RPCL

Robert Palazzini
Private markets operations & technology

TECHNICAL IMPLEMENTATION TOOLKIT

Building a Private Equity Analytics and Operations Platform with Claude

The Technical Implementation Guide: a reference architecture, the key decisions and a gated method for building a governed platform on Microsoft Azure, with Claude as part of the engineering team.

FOR
CTOs & engineering teams

VERSION
4.0 · November 2026

AUTHOR
Robert Palazzini

WEB
robertpalazzini.com

AT A GLANCE

What this guide gives you

A foundation for your technical team to build a private equity analytics and operations platform with Claude: the reference architecture, the key decisions, the controls around Claude, the method, and the decisions to make on day one.

10
gated phases, from discovery to repeatable delivery

25
sections covering method, architecture, modules and operations

10
day-one decisions to record before code is written

10
private equity metrics defined and tested, from DPI to net IRR

10
architecture decisions, with options, trade-offs and the reference choice

5
figures: the reference architecture, a workflow data model and three example screens

THE PRINCIPLE BEHIND THIS GUIDE

Treat Claude as a fast, capable engineer with no memory between sessions. Almost every practice here exists to give Claude, and new people, durable written context: operating rules, decisions, architecture and session logs, all kept in the repository.

Built for

CTOs and heads of technology, engineering leads and delivery partners, developers working with Claude, and IT and security owners. Leadership teams should start with the companion Strategic Guide.

Defining features

Absorb, don't sprawl: one platform, one origin, one sign-in, with modules mounted inside it, including process monitoring and workflow. **AI document extraction:** models in the Manager's own Microsoft Foundry project read external documents into validated, traceable data.

CONTENTS

Contents

01	Purpose and how to use this guide	05
02	Build methodology and engagement model	06
03	What you will build	09
04	Architecture decisions	10
05	Prerequisites and organisational readiness	15
06	Enabling and governing Claude	17
07	The private equity data domain	20
08	Data architecture and Azure infrastructure	24
09	Security, identity and permissions	26
10	Repository strategy and Git practices	28
11	Database architecture and the Alembic process	30
12	Data pipelines, AI document extraction and Azure Data Factory	32
13	Data quality and governance	41
14	Application architecture and the module pattern	43
15	The process monitoring and workflow module	44
16	Analytics and reporting surfaces	48
17	User interface and usability standards	49
18	Build, deployment, environments and CI	55
19	Testing strategy	57
20	Day-to-day workflow with Claude	59
21	Documentation and knowledge management	60
22	Run, support and handover	61
23	Step-by-step build sequence	63
24	Lessons learned and pitfalls to avoid	66
25	The Toolkit	68

04

Architecture decisions

Section 3 describes what the reference platform contains. This section explains the main choices behind it. The Manager is assumed to run on Microsoft 365 and Azure, with Azure SQL as the warehouse, so the decisions here are about how to use the Microsoft platform, not whether to.

Each decision shows the realistic options, what each is good at, what to watch for, the choice the reference build made and when you might reasonably choose differently. Treat the reference choice as a sensible default, not a rule. Record your own choice for each as a decision-log entry (template in the Toolkit), with the reason. Where a decision overlaps a day-one decision in Section 5.3, record it once.

4.1 Hosting the application and API

The platform is one backend API and one frontend, built into one container image (Section 14). The question is where that image runs.

Option	Good at	Watch for
Azure Container Apps	Serverless containers with revisions, traffic splitting, managed identity, custom domains and scale rules; scheduled and event-driven jobs in the same environment	Virtual network integration is set when the environment is created, so decide on networking first (Section 4.8)
Azure App Service (containers)	A mature, familiar web hosting model with deployment slots and built-in authentication options	Less natural for background jobs; scaling and revisions are coarser
Azure Functions	Event-driven and scheduled work, small APIs	Not a good home for a full web application and its frontend
Azure Kubernetes Service	Full control for large, multi-service estates	Significant operational overhead for a small team; rarely justified here

Reference approach: one Azure Container App for the platform, with Container Apps jobs in the same environment for scheduled work such as extraction batches and workflow reminders. Revisions make rollback a traffic switch.

Choose differently if: the Manager's IT standard is App Service and its team already operates it well, or a platform team already runs Kubernetes for other workloads.

4.2 Ingesting and orchestrating data

Data arrives from the accounting system or administrator, portfolio companies, CRM and files. Something must schedule, move and land it.

Option	Good at	Watch for
Azure Data Factory	Managed connectors, the self-hosted integration runtime for on-premises and private sources, scheduling, retries and monitoring	Low-code pipelines are harder to test; keep business logic out of them

END OF PREVIEW

Get the full Technical Implementation Toolkit

The full 69-page Technical Implementation Guide, with all ten architecture decisions, the Foundry extraction implementation and the workflow module design, plus the pack of editable files: Claude Code starter, templates, checklists, architecture decision records, starter prompts, a build tracker, and working extraction and workflow starter code.

robertpalazzini.com/playbook