Tag Based Protection using gCloud for Vaulted Backups

Ashika Ganesh | Last Updated: Nov 17, 2024

Overview

This document provides a way to manage backups for your Google Compute Engine Virtual Machines (VMs) using tags. By leveraging the provided script and Google Cloud Shell, you can automate the association and removal of backup plans based on VM tags, simplifying backup management and ensuring consistent protection for your dynamic cloud environments.

This guide will enable you to:

- **Automate backup association:** Easily include or exclude VMs from backup plans based on their assigned tags. This eliminates the need to manually configure individual VMs, saving time and reducing the risk of errors.
- **Simplify backup management:** Control backup policies for large numbers of VMs efficiently by grouping them with tags representing application names, environments (production, development, testing), or criticality levels.
- **Enhance protection:** Ensure that your critical VMs are always protected by dynamically associating them with backup plans based on their tags.

Set up and Permissions

You will need to obtain the following Role on the Project where your VMs exist with tags:

- Tag Viewer
- Backup and DR Backup User

Please ensure that:

1. The backup vault service agent for the vault you intend to use to backup your VMs has the `roles/backupdr.computeEngineOperator` role in the VM project. If you do not

have the correct Role for the backup vault provided in your VM Project, the script will succeed but you will encounter a failure error.
2. Both projects have the necessary APIs enabled
3. The user running the script has the required permissions in both projects

Getting Started

Follow the below steps to apply your protection:

1. Create and save your **backup_script.sh.**

2. **Open Google Cloud Shell**:

    1. Navigate to Google Cloud Console.
    2. Open the Cloud Shell by clicking the terminal icon in the top-right corner.

3. **Open the Cloud Shell File Upload Dialog**:

    1. In the Cloud Shell terminal, click the **three vertical dots** in the top-right corner of the Cloud Shell window.
    2. Select **Upload**.
4. **Select Your File**:

    1. Choose `backup_script.sh` from your local machine.
    2. The file will upload to the home directory (`~/`) in the Cloud Shell.

5. Run `chmod +x backup_script.sh` in your cloudshell to make it executable.

6. Run your file by providing the following required **parameters**:

    1. `--unprotect:`(Optional) Use this flag and skip (b) - (d) if you intent to only remove active backup plans associated with VMs with certain tags.
    2. `--backup-project-id`: The project ID where the backup plan is located.
    3. `--location`: The region where the backup plan is located (e.g., `us-central1`).
    4. `--backup-plan`: The name of your backup plan.
    5. `--tag-key`: The tag key used to identify VMs (e.g., `environment`). See here on how to create and manage tag keys.
    6. `--tag-value`: The tag value used to identify VMs (e.g., `production`). See here on how to create and manage tag values.

    Note: You must provide at least 1 project OR 1 folder.

7. `--projects`: (Optional) A comma-separated list of project IDs to include.
8. `--folders`: (Optional) A comma-separated list of folder IDs whose projects you want to include (e.g., --folders 345678901234).
    1. NOTE: Please ensure that you have all required permissions for accessing all projects within the specific folders
    2. NOTE: Please ensure the Backup Vault Service Agent is provided the required roles on the specified folders.
9. `--exclude-projects`: (Optional) A comma-separated list of project IDs to exclude from processing (e.g., –exclude-projects projectC,projectD).

7. Leverage the below examples

```
Unset
bash ./"backup_script.sh" \
--backup-project-id my-backup-project \
--location us-central1 \
--backup-plan bp-bronze \
--tag-key environment \
--tag-value test \
--projects project1,project2,project3 \
--folders 345678901234
```

This command will automatically associate backup plan `bp-bronze` to VMs in `project1`, `project2,project3` as well as all projects under folder `345678901234`, for VMs tagged with `environment:test`.

```
Unset
bash ./"backup_script.sh" \
-- unprotect
--tag-key environment \
--tag-value test \
--projects project1
```

This command will automatically remove any backup plan association from VMs within `project1` that are tagged with `environment:test`.

Automated GCP VM Backup Management

Overview

This guide provides detailed, step-by-step instructions for implementing an automated backup management system in Google Cloud Platform (GCP). This solution automates VM backup management using Cloud Run Jobs, Cloud Scheduler, and custom scripts.

Prerequisites

1. Access to Google Cloud Console
2. Required GCP Projects:
    - Backup project (where backups will be stored)
    - Target project (where VMs are located)
3. Enable required APIs in both projects:

```
Unset
# Run these commands in both projects
gcloud services enable \
    cloudscheduler.googleapis.com \
    run.googleapis.com \
    cloudbuild.googleapis.com \
    artifactregistry.googleapis.com \
    backupdr.googleapis.com
```

Detailed Implementation Steps

1. Initial Setup in Cloud Shell

1. Open Google Cloud Console (https://console.cloud.google.com)
2. Click the "Activate Cloud Shell" button (>_ icon) at the top right
3. Wait for Cloud Shell to initialize
4. Verify you're in the correct project:

```
Unset
# Check current project
gcloud config get-value project

# If needed, set the correct project
gcloud config set project prod-demo-vault
```

2. Create Working Directory and Files

1. Create and navigate to working directory:

```
Unset
mkdir test-docker
cd test-docker
```

2. Create Dockerfile:

```
Unset
# Create new file
nano Dockerfile
```

3. Copy and paste this content into the Dockerfile (use Ctrl+V or right-click paste):

```
Unset
# Use official Google Cloud SDK image from Docker Hub
FROM google/cloud-sdk:slim

# Install required packages
RUN apt-get update && apt-get install -y jq gettext

# Copy the script into the container
COPY backup_script.sh /app/
WORKDIR /app

# Make the script executable
RUN chmod +x /app/backup_script.sh

# Add authentication wrapper script
RUN echo '#!/bin/bash\n\
# Activate service account\n\
gcloud auth list\n\
echo "Current project: $(gcloud config get-value project)"\n\
\n\
# Run the backup script with provided arguments\n\
exec /app/backup_script.sh "$@"' > /app/entrypoint.sh && \
    chmod +x /app/entrypoint.sh

# Set the entrypoint
```

```
ENTRYPOINT ["/app/entrypoint.sh"]
```

4. Save the Dockerfile:

    - Press Ctrl+X
    - Press Y to confirm saving
    - Press Enter to keep the filename

5. Create cloudbuild.yaml:

```
Unset
# Create new file
nano cloudbuild.yaml
```

6. Copy and paste this content into cloudbuild.yaml:

```
Unset
steps:
  # Configure docker authentication
  - name: 'gcr.io/cloud-builders/gcloud'
    args:
      - 'auth'
      - 'configure-docker'
      - 'us-central1-docker.pkg.dev'

  # Debug step to show environment
  - name: 'ubuntu'
    args:
      - bash
      - -c
      - |
        echo "Current directory contents:"
        ls -la
        echo "Project ID: $PROJECT_ID"
        echo "Build ID: $BUILD_ID"

  # Build the container
  - name: 'gcr.io/cloud-builders/docker'
    args:
```

```
      - 'build'
      - '-t'
      -
'us-central1-docker.pkg.dev/$PROJECT_ID/backup-scripts/backup-script:latest'
      - '--no-cache'
      - '.'

  # Push to Artifact Registry
  - name: 'gcr.io/cloud-builders/docker'
    args:
      - 'push'
      -
'us-central1-docker.pkg.dev/$PROJECT_ID/backup-scripts/backup-script:latest'

images:
  -
'us-central1-docker.pkg.dev/$PROJECT_ID/backup-scripts/backup-script:latest'

options:
  logging: CLOUD_LOGGING_ONLY
```

7. Save cloudbuild.yaml:

    - Press Ctrl+X
    - Press Y to confirm saving
    - Press Enter to keep the filename

8. Copy your backup_script.sh to Cloud Shell:

    - Click the three-dot menu in Cloud Shell
    - Select "Upload file"
    - Choose your backup_script.sh file
    - Wait for upload to complete

9. Make the backup script executable:

```
Unset
chmod +x backup_script.sh
```

3. Set Up Container Registry

1. Create Artifact Registry repository:

```
Unset
gcloud artifacts repositories create backup-scripts \
    --repository-format=docker \
    --location=us-central1 \
    --description="Repository for backup scripts"
```

2. Build and push the container:

```
Unset
gcloud builds submit --config cloudbuild.yaml
```

3. Verify the build succeeds by checking the build logs in the console or using:

```
Unset
gcloud builds list --limit=1
```

4. Create and Configure Service Account

1. Create the service account:

```
Unset
# Create service account
gcloud iam service-accounts create backup-script-sa \
    --display-name="Backup Script Service Account"
```

2. Grant permissions on backup project (prod-demo-vault):

```
Unset
# Base permissions
gcloud projects add-iam-policy-binding prod-demo-vault \
```

```
--member="serviceAccount:backup-script-sa@prod-demo-vault.iam.gserviceaccount.com" \
    --role="roles/viewer" \
    --condition=None

gcloud projects add-iam-policy-binding prod-demo-vault \

--member="serviceAccount:backup-script-sa@prod-demo-vault.iam.gserviceaccount.com" \
    --role="roles/backupdr.admin" \
    --condition=None

gcloud projects add-iam-policy-binding prod-demo-vault \

--member="serviceAccount:backup-script-sa@prod-demo-vault.iam.gserviceaccount.com" \
    --role="roles/iam.serviceAccountUser" \
    --condition=None

# Additional required permissions (added based on troubleshooting)
gcloud projects add-iam-policy-binding prod-demo-vault \

--member="serviceAccount:backup-script-sa@prod-demo-vault.iam.gserviceaccount.com" \
    --role="roles/backupdr.computeEngineBackupAdmin" \
    --condition=None

gcloud projects add-iam-policy-binding prod-demo-vault \

--member="serviceAccount:backup-script-sa@prod-demo-vault.iam.gserviceaccount.com" \
    --role="roles/compute.instanceAdmin.v1" \
    --condition=None
```

3. Grant permissions on target project (prod-demo-app):

```
Unset
# Base permissions
gcloud projects add-iam-policy-binding prod-demo-app \
```

```
--member="serviceAccount:backup-script-sa@prod-demo-vault.iam.gserviceaccount.c
om" \
    --role="roles/viewer" \
    --condition=None

gcloud projects add-iam-policy-binding prod-demo-app \

--member="serviceAccount:backup-script-sa@prod-demo-vault.iam.gserviceaccount.c
om" \
    --role="roles/resourcemanager.tagViewer" \
    --condition=None

gcloud projects add-iam-policy-binding prod-demo-app \

--member="serviceAccount:backup-script-sa@prod-demo-vault.iam.gserviceaccount.c
om" \
    --role="roles/compute.viewer" \
    --condition=None

# Additional required permissions (added based on troubleshooting)
gcloud projects add-iam-policy-binding prod-demo-app \

--member="serviceAccount:backup-script-sa@prod-demo-vault.iam.gserviceaccount.c
om" \
    --role="roles/backupdr.computeEngineBackupAdmin" \
    --condition=None

gcloud projects add-iam-policy-binding prod-demo-app \

--member="serviceAccount:backup-script-sa@prod-demo-vault.iam.gserviceaccount.c
om" \
    --role="roles/compute.instanceAdmin.v1" \
    --condition=None
```

5. Verify Backup Plan

1. List existing backup plans to get the exact name:

Unset

```
gcloud alpha backup-dr backup-plans list \
    --project=prod-demo-vault \
```

```
    --location=us-central1 \
    --format="table(name,state,description)"
```

2. Note the full backup plan name from the output (it should look like: "projects/prod-demo-vault/locations/us-central1/backupPlans/bp-bronze")

6. Create Cloud Run Job

1. Create the Cloud Run job:

```
Unset
gcloud run jobs create backup-script-job \

--image=us-central1-docker.pkg.dev/prod-demo-vault/backup-scripts/backup-script
:latest \
    --service-account=backup-script-sa@prod-demo-vault.iam.gserviceaccount.com
\
    --region=us-central1 \

--args="--backup-project-id","prod-demo-vault","--location","us-central1","--ba
ckup-plan","bp-bronze","--tag-key","environment","--tag-value","production","--
projects","prod-demo-app" \
    --max-retries=3 \
    --task-timeout=3600s
```

7. Set Up Cloud Scheduler

1. Grant additional permissions for scheduler:

```
Unset
gcloud projects add-iam-policy-binding prod-demo-vault \

--member="serviceAccount:backup-script-sa@prod-demo-vault.iam.gserviceaccount.c
om" \
    --role="roles/run.invoker" \
    --condition=None
```

2. Create scheduler job:
   The schedule `0 0 * * *` runs the job in UTC daily at midnight. You can modify this using standard cron syntax.

```
Unset
gcloud scheduler jobs create http backup-script-scheduler \
    --schedule="0 0 * * *" \
    --location=us-central1 \

--uri="https://us-central1-run.googleapis.com/apis/run.googleapis.com/v1/namesp
aces/prod-demo-vault/jobs/backup-script-job:run" \
    --http-method=POST \

--oauth-service-account-email=backup-script-sa@prod-demo-vault.iam.gserviceacco
unt.com
```

8. Testing

1. Execute the job manually:
   You may be asked to pick a region. This must be the same region as your backup plan.

```
Unset
gcloud run jobs execute backup-script-job
```

2. Check execution status with
   Option 1 - with Cloud Console:

    1. Navigate to Cloud Run - 'Jobs' tab.
    2. Here you will find your “backup-script-job”. Click into the details.
    3. View a history of all passed runs that have been executed and their status.
        1. View logs on every run to see if the script executed with a success status.

   Option 2 - with CLI:

```
Unset
```
# Get the latest execution ID
LATEST_EXECUTION=$(gcloud run jobs executions list --job backup-script-job
--limit=1 --format="value(name)")
```
```

```
# View logs
gcloud logging read "resource.type=cloud_run_job AND
resource.labels.job_name=backup-script-job AND
resource.labels.execution_name=${LATEST_EXECUTION}" --limit=100
--format="table(textPayload)"
```

9. Monitoring and Troubleshooting

View Job History

```
Unset
gcloud run jobs executions list --job backup-script-job
```

Check Scheduler Job Status

```
Unset
gcloud scheduler jobs list
```

Common Issues and Solutions

1. **Permission Denied Errors**
    - Verify all IAM roles are correctly assigned
    - Check both projects have the necessary permissions
    - Ensure service account exists and is properly configured
2. **Backup Plan Not Found**
    - Verify the backup plan exists using the list command
    - Check the full backup plan name format
    - Ensure you're in the correct project
3. **Project Access Issues**
    - Verify project IDs are correct
    - Check if all required APIs are enabled
    - Ensure service account has proper project access

10. Maintenance Tasks

Update Job Configuration

```
Unset
gcloud run jobs update backup-script-job \
    [include any parameters you want to change]
```

Update Schedule

```
Unset
gcloud scheduler jobs update http backup-script-scheduler \
    --schedule="NEW_SCHEDULE"
```

Important Notes

- All commands assume you're in the test-docker directory
- Replace project IDs if different from examples
- The scheduler uses UTC timezone
- Job timeout is set to 1 hour (3600s)
- Job will retry up to 3 times on failure
- All permissions are set without conditions for simplicity

Best Practices

1. Regularly monitor job execution logs
2. Keep track of successful/failed backups
3. Test the backup restoration process
4. Maintain documentation of any custom modifications
5. Regularly review and update permissions as needed