DEVELOPER CLI, GIT & ARCHITECTURE CHEAT SHEET

Open Source Tooling, Terminal Shortcuts & Systems Reference (2026)

1. Modern Git & Version Control Workflow

VCS Engine

- `git switch -c <branch>` — Produce new branch and switch immediately.
- `git commit --amend --no-edit` — Add missed staged changes into last commit.
- `git rebase -i HEAD~3` — Interactive rebase to squash or reword last 3 commits.
- `git cherry-pick <hash>` — Apply specific commit from another branch.
- `git stash push -m "note"` — Stash dirty state; apply with `git stash pop`.
- `git restore --staged <file>` — Unstage file while retaining working edits.
- `git reset --soft HEAD~1` — Undo commit while keeping files staged.
- `git log --oneline --graph --all` — Visualize full commit tree via clean ASCII.

2. GitHub CLI (gh) & Lazygit Essentials

TUI & CLI

Tool	Command / Key	Function & Instant Productivity Impact
gh	`gh pr create --web`	Launch browser to publish new Pull Request with template.
gh	`gh pr checkout <no>`	Pull PR branch locally and switch instantly.
gh	`gh run watch`	Live-stream and watch GitHub Actions CI/CD workflows.
lazygit	`Space` / `c` / `P`	Stage file / Open commit message / Push upstream.
lazygit	`z` / `e` / `[ ]`	Undo last action / Open file in editor / Switch active panel.

3. High-Velocity Package & Runtime Managers

uv • pnpm • fnm

```
# uv (Python - Rust-powered, 10-100x faster than pip)
uv venv && source .venv/bin/activate # Sub-second virtual environment
uv pip install -r req.txt # Global content-addressed caching
uv run script.py # Executes in ephemeral isolated environment
```

```
# pnpm (Node.js - Hardlinks & symlinks saving 80% disk)
pnpm install && pnpm add -D typescript # Content-addressable store
pnpm dlx degit <repo> <target> # Instant execution without install
```

```
# fnm (Fast Node Manager - Instant Rust shell switching)
fnm install --lts && fnm use 22 # Zero startup latency across shells
```

4. Terminal Navigation & Readline Shortcuts

Keybindings

Key	Action	Key	Action
`Ctrl + A`	Move cursor to line start	`Ctrl + E`	Move cursor to line end
`Ctrl + U`	Cut from cursor to start	`Ctrl + K`	Cut from cursor to line end
`Ctrl + W`	Cut previous word	`Ctrl + Y`	Paste (Yank) last cut text
`Ctrl + R`	Reverse search history	`Ctrl + L`	Clear screen display
`Alt + .`	Insert last command argument	`Ctrl + C`	Interrupt / Kill active process

5. HTTP Status Codes & REST Architecture

API Standards

Code	RFC Status	Practical Usage & Architectural Rule
200 OK	Success	GET, PUT or PATCH succeeded with response body.
201 Created	Resource Produced	POST produced a new resource (Location header returned).
204 No Content	Empty Success	Action (DELETE/PUT) succeeded; no content in body.
304 Not Mod.	Cached Hit	ETag or If-Modified-Since valid; zero payload transferred.
400 Bad Req.	Malformed	Client syntax invalid or payload schema rejected.
401 Unauth.	Unauthenticated	Missing, expired, or invalid authorization credentials.
403 Forbidden	No Permission	Identity verified, but lacks RBAC access permissions.
404 Not Found	Missing URI	Target resource does not exist at requested endpoint.
409 Conflict	State Clash	Duplicate unique key constraint or edit version collision.
422 Unproc.	Domain Invalid	Syntax valid JSON, but business validation failed.
429 Rate Limit	Throttled	Usage quota exceeded; inspect Retry-After response header.
500 / 502 / 503	Server Errors	Internal crash / Upstream bad gateway / Overloaded service.

6. Essential Regular Expressions (Regex)

Validation

Email: `^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$`

URL (HTTP/S): `^https?:\/\/[\w.-]+(?:\.[\w\.-]+)+[\w\-\._~:/?#[\]@!\$&'\(\)\*\+,;=.]+$`

IPv4 Address: `^(?:(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.){3}(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)$`

UUID v4: `^[0-9a-f]{8}-[0-9a-f]{4}-4[0-9a-f]{3}-[89ab][0-9a-f]{3}-[0-9a-f]{12}$`

ISO 8601 Date: `^\d{4}-(?:0[1-9]|1[0-2])-(?:0[1-9]|[12]\d|3[01])$`

Semantic Version: `^v?(0|[1-9]\d*)\.(0|[1-9]\d*)\.(0|[1-9]\d*)(?:-[0-9A-Za-z.-]+)?$`

7. SQL Indexing & Relational Architecture

RDBMS Rules

Leftmost Prefix Rule: `INDEX(a, b, c)` accelerates only queries on `(a)`, `(a, b)` or `(a, b, c)`. A standalone `WHERE b = 5` cannot use this index and causes a full table scan.

JOIN Mechanics: `INNER` gives intersection. `LEFT JOIN` preserves all left rows. Avoid `SELECT *` in production to minimize serialization and I/O buffer bloat.

ACID Guarantees: Atomicity (all or nothing), Consistency (schema invariants), Isolation (concurrent safety levels), Durability (persisted on disk via WAL).

Query Diagnostics: Always profile slow endpoints using `EXPLAIN (ANALYZE, BUFFERS)` to verify index scans over expensive sequence scans.

8. High-Performance CLI Utility Stack

Rust & Go

Legacy	Modern Tool	Performance & Capability Upgrade
`cd`	**zoxide (z)**	Learns directory habits; jump instantly with `z proj`.
`cat`	**bat**	Syntax highlighting, line numbers and Git modification gutters.
`grep`	**ripgrep (rg)**	Rust-based, respects .gitignore, blazing fast recursive search.
`find`	**fd**	Intuitive syntax, multithreaded directory traversal.
`ls`	**eza**	Unicode glyphs, file permissions and color-coded Git tree state.