CodeQL Capabilities — Exam-Ready Cheat Sheet

---

1 What is CodeQL?

CodeQL is GitHub's semantic static analysis engine for code scanning. It builds a *relational database* of your code and runs declarative **QL** queries over it to detect complex vulnerabilities that simple pattern matching misses.

2 Why Use It

- Finds data-flow issues (e.g., injection), unsafe crypto, and other vuln classes across many languages.
- Integrates with GitHub Actions, runs locally via the **CodeQL CLI**, and supports authoring in VS Code.
- Produces precise results with traces and remediation context.

3 Key Terms

QL Pack A collection of related CodeQL queries for a *specific language* (e.g., Python).

Code Query (`.ql`) A single query to find a specific issue (e.g., SQLi) via CLI/Actions/VS Code.

Query / Code Suite A curated *set* of queries grouped by security theme (e.g., web-app security).

4 Default Query Suites

GitHub provides prebuilt suites in the public `github/codeql` repository. Choose suites by *domain* (e.g., security-extended) and pair them with your repo's languages in Actions or the CLI.

5 End-to-End Flow

1. **Create database** per language (semantic model of your code).
2. **Run queries/suites** over that database.
3. **Review results** with locations, data-flow traces, and guidance (great UX in VS Code).

6 Minimal GitHub Actions Setup (Multi-language)

Drop into `.github/workflows/codeql.yml`:

```yaml
name: CodeQL
on:
  push: { branches: [main] }
  pull_request: { branches: [main] }
  schedule:
    - cron: "0 2 * * 1"  # weekly

jobs:
  analyze:
    runs-on: ubuntu-latest
    permissions:
      security-events: write
      contents: read
      actions: read
    strategy:
      fail-fast: false
      matrix:
        language: [javascript-typescript, python]  # add more as needed
    steps:
      - uses: actions/checkout@v4

      - uses: github/codeql-action/init@v3
        with:
          languages: ${{ matrix.language }}
          queries: +security-extended  # or a custom suite

      - uses: github/codeql-action/autobuild@v3

      - uses: github/codeql-action/analyze@v3
        with:
          category: "/language:${{ matrix.language }}"
```

7 Local Workflow (CLI + VS Code)

```bash
# 1) Create the database (per language)
codeql database create ./db-py \
  --language=python \
  --source-root /path/to/repo

# 2) Analyze with a suite or pack
codeql database analyze ./db-py \
  github/codeql/python/ql/src/codeql-suites/python-security-extended.qls \
  --format=sarifv2.1.0 --output=results.sarif

# Tip: Use the VS Code "CodeQL" extension to open DBs,
# author queries, and inspect results interactively.
```

8 Authoring Queries (Mental Model)

- QL is *declarative*: define types/predicates, then finish with `from ... where ... select ...`.
- Data-flow pattern: define **sources** (e.g., user input) and **sinks** (e.g., SQL exec), then search for flows without sanitization.
- Start from existing queries in `github/codeql`; customize progressively.

9 Practical Tips

- Begin with GitHub-provided suites; extend with org-specific query packs.
- Cache databases in CI and shard by language for large monorepos.
- Keep CLI invocations scriptable; capture SARIF for artifacting and dashboards.
- Triage in the Code Scanning UI; iterate queries as *guardrails*, not gates, until signal is strong.

Build Note: Compile with `latexmk -pdf -shell-escape <file>` (or enable *–shell-escape* in your LaTeX tool) so `minted` can call Pygments.