Das classroom50-Repository

Jede Klasse hat ein privates Repository <ORG>/classroom50. Es ist die **einzige** Konfigurationsquelle des Klassenzimmers: Aufgaben, Klassenliste, Autograder, Resultate und die Workflows liegen dort. Angelegt wird es von gh teacher classroom add.

Aufbau

```
classroom50/
├── .github/workflows/
│   ├── autograde-runner.yaml     ← zentraler Reusable Workflow (von gh
teacher)
│   ├── publish-pages.yaml        ← stellt Autograder und Bundles auf GitHub
Pages
│   ├── collect-scores.yaml       ← sammelt nachts alle Releases ein
│   └── moodle-sync.yaml          ← BZZ-eigen: Übertrag nach Moodle
└── <CLASSROOM>/                  ← ein Ordner pro Klassenzimmer
    ├── classroom.json            ← Kurzname und Stammdaten
    ├── assignments.json          ← alle Aufgaben, Templates, Termine
    ├── roster.csv                ← Klassenliste (Anzeigedaten)
    ├── autograder.py             ← Klassen-Default: startet pygrader50
    ├── autograders/<SLUG>/       ← optional: Bewertungs-Bundle je Aufgabe
    ├── scores.json               ← eingesammelte Resultate
    └── moodle-state.json         ← was schon nach Moodle übertragen wurde
```

Ein Repository kann **mehrere** Klassenzimmer enthalten — je einen Ordner. Der Ordnername ist der Kurzname aus classroom.json und muss nicht dem Organisationsnamen entsprechen.

Die Pages-Site

publish-pages veröffentlicht Teile des Repos unter https://<ORG>.github.io/classroom50/. Von dort holt der Runner bei jeder Abgabe:

Datei	URL
Klassen-Default	https://<ORG>.github.io/classroom50/<CLASSROOM>/autograder.py
Bewertungs-Bundle	https://<ORG>.github.io/classroom50/<CLASSROOM>/autograders/<SLUG>.tar.gz
Aufgabenliste	https://<ORG>.github.io/classroom50/<CLASSROOM>/assignments.json

Das Klassen-Segment im Pfad ist Pflicht. …/classroom50/autograder.py **ohne** <CLASSROOM>/ liefert 404. Nach einem Push dauert der Deploy rund 30 Sekunden.

Wie eine Abgabe bewertet wird

Im Studi-Repo liegt `.github/workflows/autograde-runner.yaml`. Die Datei wird beim Anlegen des Repos eingesetzt und ruft nur den **zentralen** Reusable Workflow des Config-Repos auf. Der Lauf wird übersprungen, wenn die Commit-Message `CLASSROOM 50` oder `NOACTION` enthält.

1. Der `setup`-Job liest `.classroom50.yaml` aus dem Studi-Repo und den passenden Eintrag aus `assignments.json` der Pages-Site.
2. Der `grade`-Job holt `runner.py` von der Pages-Site und startet es.
3. `runner.py` lädt `autograders/<SLUG>.tar.gz` und sucht dann in dieser Reihenfolge: aufgaben-eigenes `autograder.py` → aufgaben-eigene `tests.json` → `<CLASSROOM>/autograder.py` → 0/0.
4. An der BZZ greift der dritte Punkt: `<CLASSROOM>/autograder.py` installiert pygrader50 in gepinnter Version und startet es im Studi-Checkout.
5. `runner.py` liest danach `result.json`, validiert es und publiziert Release und Commit-Status.

Der alte Weg über `BZZ-Commons/pygrader` und `py_autograding.yml` ist abgelöst. Wer ihn noch in einem Repo findet, siehe Migration von GitHub Classroom.

Bewertungs-Konfiguration

Die drei Dateien `unittests.json`, `lint.json` und `pylintrc` können entweder im Studi-Repo unter `.github/autograding/` liegen (Fallback aus der GitHub-Classroom-Zeit) oder manipulationssicher als Bundle im Config-Repo unter `<CLASSROOM>/autograders/<SLUG>/`.

Details und Vorrangregeln: Wie bewertet wird.

Sicherheit

Der Moodle-Token liegt **nur** hier, nicht in der Studi-Organisation. Der Bewertungs-Job bekommt gar keine Secrets: `autograde-runner.yaml` deklariert in seinem `workflow_call:`-Block nur `outputs:`. Der Job ist ausdrücklich **keine** Isolationsgrenze gegen selbst hinzugefügte Studi-Workflows — alles Vertrauliche gehört deshalb ins Config-Repo.

Weiter

- Überblick Classroom 50
- Klassenzimmer und Aufgaben verwalten
- Ein Klassenzimmer einrichten
- Template-Repositories