

Claude code

WORKSHOP AND CURRICULUM



Workshop overview

The main idea behind the **Claude Code Workshop** is that, in the era of AI, it is not enough to simply know how to build AI applications or agents. More importantly, we need to understand **how to effectively work with, manage, and control AI agents**.

The workshop will focus heavily on **complete, hands-on learning**, covering the entire journey from building AI-powered systems to working with autonomous agents, handling their workflows, debugging them, and integrating them into real-world development environments.

The goal is to make the workshop useful for **all levels of learners—from beginners to experienced developers**—with practical exercises, real-world use cases, and progressively advanced projects.

Contents

Part I — Foundations

- 1. What Claude Code is, and what it is not
 - 1.1 Four things that must be explicit

- 1.2 The control stack
- 1.3 Where the work runs
- 1.4 Appropriate first tasks

2. Install, verify and authenticate

- 2.1 Requirements
- 2.2 Installation
- 2.3 Verify before you log in
- 2.4 Windows: native or WSL

3. Surfaces, projects and boundaries

- 3.1 Set up the IDE
- 3.2 Create the practice workspace
- 3.3 Monorepos and large trees

4. Modes, models, effort and usage

- 4.1 The controls that matter first
- 4.2 The effort vocabulary, by surface
- 4.3 A decision rule for model and effort
- 4.4 Measure usage and context separately
- 4.5 What to optimise first

Part II — A Safe Working Method

5. Start and steer a session

- 5.1 Begin with an inspectable brief
- 5.2 Watch the beginning of unfamiliar work
- 5.3 Steer without restarting
- 5.4 Side questions, focus and recall
- 5.5 Voice input

6. Plan, comment, approve, execute

- 6.1 When to plan
- 6.2 The minimum complete plan
- 6.3 Review the plan like a contract
- 6.4 Approve the right scope
- 6.5 Require a completion report, then check it

7. Permissions, sandboxing and checkpoints

- 7.1 Permission modes
- 7.2 Rules, and the order they are evaluated in
- 7.3 Understand a prompt before approving
- 7.4 Sandboxing
- 7.5 Checkpoints and rewind

8. Protect work with Git and GitHub

- 8.1 A safe beginner sequence
- 8.2 Add a remote deliberately
- 8.3 An agent-assisted Git workflow
- 8.4 Automated review, and its limits
- 8.5 Recovery is broader than the repository

Part III — Project Knowledge and Context

9. Build an effective CLAUDE.md

- 9.1 Where instructions live
- 9.2 A useful project file
- 9.3 Initialize, then edit
- 9.4 An index, not a warehouse
- 9.5 Modular rules
- 9.6 Maintenance trigger

10. Global instructions and auto memory

- 10.1 Global instructions
- 10.2 Auto memory
- 10.3 A decision table
- 10.4 Memory audit

11. Engineer context deliberately

- 11.1 Inspect the window

- 11.2 Four ways to recover focus
- 11.3 Rewind, branch, fork, subtask
- 11.4 Reduce standing overhead

12. Sessions: resume, name, retain

- 12.1 Resume
- 12.2 Name by outcome
- 12.3 Retention, storage and privacy
- 12.4 When to start another session

Part IV — Extending Claude Code

13. Connect external tools with MCP

- 13.1 Scopes
- 13.2 Transports
- 13.3 A safe installation procedure
- 13.4 Context cost, measured rather than assumed
- 13.5 Organizational control

14. Environment variables and secrets

- 14.1 Why “Claude cannot see it” is unsafe
- 14.2 Layered controls
- 14.3 Incident response

15. Turn repeatable work into skills

- 15.1 Build one from observed work
- 15.2 Example project skill
- 15.3 Frontmatter that matters for safety
- 15.4 Skill security review

16. Delegate bounded work to subagents

- 16.1 Fresh context, forks, and why it matters
- 16.2 Where subagents live, and which one wins
- 16.3 An independent read-only reviewer
- 16.4 The frontmatter you should know
- 16.5 Limits and defaults
- 16.6 Skill or Subagent?
- 16.7 Delegation brief

17. Event-driven behaviour with hooks

- 17.1 Events
- 17.2 Handler types
- 17.3 A blocking example
- 17.4 Safer creation workflow
- 17.5 Threat model

18. Evaluate and install plugins

- 18.1 What a plugin can contain
- 18.2 Installation gate
- 18.3 Build instead of bundle

19. Output styles: Claude Code beyond software engineering

- 19.1 The built-in styles
- 19.2 Setting and creating a style
- 19.3 Where a style does and does not apply

Part V — Browser Use, Autonomy and Orchestration

20. Use Claude in Chrome safely

- 20.1 Prerequisites and connection
- 20.2 Use a dedicated browser profile
- 20.3 Escalate capability in stages
- 20.4 What the browser can carry out
- 20.5 Prompt injection and authenticated action
- 20.6 Computer use

21. Keep work moving with goals

- 21.1 What the evaluator can and cannot see
- 21.2 Anatomy of a safe goal
- 21.3 Behaviour you should expect
- 21.4 Context is the real multiplier

22. Dynamic workflows and ultracode

- 22.1 Entry points
- 22.2 Limits and permissions
- 22.3 Resume semantics, and why they matter for cost
- 22.4 When a workflow earns its cost
- 22.5 A research workflow pattern
- 22.6 Saving and distributing a workflow

23. Session-scoped scheduling with /loop

- 23.1 Three forms, three behaviours
- 23.2 Operational constraints
- 23.3 Lifecycle, and what actually stops a loop
- 23.4 The Monitor alternative
- 23.5 Suitable and unsuitable uses

24. Cloud routines and desktop scheduled tasks

- 24.1 What a routine actually is
- 24.2 Why a routine deserves a design review
- 24.3 Fire-payload text is untrusted by design
- 24.4 Before you activate one
- 24.5 Desktop scheduled tasks and headless automation

Part VI — Distributed and Cloud Operation

25. Extend local work with Remote Control

- 25.1 Operating rules
- 25.2 Mobile

26. Cloud sessions, --cloud and --teleport

- 26.1 From terminal to cloud
- 26.2 From cloud to terminal
- 26.3 GitHub access, and what it actually grants
- 26.4 Auto-fix, and a real hazard
- 26.5 Sharing a cloud session
- 26.6 Isolation and limits

27. Push events into a session with channels

- 27.1 Enabling one
- 27.2 The security model, and the part that matters
- 27.3 When a channel is the right shape

28. Publish work as artifacts

- 28.1 What an artifact is not
- 28.2 Sharing, and who sees what
- 28.3 Live data through connectors
- 28.4 Governance

Part VII — Operating at Scale

29. Parallel sessions without collisions

- 29.1 The mechanisms
- 29.2 Prevent file collisions
- 29.3 Cross-session messaging
- 29.4 Protect your attention

30. Agent teams

- 30.1 The behaviour change you must understand before enabling

- 30.2 Architecture
- 30.3 Control
- 30.4 Permissions
- 30.5 Quality gates
- 30.6 Limitations to plan around
- 30.7 When a team beats the alternatives

Part VIII — Governance, Assurance and Compliance

31. Enterprise configuration and managed policy

- 31.1 Delivery mechanisms
- 31.2 A minimum viable enterprise policy
- 31.3 Validation behaviour, and which keys fail closed
- 31.4 What a managed policy does not reach

32. Data governance, residency and retention

- 32.1 What leaves the machine
- 32.2 Retention
- 32.3 Zero Data Retention, and its exact boundary
- 32.4 Cloud execution
- 32.5 A data-governance questionnaire

33. Observability, cost control and accessibility

- 33.1 Open Telemetry
- 33.2 Cost control
- 33.3 Accessibility

Part IX — Capstone

34. Build a verified research-brief agent

- 34.1 Target outcome
- 34.2 Stage 1 — Establish control
- 34.3 Stage 2 — Create project knowledge
- 34.4 Stage 3 — Choose the register
- 34.5 Stage 4 — Make the reusable procedure
- 34.6 Stage 5 — Add an independent reviewer
- 34.7 Stage 6 — Add exactly one piece of bounded automation
- 34.8 Stage 7 — Render and verify
- 34.9 Stage 8 — Human release gate
- 34.10 A final goal

Schedule

Twelve modules delivered over three days.

#	Day	Module	Time	Participants leave able to
1	1	Foundations, setup and first session	90 min	Understand Claude Code, install and verify it, set up the workspace, and choose appropriate first tasks.
2	1	Start and steer a safe session	90 min	Write an inspectable brief, observe unfamiliar work, steer the session, and manage focus.
3	1	Plan, approve and execute	90 min	Create a complete plan, review scope, approve deliberately, and verify the completion report.
4	1	Permissions, sandboxing and Git	105 min	Use permission controls, sandboxing, checkpoints, Git and GitHub to protect work and recover safely.
5	1	Project knowledge and context	105 min	Build an effective CLAUDE.md, manage global instructions and memory, and deliberately engineer context.
6	2	Sessions and context management	90 min	Resume, name and retain sessions while knowing when to start fresh and how to reduce context overhead.
7	2	MCP, secrets and reusable skills	120 min	Connect external tools safely, protect secrets, and turn repeatable work into reusable skills.
8	2	Subagents and hooks	105 min	Delegate bounded work, create a focused reviewer, and build event-driven behaviour with safer hooks.
9	2	Plugins and output styles	90 min	Evaluate plugins before installation and configure output styles for different kinds of work.
10	3	Browser use, goals and workflows	120 min	Use browser automation, goals and dynamic workflows while controlling permissions, cost and safety.
11	3	Loops, cloud operation and agent teams	150 min	Run bounded loops, work across cloud sessions, coordinate parallel agents, and prevent collisions.
12	3	Governance and capstone	180 min	Apply enterprise controls, data governance and observability, then build and verify a complete research-brief agent

Curriculum details

Part I — Foundations

This section introduces the basic concepts of Claude Code and helps participants understand how to use it effectively in real development work.

1. What Claude Code Is

Participants will understand what Claude Code is, what it is not, and what they should clearly define before starting a task. They will learn how the control stack works, where their work runs, and which tasks are suitable for getting started with Claude Code.

2. Install, Verify & Authenticate

Participants will learn what is required to use Claude Code, how to install it, and how to verify that the installation works before logging in. Windows users will also understand the difference between using Claude Code natively and using WSL.

3. Surfaces, Projects & Boundaries

Participants will learn how to set up their IDE and create a simple workspace for practice. They will also understand how to work with large projects, including monorepos and large codebases.

4. Modes, Models, Effort & Usage

Participants will understand the main controls they need when using Claude Code. They will learn what effort levels mean, how to choose the right model and effort for different tasks, and how usage is different from context. They will also learn what to optimize first for better and more efficient results.

Outcome

By the end of this section, participants will be able to **set up Claude Code, understand its basic controls and environment, and confidently start their first coding tasks.**

Part II — A Safe Working Method

This section teaches participants how to work with Claude Code in a controlled and safe way, from starting a session to reviewing changes and recovering from mistakes.

5. Start and Steer a Session

Participants will learn how to start a Claude Code session with a clear and easy-to-check brief. They will learn how to observe Claude when working on an unfamiliar project, guide it without starting over, handle side questions, keep the session focused, and use recall when needed. They will also learn how voice input can be used during a session.

6. Plan, Comment, Approve, Execute

Participants will learn when they should ask Claude to create a plan before making changes. They will learn how to create a complete but simple plan and review it before allowing the work to continue. They will understand how to approve only the required scope and ask Claude for a completion report that can be checked against the actual changes.

7. Permissions, Sandboxing and Checkpoints

Participants will learn how permission modes control what Claude can do. They will understand how rules are evaluated and how to read a permission request before approving it. They will also learn how sandboxing provides an additional boundary and how checkpoints and rewind can help recover from unwanted changes.

8. Protect Work with Git and GitHub

Participants will learn a safe Git workflow for working with Claude Code. They will understand how to add a remote repository carefully, use Git with an AI agent, and use automated review while understanding its limitations. They will also learn that recovery involves more than just restoring the Git repository.

Outcome

By the end of this section, participants will be able to **start, guide, review, approve, and safely complete Claude Code tasks while protecting their work with permissions, checkpoints, and Git.**

Part III — Project Knowledge and Context

This section teaches participants how to give Claude Code the right project information, manage context, and keep sessions focused and useful.

9. Build an Effective CLAUDE.md

Participants will learn where Claude Code instructions live and how to create a useful project-level CLAUDE.md file. They will learn how to initialize the file, edit it based on the project's needs, and keep it focused on important information rather than storing everything in one place. They will also learn how to use modular rules and when the instructions need to be updated.

10. Global Instructions and Auto Memory

Participants will understand how global instructions work and how auto memory can help Claude remember useful information. They will learn when to use global instructions or memory and how to review stored information through a simple memory audit.

11. Engineer Context Deliberately

Participants will learn how to understand the available context and recognize when Claude is losing focus. They will explore different ways to recover focus, including rewinding, branching, forking, and using subtasks. They will also learn how to reduce unnecessary information that stays in the context.

12. Sessions: Resume, Name, Retain

Participants will learn how to resume previous sessions and name sessions based on their intended outcome. They will understand how session retention, storage, and privacy work and learn when it is better to start a fresh session.

Outcome

By the end of this section, participants will be able to **organize project instructions, manage Claude's context and memory, and maintain focused sessions for longer development tasks.**

Part IV — Extending Claude Code

This section teaches participants how to extend Claude Code with external tools, reusable skills, subagents, hooks, plugins, and custom output styles.

13. Connect External Tools with MCP

Participants will learn how MCP connects Claude Code with external tools and services. They will understand different scopes and transports, follow a safe installation process, and learn how MCP tools can affect context usage. They will also understand how organizations can control MCP usage.

14. Environment Variables and Secrets

Participants will learn how Claude Code interacts with environment variables and why simply hiding a secret from Claude is not enough for security. They will learn how to use multiple layers of protection and what to do if a secret is accidentally exposed.

15. Turn Repeatable Work into Skills

Participants will learn how to turn repeated tasks into reusable Claude Code skills. They will build a skill from an existing workflow, understand the important frontmatter, and review the skill for potential security issues.

16. Delegate Bounded Work to Subagents

Participants will learn how subagents can handle specific tasks using a fresh context. They will understand where subagents are defined, how configuration is selected, and how to create a simple read-only reviewer. They will also learn the important limits, defaults, and frontmatter settings, and understand when to use a skill versus a subagent.

17. Event-Driven Behaviour with Hooks

Participants will learn how hooks can automatically run actions when specific events happen. They will understand different handler types, create a simple blocking example, and follow a safer workflow for creating hooks. They will also learn how to think about the security risks of automated actions.

18. Evaluate and Install Plugins

Participants will learn what a Claude Code plugin can contain and how to evaluate it before installation. They will use an installation gate to check plugins carefully and understand when building a focused solution is better than installing a large bundle.

19. Output Styles: Claude Code Beyond Software Engineering

Participants will learn how output styles can change the way Claude Code responds for different types of work. They will explore the built-in styles, learn how to set or create a style, and understand where a style applies and where it does not.

Outcome

By the end of this section, participants will be able to **extend Claude Code with MCP, skills, subagents, hooks, plugins, and output styles while understanding their usage, boundaries, and security considerations.**

Part V — Browser Use, Autonomy and Orchestration

This section teaches participants how to use Claude with browsers, goals, workflows, loops, and scheduled tasks while keeping control over automated actions.

20. Use Claude in Chrome Safely

Participants will learn how to connect Claude to Chrome and set up the required prerequisites. They will learn why using a dedicated browser profile is safer and how to increase capabilities step by step. They will understand what browser actions Claude can perform, the risks of prompt injection and authenticated actions, and the basics of computer use.

21. Keep Work Moving with Goals

Participants will learn how goals can help Claude continue working toward a defined outcome. They will understand what the evaluator can and cannot see, how to create a safe goal, and what behaviour to expect. They will also learn why providing the right context is important for achieving reliable results.

22. Dynamic Workflows and Ultracode

Participants will learn how to start dynamic workflows and understand their entry points, limits, and permissions. They will learn how resume behaviour affects cost and when a workflow is useful enough to justify that cost. They will also build a simple research workflow and learn how to save and share it.

23. Session-Scoped Scheduling with /loop

Participants will learn the different ways /loop can be used and how each behaves. They will understand its operational limits, lifecycle, and what actually stops a loop. They will also compare it with the Monitor alternative and learn which tasks are suitable or unsuitable for loops.

24. Cloud Routines and Desktop Scheduled Tasks

Participants will learn what a routine is and why it should be designed and reviewed carefully before activation. They will understand why the text provided when a routine runs should be treated as untrusted input. They will also learn what to check before activating a routine and how desktop scheduled tasks can support headless automation.

Outcome

By the end of this section, participants will be able to **use browser automation, goals, workflows, loops, and scheduled tasks while understanding their permissions, limitations, costs, and safety boundaries.**

Part VI — Distributed and Cloud Operation

This section teaches participants how to extend Claude Code beyond the local terminal and work with remote sessions, cloud environments, channels, and shared artifacts.

25. Extend Local Work with Remote Control

Participants will learn how Remote Control can extend their local Claude Code work to other devices. They will understand the basic operating rules and how to use Remote Control from a mobile device.

26. Cloud Sessions, --cloud and --teleport

Participants will learn how to move work from the terminal to the cloud and bring cloud work back to the terminal. They will understand what GitHub access actually allows, the risks of automatic fixes, and how to safely share a cloud session. They will also learn about isolation and the limits of cloud sessions.

27. Push Events into a Session with Channels

Participants will learn how to enable channels and send events into an active Claude Code session. They will understand the channel security model and identify situations where using a channel makes sense.

28. Publish Work as Artifacts

Participants will learn what an artifact is and what it is not. They will understand how artifacts can be shared, who can access them, and how connectors can provide live data. They will also learn the basic governance considerations when publishing work.

Outcome

By the end of this section, participants will be able to **work across local and cloud environments, control remote sessions, receive external events, and safely share their work as artifacts.**

Part VII — Operating at Scale

This section teaches participants how to manage multiple Claude Code sessions and coordinate agents effectively when working on larger projects.

29. Parallel Sessions Without Collisions

Participants will learn how multiple Claude Code sessions can work at the same time. They will understand the mechanisms used to run parallel work, how to prevent different sessions from changing the same files, and how to communicate across sessions. They will also learn how to manage their attention when several sessions are running.

30. Agent Teams

Participants will learn how agent teams change the way Claude Code works on larger tasks. They will understand the basic architecture, how to control team members, and how permissions are managed. They will learn how to use quality gates, plan around the limitations, and understand when an agent team is appropriate compared with other approaches.

Outcome

By the end of this section, participants will be able to **run parallel Claude Code sessions, coordinate multiple agents, prevent conflicts, and manage larger development tasks in a controlled way.**

Part VIII — Governance, Assurance and Compliance

This section teaches participants how to manage Claude Code in larger organizations with clear policies, data controls, monitoring, and accessibility.

31. Enterprise Configuration and Managed Policy

Participants will learn how enterprise policies are delivered and applied across Claude Code environments. They will understand what a basic enterprise policy should include and how policy validation works. They will also learn which areas managed policies can control and where their reach ends.

32. Data Governance, Residency and Retention

Participants will learn what data can leave the local machine and how data retention works. They will understand the exact boundaries of Zero Data Retention and how cloud execution affects data handling. They will also learn how to use a data-governance questionnaire when evaluating a Claude Code setup.

33. Observability, Cost Control and Accessibility

Participants will learn how OpenTelemetry can provide visibility into Claude Code usage and activity. They will understand basic approaches to controlling costs and learn about accessibility considerations when using Claude Code.

Outcome

By the end of this section, participants will be able to **understand enterprise policies, evaluate data-handling requirements, monitor usage, control costs, and consider accessibility when deploying Claude Code at scale.**