Kronos 基于 UV 的部署：适配菜鸟教程

结合菜鸟教程（runoob）的 uv 教程逻辑，我为你梳理出**适配 Kronos 部署的、更贴合新手认知习惯的纯 uv 部署流程**，全程遵循教程的基础语法和操作逻辑，确保步骤清晰、可落地：

一、前置认知（对应教程核心概念）

uv 是新一代 Python 包/环境管理器，核心优势：

1. 兼容 pip 语法，安装依赖速度比 pip 快 10-100 倍；
2. 内置虚拟环境管理（替代 venv/conda），支持指定 Python 版本；
3. 自动处理依赖冲突，缓存下载包避免重复消耗。

二、部署环境

Ubuntu Server 24.04.4 LTS（64位）、Python 3.10+（uv 自动安装）、Kronos 源码

三、完整部署步骤（适配 runoob 教程风格）

步骤 1：安装 uv（对应教程「安装 uv」章节）

Code block

```
# 官方一键安装脚本（Linux 通用）
curl -LsSf https://astral.sh/uv/install.sh | sh

# 刷新环境变量，让系统识别 uv 命令（关键）
source ~/.bashrc

# 验证安装（教程重点：检查版本）
uv --version  # 输出 uv 0.4.x+ 即成功
```

步骤 2：创建项目目录 + 初始化 uv 环境（对应教程「uv 虚拟环境」章节）

Code block

```
# 创建 Kronos 专属目录（规范路径）
mkdir -p ~/kronos-project && cd ~/kronos-project

# 初始化 uv 虚拟环境（指定 Python 3.10.14，匹配 Kronos 要求）
# 教程核心：uv venv 替代 python -m venv，--python 指定版本（uv 自动下载）
uv venv --python 3.10.14

```

```
# 激活虚拟环境（教程重点：激活命令与 venv 一致）
source .venv/bin/activate

# 验证 Python 版本（确保环境隔离）
python --version  # 必须输出 Python 3.10.14
```

步骤 3：克隆 Kronos 源码（基础操作，教程补充）

Code block

```
# 安装 git（若未安装）
sudo apt install -y git

# 克隆源码到项目目录
git clone https://github.com/shiyu-coder/Kronos.git
cd Kronos
```

步骤 4：用 uv 安装依赖（对应教程「uv pip 用法」章节）

Code block

```
# 1. 安装项目核心依赖（uv pip 完全兼容 requirements.txt）
uv pip install -r requirements.txt

# 2. 安装 WebUI 依赖（Kronos 可视化界面）
cd webui
uv pip install -r requirements.txt

# 3. （可选）GPU 加速：安装 CUDA 版 PyTorch（教程拓展）
# 先卸载可能的 CPU 版 torch
uv pip uninstall -y torch
# 安装适配 CUDA 12.4 的 PyTorch
uv pip install torch==2.4.0 torchvision torchaudio --index-url
https://download.pytorch.org/whl/cu124

# 验证 GPU 支持（教程验证逻辑）
python -c "import torch; print(torch.cuda.is_available())"  # 输出 True 即生效
```

步骤 5：启动 Kronos Web UI（教程「运行项目」逻辑）

Code block

```
# 确保在 webui 目录 + 环境已激活
cd ~/kronos-project/Kronos/webui
source ~/kronos-project/.venv/bin/activate
```

```
4
5   # 方式 1：直接启动（测试用，对应教程「直接运行脚本」）
6   python run.py
7
8   # 方式 2：后台持久化运行（生产用，教程拓展）
9   # 安装 screen 保持进程运行
10  sudo apt install -y screen
11  screen -S kronos-webui  # 创建会话
12  python run.py  # 启动 WebUI
13  # 按 Ctrl+A+D 退出会话，进程后台运行
```

步骤 6：访问 Web UI（教程「验证运行」）

Code block

```
1   # 开放 7070 端口（Ubuntu 防火墙）
2   sudo ufw allow 7070/tcp
3   sudo ufw reload
4
5   # 访问地址（替换为服务器 IP）
6   # http://你的服务器IP:7070
```

步骤 7：模型微调（可选，对应教程「进阶用法」）

Code block

```
1   # 进入微调目录
2   cd ~/kronos-project/Kronos/finetune_csv
3   source ~/kronos-project/.venv/bin/activate
4
5   # 单 GPU/CPU 微调（教程基础版）
6   python train_sequential.py --config configs/config_ali09988_candle-5min.yaml
7
8   # 多 GPU 微调（教程拓展，DDP 模式）
9   export DIST_BACKEND=nccl
10  torchrun --standalone --nproc_per_node=2 train_sequential.py --config
    configs/config_ali09988_candle-5min.yaml
```

四、uv 常用命令速查（对应教程「常用命令」章节）

功能	uv 命令（教程核心）	说明
创建虚拟环境	`uv venv --python 3.10.14`	指定 Python 版本创建隔离环境

激活环境	`source .venv/bin/activate`	激活后所有操作仅在环境内生效
安装依赖	`uv pip install -r requirements.txt`	兼容 pip 依赖文件，速度更快
安装指定包	`uv pip install flask==2.3.3`	指定版本安装，自动解决依赖冲突
查看已安装包	`uv pip list`	替代 pip list，输出更清晰
清理缓存	`uv cache clean`	清理下载的包缓存，释放磁盘空间
退出环境	`deactivate`	退出 uv 虚拟环境

五、新手避坑（教程未提及，适配 Kronos 场景）

1. **uv 命令找不到**：执行 `echo 'export PATH="$HOME/.cargo/bin:$PATH"' >> ~/.bashrc && source ~/.bashrc` 手动添加路径；
2. **依赖安装慢**：配置国内源（教程拓展）：

```
uv pip config set global.index-url https://pypi.tuna.tsinghua.edu.cn/simple
```

3. **端口被占用**：修改 `webui/app.py` 中 `port=7070` 为其他端口（如 7071）；
4. **微调数据报错**：确保数据文件放在 `Kronos/data` 目录，包含 `open/high/low/close` 列。

总结

1. **核心逻辑**：完全遵循 runoob uv 教程的基础语法，用 uv 替代 pip/venv 完成「环境创建→依赖安装→项目运行」全流程，适配 Kronos 的 Python 3.10+ 要求；
2. **关键优势**：uv 自动安装指定版本 Python，无需手动配置系统环境，依赖安装速度远快于 pip；
3. **部署重点**：激活 uv 环境是所有操作的前提，Web UI 启动后需开放 7070 端口才能访问。

这套流程既贴合新手教程的认知逻辑，又完全满足 Kronos 部署的核心需求，从环境搭建到功能验证一步到位。

（注：文档部分内容可能由 AI 生成）