perplexity

Which specific Python libraries should I know for common DevOps tasks

Short answer: focus on a core toolbox for automation, cloud control, testing, packaging, and CI/CD integration—prioritize requests, boto3, PyYAML/ruamel.yaml, json, logging, argparse/Typer, subprocess, pytest, and jinja2; then add specific utilities like paramiko, fabric, click, docker, kubernetes, and cryptography as needed.[1] [2]

HTTP and data

- requests: session-based REST calls, retries, auth headers, and pagination for service automation and CI checks.[2] [1]
- json and PyYAML/ruamel.yaml: parse/emit config; ruamel preserves comments when editing IaC files programmatically.[1] [2]
- jinja2: render config/templates for apps, Nginx/Apache, or pipeline files from variables.[3] [1]

Cloud and infrastructure

- boto3: manage EC2/ASG, Route 53 weighted records, S3 uploads/lifecycles, CloudWatch metrics/alarms, IAM, and Secrets Manager rotations with pagination/backoff.[4] [2]
- botocore.exceptions: precise error handling for AWS calls to make scripts idempotent and safe.[2] [1]

CLI and OS integration

- argparse or Typer: structured CLIs with subcommands, help, and exit codes; Typer speeds up modern CLIs with type hints.[1] [2]
- logging and rich: standard logs plus colored/structured output for readable pipelines and local runs.[5] [1]
- subprocess and shlex: call system CLIs (docker, aws, git) reliably with captured output.[6] [1]
- pathlib and tempfile: robust file and artifact handling across platforms.[2] [1]

Remote access and config management

- paramiko and fabric: SSH to servers, run commands, fetch logs, and perform small fixes when an agent isn't available.[1] [2]
- python-dotenv: manage local env files for dev/test parity without leaking secrets.[2] [1]

Testing and quality

- pytest, pytest-mock, and requests-mock: unit tests, fixtures, and HTTP stubs so tools are CI-gated.[1] [2]
- ruff/flake8 and black: lint and format consistently; integrate with pre-commit.[5] [2]
- jsonschema or pydantic: validate API responses or config before deploy.[2] [1]

Containers and orchestration

- docker (SDK for Python): build/run containers, manage images, and query logs programmatically in CI.[6] [1]
- kubernetes (kubernetes-client): interact with clusters, apply manifests, watch rollouts, and gather pod logs for debugging.[1] [2]

Security and secrets

- cryptography: hashing, signing, and simple encryption tasks for secure pipelines.[2] [1]
- boto3 Secrets Manager/SSM Parameter Store helpers: rotate and fetch secrets safely in code.[4] [2]

Messaging and jobs (optional but useful)

- boto3 SQS helpers: enqueue jobs, poll queues, and handle dead-letter flows for decoupled pipelines.[4] [2]
- schedule or APScheduler: lightweight cron-like task runners inside automation tools.[1] [2]

Practical starter set

- Minimal requirements file: requests, boto3, PyYAML, jinja2, Typer or argparse, pytest, black, ruff, rich. This covers 80% of daily DevOps automation needs.[2] [1]

Skills to apply with these libraries

- Build a CLI that validates API health, renders configs from jinja2, and flips Route 53 canary weights with safe rollback using boto3 and retries.[4] [2]
- Create a housekeeping tool that tags stale EC2/EBS and rotates Secrets Manager entries, with logs, tests, and a dry-run flag.[4] [2]

⁂

1. learning.technical_concepts
2. learning.technical_skills
3. tools.web_servers
4. tools.aws_scaling
5. productivity.documentation
6. tools.docker_workflow

7. DevOps.jpg