快速上手实操指南

OpenSAI 快速上手指南

在一台电脑上用 10 分钟跑通集群启动、首个分布式任务、自定义应用接入与目标驱动调参。

产品概览 | 使用手册 | 智能体接入 | FAQ 与维护

只需一台电脑即可完成本指南全部操作：前三个步骤仅使用 CPU 与 Python 标准库（无需 GPU、模型权重、数据集、Docker 或 Node.js），第四步体验分布式聚类时再按提示安装 `numpy`。单机验证通过后，同一套命令与应用包可直接扩展到多台设备。



OpenSAI 四步快速上手路线

1. 安装环境并启动控制台

请准备 **Python 3.10 或更高版本**，克隆公开仓库（`git clone https://github.com/INSAI-institute/OpenSAI.git`）或解压 `OpenSAI_Release.zip`，在包含 `scripts/`、`examples/` 与 `Offloading-System/` 的项目根目录打开终端（首次安装需联网下载基础依赖）。

> **i 说明**
>
> **系统前置小贴士**：Ubuntu / Debian 系统自带 Python 通常需先执行 `sudo apt install python3-venv`。Linux / macOS 若本机未安装 `redis-server`，安装脚本会自动从源码编译便携版 Redis，此时需具备基础编译工具（`cc`、`make` 以及 `curl` 或 `wget`）；也可预先通过系统包管理器安装 `redis-server`。

Linux / macOS：

```bash
bash scripts/setup.sh
bash scripts/run.sh --with-web
source .venv/bin/activate
```

Windows PowerShell：

```powershell
powershell -ExecutionPolicy Bypass -File scripts\setup.ps1
powershell -ExecutionPolicy Bypass -File scripts\run.ps1 -WithWeb
.venv\Scripts\Activate.ps1
```

启动脚本会在本机拉起便携或本地 Redis、协调中枢（`server.py`）、1 个边缘工作设备进程（`run_device.py`）以及 Web 控制台桥接服务（`http://127.0.0.1:5000`）。

在已激活 `.venv` 的终端中检查在线设备：

```bash
opensai devices list
```

典型终端输出（JSON 摘要）：

```json
[
  {
    "device_id": "设备1",
    "allocation_index": 1,
    "status": "ONLINE",
    "cpu_cores": 8
  }
]
```

浏览器打开 `http://127.0.0.1:5000`，控制台默认显示英文，点击右上角 **`EN`** / **`中`** 按钮可随时切换中文与英文（旁侧图标可切换 **纸 / 墨** 浅深色主题，或点击 **一键演示 / Run demo** 立即体验内置流水线）。

> **✓ 验证通过标志**
>
> 终端返回至少 1 台 `ONLINE` 设备；浏览器左下角状态栏显示 **“已连接 · 实时”（`Connected · live`）**。若当前处于模拟集群预览模式（顶部提示条显示 **“演示数据”**），点击提示条右侧 **“切换到实时数据”**（或在 **“设置”** 页关闭演示数据）即可切回本机真实集群。

> **✦ 实用提示**
>
> 后续每当新开一个终端窗口时，请先进入项目根目录并重新激活虚拟环境（`source .venv/bin/activate` 或 `.venv\Scripts\Activate.ps1`）。

2. 提交首个 DAG 任务并解读决策

我们先运行一个零外部依赖的两阶段流水线 `pi_pipeline.json`：节点 1 通过蒙特卡洛采样估算圆周率，完成后触发节点 2 执行短暂汇合等待。先在本地校验结构，再提交到中枢：

```bash
opensai dag validate -f examples/dags/pi_pipeline.json
opensai dag submit -f examples/dags/pi_pipeline.json
```

`submit` 命令会立即返回本次作业编号（形如 `job_01M6S9WR8S84WV2920DQ8J3E2H`）。复制返回的 `job_id`，替换下方命令中的 `YOUR_JOB_ID` 等待执行完成并查看调度解释：

```bash
opensai result wait YOUR_JOB_ID
opensai explain YOUR_JOB_ID
```

典型返回关键字段如下：

返回字段 / 状态	含义与排查用途
`job_id`	本次执行的唯一作业号；后续查询结果、查看解释或在控制台检索均使用它
`QUEUED` / `RUNNING`	作业正在排队等待空闲槽位，或节点正在设备上运行
`SUCCEEDED`	所有必需步骤均已完成（可在 `nodes[].result` 中看到圆周率估算文本）

返回字段 / 状态	含义与排查用途
`duration_s`	整个作业从提交到完成的端到端墙钟耗时（单位：秒）
`explain` 输出	展示每个步骤被分配到哪台设备、分得几核 CPU，以及平台的决策依据

> ✓ **验证通过标志**
>
> `status` 为 `SUCCEEDED`，且在 Web 控制台左侧点击 **“任务”（Runs）** 可以看到同一作业的执行时间线、节点耗时与输出内容。若长任务等待超时，后台计算**不会**被取消，继续用原 `job_id` 查询即可。

3. 30 秒创建并接入第一个自定义应用

在 OpenSAI 中，“应用（Application）”是一份可反复按名调用的流程模板，“作业（Job）”则是它的一次具体运行。使用下面四条命令即可生成、校验、注册并运行一个纯标准库应用模板：

```bash
opensai app init my_app
opensai app validate my_app
opensai app register my_app
opensai app run my_app -i size=800000 --wait
```

> ✓ **验证通过标志**
>
> 终端返回 `SUCCEEDED` 及统计输出。打开生成的 `my_app/` 目录可以看到：

- `my_app/app.yaml`：声明业务输入（`inputs`）、可调参数旋钮（`knobs`）、执行步骤与依赖（`steps`）及默认目标（`qos`）；
- `my_app/steps/`：存放具体的 Python 步骤类。上游步骤写入 `_artifact_out` 的中间产物，会由平台自动准备到下游步骤的 `_artifact_in` 本地路径中（跨设备时自动经 QUIC 拉取）。

将该模板替换为你自己的业务逻辑后，重新执行 `validate` 和 `register` 即可完成热更新。完整开发规范见 软件使用手册 第 6 章（亦可参考内置的 `examples/apps/number_stats`）。

4. 体验目标驱动的双向自适应

内置的 `cluster_analytics` 示例将两份合成数据分片并行聚类，再合并为全局簇中心。它声明了两个允许平台自动调节的旋钮：特征维数 `dims` 与迭代次数 `iters`。先安装 `numpy` 并完成首次运行：

```bash
python -m pip install numpy
opensai app register examples/apps/cluster_analytics
opensai app run cluster_analytics -i points=30000 -i seed=1 --wait
```

开启联合放置与调参模式对比效果

默认的 `plan` 模式先完成整图设备规划，再按目标选择节点参数与核数；将 `OPENSAI_PLACEMENT` 设为 `adaptive` 后，中枢会将执行设备、CPU/GPU 目标、核数与旋钮档位纳入联合求解。按当前系统重启本地服务：

Linux / macOS：

```bash
export OPENSAI_PLACEMENT=adaptive
bash scripts/run.sh --with-web
```

Windows PowerShell：

```powershell
$env:OPENSAI_PLACEMENT="adaptive"
powershell -ExecutionPolicy Bypass -File scripts\run.ps1 -WithWeb
```

确认 `opensai devices list` 在线后，对完全相同的输入先后执行“固定高档位参数”与“3 秒截止时间目标驱动”两次调用：

```bash
# 调用 A：用 -k 显式锁定特征维数与迭代次数
opensai app run cluster_analytics -i points=30000 -i seed=1 -k dims=16 -k iters=20 --wait

# 调用 B：不锁定旋钮，交由平台按 3 秒目标自动选择配置与核数
opensai app run cluster_analytics -i points=30000 -i seed=1 --deadline 3 --wait
```

打开控制台 **“任务”（Runs）** 与 **“双向适配”（Adaptation）** 页面（或运行 `opensai explain YOUR_JOB_ID`），同时对比 **实际完成时间（`duration_s`）、业务质量分数与资源核秒占用**。

> ✦ **实用提示**
>
> 当本机算力充裕、最高档位本身即可轻松在 3 秒内完成时，平台自动保留高档位属于符合预期的最优选择；你可以通过增大 `points` 规模或收紧 `--deadline` 观察平台自动下调旋钮以优先保障时限的过程。

5. 停止服务与下一步进阶

结束本机体验时，停止中枢、工作设备与 Web 桥接进程（附加 `--all` / `-All` 一并停止由脚本拉起的便携 Redis）：

- **Linux / macOS**：`bash scripts/stop.sh --all`
- **Windows PowerShell**：`powershell -ExecutionPolicy Bypass -File scripts\stop.ps1 -All`

你的下一步目标	推荐查阅的文档与章节
将多台电脑/边缘设备连成真实集群	软件使用手册 第 3 章（多机部署与端口连通）
体验多车与路侧激光雷达协同感知	软件使用手册 第 5.6 节（`coop_perception` 与 `how2comm_perception`）
把自己的 Python / 视觉模型封装为应用	软件使用手册 第 6 章（应用包结构、旋钮与可选分支）
开启跨设备 QUIC 中继多链路传输	软件使用手册 第 9 章（`OPENSAI_MULTIPATH=1` 与路径解读）
在已有业务代码中通过 Python 调用	软件使用手册 第 11 章（`OpenSAIClient` SDK）
让 AI 智能体用自然语言操控集群	智能体接入指南 与 常见问题排查