Repo Documentation

Hardhat Enterprises

AutoAudit

—

Guide:

How to Contribute to the AutoAudit Repository

GitHub Repository Guide

AutoAudit Monorepo

The main repository is available at https:\\github.com\Hardhat-Enterprises\AutoAudit.

This monorepo centralizes codebases from all specialist teams – including backend/api, frontend, security, and engine services.

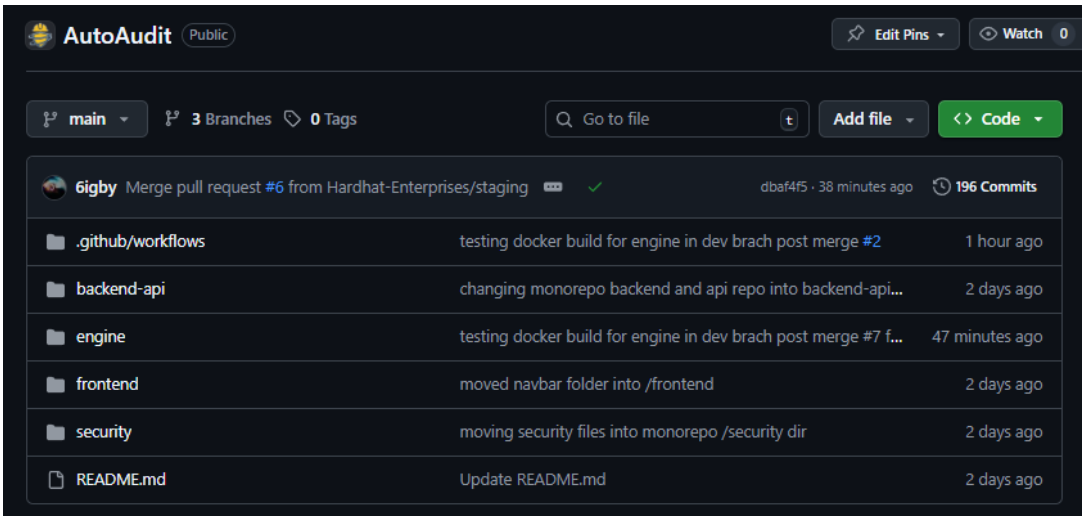
Its modular structure ensures:

- Centralized code and build management
- Clear separation of team responsibilities
- Streamlined integration, automated testing, and deploys

Repository Structure:

AutoAudit

```
.github/workflows/ # GitHub Actions / DevOps Team
/backend-api/      # API and Backend Team
/frontend/         # Frontend Team
/security/         # Security Team
/engine/           # Engine Team
```



How the Monorepo Works

Team Specific Folders:
Each team works inside their assigned root directory (/backend-api, /security, /engine, /frontend). This keeps contributions modular and auditable.

Path Filters in CI\CD:
Whenever a Pull Request (PR) is opened, GitHub Actions path filters automatically identify which folders were modified.

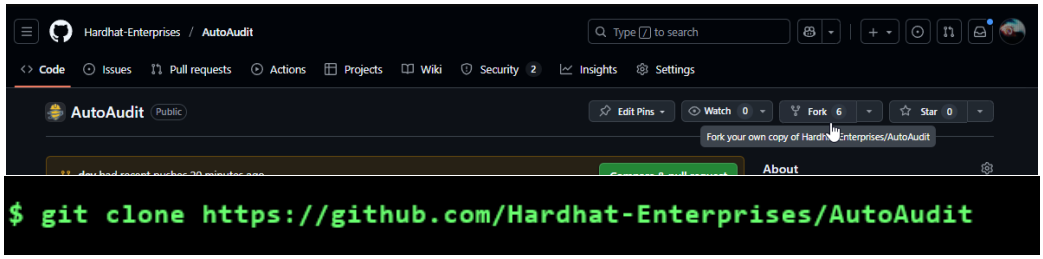
- Only build, lint, and test jobs for the affected teams\services are triggered.
- This speeds up CI\CD and prevents unrelated service failures.

PR Workflow:
All team contributions are made as PRs against the shared dev branch. Each PR is reviewed, with code quality, security, and tests run automatically on relevant parts of the codebase.

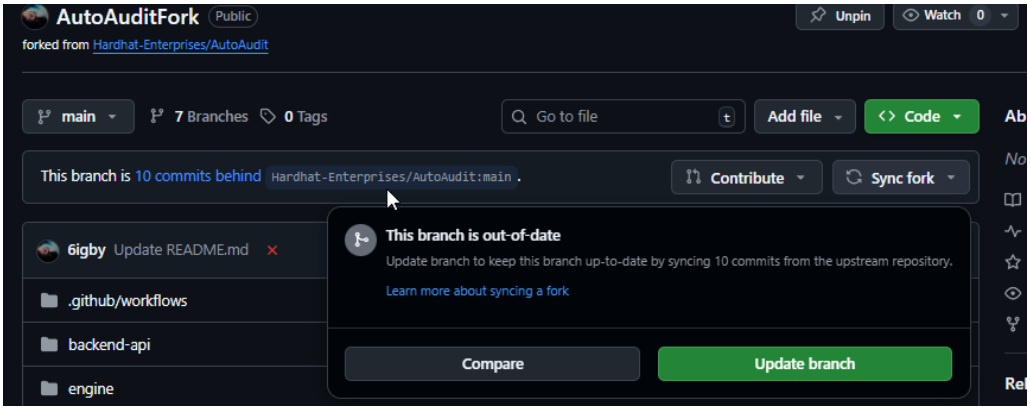
How to Start Working on the Repo

Using Git Bash: https:\\git-scm.com\downloads

1. Fork the main repo, and clone it onto your local machine:



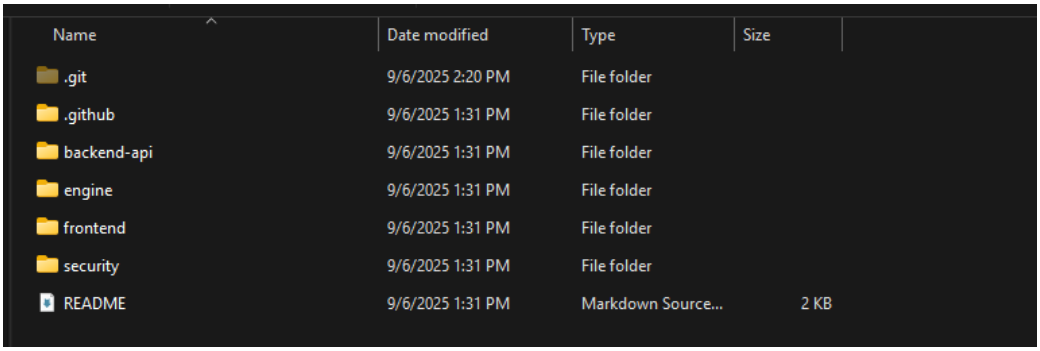
   Replace the above link with your forked repo link, sync the fork to mimic the main repo:



2. Inside the terminal, change the directory into the cloned repo:

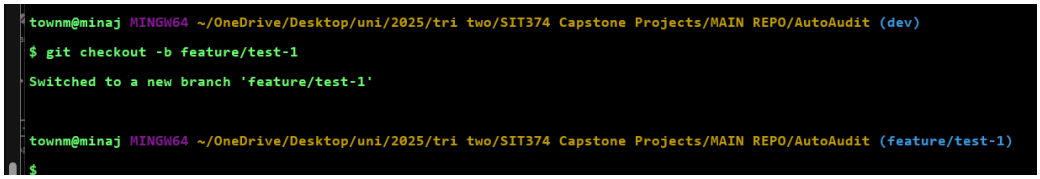
```
$ cd AutoAudit
```

4. Work in your teams ***source*** folder, this is where you will develop code and test locally.



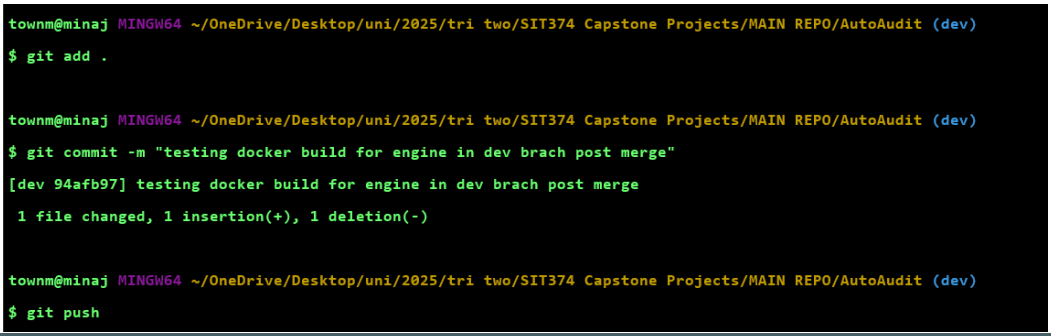
Backend/API Team Members: /backend-api
Engine Team Members: /engine
Frontend Team Members: /frontend
Security Team Members: /security

5. Create a new feature branch (locally). Use descriptive names, e.g., bugfix\fix-security-vuln.

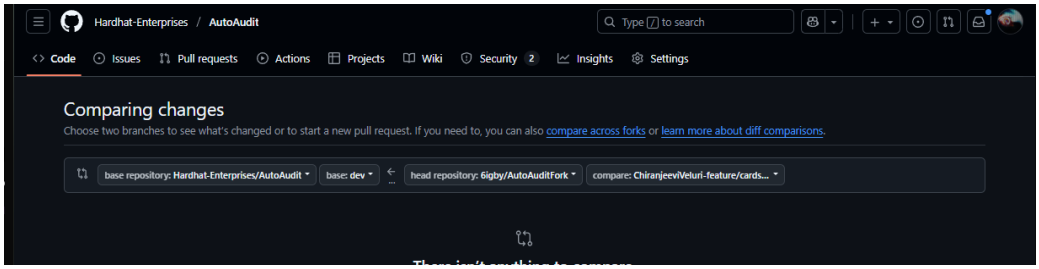


6. Commit and push your work after completion of the feature you have been working on.

- git add .
- git commit -m "feature comment"
- git push origin 'branch-feature…' (replace branch-feature with the new branch you have created).



6. Open GitHub, and locate your fork and select ‘Pull Requests’. Select ‘New Pull Request’:
   - Keep the base repo as Hardhat-Enterprises/AutoAudit.
   - Change base: to ‘dev’.
   - Head repository: your fork should automatically be selected.
   - Compare: the new branch with the feature you have pushed in previous steps.
   - Select Open Pull Request.



   If the above doesn’t work, please contact your team lead or the DevOps team lead.

PR’s from ‘dev’ to ‘staging’ Branches

Why:

The staging branch acts as a mirror of main for QA with final verification by leads. It’s the last stop before production.

How:

- Once dev accumulates many approved changes, a lead or maintainer opens a PR from dev -> staging.
- All team leads can review integrated work, run final integration tests, and sign off.
- This process ensures that only fully reviewed, integrated code reaches the staging environment.

From Staging to Main, and Production Deploy

After complete validation in staging, a PR is opened from staging to main.

Upon PR merge into main:
- The CI\CD pipeline *will* trigger a full production build.

After GCP is set up:
- Automated deployment will push the code to GCP environments (prod cluster\servers).

- All secrets and configs will be securely managed through Google Secret Manager.

This process allows for safe, incremental releases ensuring compliance, traceability, and fast rollbacks if required.