

Kearney AI Code Orchestrator

A Meta-Framework for AI-Driven Software Delivery

Version: 1.0

Date: November 2025

Contents

- 1. Executive Summary**
- 2. Business Problems Solved**
- 3. Supported Project Types**
- 4. Key Features & Capabilities**
- 5. Technology Stack**
- 6. System Requirements**
- 7. Getting Started**
- 8. Target Users**

Executive Summary

KEY VALUE PROPOSITION

The Claude Code Orchestrator enables **structured, repeatable, high-quality project delivery at scale** - whether analytics pipelines, ML systems, web applications, or optimization models.

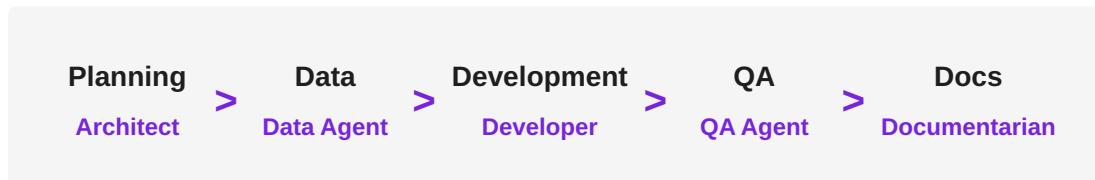
The **Claude Code Orchestrator** is a meta-framework that coordinates multiple specialized Claude Code subagents to collaboratively build complex software projects through a structured, checkpoint-driven workflow.

It solves the fundamental challenge of coordinating AI agents for enterprise-grade software delivery with quality assurance, governance compliance, and full traceability.



Core Philosophy

Rather than using a single AI agent for all tasks, the orchestrator assigns specialized agents to phases where they excel:



Each phase concludes with validated checkpoint artifacts before proceeding.

Business Problems Solved

1. Complexity Coordination

Problem: Complex projects require multiple phases with dependencies - architecture must be validated before coding, data must be processed before model training.

Solution: Checkpoint-driven workflow that sequences phases, validates completeness, and prevents downstream rework from bad early decisions.

2. Quality Assurance

Problem: Single agents can't consistently enforce quality standards across different project types and client requirements.

Solution: Automated quality gates (test coverage, security scanning, hygiene score) that must pass before phase progression.

3. Expertise Fragmentation

Problem: One AI agent can't excel at architecture design AND code implementation AND testing AND documentation.

Solution: Specialized subagents focused on their domain - Architect for design, Developer for code, QA for testing, Documentarian for guides.

4. Governance & Compliance

Problem: Client-specific compliance requirements (GDPR, HIPAA, SOC2, PCI-DSS) vary widely with no standardized enforcement.

Solution: Client governance system with YAML configurations that automatically enforce compliance requirements.

5. Decision Traceability

Problem: Complex projects involve many decisions; hard to answer "why did we choose this technology?"

Solution: Architecture Decision Records (ADRs) document context, rationale, and alternatives for significant choices.

6. Iteration & Rollback

Problem: Reverting errors after phase completion costs exponentially more effort.

Solution: Checkpoint artifacts enable non-destructive rollback to previous phases without losing progress.

Supported Project Types

The orchestrator provides templates for four primary project types:

Analytics Projects

Best for: Business intelligence, ETL pipelines, dashboards, reporting systems

| | |
|-------------------|--|
| Tech Stack | Python + SQL, pandas, DuckDB, Streamlit/Dash |
|-------------------|--|

| | |
|---------------|---|
| Agents | Architect, Data, Developer, QA, Documentarian |
|---------------|---|

| | |
|---------------------|--|
| Deliverables | Processed datasets, dashboards, data quality reports |
|---------------------|--|

Example: Customer segmentation analysis, KPI dashboards, churn prediction reports

Machine Learning Projects

Best for: Model development, inference APIs, monitoring & retraining pipelines

| | |
|-------------------|--|
| Tech Stack | Python + scikit-learn/PyTorch, MLflow, FastAPI |
|-------------------|--|

| | |
|---------------|---|
| Agents | Architect, Data, Developer, QA, Documentarian |
|---------------|---|

| | |
|---------------------|---|
| Deliverables | Trained models, inference APIs, monitoring dashboards |
|---------------------|---|

Example: Demand forecasting service, recommendation engine, fraud detection system

Web Applications

Best for: Full-stack apps, SPAs, admin dashboards, SaaS products

Tech Stack

TypeScript + React, FastAPI/Node.js, PostgreSQL

Agents

Architect, Developer, QA, Documentarian

Deliverables

Frontend code, backend APIs, tests, deployment guides

Example: Customer portal, internal tool, executive dashboard

Supply Chain & Optimization

Best for: Operations research, resource allocation, scheduling, logistics

Tech Stack

Python + PuLP/Gurobi, pandas, scipy

Agents

Architect, Data, Optimization Engineer, Developer, QA, Documentarian

Deliverables

Optimization models, dashboards, scenario analysis

Example: Route optimization, facility location, workforce scheduling

Key Features & Capabilities

Multi-Agent Orchestration

Eight specialized agents with defined responsibilities:

| Agent | Responsibility | Key Outputs |
|----------------------|-------------------------------------|---------------------------------|
| Architect | System design, technology selection | Architecture docs, ADRs |
| Data | ETL pipelines, feature engineering | Processed data, models, metrics |
| Developer | Feature implementation, APIs | Source code, tests |
| QA | Testing, validation | Test reports, coverage metrics |
| Documentarian | Documentation, guides | README, user guides, API docs |
| Consensus | Review and approve proposals | Approval decisions, feedback |
| Steward | Repository hygiene | Hygiene reports, cleanup |
| Reviewer | Code review (optional) | Review comments |

Client Governance

Automates enforcement of client-specific requirements:

Quality Gates

Test coverage (50-95%),
security scanning,
complexity limits, peer
review requirements

Compliance

SOC2, GDPR, HIPAA,
PCI-DSS, CCPA -
automated enforcement

Brand Constraints

Colors, fonts, terminology,
logos - consistent
branding

Deployment Policies

Approval requirements,
deployment windows,
rollback policies

Skills Framework

Reusable analytical methodologies:

- > **Time Series Analytics:** ARIMA, Prophet, LSTM for forecasting
- > **Optimization Modeling:** Linear programming, resource allocation
- > **Survey Data Processing:** Entity resolution, text analysis

Skills are domain-neutral - the same time series skill applies to telecom call forecasting, retail sales prediction, and healthcare admission planning.

Architecture Decision Records (ADRs)

Document significant technical decisions with:

- Context and constraints
- Decision and rationale
- Alternatives considered

- Consequences and trade-offs

Technology Stack

Core Languages

| Language | Version | Usage |
|------------|---------|------------------------------|
| Python | 3.10+ | Backend, data processing, ML |
| TypeScript | 5.6+ | Frontend, tooling |
| SQL | - | Data queries (DuckDB) |

Backend & API

- > **FastAPI** - Modern async web framework
- > **Uvicorn** - ASGI server
- > **Pydantic** - Data validation
- > **Typer** - CLI framework

Data & Analytics

- > **pandas / polars** - Data manipulation
- > **DuckDB** - In-process OLAP database
- > **scikit-learn** - Machine learning
- > **matplotlib / plotly** - Visualization

AI Integration

- > **Anthropic Claude API** - AI agent execution (claude-sonnet-4)

Infrastructure & Deployment

- > **AWS Lambda** - Serverless compute
- > **AWS Amplify** - Frontend hosting
- > **GitHub Actions** - CI/CD automation
- > **Docker** - Containerization

MLOps (Optional)

- > **MLflow** - Experiment tracking
- > **DVC** - Data version control
- > **Great Expectations** - Data quality

Observability

- > **OpenTelemetry** - Distributed tracing
- > Custom metrics collection and dashboards

System Requirements

Development Environment

| Component | Requirement |
|----------------|--------------------------------------|
| Python | 3.10+ (3.11+ recommended) |
| Node.js | 18.0+ (20.0+ for documentation site) |
| npm | 9.0+ |
| Git | 2.0+ |
| Memory | 2GB+ for local development |
| Disk | 5GB+ for development environment |

Required Configuration

```
# Environment variable for AI agent execution
export ANTHROPIC_API_KEY="your-api-key"

# Execution mode (optional, defaults to in_session)
export ORCHESTRATOR_EXECUTION_MODE="api"
```

Installation

```
# Clone repository
git clone https://github.com/your-org/clause-code-orchestrator.git
cd clause-code-orchestrator

# Install Python dependencies
pip install -e .

# Install documentation site dependencies (optional)
cd site && npm install && cd ..
```

Getting Started

Step 1: Bootstrap a New Project

```
# Analytics project
orchestrator bootstrap analytics \
--output ~/projects/customer-analytics

# ML project
orchestrator bootstrap ml-model \
--output ~/projects/demand-forecast

# Web application
orchestrator bootstrap webapp \
--output ~/projects/admin-dashboard
```

Step 2: Configure Project Intake

Edit the generated `intake.yaml`:

```
project:
  name: "Customer Analytics"
  type: "analytics"
goals:
  primary:
    - "Segment customers by value"
    - "Identify churn risk"
data:
  sources:
    - "Production database"
    - "CRM API"
orchestration:
  enabled_agents:
    - "architect"
```

```
- "data"
- "developer"
- "qa"
- "documentarian"

consensus_required:
- "planning"
- "data_engineering"
```

Step 3: Start the Workflow

```
# Start orchestration
orchestrator run start --intake intake.yaml

# Execute next phase
orchestrator run next

# Check status
orchestrator run status

# Approve consensus checkpoint
orchestrator run approve
```

Step 4: Monitor Progress

```
# View metrics dashboard
orchestrator run metrics

# Run repository hygiene check
orchestrator run repo-hygiene
```

Target Users

| Persona | Use Case | Primary Benefit |
|------------------------|-----------------------------------|--|
| Data Scientists | Analytics pipelines, ML models | Specialized Data agent, quality gates |
| ML Engineers | Production models, monitoring | Model versioning, drift detection |
| Developers | Full-stack applications | Architecture-first approach, test gates |
| Consultants | Client engagements | Governance compliance, ADR traceability |
| OR Specialists | Optimization problems | Dedicated optimization phase |

Best Fit Scenarios

- > **Complex multi-phase projects** requiring coordination across design, development, QA
 - > **Compliance-heavy engagements** (GDPR, HIPAA, SOC2 requirements)
 - > **Teams** needing clear roles and handoffs between phases
 - > **Repeatable workflows** across similar project types
 - > **Risk-sensitive projects** requiring quality gates and rollback capability
-

Summary

The Claude Code Orchestrator transforms AI-assisted software development from ad-hoc single-agent interactions into a **structured, governance-compliant, checkpoint-driven workflow.**

Specialization

Right agent for each phase

Quality

Automated gates prevent mediocre deliverables

Compliance

Client governance automatically enforced

Traceability

Full audit trail via ADRs and checkpoints

KEY TAKEAWAY

Whether you're building a customer segmentation dashboard, deploying a demand forecasting API, or creating an executive reporting tool, the orchestrator provides the structure, quality gates, and documentation to deliver **production-grade results.**

Kearney Confidential

For questions, contact the platform team or open an issue in the repository.