Distributed Task Automation with Python Celery

Duration

2 days

Description

This comprehensive course is designed for computer programming professionals who want to delve into the world of Distributed Task Automation. It offers a deep dive into tools like Celery and RabbitMQ, and covers everything from setting up a development environment with Visual Studio Code, Python, and Docker, to understanding containerization and remote data storage. You'll learn how to run PostgreSQL and RabbitMQ in containers, and get hands-on experience with Celery, from the basics to advanced concepts. The course wraps up with practical lessons on scaling and deploying Celery in a production environment. Whether you're a seasoned programmer or just starting out, this course will equip you with the skills and knowledge to effectively automate distributed tasks. Generally, this course can be customized to include distributed task automation examples relevant to your work domain.

Objectives

- Understand the concept of Distributed Task Automation and learn about tools like Celery and RabbitMQ.
- Set up a development environment with Visual Studio Code, Python, and Docker.
- Learn about containerization using Docker and Docker Compose.
- Understand remote data storage and how to run PostgreSQL in a container.
- Learn about Message Brokers and how to run RabbitMQ in a container.
- Gain a solid understanding of Celery basics, including task definition, running tasks, and worker management.
- Delve into advanced Celery concepts like task serialization, routing, and scheduling periodic tasks.
- Learn how to scale and deploy Celery in a production environment, including load balancing, high availability, and performance tuning.

Prerequisites

All students should have taken the Python Task Automation course or have significant experience with the topics covered in the Python Task Automation course. Experience with RabbitMQ and PostgreSQL is helpful but not required.

Training 4 Programmers

Training Materials

All students receive comprehensive courseware covering all topics in the course. The instructor distributes courseware via GitHub. The courseware includes documentation and extensive code samples. Students practice the topics covered through challenging hands-on lab exercises.

Software Requirements

Students will need a free, personal GitHub account to access the courseware. All students will need a modern web browser such as Google Chrome. Student machines will need a text editor like Visual Studio Code, the latest Python version, Docker Desktop, PanDoc, and OpenOffice. Students will need permission to install NPM and PyPi packages as well as the ability to download Docker images. Preconfigured student virtual machines can provided upon request.

Outline

- Overview of Distributed Task Automation
    - What is Distributed Task Automation?
    - Overview of Celery and its Features
    - Celery compared to Faust
    - What is Message Broker?
    - What is RabbitMQ?
    - RabbitMQ + PostgreSQL compared Kafka + Zookeeper
- Development Environment
    - Configure Visual Studio Code for Python Script Programming
    - Python Code Linting & Reformatting with Ruff & MyPy
    - Debugging Python Scripts with Visual Studio Code
    - Docker Desktop
- Containerization
    - What is a Container?
    - What is Docker?
    - What is Docker Hub?
    - Images and Containers
    - Create an Image with Dockerfile
    - Run Containers
    - Configure Containers with Environment Variables
    - Docker Compose
    - Docker Compose Networking
    - Docker Compose Volume
- Remote Data Storage

- What is Remote Data Storage?
- Running a Data Storage in a Container
- Running PostgreSQL in a Container
- Running PostgreSQL Client Tool in a Container
- Configure with Docker Compose
- Read/Write Data to PostgreSQL with Python SQLAlchemy

- Remote Message Broker
  - What is a Message Broker?
  - Running a Message Broker in a Container
  - Running RabbitMQ in a Container
  - Running RabbitMQ Client Tool in a Container
  - Configure with Docker Compose
  - Read/Write Data to RabbitMQ with Python and Pika

- Celery Basics
  - Installing Celery and its dependencies
  - Setting up a simple Celery project
  - Defining and running tasks
  - Task decorators and options
  - Passing arguments and results between tasks
  - Task retries and error handling
  - Monitoring and managing Celery workers

- Advanced Celery Concepts
  - Task serialization and message brokers
  - Task routing and prioritization
  - Task result backends
  - Grouping and chaining tasks
  - Scheduling periodic tasks with Celery beat

- Scaling and Deployment
  - Load balancing tasks with multiple workers
  - Deploying Celery in a production environment
  - Configuring Celery for high availability
  - Best practices for handling long-running tasks
  - Monitoring and performance tuning

- Conclusion
  - Summary of Key Concepts
  - Q&A

–   Further Resources and Next Steps