Artifact Repository Clean Policy Application Infrastructure

This uses a Terraform configuration to set up an automated artifact repository cleanup system in Google Cloud Platform, utilising Cloud Scheduler, Pub/Sub, Cloud Build, and custom IAM roles to manage artifact retention policies across multiple repositories.

Artifact Cleanup Policy Architecture

The artifact repository cleanup system is orchestrated through a series of Google Cloud Platform services.

A Cloud Scheduler job, named "artifact-cleanup-policy-applications-scheduler," runs on the 1st and 15th of each month at 12 PM, triggering a → Pub/Sub topic. This topic activates a → Cloud Build trigger, which executes a cleanup script using a GCP image.

The build generates a temporary JSON policy and applies it to all artifact repositories, implementing three distinct cleanup strategies.

For logging and auditing purposes, Cloud Build logs are stored in a dedicated storage bucket.

The entire process operates under the principle of least privilege, utilising a custom service account with specific permissions granted through a bespoke Artifact Registry Policy Admin role.

Repository Cleanup Strategies

Three targeted strategies ensure effective artifact repository maintenance:

- **Delete Artifacts Older than 30 Days**: Automatically removes artifacts exceeding a 30-day retention period to prevent unnecessary storage usage.
- **Retain Latest and Tagged Artifacts**: Safeguards artifacts tagged with "latest" or prefixed with "keep," ensuring critical builds remain accessible.
- **Retain Last Five Artifacts**: Preserves the five most recently created artifacts, maintaining a minimal yet functional version history for repositories.

These strategies are dynamically applied across all repositories using a temporary JSON policy generated during each cleanup cycle.

Terraform Setup for Automation

The Terraform configuration automates the setup of the artifact cleanup system, defining key resources such as the Cloud Scheduler job, Pub/Sub topic, and Cloud Build trigger. It creates a custom IAM role with specific permissions for artifact registry management and assigns it to a dedicated service account. The configuration also provisions a storage bucket for logging, enabling versioning for audit trails.

- Cloud Scheduler job runs bi-monthly using cron expression `0 12 1,15 * *`
- Cloud Build trigger executes a bash script to dynamically create and apply cleanup policies
- Custom IAM role includes permissions for listing, deleting, getting, and updating artifact repositories
- Service account `artifact-registry-cleanup-policy-sa` is granted the custom role for least-privilege access

GCP Console Implementation

Although the set up is using Terraform for the upkeep, to allow a better understanding, the below will allow you to create the artifact repository cleanup system in the Google Cloud Console:

1. Enable the necessary APIs: Cloud Scheduler, Cloud Build, Pub/Sub, and Artifact Registry.

2. Create a Pub/Sub topic named "artifact-cleanup-policy-application-topic".
   - Here I create this with ‘add a default subscription’ though I removed this after creation as its not required.
   - I used a Google Managed encryption key.
3. Set up a Cloud Scheduler job:
   - Navigate to the Cloud Scheduler page and click "Create Job".
   - Name it "artifact-cleanup-policy-applications-scheduler".
   - Set the frequency to `0 12 1,15 * *` (1st and 15th of each month at 12 PM).
   - Choose Pub/Sub as the target and select the topic created in step 2.
   - All else was default.
4. Create a Cloud Build trigger:
   - Go to Cloud Build → Triggers and click "Create Trigger".
   - Name it `artifact-cleanup-policy-application-trigger`.
   - Under Events, Choose in response to: Pub/Sub message.
   - Here configure it to be triggered by the Pub/Sub topic created earlier in the subscription option.
   - Choosing Inline will allow you to access the inline IDE.
     The code used is: (Yes those extra blank lines are reqd)

```
steps:
  - name: google/cloud-sdk
    args:
      - '-c'
      - >
        policytemp=$(cat <<EOF

        [
          {
            "name": "Delete Artifacts Older than 30 Days",
            "action": {"type": "delete"},
            "condition": {"olderThan": "30d"}
          },
          {
            "name": "Retain Latest and Keep Tagged Artifacts",
            "action": {"type": "keep"},
            "condition": {
              "tagPrefixes": ["latest","keep"],
              "versionNamePrefixes": ["latest", "keep"]
            }
          },
          {
            "name": "Retain Last 5 artifacts",
            "action": {"type": "keep"},
            "mostRecentVersions": {"keepCount": 5}
          }
        ]

        EOF

        )

        echo "$policytemp" > policy.json

        repos=$(gcloud artifacts repositories list
        --format='table(name,LOCATION)' | tail -n +2)

        while read -r name location; do gcloud artifacts repositories
```

```
        set-cleanup-policies "$name" --location="$location" --policy=policy.json
        --no-dry-run; done <<< "$repos"
    entrypoint: bash
logsBucket: 'gs://artifact-cleanup-policy-application-code-bucket/logs'
options:
  logging: GCS_ONLY
```

5. Create a custom IAM role with the necessary permissions for artifact management.
   - Name: `artifact-registry-cleanup-policy-admin`
   - Here you can click + Add Permissions and add
     - `artifactregistry.locations.list`
     - `artifactregistry.repositories.delete`
     - `artifactregistry.repositories.get`
     - `artifactregistry.repositories.list`
     - `artifactregistry.repositories.update`
6. Create a service account named `artifact-registry-cleanup-sa` (Under IAMs).
   - On Continue, select the `artifact-registry-cleanup-policy-admin`
   - You can now click through to end.

TEST:

You can view and edit the run under the cloud build triggers.

Select the Trigger and click RUN at the right.

You can then either click show (from the pop up) or History and the present running build