**O**SS

**T**est **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 and Configure OSS:** This is achieved through Github actions.
   1. Provision EKS cluster (if required)
   2. Deploy Spinnaker
2. **Test Automation framework:** This is developed using Java and easily extensible framework. This TestSuite (i.e., list of test cases that needs to be executed) can be configured through a secret, by default it uses config.properties we can override by using config-override.properties. This Test Framework is compiled into a Docker image and deployed using Github actions as a K8s job to be run in the same namespace where OSS is installed.
3. **Execute Test Suite and Report Test Results:** This is achieved through Github action.
   1. Run Test Cases
4. **Clean up Infra:** This is achieved through Github actions, if required.
   1. Destroy Spinnaker
   2. 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 to provision of EKS with AWS Access ID, Secret Key and AWS regionwith appropriate permissions.
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-override.properties in the AWS Secret Manager which has the required configurations to be overridden with config.properties which are present in git repo for running TestSuite
5. Create and store the halconfig that has all the Spinnaker configurations (cloud accounts, integrations etc) in AWS S3
6. For RBAC authz enabled should be set to true in hal config which is present in s3 bucket,

| authz:  groupMembership:  service: EXTERNAL  google:  roleProviderType: GOOGLE  github:  roleProviderType: GITHUB  file:  roleProviderType: FILE  ldap:  roleProviderType: LDAP  enabled: true |
| --- |

Fiat-local.yml file should be added to the s3 bucket under default/profiles

| auth:  groupMembership:  service: ldap  ldap:  url: ldap:*//RELEASE\_NAME-openldap:389*  managerDn: cn=admin,dc=example,dc=org  managerPassword: xxxxxxxxxxx  groupSearchBase: ou=groups,dc=example,dc=org  groupSearchFilter: member={0}  groupRoleAttributes: cn  userDnPattern: cn={0},dc=example,dc=org |
| --- |

# How to Execute Test Suite & Check Test Results

Go to the Git repository where GitHub Actions are configured (e.g.,<https://github.com/OpsMx/oss-test-automation-framework>).

1. Run the Github action “Provision EKS Cluster” if the creation of the EKS cluster is required (see the Section below: Creating a cluster)
2. Run the Github action “Deploy Spinnaker” (see the Section below: Installing Spinnaker)
3. Configure the Test Suite (see the Section below: How to configure the Test Suite)
4. Run the Github action “Run Test Cases” (see the Section below: Running Test Cases)
5. Run the Github action “Destroy Spinnaker” (see the Section below: Destroy the Spinnaker)
6. Run the Github action “Destroy EKS cluster” (see the Section below: Destroy the Infrastructure)
7. The Test Results are pushed to S3 and also available as a HTML link in Github actions.

# Creating a Cluster

1. Go to your 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, including the following:
   1. EKS cluster name in AWS: Specify the name of the EKS cluster in AWS to be created
   2. Kubeconfig Secret Name in AWS: Specify the name of your Kubeconfig secret (AWS secret will be created with this name).
   3. 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.

# Installing 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, including the following:
   1. Secret name of kubeconfig: Provide the AWS secret name which has cluster kubeconfig.
   2. Secret name for values file: Specify the AWS secret name which has values.yml file in it.
   3. Spinnaker namespace: Provide the namespace for deploying Spinnaker (if not exist it will create a namespace)
   4. 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

Prerequisite to run testcases:

Url ,namespace are mandatory in config-override.properties

1. Go to your Git repository.
2. Navigate to the "Actions" tab.
3. Choose "Run Test Cases"
4. Click on the "Run workflow" button.
5. Fill in the appropriate details, including the following:
   1. Secret name of kubeconfig: Specify the AWS secret name which has the kubeconfig stored in it.
   2. Namespace where our testcases should run: Provide the namespace in which our test cases should run .
   3. Regular expression to run the test cases: Regular expressions can be used to run specific test cases which are present in [CommonTests.java](https://github.com/OpsMx/oss-test-automation-framework/blob/master/api-automation/src/test/java/com/opsmx/oes/spintests/CommonTests.java) .

Ex: empty space to run all testcases

createApp\\*

\\*App\\*

\\*Pipe\\*

createApp\\*,\\*Pipe\\*

createApp\\*,\\*Pipe\\*,\\*Jenkin\\*

createApp\\*,\\*Helm\\*

1. Click on the "Run workflow" button.
2. Monitor the running workflow to check its status.

**Test Case Report and Log Report:**

For successful Run Test Cases github action we can get test case report, log reports 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 **OpsMx-Spinnaker 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. Go to your Git repository.
2. Navigate to the "Actions" tab.
3. Choose "Destroy Spinnaker"
4. Click on the "Run workflow" button.
5. Fill in the appropriate details, including the following:
6. Secret name of kubeconfig: Specify the AWS secret name where the kubeconfig is stored.
7. Spinnaker namespace: Provide the namespace for destroying Spinnaker.
8. Helm Release: Provide the release name for destroying Spinnaker
9. Click on the "Run workflow" button.
10. Monitor the running workflow to check its status.

# Destroying the Cluster

1. Go to your Git repository.
2. Navigate to the "Actions" tab.
3. Choose "Destroy EKS cluster by Terraform."
4. Click on the "Run workflow" button.
5. Fill in the appropriate details, including the following:
   1. Secret name of kubeconfig: Provide the secret name containing the cluster kubeconfig to delete the cluster.
   2. EKS cluster name in AWS: Provide the EKS cluster name in AWS for destroying the cluster
   3. KMS\_KEY will be used to delete alias for KMS\_KEY in AWS: 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 without RBAC
* 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 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 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
* 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 - Provide your AWS access key
* aws\_secret\_key=ur\_secret\_key - Provide your AWS secret key
* s3\_bucket\_to\_store\_report=opsmx-terraform-state - Provide your S3 bucket name

**Provide Jenkins details**

* master=accountjenkins - Provide jenkins account name
* job=Spin-canary-issuegen-build-Deploy-demo - 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=quay.io/opsmxpublic/canary-issuegen:${#stage("JenkinsBuild")["context"]["Buildnumber"]}

**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. (eg: testframe)

**Provide Application details**

* appEmail: Provide email address
* appName: Provide Application name (Eg: testappfortest)
* 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 (ex: rogroup)
* writeGroup= Provide write access to the groups of your choice (ex: rogroup)
* executeGroup= Provide execute access to the groups of your choice (ex: rogroup)

**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
* 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 (ex: nginx)

**Provide Docker Registry details**

* dockerAccountName= Provide docker account name
* dockerRegistryName= Provide docker registry name
* dockerRegistryOrganization= Provide docker registry organization
* dockerRegistryImage=Provide docker registry image name (ex: restapp/simpleapp)

**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 (eg: admin)

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=varshinitest-vates-lb

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:*//testframeappledeck.cve.apple.opsmx.net/gate/* username=test@opsmx.io password=xxxxxxxxx authn=true  #url=http:*//applespingate1.cve.apple.opsmx.net/* #authn=false  ######### TestCase Enabled/Disabled ############################ createApplication=enabled addEmailAndSlackNotificationForApplication=enabled createPipelineWithCronTrigger=enabled updateCronPipeline=enabled createPipelineWithDeployStage=enabled createPipelineWithJenkinsTrigger=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=opsmx-terraform-state  ######### Jenkins ############################ master=accountjenkins job=Spin-canary-issuegen-build-Deploy-demo propertyFile=file.properties jenkinsBuildStageName=JenkinsBuild  ######### Notifications ############################ emailNotificationAddress=test@opsmx.io 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=testframe  ######### App ############################ appEmail=test@opsmx.io 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 dockerRegistryTriggerPipelineName=dockerRegistryPipelinefortest 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=default k8sNamespace=testframe k8sDeploymentImage=nginx-deployment k8sDeploymentImageVersion=nginx:1.15.4 k8sDeploymentApp=nginx  ######### Docker Registry ############################ dockerAccountName=jfrog-docker dockerRegistryName=opsmx.jfrog.io dockerRegistryOrganization=restapp dockerRegistryImage=restapp/simpleapp  ######### 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=restapp rebakeStatus=true awsRegion=us-east-1 skipRegionDetection=true storeType=ebs type=bake user=admin vmType=hvm  # Deploy Stage ec2Account=aws-isd-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=test-vates-lb 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 configproperties-overide

# Note: url,k8sNamespace are mandatory where **url** is Spinnaker url to run tests, **k8sNamespace** is the deployment namespace.

| **url=http:*//spin-gate:8084/*  deleteSpinnakerApplication=disabled   ######### Jenkins ############################  jenkinsBuildStageName=JenkinsBuild jenkinsBuildAppName=jenkins-app jenkinsBuildImage=quay.io/opsmxpublic/canary-issuegen:${#stage("JenkinsBuild")["context"]["Buildnumber"]}  ######### Notifications ############################ emailNotificationAddress=**test@opsmx.io  **slackNotificationAddress=notifications   ######### Triggers ############################ webhookSource=testframetest  ######### App ############################ appName=demoapptesting2** appEmail=test@opsmx.io **######### Kubernetes ############################  k8sNamespace=spinnakerunval** |
| --- |

# Sample values.yaml

| #################################################### installSpinnaker: true #################################################### installRedis: true #################################################### global:  ssl:  enabled: false  stormdriver: false  certManager:  installed: false  customCerts:  enabled: false # Set to true if your organization requires custom TLS certs  secretName: oes-cacerts # Please do not change this  commonGate:  enabled: false  createIngress: false  affinity: {}  tolerations: []  openshift: false  githubcreationHook: false  gitea:  enabled: false  ###############################################################################  installOpenLdap: true  auth:  saml: false  oauth2: false  sapor:  config:  encrypt:  enabled: false  ## ldap configuration used in oes-gate, oes-platform and spinnaker gate for authentication and authorization  ldap:  enabled: false  ###############################################################################  preDeleteHelmHooks: false  minio:  enabled: true  ###############################################################################  autoConfiguration:  enabled: false ##################################################### # Centralized Monitoring #####################################################  enableCentralMonitoring: false ############################################################################### db:  enabled: false ############################################################################### saporgate:  enabled: false ############################################################################### # Set to true to install Elastic and Kibana enableCentralLogging: false  installationMode: None ############################################################################### spinnaker:  autoInstallSampleApps: false # Set this to FALSE if upgrading  # Initialize gitOps style Halyard  gitopsHalyard:  enabled: true  repo:  type: s3 # git, bitbucket-stash or S3, please use another template for github/stash  s3accesskey: xxxxxxxxxxxxx  s3secretkey: xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx  s3bucket: opsmx-terraform-state  secretName: opsmx-gitops-auth    halyard:  gitops:  enabled: true  image:  repository: us-docker.pkg.dev/spinnaker-community/docker/halyard  tag: 1.45.0  additionalServiceSettings:  gate.yml:  healthEndpoint: /health  kubernetes:  useExecHealthCheck: false  artifactId: us-docker.pkg.dev/spinnaker-community/docker/gate:master-latest-unvalidated-ubuntu  deck.yml:  artifactId: us-docker.pkg.dev/spinnaker-community/docker/deck:master-latest-unvalidated-ubuntu  clouddriver-caching.yml:  artifactId: us-docker.pkg.dev/spinnaker-community/docker/clouddriver:master-latest-unvalidated-ubuntu  clouddriver.yml:  artifactId: us-docker.pkg.dev/spinnaker-community/docker/clouddriver:master-latest-unvalidated-ubuntu  clouddriver-rw.yml:  artifactId: us-docker.pkg.dev/spinnaker-community/docker/clouddriver:master-latest-unvalidated-ubuntu  clouddriver-ro.yml:  artifactId: us-docker.pkg.dev/spinnaker-community/docker/clouddriver:master-latest-unvalidated-ubuntu  clouddriver-ro-deck.yml:  artifactId: us-docker.pkg.dev/spinnaker-community/docker/clouddriver:master-latest-unvalidated-ubuntu  echo.yml:  artifactId: us-docker.pkg.dev/spinnaker-community/docker/echo:master-latest-unvalidated-ubuntu  echo-scheduler.yml:  artifactId: us-docker.pkg.dev/spinnaker-community/docker/echo:master-latest-unvalidated-ubuntu  echo-worker.yml:  artifactId: us-docker.pkg.dev/spinnaker-community/docker/echo:master-latest-unvalidated-ubuntu  fiat.yml:  artifactId: us-docker.pkg.dev/spinnaker-community/docker/fiat:master-latest-unvalidated-ubuntu  front50.yml:  artifactId: us-docker.pkg.dev/spinnaker-community/docker/front50:master-latest-unvalidated-ubuntu  igor.yml:  artifactId: us-docker.pkg.dev/spinnaker-community/docker/igor:master-latest-unvalidated-ubuntu  kayenta.yml:  artifactId: us-docker.pkg.dev/spinnaker-community/docker/kayenta:master-latest-unvalidated-ubuntu  orca.yml:  artifactId: us-docker.pkg.dev/spinnaker-community/docker/orca:master-latest-unvalidated-ubuntu  rosco.yml:  artifactId: us-docker.pkg.dev/spinnaker-community/docker/rosco:master-latest-unvalidated-ubuntu |
| --- |

# 