# Python项目实战 教案

周次：10

课次：2

章节内容：项目部署与维护

课时：2

## 单元教学目标

- 知识目标：掌握部署工具的使用方法，包括pip、conda等工具的基本操作原理及版本控制工具Git的核心概念。   
- 技能目标：能够独立完成Python项目的部署流程，包括环境配置、依赖安装及服务器部署操作。   
- 素养目标：具备良好的团队协作意识，能够在项目部署过程中主动沟通并承担责任，确保交付质量。

## 教学重点

• 使用Git进行版本控制和代码管理   
• 部署流程的标准化（构建、测试、部署）   
• 基础部署工具的使用（如Docker、nginx、部署平台）

## 教学难点

• 项目部署过程中涉及的环境配置（如依赖库安装、环境变量设置）容易因不同操作系统或服务器配置差异导致问题，学生可能难以掌握统一的部署规范   
• 如何将Python项目部署到服务器并确保持续运行（如服务启动脚本编写、进程管理）是难点，学生可能对系统级操作（如使用systemd、init.d）不够熟悉

## 教学活动

- 新课导入【15分钟】：通过真实案例引发兴趣，教师展示电商平台部署失败的视频片段，提问"为什么系统无法启动？"引导学生思考部署流程，结合故事导入法，激发学习动机。   
  
- 讲授新课【150分钟】：   
 1. \*\*部署工具介绍（20分钟）\*\*：   
 - 讲授Docker容器化技术，通过案例分析法讲解容器化原理   
 - 实操演示Docker镜像构建流程，使用角色扮演法分组完成镜像构建任务   
 2. \*\*环境配置管理（25分钟）\*\*：   
 - 采用案例分析法讲解虚拟环境搭建步骤   
 - 组内协作完成Linux系统环境配置任务，使用讨论法解决配置冲突问题   
 3. \*\*版本控制与CI/CD（30分钟）\*\*：   
 - 通过Git版本控制原理的演示，结合头脑风暴法讨论版本管理实践   
 - 模拟CI/CD流水线流程，使用游戏法设计自动化部署脚本测试环节   
 4. \*\*项目部署策略（25分钟）\*\*：   
 - 用案例分析法解析高可用部署方案   
 - 分组完成部署方案设计，使用练习法验证方案可行性   
  
- 巩固练习【30分钟】：   
 1. \*\*基础部署实践（15分钟）\*\*：   
 - 分层任务：完成Docker镜像构建、Linux环境配置、Git版本控制基础操作   
 - 使用小组竞赛形式进行任务验证   
 2. \*\*高级部署挑战（15分钟）\*\*：   
 - 设计自动化部署脚本编写任务，要求包含环境变量配置和错误处理逻辑   
 - 使用角色扮演法模拟运维工程师角色完成部署验证   
 3. \*\*故障排查演练（10分钟）\*\*：   
 - 提供典型部署错误案例，通过讨论法分析问题根源   
 - 实操演练使用Ansible进行部署状态检查   
  
- 归纳总结【10分钟】：   
 - 教师用思维导图形式总结部署核心要素   
 - 学生分组复述部署流程，教师补充关键注意事项   
 - 采用头脑风暴法讨论部署实践中的常见问题及解决方案

## 教学资源

• 教学设备：   
 - 桌面计算机（配备Python开发环境）   
 - 服务器/虚拟机（支持Linux/Windows系统）   
 - 网络设备（路由器、交换机，用于网络部署演示）   
 - 终端模拟器（Windows Terminal/Mac Terminal）   
 - 操作系统（Windows 10/11、Linux发行版）   
  
• 工具推荐：   
 - 版本控制工具：Git（需安装Git客户端）   
 - 部署工具：Docker（安装Docker Engine）、Ansible（安装Ansible Playbook）   
 - 云服务：AWS EC2（演示服务器部署）、Heroku（简化部署）   
 - 虚拟化工具：VMware/VirtualBox（用于实验环境）   
 - 项目管理：Jira（项目管理示例）   
  
• 参考资料与网站：   
 - 书籍：   
 - 《Python Web Development with Flask》（实战项目部署）   
 - 《Python Crash Course》（基础概念结合部署实践）   
 - 在线资源：   
 - Python官方文档（https://docs.python.org/zh-cn/3/）   
 - GitHub开源项目（如Docker官方示例、Ansible Playbook）   
 - Real Python（Python部署与运维教程）   
 - 社区与平台：   
 - Stack Overflow（解决部署问题）   
 - Reddit r/learnpython（项目实战案例）   
 - Docker官方文档（https://docs.docker.com/）   
 - Kubernetes官方文档（https://kubernetes.io/docs/）   
  
• 项目案例：   
 - 使用Docker容器化部署Web应用（如Flask/ Django项目）   
 - 通过Ansible实现自动化服务器配置（如Nginx/nginx.conf部署）   
 - 利用GitHub Actions进行持续集成（如自动部署到Heroku）

## 教学反思

• 教学效果方面，学生对项目部署与维护的理解有所提升，能够初步掌握基本的部署流程和维护方法，但部分学生在实际操作中仍存在理解偏差，如对环境配置、依赖管理等环节掌握不牢。  
  
• 学生反馈方面，多数学生认为课程内容实用，能够将所学知识应用到实际项目中，但部分学生反映理论与实践结合不够紧密，缺乏足够的实践指导和案例支持。  
  
• 改进建议包括：增加更多实际案例和项目演练，强化理论与实践的结合；引入更多工具和方法（如Docker、CI/CD等）提升学生的实战能力；加强课堂互动，鼓励学生提问和讨论，提高学习积极性。

## 教学评价

• 过程性评价   
• 课堂参与度：观察学生在课堂讨论、代码讲解中的主动性和参与情况（如提问、协作、代码展示）   
• 代码规范性：通过代码检查工具（如PEP8）或教师抽查评估代码格式、注释、可读性   
• 项目进度：记录学生是否按时完成任务（如部署脚本编写、版本控制提交）   
• 团队协作能力：通过小组任务分配、合作讨论记录评估成员分工与沟通效率   
  
• 结果性评价   
• 项目部署正确性：评估学生是否完成部署流程（如使用Git版本控制、Docker容器化、CI/CD工具配置）   
• 技术应用深度：分析学生是否合理运用Python工具（如Flask/ Django框架、自动化脚本、API接口）   
• 技术文档完整性：评审项目文档（如README、部署指南、故障排查手册）的完整性与规范性   
• 问题解决能力：通过部署过程中出现的故障案例，评估学生是否能独立排查并修复问题   
  
• 评价方式   
• 过程性：课堂观察记录、代码提交反馈、小组协作日志   
• 结果性：项目成果提交、部署演示、技术文档评审、教师评分表   
  
• 评分标准   
• 过程性：课堂参与（20%）+ 代码规范（30%）+ 项目进度（20%）+ 团队协作（20%）   
• 结果性：部署正确性（30%）+ 技术应用深度（25%）+ 文档完整性（15%）+ 问题解决能力（10%）