



Greenfield vs Brownfield in BMAD Method — Step-by-Step Guide

5 min read · Sep 21, 2025



Vishal Mysore

Learn how the BMAD Method treats Greenfield (new) and Brownfield (existing) projects differently: analyst-led discovery, PRD-First vs Document-First flows, brownfield templates, risk controls, and concrete commands & tasks to get started.

• • •

What “Greenfield” and “Brownfield” mean in BMAD (short)

- **Greenfield:** new software projects built from scratch.
- **Brownfield:** enhancements, integrations, or modifications to an existing codebase.

These definitions are the starting point for two distinct BMAD workflows and templates.

• • •

Greenfield projects are built from scratch and follow a clean planning workflow (project brief → PRD → architecture → development). Brownfield projects require documenting the **existing** system first (using the Analyst agent’s `*document-project` task), then creating brownfield PRDs, architecture adapters and tight QA/regression coverage. BMAD provides dedicated brownfield templates, tasks, and a Test Architect role for risk management — the workflows differ up front but converge to the same SM → Dev → QA cycle after planning

• • •

Key differences in BMAD approach

Greenfield development (how BMAD recommends you start)

- Start with ideation and market research using the Analyst agent.
- Follow a clean planning workflow: **project brief → PRD → architecture → development.**

- Use standard greenfield templates (for example `prd-tmpl` and `architecture-tmpl`) to keep artifacts consistent.

Brownfield development (how BMAD changes the front-of-work)

- **First, document the existing system** — that is the critical first step in BMAD brownfield workflows, using the `*document-project` task.
- Use **specialized brownfield templates** such as `brownfield-prd-tmpl` and `brownfield-architecture-tmpl`.
- Perform comprehensive analysis of codebase, constraints, and integration points before making changes.

. . .

Brownfield — specific features & tasks you must know

BMAD includes brownfield-oriented tasks and two recommended workflow approaches depending on scope and size:

Two workflow approaches

- **PRD-First** (recommended for large codebases): Create the PRD first, then document only the relevant areas.
- **Document-First** (recommended for smaller projects): Document everything first, then create the PRD.

Specialized brownfield tasks

- `*create-brownfield-prd` — analyzes the existing system and creates enhancement requirements.
- `*create-brownfield-architecture` — designs an integration strategy with the existing system.
- `*create-brownfield-epic` and `*create-brownfield-story` — used for smaller scoped changes.

Enhanced risk management

- Brownfield projects require a **Test Architect (QA)** agent to identify regression risks, legacy dependencies, and breaking changes and to design comprehensive regression strategies

Brownfield — step-by-step recommended flow (practical)

BMAD breaks brownfield work into phases (choose the path based on scope):

Enhancement classification — classify the scope: single story (<4 hours), small feature (1–3 stories), or major enhancement (multiple epics).

Choose a workflow path

- Major enhancements → Full workflow (documentation → PRD → architecture → validation → shard docs → development).
- Smaller enhancements → Use `*create-brownfield-epic` or `*create-brownfield-story`.

Phase details (Full workflow highlights)

- **Documentation analysis** — check existing docs; run `@architect *document-project` if needed.
- **Requirements planning** — `@pm *create-brownfield-prd`.
- **Architecture planning** — `@architect` with `brownfield-architecture-tmpl` if required.
- **Validation & preparation** — PO validation, sharding of documents for IDE work.
- **Development cycle** — create stories (SM → Dev → QA) and implement with regression suites.

Safety steps (mandatory)

- Include Test Architect tasks: `@qa *risk`, `@qa *design`, `@qa *trace`, and `@qa *nfr` to identify regression risks, design regression tests, map requirement traces, and validate non-functional requirements.

Two Approaches for Brownfield Projects in the BMAD Method: PRD-First vs Document-First

In the BMAD Method, brownfield projects (working with existing codebases) can follow two distinct workflows: **PRD-First** and **Document-First**. Both are valid, and the choice depends on project size and complexity.

. . .

PRD-First Approach (Recommended for Large Codebases)

When to use: Large codebases, monorepos, or when you already know exactly what you want to build.

Workflow:

1. **Create PRD first** to define requirements.
2. **Document only relevant areas** of the codebase, based on what the PRD specifies.
3. This is **more efficient** for big systems, since it avoids documenting unused code.

Key advantage: The PRD identifies which parts of a monorepo or large codebase actually need documentation, allowing the architect to focus only on relevant modules and skip unrelated areas.

. . .

Document-First Approach (Good for Smaller Projects)

When to use: Smaller codebases, unknown systems, or exploratory changes.

Workflow:

Document the entire system first using:

```
@analyst *document-project
```

Create the PRD with full context from the documented system.

This is **more thorough** because it captures everything — ideal for projects where you need full visibility.

. . .

Technical Implementation & Guidance

- Both approaches are supported in the **BMAD knowledge base**, with clear decision logic on when to use each.
- The framework routes teams based on **project size and complexity**:
- **PRD-First** for efficiency in large systems.
- **Document-First** for comprehensive understanding in smaller or less known systems.

. . .

Key Takeaway

The choice between PRD-First and Document-First is about **efficiency vs scope management**.

- **PRD-First** prevents documentation bloat in large systems by narrowing focus to only the areas that will be modified.
- **Document-First** ensures full understanding but may generate excessive documentation for big codebases.

👉 Importantly, **both approaches ultimately converge to the same BMAD development workflow** once planning is complete.

. . .

Practical commands & tips

- To flatten a repo for analysis:

```
npx bmad-method flatten
```

This produces a single-file flattened view of the codebase to improve large-context LLM analysis.

- In an LLM-friendly IDE (e.g., Claude Code), choose the **Architect** agent and run the `*document-project` task to systematically produce artifacts that other agents can use. Validate outputs and remove misleading or unnecessary information.
- Example PM flow in a Gemini Web context (if you upload flattened code):

```
@pm
*create-brownfield-prd
```

(Use this when following the PRD-First approach.)

Example: Analyst agent manifest

The BMAD source lists an Analyst/Business Analyst agent definition with tasks and templates that indicate it is used for market research, brainstorming, and **documenting existing projects**. A condensed reference excerpt:

```
id: analyst
title: Business Analyst
icon: 🇮🇹
whenToUse: Use for market research, brainstorming, competitive analysis, creati
persona:
  role: Insightful Analyst & Strategic Ideation Partner
tasks:
  - create-deep-research-prompt.md
  - create-doc.md
  - document-project.md
  - facilitate-brainstorming-session.md
templates:
  - brainstorming-output-tmpl.yaml
```

This manifest shows the Analyst agent includes `document-project.md` among its tasks, which is the canonical brownfield discovery action in BMAD.

!!! Very Very Very Critical tip

Producing **high-quality contextual artifacts** — clear docs, sharded code summaries, and dependable flattened views — is the single most important activity before agentic coding in brownfield projects. BMAD recommends validating these artifacts carefully so downstream agents don't act on misleading information.

. . .

One-page checklist

- Classify enhancement: story / small feature / major enhancement.
- If brownfield: run `@analyst *document-project` (or `npx bmad-method flatten`) and validate artifacts.
- Choose workflow: PRD-First (large) or Document-First (small).

- Create PRD (@pm *create-brownfield-prd) or Epic (@pm *create-brownfield-epic) as appropriate.
- Architect designs adapters/integration strategy via *create-brownfield-architecture .
- QA/Test Architect runs @qa *risk, @qa *design, @qa *trace, @qa *nfr .
- Implement SM → Dev → QA cycle and keep audit trails for agent outputs



Written by Vishal Mysore

743 followers · 4 following

Holder of multiple patents in AI and software engineering. Passionate about building scalable systems, optimizing performance, & driving AI-powered innovation.

No responses yet



Bgerby

What are your thoughts?