Cloud Contraptions
TRAINING & CONSULTING

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

Migrating Flask APIs to FastAPI

Class Duration

7-14 hours of live training delivered over 1-2 days, depending on the depth of coverage and the size of the APIs being migrated.

Student Prerequisites

- Experience building or maintaining REST APIs with Flask, Flask-RESTX, or Flask-RESTful
- Comfort with Python type hints is helpful but not required
- Familiarity with pytest

Target Audience

Python teams that maintain Flask or Flask-RESTX services and want to move them to FastAPI without a big-bang rewrite. Teams starting a new service should take Build REST APIs with Python and FastAPI instead.

Description

This course moves existing Flask and Flask-RESTX APIs to FastAPI one route at a time. Flask-RESTful has had no release since 2023, and Flask-RESTX has had one. Students map blueprints, namespaces, and models to routers and Pydantic models, then port authentication, errors, and database access. They keep the API contract intact and run both apps side by side during cutover. AI coding agents do the mechanical conversion, checked by contract tests and OpenAPI diffs.

Learning Outcomes

- Plan an incremental, route-by-route migration from Flask.
- Map Flask routes and blueprints to FastAPI routers.
- Convert Flask-RESTX namespaces and models to Pydantic models.
- Preserve the API contract with OpenAPI diffs and contract tests.
- Port validation, errors, and authentication to FastAPI dependencies.
- Choose between async and sync path operations.
- Test both apps with one contract test suite.
- Run Flask and FastAPI side by side in one ASGI app.

Copyright 2026 Cloud Contraptions LLC – All Rights Reserved

- Verify AI-generated conversions with tests and OpenAPI diffs.

Training Materials

Comprehensive courseware is distributed online at the start of class. All students receive a downloadable MP4 recording of the training.

Software Requirements

Students will need a free, personal GitHub account to access the courseware. Students will need permission to install Python 3.11 or later and Visual Studio Code on their computers, along with Python packages (Flask 3, Flask-RESTX, FastAPI, Pydantic 2, SQLAlchemy 2.x, and pytest) and Visual Studio Code extensions. Students will also need access to an AI coding agent such as GitHub Copilot, Claude Code, or Cursor. If students are unable to configure a local environment, a cloud-based environment can be provided.

Training Topics

Planning the Migration

- Why teams leave Flask-RESTX
- WSGI vs. ASGI
- Inventorying routes and extensions
- Decorators and middleware
- Strangler fig vs. full rewrite
- Defining done

Capturing the Existing Contract

- Exporting the Flask-RESTX Swagger 2.0 document
- Status codes, error bodies, and headers
- Contract tests for both implementations
- Recording and replaying traffic

Routes, Blueprints, and Routers

- Flask routes to path operations
- Blueprints to `APIRouter`
- Prefixes, tags, and shared dependencies
- Path converters to typed parameters
- `request.args` to query parameters
- `request.get_json` to request bodies

Copyright 2026 Cloud Contraptions LLC – All Rights Reserved

- Response helpers and redirects
- `url_for` equivalents
- Application factories and `lifespan`

Flask-RESTX Namespaces and Models to Pydantic

- Namespaces and `Resource` classes
- `api.model` and `fields` to Pydantic
- `marshal_with` to `response_model`
- `reqparse` and `@api.expect`
- Nested models and lists
- Polymorphic payloads with discriminated unions
- Converting Marshmallow schemas
- Compatible validation error formats

OpenAPI Generation

- Swagger 2.0 to OpenAPI 3.1
- Tags, summaries, and examples
- Documenting error responses
- Diffing OpenAPI documents
- Client SDK generation
- API versioning

Authentication and Authorization

- Flask-Login to security dependencies
- Flask-JWT-Extended to bearer tokens
- Custom auth decorators
- OAuth2 scopes and API keys
- Role checks
- CORS and security headers
- Sharing sessions across both apps

Async and Sync

- How FastAPI runs `def` and `async def`
- When to stay synchronous
- Removing blocking calls
- Flask-SQLAlchemy to SQLAlchemy sessions
- Sync and async session dependencies
- BackgroundTasks versus Celery

Copyright 2026 Cloud Contraptions LLC – All Rights Reserved

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

Errors, Middleware, and Configuration

- `errorhandler` and `abort` to `HTTPException`
- Exception handlers
- `before_request` hooks to middleware
- `after_request` hooks
- Flask's `g` to dependency injection
- Flask config to pydantic-settings
- Logging, tracing, and metrics across cutover

Testing the Migration

- Flask test client to `TestClient`
- httpx `AsyncClient`
- Fixtures and dependency overrides
- One contract suite, two apps
- Database isolation in tests
- Performance and load testing

Running Flask and FastAPI Side by Side

- Mounting Flask with `a2wsgi`
- Moving traffic route by route
- Routing at the proxy or gateway
- Feature flags and rollback
- Uvicorn in containers
- Health checks
- Decommissioning Flask

Using AI Coding Agents for the Conversion

- Where agents help
- Migration conventions and worked examples
- One blueprint per change
- Contract tests and OpenAPI diffs
- Type checking and linting
- Silent behavior changes
- Human ownership of auth and data access

Copyright 2026 Cloud Contraptions LLC – All Rights Reserved