Cloud Contraptions
TRAINING & CONSULTING

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

Distributed Task Processing with Python

Class Duration

14-21 hours of live training delivered over 2-3 days. The two-day core covers foundations, Celery, and Kafka. The optional stream processing module and extra production topics add a third day.

Student Prerequisites

- Proficiency in Python programming
- 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

Experienced Python developers, backend engineers, and data engineers who run background jobs, scheduled work, and event-driven pipelines across many machines. Students should have taken Task Automation with Python or have similar experience. Teams that need scheduled orchestration on top of these tools can continue with Apache Airflow 3 for Developers.

Description

This course teaches Python developers to build and run distributed task pipelines, starting with delivery semantics, idempotency, and retries. The Celery module covers tasks, routing, canvas workflows, and Celery Beat, with RabbitMQ or Redis as the broker. The Kafka module covers Apache Kafka 4 in KRaft mode, consumer groups, and transactions from Python. Students then connect the two, with a Kafka consumer that dispatches batches of events as Celery tasks. An optional module covers stream processing with faust-streaming and Quix Streams, and the course ends with monitoring and Kubernetes deployment.

Copyright 2026 Cloud Contraptions LLC – All Rights Reserved

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

Learning Outcomes

- Choose between task queues, message brokers, event logs, and stream processors.
- Design for delivery semantics, idempotency, retries, and dead-letter handling.
- Run RabbitMQ, Redis, Kafka, and PostgreSQL with Docker Compose.
- Build Celery tasks, routes, and canvas workflows on RabbitMQ or Redis.
- Schedule periodic tasks with Celery Beat.
- Explain Kafka's KRaft architecture, partitions, consumer groups, and share groups.
- Produce and consume Kafka messages from Python with confluent-kafka.
- Dispatch batches of Kafka events as Celery tasks with correct offset handling.
- Build stateful stream processors with faust-streaming or Quix Streams.
- Monitor and autoscale task pipelines with Flower, Prometheus, and KEDA.

Training Materials

All students receive comprehensive courseware covering all topics in the course. Courseware is distributed via GitHub in the form of 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.

Software Requirements

- Free, personal GitHub account to access the courseware
- Modern web browser such as Google Chrome
- Text editor like Visual Studio Code
- Python 3.12 or later
- Docker Desktop or Podman to run brokers, Kafka, and workers in containers
- Courseware targets Celery 5.7, Apache Kafka 4.4 in KRaft mode, and faust-streaming 0.15
- Permission to install PyPI packages and the ability to download container images
- Preconfigured student virtual machines can be provided upon request

Copyright 2026 Cloud Contraptions LLC – All Rights Reserved

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

Training Topics

Foundations of Distributed Task Processing

- What is distributed task processing?
- Task queues and message brokers
- Event logs and stream processors
- Streaming vs. batch processing
- At-most-once and at-least-once delivery
- Exactly-once semantics
- Idempotency, retries, and poison messages
- Choosing a stack

Development Environment and Containers

- Configure Visual Studio Code for Python development
- Linting and formatting with Ruff
- Type checking with mypy
- Debugging Python workers with Visual Studio Code
- Containers and images
- Building images with a Dockerfile
- Docker Compose services and networks
- Volumes and health checks
- Running brokers and databases with Compose

Celery Fundamentals

- Configuring a Celery application
- RabbitMQ vs. Redis as a broker
- Acknowledgements and visibility timeout
- Broker persistence
- Defining tasks
- Typed task payloads with Pydantic
- Calling tasks with delay and apply_async
- Signatures
- Result backends
- Fire-and-forget tasks
- Running and managing workers
- Worker soft shutdown

Copyright 2026 Cloud Contraptions LLC – All Rights Reserved

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

Celery Task Design and Workflows

- Idempotent task design
- Retries, retry backoff, and time limits
- Late acknowledgement with acks_late
- Redelivery
- Queues, exchanges, and routing keys
- Routing tasks to dedicated queues and workers
- RabbitMQ quorum queues and delayed delivery
- Canvas chains, groups, and chords
- Scheduling periodic tasks with Celery Beat

Kafka Fundamentals (KRaft Mode)

- Brokers, topics, and partitions
- Offsets and retention
- KRaft controllers and quorum
- The metadata log
- Combined mode vs. dedicated controllers
- Producers and consumers
- Consumer groups
- The consumer rebalance protocol
- Idempotent producers
- Kafka transactions
- The read_committed isolation level
- Share groups for queue-style consumption
- Schema Registry overview
- Topic management with command-line tools

Working with Kafka in Python

- The confluent-kafka client
- The asyncio producer
- Producing and consuming messages
- Batch consumption patterns
- Manual vs. automatic offset commits
- Producing transactional messages from Python
- Error handling in producers and consumers
- The preview ShareConsumer API

Copyright 2026 Cloud Contraptions LLC – All Rights Reserved

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

Integrating Kafka with Celery

- Dispatching Kafka batches to Celery
- Building a batched Kafka consumer
- Coordinating Kafka offsets with Celery task acknowledgement
- Handling failures and retries
- Dead-letter topics and poison message handling
- Idempotency keys in Celery tasks
- Avoiding consumer loops inside Celery tasks

Stream Processing with Python (Optional Module)

- When stream processing beats task queues
- faust-streaming, the community fork of Faust
- faust-streaming maintenance and API stability
- Faust apps and agents
- Streams and topics
- Tables and windowing
- Changelog-backed state
- Fault tolerance and rebalancing
- Scaling Faust workers
- Quix Streams as a Python alternative
- When to hand off to Kafka Streams or Apache Flink

Monitoring and Observability

- Monitoring Celery with Flower
- The RabbitMQ management UI
- Kafka UI and Redpanda Console
- Prometheus and Grafana dashboards
- celery-exporter and Kafka Exporter
- Tracing from Kafka to Celery with OpenTelemetry
- Detecting common failure modes

Scaling and Production Deployment

- Prefork, threads, and gevent pools
- Prefetch settings
- Scaling Kafka consumers
- KEDA autoscaling on queue length and consumer lag
- High availability for brokers

Copyright 2026 Cloud Contraptions LLC – All Rights Reserved



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

- Redis Sentinel and Kafka replication
- From Docker Compose to Kubernetes
- Performance tuning and capacity planning

Copyright 2026 Cloud Contraptions LLC – All Rights Reserved