
4. Click on the "**Run workflow**" button.
5. Fill in the appropriate details:
    a. **Secret name of kubeconfig:** Specify the AWS secret name which has the kubeconfig stored in it.
    b. **Secret name of overrideconfigfile:** Specify the AWS secret name which has overrideconfig file in it.
    c. **Namespace where the testcases should run:** Provide the namespace in which our test cases should run .
    d. **Regular expression to run the test cases:** Regular expressions can be used to run specific test cases which are present in

CommonTests.java.
Here are the sample expressions to run specific testcases,
    i. Empty space to execute all the test cases
    ii. createApp*
    iii. *App*
    iv. *Pipe*
    v. createApp*,*Pipe*
    vi. createApp*,*Pipe*,*Jenkin*
    vii. createApp*,*Helm*

6. Click on the "**Run workflow**" button.
7. Monitor the running workflow to check its status.

Test Case Report and Log Report:

Test case report,log reports for successful Run Test Cases github action can be viewed by following below steps:

- Click on **Actions**
- Select **Run Test Cases** github action
- Select the Run Test Cases workflow run(ex: Run Test Cases #110) for which we need to view the reports
- Under **Test-Cases summary** section we can see the report urls
    - TestCases Report
    - TestCases Logs
- Click on **TestCases Report** to view TestCases Report which is in .html file
- Click on **TestCases Logs** to download the Log Report and view.

Destroying the Spinnaker

1. Navigate to the Git repository where GitHub Actions are configured (e.g.,https://github.com/OpsMx/oss-test-automation-framework).
2. Navigate to the "**Actions**" tab.
3. Choose "**Destroy Spinnaker**"
4. Click on the "**Run workflow**" button.
5. Fill in the appropriate details:
    a. **Secret name of kubeconfig:** Specify the AWS secret name where the kubeconfig is stored.
    b. **Spinnaker namespace:** Provide the namespace for destroying Spinnaker.
    c. **Helm Release:** Provide the release name for destroying Spinnaker
6. Click on the "**Run workflow**" button.
7. Monitor the running workflow to check its status.

Destroying the Cluster

1. Navigate to the Git repository where GitHub Actions are configured (e.g.,https://github.com/OpsMx/oss-test-automation-framework).
2. Navigate to the "**Actions**" tab.
3. Choose "**Destroy EKS cluster**".
4. Click on the "**Run workflow**" button.
5. Fill in the appropriate details:
    a. **Secret name of kubeconfig**: Provide the secret name (in AWS secret manager) containing the cluster kubeconfig to delete the cluster.
    b. **EKS cluster name in AWS**: Provide the EKS cluster name in AWS for destroying the cluster
    c. **KMS_KEY alias name**: Provide Encryption key of the kubeconfig secret to destroy the cluster.
6. Click on the "**Run workflow**" button.
7. Monitor the running workflow to check its status**.**

List of Automation Test Cases

Test Scenarios: Sample Application & Pipelines

- Create a sample application
- Create a sample pipeline in the application
- Update an existing application
- Update an existing pipeline

Test Scenarios: Automated Triggers

- Trigger a pipeline with Jenkins Builds
- Trigger a pipeline with Git Commit
- Trigger a pipeline with Webhook trigger
- Trigger a pipeline with CRON Triggers
- Trigger a pipeline with Docker Push (jfrog)

Test Scenarios: Notifications

- Slack Notifications (application-level)
- Slack Notifications (pipeline-level)
- Slack Notifications (stage-level)
- Slack Notifications (manual judgement)
- Email Notifications (application-level)
- Email Notifications (pipeline-level)
- Email Notifications (stage-level)
- Email Notifications (manual judgement)

Test Scenarios: Sample Deployments

- Create and execute a pipeline to pull an image from docker.io and deploy to K8s
- Create and execute a pipeline to Build the image using Jenkins and deploy the generated image in a K8s account

Test Scenarios: RBAC

- Create a sample application with proper RBAC permissions
- Create a sample pipeline within the application with proper RBAC permissions

Test Scenarios: Advanced deployments

- Create and execute a pipeline to bake a helm chart and deploy to K8s
- Create and execute a pipeline to bake a kustomize and deploy to K8s
- Create sample deployment (eg: nginx) in Dynamic Accounts
- Create sample deployment (eg: nginx) in AWS ECS account
- Create sample deployment (eg: nginx) in AWS account (EC2)

How to compile TestFramework into a docker image

- Clone Project Git-Repo (Eg;: https://github.com/OpsMx/oss-test-automation-framework.git)
- In the dir where it is cloned, Run the Gradle Command to build the framework (exclude running the test cases) by using the following command :
  ```
  ./gradlew clean build -x test
  ```
- Login and build the docker image using the following commands
  ```
  docker login -u <username>  -p <password>
  docker build -t  <image:tag> -f Dockerfile-gradle
  docker push <image:tag>
  ```
- After docker image of the testframe is created, update the frameworks.yml (Eg: https://github.com/OpsMx/oss-test-automation-framework/blob/master/frameworks.yml) in the following section:

  spec.template.spec.initContainers.image
- Now, the testframe is ready to be executed through the GitHub action <Run Test Cases>

How to configure the Test Suite

Provide Spinnaker Credentials

- Url: Spinnaker url to run tests

- Authn: If true, Spinnaker username and password should be provided. If false, no username and password are required
- Username: << Spinnaker username >>
- Password: << Spinnaker password>>

Provide value as enabled/disabled for the test cases to enable/disable the run

- createApplication: set this to enabled/disabled to run or stop application creation testcase
- addEmailAndSlackNotificationForApplication: set this to enabled/disabled to run or stop adding email and slack notification for application test case
- createPipelineWithCronTrigger: set this to enabled/disabled to run or stop creating pipeline with cron trigger testcase
- updateCronPipeline: set this to enabled/disabled to run or stop update cron pipeline testcase
- createPipelineWithDeployStage: set this to enabled/disabled to run or stop pipeline creation with Deploy stage test case
- createPipelineWithJenkinsTrigger: set this to enabled/disabled to run or stop pipeline creation with Jenkins trigger testcase
- createPipelineWithGitTrigger: set this to enabled/disabled to run or stop pipeline creation with Git trigger testcase
- createPipelineWithDockerRegistryTrigger: set this to enabled/disabled to run or stop pipeline creation with DockerRegistry trigger test case
- createPipelineWithJenkinsBuildAndDeploy: set this to enabled/disabled to run or stop pipeline creation with JenkinsBuildAndDeploy test case
- createPipelineWithHelmDeployment=set this to enabled/disabled to run or stop pipeline creation with Helm Deployment test case
- createPipelineWithKustomizeDeployment=set this to enabled/disabled to run or stop pipeline creation with Kustomize Deployment test case
- createPipelineWithEC2Deployment=set this to enabled/disabled to run or stop pipeline creation with EC2 Deployment test case
- createPipelineWithECSDeployment=set this to enabled/disabled to run or stop pipeline creation with ECS Deployment test case
- executePipeline: set this to enabled/disabled to run or stop the pipeline execution
- updateApplication: set this to enabled/disabled to run or stop the Application update
- deleteSpinnakerApplication: set this to enabled/disabled to run or stop to the Spinnaker Application deletion

Provide AWS Credentials

- aws_access_key=ur_access_key - Provide your AWS access key
- aws_secret_key=ur_secret_key - Provide your AWS secret key
- s3_bucket_to_store_report=ur-terraform-state - Provide your S3 bucket name

Provide Jenkins details

- master=jenkins-account - Provide jenkins account name
- job=jenkins-job- Provide jenkins job name
- propertyFile=file.properties - Provide property file to jenkins
- jenkinsBuildStageName=JenkinsBuild - Provide stage name for jenkins build stage
- jenkinsBuildAppName=jenkins-app deploymentname
- jenkinsBuildDeleteAppName=deployment jenkins-app
- jenkinsBuildImage=image-url

Note: If **jenkinsBuildStageName** changes **JenkinsBuild** should also change in the **jenkinsBuildImage.**

Provide details for Notifications

- emailNotificationAddress: Provide email address to send notifications
- slackNotificationAddress: Provide slack channel to send notifications
- startedMsg: displays Pipeline/Stage started successfully for successful pipeline
- completedMsg: displays Pipeline/Stage ended successfully completed pipeline
- failureMsg: displays Pipeline/Stage ended with a failure
- awaitingManualJudgementMsg: displays a stage is awaiting manual judgement
- continueManualJudgementMsg: displays "stage was judged to continue" message for manual judgement stage to continue
- stopManualJudgementMsg: displays "stage was judged to stop" message for manual judgement stage to continue

Triggers

- webhookSource: Determines the target URL required to trigger this pipeline, as well as how the payload can be transformed into artifacts.

Provide Git Trigger details

- githubBranch=master
- githubProject=ur-git-project - Provide your email address
- githubSlug=ur-repo-name - Provide your repo name
- githubSource=github

Provide Application details

- appEmail: Provide email address
- appName: Provide Application name
- appDescription: Provide Application Description
- updatedAppDescription: Provide the updated Application Description

App Permission Groups

- rbacStatus=set this to enabled to enable RBAC else by default it will be disabled
- readGroup= Provide read access to the groups of your choice
- writeGroup= Provide write access to the groups of your choice
- executeGroup= Provide execute access to the groups of your choice

Provide Pipeline details

- deployPipelineName: Provide the name to deploy Pipeline
- cronPipelineName: Provide the name to cron Pipeline
- updatedCronPipelineName: Provide the name to update Cron Pipeline
- jenkinsTriggerPipelineName: Provide the name to jenkins trigger Pipeline
- gitTriggerPipelineName: Provide the name to git trigger Pipeline
- dockerRegistryTriggerPipelineName: Provide the name to trigger docker Registry Pipeline
- jenkinsBuildAndDeployPipelineName: Provide name to build and deploy jenkins Pipeline
- ec2DeploymentPipelineName=Provide the name to ec2 deployment Pipeline
- ecsDeploymentPipelineName=Provide the name to ecs deployment Pipeline
- helmDeploymentPipelineName=Provide the name to helm deployment Pipeline
- kustomizeDeploymentPipelineName=Provide the name to kustomize deployment Pipeline

Provide Stage details

- deployManifestStageName=Provide the name for deploy manifest stage
- deleteManifestStageName=Provide the name for delete manifest staage
- waitStageName=Provide the name for deploy manifest wait stage
- bakeStageName=Provide the name for bake stage
- shortWaitStagePeriod=2 Provide the minimum wait period
- waitStagePeriod=60 Provide the maximum wait period

Provide K8s details

- k8sAccountName: Provide K8s account name
- k8sNamespace: Provide K8s deployment namespace
- k8sDeploymentImage=Provide K8s deployment image (ex: nginx-deployment)
- k8sDeploymentImageVersion=Provide K8s deployment image version (ex: nginx:1.15.4)
- k8sDeploymentApp=Provide K8s deployment app

Provide Docker Registry details

- dockerAccountName= Provide docker account name
- dockerRegistryName= Provide docker registry name
- dockerRegistryOrganization= Provide docker registry organisation

- dockerRegistryImage=Provide docker registry image name

Provide Cron details
cronExpression=based on your requirements provide cron expression (ex: 0 0/5 * 1/1 * ? *)

Provide EC2 details
Stage: Bake
baseAmi=Provide baseAmi
baseLabel=Provide base label
baseName=Provide base name
baseOs=Provide base OS
cloudProviderType=Provide cloud provider type (eg: aws)
bakeName=Provide bake name
package=Provide package
rebakeStatus=Provide rebake status to true/false
awsRegion=Provide aws region
skipRegionDetection=Provide skip Region Detection to true/false
storeType=Provide store type
type=Provide stage type (eg: bake)
user=Provide user name
vmType=Provide vm type

Stage: Deploy
ec2Account=Provide ec2 Account name
associatePublicIpAddressStatus=Set this to true/false
us-east-1a=us-east-1a
us-east-1b=us-east-1b
us-east-1c=us-east-1c
us-east-1d=us-east-1d
us-east-1e=us-east-1e
us-east-1f=us-east-1f
lbName=ecs-LB-test
instanceType=t2.nano

Stage: Destroy
cloudProvider=Provide cloud provider type (eg: aws)
detail=stack
stack=new
us-east-1=us-east-1
ec2_destroyStage_target=current_asg_dynamic
ec2_destroyStage_type=destroyServerGroup

Sample config.properties

```
#########  Spinnaker Credentials  ##############################
url=http://spin-gate:8084/
authn=false

#########  TestCase Enabled/Disabled  ##############################
createApplication=enabled
addEmailAndSlackNotificationForApplication=enabled
createPipelineWithCronTrigger=enabled
updateCronPipeline=enabled
createPipelineWithDeployStage=enabled
createPipelineWithJenkinsTrigger=enabled
createPipelineWithGitTrigger=enabled
createPipelineWithDockerRegistryTrigger=enabled
createPipelineWithJenkinsBuildAndDeploy=enabled
createPipelineWithHelmDeployment=enabled
createPipelineWithKustomizeDeployment=enabled
createPipelineWithEC2Deployment=enabled
createPipelineWithECSDeployment=enabled
executePipeline=enabled
updateApplication=enabled
deleteSpinnakerApplication=enabled

#########  AWS Credentials  ##############################
aws_access_key=ur_access_key
aws_secret_key=ur_secret_key
s3_bucket_to_store_report=ur-terraform-state

#########  Jenkins ##############################
master=jenkins-account
job=jenkins-job
propertyFile=file.properties
jenkinsBuildStageName=JenkinsBuild

jenkinsBuildAppName=jenkins-app
jenkinsBuildDeleteAppName=deployment jenkins-app
jenkinsBuildImage=image-url

#########  Notifications ##############################
emailNotificationAddress=someone@anonymous.com
slackNotificationAddress=notifications

startedMsg=started successfully
completedMsg=ended successfully
failureMsg=ended with a failure
awaitingManualJudgementMsg=is awaiting manual judgement
```

```
continueManualJudgementMsg=was judged to continue
stopManualJudgementMsg=was judged to stop

#########  Triggers ##############################
webhookSource=ur-source
# Git Trigger
githubBranch=master
githubProject=ur-git-project
githubSlug=ur-repo-name
githubSource=github

#########  App  ##############################
appEmail=someone@anonymous.com
appName=testapp
appDescription=Create test Application
updatedAppDescription=Updated test Application
#########  App Permission Groups ##############################
rbacStatus=enabled
readGroup=rogroup
writeGroup=rogroup
executeGroup=rogroup

#########  Pipeline  ##############################
deployPipelineName=deployPipeline
cronPipelineName=cronPipeline
updatedCronPipelineName=updatedCronPipeline
jenkinsTriggerPipelineName=jenkinsPipeline
gitTriggerPipelineName=gitTriggerPipeline
dockerRegistryTriggerPipelineName=dockerRegistryPipeline
jenkinsBuildAndDeployPipelineName=jenkinsBuildAndDeployPipeline
ec2DeploymentPipelineName=ec2DeploymentPipeline
ecsDeploymentPipelineName=ecsDeploymentPipeline
helmDeploymentPipelineName=helmDeploymentPipeline
kustomizeDeploymentPipelineName=kustomizeDeploymentPipeline

#########  Stages  ##############################
deployManifestStageName=deployManifestStage
deleteManifestStageName=deleteManifestStage
waitStageName=waitStage
bakeStageName=bakeStage
shortWaitStagePeriod=2
waitStagePeriod=60

#########  Kubernetes  ##############################
k8sAccountName=ur-k8s-account
k8sNamespace=ur-namespace
k8sDeploymentImage=ur-image
```

```
k8sDeploymentImageVersion=ur-image-ver
k8sDeploymentApp=ur-app

#########  Docker Registry  ##############################
dockerAccountName=ur-docker-account
dockerRegistryName=ur-docker-registry-name
dockerRegistryOrganization=ur-docker-org
dockerRegistryImage=ur-docker-registry-image

#########  Cron  ##############################
cronExpression=0 0/5 * 1/1 * ? *

#########  EC2  ##############################
# Bake Stage
baseAmi=ami-06ae296e502d24311
baseLabel=release
baseName=ops-ubuntu14-java-1
baseOs=trusty
cloudProviderType=aws
bakeName=bake
package=ur-app-name
rebakeStatus=true
awsRegion=us-east-1
skipRegionDetection=true
storeType=ebs
type=bake
user=admin
vmType=hvm
# Deploy Stage
ec2Account=ur-ec2-account
associatePublicIpAddressStatus=true
us-east-1a=us-east-1a
us-east-1b=us-east-1b
us-east-1c=us-east-1c
us-east-1d=us-east-1d
us-east-1e=us-east-1e
us-east-1f=us-east-1f
lbName=ur-load-balancer
instanceType=t2.nano

# Destroy Stage
cloudProvider=aws
detail=stack
stack=new
us-east-1=us-east-1
ec2_destroyStage_target=current_asg_dynamic
ec2_destroyStage_type=destroyServerGroup
```

```
##############################################################################
```

Sample config-overide.properties

Store config-overide.properties in the AWS Secret Manager which has the required configurations to be overridden with config.properties.

Note: **k8sNamespace** is the deployment namespace which is mandatory.

```
url=http://spin-gate:8084/

deleteSpinnakerApplication=disabled

k8sNamespace=spinnakerunval
```