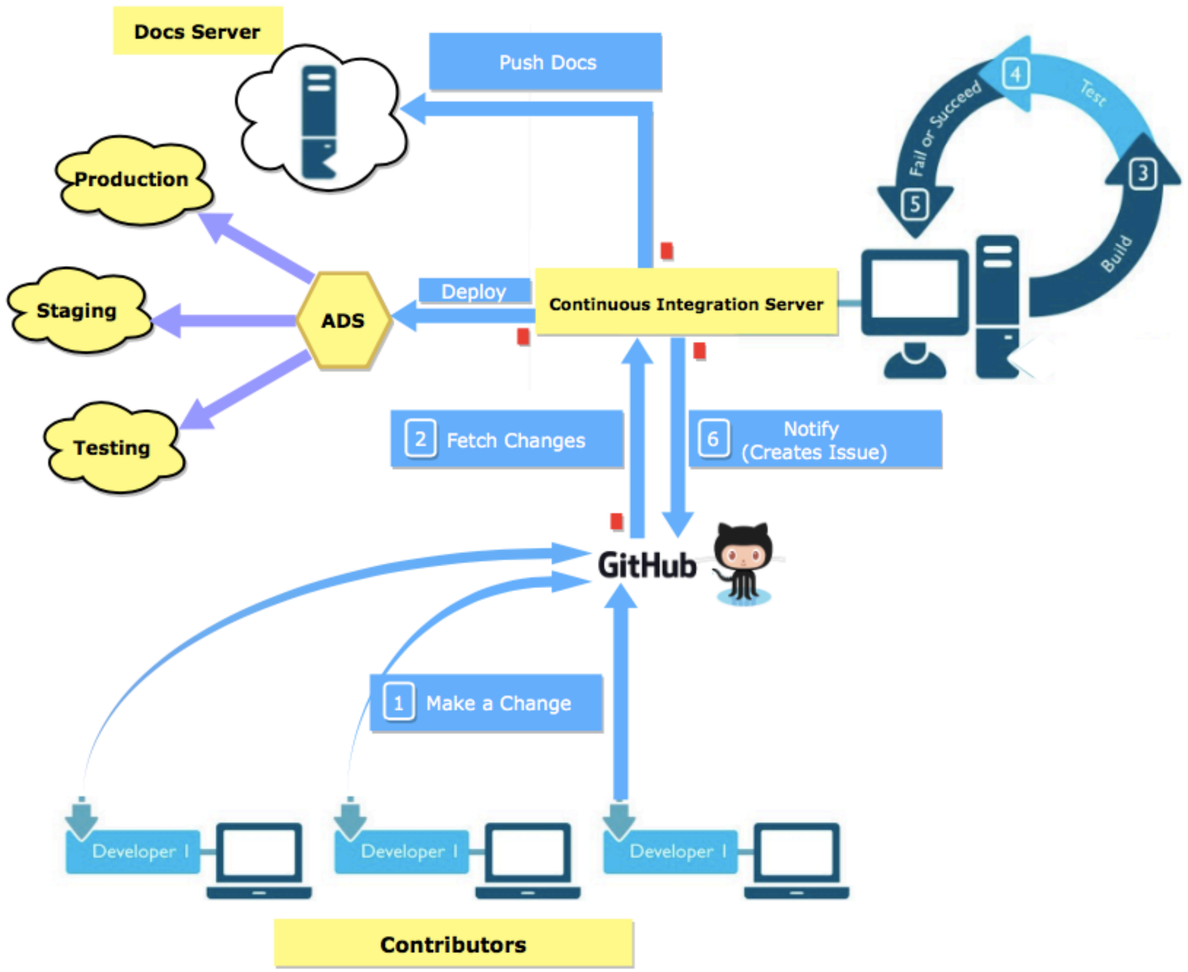
Continuous Integration Micro-service for Virtual Labs

AIM

To build a micro-service which continuously builds and tests changes made to the code in a Github repository.

INTRODUCTION

- Continuous integration (CI) is the practice of merging all developer working copies to a shared mainline.
- Continuous integration most often refers to the integration stage of the software release process and entails both an automation component(CI) and a cultural component (frequency of integration).
- The key goals of continuous integration are to address bugs faster, improve software quality, and to quicken the build process.
- A lab is currently defined as a set of related experiments and is a web application. Each lab is a repository on Github.
- With continuous integration, developers from around the world can contribute to Virtual Labs.
- The continuous integration service detects commits to the shared repository, and automatically builds and runs unit tests on the new code changes to immediately surface any functional or integration errors.



METHODOLOGY

- Our basis of the whole implementation was to build a micro-service which could be scaled over time with the requirement.
- The continuous integration service was provided in the form of APIs to the contributors/developers.
- The functionality provided by Github web-hooks was used to invoke these APIs whenever a change was made to the repo.
- The JSON payload sent by these webhooks was used to extract information about the repository to carry on further processes involved in continuous integration.
- APIs were built using flask to pipeline this continuous integration process with the current auto deployment process which would deploy the application in the production environment after the successful build and tests.
- Security issues were also dealt with by making this service exclusive to only repositories part of Virtual-Labs and VLEAD(this is configurable as per the requirement).
- This service is hosted using NGINX and G-Unicorn along with the already running services.

RESULTS

- The service was hosted in the testing environment.
- Unit test cases and integration tests created during the development ran successfully.
- Further one of the existing services from VLEAD were taken and tested with our continuous integration service.
- Our Service was able to provide it's functionality without any issues.

CONCLUSIONS

- Micro-services are a great way to cater to our requirements and offer a simple solution.
- Instead of relying on existing continuous integration services like Jenkins or Travis which weren't able to satisfy all of our functional requirements, we were able to provide them by developing our own micro-service.

Software Engg Research Center

Interns: Srikara Srikanth Pakala, MEC, Hyderabad
Savar Mehrotra, MIT, Manipal
Mentor: Sripathi Kammari, VLEAD