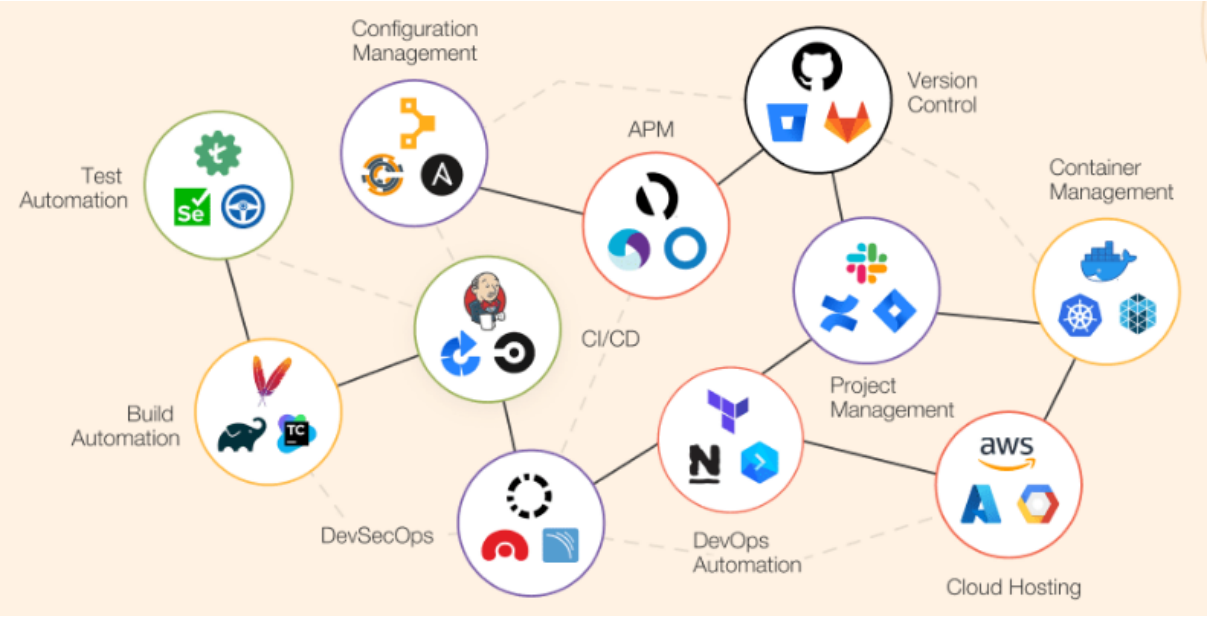
Build Tools

What are Build Tools?

- **Definition**: Build tools are software utilities that automate the tasks of compiling, linking, and packaging source code into executable applications or libraries.
- **Why are Build Tools Important in CI/CD?**:
  - **Automation**: Helps automate repetitive tasks such as compiling code, running tests, packaging binaries, and even deploying applications.
  - **Consistency**: Ensures that every build process (e.g., dev, test, prod) is identical, minimising human error.
  - **Efficiency**: Speed up development by automating builds whenever code is pushed or merged into a repository.

Popular Build Tools in CI/CD



- **Maven**:

    - **Language**: Primarily used for Java.
    - **Features**: Dependency management, project structure standardization, and automatic builds.
    - **How it's Used**: Widely used in Spring Boot projects for Java applications.
- **Gradle**:
    - **Language**: Supports multiple languages including Java, Kotlin, and Groovy.
    - **Features**: Highly customizable and faster than Maven due to incremental builds and caching.
    - **How it's Used**: Preferred for modern Java-based CI/CD pipelines, supports both Android and Java applications.
- `npm` (Node Package Manager):
    - **Language**: JavaScript/Node.js.
    - **Features**: Dependency management and building for JavaScript applications.
    - **How it's Used**: Builds web-based front-end or back-end applications in a CI/CD pipeline.

The Role of Build Tools in CI/CD Pipelines

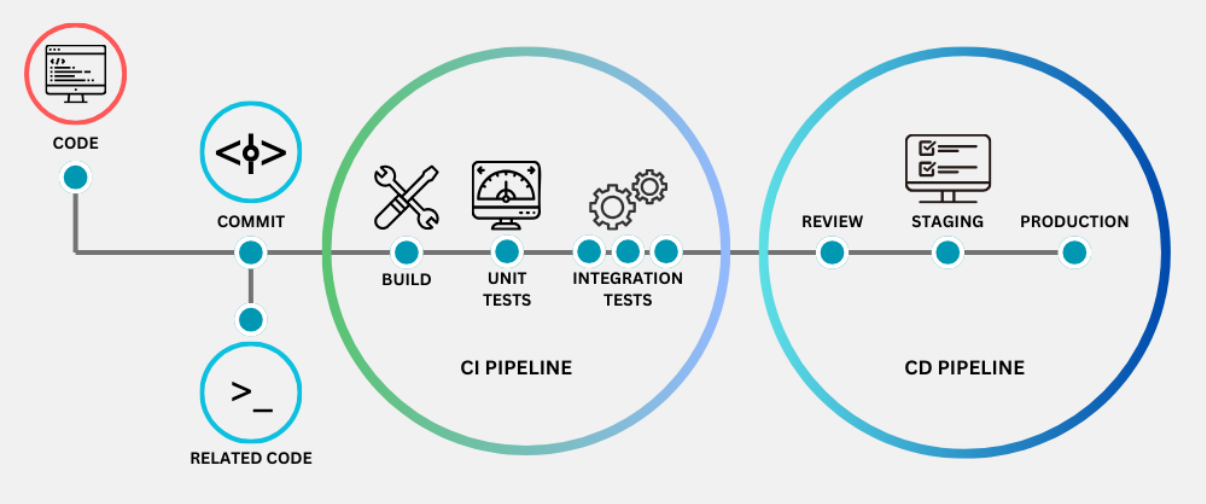
- **Integration**: When changes are pushed to the repository, the CI tool (e.g., GitHub Actions) triggers the build tool to compile and package the application.
- **Build Automation**:
    - Automatically handles downloading dependencies (e.g., Maven or Gradle), compiling the code, and running tests.
    - Ensures that the same version of the application is built every time.
- **Testing**: Many build tools, such as Maven, integrate with testing frameworks (JUnit, Selenium) to run automated tests after the build.
- **Deployment**: The packaged application can be deployed to a server, containerised (e.g., Docker), or distributed using CD tools.

Example Build Tool Workflow in a CI/CD Pipeline

1. **Code Push**: Developer pushes new code to the GitHub repository.
2. **CI Tool Trigger**: GitHub Actions detects the change and triggers the pipeline.
3. **Dependency Resolution**: Build tool (e.g., Maven) fetches dependencies from repositories.
4. **Compile & Build**: Build tool compiles and packages the code into executable binaries (JAR, WAR).
5. **Testing**: Run unit and integration tests automatically.
6. **Package & Deploy**: Build tool creates the package, and the CI/CD pipeline deploys it to staging or production.

Commonly Used Build Tools for Various Languages

- **Java**: Maven, Gradle
- **JavaScript**: `npm`, yarn
- **Python**: PyBuilder, `tox`
- **C#/.NET**: MSBuild
- **Ruby**: Rake



Build Tools and Continuous Integration

- **Continuous Integration (CI)**: Build tools automate the process of building and testing code with each integration to the repository.
    - Ensure new changes don't break existing code by running automated tests as part of the build process.
    - Fail-fast behaviour: If tests or builds fail, the developer is notified immediately.

Build Tools and Continuous Deployment

- **Continuous Deployment (CD)**: After successful build and testing, build tools package the code, ready for deployment.
    - **Artifact Creation**: Builds create deployable artifacts (e.g., JAR, WAR, or Docker images).
    - **Automated Deployment**: The pipeline can then deploy the artifact to a server, cloud, or container.
