目标

用最少的工程量，把 `snap-stanford/Biomni` 跑成一个可交互的网页版（“像 app.biomni.stanford.edu 一样”），先在一台 EC2 上上线，再逐步演进到更稳的架构。本文给出可直接落地的文件结构、代码、命令和 AWS 配置。

一、上线策略（两阶段）

Stage A = 最快上线（1 台 EC2 + Docker Compose） - EC2: Ubuntu 22.04，t3.xlarge（内存 16GB 起），EBS ≥ 200GB（第一次运行会自动拉取 ~11GB 数据湖 + 环境缓存）。 - 反向代理: Nginx（80/443），Let’s Encrypt 自动签证书（可选）。 - 后端: FastAPI 包一层 Biomni `A1` Agent 的 HTTP API。 - 前端: 极简静态网页（HTML/JS），表单 + 输出区。 - LLM: 先用 Anthropic/OpenAI API（最省事），后续可切换 AWS Bedrock。 - Secrets: `.env` 本地 + EC2 上用 `docker-compose` 注入（生产建议迁移到 SSM Parameter Store/Secrets Manager）。

Stage B = 稳定演进（ECS Fargate + S3/CloudFront + SSM） - 前端：S3 静态托管 + CloudFront。 - 后端：ECS Fargate（1 个 Service 1–3 个 Task），ALB 暴露 /api。 - 数据：EFS 作为 `/data`（跨任务持久化 Biomni 数据湖）。 - Secrets：AWS Secrets Manager/SSM。

> 建议：先完成 Stage A（1–2 天），上线可用；接着把镜像推到 ECR，再平滑迁移到 Stage B。

二、准备工作

- 一个域名（可选，若只用 IP 访问可跳过 HTTPS）。
- AWS 账号 + `awscli` 已配置（有 `EC2FullAccess`、`ECRFullAccess`、`SSMFullAccess`、`AmazonS3FullAccess`、`BedrockInvokeModel` 权限，最小权限可后续收紧）。
- 本机已安装：`git`、`docker`、`docker compose`、`make`。
- 拥有至少一种可用 LLM Key（**先用 Anthropic/OpenAI 更快**）。

三、项目骨架

```
biomni-aws/
├─ server/
│  ├─ app.py                # FastAPI 封装 Biomni
│  ├─ requirements.txt
│  └─ entrypoint.sh
├─ web/
│  ├─ index.html
│  ├─ app.js
```

```
│   └─ style.css
├─ Dockerfile
├─ compose.yaml
├─ .env.example                # 复制为 .env 后填入 Key
└─ README_RUN.md
```

四、后端（FastAPI）

server/requirements.txt

```
fastapi==0.111.0
uvicorn[standard]==0.30.0
pydantic==2.7.3
python-dotenv==1.0.1
biomni>=0.0.6
```

server/app.py

```
import os
import traceback
from fastapi import FastAPI, HTTPException
from fastapi.middleware.cors import CORSMiddleware
from pydantic import BaseModel
from dotenv import load_dotenv

load_dotenv()

# Biomni
from biomni.agent import A1

DATA_PATH = os.getenv("BIOMNI_DATA_PATH", "/data")
LLM_NAME = os.getenv("BIOMNI_LLM", os.getenv("DEFAULT_LLM", "claude-
sonnet-4-20250514"))

# 可全局化配置（与官方 README 一致）
try:
    from biomni.config import default_config
    if os.getenv("DEFAULT_LLM"):
        default_config.llm = os.getenv("DEFAULT_LLM")
    if os.getenv("BIOMNI_TIMEOUT_SECONDS"):
        default_config.timeout_seconds =
int(os.getenv("BIOMNI_TIMEOUT_SECONDS"))
```

```
except Exception:
    pass

app = FastAPI(title="Biomni API")
app.add_middleware(
    CORSMiddleware,
    allow_origins=["*"],
    allow_credentials=True,
    allow_methods=["*"],
    allow_headers=["*"],
)

# 懒加载，首次请求再初始化（避免容器冷启动时间过长）
_agent = None

def get_agent():
    global _agent
    if _agent is None:
        _agent = A1(path=DATA_PATH, llm=LLM_NAME)
    return _agent

class GoReq(BaseModel):
    prompt: str

@app.get("/health")
async def health():
    return {"ok": True}

@app.post("/api/agent/go")
async def agent_go(req: GoReq):
    try:
        agent = get_agent()
        # Biomni 的 go 返回对象可能是 str/dict，自适应处理
        result = agent.go(req.prompt)
        # 标准化为字符串
        if result is None:
            out = "(No result returned by agent)"
        else:
            out = str(result)
        return {"ok": True, "result": out}
    except Exception as e:
        traceback.print_exc()
        raise HTTPException(status_code=500, detail=str(e))
```

server/entrypoint.sh

```
#!/usr/bin/env bash
set -euo pipefail

# Biomni 首次运行会尝试下载数据湖（~11GB）；建议挂载 /data 持久化
exec uvicorn server.app:app --host 0.0.0.0 --port 8000
```

五、前端（极简版）

web/index.html

```
<!doctype html>
<html>
<head>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1" />
  <title>Biomni (Unofficial) – Minimal UI</title>
  <link rel="stylesheet" href="/web/style.css" />
</head>
<body>
  <div class="container">
    <h1>Biomni – Minimal Co-pilot</h1>
    <p class="hint">适合复杂科研任务：数据分析、流程设计、深度检索等。请输入自然语言指令。</
p>
    <textarea id="prompt" rows="8" placeholder="例如：‘对这个 scRNA-seq 数据做细胞注
释，并提出后续实验假设。数据在 /data/xxx.h5ad’"></textarea>
    <button id="run">Run</button>
    <pre id="out"></pre>
  </div>
  <script src="/web/app.js"></script>
</body>
</html>
```

web/style.css

```
body { font-family: -apple-system, BlinkMacSystemFont, Segoe UI, Roboto,
Helvetica, Arial, sans-serif; }
.container { max-width: 900px; margin: 40px auto; padding: 16px; }
textarea { width: 100%; font-size: 14px; }
#run { margin: 8px 0; padding: 10px 16px; cursor: pointer; }
pre { white-space: pre-wrap; background: #f7f7f7; padding: 12px; border-radius:
```

```
8px; }
.hint { color: #666; }
```

web/app.js

```
const $ = (s) => document.querySelector(s);
$("#run").onclick = async () => {
  const prompt = $("#prompt").value.trim();
  if (!prompt) return;
  $("#out").textContent = "Running... (首次运行可能较慢，需准备数据湖)";
  try {
    const resp = await fetch("/api/agent/go", {
      method: "POST",
      headers: { "Content-Type": "application/json" },
      body: JSON.stringify({ prompt }),
    });
    const data = await resp.json();
    $("#out").textContent = data.ok ? data.result : (data.detail || "Error");
  } catch (e) {
    $("#out").textContent = e.toString();
  }
};
```

六、容器与编排

Dockerfile

```
FROM mambaorg/micromamba:1.5.7

# 创建工作目录
WORKDIR /app

# 复制需求
COPY server/requirements.txt /app/server/requirements.txt
COPY server /app/server
COPY web /app/web
COPY server/entrypoint.sh /entrypoint.sh
RUN chmod +x /entrypoint.sh

# 用 micromamba 建一个干净的 env，安装 biomni 及依赖
RUN micromamba install -y -n biomni -c conda-forge python=3.10 &&
    micromamba run -n biomni pip install --no-cache-dir -r /app/server/
```

```
requirements.txt &&
    micromamba clean --all --yes

# 运行时环境变量（可被 docker-compose 覆盖）
ENV BIOMNI_DATA_PATH=/data
    DEFAULT_LLM=claude-sonnet-4-20250514

EXPOSE 8000
CMD ["/bin/bash", "/entrypoint.sh"]
```

compose.yaml

```
services:
  biomni:
    build: .
    image: biomni-mini:latest
    container_name: biomni
    restart: unless-stopped
    ports:
      - "8000:8000"
    environment:
      # —— 选择一种 LLM 提供方 ——
      # Anthropic（推荐先跑通）：
      ANTHROPIC_API_KEY: ${ANTHROPIC_API_KEY}
      DEFAULT_LLM: ${DEFAULT_LLM:-claude-sonnet-4-20250514}
      # OpenAI（可选）：
      OPENAI_API_KEY: ${OPENAI_API_KEY}
      # AWS Bedrock（后续可切换）：
      AWS_REGION: ${AWS_REGION}
      LLM_SOURCE: ${LLM_SOURCE}
      # Biomni
      BIOMNI_DATA_PATH: /data
      BIOMNI_TIMEOUT_SECONDS: 1200
    volumes:
      - ./data:/data
      - ./web:/app/web:ro

  nginx:
    image: nginx:1.27
    container_name: nginx
    restart: unless-stopped
    ports:
      - "80:80"
    volumes:
      - ./nginx.conf:/etc/nginx/nginx.conf:ro
      - ./web:/usr/share/nginx/html/web:ro
```

```
    depends_on:
      - biomni
```

nginx.conf

```
worker_processes  1;
error_log  /var/log/nginx/error.log warn;
pid        /var/run/nginx.pid;

events { worker_connections  1024; }

http {
  include       /etc/nginx/mime.types;
  default_type  application/octet-stream;
  sendfile        on;
  keepalive_timeout  65;

  server {
    listen 80;

    # 静态前端
    location /web/ { root /usr/share/nginx/html; try_files $uri $uri/ =404; }
    location =/ { return 302 /web/; }

    # 反代后端 API
    location /api/ {
      proxy_pass http://biomni:8000/;
      proxy_set_header Host $host;
      proxy_set_header X-Real-IP $remote_addr;
      proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
      proxy_http_version 1.1;
    }

    location /healthcheck { return 200 'ok'; }
  }
}
```

.env.example

```
# 任选一种或多种 Key；优先先用 Anthropic 跑通
ANTHROPIC_API_KEY=
OPENAI_API_KEY=

# Bedrock（后续可选）
```

```
AWS_REGION=us-east-1
LLM_SOURCE=Bedrock

# 默认 LLM 名称（可根据供应商更改）
DEFAULT_LLM=claude-sonnet-4-20250514
```

七、本地验证

```
# 1) 克隆你的 fork，并放入本文给出的文件
git clone https://github.com/yourname/biomni-aws && cd biomni-aws

# 2) 复制 .env 并填入 Key
cp .env.example .env && vi .env

# 3) 构建 & 启动
docker compose build
docker compose up -d

# 4) 访问 http://localhost/web/ ；或 curl 健康检查
curl http://localhost/healthcheck
```

> 首次调用 `/api/agent/go` 会触发 Biomni 数据湖下载（~11GB，较慢），请保证 `./data` 磁盘空间足够。

八、部署到 AWS（Stage A — 单机 EC2）

1. **创建 EC2**：Ubuntu 22.04，t3.xlarge（或更高），EBS 200–500GB，打开 **80/443/22**。
2. **安装 Docker**（User Data）：

```
#!/bin/bash
apt-get update -y
apt-get install -y ca-certificates curl gnupg
install -m 0755 -d /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | gpg --dearmor -
o /etc/apt/keyrings/docker.gpg
chmod a+r /etc/apt/keyrings/docker.gpg

echo
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/
docker.gpg] https://download.docker.com/linux/ubuntu
  $(. /etc/os-release && echo $VERSION_CODENAME) stable" |
  tee /etc/apt/sources.list.d/docker.list > /dev/null
```

```
apt-get update -y
apt-get install -y docker-ce docker-ce-cli containerd.io docker-buildx-
plugin docker-compose-plugin
usermod -aG docker ubuntu
mkdir -p /opt/biomni-aws && chown ubuntu:ubuntu /opt/biomni-aws
```

3. **拷贝项目到 EC2**：

```
# 在本机
scp -r biomni-aws ubuntu@YOUR_EC2:/opt/
ssh ubuntu@YOUR_EC2
cd /opt/biomni-aws
cp .env.example .env && nano .env  # 填写 Key
sudo docker compose build
sudo docker compose up -d
```

4. **（可选）HTTPS**：装 `certbot` + `nginx` 插件，或用 Cloudflare/ALB/ACM。

九、演进到 Stage B（ECS + S3/CloudFront）

- **镜像**：

```
aws ecr create-repository --repository-name biomni-mini
aws ecr get-login-password | docker login --username AWS --password-stdin
<account>.dkr.ecr.<region>.amazonaws.com
TAG=<account>.dkr.ecr.<region>.amazonaws.com/biomni-mini:latest
docker build -t $TAG . && docker push $TAG
```

- **后端**：创建 **ECS Fargate** Task（512–4096MB）+ Service，**挂载 EFS** 到 `/data`，把 `.env` 变量改为 **Secrets Manager/SSM**。
- **前端**：把 `web/` 上传到 **S3**，用 **CloudFront** 分发；/api 通过 **ALB** 指到 ECS 服务。

十、成本与资源建议

- **磁盘**：数据湖 + 缓存 ≥ 200GB，长期运行建议 500GB。
- **内存/CPU**：t3.xlarge 起步，GPU 不是必需（除非你集成了 GPU 工作流）。
- **流量**：首次数据下载较大，注意带宽与出网费用。
- **LLM 费用**：先选 Anthropic/OpenAI 的中杯模型，稳定后再切 Bedrock 统一计费。

十一、安全与沙箱

- Biomni **会执行 LLM 生成代码**。务必用容器隔离：
- 非 root 运行、只读根文件系统；
- 仅挂载 `/data` 可写；
- 需要时使用 seccomp/AppArmor；
- 生产流量走私有子网 + NAT；
- 凭证放 Secrets Manager/SSM，不写入镜像。

十二、常见坑位

- **数据湖下载失败/慢**：检查 `/data` 权限与磁盘空间；必要时提前在国内机器下载再同步到 EBS/EFS。
- **包冲突**：参考官方 `docs/known_conflicts.md`；某些工具默认未装，需要你额外 pip/conda。
- **Bedrock 调用**：确保账号开通对应模型的访问；若报签名错误，先用 Anthropic/OpenAI 直连验证。
- **go() 返回类型多样**：先统一转字符串返回；后续可以逐步解析结构化输出。

十三、下一步（从“能用”到“好用”）

- 前端升级为 Next.js + 流式输出（SSE/WebSocket）+ 历史会话；
- 任务队列（Celery/RQ + Redis）处理长任务；
- 异步日志与结果存档（S3 + DynamoDB/PG）；
- 权限与多租户（Cognito/自建 OAuth）；
- 可插拔工具面板（MCP/RAG 工具注册 UI）。

附：README_RUN.md（对外给同事实操版）

```
# Quick Run
cp .env.example .env && vi .env
# 本地
docker compose up -d --build
open http://localhost/web/
# 服务器
ssh ubuntu@EC2 && cd /opt/biomni-aws && docker compose up -d --build
```