🚀 豆包私人网盘搭建教程（思路版）¶

> 从零开始，用豆包 + Cloudflare 搭建一个属于你自己的私人网盘/图床。
> 本文主要提供**思路和框架**，具体风格和细节你可以自己设计～

📖 这是什么？能干嘛？¶

简单来说，你可以把豆包变成一个**私人文件服务器**，然后通过 Cloudflare 的内网穿透功能，让你在任何地方都能访问这些文件。

你可以用它来：

- 📁 私人网盘：存文件、存资料，随时下载
- 🖼️ 私人图床：存图片，生成链接分享
- 🎬 视频分享：存视频，在线预览
- 📝 文档分享：存文档，朋友可以直接看

核心思路：

1. 在豆包里用 Python 跑一个文件服务器
2. 用 Cloudflare Tunnel 把内网服务暴露到公网
3. 加上密码验证，保证安全
4. 自己设计界面风格，想怎么改就怎么改

🎒 准备工作¶

在开始之前，你需要准备以下东西：

1. 一个豆包账号¶

这个不用多说，你正在用的就是。

2. 一个 Cloudflare 账号（免费）¶

官网：https://dash.cloudflare.com/
- 注册一个账号，免费版就够用了

3. 一个域名（可选但推荐）¶

- 如果你有自己的域名最好，可以用类似 `pan.你的域名.com` 这样的地址
- 如果没有域名也没关系，Cloudflare 会给你一个临时的 `*.trycloudflare.com` 域名（每次启动会变）

4. cloudflared 客户端¶

这是 Cloudflare Tunnel 的客户端程序，后面会教你怎么弄。

第一步：搭建文件服务器（核心思路）¶

我们先在豆包里搭建一个带密码的文件服务器。

1.1 基本原理¶

用 Python 的 `http.server` 模块写一个简单的 HTTP 服务器，加上：
- 密码验证（Cookie + HMAC 签名）
- 文件列表展示
- 图片预览
- 子文件夹浏览
- 多线程支持（防止卡住）

1.2 代码框架¶

下面是一个基础版本的代码框架，你可以在此基础上自己改样式、加功能：

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
带密码验证的文件服务器 - 基础框架
你可以根据自己的喜好修改样式、配色、功能等
"""
import os
import http.server
import socketserver
import urllib.parse
import hashlib
import hmac
import time
import socket

# ========== 配置区域（这些你都可以自己改） ==========
PORT = 9000                                    # 端口号
FILE_DIR = "/path/to/your/files"               # 文件存放目录
PASSWORD = "XXXXXX"                            # 改成你自己的密码！
COOKIE_NAME = "file_server_auth"               # Cookie 名称
COOKIE_SECRET = "your_secret_key_here"         # 改成你自己的随机字符串
# ================================================


def make_cookie_value():
    """生成认证 Cookie"""
    timestamp = str(int(time.time()))
    signature = hmac.new(
        COOKIE_SECRET.encode(),
        timestamp.encode(),
        hashlib.sha256
    ).hexdigest()
    return f"{timestamp}.{signature}"


def verify_cookie(cookie_value):
    """验证 Cookie 是否有效"""
    try:
        timestamp, signature = cookie_value.split(".", 1)
        expected = hmac.new(
            COOKIE_SECRET.encode(),
            timestamp.encode(),
            hashlib.sha256
        ).hexdigest()
        if hmac.compare_digest(signature, expected):
            if int(time.time()) - int(timestamp) < 7 * 24 * 3600:  # 7天有效期
                return True
    except:
        pass
```

```
    return False


def is_image(filename):
    """判断是不是图片文件"""
    ext = os.path.splitext(filename)[1].lower()
    return ext in ['.jpg', '.jpeg', '.png', '.gif', '.bmp', '.webp', '.svg']


def is_safe_path(sub_path):
    """检查路径是否安全，防止路径遍历攻击"""
    full_path = os.path.realpath(os.path.join(FILE_DIR, sub_path))
    base_path = os.path.realpath(FILE_DIR)
    return full_path.startswith(base_path + os.sep) or full_path == base_path


def format_size(size):
    """格式化文件大小"""
    if size < 1024:
        return f"{size} B"
    elif size < 1024 * 1024:
        return f"{size/1024:.1f} KB"
    elif size < 1024 * 1024 * 1024:
        return f"{size/1024/1024:.1f} MB"
    else:
        return f"{size/1024/1024/1024:.1f} GB"


def get_login_page():
    """
    登录页面 HTML
    这里你可以自己设计样式！
    """
    return """<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>🔐 文件服务器</title>
    <style>
        /* 在这里写你自己的 CSS 样式 */
        body {
            font-family: sans-serif;
            background: #f0f0f0;
            display: flex;
            justify-content: center;
            align-items: center;
            min-height: 100vh;
        }
        .login-box {
            background: white;
```

```
            padding: 40px;
            border-radius: 10px;
            box-shadow: 0 2px 10px rgba(0,0,0,0.1);
        }
        input {
            width: 100%;
            padding: 10px;
            margin: 10px 0;
            border: 1px solid #ddd;
            border-radius: 5px;
        }
        button {
            width: 100%;
            padding: 10px;
            background: #007bff;
            color: white;
            border: none;
            border-radius: 5px;
            cursor: pointer;
        }
    </style>
</head>
<body>
    <div class="login-box">
        <h2>🔐 文件服务器</h2>
        <p>请输入密码以访问</p>
        <form method="POST" action="/login">
            <input type="password" name="password" placeholder="请输入密码">
            <button type="submit">登录</button>
        </form>
    </div>
</body>
</html>"""


def get_file_list_page(sub_path=""):
    """
    文件列表页面 HTML
    这里你可以自己设计样式！
    比如：网格视图、列表视图、缩略图预览等等
    """
    if not is_safe_path(sub_path):
        sub_path = ""

    current_dir = os.path.join(FILE_DIR, sub_path)
    if not os.path.isdir(current_dir):
        sub_path = ""
        current_dir = FILE_DIR

    items = []
```

```
    # 返回上一级
    if sub_path:
        parent_path = os.path.dirname(sub_path)
        items.append({
            'name': '..',
            'is_dir': True,
            'path': parent_path,
        })

    # 列出目录内容
    for item in sorted(os.listdir(current_dir)):
        if item.startswith('.'):
            continue
        item_path = os.path.join(current_dir, item)
        item_rel_path = os.path.join(sub_path, item) if sub_path else item

        if os.path.isdir(item_path):
            items.append({
                'name': item,
                'is_dir': True,
                'path': item_rel_path,
            })
        elif os.path.isfile(item_path):
            items.append({
                'name': item,
                'is_dir': False,
                'path': item_rel_path,
                'size': format_size(os.path.getsize(item_path)),
                'is_image': is_image(item),
            })

    # 文件夹排前面
    dirs = [i for i in items if i['is_dir']]
    files = [i for i in items if not i['is_dir']]
    items = dirs + files

    # 生成 HTML（这里只是简单示例，你可以设计得更好看）
    items_html = ""
    for item in items:
        encoded_path = urllib.parse.quote(item['path'])
        if item['is_dir']:
            items_html += f'<li><a href="/browse/{encoded_path}">📁 {item["name"]}</a><
        else:
            if item.get('is_image'):
                items_html += f'<li><a href="/files/{encoded_path}">🖼️ {item["name"]}</
            else:
                items_html += f'<li><a href="/files/{encoded_path}">📄 {item["name"]}</

    return f"""<!DOCTYPE html>
```

```
<html>
<head>
    <meta charset="UTF-8">
    <title>📁 文件列表</title>
    <style>
        /* 在这里写你自己的 CSS 样式 */
        body {{
            font-family: sans-serif;
            padding: 20px;
        }}
        ul {{
            list-style: none;
            padding: 0;
        }}
        li {{
            padding: 8px;
            border-bottom: 1px solid #eee;
        }}
        a {{
            color: #007bff;
            text-decoration: none;
        }}
    </style>
</head>
<body>
    <h2>📁 当前目录：{sub_path or '根目录'}</h2>
    <ul>
        {items_html}
    </ul>
    <p><a href="/logout">退出登录</a></p>
</body>
</html>"""


class FileServerHandler(http.server.SimpleHTTPRequestHandler):
    def __init__(self, *args, **kwargs):
        super().__init__(*args, directory=FILE_DIR, **kwargs)

    def do_GET(self):
        # 检查认证
        cookie = self.headers.get('Cookie', '')
        auth_cookie = None
        for c in cookie.split(';'):
            c = c.strip()
            if c.startswith(f"{COOKIE_NAME}="):
                auth_cookie = c[len(COOKIE_NAME)+1:]
                break

        # 登出
        if self.path == '/logout':
```

```
            self.send_response(302)
            self.send_header('Location', '/')
            self.send_header('Set-Cookie', f'{COOKIE_NAME}=; Path=/; Max-Age=0')
            self.end_headers()
            return

        # 文件下载/预览
        if self.path.startswith('/files/'):
            if auth_cookie and verify_cookie(auth_cookie):
                filename = urllib.parse.unquote(self.path[7:])
                filepath = os.path.join(FILE_DIR, filename)
                if os.path.isfile(filepath):
                    self.send_response(200)
                    if is_image(filename):
                        ext = os.path.splitext(filename)[1].lower()
                        content_types = {
                            '.jpg': 'image/jpeg', '.jpeg': 'image/jpeg',
                            '.png': 'image/png', '.gif': 'image/gif',
                        }
                        self.send_header('Content-Type', content_types.get(ext, 'applic
                    else:
                        self.send_header('Content-Type', 'application/octet-stream')
                        self.send_header('Content-Disposition', f'attachment; filename='
                    self.send_header('Content-Length', str(os.path.getsize(filepath)))
                    self.end_headers()
                    with open(filepath, 'rb') as f:
                        self.wfile.write(f.read())
                    return
            self.send_response(302)
            self.send_header('Location', '/')
            self.end_headers()
            return

        # 浏览子目录
        if self.path.startswith('/browse/'):
            if auth_cookie and verify_cookie(auth_cookie):
                sub_path = urllib.parse.unquote(self.path[8:])
                html = get_file_list_page(sub_path)
                self.send_response(200)
                self.send_header('Content-Type', 'text/html; charset=utf-8')
                self.end_headers()
                self.wfile.write(html.encode())
            else:
                self.send_response(302)
                self.send_header('Location', '/')
                self.end_headers()
            return

        # 首页
        if self.path == '/' or self.path == '':
```

```
            if auth_cookie and verify_cookie(auth_cookie):
                html = get_file_list_page()
            else:
                html = get_login_page()
            self.send_response(200)
            self.send_header('Content-Type', 'text/html; charset=utf-8')
            self.end_headers()
            self.wfile.write(html.encode())
            return

        self.send_response(404)
        self.end_headers()

    def do_POST(self):
        # 登录
        if self.path == '/login':
            content_length = int(self.headers.get('Content-Length', 0))
            post_data = self.rfile.read(content_length).decode()
            params = urllib.parse.parse_qs(post_data)
            password = params.get('password', [''])[0]

            if password == PASSWORD:
                cookie_value = make_cookie_value()
                self.send_response(302)
                self.send_header('Location', '/')
                self.send_header('Set-Cookie', f'{COOKIE_NAME}={cookie_value}; Path=/; I
                self.end_headers()
            else:
                # 密码错误，重新显示登录页（可以加错误提示）
                html = get_login_page()
                self.send_response(200)
                self.send_header('Content-Type', 'text/html; charset=utf-8')
                self.end_headers()
                self.wfile.write(html.encode())
            return

        self.send_response(404)
        self.end_headers()


if __name__ == '__main__':
    os.chdir(FILE_DIR)

    # 多线程 + IPv6 双栈支持（防止并发请求卡住）
    class DualStackThreadingTCPServer(socketserver.ThreadingMixIn, socketserver.TCPServ
        address_family = socket.AF_INET6
        allow_reuse_address = True
        daemon_threads = True

    class ThreadingTCPServer(socketserver.ThreadingMixIn, socketserver.TCPServer):
```

```
        allow_reuse_address = True
        daemon_threads = True

    try:
        httpd = DualStackThreadingTCPServer(("::", PORT), FileServerHandler)
        print(f"文件服务器运行在端口 {PORT} (多线程，支持 IPv4 和 IPv6)")
    except OSError:
        httpd = ThreadingTCPServer(("", PORT), FileServerHandler)
        print(f"文件服务器运行在 http://localhost:{PORT} (多线程)")

    print(f"文件目录: {FILE_DIR}")
    httpd.serve_forever()
```

> 💡 **提示**：
> - 上面的代码只是一个**基础框架**，样式很朴素
> - 你可以自己设计好看的界面（比如 Windows 文件夹风格、深色主题、渐变配色等等）
> - 密码一定要改成你自己的！
> - `COOKIE_SECRET` 也要改成随机字符串，更安全

1.3 启动服务器¶

把代码保存为 `file-server.py` ，然后运行：

```
python3 file-server.py &
```

启动成功后，可以在浏览器访问 `http://localhost:9000` 测试一下。

第二步：配置 Cloudflare Tunnel 内网穿透¶

现在文件服务器只能在豆包内部访问，我们需要用 Cloudflare Tunnel 把它暴露到公网。

2.1 什么是 Cloudflare Tunnel？¶

简单说，就是在你的服务器和 Cloudflare 之间建立一条加密隧道，外部流量通过 Cloudflare 进入你的服务器，不需要公网 IP，也不用开端口。

优点：

- ✅ 免费
- ✅ 安全（流量经过 Cloudflare 清洗）
- ✅ 不需要公网 IP
- ✅ 支持自定义域名

2.2 下载 cloudflared 客户端¶

方法一：从 GitHub 下载（官方）

- 下载地址：https://github.com/cloudflare/cloudflared/releases
- 找到最新版本，下载 `cloudflared-linux-amd64` 文件

方法二：用 npm 安装

```
npm install -g cloudflared
```

下载后添加执行权限：

```
chmod +x cloudflared
```

2.3 在 Cloudflare 面板创建隧道¶

1. 登录 Cloudflare Zero Trust 面板：https://dash.teams.cloudflare.com/
2. 左侧菜单找到 Networks → Tunnels
3. 点击 Create a tunnel
4. 给隧道起个名字
5. 点击 Save tunnel

2.4 获取 Token 并启动隧道¶

创建隧道后，你会看到一个安装命令，里面有一个很长的 token，复制下来。

然后在豆包里运行：

```
./cloudflared tunnel --no-autoupdate run --token 你的token
```

后台运行（推荐）：

```
nohup ./cloudflared tunnel --no-autoupdate run --token 你的token > cloudflared.log 2>&1
```

> ⚠️ **重要**：这个 token 很重要，不要泄露给别人！

2.5 配置公共主机名¶

回到 Cloudflare 面板：
1. 在隧道详情页，点击 Public Hostname 标签
2. 点击 Add a public hostname
3. 填写：
- Subdomain：比如 `pan` （这样访问地址就是 pan.你的域名.com）
- Domain：选择你的域名
- Type：HTTP
- URL：`localhost:9000` （和你文件服务器的端口一致）
4. 点击 Save hostname

第三步：测试访问¶

等待 1-2 分钟，让配置生效。

然后在浏览器里访问你配置的域名，比如：

```
https://pan.你的域名.com
```

如果能看到登录页面，输入密码能进去，就说明成功了！

🎉 恭喜你，私人网盘搭建完成！

进阶玩法（我的一些创新思路）¶

这部分是我自己摸索出来的一些实用技巧，分享给你～

📝 1. 日志记录¶

给服务器加上日志，方便排查问题和查看访问记录。

实现思路：
- 启动时把输出重定向到日志文件
- 定期查看日志

```
# 文件服务器日志
nohup python3 file-server.py > file-server.log 2>&1 &

# cloudflared 日志
nohup ./cloudflared tunnel --no-autoupdate run --token 你的token > cloudflared.log 2>&1
```

查看日志：

```
# 实时查看
tail -f file-server.log

# 查看最后 100 行
tail -n 100 cloudflared.log
```

📦 2. 打包备份¶

把所有配置文件打包成一个压缩包，方便备份和迁移。

实现思路：
- 把代码、配置、文档都打包
- 新环境直接解压就能用

```
# 打包
zip -r backup.zip code/ bin/ document/

# 解压
unzip backup.zip
```

建议：

- 定期备份，防止数据丢失
- 备份包可以存到别的地方（比如本地电脑）

💓 3. 保活脚本¶

如果担心虚拟机休眠导致服务挂掉，可以写一个简单的保活脚本。

实现思路：

- 每隔一段时间访问一下本地服务
- 写个心跳日志

```
#!/bin/bash
while true; do
    echo "[$(date)] 保活心跳" >> keepalive.log
    curl -s http://localhost:9000/ > /dev/null 2>&1
    sleep 1800  # 每30分钟一次
done
```

运行：

```
chmod +x keepalive.sh
nohup ./keepalive.sh > /dev/null 2>&1 &
```

> 💡 注意：这个不一定能 100% 延长虚拟机存活时间，只是一个尝试。

🖼️ 4. 自动保存图片¶

你可以跟 AI 约定，以后你发的图片都自动保存到指定文件夹里，这样你的图床就自动更新了。

实现思路：

- 跟 AI 说清楚规则（保存到哪个文件夹、命名规则等）
- 以后直接发图就行，不用每次手动保存

📊 5. 文件树文档¶

写一个文件树文档，实时记录你的文件结构，方便查看和管理。

实现思路：

- 用树形结构展示所有文件
- 标注文件大小、格式等信息
- 文件变动时及时更新

🔐 6. 敏感文件隐藏¶

有些敏感文件（比如密码文档、配置文件等），不要出现在公开的文件列表里。

实现思路：

- 在代码里过滤掉敏感文件名
- 或者把敏感文件放到不对外的目录
- 文件树文档里也不要显示敏感文件

❓ 常见问题¶

Q1: 访问域名显示 502 Bad Gateway¶

A: 说明 cloudflared 连不上你的文件服务器。检查：

- 文件服务器是不是在运行？
- 端口是不是一致？
- Cloudflare 面板里的 URL 是不是填对了？

Q2: 网站加载很慢或者卡住¶

A: 可能是单线程服务器的问题。确保你用的是多线程版本（代码里有 `ThreadingMixIn`）。

Q3: 隧道经常离线¶

A: 可能是虚拟机休眠了。可以试试保活脚本，或者定期来豆包说句话。

Q4: 没有自己的域名可以用吗¶

A: 可以！用 Cloudflare Quick Tunnel 模式，会给你一个临时的 `*.trycloudflare.com` 域名。缺点是每次启动都会变。

Quick Tunnel 启动命令：

```
./cloudflared tunnel --url http://localhost:9000
```

Q5: 密码忘了怎么办¶

A: 查看 `file-server.py` 里的 `PASSWORD` 配置项。

📚 相关资源¶

- **Cloudflare 官网**：https://www.cloudflare.com/
- **cloudflared 下载**：https://github.com/cloudflare/cloudflared/releases
- **Cloudflare Tunnel 文档**：https://developers.cloudflare.com/cloudflare-one/connections/connect-networks/
- **Python http.server 文档**：https://docs.python.org/3/library/http.server.html

---

💡 写在最后¶

这个教程主要提供**思路和框架**，具体的样式、功能、配色都可以自己设计。

我自己的版本用了：
- Windows 文件夹风格的界面
- 自定义的配色主题
- 图片缩略图预览
- 子文件夹浏览
- 多线程支持
- 等等...

你可以根据自己的喜好来改造，玩出自己的风格～

有问题就多问 AI，它会帮你解决的！

教程版本：v1.0（思路版）
更新时间：2026年6月20日
核心思路：豆包 + Cloudflare Tunnel + Python 文件服务器
