**Application**

Few lines of code to provide the functionality or a software that serves the needs of end users.

Types of Application:

1. Standalone/Desktop Application: The application you have downloaded in your machine (local machine), which we can only use when we are in front of the application, and we are owner of the application. They will work without dependency. Example: Paint, Calculator
2. Web Application: Application which we can be access from anywhere anytime from any kind of device but required internet connection. Example: Ansible page
3. Dynamic Application: A dynamic application requires users to log in with a username and password to access personalized data. These applications authenticate users to ensure they are valid and display the relevant data accordingly. They are interactive, responsive, and store persistent data, meaning the data remains available even after the user logs out. These applications are designed to be highly available and accessible. Example: Banking Sites or Application
4. Enterprise Web Application: An enterprise web application is a large-scale software system designed to meet the needs of an organization or business. These applications are used to manage business processes, handle large amounts of data, and support complex workflows within an organization.

A dynamic application could be a small part of an enterprise web application, but an enterprise web app typically includes many additional features for security, scalability, integration, and business-specific needs that go beyond what most dynamic applications offer.

**Software development lifecycle**

A diagram of a project

Description automatically generated

**Traditional Waterfall Model:**

When you fixed requirements then we can use this model. No new requirements can be accommodated while project is in progress. If you want to do changes then it will costly. There is no faster deliver and no feedback given to developers.

Design 🡺 Code 🡺 Test 🡺 Deploy

This can be done once in project, if we want to change anything or update anything then we will require more cost to do that as we have plan from the zero again.

**Agile Methodology:**

In the agile methodology each project is broken up into several iterations. Iteration means sprint. Here we discover new change, and we design and develop. If we had feedback to change or develop new thing then we can do it in next sprint. But we can not used automation tool for deployment, and we will have fixed date and time for deployment. It means we are delivering the fixes at fixed date and time.

Design 🡺 Code 🡺 Test 🡺 Error 🡺 Design 🡺 Code 🡺 Test 🡺 Ok 🡺 Deploy

This can be done once a iteration or sprint. It means we are deploying once an iteration. So, If we have bug fixes then we have to wait for next sprint.

**DevOps [Development + Operation]**

DevOps is a practice that allows team to manage the entire application development life cycle, that is development, testing, deployments and monitoring.

In devops we have phase like:

Plan 🡺 Code 🡺 Build 🡺 Test 🡺 Release 🡺 Deploy 🡺 Operate 🡺 Monitor 🡺 If error or we have feedback from client to change then same process can be followed.

Difference from Agile Methodology:

We can use multiple tools for this, and these tools are dependent on each other. So, we have faster deployments using these tools for automation with less manual error.

Design 🡺 Code 🡺 test 🡺 Deploy

Agile with CD will provide you an DevOps. There is a frequently deployment on environment. This will eliminate the need for AGILE and Traditional Waterfall method.

**Principle of DevOps:**

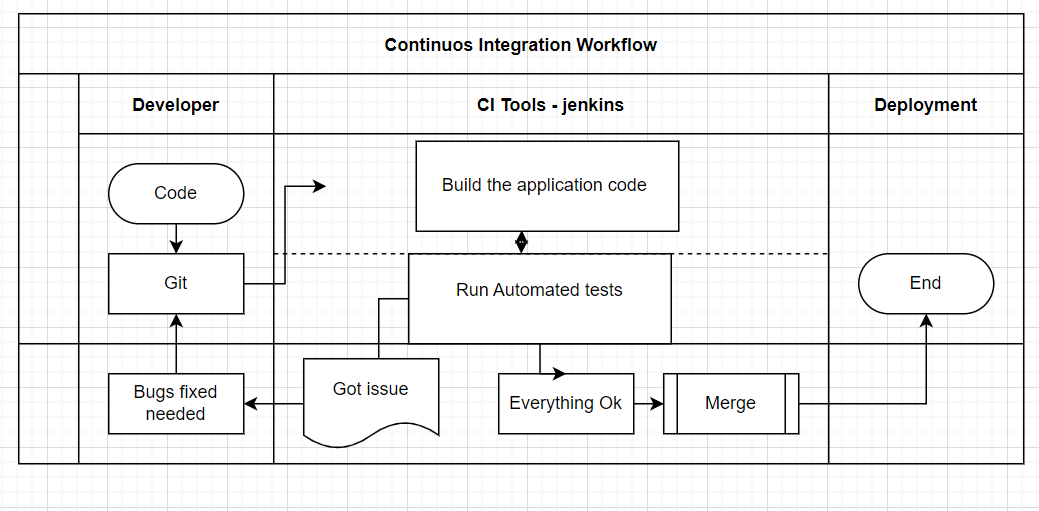
1. Continuous Version Control: Git GitHub Bitbucket GitLab SVN VS-Code
2. Continuous Integration: tools: Jenkins (Not a build tool) will integrate with build tools like Maven, Gitlab CICD pipelines, GitHub actions, Bamboo, Circle CI, TeamCity
3. Continuous Delivery: Used an automation tool to deploy the code in DEV/Test/PRE-PROD/Stagging server and manually deploy to production
4. Continuous Deployment: Complete automation for deployment no manual process involved
5. Configuration Management: IAC, Ansible, Terraform
6. Continuous Monitoring:
7. Communication and Collaboration

What is continuous Integration?

Continuous Integration is aligned with Build phase in SDLC. Build is nothing but compilation of code, testing of code and making package, tar the code and send it to environment.

Compiler: So, for writing code developer use language like Java, .net, c#, python. As the computer only know language of binary files, so compiler will convert the code in computer known language.

Developer 🡺 write code and push code to 🡺 Github 🡺 CI Sever ( Jenkins ) 🡺 Ask to build tool to do compile + test + package the code in the format required🡺



What is Configuration Management?

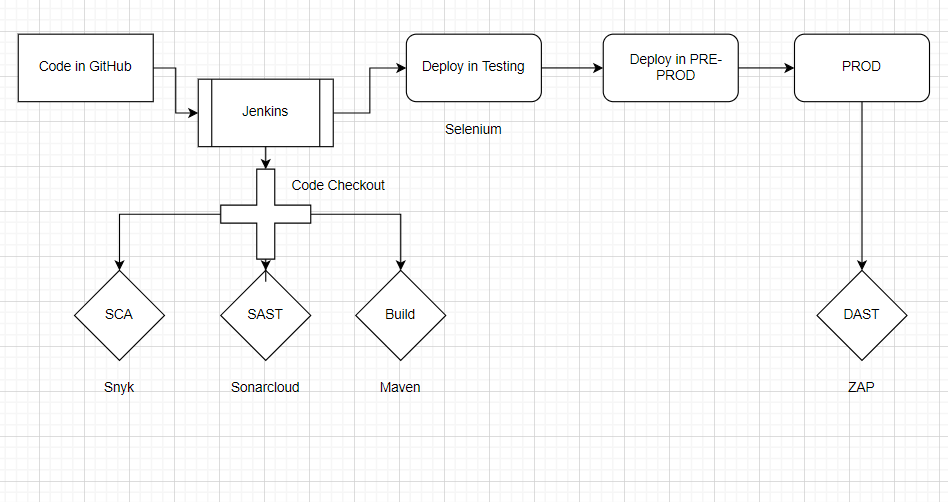
The changes which must apply on environment and management using manual or automation scripting is called Configuration Management. We have tools like Ansible, Puppet, Chef, etc. where we can use for management.

**DevSecOps:**

Code 🡺 Build 🡺 Deploy 🡺 Security Testing

Security Testing: To prevent from any DDOS attack

* SCA: Software composition analysis, 70% Dev + 30% any AI code written by DEV team. Testing done by SNYK, Blackduck, Veracode, OWASP dependency lock.
* SAST: Static application security testing, Scan source code for complexity, code guidelines, copy paste detector (Plagiarisms), programming mistakes, code coverage and generate report. Testing is done by SonarQube, Checkmarks, Microfocus source code analysis tool.
* DAST: Dynamic application security testing, after code in PROD we do DAST. This will scan actual web application for authentication, penetration issues, SQL Injections, broken token checks, unhandled exceptions. Testing is done by ZAP, web inspect, Veracode DAST.



**Workflow of DEVSECOPS**

**GitOps:**

When a Dev write the code and push on GitHub so CI/CD pipeline will take the code as when we pushed the code triggered the pipeline and then the is deployed according the devops pipeline. But this automated thing is called the GITOPS.

Dev🡺 Code🡺 git🡺 CICD🡺 GITOPS REPO🡺 Check (Polling)🡺 test/deployment

**Continuous Version Control:**

The history of updating the configuration which is managed by some kind of revision continuously can be called as continuous version control. It has ability to revert the changes you have made using that revision number. As we are changing on same file, we will have the history of the changes by whom last file got change/modification done by whom.

**Types of Version Control System:**

* Local version control system: In the system we are maintaining the files on local machines itself. So, there are some disadvantages as the files which are saved in our local machine (Laptop) cannot be used from other peoples and if the local machine crash happens all the data we will lose.
* Central version control system: In SVN we used https version of files. So, when we want to update, we require internet for accessing the links/files. If we did modification after committing the changes, we can see the revision and other will see after updating their files. If https links is deleted from the internet, we will lose all the data.
* Distributed version control system: Basic formula for distributed version control system is using Local version control system and Central version control system at once.

**GIT:** Global Information Tracking Tool/engine