multi-cloud-storage-lib

A unified interface for AWS S3, Azure Blob Storage, and Google Cloud Storage with factory pattern support.

Installation

```
npm install multi-cloud-storage-lib
```

Features

- ✅ **Multi-Cloud Support**: AWS S3, Azure Blob Storage, and Google Cloud Storage
- ✅ **Factory Pattern**: Automatic provider detection and service creation
- ✅ **Unified API**: Same interface for all cloud storage providers
- ✅ **Path Detection**: Auto-detect provider from path (s3://, azure://, gs://)
- ✅ **TypeScript Ready**: Full TypeScript support (coming soon)

Supported Providers

- **AWS S3** (`s3://`, `aws`)
- **Azure Blob Storage** (`azure://`, `az://`, `azure`)
- **Google Cloud Storage** (`gs://`, `gcp://`, `gcp`, `gcs`, `google`)

Quick Start

Basic Usage

PROF

```
const { CloudStorageFactory } = require('multi-cloud-storage-lib');

// Auto-detect provider from path
const path = 's3://my-bucket/path/to/file.csv';
const provider = CloudStorageFactory.getProviderFromPath(path);

// Configuration
const config = {
  credentials: {
    accessKey: 'your-access-key',
    secretKey: 'your-secret-key',
  },
  region: 'us-east-1',
};

// Create storage service
const storage = CloudStorageFactory.createStorageService(provider,
config);
```

```
// Validate authentication
const authResult = await storage.validateAuthentication();
console.log(authResult); // { status: 'success', message:
'Authentication verified' }

// Get object
const content = await storage.getObject('s3://my-
bucket/path/to/file.csv');

// List objects
const objects = await storage.listObjects('s3://my-bucket/path/');
```

Configuration

AWS S3

```
const config = {
  provider: 's3', // or 'aws'
  credentials: {
    accessKey: 'your-access-key-id',
    secretKey: 'your-secret-access-key',
  },
  region: 'us-east-1',
};
```

Azure Blob Storage

```
const config = {
  provider: 'azure',
  credentials: {
    // Option 1: Connection string
    connectionString: 'DefaultEndpointsProtocol=https;AccountName=...',

    // Option 2: Account name and key
    accountName: 'your-account-name',
    accountKey: 'your-account-key',
  },
};
```

PROF

Google Cloud Storage

```
const config = {
  provider: 'gcp',
  credentials: {
```

```
    // Option 1: Service account key file path
    keyFilename: '/path/to/service-account-key.json',

    // Option 2: Service account credentials object
    projectId: 'your-project-id',
    clientEmail: 'service-account@project.iam.gserviceaccount.com',
    privateKey: '-----BEGIN PRIVATE KEY-----\n...\n-----END PRIVATE KEY-
----\n',
  },
};
```

API Reference

CloudStorageFactory

`createStorageService(provider, config)`

Creates a storage service instance for the specified provider.

Parameters:

- `provider` (string): Storage provider ('s3', 'aws', 'azure', 'gcp', 'gcs', 'google')
- `config` (object): Provider-specific configuration

Returns: Storage service instance

`getProviderFromPath(path)`

Detects the storage provider from a path string.

Parameters:

- `path` (string): Storage path (e.g., 's3://bucket/path', 'azure://container/path')

PROF

Returns: Provider name string or null

CloudStorageInterface

All storage services implement this interface:

`validateAuthentication()`

Validates authentication credentials.

Returns: Promise<Object> with `{ status: 'success'|'failed', message?: string, error?: string }`

`getObject(filePath)`

Retrieves an object from storage.

Parameters:

- `filePath` (string): Full path to the file (e.g., 's3://bucket/path/file.txt')

Returns: Promise<string> - File content as string

`listObjects(path)`

Lists objects in a storage path.

Parameters:

- `path` (string): Storage path to list

Returns: Promise<Array> - Array of object metadata

`parsePath(path)` **(static)**

Parses a storage path into bucket/container and file path.

Parameters:

- `path` (string): Storage path

Returns: Object with `{ containerName, bucketName, path }`

Examples

Example 1: AWS S3

```
const { CloudStorageFactory } = require('multi-cloud-storage-lib');

const config = {
  credentials: {
    accessKey: process.env.AWS_ACCESS_KEY_ID,
    secretKey: process.env.AWS_SECRET_ACCESS_KEY,
  },
  region: 'us-east-1',
};

const storage = CloudStorageFactory.createStorageService('s3', config);
const content = await storage.getObject('s3://my-bucket/data/file.csv');
```

PROF

Example 2: Azure Blob Storage

```
const { CloudStorageFactory } = require('multi-cloud-storage-lib');

const config = {
  credentials: {
```

```
    accountName: process.env.AZURE_ACCOUNT_NAME,
    accountKey: process.env.AZURE_ACCOUNT_KEY,
  },
};

const storage = CloudStorageFactory.createStorageService('azure',
config);
const content = await storage.getObject('azure://my-
container/data/file.csv');
```

Example 3: Google Cloud Storage

```
const { CloudStorageFactory } = require('multi-cloud-storage-lib');

const config = {
  credentials: {
    keyFilename: './service-account-key.json',
  },
};

const storage = CloudStorageFactory.createStorageService('gcp', config);
const content = await storage.getObject('gs://my-bucket/data/file.csv');
```

Example 4: Auto-detect Provider

```
const { CloudStorageFactory } = require('multi-cloud-storage-lib');

const path = 's3://my-bucket/path/to/file.csv';
const provider = CloudStorageFactory.getProviderFromPath(path); //
Returns 's3'

const config = {
  credentials: {
    accessKey: process.env.AWS_ACCESS_KEY_ID,
    secretKey: process.env.AWS_SECRET_ACCESS_KEY,
  },
  region: 'us-east-1',
};

const storage = CloudStorageFactory.createStorageService(provider,
config);
const content = await storage.getObject(path);
```

PROF

Path Formats

Supported path formats:

- **AWS S3**: `s3://bucket-name/path/to/file`
- **Azure Blob**: `azure://container-name/path/to/file` or `az://container-name/path/to/file`
- **Google Cloud**: `gs://bucket-name/path/to/file` or `gcp://bucket-name/path/to/file`

Error Handling

All methods throw errors that should be caught:

```
try {
  const content = await
storage.getObject('s3://bucket/nonexistent.txt');
} catch (error) {
  console.error('Failed to get object:', error.message);
}
```

License

MIT

Contributing

Contributions are welcome! Please feel free to submit a Pull Request.