dagster

Dagster Deep Dive

Best Practices when using LLMs with Dagster

1/27/2026

Table of contents

01 Intro

02 The Development Cycle

03 Common Pitfalls

05 Dagster Development Practices

06 Official Claude Plugin

07 Demo

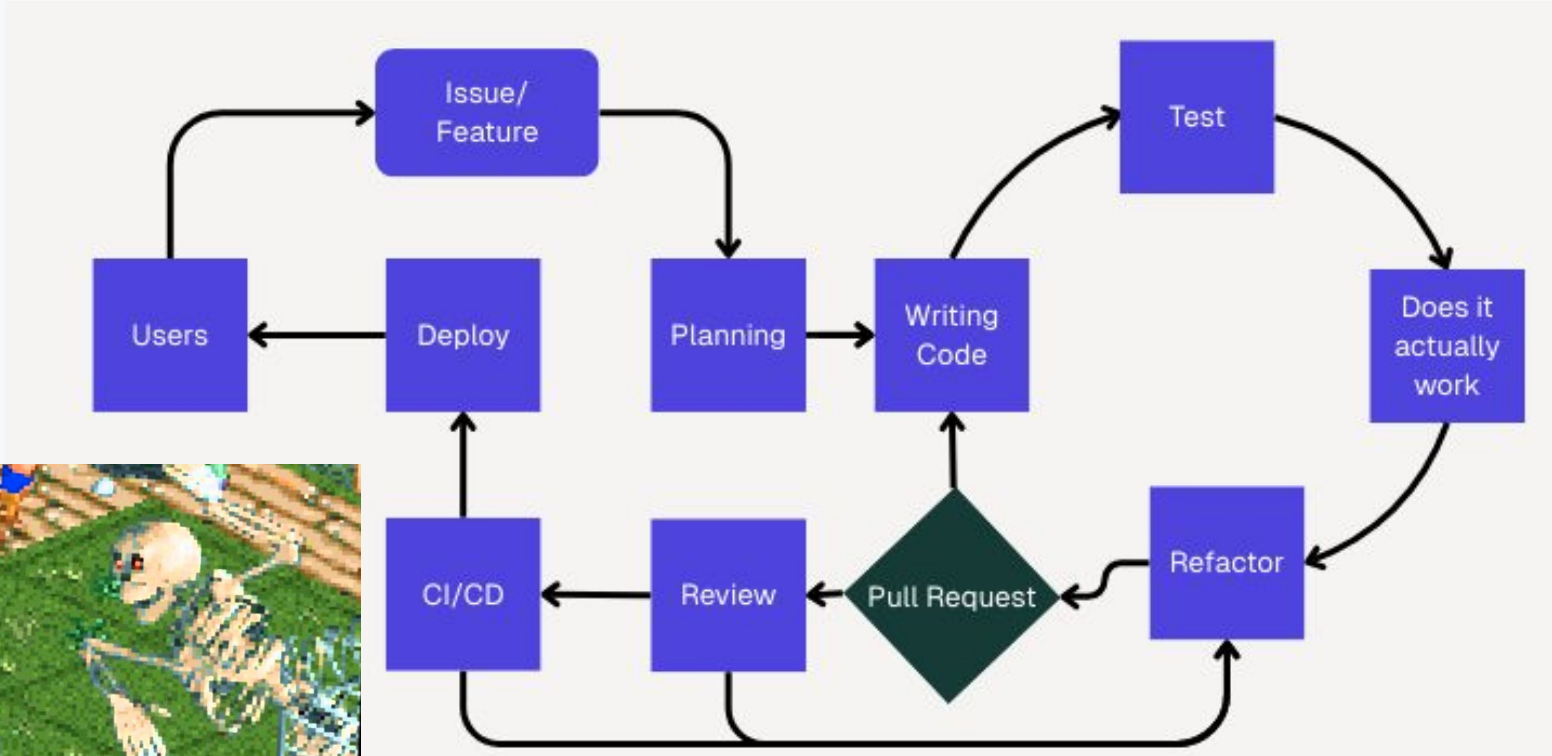
dagster

Software Engineering Best Practices

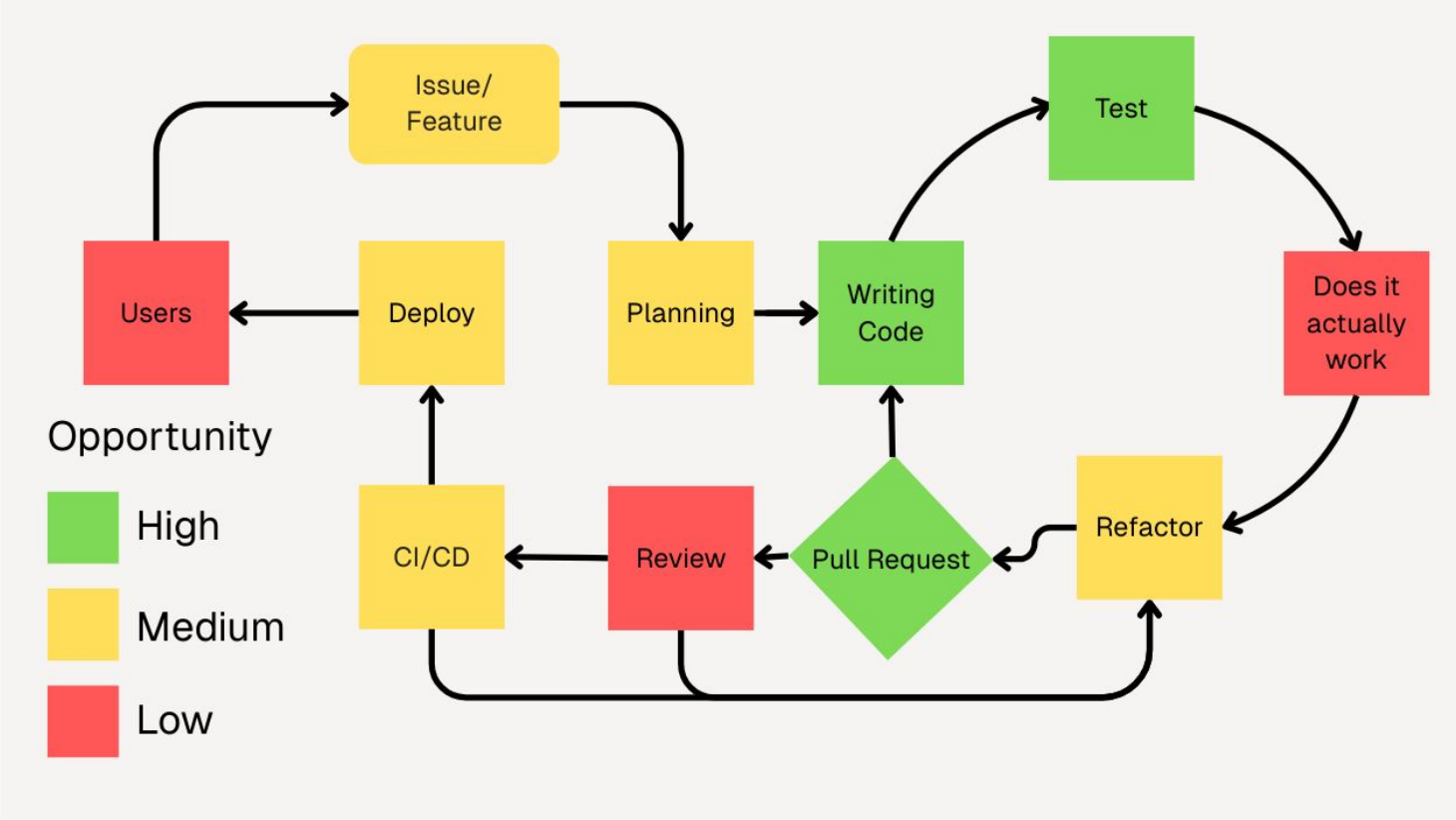
The Development Lifecycle

dagster

At a High Level



Evolving landscape



dagster

Best Practices

- You don't need to have the perfect setup immediately, incrementally implement AI where you feel the greatest pain
- Starting with a solid planning and testing foundation yields the best results
- Share LLM artifacts across the team
- Fundamentals are important
- Asynchronous work

Composition of tools and terminal applications work extremely well.

Things to Consider

Common Pitfalls

Dawn of the Brain-Rotted Zombies

The Weekend Windup #16 - Reflections, Cool Reads, Events, and More

JOE REIS

JAN 03, 2026

dagster

There’s doing the work
and there’s

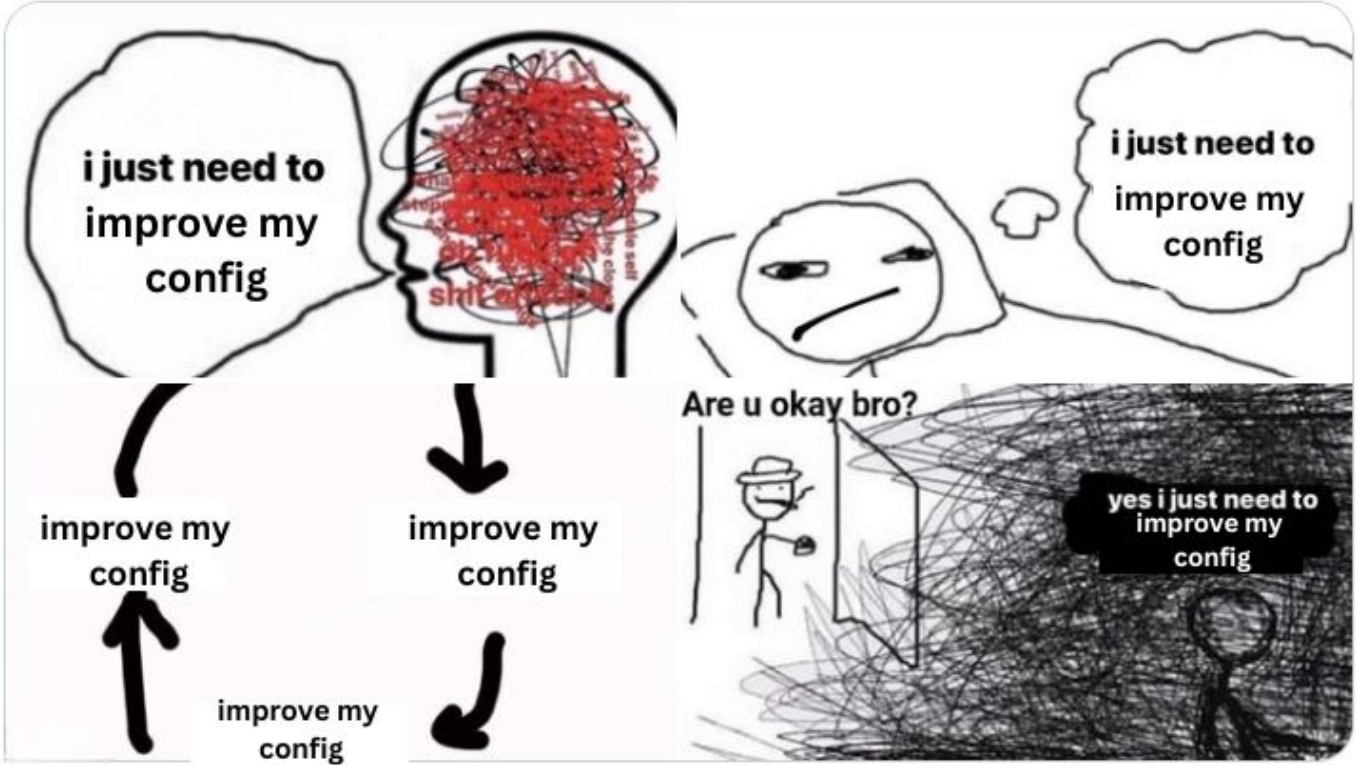
Doing the work

💯💸🔨💵💪😤

dagster

Procrastination

At some point you actually need to do work



dagster

The Less Glamorous Work

- Docs
- Maintenance
- Collaboration
- Reviews

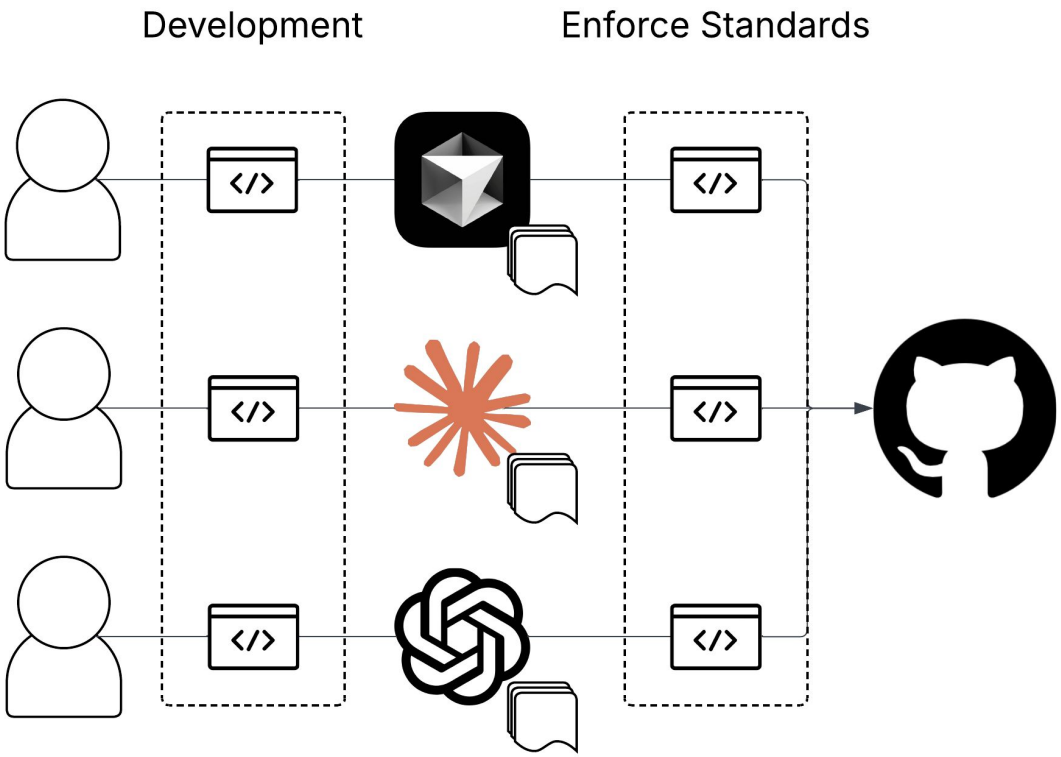
dagster

Organization Wide AI Workflows

Dagster Development Practices

dagster

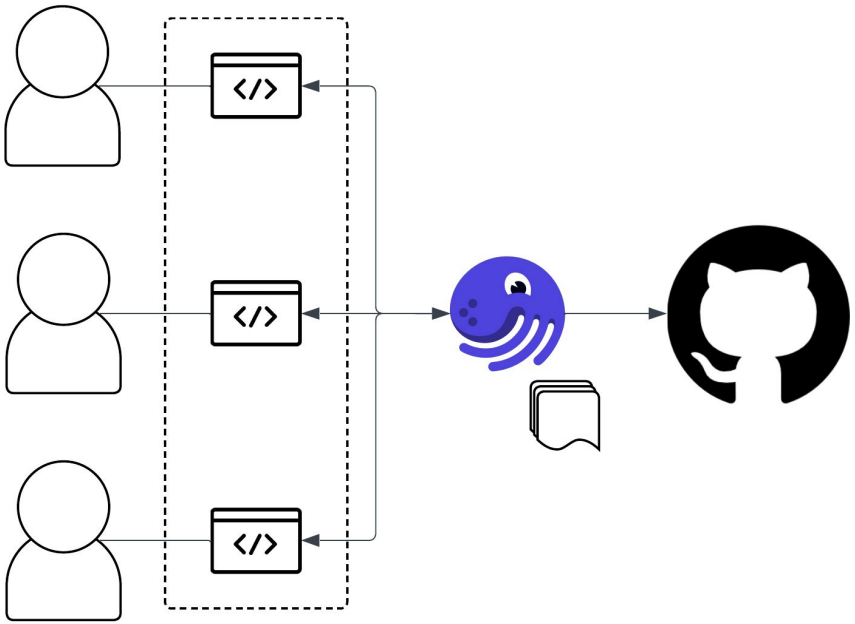
Building as an Organization



dagster

Building as an Organization

Development and
Enforce Standards



dagster

Blog > Dignified Python: 10 Rules to Improve your LLM Agents

Dignified Python: 10 Rules to Improve your LLM Agents

January 9, 2026

Colton Padden
Engineer & Developer Advocate

dagster

Blog > Dignified Python: 10 Rules to Improve your LLM Agents

Dignified Python: 10 Rules to Improve your LLM Agents

January 9, 2026

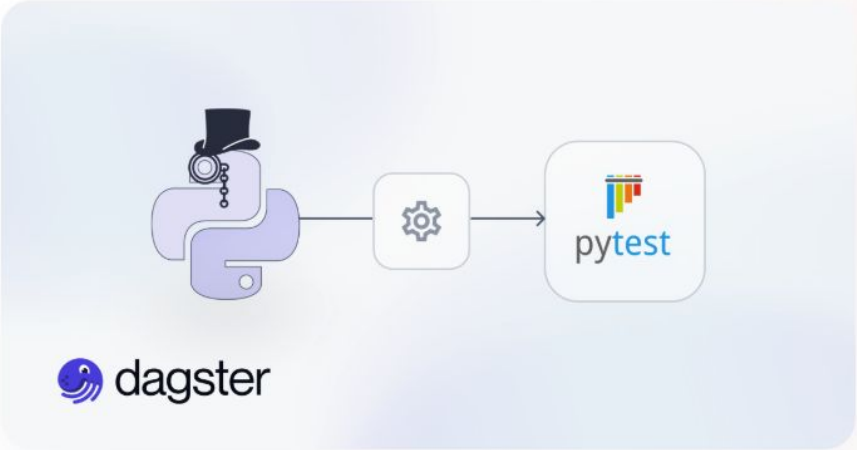
Colton Padden
Engineer & Developer Advocate

Blog > Pytest for Agent-Generated Code: Concrete Testing Strategies to Put Into Practice

Pytest for Agent-Generated Code: Concrete Testing Strategies to Put Into Practice

January 26, 2026

Dennis Hume



dagster

Idiomatic Python

Internally, we embed our Python usage rules to ensure Dagster code adheres to our coding standards at the time it's written, rather than deferring enforcement to the review process

```python
# WRONG: Exception as control flow
try:
    value = mapping[key]
    process(value)
except KeyError:
    pass

# CORRECT: Check first
if key in mapping:
    value = mapping[key]
    process(value)
```

```python
# WRONG: Silent exception swallowing
try:
    risky_operation()
except:
    pass

# CORRECT: Let exceptions bubble up
risky_operation()
```

dagster

Idiomatic Python Rules

Exception Handling Basics

Core Principle

ALWAYS use LBYL, NEVER EAFP for control flow

LBYL means checking conditions before acting. EAFP (Easier to Ask for Forgiveness than Permission) means trying operations and catching exceptions. In dignified Python, we strongly prefer LBYL.

Dictionary Access Patterns

```python
# CORRECT: Membership testing
if key in mapping:
    value = mapping[key]
    process(value)
```

dagster

Idiomatic Python Rules

When Exceptions ARE Acceptable

Exceptions are ONLY acceptable at:

1. **Error boundaries** (CLI/API level)
2. **Third-party API compatibility** (when no alternative exists)
3. **Adding context before re-raising**

Default: Let exceptions bubble up

For detailed exception handling patterns including B904 chaining, third-party API examples, and anti-patterns, see `references/exception-handling.md`.

dagster

Python Testing

We include testing as a first-class part of our centralized agentic development workflow to ensure correctness and reliability are validated during development, rather than deferred to downstream verification

```python
def test_user_service_creates_user() -> None:
    """Verify user service creates users correctly."""
    # Arrange: Configure fake with desired state
    fake_db = FakeDatabaseAdapter()
    fake_email = FakeEmailClient(
        should_fail_for=["invalid@example.com"]
    )

    service = UserService(database=fake_db, email_client=fake_email)

    # Act: Execute business logic
    user = service.create_user(
        name="Alice",
        email="alice@example.com"
    )

    # Assert: Check behavior
    assert user.id == 1
    assert user.name == "Alice"
    assert user.email == "alice@example.com"

    # Assert: Check side effects via fake's tracking
    assert len(fake_db.executed_queries) == 1
    assert "INSERT INTO users" in fake_db.executed_queries[0]
    assert len(fake_email.sent_emails) == 1
    assert fake_email.sent_emails[0]["to"] == "alice@example.com"
```

dagster

We enforce multiple levels of testing at the repository level to ensure correctness, stability, and confidence as code moves from local development to production

dagster

Level 1: Fake Infrastructure

Test of fake (see level 4)
Pct of Tests: 5

Level 2: Integration sanity

Real implementation with mocking
Pct of Tests: 5

Level 3: Pure unit tests

No dependencies at all
Pct of Tests: 10

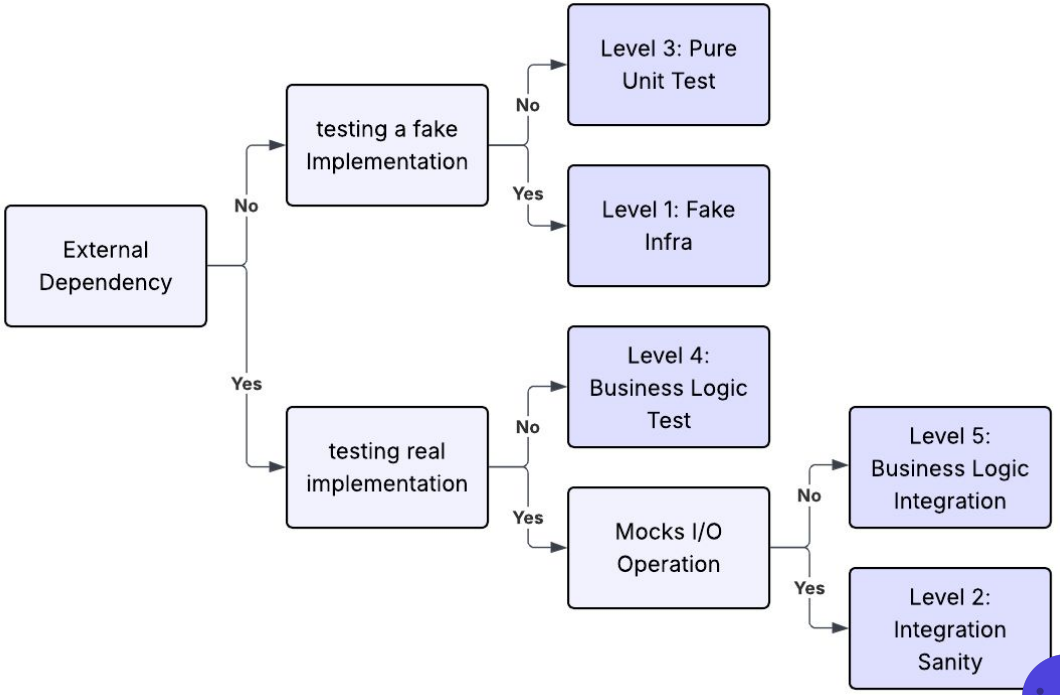
Level 4: Business logic tests

Use fake infrastructure
Pct of Tests: 70

Level 5: Integration tests

Real
Pct of Tests: 5

Testing routing



dagster

Routing rules

Layer 4: Business Logic Tests over Fakes (MAJORITY)

Purpose: Test application logic extensively with fast in-memory fakes.

Location: `tests/unit/services/`, `tests/unit/`, `tests/commands/`

When to write: For EVERY feature and bug fix. This is the default testing layer.

Why: Fast, reliable, easy to debug. Tests run in milliseconds, not seconds. This is where most testing happens.

Pattern: Configure Fakes, Execute Logic, Assert Behavior

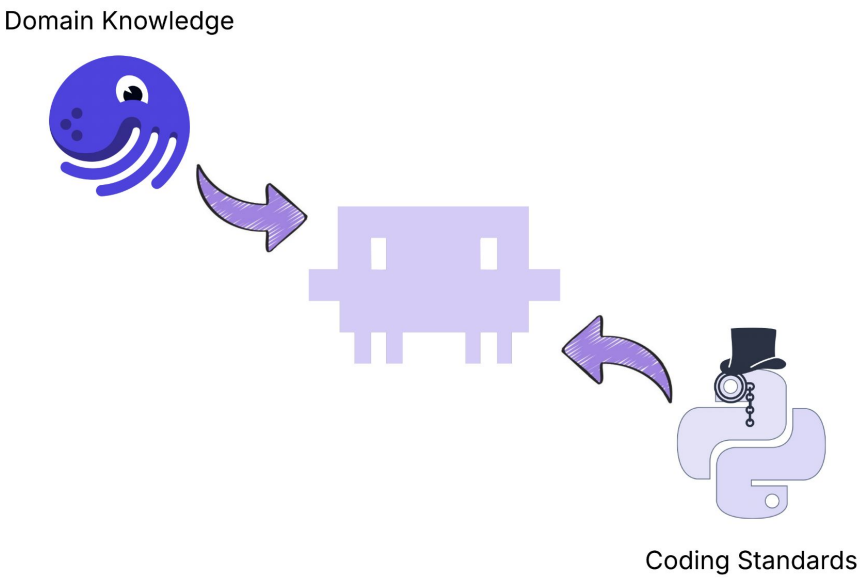
```python
def test_user_service_creates_user() -> None:
    """Verify user service creates users correctly."""
    # Arrange: Configure fake with desired state
    fake_db = FakeDatabaseAdapter()
    fake_email = FakeEmailClient(
        should_fail_for=["invalid@example.com"]
    )
```

dagster

The Claude Plugin

dagster

Our Claude plugin combines domain knowledge about Dagster best practices with the internal coding standards we use



dagster

dagster-io / **dagster-claude-plugins**

master | 1 Branch | 0 Tags | Go to file t | Add file | Code

cmpadden Merge pull request #7 from dagster-io/colton/feat/remote-dbt-recommen... a3ecd2c · last week 33 Commits

.claude-plugin	Ensure yes is piped into create- commands; refactor fold...	last week
.github	Add .github/PULL_REQUEST_TEMPLATE.md	last week
plugins	Update convention to recommend remote dbt project con...	last week
README.md	Restructure README.md by plugin	last week

README

Claude Plugins for Dagster

A collection of Claude Code plugins for building workflows and data pipelines using Dagster.

Installation

Add this marketplace to Claude Code:

```
/plugin marketplace add dagster-io/dagster-claude-plugins
```

Install a plugin:

```
/plugin install dg@claude-plugins-dagster
```

To update your marketplace plugins:

```
/plugin marketplace update
```

dagster

Claude Plugins Marketplace

```
Claude Code v2.1.15

Welcome back!

Sonnet 4.5 (1M context) · API Usage Billing ·
Dagster Labs
~/src/dagster-website

Tips for getting started
Run /init to create a CLAUDE.md file with instructions for Claude

Recent activity
No recent activity

> /plugin

Plugins  Discover  Installed  Marketplaces  (←/→or tab to cycle)

Manage marketplaces

> + Add Marketplace

  ● claude-plugins-official
    anthropics/claude-plugins-official
    48 available • 1 installed • Updated 1/22/2026

  ● dagster-claude-plugins
    dagster-io/dagster-claude-plugins
    3 available • 3 installed • Updated 1/15/2026

Enter to select · u to update · r to remove · esc to go back
```

```
Plugins  Discover  Installed  Marketplaces  (←/→or tab to cycle)

dagster-claude-plugins
dagster-io/dagster-claude-plugins

4 available plugins

Installed plugins (4):
 ● dignified-python
   Production-tested Python coding standards with version-aware type annotations and LBYL exception patterns
 ● dagster-conventions
   Comprehensive Dagster development conventions and best practices
 ● dagster-integrations
   Comprehensive index of 82+ Dagster integrations including cloud platforms, data warehouses, ETL tools, AI/ML, and more
 ● dg
   Dagster CLI commands for debugging runs, analyzing logs, and troubleshooting

❯ Browse plugins (4)
  Update marketplace (last updated 1/22/2026)
  Enable auto-update
  Remove marketplace
   Enter to select · esc to go back
```

dagster

dg Skills and Commands

dg

Commands for building and debugging.

Build:

- `/dg:create-project <name>` - Create a new Dagster project with recommended structure
- `/dg:create-workspace <name>` - Initialize a workspace for managing multiple projects
- `/dg:prototype <requirements>` - Build production-ready Dagster implementations with best practices, testing, and validation

Debug:

- `/dg:logs <run-id> [level] [limit]` - Retrieve and display logs for a run
- `/dg:troubleshoot <run-id>` - Debug failing runs by analyzing error logs

dagst

Conventions

dagster-conventions

Comprehensive Dagster development conventions and best practices.

This skill provides expert guidance for Dagster data orchestration including assets, resources, schedules, sensors, partitions, testing, and ETL patterns.

Conventions

dagster-conventions

Comprehensive Dagster development conventions and best practices.

This skill provides expert guidance for Dagster data orchestration including assets, resources, schedules, sensors, partitions, testing, and ETL patterns.

Jobs

Jobs select which assets to materialize together:

Basic Job Definition

```python
import dagster as dg

trip_update_job = dg.define_asset_job(
    name="trip_update_job",
    selection=["taxi_trips", "taxi_zones"],
)
```

Asset Selection Patterns

```python
# Select specific assets
dg.AssetSelection.assets("asset_a", "asset_b")

# Select all assets
dg.AssetSelection.all()

# Select by group
dg.AssetSelection.groups("analytics")

# Select with dependencies
dg.AssetSelection.assets("final_report").upstream()  # Include upstream
dg.AssetSelection.assets("raw_data").downstream()    # Include downstream

# Combine selections
dg.AssetSelection.all() - dg.AssetSelection.assets("excluded_asset")
dg.AssetSelection.groups("a") | dg.AssetSelection.groups("b")  # Union
dg.AssetSelection.groups("a") & dg.AssetSelection.assets("specific")  # Intersection
```

Conventions

dagster-conventions

Comprehensive Dagster development conventions and best practices.

This skill provides expert guidance for Dagster data orchestration including assets, resources, schedules, sensors, partitions, testing, and ETL patterns.

Jobs

Jobs sel

Basic J

```
import

trip_u
    na
    se
)
```

Asset S

```
# Sele
dg.Ass

# Sele
dg.Ass

# Sele
dg.Ass

# Sele
dg.Ass
dg.Ass

# Comb
dg.Ass
dg.Ass
dg.Ass
```

Schedules

Basic Schedule

```python
import dagster as dg
from my_project.defs.jobs import trip_update_job

trip_update_schedule = dg.ScheduleDefinition(
    job=trip_update_job,
    cron_schedule="0 0 5 * *",  # 5th of each month at midnight
)
```

Common Cron Patterns

Pattern	Meaning
`* * * * *`	Every minute
`0 * * * *`	Every hour (at minute 0)
`0 0 * * *`	Daily at midnight
`0 6 * * *`	Daily at 6:00 AM
`0 0 * * 1`	Weekly on Monday at midnight
`0 0 * * 1-5`	Weekdays at midnight
`0 0 1 * *`	Monthly on the 1st at midnight
`0 0 5 * *`	Monthly on the 5th at midnight
`15 5 * * 1-5`	Weekdays at 5:15 AM

Tip: Use Crontab Guru to create and test cron expressions.

Conventions

dagster-conventions

Comprehensive Dagster development conventions and best practices.

This skill provides expert guidance for Dagster data orchestration including assets, resources, schedules, sensors, partitions, testing, and ETL patterns.

Jobs

Jobs sel

Basic J

```
import

trip_u
    na
    se
)
```

Asset S

```
# Sele
dg.Ass

# Sele
dg.Ass

# Sele
dg.Ass

# Sele
dg.Ass
dg.Ass

# Comb
dg.Ass
dg.Ass
dg.Ass
```

Schedules

Basic Schedul

```
import dagster
from my_projec

trip_update_sc
    job=trip_u
    cron_sched
)
```

Common Cron

Pattern
`* * * * *`
`0 * * * *`
`0 0 * * *`
`0 6 * * *`
`0 0 * * 1`
`0 0 * * 1-5`
`0 0 1 * *`
`0 0 5 * *`
`15 5 * * 1-5`

Tip: Use Crontab

Declarative Automation

Modern automation pattern that sets conditions on assets rather than scheduling jobs.

What is Declarative Automation?

Instead of defining schedules or sensors that trigger jobs, you define **conditions** directly on assets. Dagster evaluates these conditions and automatically materializes assets when conditions are met.

Basic AutomationCondition Patterns

```python
import dagster as dg
from dagster import AutomationCondition

# Materialize when missing (never been created)
@dg.asset(
    automation_condition=AutomationCondition.on_missing()
)
def initial_load() -> None:
    pass

# Materialize on cron schedule
@dg.asset(
    automation_condition=AutomationCondition.on_cron("0 9 * * *")
)
def daily_report() -> None:
    pass

# Materialize when upstream assets update
@dg.asset(
    deps=["raw_data"],
    automation_condition=AutomationCondition.eager()
)
def processed_data() -> None:
    """Automatically updates when raw_data changes."""
    pass
```

Integrations

dagster-integrations

Comprehensive index of 82+ Dagster integrations including cloud platforms, data warehouses, ETL tools, AI/ML, data quality, monitoring, and more.

This skill helps you discover and integrate with the right tools for your data pipelines.

Integrations

dagster-integrations

Comprehensive index of 82+ Dagster integrations including cloud platforms, data warehouses, ETL tools, AI/ML, data quality, monitoring, and more.

This skill helps you discover and integrate with the right tools for your data pipelines.

Dagster Integrations Index

Navigate 82+ Dagster integrations organized by Dagster's official taxonomy. Find AI/ML tools, ETL platforms, data storage, compute services, BI tools, and monitoring integrations.

Quick Reference by Category

Category	Count	Common Tools	Reference
AI & ML	6	OpenAI, Anthropic, MLflow, W&B	`references/ai.md`
ETL/ELT	9	dbt, Fivetran, Airbyte, PySpark	`references/etl.md`
Storage	35+	Snowflake, BigQuery, Postgres, DuckDB	`references/storage.md`
Compute	15+	AWS, Databricks, Spark, Docker, K8s	`references/compute.md`
BI & Visualization	7	Looker, Tableau, PowerBI, Sigma	`references/bi.md`
Monitoring	3	Datadog, Prometheus, Papertrail	`references/monitoring.md`
Alerting	6	Slack, PagerDuty, MS Teams, Twilio	`references/alerting.md`
Testing	2	Great Expectations, Pandera	`references/testing.md`
Other	2+	Pandas, Polars	`references/other.md`

Integrations

dagster-integrations

Comprehensive index of 82+ Dagster integrations including cloud platforms, data warehouses, ETL tools, AI/ML, data quality, monitoring, and more.

This skill helps you discover and integrate with the right tools for your data pipelines.

Dagster Integrations Index

Navigate
services,

Quick

Ca
AI & Ml
ETL/EL
Storag
Compu
BI & Vi
Monito
Alertin
Testing
Other

dbt

Package: `dagster-dbt` | **Support:** Dagster-supported

Transform data using SQL models with automatic dependency management, incremental updates, and testing. Component-based integration.

Use cases:

- SQL-based data transformations in warehouses
- Build dimensional models and data marts
- Test data quality with dbt tests
- Document data lineage and schemas

Quick start (Component-based - Recommended):

```yaml
# defs/transform/defs.yaml
type: dagster_dbt.DbtProjectComponent

attributes:
  project:
    repo_url: https://github.com/dagster-io/jaffle-platform.git
    repo_relative_path: jdbt
  dbt:
    target: dev
```

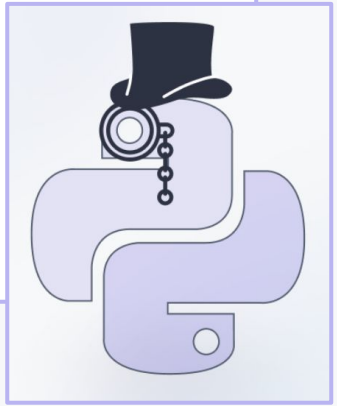
Dignified Python

dignified-python

Production-tested Python coding standards with version-aware type annotations, LBYL exception handling, and modern typing patterns.

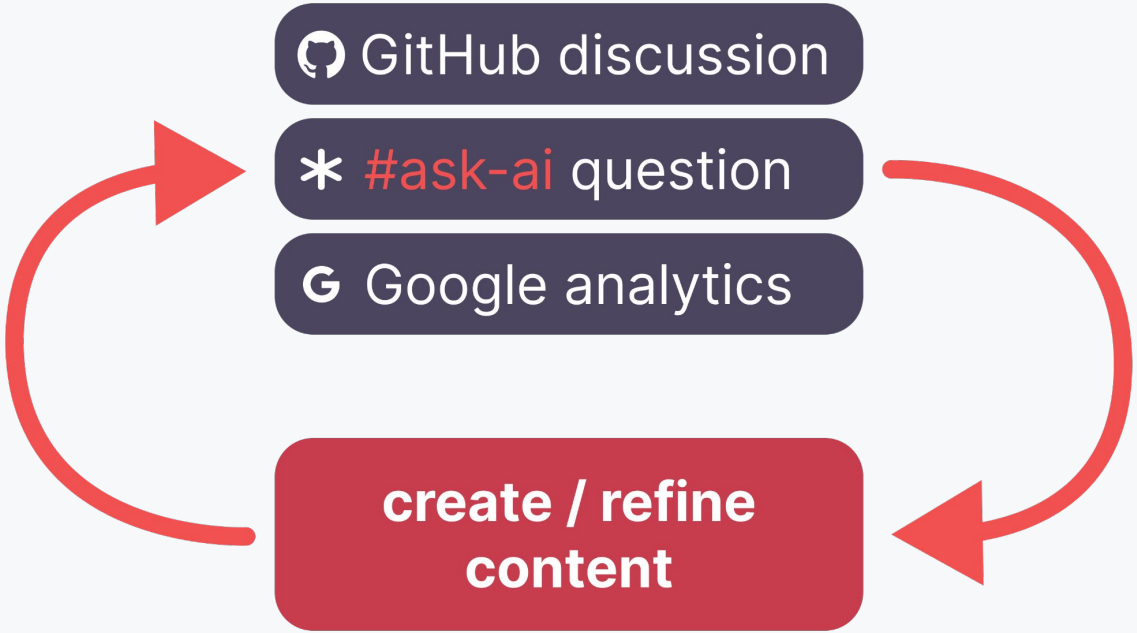
This skill provides comprehensive guidance for writing dignified Python code including:

- Automatic Python version detection (3.10-3.13)
- LBYL exception handling patterns
- Modern type syntax (list[str], str | None)
- Pathlib operations and ABC-based interfaces
- CLI patterns and subprocess handling
- API design and code organization best practices



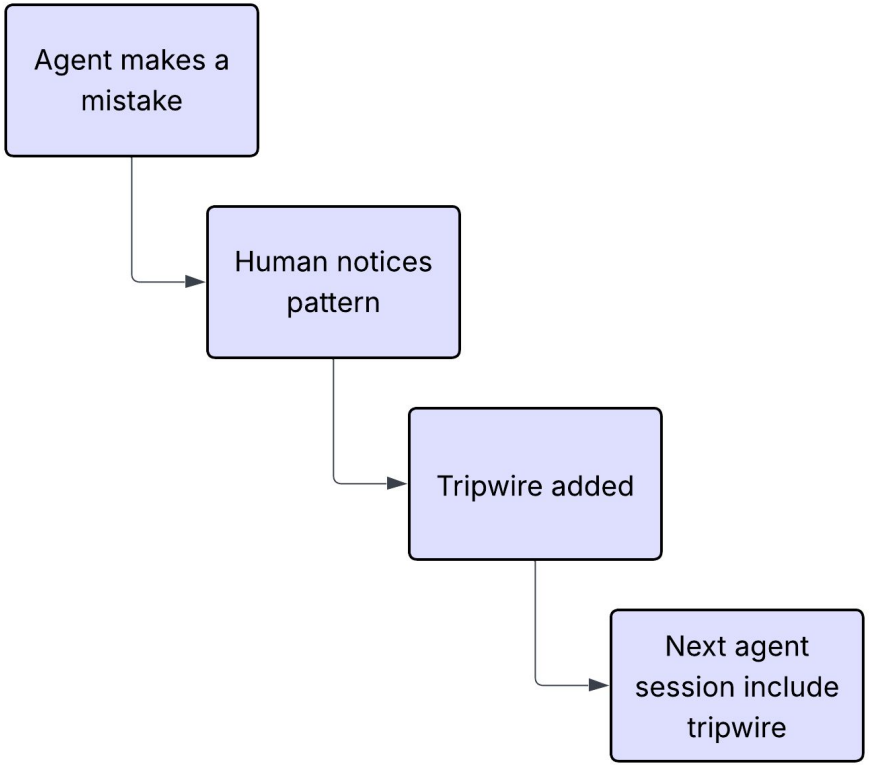
dagster

Context pipelines



dagster

Tripwires



dagster

Artifacts

Upload your screenshot / image

```
---
title: Convention-Based Code Reviews
read_when:
  - adding a new code review to CI
  - understanding how code reviews work
  - modifying code review behavior
tripwires:
  - action: "using fnmatch for gitignore-style glob patterns"
    warning: "Use pathspec library instead. fnmatch doesn't support **
recursive globs. Example: pathspec.PathSpec.from_lines('gitignore',
patterns)"
---
```

dagster

Artifacts

Upload your screenshot / image

```
---
title: Convention Based Code Reviews
read_when:
  - adding a new code review to CI
  - understanding how code reviews work
  - modifying code review behavior
tripwires:
  - action: "using fnmatch for gitignore-style glob patterns"
    warning: "Use pathspec library instead. fnmatch doesn't support **
recursive globs. Example: pathspec.PathSpec.from_lines('gitignore',
patterns)"
---
```

Navigation

Actionable Guidance

dagster

Demo

Summary

dagster-io / **dagster-claude-plugins**

master | 1 Branch | 0 Tags | Go to file | Add file | Code

cmpadden Merge pull request #7 from dagster-io/colton/feat/remote-dbt-recommen... a3ecd2c · last week | 33 Commits

.claude-plugin	Ensure yes is piped into create- commands; refactor fold...	last week
.github	Add .github/PULL_REQUEST_TEMPLATE.md	last week
plugins	Update convention to recommend remote dbt project con...	last week
README.md	Restructure README.md by plugin	last week

README

Claude Plugins for Dagster

A collection of Claude Code plugins for building workflows and data pipelines using Dagster.

Installation

Add this marketplace to Claude Code:

```
/plugin marketplace add dagster-io/dagster-claude-plugins
```

Install a plugin:

```
/plugin install dg@claude-plugins-dagster
```

To update your marketplace plugins:

```
/plugin marketplace update
```

Deep Dive | dagster

Dagster Running Dagster: How We Use Compass for AI Analytics

Nick Roach, Dagster
Anil Maharjan, Dagster
Alex Noonan, Dagster

Feb 17 | 12pm ET

February 17, 2026

Dagster Running Dagster: How We Use...

In this Deep Dive, we're joined by Dagster Analytics Lead Anil...

Welcome to Dagster University

Learn how to leverage Dagster and build robust, production-ready data platforms with ease.

Courses

dagster university — Dagster & ETL

Dagster & ETL
Course
5.0 (2 reviews)
Learn how to ingest data to power your assets. You'll build custom pipelines and see how to use Embedded ETL and Dagster Components...

dagster university — Dagster Essentials

Dagster Essentials
Course
4.8 (173 reviews)
In this course, learn the basics of Dagster, including how to represent a data pipeline as the data assets it produces and orchestrate a...

dagster university — Dagster & dbt

Dagster & dbt
Course
4.9 (20 reviews)
Learn how to integrate and orchestrate dbt projects with Dagster. You'll load dbt models into Dagster as assets, build dependencies,...

View all courses

dagster

dagster

Thank you!