🚀 SentiFlow Deployment Guide

This guide will walk you through deploying SentiFlow for the AI Accelerate Hackathon.

📋 Prerequisites

Required Accounts & Services

1. **Google Cloud Platform**
    - Active GCP project
    - Vertex AI API enabled
    - Cloud Run API enabled
    - Billing enabled
2. **Elastic Cloud**
    - Elastic Cloud account
    - Deployment created (version 8.11+)
    - API key generated

Required Tools

- Python 3.11+
- Docker (for containerized deployment)
- gcloud CLI (for Cloud Run deployment)
- Git

⚙️ Setup Instructions

1. Clone & Navigate

```
cd sentiflow
```

2. Configure Environment Variables

Copy the example environment file:

```
cp .env.example .env
```

Edit `.env` with your credentials:

```

```

```
# Google Cloud Configuration
GCP_PROJECT_ID=your-project-id
VERTEX_AI_LOCATION=us-central1

# Elastic Cloud Configuration
ELASTIC_CLOUD_ID=your-cloud-id
ELASTIC_API_KEY=your-api-key

# Model Configuration
GEMINI_MODEL=gemini-2.0-flash-exp
EMBEDDING_MODEL=text-embedding-004

# Elasticsearch Configuration
ELASTIC_INDEX_NAME=sentiflow_docs
```

3. Set Up Google Cloud

Enable required APIs:

```
gcloud auth login
gcloud config set project YOUR_PROJECT_ID

# Enable APIs
gcloud services enable aiplatform.googleapis.com
gcloud services enable run.googleapis.com
gcloud services enable containerregistry.googleapis.com
```

4. Set Up Elastic Cloud

1. Go to https://cloud.elastic.co/
2. Create a deployment (if not exists)
3. Copy the Cloud ID
4. Generate API key:
    - Go to Management → Security → API keys
    - Create new API key
    - Copy the key

🏃 Local Development

Option 1: Using Setup Script (Recommended)

```
chmod +x setup.sh
./setup.sh
```

This script will:

- Create virtual environment
- Install dependencies
- Set up Elasticsearch index
- Ingest sample documents
- Run tests

Option 2: Manual Setup

```
# Create virtual environment
python3 -m venv venv
source venv/bin/activate  # On Windows: venv\Scripts\activate

# Install dependencies
pip install -r backend/requirements.txt

# Setup Elasticsearch
python backend/pipelines/setup_elastic.py

# Ingest sample documents
python backend/pipelines/ingest.py data/sample_docs --category knowledge_base

# Run the application
python backend/app.py
```

Access the application:

- **Chat Interface**: http://localhost:8080
- **Analytics Dashboard**: http://localhost:8080/dashboard
- **API Health**: http://localhost:8080/api/health

Docker Deployment

Build Docker Image

```
docker build -t sentiflow .
```

Run Container

```
docker run -p 8080:8080 --env-file .env sentiflow
```

Push to Google Container Registry

```

```

```
# Tag image
docker tag sentiflow gcr.io/YOUR_PROJECT_ID/sentiflow

# Configure Docker
gcloud auth configure-docker

# Push
docker push gcr.io/YOUR_PROJECT_ID/sentiflow
```

Google Cloud Run Deployment

Option 1: Using Deployment Script

```
chmod +x deploy.sh
./deploy.sh
```

Option 2: Manual Deployment

```
# Set variables
export PROJECT_ID=your-project-id
export REGION=us-central1

# Deploy
gcloud run deploy sentiflow \
  --image gcr.io/${PROJECT_ID}/sentiflow \
  --platform managed \
  --region ${REGION} \
  --allow-unauthenticated \
  --set-env-vars "GCP_PROJECT_ID=${GCP_PROJECT_ID}" \
  --set-env-vars "ELASTIC_CLOUD_ID=${ELASTIC_CLOUD_ID}" \
  --set-env-vars "ELASTIC_API_KEY=${ELASTIC_API_KEY}" \
  --memory 2Gi \
  --cpu 2 \
  --timeout 300 \
  --max-instances 10
```

Get your service URL:

```
gcloud run services describe sentiflow \
  --platform managed \
  --region ${REGION} \
  --format 'value(status.url)'
```

🧪 Testing the Deployment

1. Health Check

```
curl https://YOUR_SERVICE_URL/api/health
```

2. Test Chat API

```
curl -X POST https://YOUR_SERVICE_URL/api/chat \
  -H "Content-Type: application/json" \
  -d '{"message": "What is your return policy?"}'
```

3. Test Sentiment Analysis

```
curl -X POST https://YOUR_SERVICE_URL/api/sentiment \
  -H "Content-Type: application/json" \
  -d '{"text": "This product is amazing!"}'
```

📊 Monitoring & Logs

View Cloud Run Logs

```
gcloud logging read "resource.type=cloud_run_revision AND
resource.labels.service_name=sentiflow" \
  --limit 50 \
  --format json
```

View Metrics

- Go to Cloud Console → Cloud Run → sentiflow
- Check metrics dashboard

🔧 Troubleshooting

Common Issues

1. Import errors (vertexai, elasticsearch)

- Solution: Install dependencies: `pip install -r backend/requirements.txt`

2. Authentication errors

- Solution: Check `.env` file credentials

- Verify GCP authentication: `gcloud auth list`

3. Elasticsearch connection failed

- Solution: Verify `ELASTIC_CLOUD_ID` and `ELASTIC_API_KEY`
- Check Elastic Cloud deployment is running

4. Memory errors on Cloud Run

- Solution: Increase memory in deployment: `--memory 4Gi`

5. Cold start timeouts

- Solution:
  - Increase timeout: `--timeout 600`
  - Set minimum instances: `--min-instances 1`

Performance Optimization

For Production:

1. **Enable Caching**
   - Add Redis for conversation history
   - Cache embedding results
2. **Optimize Elasticsearch**
   - Increase shard count for large datasets
   - Use index lifecycle management
3. **Scale Cloud Run**
   - Set appropriate min/max instances
   - Use CPU-based autoscaling
4. **Monitor Costs**
   - Track Vertex AI API calls
   - Monitor Elasticsearch queries
   - Set budget alerts

Hackathon Submission Checklist

- [ ] Application deployed to Cloud Run
- [ ] Public URL accessible
- [ ] Sample documents ingested
- [ ] All features working:
  - [ ] Chat interface
  - [ ] Sentiment analysis
  - [ ] Hybrid search

    - [ ] Analytics dashboard
- [ ] README.md complete with:
    - [ ] Architecture diagram
    - [ ] Technology stack
    - [ ] Setup instructions
    - [ ] Demo video/screenshots
- [ ] GitHub repository:
    - [ ] All code committed
    - [ ] Proper .gitignore
    - [ ] Clear documentation

📞 Support

For issues during hackathon:

- Check logs: `gcloud logging read`
- Verify environment: `python backend/config.py`
- Test components individually

🏆 Demo Tips

1. **Prepare Test Queries**:
    - "What is your return policy?"
    - "I'm frustrated! My order hasn't arrived!"
    - "How do I track my shipment?"
2. **Show Key Features**:
    - Real-time sentiment detection
    - Context-aware responses
    - Analytics dashboard
    - Hybrid search results
3. **Highlight Innovation**:
    - Gemini 2.0 Flash integration
    - Elastic hybrid search (vector + keyword)
    - Real-time sentiment adaptation
    - Production-ready architecture

Good luck with your submission! 🚀