

* DevOps
* Any piece of software requires development and operations.
* Portmanteau (combination of two words)
  + DEVelopment
    - Coding
    - Testing
    - Planning
  + OPerationS
    - Deployment
      * Getting servers
      * Buying a domain name
    - Monitoring
      * Making sure all the servers are running.
    - Performing updates
      * Performing updates at times to create minimal disruption.
* In the past these were two very separate fields.
  + Developers did there thing
  + IT infrastructure people were responsible for operations.
* DevOps tries to combine the two and make them feed into each other.
* Web Applications are never completed.
* They just keep getting incrementally improved.
  + A developer might write a new feature for an application. DEV
  + When it is deployed he might get feedback from the monitoring logs. OPS
  + Real world example.
    - Developer writes a new save favorite orders feature for a pizza website.
    - Once deployed we can monitor the application.
    - We can see traffic is up 60%
    - A new feature is proposed to expand upon favorite orders.
* DevOps is heavily reliant on automation tools.
  + GitHub for source control.
  + Jenkins to quickly build and test applications.
  + Cloud technologies that allow people to easily and automatically spin up vm or databases to deploy applications.
* Google does not have IT deployment people.
  + SRE (Site Reliability Engineer)
  + Google does not see reliability in finished applications as a deployment issue per se.
  + They see reliability as part of the code as well.

CI,CD, CDD

* Continuous Integration
  + A process by which a team of developers will put all of their code into a central repository to build an application.
  + Every time a developer pushes their code to the remote repository. It will automatically test and analyze their code.
    - We saw this when I made pushes to a repo with an attached sonarcloud.
  + Benefits to CI.
    - By constantly pushing up code to a repo and pulling code progress is never lost.
      * A hardware failure on someone’s local computer does not mean days of work is lost.
    - Allows Developers to get feedback very quickly.
      * Auto generated code analysis and test reports.
      * Peer review by other developers.
* Continuous Delivery
  + One step beyond CI.
  + IF all the tests are passed then a deliverable is created and stored somewhere.
    - A deliverable is just a finished piece of software.
    - Usually stored in a staging area.
    - Examples
      * .jar
      * Shell script
      * .exe
* Continuous Deployment
  + One step beyond CD
  + IF all the tests are passed the deliverable will automatically be deployed to end users.
    - Amazon does 9000 deployments a day.
    - Theis process is all automated.
* This whole process of setting up CI or CD or CDD is called setting up a pipeline.

SonarCloud

* Static Code Analysis
  + It reads your code and tries find problems in it.
    - Bad coding practices
    - Security vulnerabilities
    - Potential bugs
  + Also offers suggestions.
* It does not execute your tests per se to see if they work.
  + Computer peer review

Continuous Integration steps

* The main branch will have a skeleton project initially.
* NEVER push directly to the main branch.
* Create a new branch named after whatever feature you are working on.
* Push your new code up to that branch.
* Pull request from that feature branch into main.
* Other developers usually seniors will perform a code review.
  + Look at your source code.
  + Look at your tests.
  + Look at your build to make sure your tests work correctly and the application is not broken.
* Approve your pull request.
* Merge that feature branch into the main.
  + They might have to resolve merge conflicts in order to proceed

Jenkins

* Jenkins is a continuous integration tool, build server.
* Jenkins can be set up to read repositories.
* It can detect changes to those repositories and build the application using the latest version of the code.
* Often the built application (deliverable) is then deployed on a separate server. (deployment server)
* Jenkins terminology
  + Job
    - A project in Jenkins
    - Configure
      * URL of the repo
      * How to build the project
        + We used gradle.
      * When to build the app
        + Press build now.
        + GitHub webhook
  + Build
    - Any time Jenkins runs build in a job.
      * Blue spheres indicating a successful build.
  + Webhook
    - Anytime we pushed to the repo.
    - GitHub would look at the webhook and send a message to that url telling Jenkins it was time to build a new version.