Cloud Contraptions
TRAINING & CONSULTING

To discuss this course and customizations:
Call: 434-509-5680 or Email: sales@cloudcontraptions.com

Linux Command Line and Bash for Developers

Class Duration

14 hours of live training delivered over 2 days.

Student Prerequisites

- Software development experience (any language)
- Ability to open a terminal and navigate directories
- Basic familiarity with editing text files
- No prior Bash scripting or Linux administration experience required

Target Audience

This course is designed for professional developers who work on or deploy to Linux but have never built real command-line fluency — developers who copy shell commands from documentation or AI assistants without fully understanding them. It suits application developers moving closer to operations, data engineers who live in pipelines, and anyone whose CI, containers, or cloud servers run Linux, which today is nearly everyone.

Description

The command line is the one developer tool that never goes out of date, yet most developers use perhaps a dozen commands and treat the rest as folklore. This course builds genuine fluency from the ground up: how the shell actually interprets what you type, how files, permissions, and processes work, and how to compose small tools into powerful pipelines with pipes and redirection. Participants get extensive hands-on practice with the text-processing workhorses — grep, sed, awk, and jq — plus find and xargs for operating on files in bulk.

The second day turns fluency into automation. Participants write real Bash scripts with variables, conditionals, loops, and functions, and learn the defensive practices that separate reliable scripts from time bombs: quoting discipline, error handling with `set -euo pipefail`, and static analysis with ShellCheck. The course covers the surrounding toolkit — SSH and scp for remote work, tmux for persistent sessions, and package managers — and finishes by building practical developer automation scripts. Just as important,

To discuss this course and customizations:
Call: 434-509-5680 or Email: sales@cloudcontraptions.com

it teaches judgment: recognizing the point where a shell script has outgrown Bash and should be rewritten in Python.

Learning Outcomes

- Explain how the shell parses commands, expansions, and quoting, and predict what a command will do before running it
- Navigate and manage files confidently, including permissions, ownership, links, and the standard filesystem layout
- Inspect and control processes with ps, kill, signals, job control, and exit codes
- Compose pipelines using pipes, redirection of stdout and stderr, command substitution, and here-documents
- Search and transform text with grep, sed, and awk, and process JSON with jq
- Operate on file sets in bulk with find and xargs, safely handling unusual filenames
- Write Bash scripts using variables, conditionals, loops, functions, and argument parsing
- Build robust scripts with `set -euo pipefail`, traps, and deliberate error handling
- Lint scripts with ShellCheck and fix the issues it reports
- Work on remote machines effectively with ssh, scp, key-based authentication, and tmux sessions
- Install and manage software with system package managers (apt, dnf) and Homebrew
- Decide when a task belongs in Bash and when to reach for Python instead

Training Materials

Comprehensive courseware is distributed online at the start of class. All students receive a downloadable MP4 recording of the training.

Software Requirements

- A Linux environment: native Linux, macOS terminal, WSL2 on Windows, or a provided cloud VM
- Bash 5.x available as the shell
- Text editor or IDE (VS Code recommended)
- ShellCheck installed locally or as an editor extension

- SSH client and tmux installed
- jq installed

Training Topics

Shell Fundamentals

- What a shell is: Bash, zsh, and POSIX sh
- How commands are parsed: words, expansions, quoting
- Environment variables, PATH, and shell startup files
- Tab completion, history, and line-editing shortcuts
- Getting help: man pages, `--help`, and tldr

Files, Permissions, and Processes

- The filesystem hierarchy and where things live
- Permissions, ownership, chmod, chown, and umask
- Hard links, symbolic links, and hidden files
- Processes, ps, top, signals, and kill
- Job control: foreground, background, and nohup
- Exit codes and why they matter for automation

Pipes and Redirection

- stdin, stdout, stderr, and file descriptors
- Pipes and building multi-stage pipelines
- Redirection: `>`, `>>`, `2>&1`, and `/dev/null`
- Command substitution and process substitution
- Here-documents and here-strings
- tee, sort, uniq, cut, tr, wc, head, and tail

Text Tooling

- grep and regular expressions that actually work
- sed for stream editing and in-place substitution
- awk for column-oriented processing and reports
- jq for querying and reshaping JSON
- Combining tools: real log-analysis pipelines

find and xargs

- find by name, type, size, time, and permissions
- Executing actions with `-exec` and `-delete`
- xargs for bulk operations and parallelism

- Handling spaces and odd filenames with `-print0`

Bash Scripting Fundamentals

- Shebangs, executable bits, and script anatomy
- Variables, quoting discipline, and arrays
- Conditionals: `if`, `[[ ]]`, and case statements
- Loops: for, while, and reading input line by line
- Functions, local variables, and return values
- Parsing arguments and providing usage help

Robust Scripts and Error Handling

- `set -euo pipefail`: what it does and its sharp edges
- Traps for cleanup and temporary files
- Checking prerequisites and failing fast with messages
- Logging, verbosity flags, and dry-run modes
- Linting with ShellCheck and fixing what it finds

Remote Work: SSH, scp, and tmux

- SSH key generation and key-based authentication
- SSH config files, agents, and jump hosts
- Copying files with scp and rsync basics
- tmux: sessions, windows, panes, and detaching
- Surviving disconnects on long-running jobs

Package Managers

- apt and dnf on Linux servers
- Homebrew on macOS and Linux
- Installing developer tools reproducibly
- Understanding what package managers change on your system

Developer Automation Scripts

- Project bootstrap and environment-check scripts
- Wrapping repetitive git, container, and deploy commands
- Cron basics for scheduled tasks
- Making scripts portable across macOS and Linux
- When to reach for Python instead: complexity signals, data structures, testing, and dependencies