

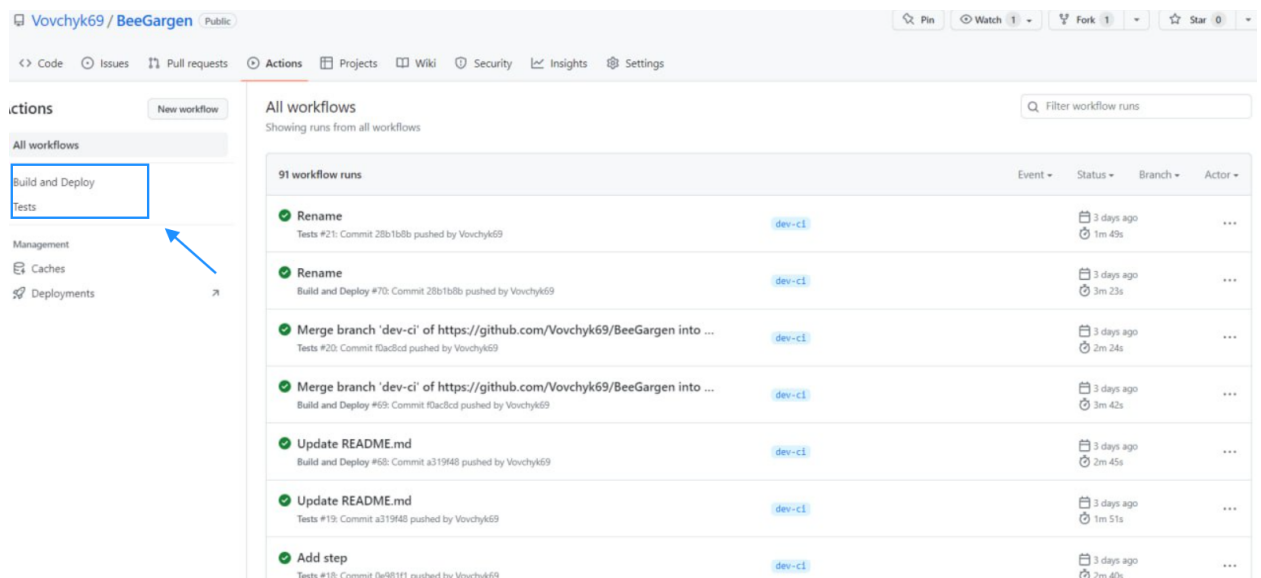
## Bee Garden – CI/CD

In our full stack application **Heard**, we have implemented a CI/CD pipeline to ensure efficient and reliable code deployment.

The pipeline consists of three primary actions that ensure the reliability and quality of our application and facilitates smooth and efficient deployment:

- ***Build and Deploy to Heroku.***

Whenever a commit is made to a certain branch, our CI system automatically triggers a build process that compiles the code and deploys it to Heroku. This step allows us to quickly propagate changes to our production environment.



The screenshot displays the GitHub Actions interface for the repository Vovchyk69/BeeGargen. On the left sidebar, the 'Actions' tab is selected, and the 'Build and Deploy' workflow is highlighted with a blue box and an arrow. The main area shows a list of 91 workflow runs. The first few runs are detailed below:

Run Name	Event	Status	Branch	Actor	Time
Rename	Tests #21: Commit 28b1b8b pushed by Vovchyk69	dev-ci	3 days ago	...	1m 49s
Rename	Build and Deploy #70: Commit 28b1b8b pushed by Vovchyk69	dev-ci	3 days ago	...	3m 23s
Merge branch 'dev-ci' of https://github.com/Vovchyk69/BeeGargen into ...	Tests #20: Commit f0ac8cd pushed by Vovchyk69	dev-ci	3 days ago	...	2m 24s
Merge branch 'dev-ci' of https://github.com/Vovchyk69/BeeGargen into ...	Build and Deploy #69: Commit f0ac8cd pushed by Vovchyk69	dev-ci	3 days ago	...	3m 42s
Update README.md	Build and Deploy #68: Commit a319488 pushed by Vovchyk69	dev-ci	3 days ago	...	2m 45s
Update README.md	Tests #19: Commit a319488 pushed by Vovchyk69	dev-ci	3 days ago	...	1m 51s
Add step	Tests #18: Commit 0a981f1 pushed by Vovchyk69	dev-ci	3 days ago	...	2m 40s

We choose Heroku as a main deployment platform because of several reasons:

1. *Ease of Use*. Heroku provides a clear and straightforward deployment process. It has good support for Node.js applications.
2. *Platform as a Service*. It charges you only for the resources that you actually use and takes care of horizontal scaling and load balancing.

Latest activity

[All Activity](#) 



vovchykhalamaha@gmail.com: Deployed 28b1b8bf

May 14 at 1:58 PM · v90 · [Compare diff](#)



vovchykhalamaha@gmail.com: Deployed f0ac8cda

May 14 at 1:57 PM · v89 · [Compare diff](#)



vovchykhalamaha@gmail.com: Build succeeded

May 14 at 1:57 PM · [View build log](#)



vovchykhalamaha@gmail.com: Build succeeded

May 14 at 1:57 PM · [View build log](#)



vovchykhalamaha@gmail.com: Deployed a319f48c

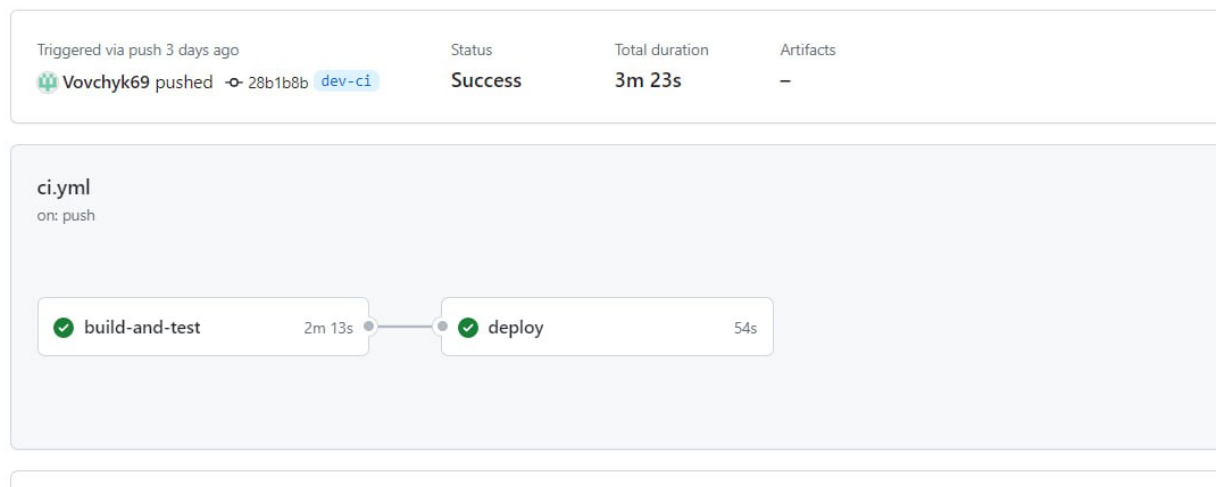
May 14 at 1:42 PM · v88 · [Compare diff](#)



vovchykhalamaha@gmail.com: Build succeeded

- **Test Verification:**

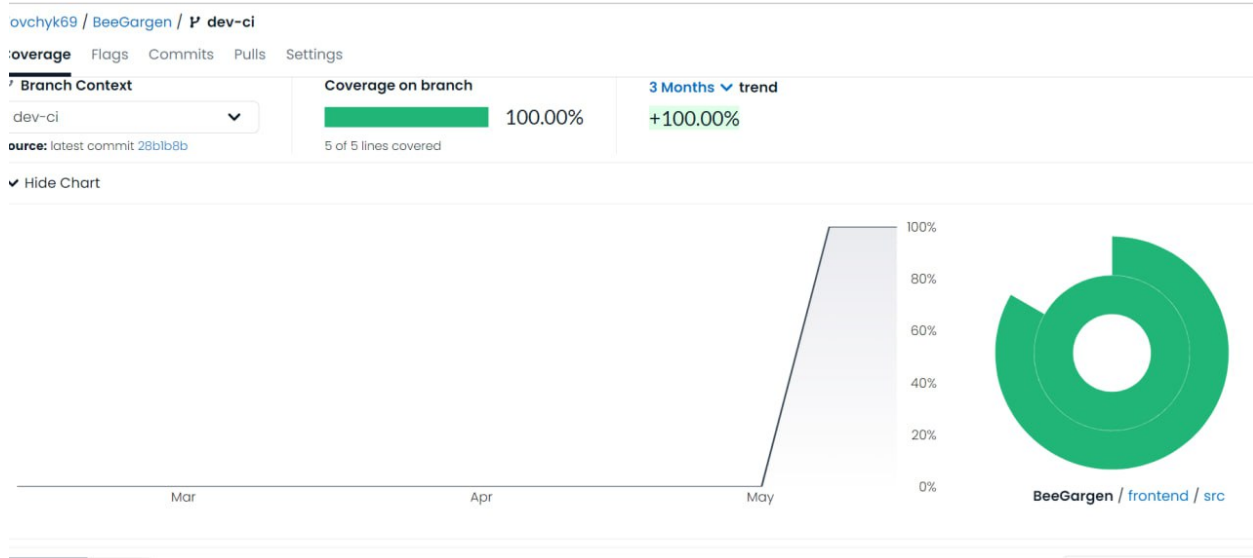
To maintain the integrity and ensure quality of our codebase, we have implemented tests verification steps. As part of our CI pipeline, these tests are executed after the build process. We verify that all tests pass successfully, ensuring that the code meets the best quality standards, best practises and behaves as expected.



- **Test Coverage Validation:**

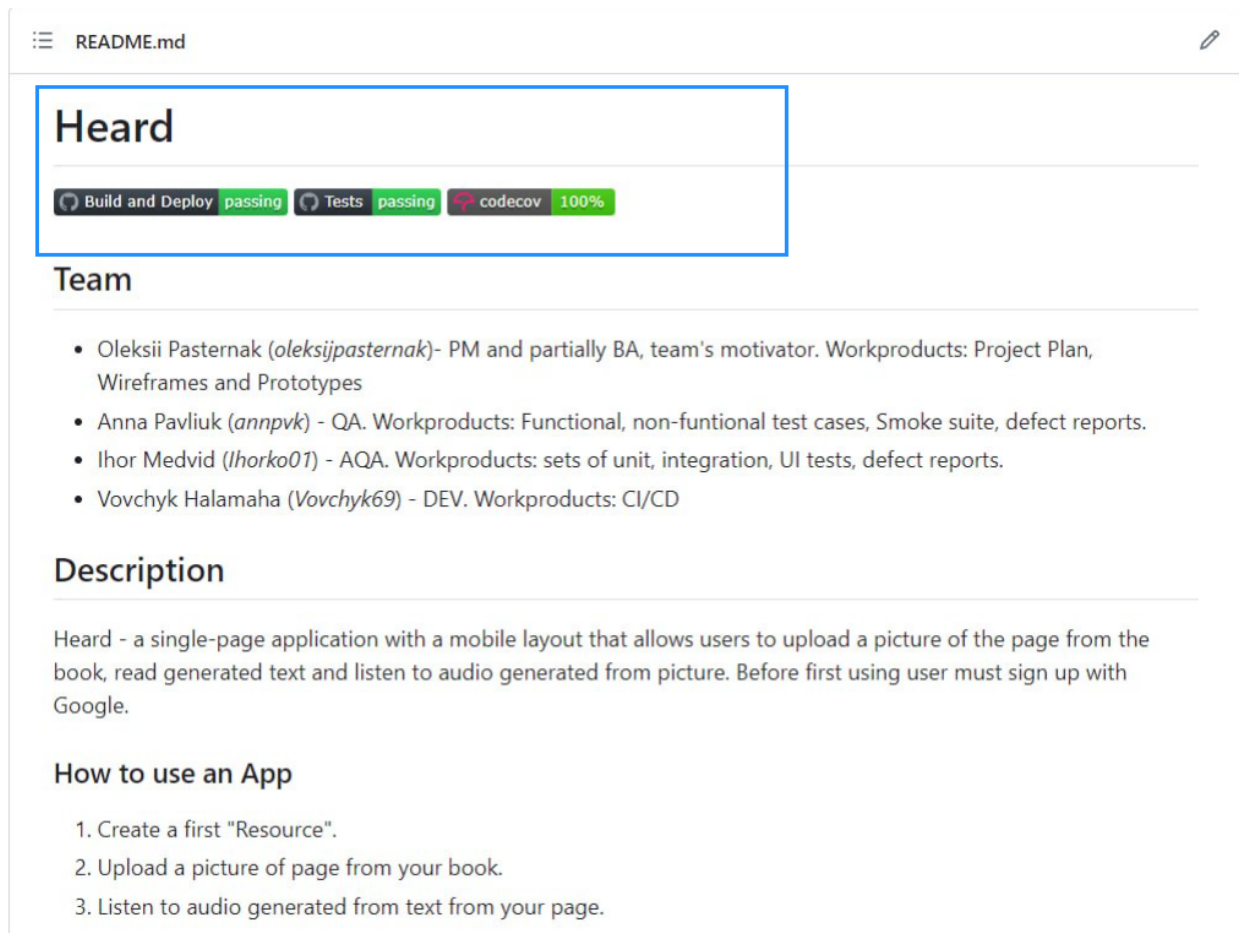
Code coverage is an additional step that is considered as a very important step to meet the code quality expectations.

We utilize **CodeCov** tool that allows us to measure the extent to which our code is covered by tests. In the CI pipeline, we analyze the test coverage and enforce a minimum threshold. This practice ensures that our codebase has good test coverage, reducing the likelihood of unfound issues.



It's important to mention that we do not have support for blue-green deployments due to Heroku's limitations that we discovered in the middle of development. Yes, It's indeed possible to have multiple environments including staging environments, but there is no capability of redirecting traffic between different workers.

However, we have a staging environment in place which acts as a preprod environment to provide testing and previewing changes before actually deploying to production. In situations when working on commercial products, this technique helps mitigate risks and ensures a smoother transition deployment and updates.



Also our CI/CD pipeline incorporates a rollback mechanism. If either the build process fails or any tests do not pass, the deployment to Heroku will not be even executed. If deployment fails, it will be rolled back to the previous state.

## Conclusions

By implementing this CI/CD pipeline, we introduced better processes to the team for our project, more frequent and reliable deployments, and ensured a high level of code quality. To sum up, these practices play a huge role in software development and contribute to the overall stability of the system.