Snowflake Streamlit Deployment Guide

Prerequisites

1. Snowflake account with appropriate permissions
2. Snowflake CLI (`snow`) installed and configured
3. Model files and data files ready for upload

Step 1: Install Snowflake CLI

bash

```
# Install Snowflake CLI
pip install snowflake-cli-labs

# Configure connection
snow connection add
# Follow prompts to enter:
#   - Connection name: my_snowflake_connection
#   - Account identifier: <YOUR_ACCOUNT>.snowflakecomputing.com
#   - Username: <YOUR_USERNAME>
#   - Password: <YOUR_PASSWORD>
#   - Role: <YOUR_ROLE>
#   - Warehouse: SHAP_MAP_WH
#   - Database: SHAP_MAP_DB
#   - Schema: GEOSPATIAL_ML

# Test connection
snow connection test --connection my_snowflake_connection
```

Step 2: Create Snowflake Resources

sql

```
-- Connect to Snowflake using SnowSQL or Snowsight
snowsql -a <ACCOUNT> -u <USERNAME>

-- Create resources
USE ROLE ACCOUNTADMIN;  -- Or your admin role

CREATE DATABASE IF NOT EXISTS SHAP_MAP_DB;
CREATE SCHEMA IF NOT EXISTS SHAP_MAP_DB.GEOSPATIAL_ML;

CREATE WAREHOUSE IF NOT EXISTS SHAP_MAP_WH
  WITH WAREHOUSE_SIZE = 'MEDIUM'
  AUTO_SUSPEND = 300
  AUTO_RESUME = TRUE
  INITIALLY_SUSPENDED = TRUE;

CREATE STAGE IF NOT EXISTS SHAP_MAP_DB.GEOSPATIAL_ML.ML_ASSETS
  DIRECTORY = (ENABLE = TRUE);

-- Grant permissions (replace <YOUR_ROLE> with your role name)
GRANT USAGE ON WAREHOUSE SHAP_MAP_WH TO ROLE <YOUR_ROLE>;
GRANT USAGE ON DATABASE SHAP_MAP_DB TO ROLE <YOUR_ROLE>;
GRANT USAGE ON SCHEMA SHAP_MAP_DB.GEOSPATIAL_ML TO ROLE <YOUR_ROLE>;
GRANT ALL ON STAGE SHAP_MAP_DB.GEOSPATIAL_ML.ML_ASSETS TO ROLE <YOUR_ROLE>;

-- Verify stage creation
LIST @SHAP_MAP_DB.GEOSPATIAL_ML.ML_ASSETS;
```

Step 3: Upload Files to Stage

bash

```
# Navigate to your project directory with model and data files
cd /path/to/your/project

# Upload equipment model files
snow stage put \
  xgboost_model_v3.json \
  @SHAP_MAP_DB.GEOSPATIAL_ML.ML_ASSETS/models/ \
  --connection my_snowflake_connection \
  --overwrite

snow stage put \
  ocp_wiredown_model_data.gpkg \
  @SHAP_MAP_DB.GEOSPATIAL_ML.ML_ASSETS/data/ \
  --connection my_snowflake_connection \
  --overwrite

snow stage put \
  feature_names.json \
  @SHAP_MAP_DB.GEOSPATIAL_ML.ML_ASSETS/models/ \
  --connection my_snowflake_connection \
  --overwrite

# Upload transformer model files (if you have them)
snow stage put \
  transformer_model.bst \
  @SHAP_MAP_DB.GEOSPATIAL_ML.ML_ASSETS/models/ \
  --connection my_snowflake_connection \
  --overwrite

snow stage put \
  transformer_model_data.gpkg \
  @SHAP_MAP_DB.GEOSPATIAL_ML.ML_ASSETS/data/ \
  --connection my_snowflake_connection \
  --overwrite

snow stage put \
  structure_feature_names.json \
  @SHAP_MAP_DB.GEOSPATIAL_ML.ML_ASSETS/models/ \
  --connection my_snowflake_connection \
  --overwrite

# Verify uploads
```

```
snow stage list @SHAP_MAP_DB.GEOSPATIAL_ML.ML_ASSETS/ \
  --connection my_snowflake_connection
```

Step 4: Prepare Streamlit Application

Create the following directory structure:

```
my_shap_map_app/
├── streamlit_app.py          # The complete code from the artifact
├── environment.yml           # Conda dependencies
└── .streamlit/
    └── secrets.toml          # Snowflake credentials (for local testing)
```

environment.yml:

yaml

```yaml
name: shap_map_env
channels:
  - snowflake
  - conda-forge
dependencies:
  - python=3.9
  - snowflake-snowpark-python
  - streamlit>=1.28.0
  - pandas>=1.5.0
  - numpy>=1.23.0
  - geopandas>=0.12.0
  - shapely>=2.0.0
  - fiona>=1.9.0
  - xgboost>=1.7.0
  - shap>=0.42.0
  - matplotlib>=3.6.0
  - pyproj>=3.4.0
```

.streamlit/secrets.toml (for local testing only):

toml

```
[snowflake]
account = "<YOUR_ACCOUNT>"
user = "<YOUR_USERNAME>"
password = "<YOUR_PASSWORD>"
role = "<YOUR_ROLE>"
warehouse = "SHAP_MAP_WH"
database = "SHAP_MAP_DB"
schema = "GEOSPATIAL_ML"
```

Step 5: Test Locally (Optional)

bash

```
# Install dependencies
conda env create -f environment.yml
conda activate shap_map_env

# Run Streamlit locally
streamlit run streamlit_app.py

# Access at http://localhost:8501
```

Step 6: Deploy to Snowflake

bash

```
# Navigate to app directory
cd my_shap_map_app

# Deploy to Snowflake
snow streamlit deploy \
  --connection my_snowflake_connection \
  --warehouse SHAP_MAP_WH \
  --database SHAP_MAP_DB \
  --schema GEOSPATIAL_ML \
  --replace

# Note: The app will be named based on your directory name
# You can specify a custom name with --name flag:
snow streamlit deploy \
  --name SHAP_MAP_APP \
  --connection my_snowflake_connection \
  --warehouse SHAP_MAP_WH \
  --database SHAP_MAP_DB \
  --schema GEOSPATIAL_ML \
  --replace
```

Step 7: Access Your Application

Option 1: Via Snowsight

1. Log in to Snowsight (https://app.snowflake.com)
2. Navigate to **Data** > **Databases**
3. Select `SHAP_MAP_DB` > `GEOSPATIAL_ML`
4. Click on **Streamlit** in the left sidebar
5. Click on your app name (e.g., `SHAP_MAP_APP`)
6. The application will launch in a new tab

Option 2: Via Direct URL

The URL format is:

```
https://app.snowflake.com/<ORGNAME>/<ACCOUNT>/#/streamlit-apps/SHAP_MAP_DB.GEOSPATIAL_ML.SHAP_MAP_APP
```

Step 8: Grant Access to Users

```sql
-- Grant access to other users/roles
USE ROLE ACCOUNTADMIN;

-- Grant usage on database and schema
GRANT USAGE ON DATABASE SHAP_MAP_DB TO ROLE <TARGET_ROLE>;
GRANT USAGE ON SCHEMA SHAP_MAP_DB.GEOSPATIAL_ML TO ROLE <TARGET_ROLE>;

-- Grant access to the Streamlit app
GRANT USAGE ON STREAMLIT SHAP_MAP_DB.GEOSPATIAL_ML.SHAP_MAP_APP TO ROLE <TARGET_ROLE>

-- Grant warehouse usage for compute
GRANT USAGE ON WAREHOUSE SHAP_MAP_WH TO ROLE <TARGET_ROLE>;

-- Grant read access to the stage
GRANT READ ON STAGE SHAP_MAP_DB.GEOSPATIAL_ML.ML_ASSETS TO ROLE <TARGET_ROLE>;
```

Step 9: Update and Redeploy

When you need to update the application:

```bash
# Make changes to streamlit_app.py

# Redeploy (use --replace to overwrite existing app)
snow streamlit deploy \
  --name SHAP_MAP_APP \
  --connection my_snowflake_connection \
  --warehouse SHAP_MAP_WH \
  --database SHAP_MAP_DB \
  --schema GEOSPATIAL_ML \
  --replace
```

Troubleshooting

Issue: "Module not found" errors

Solution: Ensure all dependencies are in `environment.yml`:

```yaml
dependencies:
  - python=3.9
  - snowflake-snowpark-python
  - streamlit
  - pandas
  - numpy
  - geopandas
  - shapely
  - fiona
  - xgboost
  - shap
  - matplotlib
  - pyproj
```

Issue: "File not found" when loading from stage

Solution: Verify stage paths and file existence:

```sql
-- List all files in stage
LIST @SHAP_MAP_DB.GEOSPATIAL_ML.ML_ASSETS;

-- Check specific directories
LIST @SHAP_MAP_DB.GEOSPATIAL_ML.ML_ASSETS/models/;
LIST @SHAP_MAP_DB.GEOSPATIAL_ML.ML_ASSETS/data/;
```

Issue: Application is slow

Solutions:

1. Increase warehouse size:

```sql
ALTER WAREHOUSE SHAP_MAP_WH SET WAREHOUSE_SIZE = 'LARGE';
```

2. Optimize data loading with caching:
   - The `@st.cache_resource` decorator is already used
   - Ensure it's not being cleared unnecessarily

3. Reduce viewport limit in sidebar settings

Issue: SHAP plots fail to generate

Solution: Check that:

1. Model file is compatible with xgboost version
2. Feature names match exactly between model and data
3. There's enough memory (increase warehouse size)

Issue: Geometry rendering problems

Solution:

1. Verify GeoPackage CRS is EPSG:4326 or can be reprojected
2. Check for null/invalid geometries in data:

```python
# Add to streamlit_app.py for debugging
st.write(f"Null geometries: {gdf.geometry.isna().sum()}")
st.write(f"Invalid geometries: {(~gdf.geometry.is_valid).sum()}")
```

Performance Optimization

1. Warehouse Sizing

Start with MEDIUM, scale up as needed:

```sql
-- For development/testing
ALTER WAREHOUSE SHAP_MAP_WH SET WAREHOUSE_SIZE = 'SMALL';

-- For production with multiple users
ALTER WAREHOUSE SHAP_MAP_WH SET WAREHOUSE_SIZE = 'LARGE';

-- For heavy SHAP computations
ALTER WAREHOUSE SHAP_MAP_WH SET WAREHOUSE_SIZE = 'XLARGE';
```

2. Data Optimization

Optimize GeoPackage files before uploading:

```python
# Pre-processing script (run before upload)
import geopandas as gpd

# Load data
gdf = gpd.read_file("your_data.gpkg")

# Remove unnecessary columns
keep_cols = ['ID', 'geometry', 'feature1', 'feature2', ...]  # Only what you need
gdf = gdf[keep_cols]

# Simplify geometries if too complex
gdf['geometry'] = gdf.geometry.simplify(tolerance=0.0001, preserve_topology=True)

# Ensure EPSG:4326
gdf = gdf.to_crs(epsg=4326)

# Save optimized version
gdf.to_file("optimized_data.gpkg", driver="GPKG")
```

3. Caching Strategy

The application uses `@st.cache_resource` for model loading. Monitor cache behavior:

```python
# Add to streamlit_app.py for monitoring
st.sidebar.markdown("---")
st.sidebar.markdown("### 🔍 Cache Info")
if st.sidebar.button("Clear Cache"):
    st.cache_resource.clear()
    st.rerun()
```

4. Viewport Loading

Adjust `LIMIT` based on zoom level (advanced):

```python
# Add to streamlit_app.py
zoom_level = st.session_state.get('map_zoom', 10)
dynamic_limit = min(5000, int(1000 * (15 - zoom_level)))  # More features at higher zoom
```

Monitoring and Logging

View Streamlit Logs

```sql
-- Query Streamlit app logs
SELECT *
FROM TABLE(INFORMATION_SCHEMA.STREAMLIT_LOGS(
    STREAMLIT_NAME => 'SHAP_MAP_DB.GEOSPATIAL_ML.SHAP_MAP_APP',
    START_TIME => DATEADD('hour', -1, CURRENT_TIMESTAMP())
))
ORDER BY TIMESTAMP DESC;
```

Monitor Warehouse Usage

```sql
-- Check warehouse credit usage
SELECT
    WAREHOUSE_NAME,
    SUM(CREDITS_USED) as TOTAL_CREDITS,
    COUNT(*) as QUERY_COUNT
FROM SNOWFLAKE.ACCOUNT_USAGE.WAREHOUSE_METERING_HISTORY
WHERE WAREHOUSE_NAME = 'SHAP_MAP_WH'
    AND START_TIME >= DATEADD('day', -7, CURRENT_TIMESTAMP())
GROUP BY WAREHOUSE_NAME;

-- Check query performance
SELECT
    QUERY_ID,
    QUERY_TEXT,
    EXECUTION_TIME,
    WAREHOUSE_SIZE
FROM SNOWFLAKE.ACCOUNT_USAGE.QUERY_HISTORY
WHERE WAREHOUSE_NAME = 'SHAP_MAP_WH'
    AND START_TIME >= DATEADD('hour', -1, CURRENT_TIMESTAMP())
ORDER BY EXECUTION_TIME DESC
LIMIT 10;
```

Advanced Configuration

Multi-Environment Setup

Create separate environments for dev/staging/prod:

```sql
-- Development
CREATE DATABASE SHAP_MAP_DEV;
CREATE SCHEMA SHAP_MAP_DEV.GEOSPATIAL_ML;
CREATE STAGE SHAP_MAP_DEV.GEOSPATIAL_ML.ML_ASSETS;
CREATE WAREHOUSE SHAP_MAP_DEV_WH WITH WAREHOUSE_SIZE = 'XSMALL';

-- Production
CREATE DATABASE SHAP_MAP_PROD;
CREATE SCHEMA SHAP_MAP_PROD.GEOSPATIAL_ML;
CREATE STAGE SHAP_MAP_PROD.GEOSPATIAL_ML.ML_ASSETS;
CREATE WAREHOUSE SHAP_MAP_PROD_WH WITH WAREHOUSE_SIZE = 'LARGE';
```

Deploy to different environments:

```bash
# Deploy to dev
snow streamlit deploy \
  --name SHAP_MAP_APP_DEV \
  --database SHAP_MAP_DEV \
  --schema GEOSPATIAL_ML \
  --warehouse SHAP_MAP_DEV_WH \
  --connection my_snowflake_connection

# Deploy to prod
snow streamlit deploy \
  --name SHAP_MAP_APP \
  --database SHAP_MAP_PROD \
  --schema GEOSPATIAL_ML \
  --warehouse SHAP_MAP_PROD_WH \
  --connection my_snowflake_connection
```

Using Snowflake Tables Instead of GeoPackage

For very large datasets, consider loading data into Snowflake tables:

sql

```sql
-- Create table for equipment data
CREATE TABLE SHAP_MAP_DB.GEOSPATIAL_ML.EQUIPMENT_DATA (
    ID VARCHAR,
    HAS_WIREDOWN INTEGER,
    POF FLOAT,
    GEOMETRY GEOGRAPHY,
    -- Add all your feature columns
    FEATURE1 FLOAT,
    FEATURE2 FLOAT,
    -- ... more features
);

-- Load data from stage (after converting to CSV/Parquet)
COPY INTO SHAP_MAP_DB.GEOSPATIAL_ML.EQUIPMENT_DATA
FROM @SHAP_MAP_DB.GEOSPATIAL_ML.ML_ASSETS/data/equipment_data.parquet
FILE_FORMAT = (TYPE = PARQUET);
```

Then modify the loading code in `streamlit_app.py`:

python

```python
# Replace load_geopackage_from_stage with:
def load_data_from_table(table_name: str) -> gpd.GeoDataFrame:
    """Load data from Snowflake table"""
    df = session.table(table_name).to_pandas()

    # Convert GEOGRAPHY to shapely geometries
    from shapely import wkb
    df['geometry'] = df['GEOMETRY'].apply(lambda x: wkb.loads(bytes.fromhex(x)))

    gdf = gpd.GeoDataFrame(df, geometry='geometry', crs='EPSG:4326')
    return gdf
```

Security Best Practices

1. Role-Based Access Control

sql

```sql
-- Create custom roles
CREATE ROLE SHAP_MAP_VIEWER;
CREATE ROLE SHAP_MAP_ADMIN;

-- Grant viewer permissions (read-only)
GRANT USAGE ON DATABASE SHAP_MAP_DB TO ROLE SHAP_MAP_VIEWER;
GRANT USAGE ON SCHEMA SHAP_MAP_DB.GEOSPATIAL_ML TO ROLE SHAP_MAP_VIEWER;
GRANT USAGE ON STREAMLIT SHAP_MAP_DB.GEOSPATIAL_ML.SHAP_MAP_APP TO ROLE SHAP_MAP_VIEV
GRANT USAGE ON WAREHOUSE SHAP_MAP_WH TO ROLE SHAP_MAP_VIEWER;
GRANT READ ON STAGE SHAP_MAP_DB.GEOSPATIAL_ML.ML_ASSETS TO ROLE SHAP_MAP_VIEWER;

-- Grant admin permissions (can update models/data)
GRANT ALL ON DATABASE SHAP_MAP_DB TO ROLE SHAP_MAP_ADMIN;
GRANT ALL ON SCHEMA SHAP_MAP_DB.GEOSPATIAL_ML TO ROLE SHAP_MAP_ADMIN;
GRANT ALL ON STAGE SHAP_MAP_DB.GEOSPATIAL_ML.ML_ASSETS TO ROLE SHAP_MAP_ADMIN;

-- Assign roles to users
GRANT ROLE SHAP_MAP_VIEWER TO USER data_analyst_user;
GRANT ROLE SHAP_MAP_ADMIN TO USER ml_engineer_user;
```

2. Secrets Management

Never hardcode credentials in `streamlit_app.py`. Always use Snowflake secrets:

python

```python
# Good - using Snowflake connection
session = st.connection("snowflake").session()

# Bad - hardcoding credentials
# connection_parameters = {
#     "account": "myaccount",
#     "password": "hardcoded_password"  # NEVER DO THIS
# }
```

3. Network Policies

sql

```
-- Restrict access to specific IP ranges
CREATE NETWORK POLICY SHAP_MAP_POLICY
  ALLOWED_IP_LIST = ('192.168.1.0/24', '10.0.0.0/8');

ALTER USER data_analyst_user SET NETWORK_POLICY = SHAP_MAP_POLICY;
```

Maintenance Tasks

Regular Updates

Monthly:

1. Check warehouse credit usage
2. Review query performance logs
3. Update model files if retrained
4. Check for Streamlit/dependency updates

Quarterly:

1. Review user access permissions
2. Optimize data storage (remove old versions)
3. Update documentation
4. Evaluate warehouse sizing

Model Updates

To update models without redeploying the entire app:

```bash
# Upload new model version
snow stage put new_model_v4.json \
  @SHAP_MAP_DB.GEOSPATIAL_ML.ML_ASSETS/models/ \
  --overwrite

# Update MODEL_CONFIGS in streamlit_app.py to point to new file
# Then redeploy
snow streamlit deploy --name SHAP_MAP_APP --replace
```

Data Refresh

```bash
# Upload updated data
snow stage put updated_data.gpkg \
  @SHAP_MAP_DB.GEOSPATIAL_ML.ML_ASSETS/data/ \
  --overwrite

# Clear cache in app to reload data
# (Add cache clear button in sidebar as shown above)
```

Cost Management

1. Auto-Suspend Warehouses

```sql
ALTER WAREHOUSE SHAP_MAP_WH SET
    AUTO_SUSPEND = 60  -- Suspend after 1 minute of inactivity
    AUTO_RESUME = TRUE;
```

2. Set Resource Monitors

```sql
CREATE RESOURCE MONITOR SHAP_MAP_MONITOR
WITH CREDIT_QUOTA = 100  -- 100 credits per month
    FREQUENCY = MONTHLY
    START_TIMESTAMP = IMMEDIATELY
    TRIGGERS
        ON 75 PERCENT DO NOTIFY
        ON 100 PERCENT DO SUSPEND
        ON 110 PERCENT DO SUSPEND_IMMEDIATE;

ALTER WAREHOUSE SHAP_MAP_WH SET RESOURCE_MONITOR = SHAP_MAP_MONITOR;
```

3. Monitor Costs

```sql
```

```
-- Daily credit usage
SELECT
    DATE(START_TIME) as DATE,
    SUM(CREDITS_USED) as DAILY_CREDITS
FROM SNOWFLAKE.ACCOUNT_USAGE.WAREHOUSE_METERING_HISTORY
WHERE WAREHOUSE_NAME = 'SHAP_MAP_WH'
    AND START_TIME >= DATEADD('day', -30, CURRENT_TIMESTAMP())
GROUP BY DATE(START_TIME)
ORDER BY DATE DESC;
```

Backup and Recovery

Backup Models and Data

bash

```
# Download files from stage to local backup
snow stage get @SHAP_MAP_DB.GEOSPATIAL_ML.ML_ASSETS/models/ \
  /local/backup/models/ \
  --connection my_snowflake_connection

snow stage get @SHAP_MAP_DB.GEOSPATIAL_ML.ML_ASSETS/data/ \
  /local/backup/data/ \
  --connection my_snowflake_connection
```

Version Control

Keep versions of your models:

bash

```
# Upload with version suffix
snow stage put xgboost_model_v3.json \
  @SHAP_MAP_DB.GEOSPATIAL_ML.ML_ASSETS/models/archive/xgboost_model_v3_2025_01.json
```

Additional Resources

- **Snowflake Streamlit Documentation:** https://docs.snowflake.com/en/developer-guide/streamlit/about-streamlit
- **Snowflake CLI Documentation:** https://docs.snowflake.com/en/developer-guide/snowflake-cli/index
- **Streamlit Documentation:** https://docs.streamlit.io/

- **XGBoost Documentation:** https://xgboost.readthedocs.io/
- **SHAP Documentation:** https://shap.readthedocs.io/
- **Geopandas Documentation:** https://geopandas.org/

Support and Community

- **Snowflake Community:** https://community.snowflake.com/
- **Streamlit Community:** https://discuss.streamlit.io/
- **GitHub Issues:** Report issues specific to your deployment

Checklist for Go-Live

- [ ] All resources created (database, schema, warehouse, stage)
- [ ] Files uploaded to stage and verified
- [ ] Permissions granted to all users/roles
- [ ] Application deployed and tested
- [ ] Resource monitors configured
- [ ] Auto-suspend enabled on warehouse
- [ ] Documentation shared with users
- [ ] Backup procedures established
- [ ] Monitoring queries saved
- [ ] Support contacts identified

Note: Replace all placeholder values (`<YOUR_ACCOUNT>`, `<YOUR_ROLE>`, etc.) with your actual Snowflake account details before executing commands.