Python Scraper Configuration Changes for Abacus AI Storage

Date: December 13, 2025
Objective: Update Python scraper to work WITHOUT AWS S3, allowing deployment to Render with minimal configuration

Summary of Changes

The Python scraper has been updated to make AWS S3 storage **completely optional**. The scraper can now be deployed with just `DATABASE_URL` and `API_SECRET_KEY`, while still populating the database with player data.

✅ Changes Made

1. Storage Module (`storage/__init__.py`)

Before: S3 was always imported and required
After: Conditional S3 imports with auto-detection

```
# Check if S3 credentials are configured
S3_ENABLED = bool(
    os.getenv("AWS_ACCESS_KEY_ID") and
    os.getenv("AWS_SECRET_ACCESS_KEY") and
    os.getenv("S3_BUCKET_NAME")
)

# Conditionally import S3 functions
# Sets stub functions to None if S3 is not configured
```

Impact:
- ✅ Scraper starts without S3 credentials
- ✅ Logs warning: “S3 storage disabled”
- ⚠️ Image uploads are skipped

---

2. Main Pipeline (`main.py`)

Before: Always attempted S3 upload after downloading images
After: Skips S3 upload and database image insertion if S3 is disabled

```
if S3_ENABLED:
    # Upload images to S3
    # Insert image records into database
else:
    logger.warning("PHASE 2 & 3 SKIPPED: S3 storage not configured")
    logger.warning("Images downloaded locally but not uploaded")
```

Impact:
- ✅ Image download still works (local storage only)
- ✅ Player data still inserted into database
- ⚠️ Image URLs remain NULL in database

---

3. Backup Module (`backup/__init__.py`)

Before: Backup module always imported boto3
After: Conditional backup imports with auto-detection

```python
# Check if S3 backup is available
BACKUP_ENABLED = bool(
    os.getenv("AWS_ACCESS_KEY_ID") and
    os.getenv("AWS_SECRET_ACCESS_KEY") and
    os.getenv("BACKUP_BUCKET")
)

# Conditionally import backup functions
```

Impact:
- ✅ Scraper starts without backup configuration
- ✅ Logs warning: "Database backup disabled"
- ⚠️ Backup endpoints return 503 error

---

4. API Server (`app.py`)

Added Features:
- New decorator: `@require_backup_enabled` for backup endpoints
- Updated health endpoint to show S3 and backup status
- All backup endpoints now return 503 if S3 is not configured

Health Endpoint Response:

```json
{
  "status": "healthy",
  "database": "connected",
  "s3_storage": "disabled",
  "backup_service": "disabled",
  "timestamp": "2025-12-13T...",
  "notes": {
    "s3_storage": "Image uploads will be skipped",
    "backup_service": "Database backups disabled"
  }
}
```

Backup Endpoint Response (when disabled):

```json
{
  "error": "Backup not configured",
  "message": "S3 credentials required for backup operations",
  "hint": "Set AWS_ACCESS_KEY_ID, AWS_SECRET_ACCESS_KEY, and BACKUP_BUCKET"
}
```

---

5. Environment Configuration (`.env.example`)

Before: All AWS credentials listed as required
After: Clearly marked as optional with comments

```
# REQUIRED: Core Configuration
DATABASE_URL=postgresql://...
API_SECRET_KEY=your-secret-key

# OPTIONAL: AWS S3 Configuration
# Leave these blank to deploy without S3
# AWS_ACCESS_KEY_ID=...
# AWS_SECRET_ACCESS_KEY=...
# S3_BUCKET_NAME=...
```

Impact:
- ✅ Clear distinction between required and optional variables
- ✅ Users can deploy without AWS account

---

6. New Deployment Guide (`RENDER_DEPLOYMENT_GUIDE.md`)

Created comprehensive step-by-step guide for deploying to Render without AWS:

Contents:
- Prerequisites (only DATABASE_URL and API_SECRET_KEY)
- Step-by-step Render deployment instructions
- Health check testing
- API endpoint documentation
- Troubleshooting section
- Notes on what works without S3

---

7. Updated Documentation

README.md:
- Added prominent note about S3-free deployment
- Link to RENDER_DEPLOYMENT_GUIDE.md

requirements.txt:
- Added comment indicating boto3 is optional
- Kept boto3 installed (for those who want S3 later)

🎯 What Works Without S3

✅ Fully Functional:

- Database scraping (NBA Stats API, Basketball-Reference)
- Player data collection (stats, biomechanics, career data)
- Database population (Shooter, UserProfile, UserAnalysis tables)
- API endpoints for data retrieval
- Health checks
- Database connection testing

⚠️ Skipped Operations:

- Image uploads to cloud storage
- Profile image URLs (remain NULL)
- S3-based backup operations
- Automatic backup scheduling

🚀 Deployment Steps (Updated)

Minimal Deployment (No AWS):

1. **Configure Environment** (Render):
   `bash`
   ```
   DATABASE_URL=postgresql://user:pass@host:5432/db
   API_SECRET_KEY=your-secret-key
   ```

2. **Deploy to Render**:
   - Build: `pip install -r requirements.txt`
   - Start: `gunicorn app:app --bind 0.0.0.0:$PORT --workers 2 --timeout 120`

3. **Test Health**:
   `bash`
   ```
   curl https://your-app.onrender.com/health
   ```

4. **Trigger Scrape**:
   `bash`
   ```
   curl -X POST https://your-app.onrender.com/api/scrape/nba \
     -H "Authorization: Bearer YOUR_API_SECRET_KEY"
   ```

📊 Configuration Matrix

Feature	Without S3	With S3
Database Scraping	✅ Works	✅ Works
Player Data	✅ Stored	✅ Stored
Image Uploads	⚠️ Skipped	✅ Works
Image URLs	❌ NULL	✅ S3 URLs
Database Backups	❌ Disabled	✅ Works
API Endpoints	✅ Works	✅ Works
Deployment	✅ Simple	⚠️ Complex

🔄 Adding S3 Later

If you want to enable S3 storage after deployment:

1. **Create AWS S3 Bucket**
2. **Generate IAM Credentials** (S3 access)
3. **Add Environment Variables** in Render:
   `bash`
   `  AWS_ACCESS_KEY_ID=...`
   `  AWS_SECRET_ACCESS_KEY=...`
   `  S3_BUCKET_NAME=...`
   `  AWS_REGION=us-east-1`
4. **Restart Service**
5. **Re-run Image Scraping**: `POST /api/images/scrape`

🐛 Troubleshooting

"S3 Not Configured" Warnings

```
# This is expected and normal!
# The scraper will still work and populate the database
```

Backup Endpoints Return 503

```
# Expected behavior when S3 is not configured
# Add AWS credentials to enable backup operations
```

Images Not Showing in Database

```
# Expected: image_url fields will be NULL without S3
# Frontend should use Abacus AI storage for user uploads
```

📝 Notes for Frontend Integration

The frontend (deployed on Abacus AI) should:
- ✅ Use Abacus AI's built-in storage for user uploads
- ✅ Query scraper API for player data only
- ⚠️ Not rely on profile image URLs from database (use CDN or Abacus AI storage)

✅ Verification Checklist

- [x] Storage module conditionally imports S3 functions
- [x] Main pipeline skips S3 upload if disabled
- [x] Backup module conditionally imports backup functions
- [x] API server handles missing S3 gracefully
- [x] Health endpoint shows S3/backup status
- [x] All backup endpoints protected with decorator
- [x] `.env.example` clearly marks AWS as optional
- [x] Deployment guide created for S3-free deployment
- [x] README.md updated with quick deploy note
- [x] requirements.txt commented for boto3

🎉 Result

The Python scraper is now ready to deploy to Render **without AWS credentials**. It will populate the PostgreSQL database with player data while skipping image uploads. The frontend on Abacus AI will handle user-uploaded media using Abacus AI's built-in storage.

Next Steps:
1. ✅ Deploy to Render (using RENDER_DEPLOYMENT_GUIDE.md)
2. ⏳ Test health endpoint
3. ⏳ Trigger initial NBA scrape
4. ⏳ Verify database population
5. ⏳ Connect frontend to scraper API