How deep should my programming or scripting skills be for DevOps readiness

Short answer: aim for strong “automation engineer” skills in Bash and Python—enough to design small tools, read others’ code, and safely change production pipelines, but not full software-engineer depth. Think: build scripts of 50–300 lines, write tests, handle errors, and use APIs/SDKs confidently.[1] [2]

Bash depth

- Comfortable with variables, arrays, functions, exit codes, traps, and set -euo pipefail to write safe scripts that fail fast. This enables reliable CI steps and server automations.[2] [1]
- Proficient with grep/awk/sed, process control, and pipelines to parse logs, transform config, and glue tools together during deployments.[3] [1]
- Able to package scripts with usage/help, config via env/flags, and clear logging so teammates can run them in pipelines.[4] [3]

Python depth

- Build CLI tools with argparse/typer, call REST APIs with requests, parse JSON/YAML, and work with files/streams for day-to-day automation. This covers most DevOps utilities.[1] [2]
- Use cloud SDKs (e.g., boto3) for CRUD on infrastructure, tagging, and housekeeping jobs with proper pagination/retries and idempotency.[5] [2]
- Write unit tests with pytest and basic mocks; structure code into modules; log with levels; handle exceptions cleanly so pipelines fail with useful messages.[3] [1]

Reading and modifying code

- Read unfamiliar Bash/Python and explain what it does; make safe edits for bug fixes or new flags without breaking behavior. This is often tested in take-home tasks.[2] [1]
- Follow conventions and style tools (shellcheck/flake8/black) to keep changes review-friendly and pipeline-ready.[3] [2]

API and data handling

- Understand HTTP methods/status codes, auth headers/tokens, and pagination patterns to automate external/internal services.[1] [3]
- Confident with serialization (JSON/YAML), templates (Jinja2), and small data transforms used in CI/CD and IaC workflows.[2] [1]

Pipeline integration

- Wrap tools with clear exit statuses and logs, pass artifacts between steps, and parameterize behavior for dev/stage/prod. This is the glue skill for CI/CD.[3] [2]
- Add pre-flight checks, dry-run modes, and rollback hooks so scripts are safe to run against production.[4] [1]

Practical targets to gauge readiness

- Write a log parser and an API health checker in Bash and Python, each with help flags and tests, in under a day. This simulates common interview tasks.[1] [2]
- Automate a small AWS workflow (e.g., rotate a secret, tag stale resources, or roll an Auto Scaling Group) using boto3 with retries and idempotency.[5] [2]
- Create a GitHub Actions or Jenkins step that builds, tests, and deploys a container, failing fast on test or health-check errors with clear diagnostics.[2] [3]

What is not required

- Deep algorithms, advanced data structures, or framework-level backend development are optional; focus on robust automation, clear error handling, and integration with cloud and CI tools.[1] [2]

How to level up efficiently

- Daily 45–60 minutes: alternate Bash and Python challenges tied to real tasks (log parsing, API calls, file templating), always with tests and a short README. This builds reliable muscle memory for interviews.[3] [2]
- Weekly mini-project: one end-to-end script that runs locally and in CI (e.g., canary flip via Route 53, or pipeline pre-flight validator) to practice safe changes at production boundaries.[2] [1]

⁂

1. learning.technical_concepts
2. learning.technical_skills
3. productivity.documentation
4. productivity.communication
5. tools.aws_scaling
6. DevOps.jpg