

Cloud Run Implementation Guide



Table of Contents

Introduction	3
Configuration Details	3
Implementation Details	3
Service Name & Revision Name	5
Traffic Split	5
Deleting Active Server Group	5
Testing	6
Service Creation	7
Second Revision Creation	11
Spitting Traffic between the Revisions	13
Deleting a Revision	14
Deleting a Service	17
Service Creation thru Artifact	18
Sample Cloudrun Deployment Pipeline	22

Introduction

This document is intended to provide details of the implementation changes that are added to the Spinnaker code base, in order to support Google Cloud Run as a cloud provider.

Configuration Details

The following configuration details need to be added in the **clouddriver-local.yml** file before proceeding with the implementation.

```
cloudrun:
  enabled: true
  accounts:
    - name: my-cloudrun-account
      requiredGroupMembership: []
      permissions: {}
      project: my-orbit-project-71824
      jsonPath: /home/ubuntu/.gcp/my-orbit-project-71824-ce91bf940a59.json
```

Implementation Details

The following Spinnaker microservices have been modified to support Cloud Run.

- Cloud driver
- Orca
- Halyard
- Deck

Cloud Run deployment includes Service and Revision components. The following table represents the mapping between CloudRun and Spinnaker resources:

Spinnaker Resource	Cloud Run Resource
Account	GCP Project
Load Balancer	Cloud Run Service
Server Group	Cloud Run Revision

Instance	Cloud Run Container Instance
----------	------------------------------

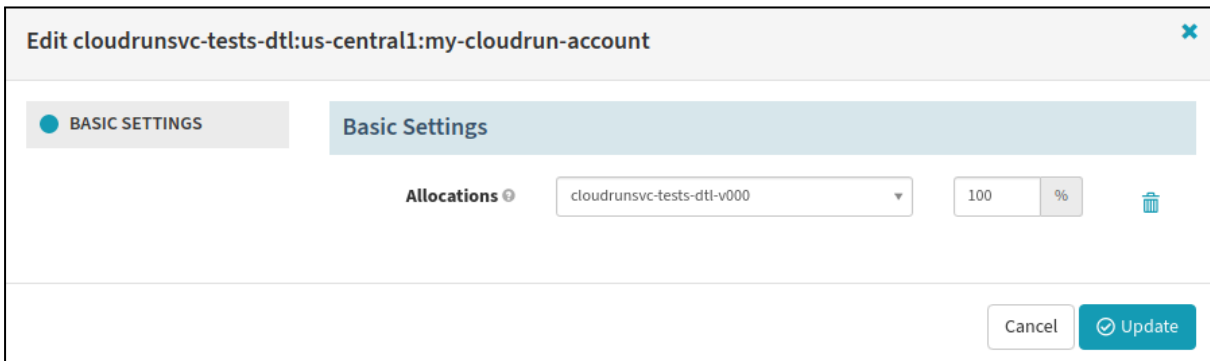
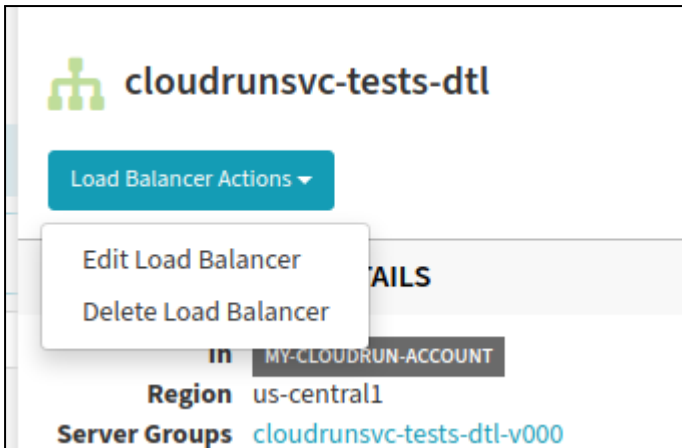
In Spinnaker, a user can create a **Server Group** from the CLUSTERS pane by clicking the **Create Server Group** button. Similarly, a **Load Balancer** can be created from the LOAD BALANCERS pane. But, as in Google App engine, the load balancer (Cloud Run Service) cannot be created without a Cloud Run Revision. So, **Create Server Group** action leads to the creation of Cloud Run Service and Revision.

A service yaml needs to be added in the **Service Yaml** text box that appears when the **Create Server Group** button is clicked. When the **Create** button is clicked, a series of actions take place resulting in the creation of Cloud Run Service and Revision resources in the Google Cloud and display of the information related to these resources in the Spinnaker UI.

The screenshot shows a 'Create New Server Group' dialog box. On the left, there are two tabs: 'BASIC SETTINGS' (active) and 'SERVICE YAML'. Under 'BASIC SETTINGS', there are three input fields: 'Account' with a dropdown menu showing 'my-cloudrun-account', 'Stack' with a text input containing 'test', and 'Detail' with an empty text input. Below these fields, a message box states 'Your server group will be in the cluster: cloudrunsvc-test (new cluster)'. The 'SERVICE YAML' tab is currently empty. At the bottom right of the dialog, there are two buttons: 'Cancel' and 'Create'.

After creating the server group, the load balancer gets displayed with service details and options such as

- **Edit Load Balancer** (for splitting traffic)
- **Delete Load Balancer** (to delete a Service)



Service Name & Revision Name

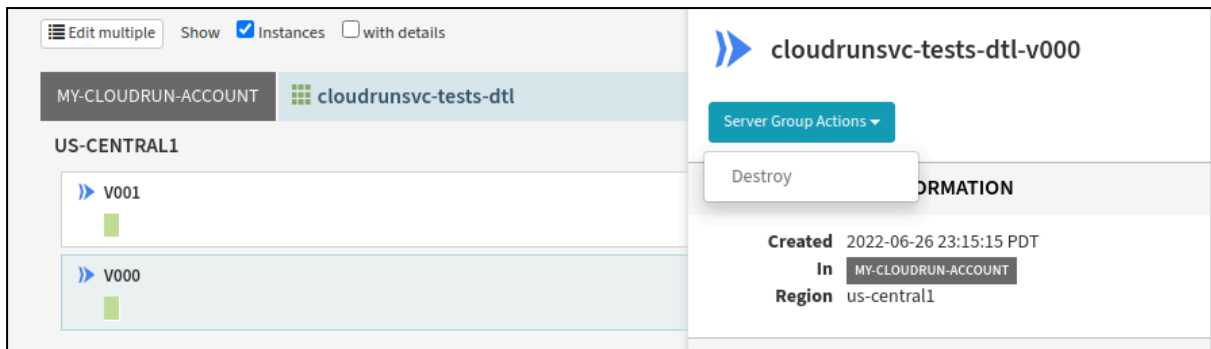
The Service name & Revision names are managed by Spinnaker irrespective of their values in the Service Yaml. This helps Spinnaker to manage the segregation of server groups and clusters. So any other names given to the service and revision in the Service yaml will be overridden by Spinnaker using the Account name, Stack and Detail field values.

Traffic Split

There is a section in the Service yaml for representing traffic split but as part of this implementation we are ignoring the section because the user needs to know the existing traffic routing in order to change it. And if a user doesn't add the traffic section in the Service yaml, a gcloud command will route 100% traffic to the newly created revision which is not needed. Hence the better way is to preserve the existing traffic split until the user performs an explicit split using **Edit Load Balancer** action to route some or total percentage of traffic to the newly created Revision.

Deleting Active Server Group

A Revision cannot be deleted if it's serving traffic. Hence the **Destroy** option is disabled for those server groups having non-zero traffic percent.



Testing

Apart from the unit test cases, the following manual tests have been done in the development environment.

- [Service Creation](#)
- [Second Revision Creation](#)
- [Spitting Traffic between the Revisions](#)
- [Deleting a Revision](#)
- [Deleting a Service](#)
- [Service Creation thru Artifact](#)

The setup used for testing is given below:

- Google Project: **my-orbit-project-71824** (ID: 135005621049)
- Service Account:
spinnaker-cloudrun-account@my-orbit-project-71824.iam.gserviceaccount.com
- Image: us-docker.pkg.dev/cloudrun/container/hello

The Service Account requires permissions to deploy the CloudRun Services. A Service Account credentials key(a json file) is added in Spinnaker in the form of cloudrun account as given below:

```
cloudrun:
  enabled: true
  accounts:
    - name: my-cloudrun-account
      requiredGroupMembership: []
      permissions: {}
      project: my-orbit-project-71824
      jsonPath:
/home/ubuntu/.gcp/my-orbit-project-71824-ce91bf940a59.json
```

Service Creation

- Create an application in Spinnaker - cloudrunsvc.
- Enter the values in the **Stack** and **Detail** fields.

Create New Server Group

BASIC SETTINGS

Basic Settings

Account: my-cloudrun-account

Stack: test

Detail: dl

Your server group will be in the cluster:
cloudrunsvc-test-dl (new cluster)

SERVICE YAML

Service Yaml

```
apiVersion: serving.knative.dev/v1
kind: Service
metadata:
  name: cloudrunsvc
  namespace: '135005621049'
labels:
```

Cancel Create

- Create a server group using the following service yaml:

```
apiVersion: serving.knative.dev/v1
kind: Service
metadata:
  name: cloudrunsvc
  namespace: '135005621049'
labels:
  cloud.googleapis.com/location: us-central1
annotations:
  run.googleapis.com/client-name: cloud-console
  serving.knative.dev/creator:
spinnaker-cloudrun-account@my-orbit-project-71824.iam.gserviceaccount.com
  serving.knative.dev/lastModifier:
spinnaker-cloudrun-account@my-orbit-project-71824.iam.gserviceaccount.com
  client.knative.dev/user-image:
us-docker.pkg.dev/cloudrun/container/hello
  run.googleapis.com/ingress: all
  run.googleapis.com/ingress-status: all
spec:
```

```

template:
  metadata:
    name: cloudrunsvc-hkb-v001
    annotations:
      run.googleapis.com/client-name: cloud-console
      autoscaling.knative.dev/minScale: '1'
      autoscaling.knative.dev/maxScale: '3'
  spec:
    containerConcurrency: 80
    timeoutSeconds: 200
    serviceAccountName:
spinnaker-cloudrun-account@my-orbit-project-71824.iam.gserviceaccount.com
    containers:
      - image: us-docker.pkg.dev/cloudrun/container/hello
        ports:
          - name: http1
            containerPort: 8080
    resources:
      limits:
        cpu: 1000m
        memory: 256Mi

```

- Verify that the operation is successful.

Creating your Server Group

- ✓ Determine Source Server Group 00:00
- ✓ Determine Health Providers 00:00
- ✓ Create Server Group 00:00
- ✓ Monitor Deploy 00:20
- ✓ Force Cache Refresh 00:00
- ✓ Wait For Up Instances 00:20
- ✓ Force Cache Refresh 00:00
- ✓ Get Commits 00:00

🟢 **Operation succeeded!**

You can [monitor this task from the Tasks view](#).

- Verify that the **Service** and **Revision** are created in the cloud.

Cloud Run Services [+ CREATE SERVICE](#) [MANAGE CU...](#)

SERVICES JOBS PREVIEW

Filter Name : cloudrunsvc-tests-dtl Filter services

<input type="checkbox"/>	<input checked="" type="radio"/>	Name ↑	Req/sec ?	Region	Authentication ?
<input type="checkbox"/>	<input checked="" type="radio"/>	cloudrunsvc-tests-dtl	0	us-central1	REQUIRE_AUTHENT

cloudrunsvc-tests-dtl Region: us-central1 URL: https://cloudrunsvc-tests-dtl-y7jpao7s3a-uc.a.r

METRICS SLOS LOGS REVISIONS TRIGGERS DETAILS YAML PEF

[EDIT](#)

```

1  apiVersion: serving.knative.dev/v1
2  kind: Service
3  metadata:
4    name: cloudrunsvc-tests-dtl
5    namespace: '135005621049'
6    selfLink: /apis/serving.knative.dev/v1/namespaces/135005621049/services/cloudrunsvc-test
7    uid: 4ac32d0d-2cef-4f7b-a682-a958786605fb
8    resourceVersion: AAX1bEAZLPU
9    generation: 1
10   creationTimestamp: '2022-06-27T11:30:34.496539Z'
11   labels:
12     cloud.googleapis.com/location: us-central1
13   annotations:
14     run.googleapis.com/client-name: cloud-console
15     serving.knative.dev/creator: spinnaker-cloudrun-account@my-orbit-project-71824.iam.gse
16     serving.knative.dev/lastModifier: spinnaker-cloudrun-account@my-orbit-project-71824.ia
17     client.knative.dev/user-image: us-docker.pkg.dev/cloudrun/container/hello
18     spinnaker/application: cloudrunsvc
19     run.googleapis.com/ingress: all
20     run.googleapis.com/ingress-status: all
21   spec:
22     template:
23       metadata:
24         name: cloudrunsvc-tests-dtl-v000
25         annotations:
26           run.googleapis.com/client-name: cloud-console
27           autoscaling.knative.dev/minScale: '1'
28           autoscaling.knative.dev/maxScale: '3'
29       spec:
30         containerConcurrency: 80

```

cloudrunsvc-tests-dtl Region: us-central1 URL: https://cloudrunsvc-tests-dtl-y7jpao7s3a-uc

METRICS SLOS LOGS **REVISIONS** TRIGGERS DETAILS YAML

Revisions [MANAGE TRAFFIC](#)

Filter Filter revisions

Name	Traffic	Deployed	Revision URLs (tags)	Act
cloudrunsvc-tests-dtl-v000	100% (to latest)	8 minutes ago	+	⋮

- Verify if the load balancer and server group are created.

SPINNAKER Search Projects Applications Pipeline Templates Search

cloudrunsvc

Filters

Search by field

MY-CLOUDRUN-ACCOUNT cloudrunsvc-tests-dtl

ACCOUNT

US-CENTRAL1

V000

cloudrunsvc-tests-dtl-v000

Server Group Actions

SERVER GROUP INFORMATION

Created 2022-06-26 23:15:15 PDT

In MY-CLOUDRUN-ACCOUNT

Region us-central1

SIZE

Min 1

Max 3

Current 1

HEALTH

Instances 1 ▲ : 100%

SPINNAKER Search Projects Applications Pipeline Templates Search

cloudrunsvc

Filters

Show Server Groups Instances

Search by field

MY-CLOUDRUN-ACCOUNT cloudrunsvc-tests-dtl

ACCOUNT

US-CENTRAL1

cloudrunsvc-tests-dtl-v000

cloudrunsvc-tests-dtl

Load Balancer Actions

LOAD BALANCER DETAILS

In MY-CLOUDRUN-ACCOUNT

Region us-central1

Server Groups cloudrunsvc-tests-dtl-v000

TRAFFIC SPLIT

cloudrunsvc-tests-dtl-v000:

DNS

HTTPS

https://cloudrunsvc-tests-dtl-y7jpao7s3a-uc.a.run.app

Second Revision Creation

1. Create a server group using the following service yaml:

```
apiVersion: serving.knative.dev/v1
kind: Service
metadata:
  name: cloudrunsvc
  namespace: '135005621049'
  labels:
    cloud.googleapis.com/location: us-central1
  annotations:
    run.googleapis.com/client-name: cloud-console
    serving.knative.dev/creator:
spinnaker-cloudrun-account@my-orbit-project-71824.iam.gserviceaccount.com
    serving.knative.dev/lastModifier:
spinnaker-cloudrun-account@my-orbit-project-71824.iam.gserviceaccount.com
    client.knative.dev/user-image:
us-docker.pkg.dev/cloudrun/container/hello
    run.googleapis.com/ingress: all
    run.googleapis.com/ingress-status: all
spec:
  template:
    metadata:
      name: cloudrunsvc-hkb-v002
      annotations:
        run.googleapis.com/client-name: cloud-console
        autoscaling.knative.dev/minScale: '1'
        autoscaling.knative.dev/maxScale: '3'
    spec:
      containerConcurrency: 80
      timeoutSeconds: 200
      serviceName:
spinnaker-cloudrun-account@my-orbit-project-71824.iam.gserviceaccount.com
      containers:
        - image: us-docker.pkg.dev/cloudrun/container/hello
          ports:
            - name: http1
              containerPort: 8080
          resources:
            limits:
              cpu: 1000m
              memory: 256Mi
```

2. Verify if the second revision is created.

The screenshot shows the Cloud Run 'Service details' page for 'cloudrunsvc-tests-dtl'. The service is in the 'us-central1' region with a URL of 'https://cloudrunsvc-tests-dtl-y7jpa...'. The 'REVISIONS' tab is selected, showing a table of revisions. The first revision, 'cloudrunsvc-tests-dtl-v001', is selected and has 0% traffic. The second revision, 'cloudrunsvc-tests-dtl-v000', is active and has 100% traffic. A 'MANAGE TRAFFIC' button is visible above the table.

Name	Traffic	Deployed	Revision URLs (tags)	Act
cloudrunsvc-tests-dtl-v001	0%	12 minutes ago	+	⋮
cloudrunsvc-tests-dtl-v000	100%	42 minutes ago		⋮

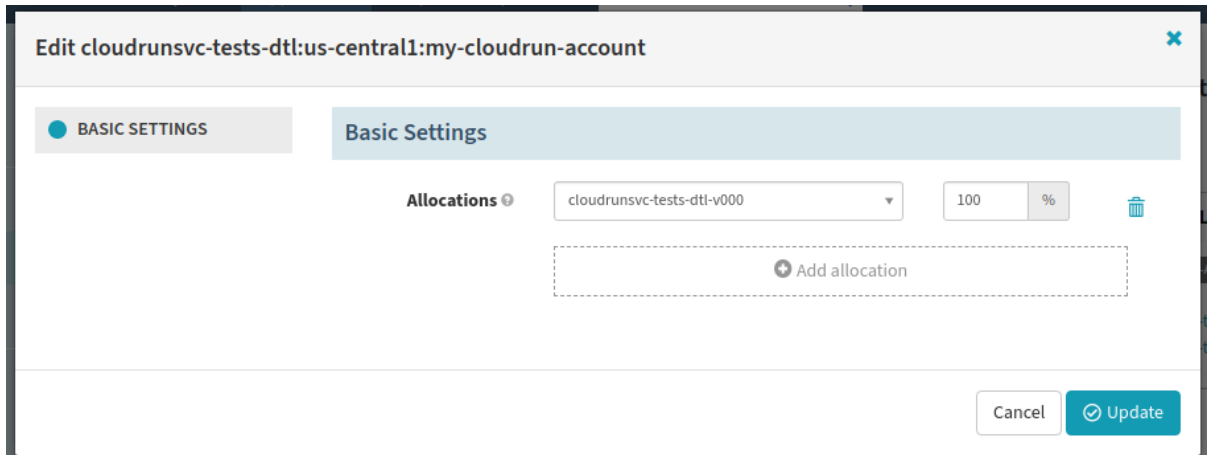
3. Verify if the changes are reflected in Spinnaker.

The first screenshot shows the Spinnaker configuration page for 'cloudrunsvc-tests-dtl'. The 'ACCOUNT' is 'MY-CLOUDRUN-ACCOUNT' and the 'REGION' is 'US-CENTRAL1'. Under 'US-CENTRAL1', two revisions are listed: 'V001' and 'V000'. The 'Show' options are 'Instances' (checked) and 'with details' (unchecked).

The second screenshot shows the same configuration page but with 'Server Groups' (checked) and 'Instances' (unchecked) selected. The 'US-CENTRAL1' section now shows two server groups: 'cloudrunsvc-tests-dtl-v000' and 'cloudrunsvc-tests-dtl-v001'. The 'STACK' field is also visible at the bottom.

Spitting Traffic between the Revisions

1. Choose **Edit Load Balancer** from the Load Balancer Actions.



Edit cloudrunsvc-tests-dtl:us-central1:my-cloudrun-account

BASIC SETTINGS Basic Settings

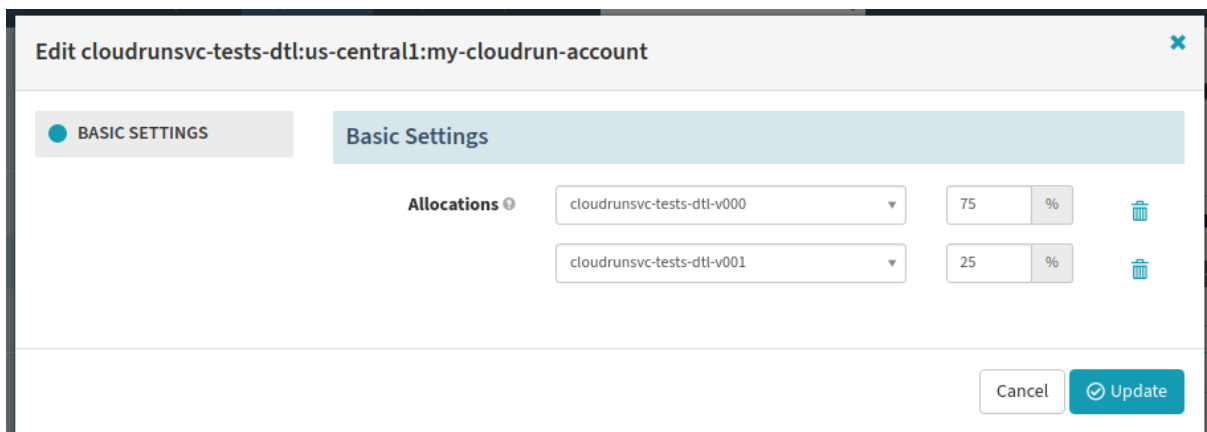
Allocations ⓘ

cloudrunsvc-tests-dtl-v000	100	%	
----------------------------	-----	---	--

[+ Add allocation](#)

Cancel [Update](#)

2. Allocate traffic between the two revisions equally and click **Update**.



Edit cloudrunsvc-tests-dtl:us-central1:my-cloudrun-account

BASIC SETTINGS Basic Settings

Allocations ⓘ

cloudrunsvc-tests-dtl-v000	75	%	
cloudrunsvc-tests-dtl-v001	25	%	

Cancel [Update](#)

3. Verify if the split happened properly in the cloud console.

Cloud Run Service details [EDIT AND DEPLOY NEW REVISION](#) [SET UP CONTINUOUS DEPLOYMENT](#)

cloudrunsvc-tests-dtl Region: us-central1 URL: <https://cloudrunsvc-tests-dtl-y7jpao7s3a-uc.a.run.app>

METRICS SLOS LOGS **REVISIONS** TRIGGERS DETAILS YAML PERMISSIONS

Revisions [MANAGE TRAFFIC](#)

Filter Filter revisions

<input type="radio"/>	Name	Traffic	Deployed	Revision URLs (tags)	Actions
<input checked="" type="radio"/>	cloudrunsvc-tests-dtl-v001	25%	16 minutes ago	+	
<input type="radio"/>	cloudrunsvc-tests-dtl-v000	75%	46 minutes ago		

4. Verify if the Load Balancers details are updated with the latest traffic split.

Show Server Groups Instances

MY-CLOUDRUN-ACCOUNT **cloudrunsvc-tests-dtl**

US-CENTRAL1

- [cloudrunsvc-tests-dtl-v001](#)
- [cloudrunsvc-tests-dtl-v000](#)

cloudrunsvc-tests-dtl

[Load Balancer Actions](#)

LOAD BALANCER DETAILS

In **MY-CLOUDRUN-ACCOUNT**

Region us-central1

Server Groups [cloudrunsvc-tests-dtl-v001](#)
[cloudrunsvc-tests-dtl-v000](#)

TRAFFIC SPLIT

cloudrunsvc-tests-dtl-v000:	75
cloudrunsvc-tests-dtl-v001:	25

DNS

HTTPS

<https://cloudrunsvc-tests-dtl-y7jpao7s3a-uc.a.run.app>

Deleting a Revision

1. Select a Server Group having zero traffic and click **Destroy**. Before this step, another version was created and traffic was split between the latest two revisions(v002 & v001).

Edit multiple Show Instances with details

MY-CLOUDRUN-ACCOUNT cloudrunsvc-tests-dtl

US-CENTRAL1

- V002
- V001
- V000

Server Group Actions

- Destroy
- Disabled

SERVER GROUP INFORMATION

- Created** 2022-06-26 23:15:15 PDT
- In** MY-CLOUDRUN-ACCOUNT
- Region** us-central1

SIZE

Really destroy cloudrunsvc-tests-dtl-v000?

Reason

2. Verify that the deletion is successful.

Destroying cloudrunsvc-tests-dtl-v000

- ✓ Determine Health Providers 00:00
- ✓ Disable Server Group 00:00
- ✓ Monitor Server Group 00:00
- ✓ Wait For Down Instances 00:00
- ✓ Wait For Server Group Disabled 00:00
- ✓ Force Cache Refresh 00:00
- ✓ Destroy Server Group 00:00
- ✓ Monitor Server Group 00:05
- ✓ Wait For Destroyed Server Group 01:00

✓ **Operation succeeded!**

You can [monitor this task from the Tasks view.](#)

Edit multiple Show Instances with details [Create Server Group](#)

MY-CLOUDRUN-ACCOUNT **cloudrunsvc-tests-dtl** 2 ▲ : 100%

US-CENTRAL1

>> V002 1 ▲ : 100%

■

>> V001 1 ▲ : 100%

■

3. Also, verify that the **Revision** is deleted from the Cloud.

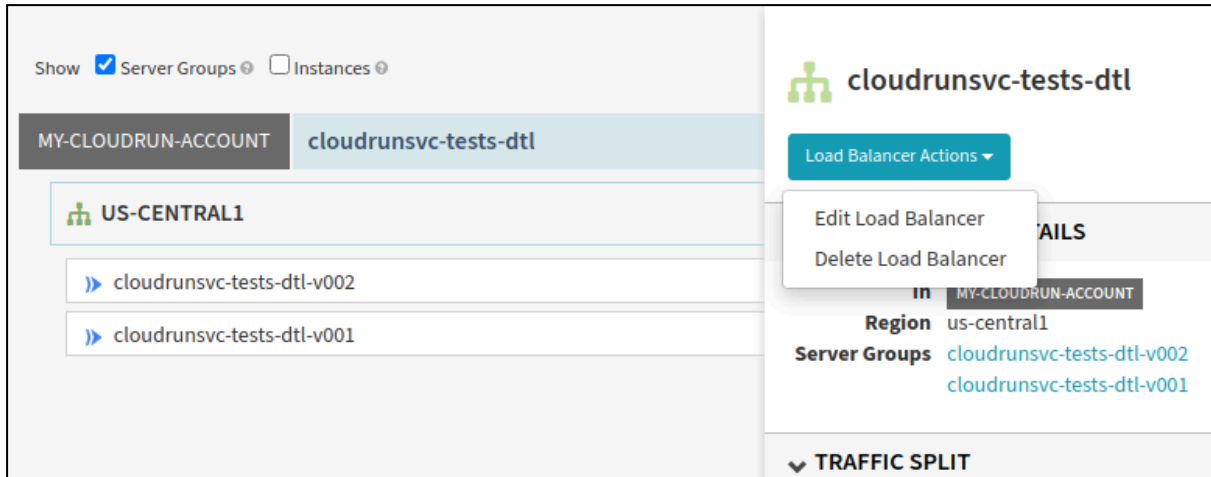
Revisions [MANAGE TRAFFIC](#)

Filter Filter revisions

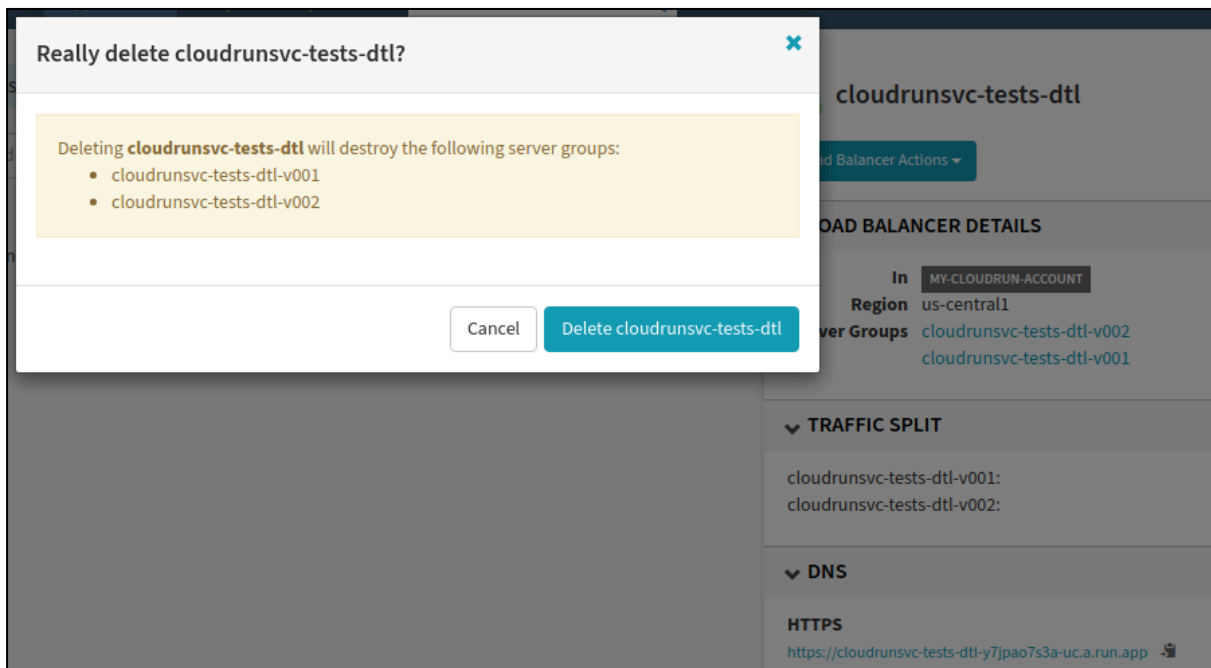
<input type="radio"/>	Name	Traffic	Deployed	Revision URLs (tags) ?	Action
<input checked="" type="radio"/>	cloudrunsvc-tests-dtl-v002	25%	17 minutes ago	+	⋮
<input type="radio"/>	cloudrunsvc-tests-dtl-v001	75%	46 minutes ago		⋮

Deleting a Service

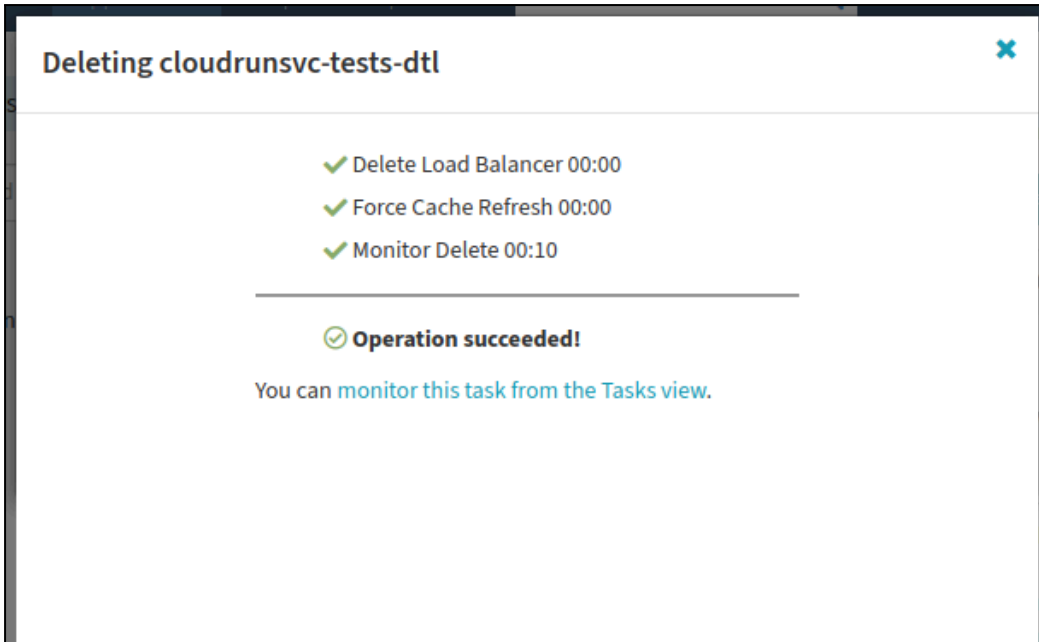
1. Select the service that you need to delete and click **Delete Load Balancer**.



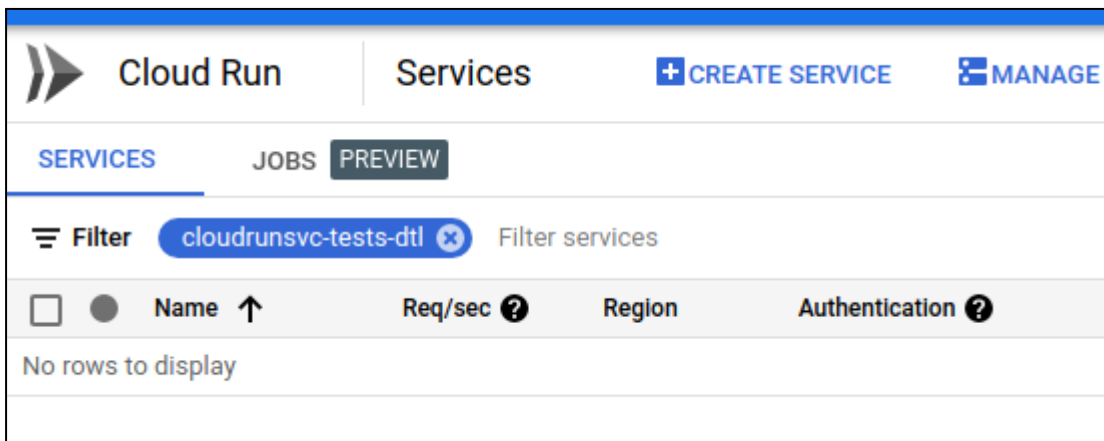
2. A popup appears asking for confirmation.



3. Once confirming the deletion, verify that the delete operation is successful in Spinnaker.

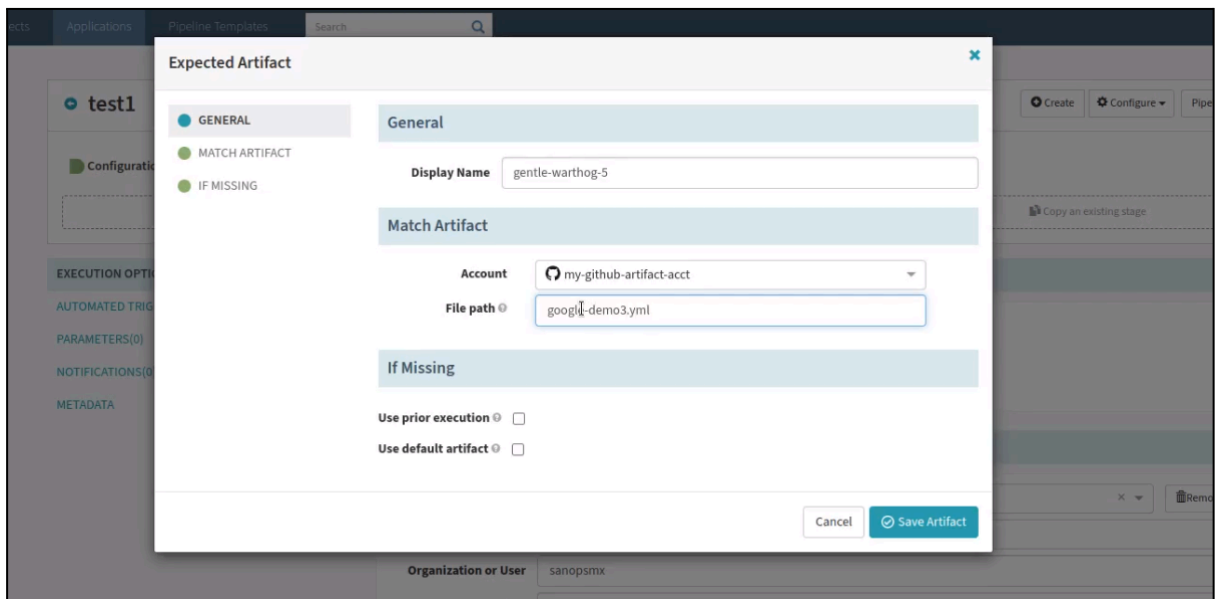
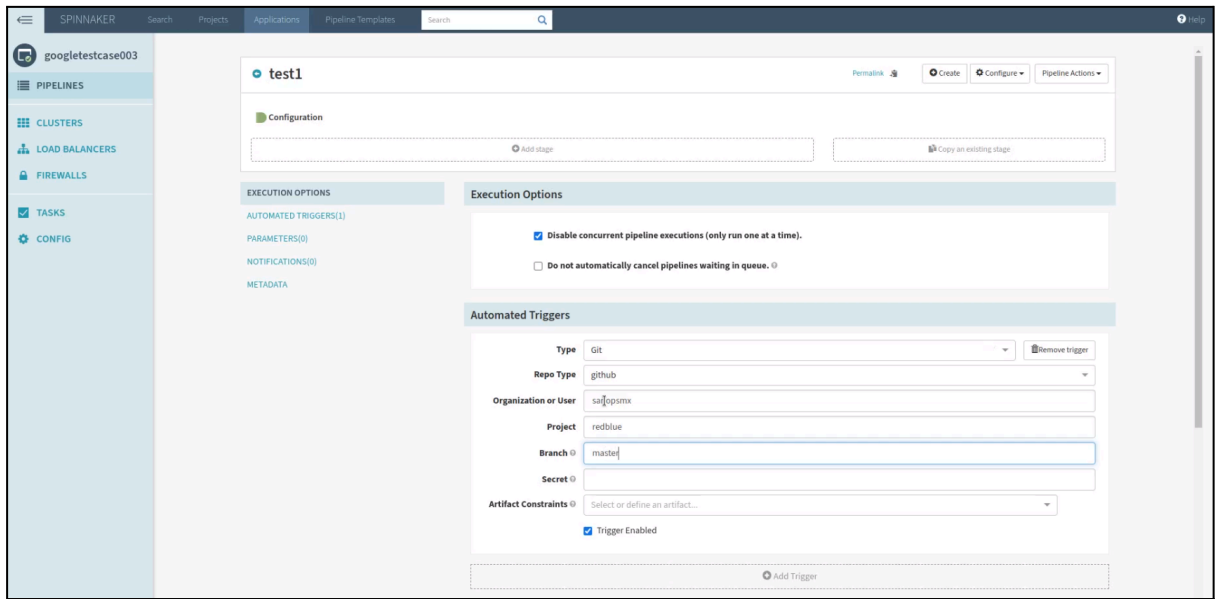


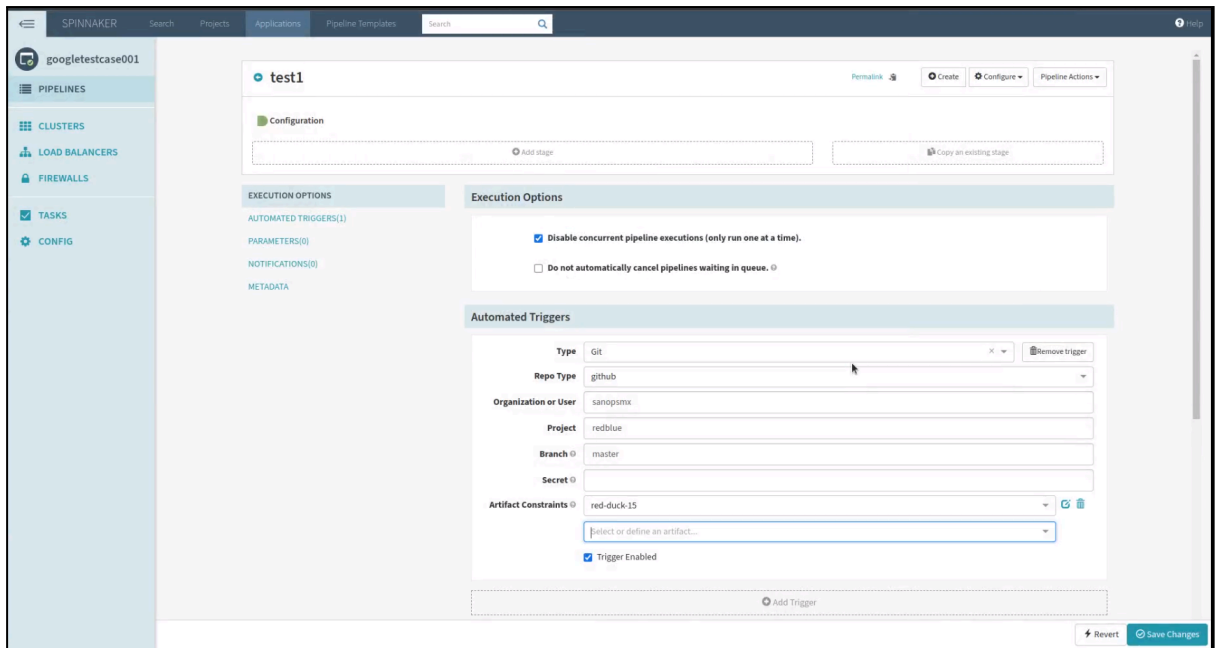
4. Also, verify if the **Service** is deleted from the cloud.



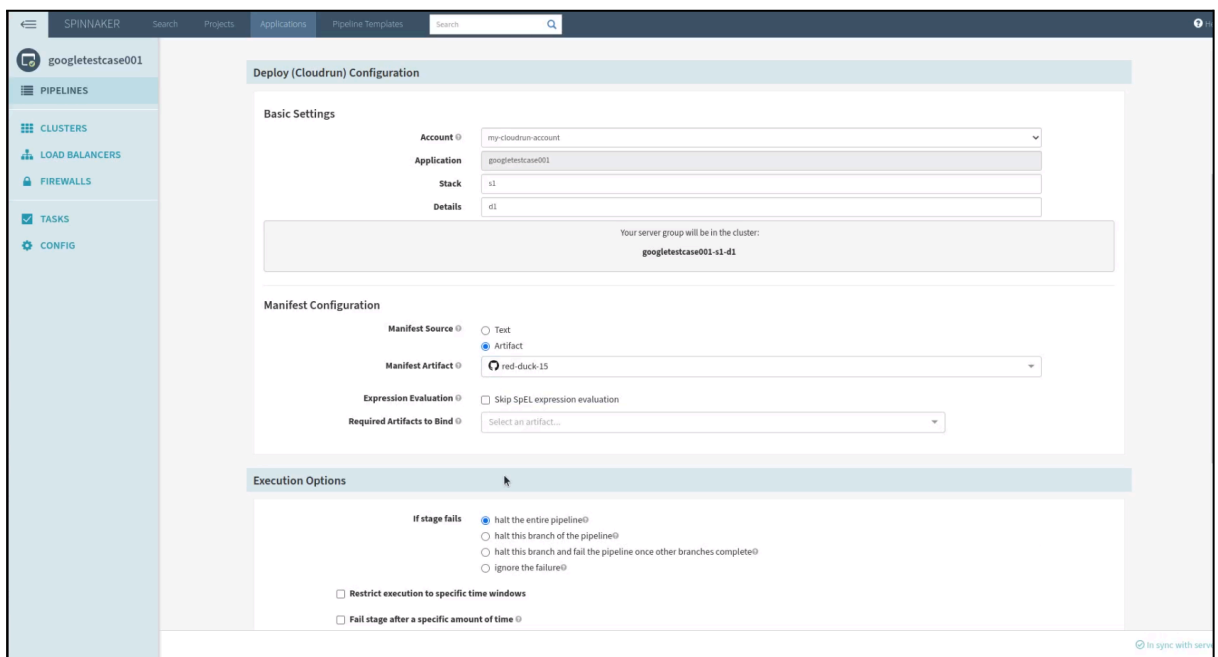
Service Creation thru Artifact

1. Create an application in Spinnaker - googletestcase001.
2. Create a pipeline named test1.
3. Enter the artifact details in the configuration section, as shown in the screenshots below:

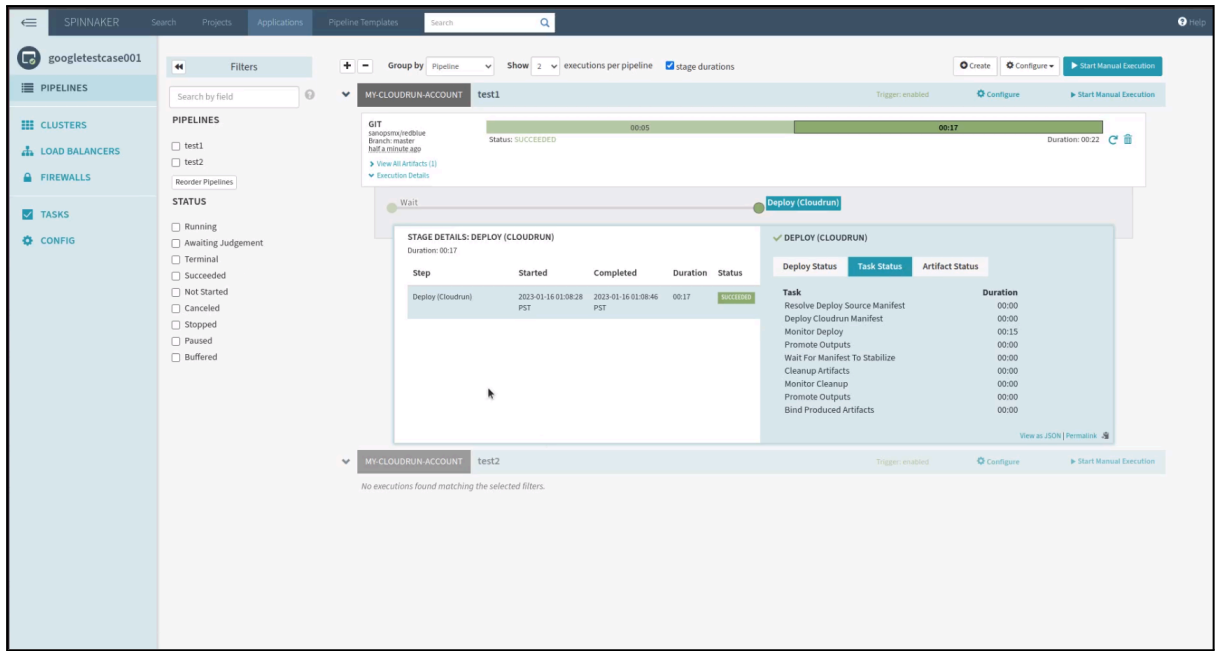




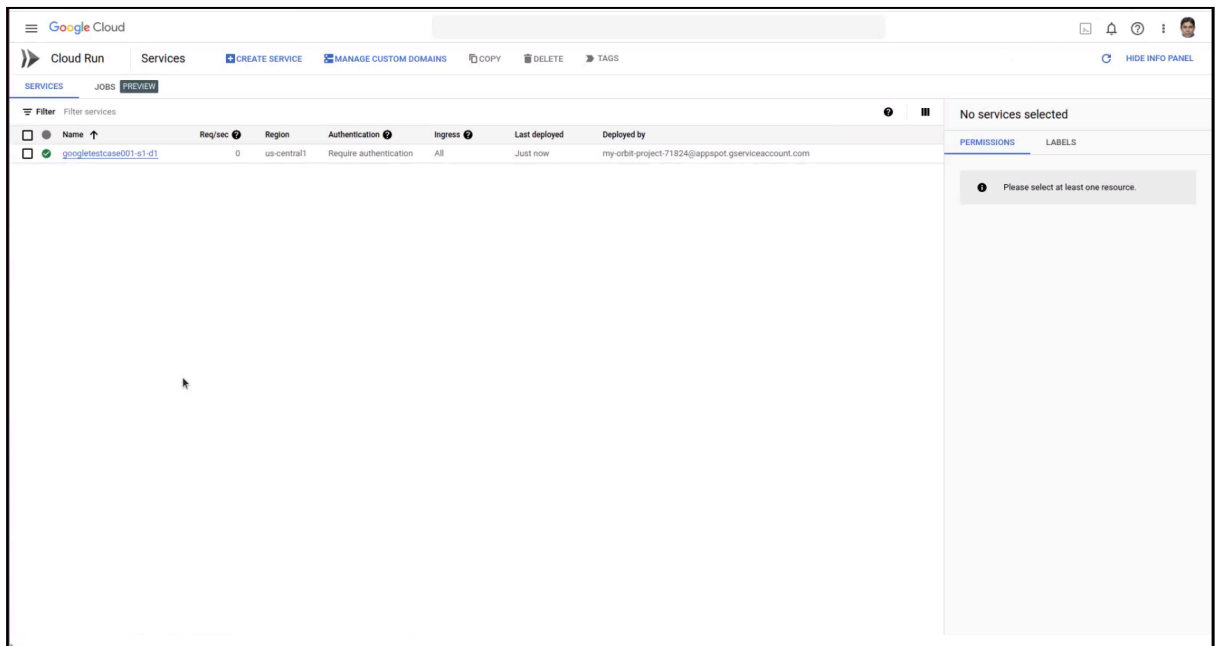
4. Add the Deploy(Cloudrun) stage to the existing pipeline and enter the values in the **Stack** and **Detail** fields as shown in the screenshot below.



5. Save the pipeline and trigger it.
6. Ensure that the deploy(Cloudrun) stage is run successfully and all the tasks are completed.



7. Verify that the service(googletestcase001-s1-d1) is created successfully.



```

1 apiVersion: serving.knative.dev/v1
2 kind: Service
3 metadata:
4   name: googletestcase001-s1-d1
5   namespace: '135085621049'
6   selfLink: /apis/serving.knative.dev/v1/namespaces/135085621049/services/googletestcase001-s1-d1
7   uid: c2a6ab34-4ac2-452f-93cc-83bc942cd7bd
8   resourceVersion: AAXyKe399/0
9   generation: 1
10  creationTimestamp: '2023-01-16T09:08:33.996259Z'
11  labels:
12    cloud.googleapis.com/location: us-central1
13  annotations:
14    run.googleapis.com/client-name: cloud-console
15    serving.knative.dev/creator: my-orbit-project-71824@agppot.gserviceaccount.com
16    serving.knative.dev/lastModifier: my-orbit-project-71824@agppot.gserviceaccount.com
17    client.knative.dev/user-image: gcr.io/my-orbit-project-71824/test-image:tag1
18    spinnaker/application: googletestcase001
19    run.googleapis.com/operation-id: 329336ac-f455-4af8-8eee-195f902aa57b
20    run.googleapis.com/ingress: all
21    run.googleapis.com/ingress-status: all
22  spec:
23    template:
24      metadata:
25        name: googletestcase001-s1-d1-v000
26        annotations:
27          run.googleapis.com/client-name: cloud-console
28          autoscaling.knative.dev/maxScale: '100'
29      spec:
30        containerConcurrency: 80
31        timeoutSeconds: 300
32        serviceName: spinnaker-cloudrun-account@my-orbit-project-71824.iam.gserviceaccount.com
33        containers:
34          - image: gcr.io/my-orbit-project-71824/test-image:tag1
35            ports:
36              - name: http
37                containerPort: 8080
38            resources:
39              limits:
40                cpu: 1000m
41                memory: 512Mi
42            traffic:
43              - percent: 100

```

To know more about artifacts, refer to the below spinnaker documentation.

1. <https://spinnaker.io/docs/reference/ref-artifacts/types/>
2. https://spinnaker.io/docs/setup/other_config/artifacts/

Sample Cloudrun Deployment Pipeline



A sample Cloudrun deployment pipeline is given below. It includes the following stages:

- **Trigger on a GitHub webhook** - github webhook is triggered by a push to the forked repository.
- **Deploy** - the updated app is deployed in Cloudrun.
- **Edit Load Balancer** - the traffic is split between the two server groups.
- **Manual Judgement** - verification is done to know if all the traffic can be moved to the new server group. This stage could be a stand-in for integration tests or canary analysis.
- **Enable** - 100% of traffic is sent to the new server group.

- **Wait** - before destroying the old server group 2 minutes of wait time is followed, to be sure everything is OK. In a real deployment pipeline, it's likely that this wait time would be longer (an hour or more).
- **Destroy** - the old server group is destroyed.

To know more about deployment pipelines, refer to the below spinnaker documentation.

<https://spinnaker.io/docs/guides/tutorials/codelabs/appengine-source-to-prod/>