Cloud Contraptions
TRAINING & CONSULTING

To discuss this course and customizations:
Call: +1 434-509-6890 or Email: sales@cloudcontraptions.com

Distributed Task Automation with Python, Kafka, and Celery

Class Duration

14 hours of live training delivered over 2 days.

Student Prerequisites

- Proficiency in Python programming, including experience with Python 3.x
- Familiarity with basic concepts of distributed systems and task automation
- Experience with Docker and containerization concepts is beneficial but not required
- Basic understanding of message brokers and task queues is helpful but not required

Target Audience

This course is designed for experienced Python developers who want to build scalable, distributed task pipelines with Apache Kafka and Celery. All students should have taken Task Automation with Python or have significant experience with the topics it covers.

Description

This course equips Python developers with practical expertise in distributed task automation using Apache Kafka 4.x (KRaft mode) and Celery 5.6, building scalable, batch-oriented data pipelines where Kafka serves as the event ingress and Celery, backed by Redis, distributes work to a pool of containerized workers. Students start with the fundamentals of task automation and the trade-offs between streaming and batch processing, then move into hands-on environment setup with Docker. The course teaches Kafka deeply enough to use it well (brokers, topics, partitions, consumers, delivery semantics, idempotent producers, and transactions for exactly-once processing) and then moves to Celery for distributed task processing, with honest treatment of trade-offs such as Redis vs RabbitMQ as broker and when not to use Kafka-to-Celery. The two are then integrated into

the central pattern of the course: a Kafka consumer that reads events in batches and dispatches them as Celery tasks, with DLQ topics and idempotency keys handling failures. Monitoring (Kafka UI/Redpanda Console, Flower, Prometheus, Grafana, OpenTelemetry) and deployment (KEDA-driven autoscaling, HA Redis/Kafka, Kubernetes) round out the course, leaving students with immediately applicable skills for automating complex workflows at scale.

Learning Outcomes

- Understand the concept and application of distributed task automation, compare streaming and batch processing trade-offs, and choose the right model for a workload.
- Set up and configure a Python development environment for distributed task programming.
- Learn the basics of containerization and how to use Docker for creating and running containers.
- Gain in-depth knowledge of Kafka 4.x in KRaft mode: brokers, topics, partitions, offsets, consumers, consumer groups, and delivery semantics including idempotent producers and transactions for exactly-once processing.
- Use confluent-kafka to produce and consume Kafka messages in batches, and master Celery with Redis as the broker: defining tasks, running workers, chains, groups, chords, and periodic tasks.
- Integrate Kafka and Celery: read events from Kafka in batches and dispatch them as Celery tasks with proper offset/acknowledgment coordination, DLQ handling, and idempotency.
- Monitor and manage Kafka and Celery applications with Kafka UI, Flower, Prometheus, Grafana, and OpenTelemetry tracing.
- Deploy Kafka and Celery in production with KEDA-driven autoscaling and high-availability patterns.

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. All students receive a downloadable MP4 recording of the training.

To discuss this course and customizations:
Call: +1 434-509-6890 or Email: sales@cloudcontraptions.com

Software Requirements

- Free, personal GitHub account to access the courseware
- Modern web browser such as Google Chrome
- Text editor like Visual Studio Code
- Python 3.11+ and Docker Desktop (Kafka, Redis, and Celery workers all run in Docker)
- Permission to install PyPI packages and the ability to download Docker images
- Preconfigured student virtual machines can be provided upon request

Training Topics

Overview of Distributed Task Automation

- What is distributed task automation?
- Streaming vs. batch processing: trade-offs and when to use each
- Overview of Apache Kafka 4.x (KRaft mode)
- Overview of Celery
- Architecture: Kafka as event ingress, Celery for batch task processing
- When NOT to use Kafka-to-Celery (use stream processing or stateless consumers instead)
- Comparison with alternatives (RabbitMQ, Redis Streams, Kafka Streams)

Development Environment

- Configure Visual Studio Code for Python script programming
- Python code linting and reformatting with Ruff and MyPy
- Debugging Python scripts with Visual Studio Code
- Docker Desktop

Containerization with Docker

- Containers, Docker, and Docker Hub essentials
- Building images with Dockerfile
- Running containers and configuring with environment variables
- Docker Compose: networking and volumes

Kafka Fundamentals

- Brokers, topics, partitions, and offsets
- KRaft mode (no ZooKeeper): controllers, quorum, and metadata log

- Kafka 4.x deployment model: combined mode (dev) vs isolated mode (production)
- Producers and consumers
- Consumer groups and rebalancing
- Delivery semantics: at-most-once, at-least-once, exactly-once
- Idempotent producers and transactions (enable.idempotence, acks=all)
- Consumer isolation levels (read_committed for EOS)
- Schema management with Schema Registry (Avro / JSON Schema / Protobuf, brief overview)
- Running Kafka in Docker
- Topic management with command-line tools

Working with Kafka in Python

- Using confluent-kafka (the recommended client; native AIOProducer for asyncio workloads)
- Producing messages
- Consuming messages
- Batch consumption patterns
- Manual vs. automatic offset commits
- Producing idempotent and transactional messages from Python
- Error handling in producers and consumers

Celery Fundamentals

- What is Celery? Celery 5.6 (Recovery release) overview
- Choosing a broker: Redis vs RabbitMQ - trade-offs (visibility timeout, acks, persistence)
- Configuring Redis as the Celery broker (with honest caveats)
- Defining tasks (with Pydantic task models, Celery 5.5+)
- Running Celery workers (including soft shutdown)
- Running Celery with Docker
- Result backends (and when to skip them for high-throughput fire-and-forget pipelines)

Batch Task Patterns with Celery

- Task chains
- Task groups and chords
- Periodic tasks with Celery Beat

To discuss this course and customizations:
Call: +1 434-509-6890 or Email: sales@cloudcontraptions.com

- Batching strategies for bulk workloads
- Error handling and retries

Integrating Kafka with Celery

- The pattern: Kafka consumer dispatches batches to Celery
- Building a batched Kafka consumer
- Submitting tasks to Celery from the consumer
- Coordinating Kafka offsets with Celery task acknowledgment
- Handling failures and retries end-to-end
- Dead-letter topics (DLQs) and poison message handling
- Idempotency keys at the Celery task layer
- Anti-pattern: running the Kafka consumer loop as a Celery task

Monitoring and Management

- Monitoring Kafka with Kafka UI and Redpanda Console
- Monitoring Celery with Flower
- Prometheus + Grafana with celery-exporter and Kafka Exporter
- OpenTelemetry tracing across the Kafka-to-Celery boundary
- Logging and metrics
- Common failure modes and how to detect them

Deployment and Scaling

- Scaling Celery workers (concurrency modes: prefork, threads, gevent)
- Scaling Kafka consumers via consumer groups
- KEDA autoscaling: ScaledObject on Redis queue length and Kafka lag
- High availability: Kafka replication, Redis Sentinel/Cluster, quorum queues
- From Docker Compose to Kubernetes

