

This is a copy of a chat between Claude and Anand. Content may include unverified or unsafe content that do not represent the views of Anthropic. Shared snapshot may contain attachments and data not displayed here. [Report](#)

## A Claude code tutorials video resources

I'd be happy to help you find resources for learning Claude Code! Let me search for the latest video tutorials and resources available.

W web\_search

Great! I found some excellent video and tutorial resources for learning Claude Code. Here are the best options:

## Video Tutorials

**1. DeepLearning.AI Course - "Claude Code: A Highly Agentic Coding Assistant"** This comprehensive course covers refactoring notebooks, removing redundant code, creating dashboards with web applications, and building front-end applications using Claude Code with Figma MCP servers . It's a systematic course that covers best practices even for experienced users.

- Link: <https://learn.deeplearning.ai/courses/claude-code-a-highly-agentic-coding-assistant/>

**2. Beginner's Tutorial - Build a Movie App in 15 Minutes** This complete beginner's tutorial shows how to set up Claude Code and add a watchlist feature to a movie discovery app, covering plan mode, CLAUDE.md files, and more . Available on YouTube with a written guide.

- Link: <https://creatoreconomy.so/p/claude-code-beginners-tutorial-build-a-movie-app-in-15-minutes>

**3. Sid Bharath's Complete Guide** This tutorial teaches Claude Code by building a complete personal finance tracker web app, covering all features, usage patterns, and tips and tricks . Includes video content for visual learners.

- Link: <https://www.siddharthbharath.com/claude-code-the-complete-guide/>

## Written Resources

### Official Anthropic Documentation:

- **Best Practices Guide:** <https://www.anthropic.com/engineering/claude-code-best-practices>
- **Official Docs:** <https://docs.claude.com> (check the Claude Code section)

### Comprehensive Tutorials:

- **DataCamp Tutorial:** Practical examples using Supabase Python SDK
- **Codecademy Tutorial:** Setup, generation, debugging, and documentation
- **ClaudeLog Tutorial:** Essential commands and workflow optimization

These resources cover everything from basic setup to advanced features like MCP integration, GitHub PR reviews, and multi-agent workflows. I'd recommend starting with either the DeepLearning.AI course or the 15-minute movie app tutorial depending on your learning style!

A What are the topics to learn ? Code , roles , skills , ...

Let me search for more specific information about the learning curriculum and topics covered in Claude Code.

W web\_search

Perfect! Here's a comprehensive breakdown of the key topics and concepts you need to learn for Claude Code:

## Core Topics to Learn

### 1. Fundamental Concepts

- **Basic Commands & Interface:** Terminal usage, slash commands, and Claude Code CLI
- **Claude.md Files:** Creating persistent memory (user, project, and local memory files) to store preferences, coding standards, and context
- **Plan Mode vs Execute Mode:** Understanding when Claude plans versus when it executes code

- **Context Management:** Using `/clear`, `/compact`, and managing Claude's context window efficiently

## 2. Skills (Agent Skills)

Skills are modular capabilities that extend Claude's functionality through organized folders containing instructions, scripts, and resources, with each Skill consisting of a SKILL.md file with instructions that Claude reads when relevant

### Types of Skills:

- **Personal Skills:** Available across all your projects (`~/.claude/skills/`)
- **Project Skills:** Shared with your team (`.claude/skills/`)
- **Plugin Skills:** Bundled with installed plugins

### Popular Skill Categories:

- Document creation (Word, PowerPoint, Excel, PDF)
- Test-driven development
- Git workflows and worktrees
- Code quality analysis
- Web app testing with Playwright
- Security analysis

## 3. Advanced Features

**Hooks:** Shell commands that trigger at specific events in Claude's lifecycle, enabling automation of everything from running tests to formatting code and even calling other AI agents

**Sub-Agents:** Design and build a team of specialized AI assistants, each with its own unique context, tools, and system prompt, to handle specific tasks like code reviews, debugging, or security analysis

### MCP (Model Context Protocol):

- Integration with external APIs and services
- Creating custom MCP servers
- Using Figma MCP for design-to-code workflows

## 4. Development Workflows

### Project Types You'll Build:

- RAG chatbots with front-end to back-end implementation, including refactoring code, writing tests, and using GitHub integration for pull requests
- Jupyter notebook refactoring, removing redundant code, and creating dashboards with web applications
- Full-stack applications
- Data analysis pipelines

### Key Workflow Skills:

- Parallel sessions management
- GitHub integration (PRs, issues, commits)
- Test-driven development
- Code refactoring
- Debugging strategies

## 5. Professional Skills & Roles

### Who This Is For:

- Software developers and engineers
- AI engineers
- Data scientists
- Application developers
- Technical leads managing AI-assisted teams

### Prerequisites:

- Software development experience
- Familiarity with version control (Git)
- Python proficiency (for most courses)
- Next.js or React (for web projects)

## 6. Cost & Performance Optimization

- Understanding tokenomics
- Managing context windows
- Best practices for efficient interactions
- Labor economics of AI development

## 7. Security & Best Practices

- Secure command access with `allowed-tools`
- Understanding code execution risks
- Version control and collaboration
- Testing and validation workflows

## Recommended Learning Path

1. **Start with basics:** Commands, interface, basic project creation
2. **Learn Skills:** How to create, use, and manage them
3. **Master memory management:** CLAUDE.md files and context
4. **Explore advanced automation:** Hooks and sub-agents
5. **Integrate with tools:** GitHub, MCP servers, testing frameworks
6. **Build real projects:** Apply everything in hands-on scenarios

The key insight is that Claude Code isn't just a chatbot—it's a platform for building AI-driven development workflows with extensible, reusable components (Skills) that can work together to automate complex tasks!

- A I want to review a pull request, want to define some set of rules to review..  
What are the details in need to do and get help from Claude code to review pull requests

Great question! Let me search for specific information about using Claude Code for pull request reviews and best practices.

W web\_search

Excellent! Here's a comprehensive guide for setting up Claude Code to review pull requests with custom rules:

## Setup Methods

You have **three main approaches**:

### 1. GitHub Actions (Automated)

The easiest setup is through Claude Code in the terminal - just run `/install-github-app`, which guides you through setting up the GitHub app and required secrets

### 2. Terminal Command (Manual)

Use the `/security-review` command for ad-hoc reviews

### 3. IDE Integration

Review staged diffs directly in VS Code or JetBrains

---

## Defining Your Review Rules

### Method 1: CLAUDE.md File (Recommended)

Create a `CLAUDE.md` file at the root of your repository with your review rules:

```
markdown
```

```
# Code Review Guidelines
```

```
## Project Context
```

- Framework: [Your framework]
- Data layer: [e.g., Prisma + Postgres]
- Architecture: [Your architecture]

```
## Primary Goals
```

- Security and correctness over style
- Flag input validation, auth, access control, and database queries
- Check for proper error handling
- Validate API security measures
- Ensure compliance with internal coding standards

```
## Non-Goals
```

- Do not reformat files or rename symbols unless asked
- Avoid commenting on unchanged files

- Ignore style preferences (handled by formatters)

### ## When Reviewing PRs

- Focus only on files that changed
- Propose minimal, targeted fixes with line references
- If risk is low, say so briefly
- Check for common vulnerability patterns

Example template: "Data layer is Prisma + Postgres. Primary goals - Security and correctness over style. Flag input validation, auth, access control, and database queries. When reviewing PRs - Focus on files that changed, propose minimal, targeted fixes with line references"

## Method 2: Custom Workflow Prompts

Use the prompt parameter in the workflow file to provide workflow-specific instructions

## Step-by-Step Setup

### Prerequisites:

- GitHub admin access
- Anthropic API key
- Repository with PR workflow

### GitHub Actions Setup:

#### 1. Install GitHub App

```
bash

# In Claude Code terminal
/install-github-app
```

#### 2. Add API Key to Secrets

- Go to: Settings → Secrets and variables → Actions
- Add `ANTHROPIC_API_KEY` as a repository secret

#### 3. Create Workflow File Create `.github/workflows/claude.yml`:

```
yaml
```

```

name: Claude Code Review
on:
  pull_request:
    types: [opened, synchronize]
  issue_comment:
    types: [created]

permissions:
  contents: read          # Read code
  pull-requests: write    # Post PR comments
  # Add contents: write if you want Claude to push commits

jobs:
  claude-review:
    runs-on: ubuntu-latest
    steps:
      - uses: anthropics/claude-code-action@v1
        with:
          anthropic-api-key: ${ secrets.ANTHROPIC_API_KEY }
  ...

```

#### **\*\*4. Trigger Reviews\*\***

- **\*\*Automatic\*\***: Reviews trigger when PRs open
- **\*\*Manual\*\***: Comment `@claude` in any PR to request analysis

---

#### **## \*\*Custom Review Focus Areas\*\***

Specific, targeted prompts produce better review results than vague requests - fo

#### **### \*\*Example Prompts:\*\***

##### **\*\*Security Review:\*\***

...

"Security review the authentication module. Check for token leakage, session fixation, and access controls"

...

##### **\*\*Quality Review:\*\***

...

"Review PaymentProcessor.js for error handling, input validation,



```
and edge cases. Suggest specific improvements"
```

```
```
```

```
**Architecture Review:**
```

```
```
```

```
"Review the shopping cart feature structure. Assess separation  
of concerns and suggest improvements"
```

```
```
```

```
**Legacy Code:**
```

```
```
```

```
"This code feels fragile. Identify technical debt and suggest  
a low-risk refactoring strategy"
```

---

## Security-Focused Reviews

The `/security-review` command uses a specialized security-focused prompt that checks for common vulnerability patterns including SQL injection, XSS, authentication issues, and more

Common patterns checked:

- Input validation issues
- SQL injection vulnerabilities
- Cross-site scripting (XSS)
- Authentication/authorization flaws
- Remote code execution risks
- DNS rebinding vulnerabilities
- Token leakage
- Session fixation

---

## Best Practices

### Permissions (Least Privilege)

Comment-triggered runs that only read code and post PR comments can use contents: read and pull-requests: write - start read-only whenever possible

## Integration Workflow

Claude Code integrates well into team development processes - use for pre-PR self-review before submitting pull requests, or as a second opinion when unsure about human reviewer feedback

## Human-in-the-Loop

- AI reviews complement, not replace, human reviews
- Always have final human sign-off
- Use AI as an "additional set of eyes"

---

## Advanced Features

### Custom Skills for Reviews

Create a custom Skill in `.claude/skills/code-review/SKILL.md`:

```
markdown
```

```
# Code Review Skill
```

```
Review code changes focusing on:
```

1. Security vulnerabilities
2. Performance bottlenecks
3. Test coverage
4. Documentation completeness
5. Breaking changes

```
Always provide:
```

- Severity level (critical/high/medium/low)
- File and line references
- Suggested fix with code example

### Path-Specific Reviews

Configure workflows to trigger on specific file changes:

```
yaml
```

```
on:  
  pull_request:  
    paths:
```

```
- 'src/security/**'  
- 'src/api/**'
```

## What Claude Code Analyzes

Claude Code analyzes the entire diff context, understanding not just the changes being made but also how those changes fit into the broader codebase architecture and whether they follow established patterns and conventions

The review covers:

- Code quality concerns
- Security vulnerabilities
- Performance problems
- Architectural consistency
- Logic errors
- Bug identification
- Compliance with coding standards

## Testing Your Setup

1. Create a test PR
2. Tag `@claude` in a comment
3. Check action logs in the Actions tab
4. Verify PR comments appear [Start your own conversation](#)
5. Review permissions if issues occur