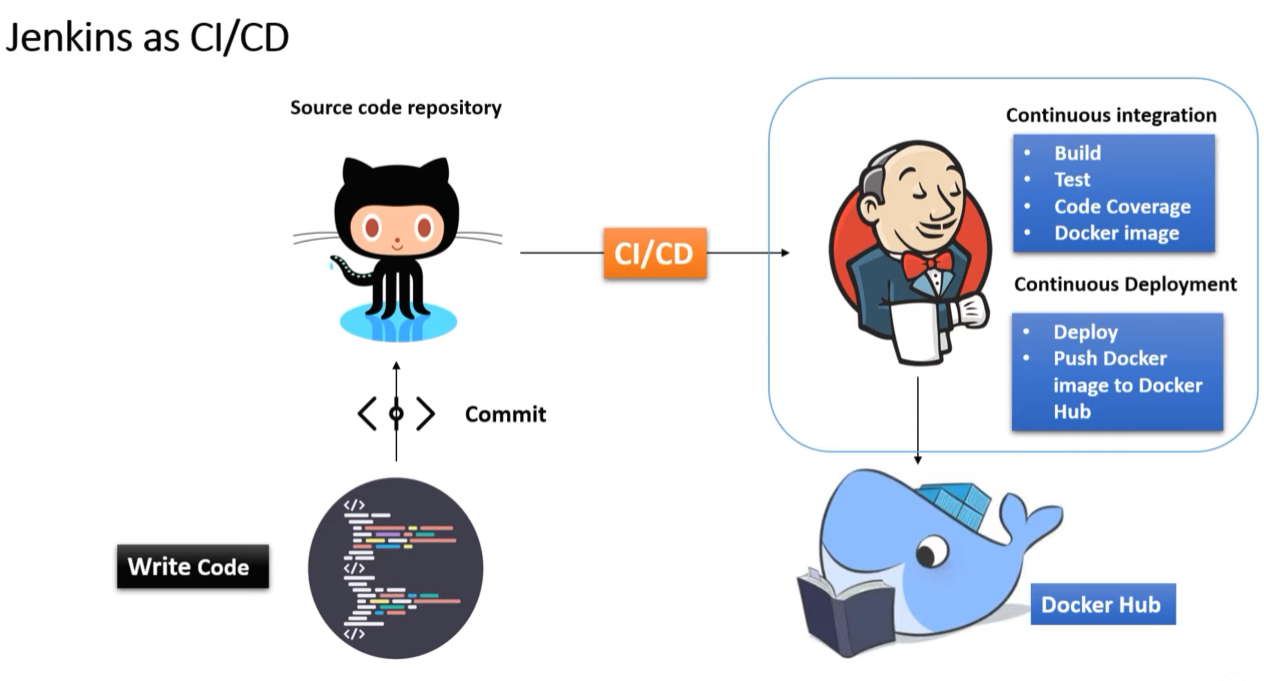


Why GitHub Actions when we have lot of CI/CD tools like Jenkins, GitLab, circleci, etc?



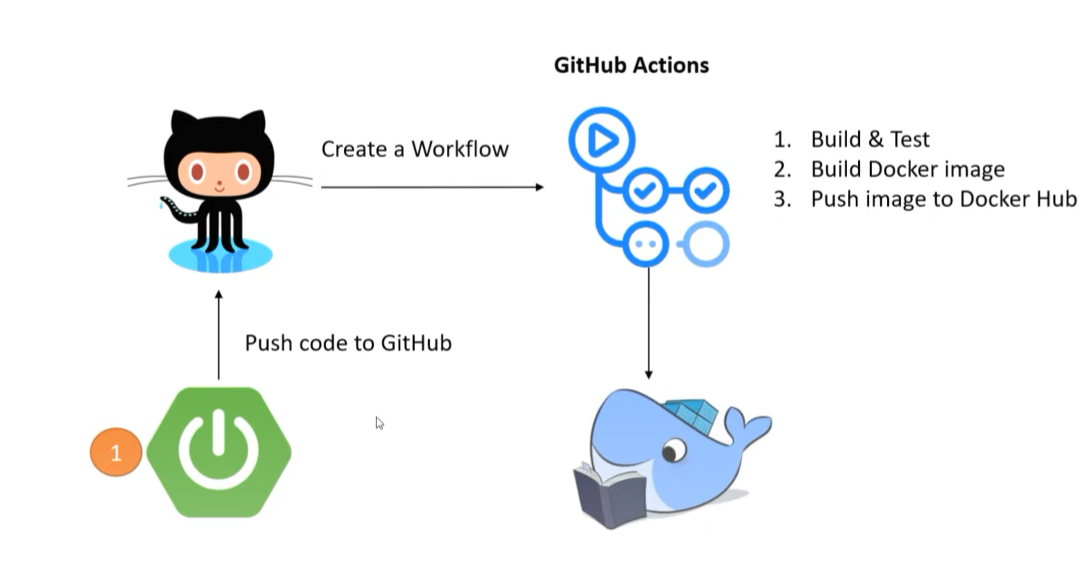
When we use Jenkins, entire CI/CD flow is taken care by it which is a 3rd party tool and in realtime this require and additional resource, atleast a VM with some memory to install it in hosted server.

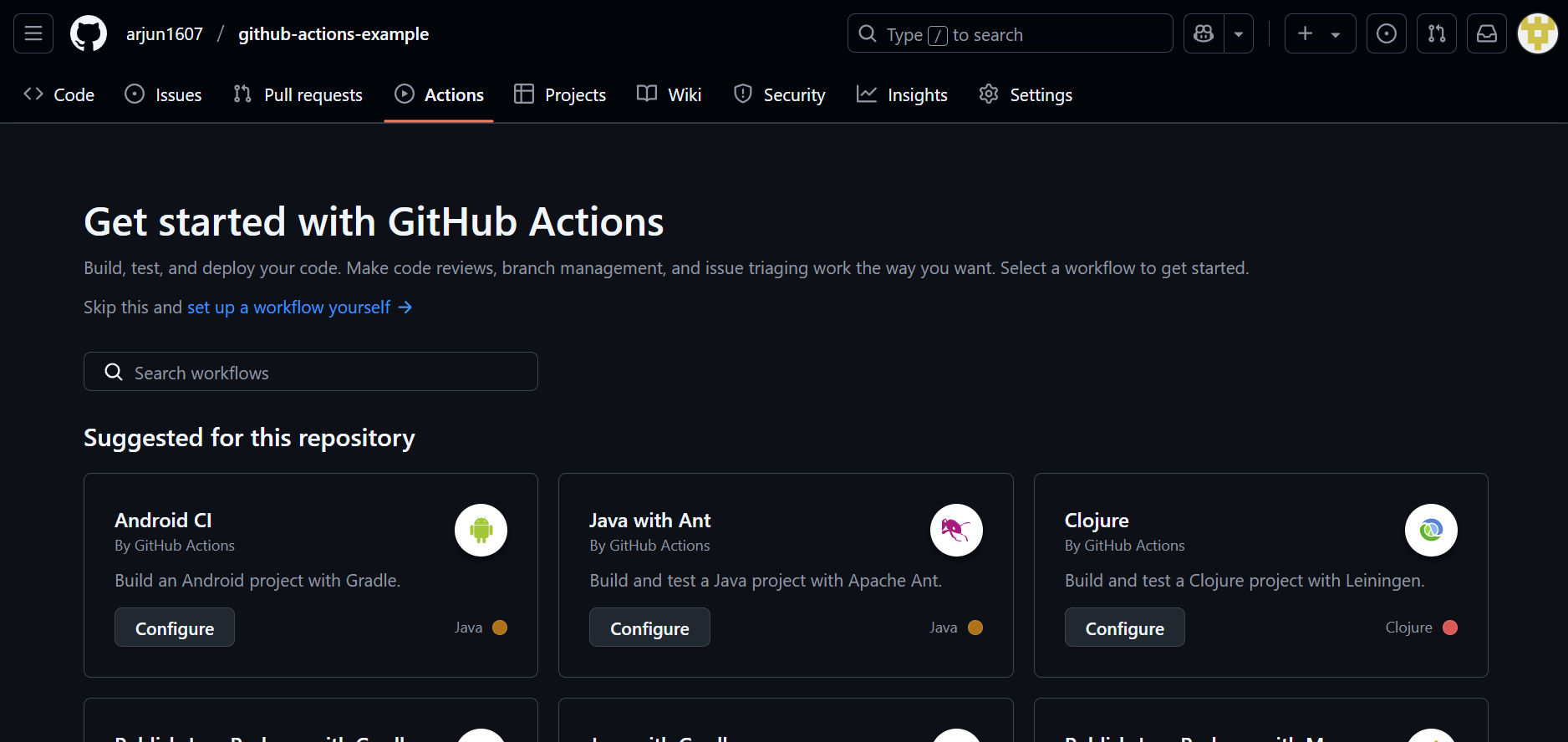
Basically you need a separate infrastructure for this which is completely an extra burden for a developer.

So when we use Github Actions where our source code repo and CI/CD pipeline will run on the same platform. Using it we don’t need any dedicated infrastructure to configure Github Action or to perform the CI/CD workflow. Github Actions help us to automate our software development workflow in the same place where we store our code.

With Github Actions we can build end to end continous integration and continous development pipeline directly in our repository itself. And the major advantage that the workflow can run in Linus, Mac OS, Windows, any type of OS on github hosted machines.

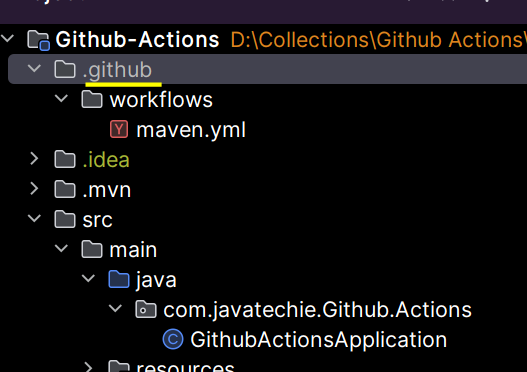
In the case of private repositories, then we have the flexibility of configuring our own machine with the choice of our environment.



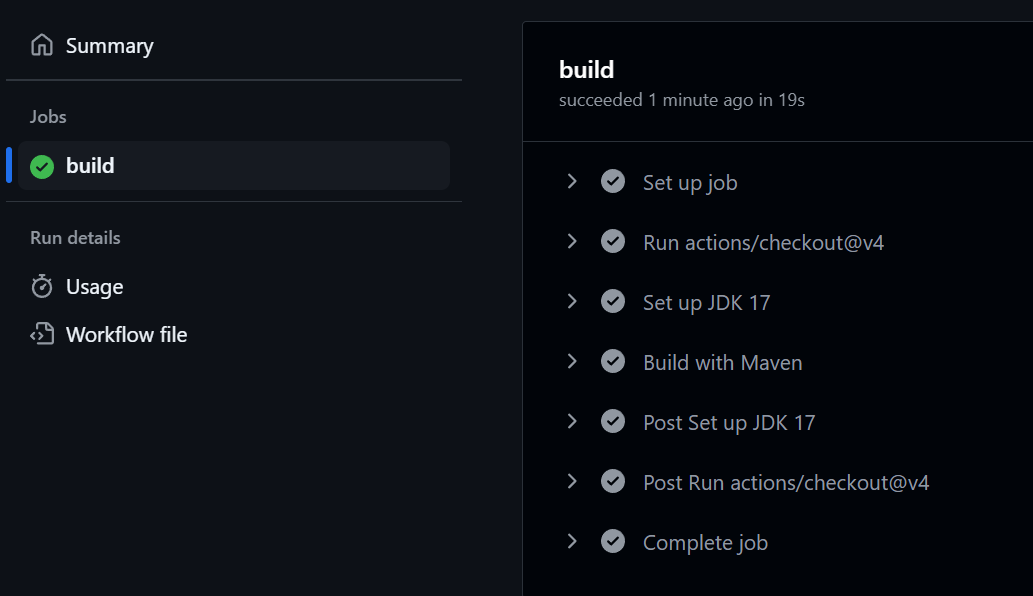


Select “Java with Maven” workflow and configure.

Edit maven.yml and create pull request to base branch (master).



After merge we can see this github directory in our project. In this workflow directory we need to mention the actions we need to perform to complete the CI/CD pipeline.

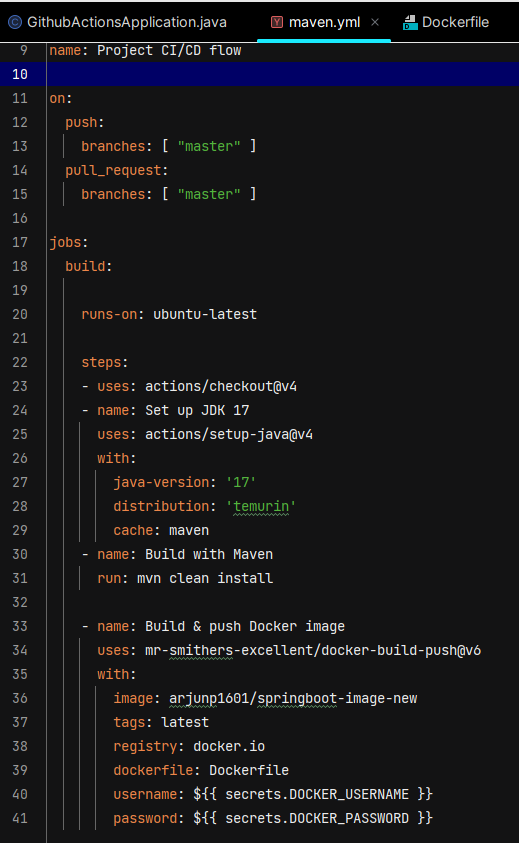


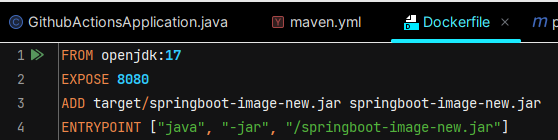
Add more steps –

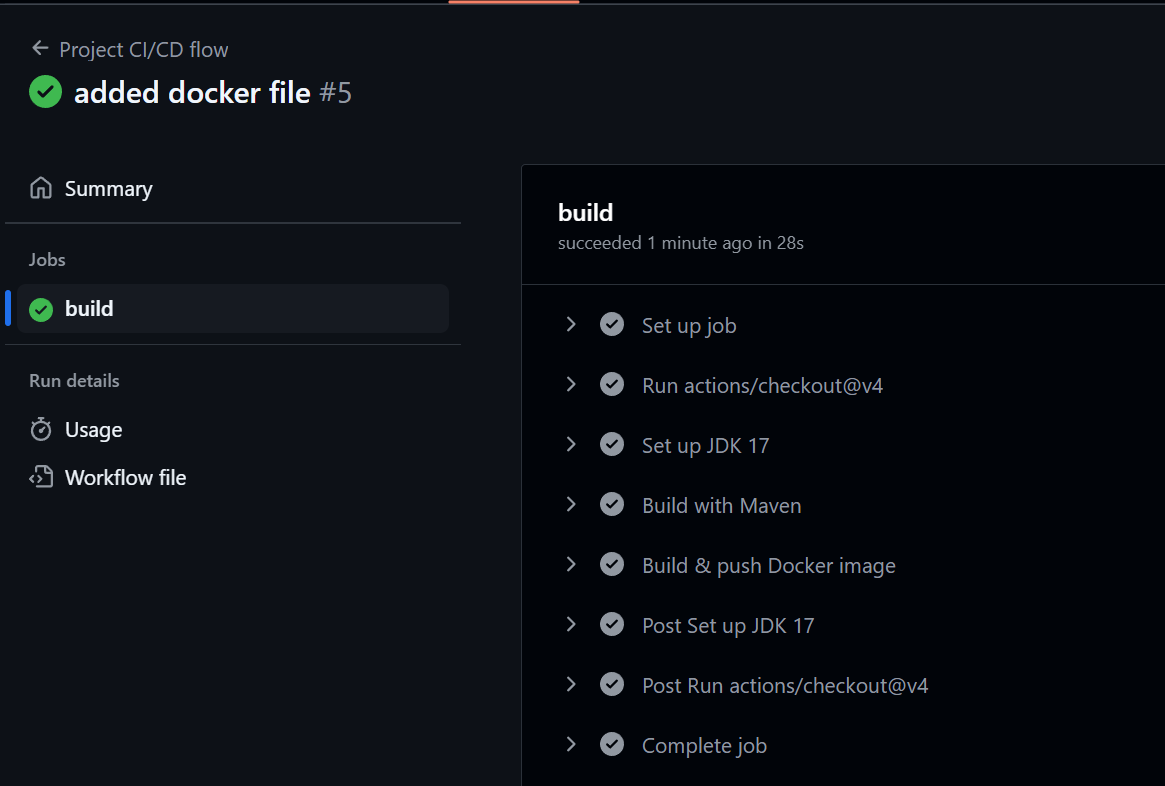
1. Login to your docker hub
2. Build a docker image
3. Push that image to docker hub

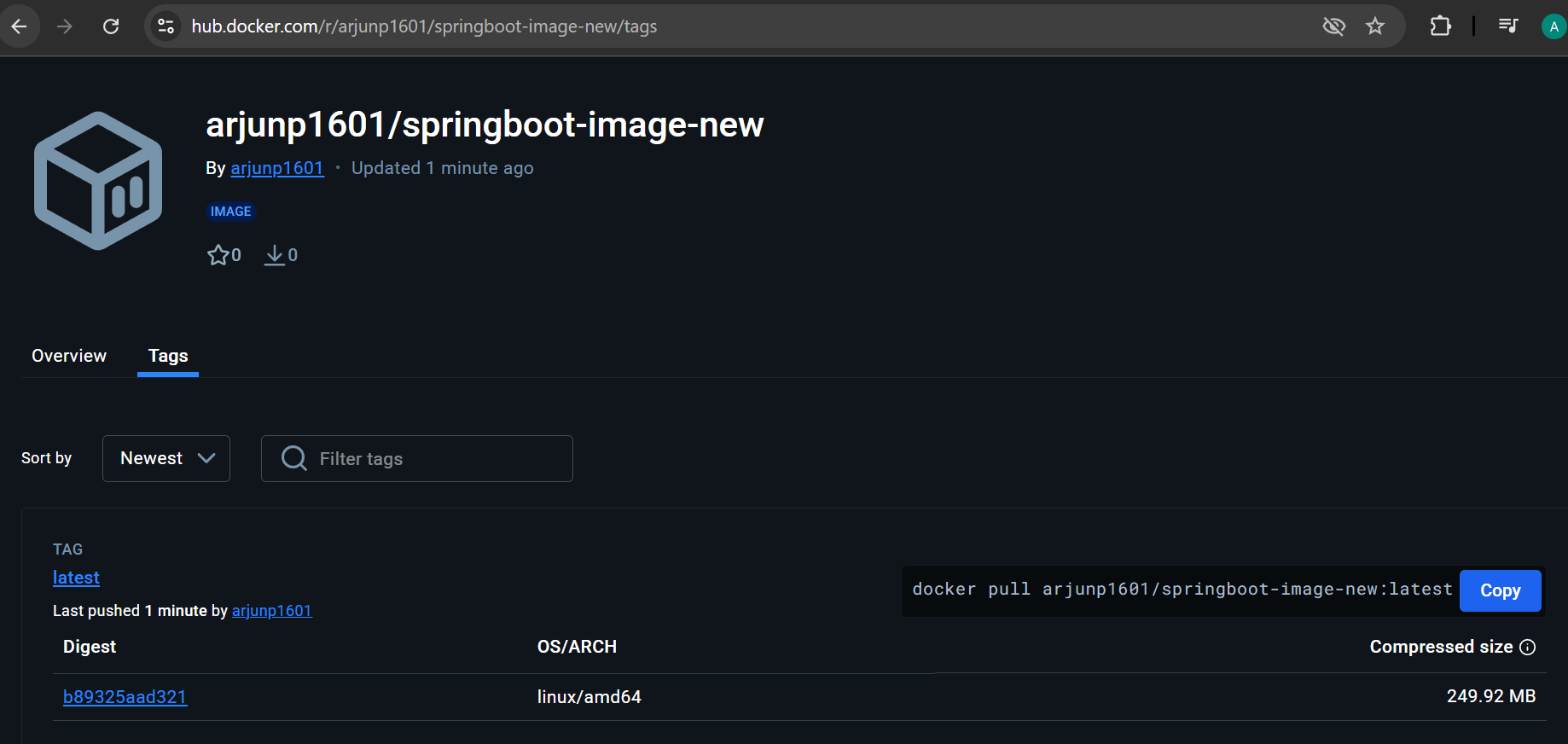
Rather than we do it, the github provides readymade action which we can directly use to create docker image and push that to docker hub.

Github Actions will reach your Dockerfile to create image.









Now you can pull the image and run on local.

docker run -p 8080:8080 arjunp1601/springboot-image-new