Google Cloud App Engine Deployment Guide

Table of Contents

Prerequisites

System Requirements

- **Google Cloud SDK**: Version 400.0.0 or later
- **Python**: 3.7+ (for Python applications)
- **Node.js**: 14+ (for Node.js applications)
- **Docker**: Latest version (for containerized deployments)
- **Git**: For version control

Account Setup

1. **Google Cloud Account**: Active billing account required
2. **Project Permissions**: Editor or App Engine Admin role
3. **APIs Enabled**:
   - App Engine Admin API
   - Cloud Build API
   - Cloud Storage API

Local Development Environment

```bash
```

```
# Install Google Cloud SDK
curl https://sdk.cloud.google.com | bash
exec -l $SHELL

# Initialize gcloud
gcloud init
gcloud auth login
gcloud config set project YOUR_PROJECT_ID
```

Project Setup

1. Create New Google Cloud Project

bash

```
# Create project
gcloud projects create YOUR_PROJECT_ID --name="Your App Name"

# Set as default project
gcloud config set project YOUR_PROJECT_ID

# Enable billing (replace BILLING_ACCOUNT_ID)
gcloud beta billing projects link YOUR_PROJECT_ID \
  --billing-account=BILLING_ACCOUNT_ID
```

2. Enable Required APIs

bash

```
gcloud services enable appengine.googleapis.com
gcloud services enable cloudbuild.googleapis.com
gcloud services enable storage.googleapis.com
```

3. Initialize App Engine Application

bash

```
# Initialize App Engine (select region when prompted)
gcloud app create --region=us-central1
```

4. Directory Structure

```
your-app/
├── app.yaml            # App Engine configuration
├── requirements.txt    # Python dependencies
├── main.py             # Application entry point
├── static/             # Static files
│   ├── css/
│   ├── js/
│   └── images/
├── templates/          # Template files
├── .gcloudignore       # Files to ignore during deployment
└── cloudbuild.yaml     # Build configuration (optional)
```

Application Configuration

app.yaml Configuration

Basic Python Application

yaml

```yaml
runtime: python39

env_variables:
  DATABASE_URL: "postgresql://user:pass@host:port/db"
  SECRET_KEY: "your-secret-key"
  DEBUG: "False"

automatic_scaling:
  target_cpu_utilization: 0.65
  target_throughput_utilization: 0.65
  min_instances: 1
  max_instances: 10

resources:
  cpu: 1
  memory_gb: 0.5
  disk_size_gb: 10

handlers:
- url: /static
  static_dir: static

- url: /.*
  script: auto
```

Node.js Application

```yaml
runtime: nodejs16

env_variables:
  NODE_ENV: production
  DATABASE_URL: "mongodb://user:pass@host:port/db"

automatic_scaling:
  target_cpu_utilization: 0.65
  min_instances: 1
  max_instances: 5

resources:
  cpu: 1
  memory_gb: 1

handlers:
- url: /static
  static_dir: public

- url: /.*
  secure: always
  script: auto
```

Advanced Configuration Options

```yaml
```

```yaml
# Custom runtime with Docker
runtime: custom
env: flex

# Manual scaling
manual_scaling:
 instances: 2

# VPC settings
vpc_access_connector:
 name: projects/PROJECT_ID/locations/REGION/connectors/CONNECTOR_NAME

# Health checks
liveness_check:
 path: "/health"
 check_interval_sec: 30
 timeout_sec: 4
 failure_threshold: 2
 success_threshold: 2

readiness_check:
 path: "/readiness"
 check_interval_sec: 5
 timeout_sec: 4
 failure_threshold: 2
 success_threshold: 2
```

Environment Variables and Secrets

Using Secret Manager

yaml

```yaml
# In app.yaml
env_variables:
 DATABASE_PASSWORD: ${DATABASE_PASSWORD}

# Deploy with secret
gcloud app deploy --set-env-vars=DATABASE_PASSWORD="$(gcloud secrets versions access latest --secret=db-pass
```

Multiple Environments

yaml

```yaml
# app-staging.yaml
runtime: python39
service: staging
env_variables:
  ENV: "staging"
  DEBUG: "True"

# app-production.yaml
runtime: python39
service: default
env_variables:
  ENV: "production"
  DEBUG: "False"
```

Deployment Process

1. Pre-deployment Checklist

- [ ] Code tested locally
- [ ] Dependencies updated in requirements.txt/package.json
- [ ] Environment variables configured
- [ ] Database migrations ready
- [ ] Static files optimized
- [ ] Security configurations verified

2. Basic Deployment Commands

Deploy to Default Service

```bash
# Deploy current directory
gcloud app deploy

# Deploy specific configuration
gcloud app deploy app.yaml

# Deploy with custom version
gcloud app deploy --version=v1-2-3

# Deploy without promoting to live traffic
gcloud app deploy --no-promote
```

Deploy Multiple Services

```bash
# Deploy all services
gcloud app deploy app.yaml worker.yaml scheduler.yaml

# Deploy specific service
gcloud app deploy worker.yaml --service=background-worker
```

3. Advanced Deployment Options

Blue-Green Deployment

```bash
# Deploy new version without promoting
gcloud app deploy --version=blue --no-promote

# Test the new version
curl https://blue-dot-YOUR_PROJECT_ID.appspot.com

# Split traffic between versions
gcloud app services set-traffic default --splits=green=50,blue=50

# Promote new version to 100% traffic
gcloud app services set-traffic default --splits=blue=100
```

Automated Deployment with Cloud Build

```yaml
```

```
# cloudbuild.yaml
steps:
- name: 'gcr.io/cloud-builders/gcloud'
 args:
 - 'app'
 - 'deploy'
 - '--version=${SHORT_SHA}'
 - '--no-promote'

- name: 'gcr.io/cloud-builders/gcloud'
 args:
 - 'app'
 - 'services'
 - 'set-traffic'
 - 'default'
 - '--splits=${SHORT_SHA}=100'

timeout: '1600s'
```

4. Deployment Verification

```bash
# Check deployment status
gcloud app versions list

# View application logs
gcloud app logs tail -s default

# Browse to application
gcloud app browse

# Check service status
gcloud app services list
```

Environment Management

Version Management

```bash
```

```
# List all versions
gcloud app versions list

# Delete old versions
gcloud app versions delete v1 v2 v3

# Migrate traffic gradually
gcloud app services set-traffic default \
  --splits=v1=10,v2=90 \
  --split-by=cookie
```

Traffic Splitting Strategies

```bash
# IP-based splitting
gcloud app services set-traffic default \
  --splits=v1=25,v2=75 \
  --split-by=ip

# Random splitting
gcloud app services set-traffic default \
  --splits=v1=50,v2=50 \
  --split-by=random

# Cookie-based splitting
gcloud app services set-traffic default \
  --splits=v1=30,v2=70 \
  --split-by=cookie
```

Environment-specific Configurations

```bash
# Deploy to staging
gcloud app deploy app-staging.yaml --project=staging-project

# Deploy to production
gcloud app deploy app-production.yaml --project=production-project
```

Monitoring and Logging

Application Logging

python

```python
# Python logging example
import logging
import google.cloud.logging

# Setup Cloud Logging
client = google.cloud.logging.Client()
client.setup_logging()

# Use standard logging
logging.info("Application started")
logging.error("An error occurred", extra={"user_id": 123})
```

Monitoring Setup

bash

```bash
# View logs
gcloud app logs read --service=default --version=latest

# Real-time log streaming
gcloud app logs tail --service=default

# Filter logs by severity
gcloud app logs read --filter="severity >= ERROR"
```

Performance Monitoring

yaml

```yaml
# In app.yaml - Enable detailed monitoring
env_variables:
  GOOGLE_CLOUD_PROFILER_ENABLE: "true"
  GOOGLE_CLOUD_PROFILER_VERSION: "1.0.0"
```

Custom Metrics and Alerts

python

```
# Custom metrics example
from google.cloud import monitoring_v3
import time

def record_custom_metric(value):
    client = monitoring_v3.MetricServiceClient()
    project_name = f"projects/{PROJECT_ID}"

    series = monitoring_v3.TimeSeries()
    series.metric.type = "custom.googleapis.com/my_metric"
    series.resource.type = "gae_app"

    point = series.points.add()
    point.value.double_value = value
    point.interval.end_time.seconds = int(time.time())

    client.create_time_series(name=project_name, time_series=[series])
```

Troubleshooting

Common Issues and Solutions

Deployment Failures

bash

```
# Issue: Build timeout
# Solution: Increase timeout in cloudbuild.yaml
timeout: '2400s'

# Issue: Memory exceeded during build
# Solution: Use Cloud Build with higher machine type
options:
  machineType: 'E2_HIGHCPU_8'
```

Runtime Errors

bash

```
# Check application logs
gcloud app logs read --limit=50

# Debug instance issues
gcloud app instances list
gcloud app instances ssh INSTANCE_ID --service=SERVICE --version=VERSION
```

Performance Issues

```bash
# Monitor instance metrics
gcloud app operations list

# Check resource utilization
gcloud app versions describe VERSION --service=SERVICE
```

Debug Commands

```bash
# Enable debug mode (local development)
export GOOGLE_APPLICATION_CREDENTIALS="path/to/service-key.json"
export FLASK_ENV=development  # For Flask apps

# Local testing with App Engine environment
dev_appserver.py app.yaml

# Test specific service locally
dev_appserver.py app.yaml worker.yaml
```

Best Practices

Security

1. **Environment Variables**: Never commit secrets to version control
2. **IAM Roles**: Use principle of least privilege
3. **HTTPS**: Always enforce secure connections
4. **Secret Manager**: Store sensitive data in Google Secret Manager
5. **VPC**: Use VPC connectors for database connections

Performance Optimization

1. **Caching**: Implement appropriate caching strategies
2. **Static Files**: Use CDN for static content
3. **Database**: Optimize queries and use connection pooling
4. **Monitoring**: Set up alerts for key metrics
5. **Scaling**: Configure appropriate scaling parameters

Development Workflow

```bash
# Recommended workflow
git checkout -b feature/new-feature
# Make changes
git add .
git commit -m "Add new feature"
git push origin feature/new-feature

# Deploy to staging for testing
gcloud app deploy app-staging.yaml --project=staging-project

# After approval, merge and deploy to production
git checkout main
git merge feature/new-feature
gcloud app deploy --project=production-project
```

Cost Optimization

1. **Instance Classes**: Choose appropriate instance sizes
2. **Automatic Scaling**: Configure min/max instances carefully
3. **Version Cleanup**: Regularly delete old versions
4. **Traffic Splitting**: Use for gradual rollouts
5. **Monitoring**: Set up billing alerts

Maintenance

```bash
```

```
# Regular maintenance tasks
# 1. Update dependencies
pip freeze > requirements.txt  # Python
npm audit fix              # Node.js

# 2. Clean up old versions
gcloud app versions list
gcloud app versions delete OLD_VERSION_1 OLD_VERSION_2

# 3. Review logs and metrics
gcloud app logs read --filter="severity >= WARNING" --limit=100

# 4. Update security configurations
gcloud app ssl-certificates list
```

Appendix

Useful Commands Reference

bash

```
# Project management
gcloud projects list
gcloud config set project PROJECT_ID

# App Engine operations
gcloud app browse
gcloud app describe
gcloud app logs tail

# Version management
gcloud app versions list
gcloud app versions migrate VERSION
gcloud app versions stop VERSION

# Service management
gcloud app services list
gcloud app services delete SERVICE
gcloud app services set-traffic SERVICE --splits=VERSION=100

# Instance management
gcloud app instances list
gcloud app instances delete INSTANCE --service=SERVICE --version=VERSION
```

Sample Applications

```python
# main.py - Simple Flask application
from flask import Flask, render_template
import logging

app = Flask(__name__)

@app.route('/')
def hello():
    logging.info('Hello endpoint accessed')
    return 'Hello, Google App Engine!'

@app.route('/health')
def health():
    return {'status': 'healthy'}, 200

if __name__ == '__main__':
    app.run(host='127.0.0.1', port=8080, debug=True)
```

Additional Resources

- Google Cloud Documentation
- App Engine Pricing
- Best Practices Guide
- Migration Guide

Document Version: 1.0
Last Updated: August 2025
Author: Cloud Engineering Team