Ollama on Ubuntu: Llama 3.2 + Web App Backend

Step-by-step guide (updated 2026-01-06)

Ollama is a tool that helps you run large language models (LLMs) on your own machine (server or laptop). After a model is downloaded once, you can chat and build applications without sending your prompts to a cloud service.

What you will learn

• What Ollama is and why people use it
• Hardware requirements (simple sizing guidance)
• Install Ollama on Ubuntu Server
• Download and run **llama3.2** (1B/3B sizes)
• Use Ollama as an API backend for a web app (Node.js example)
• Expose Ollama safely on your LAN (OLLAMA_HOST)
• Common troubleshooting (disk full, ports, firewall)

1. What is Ollama?

Ollama is a local model runner. It installs a background service and provides:
• A CLI: **ollama** (pull models, run chats)
• A local HTTP API (default: **127.0.0.1:11434**) so your apps can call the model programmatically
• A model library (e.g., Llama, Mistral, Gemma) you can download and run locally

Why use it? Privacy, offline use after download, and easy integration into your own apps.

2. Models ("variants") you can run in Ollama

Ollama itself is one program. The “variants” usually refer to the **models** and their sizes (1B, 3B, 7B, 13B, etc.). Smaller models are faster and need less RAM; larger models can be smarter but need more resources.

Model family	Example Ollama name	Typical use
General chat	llama3.2, mistral, gemma	Everyday Q&A;, summaries, basic reasoning
Coding	codellama, deepseek-coder (if available)	Code help, explanations, refactoring
Vision (image)	llava (if available)	Image + text tasks (requires a vision-capable model)

3. Minimum hardware requirements (practical)

Exact requirements depend on the model size and quantization, but the table below works well for planning.

Target model size	Recommended RAM	CPU/GPU notes	Good for
1B–3B (llama3.2:1b / 3b)	8–16 GB	CPU-only is fine; SSD helps	Chatbots, simple assistants
7B–13B	16–32 GB	CPU works but slower; GPU helps a lot	Better reasoning, longer writing
30B–70B	64 GB+	Usually needs strong GPU/VRAM	Advanced tasks (heavy setup)

4. Install Ollama on Ubuntu Server

These steps follow Ollama's official Linux installation method.

```
sudo apt update && sudo apt upgrade -y

sudo apt install -y curl ca-certificates

curl -fsSL https://ollama.com/install.sh | sh
```

Verify installation:

```
ollama --version
systemctl status ollama
```

5. Download and run Llama 3.2

Llama 3.2 is available in small sizes (1B and 3B). You can pull the default tag or specify the size tag.

```
ollama pull llama3.2
# or explicitly:
ollama pull llama3.2:1b
ollama pull llama3.2:3b
```

Run an interactive chat:

```
ollama run llama3.2
# exit with Ctrl+d or /bye
```

List installed models:

```
ollama list
```

6. Use Ollama as an API (quick test)

Ollama provides an HTTP API (default: localhost:11434). Example generate request (non-streaming):

```
curl http://localhost:11434/api/generate -d '{
  "model": "llama3.2",
  "prompt": "Explain AI in 1 line for kids.",
  "stream": false
}'
```

7. Expose Ollama on your LAN (optional)

By default Ollama binds to 127.0.0.1:11434 (local only). To allow LAN access, set **OLLAMA_HOST**. For systemd-managed installs, use a service override.

```
sudo systemctl edit ollama.service
```

Add this override, then save:

```
[Service]
Environment="OLLAMA_HOST=0.0.0.0:11434"

sudo systemctl daemon-reexec
sudo systemctl restart ollama
```

If using a firewall (UFW), allow the port:

```
sudo ufw allow 11434/tcp
```

8. Use Ollama as the backend for a web app (Node.js)

Recommended architecture: **Browser** → **Your Backend** → **Ollama**. Do not expose Ollama directly to the public internet. Put authentication and rate-limiting on your backend.

8.1 Install Node.js + create a backend

```
sudo apt update
sudo apt install -y nodejs npm

mkdir -p ~/ollama-backend && cd ~/ollama-backend
npm init -y
npm i express
```

8.2 Create server.js (API + simple health page)

```
const express = require("express");
const app = express();
```

```
app.use(express.json());

const OLLAMA_URL = process.env.OLLAMA_URL || "http://localhost:11434";

app.post("/chat", async (req, res) => {
  try {
    const prompt = req.body.prompt || "Hello!";
    const response = await fetch(`${OLLAMA_URL}/api/generate`, {
      method: "POST",
      headers: { "Content-Type": "application/json" },
      body: JSON.stringify({ model: "llama3.2", prompt, stream: false }),
    });
    const data = await response.json();
    res.json({ reply: data.response });
  } catch (err) {
    res.status(500).json({ error: err.message });
  }
});

app.get("/", (req, res) => res.send("Backend is running. Use POST /chat"));

app.listen(3000, "0.0.0.0", () => console.log("Backend running on port 3000"));
```

8.3 Run and test

```
node server.js
```

Test from the same server:

```
curl -X POST http://localhost:3000/chat \
  -H "Content-Type: application/json" \
  -d '{"prompt":"Explain music in 10 words"}'
```

9. Troubleshooting (common)

Disk full: Check **df -h**. Free space or extend the filesystem (common on LVM installs).
Port not reachable: Check listening sockets (**ss -lntp**) and firewall rules (UFW).
Browser shows “Cannot GET /chat”: /chat is POST-only; use POST or add a UI route.
Remote Ollama access: Set OLLAMA_HOST or keep Ollama local and proxy through your backend.

References (official)

• https://docs.ollama.com/linux
• https://docs.ollama.com/api/generate
• https://docs.ollama.com/faq
• https://ollama.com/library/llama3.2