Training 4 Programmers

To discuss this course and customizations:
Call: +1 434-509-5680 or Email: sales@training4programmers.com

Distributed Task Automation with Python Faust

Duration

2 days

Description

This comprehensive course provides an in-depth exploration of Distributed Task Automation using Python Faust and Kafka. It begins with an overview of Distributed Task Automation and a detailed study of Python Faust and Kafka. The course then delves into the practical aspects of setting up a Python development environment, containerization using Docker, and managing Kafka clusters. Learners will gain a deep understanding of Python Faust, including stream processing, state management, and fault tolerance. The course also covers monitoring and managing Kafka and Faust applications, deploying them in production, and optimizing their performance. The final section of the course guides learners through the implementation of a real-time data pipeline with Faust and Kafka. Generally, this course can be customized to include distributed task automation examples relevant to your work domain.

Objectives

- Understand the concept and application of Distributed Task Automation.
- Set up and configure a Python development environment for script programming.
- Learn the basics of containerization and how to use Docker for creating and running containers.
- Gain in-depth knowledge of Kafka, its architecture, and how to set up a Kafka cluster.
- Master the basics and advanced concepts of Python Faust, including agents, stream processing, state management, and fault tolerance.
- Learn how to monitor and manage Kafka and Faust applications, including error handling and retry logic.
- Understand the best practices for deploying Kafka and Faust in production, ensuring high availability and optimizing performance.
- Implement a real-time data pipeline with Faust and Kafka.

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 Kafka and Zookeeper is helpful but not required.

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 Python Faust
  - Faust compared to Celery
  - What is Streaming?
  - What is Kafka?
  - What is Zookeeper?
  - Kafka + Zookeeper compared RabbitMQ + PostgreSQL
- 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

- Kafka Basics
  - What is Kafka?
  - Kafka Architecture and Components
  - Setting Up a Kafka Cluster
  - Running Kafka in a Container
    - Running Kafka and Zookeeper in Docker
    - Configuring Kafka with Docker Compose
  - Kafka Producers and Consumers
    - Writing Data to Kafka with Python
    - Reading Data from Kafka with Python
- Faust Basics
  - Introduction to Python Faust
    - Differences between Faust and Celery
  - Installing Faust and Dependencies
  - Setting Up a Simple Faust Project
    - Creating a Faust App
    - Defining and Running Agents
  - Faust Agents
    - Creating Agents for Stream Processing
    - Processing Streams in Real-Time
  - Handling Kafka Topics with Faust
    - Reading from Kafka Topics
    - Writing to Kafka Topics
- Advanced Faust Concepts
  - Stream Processing and Windowing
    - Tumbling, Hopping, and Sliding Windows
  - State Management in Faust
    - Using Faust Tables for State Management
    - Persisting State Beyond Kafka
  - Fault Tolerance and Recovery
    - Checkpointing and Replayability
  - Aggregating and Joining Streams
    - Combining Multiple Streams
    - Aggregations and Reductions
  - Scaling Faust Applications
    - Parallelism and Partitioning in Kafka

    - Running Multiple Faust Workers
- Monitoring and Management
  - Monitoring Kafka and Faust
    - Using Kafka Monitoring Tools (e.g., Kafka Manager, Confluent Control Center)
    - Logging and Metrics in Faust
  - Handling Errors and Retries
    - Configuring Error Handling in Faust
    - Implementing Retry Logic
- Scaling and Deployment
  - Deploying Kafka and Faust in Production
    - Best Practices for Kafka Cluster Deployment
    - Deploying Faust Apps with Docker and Kubernetes
  - High Availability and Fault Tolerance
    - Configuring Kafka for High Availability
    - Ensuring High Availability in Faust Applications
  - Performance Tuning
    - Kafka Performance Tuning
    - Optimizing Faust Performance
  - Implementing a Real-Time Data Pipeline with Faust and Kafka
- Conclusion
  - Summary of Key Concepts
  - Q&A
  - Further Resources and Next Steps