

Provider Architecture Summary

Overview

The Social Media Service now implements a **flexible provider architecture** that allows seamless integration with multiple social media management platforms. This architecture supports **Ayrshare** (default) and **Buffer** as providers, with the ability to easily add more providers in the future.

Architecture Components

1. Base Provider Abstraction

File: app/services/providers/base_provider.py

- **Purpose:** Defines the interface that all social media providers must implement
- **Key Classes:**
 - `SocialMediaProvider` - Abstract base class with required methods
 - `ProviderError` - Unified exception handling across providers

Required Methods:

- ```
- authenticate() -> Dict[str, Any]
- get_profiles() -> List[Dict[str, Any]]
- create_post(profile_ids, text, media, scheduled_at) -> Dict[str, Any]
- update_post(post_id, data) -> Dict[str, Any]
- delete_post(post_id) -> Dict[str, Any]
- get_post_analytics(post_id) -> Dict[str, Any]
- test_connection() -> bool
```

### 2. Ayrshare Provider (Default)

**File:** app/services/providers/ayrshare\_provider.py

- **Purpose:** Implements Ayrshare API integration
- **API Endpoint:** <https://app/ayrshare.com/api>
- **Authentication:** Bearer token (API key)
- **Supported Platforms:**
  - Facebook, Twitter/X, Instagram, LinkedIn
  - TikTok, YouTube, Pinterest, Reddit
  - Telegram, Threads, Bluesky, Google Business, Snapchat

**Key Features:**

- Multi-platform posting
- Post scheduling with ISO 8601 timestamps
- Analytics and insights
- Multi-user accounts
- Webhooks for post status
- **White-label reselling support**

**Business Model:**

- Business Plan: **\$499/month**
- Supports white-label reselling
- Revenue model ready

### 3. Buffer Provider (Alternative)

**File:** app/services/providers/buffer\_provider.py

- **Purpose:** Implements Buffer API integration (refactored from original BufferService)
- **API Endpoint:** <https://api.bufferapp.com/1>
- **Authentication:** Access token (OAuth)
- **Supported Platforms:** Facebook, Twitter, Instagram, LinkedIn, Pinterest

**Key Features:**

- Social media scheduling
- Queue management
- Basic analytics
- Profile management

### 4. Provider Factory

**File:** app/services/providers/provider\_factory.py

- **Purpose:** Factory pattern for creating provider instances
- **Key Components:**
  - `ProviderFactory.create()` - Creates provider based on configuration
  - `get_provider()` - Convenience function for dependency injection
  - Provider registry for extensibility

**Usage Example:**

```
from app.services.providers.provider_factory import get_provider

Get default provider (from settings)
provider = get_provider()

Get specific provider
provider = get_provider('ayrshare', api_key='your-key')
provider = get_provider('buffer', access_token='your-token')

Use provider
profiles = await provider.get_profiles()
post = await provider.create_post(
 profile_ids=['profile1', 'profile2'],
 text='Hello, world!',
 scheduled_at=datetime(2025, 1, 1, 12, 0, 0)
)
```

## Configuration

### Environment Variables

Add to `.env` file:

```
Choose your provider
SOCIAL_MEDIA_PROVIDER="ayrshare" # or "buffer"

Ayrshare Configuration (Default)
AYRSHARE_API_URL="https://app.ayrshare.com/api"
AYRSHARE_API_KEY="your-ayrshare-api-key"

Buffer Configuration (Alternative)
BUFFER_API_URL="https://api.bufferapp.com/1"
BUFFER_ACCESS_TOKEN="your-buffer-access-token"
```

## Application Settings

File: app/core/config.py

```
class Settings(BaseSettings):
 # Social Media Provider Configuration
 SOCIAL_MEDIA_PROVIDER: str = "ayrshare" # Default provider

 # Ayrshare API Configuration
 AYRSHARE_API_URL: str = "https://app.ayrshare.com/api"
 AYRSHARE_API_KEY: Optional[str] = None

 # Buffer API Configuration
 BUFFER_API_URL: str = "https://api.bufferapp.com/1"
 BUFFER_ACCESS_TOKEN: Optional[str] = None
```

## Service Layer Updates

All service layers have been updated to use the provider abstraction:

### Updated Services

1. **SocialAccountService** ( app/services/social\_account\_service.py )
  - Method renamed: sync\_with\_buffer() → sync\_with\_provider()
  - Method updated: test\_connection() - now uses provider abstraction
  - Removed: buffer\_service parameter from all methods
2. **ScheduledPostService** ( app/services/scheduled\_post\_service.py )
  - Method renamed: schedule\_with\_buffer() → schedule\_with\_provider()
  - All methods now use get\_provider() internally
  - Media format updated for cross-provider compatibility
3. **PostAnalyticsService** ( app/services/post\_analytics\_service.py )
  - Analytics retrieval now uses provider abstraction
  - Supports standardized analytics format
4. **BufferConfigService** ( app/services/buffer\_config\_service.py )
  - Imports updated to use provider abstraction
  - Backward compatible with existing Buffer configuration

### Error Handling

- All BufferAPIError exceptions replaced with ProviderError
- Consistent error handling across providers

- Detailed error messages and logging

## API Endpoints

---

All API endpoints continue to work with the new provider architecture:

- `/api/v1/social-accounts/*` - Social account management
- `/api/v1/posts/*` - Post scheduling and management
- `/api/v1/analytics/*` - Analytics retrieval
- `/api/v1/campaigns/*` - Campaign management
- `/api/v1/buffer/config/*` - Provider configuration (legacy name)

## Benefits

---

### 1. Flexibility

- Switch between providers with a single configuration change
- No code changes required to change providers
- Support for multiple providers simultaneously (future)

### 2. Extensibility

- Easy to add new providers (Hootsuite, Sprout Social, etc.)
- Register custom providers via `ProviderFactory.register_provider()`
- Standardized interface ensures consistency

### 3. White-Label Revenue Model

- Ayrshare supports white-label reselling
- Build custom social media management solutions
- Revenue sharing opportunities

### 4. Maintainability

- Clean separation of concerns
- Provider-specific logic isolated in provider classes
- Service layers remain provider-agnostic

### 5. Testing

- Mock providers for unit testing
- Test different providers independently
- Consistent test interface

## Adding a New Provider

---

To add a new provider (e.g., Hootsuite):

## Step 1: Create Provider Class

```
app/services/providers/hootsuite_provider.py
from app.services.providers.base_provider import SocialMediaProvider, ProviderError

class HootsuiteProvider(SocialMediaProvider):
 def __init__(self, api_key: str):
 self.api_key = api_key
 self.base_url = "https://platform.hootsuite.com/v1"

 @async def authenticate(self) -> Dict[str, Any]:
 # Implement Hootsuite authentication
 pass

 @async def get_profiles(self) -> List[Dict[str, Any]]:
 # Implement profile retrieval
 pass

 # ... implement other required methods
```

## Step 2: Register Provider

```
app/services/providers/__init__.py
from app.services.providers.hootsuite_provider import HootsuiteProvider

__all__ = [
 'SocialMediaProvider',
 'AyrshareProvider',
 'BufferProvider',
 'HootsuiteProvider', # Add new provider
 'ProviderFactory',
 'get_provider',
]
```

```
app/services/providers/provider_factory.py
class ProviderFactory:
 _providers = {
 'ayrshare': AyrshareProvider,
 'buffer': BufferProvider,
 'hootsuite': HootsuiteProvider, # Register new provider
 }
```

## Step 3: Update Configuration

```
app/core/config.py
class Settings(BaseSettings):
 SOCIAL_MEDIA_PROVIDER: str = "ayrshare"

 # Hootsuite Configuration
 HOOTSUITE_API_URL: str = "https://platform.hootsuite.com/v1"
 HOOTSUITE_API_KEY: Optional[str] = None
```

## Step 4: Use New Provider

```
.env
SOCIAL_MEDIA_PROVIDER="hootsuite"
HOOTSUITE_API_KEY="your-api-key"
```

That's it! The service will now use Hootsuite as the provider.

## Migration from BufferService

### Changes Made

#### 1. BufferService → BufferProvider

- Moved from `app/services/buffer_service.py`
- Now implements `SocialMediaProvider` interface
- API interactions remain the same

#### 2. Imports Updated

- `from app.services.buffer_service import BufferService, BufferAPIError`
- → `from app.services.providers.provider_factory import get_provider`
- → `from app.services.providers.base_provider import ProviderError`

#### 3. Method Signatures

- Removed `buffer_service: BufferService` parameters
- Services now use `get_provider()` internally

#### 4. Backward Compatibility

- All existing functionality preserved
- API endpoints unchanged
- Database models unchanged

## Testing Migration

To test with Buffer (legacy provider):

```
SOCIAL_MEDIA_PROVIDER="buffer"
BUFFER_ACCESS_TOKEN="your-buffer-token"
```

To test with Ayrshare (new default):

```
SOCIAL_MEDIA_PROVIDER="ayrshare"
AYRSHARE_API_KEY="your-ayrshare-key"
```

## Future Enhancements

### 1. Multi-Provider Support

- Support multiple providers simultaneously
- Route posts to optimal provider based on platform
- Failover between providers

## 2. Provider-Specific Features

- Expose provider-specific features via metadata
- Advanced analytics for Ayrshare
- Queue optimization for Buffer

## 3. Provider Analytics

- Compare performance across providers
- Cost optimization recommendations
- Usage analytics and reporting

## 4. Dynamic Provider Loading

- Load providers from plugins
- Hot-reload provider implementations
- Community provider marketplace

## Ayrshare vs Buffer Comparison

| Feature                      | Ayrshare        | Buffer        |
|------------------------------|-----------------|---------------|
| <b>Supported Platforms</b>   | 13+ platforms   | 5 platforms   |
| <b>White-Label Reselling</b> | ✓ Yes           | ✗ No          |
| <b>Multi-User Accounts</b>   | ✓ Yes           | ⚠ Limited     |
| <b>Webhooks</b>              | ✓ Yes           | ⚠ Limited     |
| <b>Post Scheduling</b>       | ✓ Advanced      | ✓ Basic       |
| <b>Analytics</b>             | ✓ Comprehensive | ✓ Basic       |
| <b>API Rate Limits</b>       | Higher          | Lower         |
| <b>Pricing</b>               | \$499/month     | Variable      |
| <b>Revenue Model</b>         | ✓ Reselling     | ✗ Direct only |
| <b>Documentation</b>         | ✓ Excellent     | ✓ Good        |

## Conclusion

The provider architecture provides a **future-proof, flexible, and scalable** foundation for social media management. With **Ayrshare as the default provider**, the service is now ready for:

- ✓ White-label reselling
- ✓ Multi-platform support (13+ platforms)
- ✓ Revenue generation
- ✓ Easy provider switching

- Extensibility for future providers

**Next Steps:**

1. Configure Ayrshare API key in production
2. Test provider switching functionality
3. Update client applications (if needed)
4. Monitor provider performance
5. Implement multi-provider routing (future)

## Resources

---

- **Ayrshare Documentation:** <https://www.ayrshare.com/docs/introduction>
  - **Buffer API Documentation:** <https://buffer.com/developers/api>
  - **Provider Architecture Code:** `app/services/providers/`
  - **Configuration:** `app/core/config.py`
  - **Environment Template:** `.env.example`
- 

**Implementation Date:** December 24, 2025

**Status:** Complete

**Default Provider:** Ayrshare

**Alternative Provider:** Buffer

**Estimated Implementation Time:** 45-60 minutes

**Actual Time:** ~45 minutes