MMM Trainer Deployment Guide

Self-Hosted Marketing Mix Modeling Platform

Deployment Documentation for Customers

January 9, 2026

Contents

1 Executive Summary

The MMM Trainer is a cloud-native Marketing Mix Modeling platform built on Google Cloud Platform (GCP). It provides a web-based interface for data scientists and analysts to run Facebook's Robyn MMM framework at scale, processing data from Snowflake and delivering production-ready model insights.

1.1 Key Features

- **Web-Based Interface:** Streamlit application accessible via browser
- **Snowflake Integration:** Direct connection to your data warehouse
- **Robyn MMM Engine:** Industry-standard R/Robyn framework
- **Cloud-Native Architecture:** Scalable, secure, and cost-effective
- **Batch Processing:** Queue system for running multiple experiments
- **Results Management:** Automated storage and visualization of model outputs

1.2 Infrastructure Overview

The platform consists of:

- **Web Service:** Streamlit application on Cloud Run (always-on)
- **Training Jobs:** R/Robyn execution on Cloud Run Jobs (on-demand)
- **Storage:** Google Cloud Storage for data and model artifacts
- **Scheduler:** Cloud Scheduler for batch job processing
- **Secrets:** Google Secret Manager for credential management

2 Cost Estimates

All costs are based on actual production measurements using Google Cloud Platform services in the `europe-west1` region. Costs are billed per-second for compute resources.

2.1 Training Job Performance

Workload	Iterations × Trials	Duration	Cost/Job	Use Case
Test Run	200 × 3 = 600	0.8 min	$0.01	Quick validation
Benchmark	**2,000 × 5 = 10K**	**12 min**	**$0.20**	**Standard testing**
Production	**10,000 × 5 = 50K**	**80-120 min**	**$1.33-$2.00**	**Production runs**
Large Production	10,000 × 10 = 100K	160-240 min	$2.67-$4.00	High-quality runs

Table 1: Training job durations and costs (8 vCPU, 32GB RAM configuration)

Note: Benchmark runs (12-18 minutes) are ideal for experimentation and testing. Production runs (80-120 minutes) provide more thorough model exploration and are recommended for final models.

2.2 Monthly Cost Scenarios

Usage Level	Web Calls	Training Jobs	Benchmark Cost	Production Cost
Light	100	10	$4	$15-22
Moderate	500	50	$12	$69-102
Heavy	1,000	100	$22	$135-202
Very Heavy	5,000	500	$102	$667-1002

Table 2: Monthly cost estimates by usage volume

2.3 Cost Breakdown

Fixed Monthly Costs (˜$2/month):

- GCS storage: $0.50-$2.00 (depends on data retention)
- Secret Manager: $0.36 (6 secrets × $0.06)
- Cloud Scheduler: $0.30 (covered by free tier)
- Artifact Registry: $0.50 (container images)

Variable Costs (per usage):

- Training jobs: $0.20 (benchmark) to $1.33-$2.00 (production) per job
- Web service: ˜$0.002 per request (negligible)
- Snowflake: Separate (depends on warehouse size and queries)

Key Cost Drivers:

- Training jobs account for 90-99% of total costs at scale

- Faster machines have similar costs due to per-second billing
- Storage costs are minimal with lifecycle policies
- Network egress charged only for downloads outside GCP region

3 Minimum Technical Requirements

3.1 Required Skills and Knowledge

The technical team maintaining this application should have:

3.1.1 Essential Skills

- **Google Cloud Platform:**
  - Basic understanding of Cloud Run, GCS, and IAM
  - Ability to navigate GCP Console
  - Understanding of billing and cost management
- **Infrastructure as Code:**
  - Basic Terraform knowledge for infrastructure changes
  - Ability to read and modify Terraform configuration files
- **Version Control:**
  - Git and GitHub workflows
  - Understanding of CI/CD concepts
- **Container Technology:**
  - Basic Docker concepts
  - Understanding of container registries

3.1.2 Recommended Skills

- **Programming Languages:**
  - Python (for Streamlit application modifications)
  - R (for Robyn MMM customizations)
- **Data Warehouse:**
  - Snowflake query optimization
  - SQL for data extraction
- **Monitoring and Debugging:**
  - Cloud Logging for troubleshooting
  - Performance monitoring and optimization

3.2 Required Tools

All team members should have access to:

Tool	Version	Purpose
Google Cloud SDK	Latest	GCP CLI operations
Terraform	$\geq$ 1.5.0	Infrastructure management
Docker	Latest	Container testing (optional)
Git	Latest	Version control
Python	$\geq$ 3.11	Local development (optional)

Table 3: Required development tools

3.3 Access Requirements

3.3.1 Google Cloud Platform

- **For Monitoring:** Viewer role
- **For Deployments:** Editor or specific roles:
  - Cloud Run Admin
  - Storage Admin
  - Secret Manager Admin
  - Service Account Admin
- **For Debugging:** Logs Viewer, Monitoring Viewer

3.3.2 GitHub Repository

- **For Development:** Write access
- **For Releases:** Maintain or Admin access
- **For Secrets Management:** Admin access

3.3.3 Snowflake

- Read access to source data tables
- Access to a dedicated warehouse for queries
- Appropriate role (not ACCOUNTADMIN in production)

4 Infrastructure Requirements

4.1 Google Cloud Platform Resources

4.1.1 Required GCP APIs

The following APIs must be enabled in your GCP project:

- `run.googleapis.com` - Cloud Run
- `artifactregistry.googleapis.com` - Container registry
- `cloudbuild.googleapis.com` - Container builds
- `cloudscheduler.googleapis.com` - Job scheduling
- `secretmanager.googleapis.com` - Secrets management
- `storage.googleapis.com` - Cloud Storage
- `iamcredentials.googleapis.com` - Service accounts
- `iam.googleapis.com` - IAM management
- `cloudresourcemanager.googleapis.com` - Project management

4.1.2 Required GCP Resources

Resource	Purpose	Example Name
GCP Project	Container for all resources	`mmm-production`
GCS Bucket (State)	Terraform state storage	`project-tf-state`
GCS Bucket (App)	Model outputs & data	`mmm-app-output`
Artifact Registry	Docker image storage	`mmm-repo`
Workload Identity Pool	GitHub authentication	`github-pool`
Service Account (Deploy)	CI/CD deployment	`github-deployer`
Service Account (Web)	Web service runtime	`mmm-web-sa`
Service Account (Training)	Training job runtime	`mmm-training-sa`

Table 4: Required GCP resources

4.2 Compute Resources

4.2.1 Web Service Configuration

- **Platform:** Cloud Run Service
- **CPU:** 2 vCPU
- **Memory:** 4GB RAM
- **Min Instances:** 0 (cost-optimized) or 1+ (low latency)
- **Max Instances:** 10 (adjustable)
- **Timeout:** 60 seconds per request
- **Concurrency:** 80 requests per instance

4.2.2 Training Job Configuration

- **Platform:** Cloud Run Jobs
- **CPU:** 8 vCPU (recommended for 12-minute runs)
- **Memory:** 32GB RAM
- **Timeout:** 6 hours (adjustable)
- **Max Retries:** 1
- **Parallelism:** 1 (single job at a time)

Note: CPU/memory can be adjusted in Terraform configuration. More cores = faster training but similar cost due to per-second billing.

4.3 Storage Requirements

4.3.1 Google Cloud Storage

- **Estimated Usage:**
  - Base: 80GB for historical data
  - Growth: ˜2GB per training run
- **Lifecycle Policies:**
  - Move to Nearline after 30 days (50% cost reduction)
  - Move to Coldline after 90 days (80% cost reduction)
- **Backup:** Enabled via versioning

4.4 Network Requirements

- **Outbound:** HTTPS access to:
  - Snowflake data warehouse
  - GitHub (for CI/CD)
  - Docker Hub / Package registries
- **Inbound:** HTTPS only (TLS 1.2+)
- **Domain:** Optional custom domain via Cloud Run
- **Firewall:** Managed by GCP (no configuration needed)

4.5 Security Requirements

4.5.1 Authentication

- **User Authentication:** Google OAuth 2.0
  - Domain-restricted access (e.g., @company.com)
  - OAuth consent screen configuration
  - Client ID and Client Secret
- **Service Authentication:** GCP service accounts
- **Snowflake Authentication:** RSA private key (preferred) or password

4.5.2 Secrets Management

- All secrets stored in Google Secret Manager
- Automatic secret rotation supported
- No secrets in code or environment variables
- Audit logging for secret access

4.5.3 IAM Permissions

Service accounts follow principle of least privilege:

- **Web Service:** Artifact Registry Reader, Storage Object Admin, Secret Accessor
- **Training Job:** Storage Object Admin, Secret Accessor
- **Deployer:** Cloud Run Admin, Storage Admin, Service Account User

5 Deployment Overview

5.1 Deployment Architecture

The application uses GitHub Actions for continuous deployment with Terraform managing infrastructure as code.

5.1.1 Deployment Flow

1. Developer pushes code to GitHub repository
2. GitHub Actions triggered automatically
3. Docker images built (web service + training job)
4. Images pushed to Artifact Registry
5. Terraform applies infrastructure changes
6. Cloud Run service and jobs updated
7. Health checks verify deployment

5.1.2 Environments

- **Production:** Deployed from `main` branch
- **Development:** Deployed from `dev` or `feat-*` branches
- Separate Terraform workspaces for isolation

5.2 Initial Setup Steps

5.2.1 Phase 1: GCP Project Setup (1-2 hours)

1. Create or select GCP project
2. Enable required APIs
3. Create Terraform state bucket
4. Set up billing alerts

5.2.2 Phase 2: Authentication Setup (1-2 hours)

1. Configure Workload Identity Federation for GitHub
2. Create service accounts with appropriate roles
3. Set up Google OAuth consent screen and credentials
4. Generate or obtain Snowflake RSA key pair

5.2.3 Phase 3: Infrastructure Deployment (30 minutes)

1. Create Artifact Registry repository
2. Create GCS bucket for outputs
3. Configure Terraform backend and variables
4. Update GitHub repository secrets

5.2.4 Phase 4: Application Deployment (15 minutes)

1. Push code to trigger CI/CD
2. Monitor GitHub Actions workflow
3. Verify Cloud Run service deployment
4. Test web interface access

5.2.5 Phase 5: Verification (30 minutes)

1. Test Snowflake connection
2. Run benchmark training job
3. Verify results in GCS
4. Check logs and monitoring

Total Estimated Time: 3-5 hours for initial deployment

5.3 Configuration Files

Key configuration files to customize:

- `infra/terraform/backend.tf` - Terraform state bucket
- `infra/terraform/envs/prod.tfvars` - Production settings
- `infra/terraform/envs/dev.tfvars` - Development settings
- `.github/workflows/ci.yml` - Production CI/CD
- `.github/workflows/ci-dev.yml` - Development CI/CD

5.4 Required Credentials

Collect these credentials before deployment:

Credential	Description
GCP Project ID	Your Google Cloud project identifier
GCP Project Number	Numeric project identifier (auto-generated)
OAuth Client ID	From GCP OAuth consent screen
OAuth Client Secret	From GCP OAuth consent screen
Cookie Secret	Random 32-byte hex (generate with `openssl rand -hex 32`)
SF Account	Snowflake account identifier
SF User	Snowflake username
SF Private Key	RSA private key in PEM format (preferred)
SF Warehouse	Snowflake warehouse name
SF Database	Snowflake database name
SF Schema	Snowflake schema name

Table 5: Required credentials for deployment

6 Ongoing Maintenance

6.1 Regular Maintenance Tasks

6.1.1 Daily/Weekly

- Monitor Cloud Run service health
- Check training job success rates
- Review logs for errors or warnings
- Monitor GCS storage growth

6.1.2 Monthly

- Review GCP billing and optimize costs
- Analyze Snowflake warehouse usage
- Clean up old training artifacts (automated via lifecycle)
- Review and update access permissions

6.1.3 Quarterly

- Update Python dependencies (security patches)
- Update R packages for Robyn
- Review and optimize Cloud Run resources
- Test disaster recovery procedures
- Rotate sensitive credentials (OAuth, Snowflake keys)

6.2 Monitoring and Alerting

6.2.1 Key Metrics to Monitor

- **Web Service:**
  - Request latency (target: ¡2s)
  - Error rate (target: ¡1%)
  - CPU/Memory utilization
- **Training Jobs:**
  - Job success rate (target: ¿95%)
  - Average execution time
  - Resource utilization
- **Storage:**
  - Storage growth rate
  - Bucket size vs lifecycle policies
- **Costs:**

  - Monthly spend by service
  - Cost per training job
  - Storage costs

6.2.2 Recommended Alerts

- Budget alert at 50% and 90% of monthly budget
- Error rate exceeds 5% for 5 minutes
- Training job failures exceed 10%
- Storage exceeds 500GB (adjust based on usage)

6.3 Backup and Disaster Recovery

6.3.1 What is Backed Up

- **Code:** Version controlled in GitHub
- **Infrastructure:** Defined in Terraform (version controlled)
- **Terraform State:** Stored in GCS with versioning enabled
- **Model Artifacts:** Stored in GCS with retention policies
- **Secrets:** Managed in Google Secret Manager with versions

6.3.2 Recovery Time Objectives

- **Web Service:** ¡15 minutes (redeploy from GitHub)
- **Infrastructure:** ¡30 minutes (Terraform apply)
- **Data Loss:** Minimal (GCS has 11 nines durability)

6.3.3 Disaster Recovery Procedure

1. Create new GCP project (if primary is unavailable)
2. Run initial setup (APIs, buckets, service accounts)
3. Deploy infrastructure via Terraform
4. Trigger CI/CD deployment from GitHub
5. Restore secrets from backup documentation
6. Verify functionality and notify users

6.4 Security Maintenance

6.4.1 Credential Rotation Schedule

- **Snowflake Private Key:** Annually or upon personnel changes
- **OAuth Secrets:** Annually
- **Service Account Keys:** Not needed (uses Workload Identity)
- **Cookie Secret:** Annually or after security incidents

6.4.2 Access Review

- Review GCP IAM permissions quarterly
- Remove access for departed team members immediately
- Audit service account permissions annually
- Review Google OAuth allowed domains

6.4.3 Security Updates

- Apply Python security updates within 30 days
- Update base Docker images quarterly
- Monitor GitHub security advisories
- Subscribe to GCP security bulletins

6.5 Troubleshooting Common Issues

6.5.1 Web Service Not Responding

1. Check Cloud Run service status in GCP Console
2. Review logs in Cloud Logging
3. Verify service account permissions
4. Check OAuth configuration
5. Restart service if needed (redeploy from GitHub)

6.5.2 Training Jobs Failing

1. Check job execution logs in Cloud Logging
2. Verify Snowflake connectivity and credentials
3. Check GCS bucket permissions
4. Review memory/CPU allocation (increase if OOM errors)
5. Examine `robyn_console.log` in GCS results folder

6.5.3 High Costs

1. Review Cloud Run billing in GCP Console
2. Check number of training jobs executed
3. Verify min_instances setting (should be 0 for cost optimization)
4. Review GCS storage lifecycle policies
5. Analyze Snowflake warehouse usage

6.6 Update Procedures

6.6.1 Updating Application Code

1. Create feature branch from `dev`
2. Make and test changes locally
3. Push to GitHub (triggers dev deployment)
4. Test in dev environment
5. Create pull request to `main`
6. Review and merge (triggers production deployment)

6.6.2 Updating Infrastructure

1. Modify Terraform files in `infra/terraform/`
2. Test changes in dev environment first
3. Review Terraform plan output carefully
4. Apply changes via GitHub Actions
5. Verify resources in GCP Console

6.6.3 Updating Dependencies

1. Update `requirements.txt` (Python) or `Dockerfile.training-base` (R)
2. Test locally or in dev environment
3. Check for breaking changes in package release notes
4. Deploy to dev, verify functionality
5. Deploy to production after successful testing

7 Support and Resources

7.1 Documentation

- **README.md** - Project overview and quick start
- **ARCHITECTURE.md** - System architecture and components
- **DEVELOPMENT.md** - Local development setup
- **REQUIREMENTS.md** - Detailed technical requirements
- **DEPLOYMENT_GUIDE.md** - Step-by-step deployment instructions
- **COST_OPTIMIZATION.md** - Cost management strategies

7.2 External Resources

- **Google Cloud Documentation:** `https://cloud.google.com/docs`
- **Cloud Run Pricing:** `https://cloud.google.com/run/pricing`
- **Terraform GCP Provider:** `https://registry.terraform.io/providers/hashicorp/google/latest`
- **Robyn MMM:** `https://github.com/facebookexperimental/Robyn`
- **Streamlit Documentation:** `https://docs.streamlit.io`

7.3 Getting Help

For technical support:

1. Review relevant documentation in the `docs/` directory
2. Check Cloud Logging for error details
3. Review GitHub repository issues
4. Contact your implementation partner or technical lead

8 Appendix A: Quick Reference

8.1 Cost Calculator

Use this formula to estimate monthly costs:

$$\text{Monthly Cost} = \text{Fixed} + (\text{Jobs} \times \text{Cost per Job}) \tag{1}$$

Where:

- Fixed = $2-$3/month
- Jobs = Number of training jobs per month
- Cost per Job = $0.20 (benchmark) or $1.33-$2.00 (production)

Example: 100 production jobs/month = $2 + (100 × $1.67 avg) = $169/month

8.2 GCP Regions

Recommended regions for deployment:

- `europe-west1` (Belgium) - EU, low latency
- `europe-west4` (Netherlands) - EU, low latency
- `us-central1` (Iowa) - US, low cost
- `us-east1` (South Carolina) - US, low cost

Choose region closest to your Snowflake instance for optimal performance.

8.3 Service Limits

Default GCP quotas (can be increased via support request):

- Cloud Run services per project: 100
- Cloud Run jobs per project: 1000
- Concurrent Cloud Run job executions: 100
- GCS buckets per project: 10,000
- Cloud Run request timeout: 60 minutes

8.4 Glossary

Cloud Run GCP's serverless container platform for web services and batch jobs

GCS Google Cloud Storage - object storage service

IAM Identity and Access Management - GCP's permission system

Robyn Facebook's open-source Marketing Mix Modeling framework

Terraform Infrastructure as Code tool for managing cloud resources

vCPU Virtual CPU - compute allocation unit in cloud environments

Workload Identity GCP's recommended way to authenticate from external services

9 Appendix B: Deployment Checklist

Use this checklist for initial deployment:

9.1 Pre-Deployment

- Install required tools (gcloud, terraform, docker, git)
- Create GCP project with billing enabled
- Enable required GCP APIs
- Create Terraform state bucket
- Create Artifact Registry repository
- Set up Workload Identity Federation for GitHub
- Create deployer service account with permissions
- Create output GCS bucket
- Configure Google OAuth (consent screen + credentials)
- Generate Snowflake RSA key pair (or obtain password)
- Configure GitHub repository secrets

9.2 Configuration

- Update `infra/terraform/backend.tf` with state bucket
- Update `infra/terraform/envs/prod.tfvars` with project details
- Update `.github/workflows/ci.yml` with project ID and region
- Verify all environment variables in workflows
- Review and adjust resource allocations (CPU/memory)

9.3 Deployment

- Push code to GitHub (main branch)
- Monitor GitHub Actions workflow
- Verify Docker images in Artifact Registry
- Verify Cloud Run service deployment
- Verify Cloud Run job registration
- Verify Cloud Scheduler jobs created

9.4 Verification

- Access web application URL
- Test Google OAuth login
- Test Snowflake connection
- Test data query and preview
- Run test training job (200×3 iterations)
- Verify results in GCS bucket
- Check Cloud Logging for errors
- Verify IAM permissions are correct
- Set up billing alerts
- Document deployment details

9.5 Post-Deployment

- Run benchmark training job (2000×5)
- Run production training job (10000×5)
- Verify all features work as expected
- Train team on platform usage
- Set up monitoring dashboards
- Configure alerting rules
- Document any customizations made
- Schedule first maintenance review