Schema Repository Setup Guide (with Code Snippets)

1. Repository Naming Best Practices

- Use lowercase with hyphens (kebab-case)
- Include semantic version (MAJOR.MINOR.PATCH)
- Example repo name: largegroup-schemas

2. Folder Structure

```
largegroup-schemas/
  schemas/
    json/
      account/v1.0.0/account.schema.json
    openapi/v1.0.0/openapi.yaml
  scripts/
    prepush.py
  .github/workflows/validate.yml
  requirements.txt
  README.md
```

3. Python Validation Script (prepush.py)

```
import subprocess, os, sys, json, yaml
from jsonschema import Draft202012Validator, exceptions
from openapi_spec_validator import validate_spec
from openapi_spec_validator.handlers import UrlHandler

def get_staged_files():
    result = subprocess.run(["git", "diff", "--cached", "--name-only"],
                            capture_output=True, text=True)
    return result.stdout.strip().split("\n")

def validate_json_schema(path):
    with open(path) as f:
        schema = json.load(f)
    Draft202012Validator.check_schema(schema)
    print(f"■ JSON Schema valid: {path}")

def validate_openapi_spec(path):
    with open(path) as f:
        spec_dict = yaml.safe_load(f)
    validate_spec(spec_dict, handlers=UrlHandler())
    print(f"■ OpenAPI spec valid: {path}")

if __name__ == "__main__":
    errors = False
    files = sys.argv[1:] if len(sys.argv) > 1 else get_staged_files()
    for path in files:
        if not path or not os.path.exists(path): continue
        try:
```

```
            if path.endswith(".schema.json"):
                validate_json_schema(path)
            elif path.endswith((".yaml", ".yml")):
                validate_openapi_spec(path)
        except Exception as e:
            print(f"■ Validation failed for {path}: {e}")
            errors = True
    sys.exit(1 if errors else 0)
```

4. requirements.txt

```
jsonschema==4.23.0
openapi-spec-validator==0.7.1
PyYAML==6.0.2
```

5. VS Code Tasks (.vscode/tasks.json)

```
{
  "version": "2.0.0",
  "tasks": [
    {
      "label": "Validate All (pre-push)",
      "type": "shell",
      "command": "python scripts/prepush.py",
      "group": { "kind": "build", "isDefault": true }
    }
  ]
}
```

6. GitHub Actions Workflow (.github/workflows/validate.yml)

```
name: Validate Changed Schemas
on:
  push:
    branches: [ "main", "develop" ]
  pull_request:
    branches: [ "main", "develop" ]

jobs:
  validate:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - uses: actions/setup-python@v5
        with:
          python-version: "3.11"
      - run: pip install -r requirements.txt
      - id: changes
        uses: tj-actions/changed-files@v45
        with:
          files: |
            **/*.schema.json
```

```
          **/*.yaml
          **/*.yml
      - run: |
          if [ -n "${{ steps.changes.outputs.all_changed_files }}" ]; then
            python scripts/prepush.py ${{ steps.changes.outputs.all_changed_files }}
          else
            echo "No schema changes detected."
          fi
```

7. README.md

```
# Large Group Schemas

This repository contains canonical JSON Schema and OpenAPI specifications.

## Purpose
- Define data models (JSON Schema Draft 2020-12)
- Define API specifications (OpenAPI 3.0+)
- Ensure consistency and automate validation

## Structure
schemas/json/, schemas/openapi/, scripts/, .github/

## Local Validation
pip install -r requirements.txt
python scripts/prepush.py

## GitHub Actions CI
Validates only changed schema files on push/PR.

## License
MIT License © 2025 Large Group Schema Team
```