The AI Automation Starter Kit

A Practical Guide to Local LLMs

By SuperSQA.com

Table of Contents

Click any section below to jump directly to that content. Links work in both web view and PDF format.

Welcome, Engineer!

AI is a powerful tool, and with a solid engineering foundation, it can become your next superpower. This guide is your first step into a world of uncompromised privacy, zero running costs, and complete control over your AI workflows. We'll show you how to leverage the power of local LLMs with **Ollama** to supercharge your automation and testing processes.

This is a **printable PDF version** of our online guide, with expanded content, code examples, and a complete API toolkit to get you started immediately.

1. What is Ollama?

Ollama is a powerful, open-source tool that helps you run large language models (LLMs) on your computer. This means you can build and test AI tools without needing an internet connection or expensive cloud subscriptions. It's a game-changer for anyone who works in software development and quality assurance.

> ### Think of it like Docker, but for AI.
>
> Just as Docker packages a full application to make it easy to run, Ollama does the same thing for AI models. It takes care of all the complex parts so you can focus on building and testing your projects.

The main reason QA professionals love Ollama is because it keeps your data private and secure. By running AI models on your own machine, your confidential information never leaves your computer. This makes it a perfect solution for testing and building secure local AI applications.

2. The Ollama Advantage

- **Unmatched Privacy & Security:** Your data never leaves your computer. This is essential for handling confidential test data and proprietary code without risk.
- **Zero Cost to Run:** There are no API fees or usage costs. After the initial model download, you can run as many tests and queries as you want without worrying about a bill.
- **Offline Capability:** All models run entirely offline. You can test and develop your AI applications anywhere, without needing a reliable internet connection.
- **Complete Control:** You have full control over the models, their versions, and their configurations. This is critical for creating consistent and repeatable tests.

3. How It Works: A Simple 3-Step Process

1. **Install Ollama:** Download and install the desktop application. It runs a local server for you.
2. **Pull a Model:** Use a simple command to download a model from the Ollama library. It's like pulling a Docker image.
3. **Interact with the Model:** Start chatting in your terminal or make an API call from your code. It's that easy!

4. Key Model Specifications: The Anatomy of an AI

When choosing a model, two specifications are the most important to consider for performance and capability: the number of **parameters** and the **context window**.

- **Parameters:** The total number of parameters in a model's neural network, measured in billions (B). This is a primary indicator of a model's intelligence and ability to handle complex tasks. More parameters generally mean a smarter model, but they also demand significantly more GPU memory (VRAM) and processing power to run.
- **Context Window:** This defines the maximum number of tokens (words or pieces of data) a model can process at one time. It's essentially the model's memory for a single conversation or task. A larger context window allows the model to "remember" more of a conversation or analyze larger documents and codebases, which is crucial for complex, multi-step tasks.

5. Essential CLI Commands for QA

The command-line interface (CLI) is a powerful tool for rapid testing and managing your local environment.

Quick Reference: Your Daily Toolkit

- `ollama run llama3`: Pulls a model if it doesn't exist and starts an interactive session.
- `ollama pull llama3`: Downloads a model without running it. Perfect for setting up your environment offline.
- `ollama list`: Shows all the models you have downloaded locally.
- `ollama rm llama3`: Removes a specific model from your local machine, freeing up disk space.

Comprehensive Command Reference

Command	Description
`ollama -h`	Displays a list of all available commands and flags.
`ollama serve`	Starts the Ollama server. This is often not needed as the desktop app handles it.
`ollama show <model>`	Shows detailed information about a model's parameters and configuration.
`ollama ps`	Lists all currently running models.
`ollama create <model> -f <modelfile>`	Creates a custom model from a Modelfile.
`ollama cp <source> <target>`	Copies a model with a new name.
`ollama push <model>`	Uploads a model to a registry.

6. Using the REST API for Automation

The true power of Ollama for QA automation comes from its REST API. You can send requests to your local Ollama server from any programming language or testing tool to integrate LLMs directly into your test suites and applications.

API Endpoints: Your Reference

Endpoint	Description
`POST /api/generate`	Generates a response from a model for a single-turn prompt.
`POST /api/chat`	Generates a chat completion with full conversation history.

Endpoint	Description
`GET /api/tags`	Lists all local models (equivalent to `ollama list`).
`POST /api/pull`	Downloads a model from the library (equivalent to `ollama pull`).
`DELETE /api/delete`	Deletes a model from the local machine (equivalent to `ollama rm`).
`GET /api/show`	Shows information about a model (equivalent to `ollama show`).

7. The AI QA Pro Toolkit: Your Projects

Here are the complete, ready-to-use scripts and assets from the toolkit. These projects are designed to get you started immediately. The `llama3` model is used for the examples.

Python Scripts for Automation

These scripts are your hands-on toolkit for testing and building with AI.

`ollama_generate.py`

```
import requests
import json

# This is a simplified example for the printable guide.
# Get the full, runnable scripts in the GitHub repository.

def generate_text(prompt, model='llama3'):
    url = 'http://localhost:11434/api/generate'
    headers = {'Content-Type': 'application/json'}
    data = {
        'model': model,
        'prompt': prompt,
        'stream': False
    }

    try:
        response = requests.post(
            url, headers=headers, data=json.dumps(data)
        )
        response.raise_for_status()
        return response.json()['response']
    except requests.exceptions.RequestException as e:
        print(f'Error during API call: {e}')
        return None
```

```
# Example Usage:
result = generate_text('Write 5 high-level test cases for a user login f
if result:
    print(result)
```

`ollama_chat.py`

```
import requests
import json

# This is a simplified example for the printable guide.
# Get the full, runnable scripts in the GitHub repository.

def chat_completion(messages, model='llama3'):
    url = 'http://localhost:11434/api/chat'
    headers = {'Content-Type': 'application/json'}
    data = {
        'model': model,
        'messages': messages,
        'stream': False
    }

    try:
        response = requests.post(
            url,
            headers=headers,
            data=json.dumps(data)
        )
        response.raise_for_status()
        return response.json()['message']['content']
    except requests.exceptions.RequestException as e:
        print(f'Error during API call: {e}')
        return None

# Example Usage with a persona:
chat_history = [
    {'role': 'system',
     'content': 'You are a QA automation expert who helps write test pla
    {'role': 'user',
```

```
    'content': 'Write 5 high-level test cases for a user login form.'}
]

result = chat_completion(chat_history)
if result:
    print(result)
```

8. Your Postman Collection

This collection is a complete set of pre-built API calls for Ollama. Simply **import the `.json` file into Postman** and start testing immediately. This will save you hours of setup time.

`ollama_api_collection.json`

```
{
  "info": {
    "_postman_id": "a910f5e1-2771-460d-9b1b-f72671b56950",
    "name": "Ollama API Collection"
  },
  "item": [
    {
      "name": "Model Management",
      "item": [
        { "name": "List All Local Models" },
        { "name": "Pull a Model" },
        { "name": "Delete a Model" }
      ]
    },
    {
      "name": "Generate Completions",
      "item": [
        { "name": "Generate Text (Non-Streaming)" },
        { "name": "Generate JSON Output" },
        { "name": "Generate with Temperature and Seed" }
      ]
    },
    {
```

```
      "name": "Chat Completions",
      "item": [
        { "name": "Single-Turn Chat" },
        { "name": "Multi-Turn Chat with Persona" }
      ]
    }
  ]
}
```

This document is part of a larger toolkit for QA professionals. To learn more about AI automation, connect with our community, and take your skills to the next level, visit **SuperSQA.com**.