Automated GCP VM Backup Management

Ashika Ganesh | Last Updated: Nov 26, 2024

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: Make sure your XXX-compute@developer.gserviceaccount.com has Roles: GCS Object Lister, Logs Writer, and Storage Object User.

```
Unset
steps:
 # Print the working directory contents for debugging
 - name: 'ubuntu'
   args: ['ls', '-la']

 # Build the container
 - name: 'gcr.io/cloud-builders/docker'
   args:
     - 'build'
     - '-t'
     -
'us-central1-docker.pkg.dev/$PROJECT_ID/backup-scripts/backup-script:latest'
     - '.'
  # Push to Artifact Registry
 - name: 'gcr.io/cloud-builders/docker'
   args:
     - 'push'
     -
'us-central1-docker.pkg.dev/$PROJECT_ID/backup-scripts/backup-script:latest'

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

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 locally

```
Unset
# Build locally
docker build -t backup-script:latest .
```

3. Tag for Artifact Registry. Update the project name to be your current project where backups reside.

```
Unset
# Tag for Artifact Registry
```

```
docker tag backup-script:latest
us-central1-docker.pkg.dev/YOUR-PROJECT-NAME/backup-scripts/backup-script:lates
t
```

4. Push to Artifact Registry. Update the project name to be your current project where backups reside.

```
Unset
# Push to Artifact Registry
docker push
us-central1-docker.pkg.dev/YOUR-PROJECT-NAME/backup-scripts/backup-script:lates
t
```

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 the backup project where your backup plan and backup vaults reside (YOUR-PROJECT-NAME):

```
Unset
# Base permissions
gcloud projects add-iam-policy-binding YOUR-PROJECT-NAME \

--member="serviceAccount:backup-script-sa@YOUR-PROJECT-NAME.iam.gserviceacco
unt.com" \
    --role="roles/viewer" \
    --condition=None

gcloud projects add-iam-policy-binding YOUR-PROJECT-NAME \
--member="serviceAccount:backup-script-sa@YOUR-PROJECT-NAME.iam.gserviceacco
unt.com" \
    --role="roles/backupdr.admin" \
    --condition=None

gcloud projects add-iam-policy-binding YOUR-PROJECT-NAME \

--member="serviceAccount:backup-script-sa@YOUR-PROJECT-NAME.iam.gserviceacco
unt.com" \
    --role="roles/iam.serviceAccountUser" \
    --condition=None

# Additional required permissions (added based on troubleshooting)
gcloud projects add-iam-policy-binding YOUR-PROJECT-NAME \

--member="serviceAccount:backup-script-sa@YOUR-PROJECT-NAME.iam.gserviceacco
unt.com" \
    --role="roles/backupdr.computeEngineBackupAdmin" \
    --condition=None

gcloud projects add-iam-policy-binding YOUR-PROJECT-NAME \

--member="serviceAccount:backup-script-sa@YOUR-PROJECT-NAME.iam.gserviceacco
unt.com" \
    --role="roles/compute.instanceAdmin.v1" \
    --condition=None
```

3. Grant permissions on target project (TARGET-PROJECT-NAME):

```
Unset
# Base permissions
gcloud projects add-iam-policy-binding TARGET-PROJECT-NAME \

--member="serviceAccount:backup-script-sa@TARGET-PROJECT-NAME.iam.gservicea
ccount.com" \
    --role="roles/viewer" \
    --condition=None

gcloud projects add-iam-policy-binding TARGET-PROJECT-NAME \
--member="serviceAccount:backup-script-sa@TARGET-PROJECT-NAME.iam.gserviceaccount.com" \
    --role="roles/resourcemanager.tagViewer" \
    --condition=None

gcloud projects add-iam-policy-binding TARGET-PROJECT-NAME \

--member="serviceAccount:backup-script-sa@TARGET-PROJECT-NAME.iam.gserviceaccount.com" \
    --role="roles/compute.viewer" \
    --condition=None

# Additional required permissions (added based on troubleshooting)
gcloud projects add-iam-policy-binding TARGET-PROJECT-NAME \

--member="serviceAccount:backup-script-sa@TARGET-PROJECT-NAME.iam.gserviceaccount.com" \
    --role="roles/backupdr.computeEngineBackupAdmin" \
    --condition=None

gcloud projects add-iam-policy-binding TARGET-PROJECT-NAME \

--member="serviceAccount:backup-script-sa@TARGET-PROJECT-NAME.iam.gserviceaccount.com" \
    --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=YOUR-PROJECT-NAME \
    --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. Be sure to update YOUR-PROJECT-NAME with your project.
    - If you want to make changes after creation, you can modify through editing the YAML in the Cloud run Jobs YAML tab.

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

--image=us-central1-docker.pkg.dev/YOUR-PROJECT-NAME/backup-scripts/backup-s
cript:latest \

--service-account=backup-script-sa@YOUR-PROJECT-NAME.iam.gserviceaccount.com
\
    --region=us-central1 \
    --args="--backup-project-id=YOUR-PROJECT-NAME" \
    --args="--location=us-central1" \
    --args="--backup-plan=bp-automation" \
    --args="--tag-key=environment" \
    --args="--tag-value=production" \
    --args="--projects=TARGET-PROJECT-NAME" \
    --max-retries=3 \
    --task-timeout=3600s
```

7. Set Up Cloud Scheduler

1. Grant additional permissions for scheduler:

```
Unset
gcloud projects add-iam-policy-binding YOUR-PROJECT-NAME \

--member="serviceAccount:backup-script-sa@YOUR-PROJECT-NAME.iam.gserviceacco
unt.com" \
    --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/YOUR-PROJECT-NAME/jobs/backup-script-job:run" \
    --http-method=POST \

--oauth-service-account-email=backup-script-sa@YOUR-PROJECT-NAME.iam.gservic
eaccount.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