Python Scraper Deployment Guide (Render - No AWS Required)

This guide walks through deploying the Python scraper service to Render **without AWS S3 credentials**. The scraper will populate the PostgreSQL database with player data, but skip image uploads.

📋 Prerequisites

- Render account (free tier works)
- PostgreSQL database URL from Abacus AI
- API Secret Key for authentication

🚀 Quick Start

Step 1: Environment Variables

The scraper requires **only 2 environment variables** for basic operation:

```
DATABASE_URL=postgresql://user:password@host:5432/basketball_shooting_db
API_SECRET_KEY=your-secret-key-here
```

Optional (leave blank for now):

```
PORT=5000                    # Render sets this automatically
DEBUG=false                  # Production setting
NEXTJS_API_URL=              # Leave blank
```

Step 2: Deploy to Render

1. **Create New Web Service**
   - Go to Render Dashboard (https://dashboard.render.com/)
   - Click “New +” → “Web Service”
   - Connect your GitHub repository

2. **Configure Service**
   - **Name**: `basketball-scraper` (or your choice)
   - **Region**: Choose closest to your database
   - **Branch**: `main`
   - **Root Directory**: `python-scraper`
   - **Runtime**: `Python 3`
   - **Build Command**: `pip install -r requirements.txt`
   - **Start Command**: `gunicorn app:app --bind 0.0.0.0:$PORT --workers 2 --timeout 120`

3. **Add Environment Variables**
   - Click “Environment” tab
   - Add:
     - `DATABASE_URL` → (your Abacus AI PostgreSQL URL)

- `API_SECRET_KEY` → (generate a secure random key)

4. **Deploy**
   - Click “Create Web Service”
   - Wait for deployment (2-3 minutes)

Step 3: Test the Service

Once deployed, test the health endpoint:

```
curl https://your-app.onrender.com/health
```

Expected response:

```
{
  "status": "healthy",
  "database": "connected",
  "s3_storage": "disabled",
  "timestamp": "2025-12-13T..."
}
```

Step 4: Trigger Data Scraping

Trigger the NBA scraper to populate your database:

```
curl -X POST https://your-app.onrender.com/api/scrape/nba \
  -H "Authorization: Bearer YOUR_API_SECRET_KEY" \
  -H "Content-Type: application/json"
```

Note: The scraper will:
- ✅ Collect NBA player data (stats, biomechanics)
- ✅ Insert data into PostgreSQL database
- ⚠️ Skip image uploads (S3 not configured)
- ⚠️ Skip backup operations (S3 not configured)

📊 Available Endpoints

Health Check

```
GET /health
```

Scraping Operations

```
POST /api/scrape/nba              # Scrape current NBA players
POST /api/scrape/historical       # Scrape historical players
POST /api/scrape/full             # Full scrape (all sources)
```

Data Retrieval

```
GET /api/shooters                # Get all shooters (paginated)
GET /api/shooters/:id            # Get specific shooter
GET /api/stats                   # Database statistics
```

Database Management

```
GET /api/db/test                 # Test database connection
GET /api/db/stats                # Database statistics
```

🔒 Authentication

All scraping and management endpoints require authentication:

```
Authorization: Bearer YOUR_API_SECRET_KEY
```

⚙️ Configuration Options

Minimal Configuration (No S3)

```
DATABASE_URL=postgresql://...
API_SECRET_KEY=your-secret-key
```

Full Configuration (With S3 - Optional)

```
DATABASE_URL=postgresql://...
API_SECRET_KEY=your-secret-key
AWS_ACCESS_KEY_ID=your-aws-key
AWS_SECRET_ACCESS_KEY=your-aws-secret
S3_BUCKET_NAME=basketball-shooters-db
AWS_REGION=us-east-1
```

🎯 What Happens Without S3?

When S3 credentials are **not configured**:

✅ **Still Works**:
- Database scraping and population
- Player statistics collection
- Biomechanical data gathering
- API endpoints for data retrieval
- Database management operations

⚠️ **Skipped Operations**:
- Image uploads to cloud storage
- Profile image URLs (will be NULL)
- Image backups
- S3-based backup/restore operations

🔄 Adding S3 Later (Optional)

If you want to enable image storage later:

1. Create an AWS S3 bucket
2. Generate IAM credentials with S3 access
3. Add environment variables to Render:
   `bash`
   `AWS_ACCESS_KEY_ID=...`
   `AWS_SECRET_ACCESS_KEY=...`
   `S3_BUCKET_NAME=...`
   `AWS_REGION=us-east-1`
4. Restart the service
5. Re-run image scraping: `POST /api/images/scrape`

🐛 Troubleshooting

Database Connection Fails

```
# Check DATABASE_URL format
postgresql://username:password@host:port/database

# Test connection
curl https://your-app.onrender.com/api/db/test
```

Scraper Times Out

```
# Increase timeout in Render settings
# Or use async scraping endpoint:
POST /api/scrape/nba?async=true
```

“S3 Not Configured” Warnings

```
# This is expected! The scraper will still work.
# Images are skipped, but data is collected.
```

📝 Notes

1. **Free Tier Limitations**:
   - Render free tier sleeps after 15 minutes of inactivity
   - First request after sleep takes ~30 seconds to wake up
   - Consider upgrading for production use

2. **Database Requirements**:
   - PostgreSQL 12+
   - ~100MB storage for 100-200 players (without images)
   - Connection pooling recommended

3. **Scraping Frequency**:
   - NBA data updates: Daily

- Historical data: One-time setup
- Recommended: Weekly scrapes for active players

🎉 Success!

Your scraper is now deployed and ready to populate your database. The frontend (Abacus AI deployment) will use Abacus AI's built-in storage for user uploads instead of S3.

Next Steps:
1. ✅ Deploy Python scraper to Render (you are here)
2. ⏳ Trigger initial data scrape
3. ⏳ Connect frontend to scraper API
4. ⏳ Test end-to-end flow