IMDb Movie Data APIs

Comprehensive Developer Documentation

A Complete Guide to Fetching Movie Details via OMDb, TMDb, and the Official IMDb GraphQL API

Document Version	2.2
Date	March 27, 2026
Classification	Developer Reference
Audience	Software Engineers, Product Managers, Tech Leads
APIs Covered	OMDb API • TMDb API • IMDb Official GraphQL API

Table of Contents

1 Introduction & Landscape Overview

1.1 Why This Document Exists

IMDb (Internet Movie Database) is the world's most authoritative source of movie, television, and entertainment metadata, housing information on over 10 million titles, 14 million cast and crew records, and 1.6 billion user ratings. However, unlike many modern platforms, IMDb does not offer a straightforward public REST API for general developer use. Instead, the ecosystem is fragmented across several official and unofficial pathways.

This document provides a single, comprehensive reference for application developers who need to programmatically fetch movie details, ratings, cast information, posters, and related metadata. It covers every viable option as of March 2026, with fully worked code examples, response schemas, authentication flows, rate-limit details, and architecture recommendations.

1.2 IMDb API Landscape in 2026

The current landscape for accessing IMDb-sourced or IMDb-equivalent movie data consists of four primary channels, each with distinct trade-offs in terms of data completeness, cost, ease of integration, and licensing terms:

• **OMDb API** — The Open Movie Database. A free/low-cost RESTful API that mirrors much of IMDb's core data. Requires an API key (free tier available). Most popular choice for hobby and small-scale projects.

• **TMDb API** — The Movie Database. A community-driven, richly featured API with extensive image assets, translations, and discovery features. Free for non-commercial use. Excellent documentation.

• **Official IMDb GraphQL API** — Powered by AWS Data Exchange. The only truly official channel for licensed IMDb data. Provides GraphQL queries for canonical IMDb metadata. Enterprise-grade pricing.

• **IMDb Non-Commercial Datasets** — Bulk TSV files released by IMDb for personal/educational use. Updated daily. Requires self-hosted ingestion pipeline.

1.3 Quick Comparison Matrix

Feature	OMDb API	TMDb API	IMDb GraphQL	IMDb Datasets	Third-Party Scrapers
`Auth Method`	API Key	API Key / Bearer Token	AWS IAM + API Key	None (public)	Varies
`Protocol`	REST (JSON/XML)	REST (JSON)	GraphQL	TSV Bulk Files	REST (varies)
`Free Tier`	1,000 req/day	~40 req/10s	Free Trial only	Yes (non-commercial)	Limited
`IMDb Ratings`	Yes	No (own ratings)	Yes (canonical)	Yes	Varies

Feature	OMDb API	TMDb API	IMDb GraphQL	IMDb Datasets	Third-Party Scrapers
`Poster Images`	Patron only	Yes (CDN)	No	No	Varies
`Real-Time`	Yes	Yes	Yes	Daily batch	Varies
`Commercial OK`	Yes (paid)	License needed	Yes (paid)	No	Risk of TOS violation
`Data Quality`	Good	Excellent	Authoritative	Authoritative	Variable

Table 1: Comparison of available movie data API options (as of March 2026)

2 OMDb API — The Open Movie Database

2.1 Overview & Authentication

The OMDb API is a RESTful web service that provides movie information sourced from a combination of IMDb data and community contributions. It is the most popular unofficial alternative for developers who need quick access to movie metadata without enterprise-level licensing.

■ **Base URL:** `https://www.omdbapi.com/` | **Authentication:** API key passed as query parameter `&apikey=YOUR_KEY`

To obtain an API key, visit https://www.omdbapi.com/apikey.aspx and submit your email address. A free key (1,000 requests/day) is emailed immediately. Paid tiers are available via Patreon for higher limits and poster access.

2.2 Endpoints & Parameters

The OMDb API exposes a single endpoint that accepts different parameter combinations to perform either a **lookup by ID/title** or a **search by keyword**. All requests are HTTP GET.

By ID or Title (Detail Lookup)

Parameter	Required	Description	Example
`i`	Yes*	A valid IMDb ID (e.g., tt0111161)	i=tt0111161
`t`	Yes*	Movie title to search for (returns first match)	t=The+Shawshank+Redemption
`type`	No	Type of result: movie, series, episode	type=movie
`y`	No	Year of release	y=1994
`plot`	No	short (default) or full plot text	plot=full
`r`	No	Response format: json (default) or xml	r=json
`callback`	No	JSONP callback name	callback=myFunc
`v`	No	API version (reserved for future)	v=1

** Either 'i' or 't' is required. If both provided, 'i' takes priority.*

By Search (Keyword Search)

Parameter	Required	Description	Example
`s`	Yes	Movie title keyword(s) to search for	s=Batman
`type`	No	Type of result: movie, series, episode	type=movie

Parameter	Required	Description	Example
`y`	No	Year of release	y=2022
`r`	No	Response format: json (default) or xml	r=json
`page`	No	Page number of results (1-100, default=1)	page=2

Table 2–3: OMDb API request parameters for detail lookup and search

2.3 Response Schema & Fields

A successful detail lookup returns a JSON object with the following fields. All values are returned as strings. The `Response` field indicates success (`"True"`) or failure (`"False"`).

Field	Type	Description	Example Value
`Title`	String	Full movie title	The Shawshank Redemption
`Year`	String	Release year or range for series	1994
`Rated`	String	Content rating (MPAA)	R
`Released`	String	Full release date	14 Oct 1994
`Runtime`	String	Duration in minutes	142 min
`Genre`	String	Comma-separated genre list	Drama
`Director`	String	Director name(s)	Frank Darabont
`Writer`	String	Writer name(s)	Stephen King, Frank Darabont
`Actors`	String	Lead cast (comma-separated)	Tim Robbins, Morgan Freeman...
`Plot`	String	Short or full plot synopsis	Two imprisoned men bond...
`Language`	String	Language(s)	English
`Country`	String	Country/countries of origin	United States
`Awards`	String	Awards summary text	Nominated for 7 Oscars...
`Poster`	String	URL to poster image	https://m.media-amazon...
`Ratings`	Array	Array of {Source, Value} objects	See below
`Metascore`	String	Metacritic score (0-100)	82
`imdbRating`	String	IMDb rating (0.0-10.0)	9.3
`imdbVotes`	String	Total IMDb votes (comma-formatted)	2,891,892
`imdbID`	String	Unique IMDb identifier	tt0111161
`Type`	String	movie, series, or episode	movie

Field	Type	Description	Example Value
`DVD`	String	DVD release date	21 Dec 1999
`BoxOffice`	String	US box office gross	$28,767,189
`Production`	String	Production company	N/A
`Website`	String	Official website URL	N/A
`Response`	String	True if result found	True

Table 4: Complete OMDb API response fields for a detail lookup

Ratings Array Structure:

```
"Ratings": [
  {"Source": "Internet Movie Database", "Value": "9.3/10"},
  {"Source": "Rotten Tomatoes",         "Value": "91%"},
  {"Source": "Metacritic",             "Value": "82/100"}
]
```

2.4 Search Endpoint Deep Dive

The search endpoint (`?s=keyword`) returns a paginated list of matching titles. Each page returns up to 10 results. The response includes a `totalResults` field for pagination logic.

Search Response Schema:

```
{
  "Search": [
    {
      "Title":  "Batman Begins",
      "Year":   "2005",
      "imdbID": "tt0372784",
      "Type":   "movie",
      "Poster": "https://m.media-amazon.com/images/M/...jpg"
    },
    { ... }
  ],
  "totalResults": "564",
  "Response": "True"
}
```

■ **Tip:** Search results are limited in fields. Use the returned `imdbID` from search results to make a follow-up detail lookup (`?i=tt0372784`) for complete metadata.

2.5 Code Examples

Python (requests library):

```
import requests

API_KEY = "your_api_key_here"
BASE_URL = "https://www.omdbapi.com/"

# --- Lookup by IMDb ID ---
params = {"i": "tt0111161", "apikey": API_KEY, "plot": "full"}
response = requests.get(BASE_URL, params=params)
movie = response.json()

print(f"Title:  {movie['Title']}")
print(f"Year:   {movie['Year']}")
print(f"Rating: {movie['imdbRating']}")
print(f"Genre:  {movie['Genre']}")
print(f"Plot:   {movie['Plot'][:100]}...")

# --- Search by keyword ---
search_params = {"s": "Inception", "type": "movie", "apikey": API_KEY}
results = requests.get(BASE_URL, params=search_params).json()

if results["Response"] == "True":
    for item in results["Search"]:
        print(f"{item['Title']} ({item['Year']}) - {item['imdbID']}")
```

JavaScript (Fetch API):

```
const API_KEY = "your_api_key_here";
const BASE_URL = "https://www.omdbapi.com/";

// Lookup by title
async function getMovie(title) {
  const url = `${BASE_URL}?t=${encodeURIComponent(title)}&apikey=${API_KEY}`;
  const res = await fetch(url);
  const data = await res.json();

  if (data.Response === "True") {
    console.log(`${data.Title} (${data.Year}) — ${data.imdbRating}/10`);
    console.log(`Genre: ${data.Genre}`);
    console.log(`Director: ${data.Director}`);
  } else {
    console.error(`Error: ${data.Error}`);
  }
}

getMovie("Inception");
```

cURL:

```
# Lookup by IMDb ID
curl "https://www.omdbapi.com/?i=tt0111161&apikey=YOUR_KEY&plot=full"

# Search by keyword
curl "https://www.omdbapi.com/?s=Batman&type=movie&page=1&apikey=YOUR_KEY"
```

2.6 Error Handling

OMDb always returns HTTP 200, even on errors. Check the `Response` field in the JSON body. When `Response` is `"False"`, an `Error` field will contain the reason.

Error Message	Cause	Resolution
`Movie not found!`	No title matches query	Verify title/ID spelling
`Incorrect IMDb ID.`	Malformed IMDb ID format	Use format tt + 7-8 digits
`Invalid API key!`	Key missing, expired, or wrong	Re-register at omdbapi.com
`Request limit reached!`	Exceeded daily quota	Upgrade plan or wait 24h
`Too many results.`	Search query too broad	Add year or type filters

2.7 Rate Limits & Pricing Tiers

Tier	Daily Limit	Poster Access	Cost
`Free`	1,000 requests/day	No (low-res only)	$0 / month
`Patron (Basic)`	100,000 requests/day	Yes (up to 2000x3000)	$1 / month (Patreon)
`Patron (Premium)`	Unlimited*	Yes (full archive)	$5 / month (Patreon)

** Subject to reasonable use policy. Pricing as of March 2026.*

3 TMDb API — The Movie Database

3.1 Overview & Authentication

TMDb (The Movie Database) is a community-driven entertainment database launched in 2008. It provides one of the most comprehensive and well-documented APIs in the entertainment data space, with support for over 100 languages, extensive image assets via CDN, and a rich set of discovery and filtering endpoints. TMDb is free for non-commercial use with attribution.

■ **Base URL:** `https://api.themoviedb.org/3/` | **Auth:** API Key (query param) or Bearer Token (header)
Registration: https://www.themoviedb.org/settings/api

TMDb supports two authentication methods: (1) passing `api_key` as a query parameter, or (2) passing a Bearer access token in the `Authorization` header. The Bearer token method (API Read Access Token) is the recommended modern approach.

```
# Method 1: API Key as query parameter
GET https://api.themoviedb.org/3/movie/550?api_key=YOUR_API_KEY

# Method 2: Bearer Token (recommended)
GET https://api.themoviedb.org/3/movie/550
Authorization: Bearer YOUR_ACCESS_TOKEN
```

3.2 Core Movie Endpoints

Method	Endpoint	Description
GET	/movie/{movie_id}	Get top-level details of a movie by TMDb ID
GET	/movie/{movie_id}/credits	Get cast and crew for a movie
GET	/movie/{movie_id}/images	Get poster, backdrop, and logo images
GET	/movie/{movie_id}/videos	Get trailers, teasers, clips
GET	/movie/{movie_id}/reviews	Get user reviews
GET	/movie/{movie_id}/similar	Get similar movies
GET	/movie/{movie_id}/recommendations	Get recommended movies
GET	/movie/{movie_id}/keywords	Get keywords associated with movie
GET	/movie/{movie_id}/release_dates	Get release dates and certifications by country
GET	/movie/{movie_id}/external_ids	Get IMDb ID, Facebook, Twitter, Instagram IDs
GET	/movie/{movie_id}/watch/providers	Get streaming/buy/rent availability by region

Method	Endpoint	Description
GET	/movie/popular	Get current popular movies
GET	/movie/top_rated	Get top-rated movies of all time
GET	/movie/now_playing	Get movies currently in theaters
GET	/movie/upcoming	Get upcoming movie releases

Table 5: TMDb core movie API endpoints

3.3 Search & Discover

TMDb provides two distinct mechanisms for finding movies: **Search** (text-based keyword matching) and **Discover** (filter-based exploration with 30+ parameters).

Search Endpoint:

```
GET /search/movie?query=Inception&language=en-US&page=1&include_adult=false

# Optional parameters:
#   query          (required) — Text query to search
#   language       — ISO 639-1 language code (default: en-US)
#   page           — Page number (1-500)
#   include_adult  — Include adult content (default: false)
#   region         — ISO 3166-1 country code to filter release dates
#   year           — Filter by exact release year
#   primary_release_year — Filter by primary release year
```

Discover Endpoint (selected parameters):

Parameter	Type	Description	Example
sort_by	string	Sort results by field	popularity.desc
with_genres	string	Filter by genre IDs (comma/pipe separated)	28,12
primary_release_date.gte	date	Movies released on or after date	2024-01-01
primary_release_date.lte	date	Movies released on or before date	2024-12-31
vote_average.gte	number	Minimum vote average	7.0
vote_count.gte	number	Minimum number of votes	100
with_original_language	string	ISO 639-1 language code	en
with_runtime.gte	integer	Minimum runtime in minutes	90
with_watch_providers	string	Filter by streaming provider ID	8 (Netflix)

Parameter	Type	Description	Example
watch_region	string	Country for watch providers	US

3.4 Images & Configuration

TMDb serves images via a CDN. To construct a full image URL, you must first call the `/configuration` endpoint (cached) to get the base URL and available sizes, then concatenate: `base_url + size + file_path`.

```
# Step 1: Get configuration (cache this — it rarely changes)
GET /configuration

# Response (partial):
{
  "images": {
    "base_url": "http://image.tmdb.org/t/p/",
    "secure_base_url": "https://image.tmdb.org/t/p/",
    "poster_sizes": ["w92","w154","w185","w342","w500","w780","original"],
    "backdrop_sizes": ["w300","w780","w1280","original"]
  }
}

# Step 2: Construct image URL
# https://image.tmdb.org/t/p/w500/9O7gLzmreU0nGkIB6K3BsJbzvNv.jpg
```

3.5 Code Examples

Python (requests + Bearer Token):

```
import requests

ACCESS_TOKEN = "your_read_access_token"
BASE = "https://api.themoviedb.org/3"
HEADERS = {
    "Authorization": f"Bearer {ACCESS_TOKEN}",
    "Accept": "application/json"
}

# Get movie details
movie_id = 27205  # Inception
resp = requests.get(f"{BASE}/movie/{movie_id}", headers=HEADERS)
movie = resp.json()

print(f"Title:    {movie['title']}")
print(f"Tagline:  {movie['tagline']}")
print(f"Rating:   {movie['vote_average']}/10 ({movie['vote_count']} votes)")
print(f"Budget:   ${movie['budget']:,}")
print(f"Revenue:  ${movie['revenue']:,}")
print(f"Genres:   {[g["name"] for g in movie["genres"]]}")

# Search for movies
search = requests.get(f"{BASE}/search/movie",
    headers=HEADERS,
    params={"query": "Dark Knight", "year": 2008}
).json()

for m in search["results"][:5]:
    print(f"  {m['title']} ({m['release_date'][:4]}) - ID: {m['id']}")
```

JavaScript (Fetch + Bearer Token):

```
const TOKEN = "your_read_access_token";
const BASE = "https://api.themoviedb.org/3";
const headers = {
  Authorization: `Bearer ${TOKEN}`,
  "Content-Type": "application/json"
};

async function getMovieDetails(movieId) {
  const res = await fetch(
    `${BASE}/movie/${movieId}?append_to_response=credits,videos,images`,
    { headers }
  );
  const data = await res.json();
  console.log(data.title, data.vote_average);
  console.log("Director:", data.credits.crew
    .find(c => c.job === "Director")?.name);
  return data;
}

getMovieDetails(27205);
```

3.6 Append-to-Response Pattern

One of TMDb's most powerful features is the `append_to_response` query parameter, which allows you to fetch multiple related resources in a single API call without incurring additional rate-limit hits. This dramatically reduces the number of requests needed.

```
# Fetch movie details + credits + videos + images + reviews in ONE call
GET /movie/27205?append_to_response=credits,videos,images,reviews,keywords

# The response will contain all standard movie fields PLUS:
# - movie.credits.cast[]  and  movie.credits.crew[]
# - movie.videos.results[]
# - movie.images.posters[]  and  movie.images.backdrops[]
# - movie.reviews.results[]
# - movie.keywords.keywords[]
```

✓ **Performance Tip:** Using append_to_response counts as a single request against your rate limit, regardless of how many sub-resources you include. Always use this for detail pages in your app.

3.7 Rate Limits & Pricing

TMDb enforces a rate limit of approximately 40 requests per 10-second window. The API is free for non-commercial use with proper attribution. Commercial use requires a separate license. Contact sales@themoviedb.org for commercial pricing.

Use Case	Cost	Rate Limit	Attribution Required
`Non-Commercial / Developer`	Free	~40 req / 10 seconds	Yes — TMDb logo + link
`Commercial License`	Contact Sales	Custom	Yes — per agreement

4 Official IMDb GraphQL API (via AWS)

4.1 Overview & Provisioning

In 2023, IMDb launched its official API, powered by AWS Data Exchange, providing a GraphQL-based interface to canonical IMDb metadata. This is the only authorized way to access IMDb's proprietary data (including the official 1-10 star ratings) for commercial purposes. The API is delivered via a single GraphQL endpoint, enabling developers to request exactly the fields they need.

■ **Access:** Subscribe via the AWS Marketplace at `https://aws.amazon.com/marketplace/pp/prodview-...` (search 'IMDb')
Protocol: GraphQL over HTTPS | **Auth:** AWS IAM Signature + API Key

Key Benefits of the Official API:

• JSON payloads with a simplified, consistent data structure

• Single URL for all queries — no URL parameter management

• Request only the fields you need (GraphQL efficiency)

• Multiple entities in one request (batch title/name lookups)

• Hands-on support from IMDb's technical staff during integration

• Daily-updated canonical data from the world's largest entertainment database

4.2 GraphQL Query Structure

The IMDb GraphQL API uses a single POST endpoint. Queries are sent in the request body. Here is the general structure:

```
POST https://api.graphql.imdb.com/
Content-Type: application/json
x-amzn-marketplace-token: YOUR_MARKETPLACE_TOKEN

{
  "query": "query GetTitle($id: ID!) {\n    title(id: $id) {\n      titleText { text }\n releaseYear { year }\n      ratingsSummary { aggregateRating voteCount }\n      genres { genres { text } }\n      plot { plotText { plainText } }\n      principalCredits { credits { name { nameText { text } } } }\n    }\n  }",
  "variables": { "id": "tt0111161" }
}
```

4.3 Sample Queries

Get Title Details:

```
query {
  title(id: "tt0468569") {
    titleText { text }
    originalTitleText { text }
    releaseYear { year endYear }
    titleType { id text }
    ratingsSummary {
      aggregateRating
      voteCount
    }
    runtime { seconds displayableProperty { value { plainText } } }
    genres { genres { id text } }
    plot { plotText { plainText } }
    primaryImage { url width height }
  }
}
```

Search by Keyword:

```
query {
  mainSearch(first: 5, options: {
    searchTerm: "Inception"
    type: TITLE
  }) {
    edges {
      node {
        entity {
          ... on Title {
            id
            titleText { text }
            releaseYear { year }
            ratingsSummary { aggregateRating }
          }
        }
      }
    }
  }
}
```

4.4 Pricing & Licensing

The official IMDb API is an enterprise product with pricing through the AWS Marketplace. A free trial is available for evaluation. Production licensing is negotiated based on use case, volume, and data products selected. Contact developer.imdb.com for current pricing.

■ **Important:** The official API data is subject to IMDb's licensing terms. Redistribution, caching beyond permitted windows, and reselling of data are strictly prohibited without explicit written agreement.

5 Setup Walkthrough — Getting API Keys

This section provides step-by-step instructions for obtaining authentication credentials for each of the three primary movie data APIs. Follow the relevant subsection(s) based on which API you plan to integrate.

5.1 OMDb API Key Registration

OMDb uses a simple API key model. Keys are free for basic usage and can be obtained in under a minute.

Step 1	Navigate to https://www.omdbapi.com/apikey.aspx
Step 2	Select your account type — choose **FREE! (1,000 daily limit)** for testing, or a Patron tier for production
Step 3	Enter your email address and a brief description of your project
Step 4	Click **Submit**. Your API key will be sent to your email within seconds
Step 5	Click the **activation link** in the email to activate your key
Step 6	Test your key immediately by visiting: `https://www.omdbapi.com/?apikey=YOUR_KEY&t=Inception`

■ **Tip:** Free keys are limited to 1,000 requests/day. For production apps, consider the Patreon tiers ($1-$5/month) which provide up to unlimited requests and high-resolution poster access.

5.2 TMDb API Key & Bearer Token

TMDb provides two forms of authentication. The newer Bearer Token (API Read Access Token) is recommended for all new integrations.

Step 1	Create a free account at https://www.themoviedb.org/signup
Step 2	After email verification, log in and navigate to **Settings** (click your avatar → Settings)
Step 3	In the left sidebar, click **API**
Step 4	Click **Create** or **Request an API Key**. Select **Developer** for non-commercial use
Step 5	Fill in the application form — provide your app name, URL, and a brief description of usage
Step 6	Upon approval (usually instant), you will see two credentials on the API page: • **API Key (v3 auth)** — a 32-character hexadecimal string • **API Read Access Token (v4 auth)** — a long JWT-format Bearer token
Step 7	Test with cURL: `curl --request GET 'https://api.themoviedb.org/3/movie/550' \ --header 'Authorization: Bearer YOUR_TOKEN'`

5.3 AWS Account & IAM Setup for IMDb GraphQL API

The official IMDb API is delivered through AWS Data Exchange and requires an AWS account with properly configured IAM (Identity and Access Management) credentials. This is the most involved setup of the three APIs, but it provides access to canonical IMDb data.

Step 1 — Create an AWS Account

If you do not already have an AWS account, sign up at https://aws.amazon.com/. You will need a valid credit card, phone number for verification, and an email address. The account itself is free — you only pay for services you use.

Step 2 — Create a Dedicated IAM User

■ **Important:** Never use your AWS root account credentials for API access. Always create a dedicated IAM user with the minimum permissions required. This follows the principle of least privilege and limits blast radius if credentials are compromised.

1. Sign in to the **AWS Management Console** at https://console.aws.amazon.com/
2. In the top search bar, type **IAM** and click on the **IAM** service
3. In the left navigation panel, click **Users** → then click **Create user**
4. Enter a descriptive username, e.g., `imdb-api-user`
5. On the **Set permissions** page, select **Attach policies directly**
6. Search for and attach the policy: `AWSDataExchangeSubscriberFullAccess`
7. Review the configuration summary and click **Create user**

5.4 Generating AWS Access Keys

Once your IAM user is created, you need to generate an Access Key pair (Access Key ID + Secret Access Key) for programmatic API access.

1. In the IAM console, click on the user you just created (`imdb-api-user`)
2. Navigate to the **Security credentials** tab
3. Scroll down to the **Access keys** section
4. Click **Create access key**
5. Select the use case: choose **Application running outside AWS** (or **Third-party service**)
6. Optionally add a description tag (e.g., "IMDb API integration")
7. Click **Create access key**

AWS will now display your two credentials:

Credential	Format	Example	Notes
`Access Key ID`	20-char alphanumeric	AKIAIOSFODNN7EXAMPLE	Starts with AKIA... (for long-lived keys)
`Secret Access Key`	40-char base64 string	wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY	Shown ONCE — save immediately

✗ **Critical:** The Secret Access Key is displayed only once at creation time. If you lose it, you must delete the access key and create a new pair. Copy both values immediately and store them securely.

5.5 Subscribing to IMDb on AWS Marketplace

After creating your IAM user and access keys, you must subscribe to the IMDb data product on AWS Marketplace to gain access to the GraphQL API endpoint.

1. Navigate to **AWS Marketplace** at https://aws.amazon.com/marketplace/
2. Search for **"IMDb"** in the marketplace search bar
3. Select the **IMDb** data product (published by IMDb.com, Inc.)
4. Review the product description, pricing tiers, and data schema documentation
5. Click **Continue to Subscribe**
6. Accept the offer terms and conditions (IMDb licensing agreement)
7. Click **Subscribe** — a free trial period is typically available for evaluation
8. Once subscribed, you will receive access to the GraphQL endpoint URL and any additional marketplace tokens required for authentication

5.6 Configuring Credentials Locally

There are several ways to make your AWS credentials available to your application. Here are the most common methods, ordered from simplest to most production-ready:

Method A — Environment Variables (Recommended for Development)

```
# Linux / macOS — add to ~/.bashrc, ~/.zshrc, or ~/.profile
export AWS_ACCESS_KEY_ID="AKIAIOSFODNN7EXAMPLE"
export AWS_SECRET_ACCESS_KEY="wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY"
export AWS_DEFAULT_REGION="us-east-1"

# Windows (PowerShell)
$env:AWS_ACCESS_KEY_ID = "AKIAIOSFODNN7EXAMPLE"
$env:AWS_SECRET_ACCESS_KEY = "wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY"
$env:AWS_DEFAULT_REGION = "us-east-1"

# Windows (Command Prompt)
set AWS_ACCESS_KEY_ID=AKIAIOSFODNN7EXAMPLE
set AWS_SECRET_ACCESS_KEY=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
set AWS_DEFAULT_REGION=us-east-1
```

Method B — AWS CLI Configuration (Recommended)

```
# Install AWS CLI (if not already installed)
pip install awscli

# Run the interactive configuration wizard
aws configure

# You will be prompted to enter:
#   AWS Access Key ID [None]: AKIAIOSFODNN7EXAMPLE
#   AWS Secret Access Key [None]: wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
#   Default region name [None]: us-east-1
#   Default output format [None]: json

# This creates two files:
#   ~/.aws/credentials  — stores your access keys
#   ~/.aws/config       — stores region and output preferences

# Verify the configuration
aws sts get-caller-identity
```

Method C — Named Profiles (Multiple AWS Accounts)

```
# Create a named profile for IMDb work
aws configure --profile imdb

# Use the profile in your application
export AWS_PROFILE=imdb

# Or in Python (boto3)
import boto3
session = boto3.Session(profile_name="imdb")
client = session.client("dataexchange", region_name="us-east-1")
```

Method D — .env File (Application Development)

```
# Create a .env file in your project root (add to .gitignore!)
# File: .env
AWS_ACCESS_KEY_ID=AKIAIOSFODNN7EXAMPLE
AWS_SECRET_ACCESS_KEY=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
AWS_DEFAULT_REGION=us-east-1
OMDB_API_KEY=your_omdb_key
TMDB_BEARER_TOKEN=your_tmdb_bearer_token

# Python: Load with python-dotenv
from dotenv import load_dotenv
import os

load_dotenv()
aws_key = os.getenv("AWS_ACCESS_KEY_ID")
aws_secret = os.getenv("AWS_SECRET_ACCESS_KEY")
omdb_key = os.getenv("OMDB_API_KEY")
tmdb_token = os.getenv("TMDB_BEARER_TOKEN")
```

Method E — Signing Requests with boto3 (Python)

For calling the IMDb GraphQL API, you need to sign requests using AWS Signature Version 4. The `boto3` and `requests-aws4auth` libraries handle this automatically:

```
import boto3
import requests
from requests_aws4auth import AWS4Auth

# Create AWS auth signer
session = boto3.Session()
credentials = session.get_credentials().get_frozen_credentials()

auth = AWS4Auth(
    credentials.access_key,
    credentials.secret_key,
    "us-east-1",
    "execute-api",
    session_token=credentials.token  # For temp credentials
)

# Make signed request to IMDb GraphQL API
query = """
  query { title(id: "tt0111161") {
    titleText { text }
    ratingsSummary { aggregateRating voteCount }
  }}
"""

response = requests.post(
    "https://api.graphql.imdb.com/",
    json={"query": query},
    auth=auth,
    headers={"Content-Type": "application/json"}
)

print(response.json())
```

5.7 Security Best Practices for Key Management

API keys and AWS credentials are sensitive secrets. A leaked key can result in unauthorized access, unexpected charges, and data exposure. Follow these practices rigorously:

Practice	Why It Matters	Implementation
`Never hardcode keys in source code`	Keys in code end up in Git history, CI logs, and error messages	Use environment variables, .env files, or a secrets manager
`Add .env to .gitignore`	Prevents accidental commit of credentials to repositories	echo '.env' >> .gitignore (do this before first commit)
`Use IAM Roles on AWS infra`	Eliminates need for long-lived access keys entirely	Attach roles to EC2 instances, Lambda functions, ECS tasks
`Rotate keys regularly`	Limits damage window if a key is compromised	AWS IAM → Security credentials → Create new key → Delete old key
`Set up billing alerts`	Catches abuse from leaked AWS credentials early	AWS Budgets → Create budget → Set threshold alerts
`Use separate keys per environment`	Isolates dev/staging/prod to prevent cross-contamination	Create separate IAM users or profiles for each environment
`Use AWS Secrets Manager for production`	Centralized, auditable, auto-rotating secret storage	Store keys in Secrets Manager; fetch at runtime via SDK
`Enable MFA on your AWS root account`	Prevents full account takeover even if password is leaked	IAM → Dashboard → Activate MFA on root account

Table: Security best practices for API key and credential management

✓ Quick Credential Checklist Before Going to Production:
✓ No keys in source code or Git ✓ .env in .gitignore ✓ IAM user has minimum required permissions ✓ MFA enabled on AWS root
✓ Billing alerts configured ✓ Keys rotated within last 90 days ✓ Separate keys for dev/staging/prod

6 IMDb Datasets (Non-Commercial)

6.1 Available TSV Files

IMDb publishes a set of compressed TSV (tab-separated value) files daily for personal and non-commercial use. These are available at `https://datasets.imdbws.com/`. Each file is gzipped and can range from a few MB to several GB.

File	Description	Key Fields
`title.basics.tsv.gz`	Basic title information for all titles	tconst, titleType, primaryTitle, startYear, genres
`title.ratings.tsv.gz`	IMDb ratings and vote counts	tconst, averageRating, numVotes
`title.crew.tsv.gz`	Director and writer information	tconst, directors, writers
`title.principals.tsv.gz`	Principal cast/crew for titles	tconst, nconst, category, characters
`title.akas.tsv.gz`	Alternative titles by region/language	titleId, title, region, language, isOriginalTitle
`title.episode.tsv.gz`	TV episode to parent series mapping	tconst, parentTconst, seasonNumber, episodeNumber
`name.basics.tsv.gz`	Person information	nconst, primaryName, birthYear, primaryProfession, knownForTitles

Table 6: IMDb dataset files and their contents

6.2 Schema & Data Dictionary

All dataset files use a tab character as the delimiter. Missing or null values are represented by `\N`. Multi-value fields (like genres) use comma-separated values within a single column. The `tconst` field is the unique IMDb title identifier (e.g., tt0111161) and `nconst` is the person identifier (e.g., nm0000209).

6.3 Ingestion Pipeline Example (Python + Pandas)

```
import pandas as pd

# Download and load title basics
titles = pd.read_csv(
    "https://datasets.imdbws.com/title.basics.tsv.gz",
    sep="\t", na_values="\\N", low_memory=False,
    dtype={"startYear": "Int64", "endYear": "Int64", "runtimeMinutes": "Int64"}
)

# Load ratings
ratings = pd.read_csv(
    "https://datasets.imdbws.com/title.ratings.tsv.gz",
    sep="\t", na_values="\\N"
)

# Merge titles with ratings
movies = titles[titles["titleType"] == "movie"].merge(ratings, on="tconst")

# Filter: movies with 10,000+ votes and rating >= 7.5
top_movies = movies[
    (movies["numVotes"] >= 10000) & (movies["averageRating"] >= 7.5)
].sort_values("averageRating", ascending=False)

print(f"Found {len(top_movies)} highly-rated movies")
print(top_movies[["primaryTitle","startYear","averageRating","numVotes"]].head(20))
```

■ **License Restriction:** IMDb datasets are for personal and non-commercial use ONLY. Any commercial application requires a separate license through developer.imdb.com.

7 Architecture Patterns & Best Practices

7.1 Caching Strategy

Movie metadata changes infrequently, making aggressive caching highly effective. A recommended caching strategy uses multiple layers to minimize API calls and reduce latency:

• **L1 — In-Memory Cache (e.g., Redis):** Cache individual movie lookups with a TTL of 24 hours. Use the IMDb ID or TMDb ID as the cache key. This handles the vast majority of repeated requests.

• **L2 — Application Database:** Persist frequently accessed movie data in your own database. Refresh on a schedule (daily or weekly) to keep ratings and vote counts current.

• **L3 — CDN for Images:** Cache poster and backdrop images on your own CDN or S3 bucket to avoid repeated image fetches. TMDb image URLs are stable once fetched.

```
# Python: Simple Redis caching example
import redis, json, requests

r = redis.Redis(host="localhost", port=6379, db=0)
TTL = 86400  # 24 hours

def get_movie(imdb_id: str) -> dict:
    cache_key = f"movie:{imdb_id}"
    cached = r.get(cache_key)
    if cached:
        return json.loads(cached)

    # Cache miss — call OMDb API
    resp = requests.get("https://www.omdbapi.com/",
        params={"i": imdb_id, "apikey": API_KEY, "plot": "full"}
    ).json()

    if resp.get("Response") == "True":
        r.setex(cache_key, TTL, json.dumps(resp))
    return resp
```

7.2 Fallback & Multi-Source Architecture

For production applications, a multi-source architecture provides the best combination of data richness and reliability. The recommended pattern is to use TMDb as the primary source for its rich metadata and images, with OMDb as a fallback and source of IMDb-specific ratings data.

```
# Multi-source movie lookup
async def get_movie_enriched(query: str) -> dict:
    # Step 1: Search TMDb for canonical data
    tmdb_results = await tmdb_search(query)
    if not tmdb_results:
        return await omdb_search(query)  # Fallback

    movie = tmdb_results[0]
    tmdb_id = movie["id"]

    # Step 2: Get full TMDb details + credits + images
    details = await tmdb_details(tmdb_id,
        append="credits,images,videos,external_ids")

    # Step 3: Get IMDb rating from OMDb using IMDb ID
    imdb_id = details["external_ids"].get("imdb_id")
    if imdb_id:
        omdb_data = await omdb_lookup(imdb_id)
        details["imdb_rating"] = omdb_data.get("imdbRating")
        details["imdb_votes"] = omdb_data.get("imdbVotes")
        details["metascore"] = omdb_data.get("Metascore")

    return details
```

7.3 Security Considerations

When integrating movie data APIs into your application, follow these security best practices:

• **Never expose API keys in client-side code.** Always proxy API calls through your backend server. Store keys in environment variables or a secrets manager (e.g., AWS Secrets Manager, HashiCorp Vault).

• **Implement request signing for the IMDb GraphQL API.** Since it uses AWS IAM authentication, ensure your IAM roles follow the principle of least privilege.

• **Rate-limit your own users.** Implement per-user rate limiting on your backend to prevent any single user from exhausting your API quota.

• **Validate and sanitize all user input** before passing it to API query parameters to prevent injection attacks. URL-encode all query values.

• **Use HTTPS exclusively** for all API calls. Both OMDb and TMDb support HTTPS.

• **Monitor API usage.** Set up alerts when approaching rate limits or unusual usage patterns that could indicate abuse or credential leakage.

8 Appendices

8.1 HTTP Status Code Reference

Code	Status	Meaning	Action
200	OK	Request successful	Parse response body
401	Unauthorized	Invalid or missing API key/token	Check authentication credentials
403	Forbidden	Access denied (insufficient permissions)	Review API plan and permissions
404	Not Found	Resource does not exist	Verify the ID or endpoint URL
422	Unprocessable	Invalid parameters	Check parameter types and values
429	Too Many Requests	Rate limit exceeded	Implement backoff; wait before retrying
500	Server Error	Internal API error	Retry with exponential backoff
503	Service Unavailable	API temporarily down	Retry after delay; check status page

Note: OMDb always returns HTTP 200 even for application errors. Always check the Response field in the body.

8.2 Full Response Field Glossary

This glossary maps common movie data fields across all three API sources for cross-reference when building a multi-source integration.

Data Point	OMDb Field	TMDb Field	IMDb GraphQL
Movie Title	Title	title	titleText.text
Release Year	Year	release_date	releaseYear.year
IMDb Rating	imdbRating	N/A (own rating)	ratingsSummary.aggregateRating
User Votes	imdbVotes	vote_count	ratingsSummary.voteCount
Plot Summary	Plot	overview	plot.plotText.plainText
Genre(s)	Genre	genres[].name	genres.genres[].text
Runtime	Runtime	runtime (minutes)	runtime.seconds
Director(s)	Director	credits.crew[]	principalCredits[].credits[]
Cast	Actors	credits.cast[]	principalCredits[].credits[]
Poster URL	Poster	poster_path (+ base)	primaryImage.url
IMDb ID	imdbID	external_ids.imdb_id	id

Data Point	OMDb Field	TMDb Field	IMDb GraphQL
`Budget`	N/A	budget	N/A (separate product)
`Revenue`	N/A	revenue	N/A (Box Office product)
`Tagline`	N/A	tagline	N/A

Table 7: Cross-API field mapping reference

8.3 Useful Resources & Links

- **OMDb API Official Site**
  `https://www.omdbapi.com/`
- **OMDb API Key Registration**
  `https://www.omdbapi.com/apikey.aspx`
- **TMDb API Documentation**
  `https://developer.themoviedb.org/docs/getting-started`
- **TMDb API Reference**
  `https://developer.themoviedb.org/reference/intro/getting-started`
- **IMDb Developer Portal**
  `https://developer.imdb.com/`
- **IMDb API Help Article**
  `https://help.imdb.com/article/imdb/general-information/introducing-the-imdb-api/`
- **IMDb Non-Commercial Datasets**
  `https://datasets.imdbws.com/`
- **AWS Data Exchange (IMDb)**
  `https://aws.amazon.com/marketplace/ (search: IMDb)`
- **OMDb Python Wrapper (omdb.py)**
  `https://pypi.org/project/omdb/`
- **TMDb Python Wrapper (tmdbv3api)**
  `https://pypi.org/project/tmdbv3api/`

— End of Document —

This document was prepared as a comprehensive developer reference.
For the latest API changes, always consult the official documentation links above.