Spickzettel: Wie funktionieren GitHub Actions technisch?

1. Technische Architektur

- **Event-basiertes System**: Workflows werden durch Events (`push`, `pull_request`, `schedule`) ausgelöst.
- **Runner**: Workflows laufen auf GitHub-gehosteten oder selbstgehosteten Maschinen.
- **Workflows & Jobs**:
  - **Workflows** sind YAML-Dateien in `.github/workflows/`.
  - **Jobs** laufen unabhängig oder sequentiell (`needs:`-Abhängigkeiten).
  - **Steps** innerhalb eines Jobs führen Actions oder Shell-Befehle aus.

2. Ablauf eines Workflows

1. **Event tritt ein** (z. B. Push auf `main` → löst CI/CD-Workflow aus).
2. **Workflow-Datei wird geladen & analysiert**.
3. **Jobs werden parallel oder sequenziell auf Runnern gestartet**.
4. **Runner lädt Repository und führt Jobs aus**:
   - `actions/checkout@v3` holt den Code.
   - Steps führen definierte Kommandos oder Actions aus.
5. **Ergebnisse (Artefakte, Logs) werden gespeichert**.

3. Runner-Typen

- **Gehostete Runner** (GitHub bietet kostenlose VMs für `ubuntu-latest`, `windows-latest`, `macos-latest`).
- **Self-hosted Runner** (eigene Maschinen für mehr Kontrolle und Performance).
- **Docker-Runner** für containerisierte Builds.

4. Sicherheitsmechanismen

- **Token & Secrets** (`secrets.GITHUB_TOKEN`) für sicheren Zugriff auf GitHub-APIs.
- **Sandboxing & Isolierung** für jeden Job (neue VM für jeden Run).
- **Branch Protection Rules** verhindern ungewollte Änderungen.

5. Logging & Debugging

- **GitHub Actions UI → Actions Tab** zeigt Logs und Fehler.
- **Manuelles Debugging aktivieren**:
- `steps:`
- `  - name: Debug aktivieren`
- `    run: env`
- **Logs durchsuchen & Debug-Modus nutzen**:
- `GITHUB_ACTIONS_RUNNER_DEBUG=true`

Best Practices

- **Workflows modular halten** (z. B. separate Jobs für Build & Tests).
- **Caching (`actions/cache`) nutzen**, um Build-Zeiten zu reduzieren.
- **Self-hosted Runner für Performance-intensive Workflows verwenden**.
- **Verwendung von Secrets & Tokens minimieren, wo nicht nötig**.