Spec Driven Development with GitHub SpecKit

AI Assist Generated Content

What is GitHub SpecKit?

GitHub SpecKit is an open-source toolkit and methodology developed by GitHub to bring structure and discipline to AI-powered software development. It formalizes a process called **Spec-Driven Development (SDD)**, which aims to solve the problems of inconsistent, unpredictable, and hard-to-maintain code often generated by "vibe coding" (i.e., giving vague, iterative prompts to an AI assistant).

The core philosophy of SpecKit is to make the **specification** a living, central, and executable artifact that guides the entire development lifecycle. Instead of a human writing code and an AI assisting, the human first defines a detailed specification, and then the AI assistant uses that spec as its "source of truth" to generate plans, tasks, and the code itself.

Key Goals and Features:

- **Solves "Vibe Coding":** It replaces chaotic, ad-hoc prompting with a structured, gated workflow, ensuring the AI's output is aligned with clear business and technical goals.
- **AI Agent-Agnostic:** The toolkit is designed to work with various AI coding assistants, such as GitHub Copilot, Claude Code, Gemini CLI, and others.
- **Human-in-the-Loop:** It is not a fully autonomous system. It enforces explicit checkpoints where a human developer must review, refine, and approve the AI-generated artifacts (spec, plan, tasks) before proceeding.
- **Core Artifacts:** SpecKit establishes a set of key files in your repository to manage the process:
  - `constitution.md`: A high-level document that defines the "non-negotiable" principles for the project. This includes rules about coding standards, preferred frameworks, security requirements, and testing methodologies. The AI must adhere to this constitution when generating its plan.
  - `spec.md`: The feature specification. This file details the "what" and "why" of a feature, focusing on user stories, goals, user interactions, and edge cases, but *not* the technical implementation.

- `plan.md`: The technical implementation plan. Generated by the AI based on the `spec.md` and `constitution.md`, this file outlines the "how," including architecture, data models, file structures, and technology choices.
- `tasks.md`: A detailed, step-by-step checklist of executable tasks the AI will follow to implement the plan. Each task should be small, testable, and verifiable.

Step-by-Step Workflows Using SpecKit

The SpecKit workflow is divided into distinct, gated phases, often initiated using slash commands (like `/specify`, `/plan`) within your AI assistant's chat interface.

Workflow 1: Starting a New (Greenfield) Project

This workflow guides you from an empty folder to a fully scaffolded, spec-driven project.

Step 1: Installation and Initialization First, you must install the SpecKit CLI (often via a tool like `uv`). You then initialize your project.

1. **Install SpecKit:** (Follow the official repository's instructions). A common method shown is using `uvx`: `uvx --from git+https://github.com/github/spec-kit.git specify init <YOUR_PROJECT_NAME>`
2. **Initialize Project:** This command creates the necessary directory structure (e.g., `.github/` for agent prompts, `.specify/` for specs) and the core artifact files.

Step 2: Define the Constitution Before you build anything, you set the rules.

1. Navigate to the generated `/memory/constitution.md` file.
2. Edit this file to define your project's standards. This is your one-time setup to guide all future AI work.
   - *Example:* "All new API endpoints must include unit tests." or "Use React with TypeScript for all front-end components."

3. You can also use an AI command like `/speckit.constitution` to help you draft these principles.

Step 3: Specify the Feature (`/specify`) Now, you define your first feature, focusing on *what* it is, not *how* it's built.

1. In your AI assistant, use the specify command: `/speckit.specify`
2. Provide a high-level prompt describing your feature.
   - *Example:* `/speckit.specify Build a user authentication system that allows users to sign up with an email and password and log in.`
3. The AI will generate a detailed `spec.md` file, including user stories, acceptance criteria, and a "Needs Clarification" section.

Step 4: Review and Clarify the Spec This is a critical human-in-the-loop step.

1. Open the newly created `spec.md` file.
2. Manually review the AI's interpretation of your feature. Pay close attention to the "Needs Clarification" or "Edge Cases" sections.
3. Use the `/speckit.clarify` command or simply chat with your AI to answer these questions and refine the spec until it's accurate and complete.

Step 5: Generate the Technical Plan (`/plan`) Once the "what" is locked, you generate the "how."

1. In your AI assistant, use the plan command: `/speckit.plan`
2. The AI will read the `constitution.md` and the approved `spec.md`.
3. It will generate a `plan.md` file, which includes the proposed architecture, data models, API endpoints, and a list of files to be created or modified.

Step 6: Review the Plan Again, you must review the AI's technical proposal.

1. Open `plan.md` and any related files (e.g., `data-model.md`).

2. Verify that the technical approach aligns with your constitution and best practices.
3. Request changes from the AI until you approve the plan.

Step 7: Generate Tasks (`/tasks`) Break the plan into small, actionable steps.

1. In your AI assistant, use the tasks command: `/speckit.tasks`
2. The AI will read the `plan.md` and generate a `tasks.md` file, which is a detailed checklist.
   - *Example:* "1. Create file `src/models/user.js`..." "2. Add `email` and `passwordHash` fields to User schema..."

Step 8: Implement the Tasks (`/implement`) This is where the AI writes the code.

1. In your AI assistant, use the implement command: `/speckit.implement`
2. The AI will now execute every task from `tasks.md` one by one, writing and saving the code to your file system.
3. Your role is to monitor the process, answer any questions the AI has, and intervene if it gets stuck.

Step 9: Test and Review Once implementation is complete, you review the final output, run tests, and merge the code.

Workflow 2: Adding a New Feature to an Existing (Brownfield) Project

This workflow assumes SpecKit is already initialized and a codebase exists. The process is largely the same but focuses on a single new feature.

Step 1: Installation and Initialization First, you must install the SpecKit CLI (often via a tool like `uv`). You then initialize your project.

Step 2: Start a New Feature Branch: As per good Git practice, create a new branch for your feature (e.g., `feature/add-profile-page`).

Step 3: Specify the New Feature (`/specify`): Use the `/speckit.specify` command, but this time, be specific about how it interacts with the existing system.

Example: `/speckit.specify Create a new user profile page that displays the user's email from the existing 'User' model and allows them to update their 'firstName' and 'lastName' fields.`

Step 4: Review and Clarify Spec: Review the generated `spec.md` for this new feature. Ensure the AI understands the *existing* components it needs to use.

Step 5: Generate the Plan (`/plan`): Run `/speckit.plan`. The AI will (critically) analyze your existing codebase *in addition* to the new spec and constitution. Its plan will detail which *existing* files to modify and which *new* files to create.

Step 6: Review the Plan: This is the most important step for brownfield projects. Scrutinize the plan to ensure the AI isn't proposing destructive or nonsensical changes to your existing code.

Step 7: Generate Tasks (`/tasks`): Run `/speckit.tasks` to get the detailed checklist for modifying and adding files.

Step 8: Implement and Monitor (`/implement`): Run `/speckit.implement`. Closely watch as the AI modifies your code.

Step 9: Test and Merge: Thoroughly test the new feature and its integration with the existing application before merging.

This video provides a tutorial on using Spec Kit with an AI agent to build an application, which can help visualize the workflows described. A tutorial for using GitHub Spec Kit