# **🚀 COMPLETE DEPLOYMENT GUIDE**

## **From Local Testing → LangSmith UI → Pre-commit → PR → Deployment**

# **📍 PHASE 1: FIX CLIENT INTEGRATION**

## **✅ STEP 1.1: Fix Graph Export Issues**

### **Navigate to your project:**

cd app/backend

### **Fix app/my\_agent/init.py:**

"""

My Agent module initialization

"""

# ✅ CRITICAL: Export graph for client's agent\_rest.py

from app.my\_agent.agent import graph, process\_search\_request

from app.my\_agent.agents.search\_summarize\_agent import SearchAndSummarizeAgent

from app.my\_agent.utils.state\_models import SearchAgentState, SearchWorkflowStatus

\_\_all\_\_ = [

'graph', # ← CLIENT'S agent\_rest.py imports this

'process\_search\_request',

'SearchAndSummarizeAgent',

'SearchAgentState',

'SearchWorkflowStatus'

]

### **Ensure app/my\_agent/agent.py ends with:**

# =============================================================================

# CLIENT INTEGRATION + LANGSMITH

# =============================================================================

# Export the graph (REQUIRED by client's agent\_rest.py)

graph = create\_search\_summarize\_workflow()

# Demo function for LangSmith UI testing

async def demo\_search\_langsmith():

"""Demo function optimized for LangSmith UI"""

demo\_input = {

"verification\_needed": True,

"geographic\_region": "IT",

"firstName": "Paolo",

"lastName": "Corvisieri",

"workplaceName": "DISTRETTO SANITARIO FIUMICINO",

"address": "Velletri",

"specialtyCode": "18"

}

return await process\_search\_request(demo\_input)

if \_\_name\_\_ == "\_\_main\_\_":

asyncio.run(main())

## **✅ STEP 1.2: Test Client Integration**

### **Test that client's code can import your graph:**

cd app/backend

python -c "from app.my\_agent import graph; print('✅ SUCCESS: Client can import your graph!')"

**❌ If this fails:** Fix your \_\_init\_\_.py or agent.py until it works **✅ If this succeeds:** Continue to next step

# **📍 PHASE 2: LANGSMITH UI LOCAL TESTING**

## **✅ STEP 2.1: Install LangSmith Tools**

cd app/backend

pip install langgraph-cli langsmith

## **✅ STEP 2.2: Get Client's LangSmith Credentials**

**Ask client for:**

* LangSmith API Key
* LangSmith Project Name
* LangSmith Endpoint URL

## **✅ STEP 2.3: Configure Environment Variables**

### **Copy client's .env.example to .env:**

cp .env.example .env

### **Add to app/backend/.env:**

# CLIENT'S LANGSMITH CREDENTIALS (get from client)

LANGCHAIN\_TRACING\_V2=true

LANGCHAIN\_ENDPOINT=client\_provided\_langsmith\_url

LANGCHAIN\_API\_KEY=client\_provided\_api\_key

LANGCHAIN\_PROJECT=client\_provided\_project\_name

# YOUR AZURE OPENAI (existing)

AZURE\_OPENAI\_API\_KEY=your\_existing\_key

AZURE\_OPENAI\_ENDPOINT=your\_existing\_endpoint

AZURE\_OPENAI\_API\_VERSION=2024-02-15-preview

AZURE\_OPENAI\_DEPLOYMENT\_NAME=gpt-4

## **✅ STEP 2.4: Update langgraph.json**

### **Edit app/backend/langgraph.json:**

{

"dependencies": ["."],

"graphs": {

"search\_agent": "./app/my\_agent/agent.py:graph"

},

"env": ".env",

"python\_version": "3.12"

}

## **✅ STEP 2.5: Test LangSmith Connection**

cd app/backend

python -c "

import os

from langsmith import Client

client = Client()

print('✅ LangSmith connection successful!')

print('Project:', os.getenv('LANGCHAIN\_PROJECT'))

"

## **✅ STEP 2.6: Start LangSmith UI**

cd app/backend

langgraph dev

**Expected Output:**

Starting LangGraph API server...

- Local LangGraph API: http://localhost:8123

- LangGraph Studio UI: http://localhost:8123

## **✅ STEP 2.7: Test in LangSmith UI**

### **Open Browser:**

http://localhost:8123

### **Verify:**

* [ ] You see "search\_agent" in graph list
* [ ] Graph shows 4 nodes (select\_tools → execute\_search → summarize\_results → END)
* [ ] You can click on search\_agent

### **Test with Sample Input:**

{

"search\_input": {

"verification\_needed": true,

"geographic\_region": "IT",

"firstName": "Paolo",

"lastName": "Corvisieri",

"workplaceName": "DISTRETTO SANITARIO FIUMICINO",

"address": "Velletri",

"specialtyCode": "18"

},

"vr\_data": {

"validation.id": 123456,

"individual.firstName": "Paolo",

"individual.lastName": "Corvisieri"

}

}

### **Expected Results:**

* [ ] Workflow executes without errors
* [ ] You see messages with emojis flowing
* [ ] Final summary appears with confidence score
* [ ] Traces appear in client's LangSmith (if client checks)

**❌ If UI doesn't work:** Fix graph export and restart **✅ If UI works:** Continue to pre-commit phase

# **📍 PHASE 3: PRE-COMMIT & CODE QUALITY**

## **✅ STEP 3.1: Install Pre-commit Tools**

cd app/backend

pip install pre-commit black ruff mypy isort pytest pytest-asyncio

## **✅ STEP 3.2: Install Client's Pre-commit Hooks**

cd app/backend

pre-commit install

## **✅ STEP 3.3: Run Individual Tools First (Optional Check)**

### **Check formatting issues:**

black --check app/my\_agent/

ruff check app/my\_agent/

isort --check-only app/my\_agent/

mypy app/my\_agent/

## **✅ STEP 3.4: Auto-Fix Issues**

# Auto-format code

black app/my\_agent/

# Auto-fix linting

ruff check --fix app/my\_agent/

# Auto-sort imports

isort app/my\_agent/

## **✅ STEP 3.5: Fix Type Checking Issues**

### **If mypy shows errors, add these imports to your files:**

from typing import Dict, List, Any, Optional, Union

### **Add type hints to functions missing them:**

# Before

def my\_function():

# After

def my\_function() -> Any:

## **✅ STEP 3.6: Run Official Pre-commit Check**

cd app/backend

pre-commit run --all-files

### **Expected Output:**

✅ black............................Passed

✅ ruff.............................Passed

✅ mypy.............................Passed

✅ isort............................Passed

**❌ If any checks fail:** Fix the issues and run again **✅ If all checks pass:** Continue to deployment phase

# **📍 PHASE 4: UPDATE CLIENT CONFIGURATION**

## **✅ STEP 4.1: Update pyproject.toml**

### **Add your dependencies to app/backend/pyproject.toml:**

dependencies = [

# ... keep ALL existing client dependencies ...

# ADD YOUR SEARCH AGENT DEPENDENCIES:

"langgraph>=0.0.26",

"langchain-core>=0.1.0",

"langchain-openai>=0.0.5",

"python-dotenv>=1.0.0",

"pydantic>=2.0.0",

"aiohttp>=3.8.0",

"tenacity>=8.2.0",

]

## **✅ STEP 4.2: Create Basic Tests**

### **Create app/my\_agent/tests/ directory:**

mkdir -p app/my\_agent/tests

touch app/my\_agent/tests/\_\_init\_\_.py

### **Create app/my\_agent/tests/test\_integration.py:**

"""Integration tests for search agent."""

import pytest

def test\_graph\_import():

"""Test graph can be imported."""

from app.my\_agent.agent import graph

assert graph is not None

def test\_module\_exports():

"""Test module exports work."""

from app.my\_agent import graph, process\_search\_request

assert graph is not None

assert process\_search\_request is not None

@pytest.mark.asyncio

async def test\_workflow\_creation():

"""Test workflow can be created."""

from app.my\_agent.agent import create\_search\_summarize\_workflow

workflow = create\_search\_summarize\_workflow()

assert workflow is not None

def test\_client\_integration():

"""Test client's agent\_rest.py can import our graph."""

# This is what client's agent\_rest.py does

from app.my\_agent import graph

assert graph is not None

print("✅ Client integration works!")

## **✅ STEP 4.3: Run Tests**

cd app/backend

pytest app/my\_agent/tests/ -v

**Expected Output:**

✅ test\_graph\_import PASSED

✅ test\_module\_exports PASSED

✅ test\_workflow\_creation PASSED

✅ test\_client\_integration PASSED

# **📍 PHASE 5: GIT & DEPLOYMENT**

## **✅ STEP 5.1: Final Verification**

### **Test everything works together:**

cd app/backend

# Test client integration

python -c "from app.my\_agent import graph; print('✅ Graph import works')"

# Test pre-commit

pre-commit run --all-files

# Test your agent locally

python -m app.my\_agent.test\_search\_agent

# Test LangSmith

langgraph dev &

# Open http://localhost:8123 and verify search\_agent works

## **✅ STEP 5.2: Create Feature Branch**

# Navigate to project root

cd /path/to/client/repo

# Create feature branch

git checkout -b feature/search-summarize-agent-milestone1

# Check status

git status

## **✅ STEP 5.3: Add Your Changes**

# Add only your changes

git add app/backend/app/my\_agent/

git add app/backend/langgraph.json

git add app/backend/pyproject.toml

git add app/backend/.env

# Check what you're committing

git diff --cached

## **✅ STEP 5.4: Final Pre-commit Check**

# This will run on your staged files

git commit -m "test commit" --dry-run

**If pre-commit fails:** Fix issues and re-add files

## **✅ STEP 5.5: Commit Changes**

git commit -m "feat: Add Search & Summarize Agent for Milestone 1

- Integrate multi-source search agent with LangSmith visualization

- Add tool selection and intelligent summarization capabilities

- Configure for Italian and French medical professional verification

- Maintain compatibility with existing client infrastructure

- Add integration tests and proper module exports

Features:

- Search workflow with 4-node graph structure

- LangSmith UI integration for real-time visualization

- Support for medical professional verification

- Compatible with client's agent\_rest.py integration

- Comprehensive error handling and logging

Technical Details:

- Added app/my\_agent/ module with complete workflow

- Updated langgraph.json for graph configuration

- Added dependencies to pyproject.toml

- Configured environment variables for LangSmith

- Proper module exports for client integration"

## **✅ STEP 5.6: Push Feature Branch**

git push origin feature/search-summarize-agent-milestone1

## **✅ STEP 5.7: Create Merge Request**

### **In GitLab:**

1. **Navigate to:** Project → Merge Requests → New Merge Request
2. **Source Branch:** feature/search-summarize-agent-milestone1
3. **Target Branch:** dev
4. **Title:** "Add Search & Summarize Agent - Milestone 1"

### **Description Template:**

## 🎯 Search & Summarize Agent - Milestone 1

### ✅ Features Added:

- Multi-source search and summarization workflow

- LangSmith integration for real-time visualization

- Medical professional verification capabilities

- Tool selection and intelligent summarization

### 🔧 Technical Changes:

- \*\*Added:\*\* `app/my\_agent/` module with complete workflow

- \*\*Updated:\*\* `langgraph.json` for LangSmith integration

- \*\*Updated:\*\* `pyproject.toml` with required dependencies

- \*\*Added:\*\* Integration tests for workflow validation

### 🧪 Testing:

- ✅ All pre-commit checks passing

- ✅ Graph imports successfully for client integration

- ✅ LangSmith UI tested locally

- ✅ Integration tests pass

- ✅ Compatible with existing client infrastructure

### 🎨 LangSmith Demo:

Agent available as `search\_agent` in LangSmith UI with:

- 4-node workflow visualization

- Real-time execution tracking

- Tool selection and summarization steps

- Comprehensive error handling

### 📋 Checklist:

- [x] Code quality checks pass

- [x] Integration tests included

- [x] LangSmith integration tested

- [x] Client compatibility verified

- [x] No modifications to existing client code

1. **Assign:** Relevant reviewers
2. **Labels:** Add appropriate labels
3. **Click:** "Create Merge Request"

# **📍 PHASE 6: MONITOR DEPLOYMENT**

## **✅ STEP 6.1: Watch CI/CD Pipeline**

### **In GitLab:**

1. **Go to:** Project → CI/CD → Pipelines
2. **Find:** Your merge request pipeline
3. **Monitor stages:**
   * ✅ Lint (pre-commit checks)
   * ✅ Build (Docker container)
   * ✅ Test (integration tests)
   * ✅ Deploy (to environment)

### **If Pipeline Fails:**

1. **Click:** Failed job to see logs
2. **Fix:** Issues in your code
3. **Push:** New commit to same branch
4. **Wait:** For pipeline to re-run

## **✅ STEP 6.2: Get Deployment Endpoint**

### **After Successful Deployment:**

1. **Go to:** Project → Deployments → Environments
2. **Find:** Development (or staging) environment
3. **Copy:** Generated endpoint URL
4. **Example:** https://dev-abc123.client-domain.com:8123

## **✅ STEP 6.3: Test Deployed Agent**

### **Access Deployed LangSmith UI:**

# Open deployed endpoint:

https://dev-abc123.client-domain.com:8123

# Verify:

- search\_agent appears in graph list

- Graph structure shows 4 nodes

- Can execute with sample input

- Traces appear in client's LangSmith cloud

### **Test Sample Input:**

{

"search\_input": {

"verification\_needed": true,

"geographic\_region": "IT",

"firstName": "Paolo",

"lastName": "Corvisieri",

"workplaceName": "DISTRETTO SANITARIO FIUMICINO"

}

}

## **✅ STEP 6.4: Verify Client Integration**

### **Check that client's REST API works:**

# If client provides API endpoints, test:

curl -X POST "https://dev-abc123.client-domain.com/api/v1/agent/search" \

-H "Content-Type: application/json" \

-d '{"search\_input": {...}}'

# **📍 PHASE 7: CLIENT DEMO PREPARATION**

## **✅ STEP 7.1: Prepare Demo Materials**

### **Create Demo Script:**

# LangSmith Demo Script

## 1. Introduction (2 min)

"Today I'll demonstrate our Search & Summarize Agent integrated with LangSmith for real-time visualization."

## 2. Architecture Overview (3 min)

- Show graph structure: 4-node workflow

- Explain tool selection logic

- Highlight intelligent summarization

## 3. Live Demo (10 min)

- Input: Italian medical professional data

- Watch: Real-time execution

- Observe: Tool selection and search process

- Review: Final summarization with confidence scores

## 4. Technical Integration (5 min)

- Demonstrate: Client's infrastructure compatibility

- Show: LangSmith trace visibility

- Explain: Deployment process and monitoring

### **Prepare Sample Data:**

{

"search\_input": {

"verification\_needed": true,

"geographic\_region": "IT",

"firstName": "Paolo",

"lastName": "Corvisieri",

"workplaceName": "DISTRETTO SANITARIO FIUMICINO",

"address": "Velletri",

"specialtyCode": "18"

},

"vr\_data": {

"validation.id": 1019001316927771,

"individual.firstName": "Paolo",

"individual.lastName": "Corvisieri",

"workplace.usualName": "DISTRETTO SANITARIO FIUMICINO"

}

}

## **✅ STEP 7.2: Demo Checklist**

### **Before Demo:**

* [ ] Deployed endpoint is accessible
* [ ] search\_agent appears in LangSmith UI
* [ ] Sample input data ready
* [ ] Client's LangSmith dashboard accessible
* [ ] Screen sharing setup tested

### **During Demo:**

* [ ] Show graph architecture first
* [ ] Input sample data step-by-step
* [ ] Narrate each workflow node as it executes
* [ ] Highlight real-time visualization benefits
* [ ] Show final results and confidence scores
* [ ] Answer client questions

### **After Demo:**

* [ ] Provide access to deployed endpoint
* [ ] Share documentation and sample inputs
* [ ] Discuss next phase requirements
* [ ] Get feedback and improvement suggestions

# **🎯 SUCCESS CRITERIA CHECKLIST**

## **✅ Local Development:**

* [ ] Search agent works locally
* [ ] Client integration (from app.my\_agent import graph) works
* [ ] LangSmith UI accessible at localhost:8123
* [ ] All pre-commit checks pass

## **✅ Deployment:**

* [ ] GitLab CI/CD pipeline passes
* [ ] Deployed endpoint generated
* [ ] search\_agent visible in deployed LangSmith UI
* [ ] Agent executes successfully in production

## **✅ Client Integration:**

* [ ] Compatible with client's agent\_rest.py
* [ ] No modifications to client's existing code
* [ ] Traces appear in client's LangSmith dashboard
* [ ] Ready for client demo

# **🚨 TROUBLESHOOTING QUICK FIXES**

## **❌ Graph Import Fails:**

# Fix your exports:

# 1. Check app/my\_agent/\_\_init\_\_.py exports 'graph'

# 2. Check app/my\_agent/agent.py creates 'graph'

# 3. Test: python -c "from app.my\_agent import graph"

## **❌ Pre-commit Fails:**

# Auto-fix most issues:

black app/my\_agent/

ruff check --fix app/my\_agent/

isort app/my\_agent/

# Then run: pre-commit run --all-files

## **❌ LangSmith UI Empty:**

# Check graph export:

python -c "from app.my\_agent.agent import graph; print(type(graph))"

# Restart: langgraph dev

## **❌ Pipeline Fails:**

# Check GitLab pipeline logs

# Fix issues in your code

# Push new commit to same branch

# **🚀 YOU'RE READY TO DEPLOY!**

**Follow this guide step-by-step and you'll have a successful deployment with LangSmith integration! 💪**

## **📋 COMPLETE STEP-BY-STEP DEPLOYMENT GUIDE## 🎯 THERE'S YOUR COMPLETE DEPLOYMENT ROADMAP!**

This guide takes you from **where you are now** (working search agent locally) all the way to **successful client demo** with deployed LangSmith UI.

### **📋 WHAT THIS COVERS:**

1. **🔧 Fix Client Integration** - Make your agent work with their existing code
2. **🎨 LangSmith UI Testing** - Get the visualization working locally
3. **✅ Pre-commit & Quality** - Pass all their code quality checks
4. **📝 Configuration** - Update the 3 files you need to modify
5. **🚀 Git & Deployment** - From feature branch to production
6. **🎪 Demo Preparation** - Ready for client presentation

### **🎯 FOLLOW BLINDLY:**

* Each step has **exact commands** to run
* **Clear success/failure criteria** at each stage
* **Troubleshooting fixes** for common issues
* **No guesswork** - just follow the checklist

### **💪 YOUR CURRENT STATUS:**

✅ **Search agent works locally** ⚠️ **Need to fix pre-commit issues** → Phase 3 will solve this  
 ⚠️ **Need client integration** → Phase 1 will solve this

**Start with Phase 1 (Fix Client Integration) and work through each phase sequentially. You've got this! 🚀**