Analyze the SDLC

DevOps, Agile vs. Waterfall, and Where GitHub Fits

Concise, practical study guide

Learning Objectives

- Understand what DevOps is (origin and the CALMS model).
- Contrast Waterfall and Agile, with trade-offs and use-cases.
- Recognize common Agile frameworks (Kanban, Scrum, XP) and practices.
- Map the DevOps lifecycle phases and where GitHub capabilities fit.

1 DevOps at a Glance (CALMS)

CALMS pillars

- **Culture** — cross-functional collaboration, shared responsibility for outcomes.
- **Automation** — remove toil; make builds, tests, and releases repeatable.
- **Lean** — eliminate waste, optimize flow of value, focus on small batch sizes.
- **Measurement** — observe with meaningful metrics; improve constraints.
- **Sharing** — open knowledge flow across teams; blameless learning.

Typical supporting metrics (examples). Deployment Frequency, Lead Time for Changes, Change Failure Rate, and Mean Time to Restore. These help validate improvements across CALMS dimensions.

2 Waterfall SDLC (Classic Model)

2.1 Phases

1. **Requirements** — clarify scope, risks, constraints, acceptance criteria.
2. **Design** — architecture, interfaces, data models, UX flows.
3. **Implementation** — coding and integration of components.
4. **Verification** — testing against requirements/specification.
5. **Deployment & Maintenance** — release, operate, and support.

2.2 Strengths & Limitations

Strengths	Limitations
Clear stage gates and artifacts; easier regulatory traceability.	Struggles with frequent change; feedback comes late.
Predictable budgets and timelines for stable scope.	Risk of building the “wrong” thing before validation.
Well-suited for low-uncertainty domains.	Long cycles discourage continuous learning and adaptation.

3 Agile Fundamentals

3.1 Values (Manifesto, summarized)

- **Individuals and interactions** over processes and tools.
- **Working software** over comprehensive documentation.
- **Customer collaboration** over contract negotiation.
- **Responding to change** over following a plan.

3.2 Common Frameworks and Practices

- **Kanban** — visualize work, limit work-in-progress (WIP), manage flow.
- **Scrum** — timeboxed sprints, roles (PO/SM/Team), ceremonies, backlog.
- **Extreme Programming (XP)** — TDD, pair programming, continuous integration, refactoring.

3.3 Agile vs. Waterfall (Key Contrasts)

Agile Principle	Why It Matters
Individuals & interactions	Optimize collaboration and flow of value.
Working software	Ship usable value sooner for earlier feedback.
Customer collaboration	Continuous feedback tightens product–market fit.
Responding to change	Embrace inevitable change without heavy ceremony.

4 DevOps Lifecycle (Where GitHub Fits)

4.1 End-to-End View

Lifecycle Phase	Typical GitHub Capabilities
Continuous Planning	Issues, labels, milestones; **Projects** for boards; Discussions; Wikis; organization-level planning across repositories.
Code & Review	Pull Requests, code review workflows, branch protections, required reviews, status checks, **Codespaces** for dev environments, **Copilot** for assistance.
Build & Test (CI)	**Actions** workflows for build/test, matrix builds, caching; packages, reusable workflows; marketplace actions.
Security & Quality	Secret scanning, dependency scanning/updates, code scanning (SAST), branch protections, required checks.
Continuous Delivery/Deployment	Actions for deployment to environments; environment protection rules; Pages for static sites; registries via **Packages**.
Operate & Learn	Release notes; Discussions; issues for incidents/postmortems; CI signals surfaced in PRs; artifacts and logs for traceability.

4.2 Minimal Practical Checklist

- ✓ Track work with Issues, milestones, and a GitHub Projects board.
- ✓ Enforce branch protections and required status checks.
- ✓ Build and test with Actions on pull requests and default branch.
- ✓ Enable secret scanning, code scanning, and dependency updates.
- ✓ Use environments and required approvals for production deploys.
- ✓ Publish release notes and capture post-incident learnings.

Glossary (Quick Reference)

CALMS	Culture, Automation, Lean, Measurement, Sharing.
CI/CD	Continuous Integration / Continuous Delivery (or Deployment).
WIP	Work in Progress, often limited in Kanban to improve flow.
Sprint	Timeboxed iteration in Scrum (e.g., 1–2 weeks).

Tip. Start small: choose one value stream, instrument its flow (lead time, deployment frequency, change failure rate, and time to restore), and iterate on constraints. Use GitHub branch protections and Actions to create crisp, automated feedback early.

This handout was generated from concise study notes on SDLC, Agile vs. Waterfall, and DevOps lifecycle mapping to GitHub.