GitHub Code Scanning Quick Reference

CodeQL and Workflow Triggers Cheat Sheet

Learning Objectives

- Understand how code scanning fits into the SDLC and shift-left security.
- Choose an appropriate scan *frequency* (continuous, scheduled, on-demand).
- Select *triggering events* that match your development model.
- Edit the default CodeQL Actions workflow for production-grade repos.

1. Code Scanning in the SDLC (Shift Left)

Early, automated detection reduces remediation cost and prevents reputational impact. Integrate scanning from planning/requirements through pull requests and CI to encourage a proactive security culture and higher code quality.

2. Frequency: Trade-offs

Mode	When to use	Pros	Cons
Continuous (on push/PR)	Fast-paced repos; short feedback loops	Earliest detection; tight PR gating	Higher runner cost; alert fatigue if uncurated
Scheduled (cron)	Slower-moving repos; baseline monitoring	Predictable cost; steady coverage	Delayed detection between runs
On-demand (workflow_dispatch)	Pre-release checks; governance workflows	Flexible; ad hoc before deploy	Manual step; can be forgotten

3. Recommended Triggers

Combine triggers to balance cost and coverage:

- **push**: Immediate feedback on protected branches.
- **pull_request**: Enforce PR checks before merge.
- **schedule**: Nightly or hourly baselines for drift.
- **workflow_dispatch**: Manual scans for releases/hotfixes.

4. Production-Ready CodeQL Workflow Skeleton

```yaml
name: CodeQL scan

on:
  push:
    branches: [ "main", "release/**" ]
  pull_request:
    branches: [ "main" ]
  schedule:
    - cron: "0 3 * * *"   # nightly at 03:00 UTC
  workflow_dispatch: {}

permissions:
  contents: read
  security-events: write
  actions: read

jobs:
  analyze:
    name: Analyze
    runs-on: ubuntu-latest
    timeout-minutes: 90

    strategy:
      fail-fast: false
      matrix:
        language: [ "javascript-typescript", "python" ]

    steps:
      - uses: actions/checkout@v4

      # Optional: prune paths to save time/cost
      - name: Filter paths (optional)
        run: |
          echo "Excluding large generated/vendor dirs via CodeQL config"

      - name: Initialize CodeQL
        uses: github/codeql-action/init@v3
        with:
          languages: ${{ matrix.language }}
          config-file: .github/codeql/codeql-config.yml

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

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

Optional `codeql-config.yml` to exclude noisy paths or tune queries:

```yaml
name: org-defaults
paths-ignore:
  - "dist/**"
  - "build/**"
  - "**/vendor/**"
  - "**/*.min.js"
queries:
  - uses: security-extended
  - uses: security-and-quality
```

5. Cost and Signal Tuning

- Use `paths`/`paths-ignore` on triggers to scope scans.
- Maintain a `codeql-config.yml` to exclude generated/vendor code.

- Prefer `matrix` languages to one job per language (parallelism).
- Curate queries: start with `security-extended`; add selectively.
- Triage rigorously: severity filters, labels, and periodic "fix-a-thon" sprints.

6. Notifications and Integrations

- Route alert summaries to Teams/Slack via workflow steps or the API.
- Create tracking issues automatically from alerts for ownership.
- Cache dependencies to speed runs; consider self-hosted runners for heavy scans.

7. Dependabot Baseline (Example)

```yaml
# .github/dependabot.yml
version: 2
updates:
  - package-ecosystem: "npm"
    directory: "/"
    schedule:
      interval: "daily"
    labels: [ "dependencies" ]
```

8. Practical Cases

Contoso: integrates scanning early to cut remediation cost and improve maintainability.
Wingtip Toys: push-triggered scans for instant feedback plus scheduled scans pre-release.

9. Setup Checklist

- Enable GHAS features at org/repo level as needed.
- Add CodeQL workflow with push, PR, schedule, and manual triggers.
- Create `codeql-config.yml` to exclude noise and select queries.
- Establish triage SOP (labels, SLAs, ownership, dismissal reasons).
- Wire notifications and create tracking issues from alerts.
- Review runner usage and adjust schedules/paths to control cost.