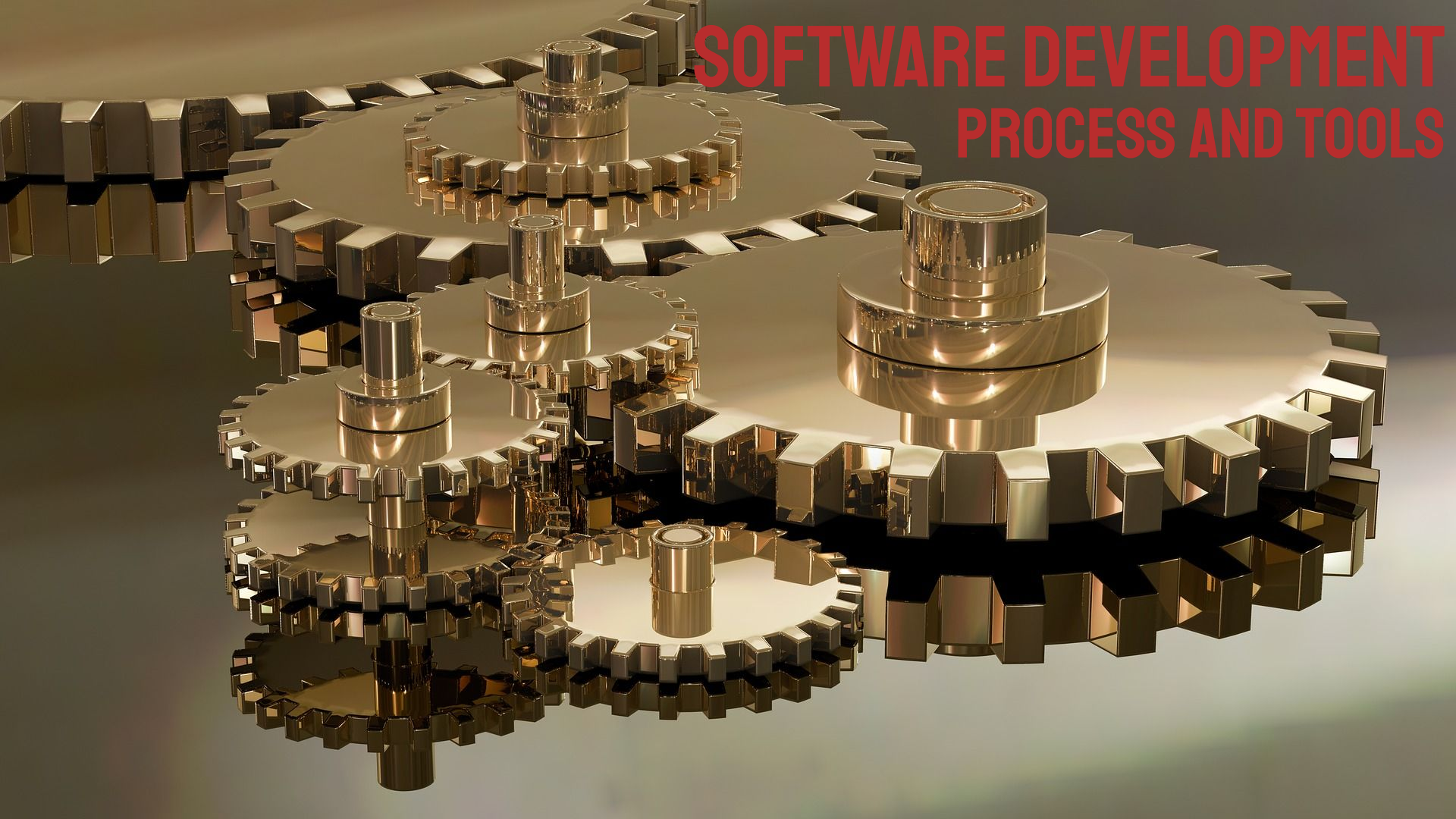

SOFTWARE DEVELOPMENT
PROCESS AND TOOLS

WHAT WE'LL LEARN

SOFTWARE DEVELOPMENT PROCESS

CONTINUOUS INTEGRATION AND DELIVERY

SOURCE CONTROL SYSTEMS

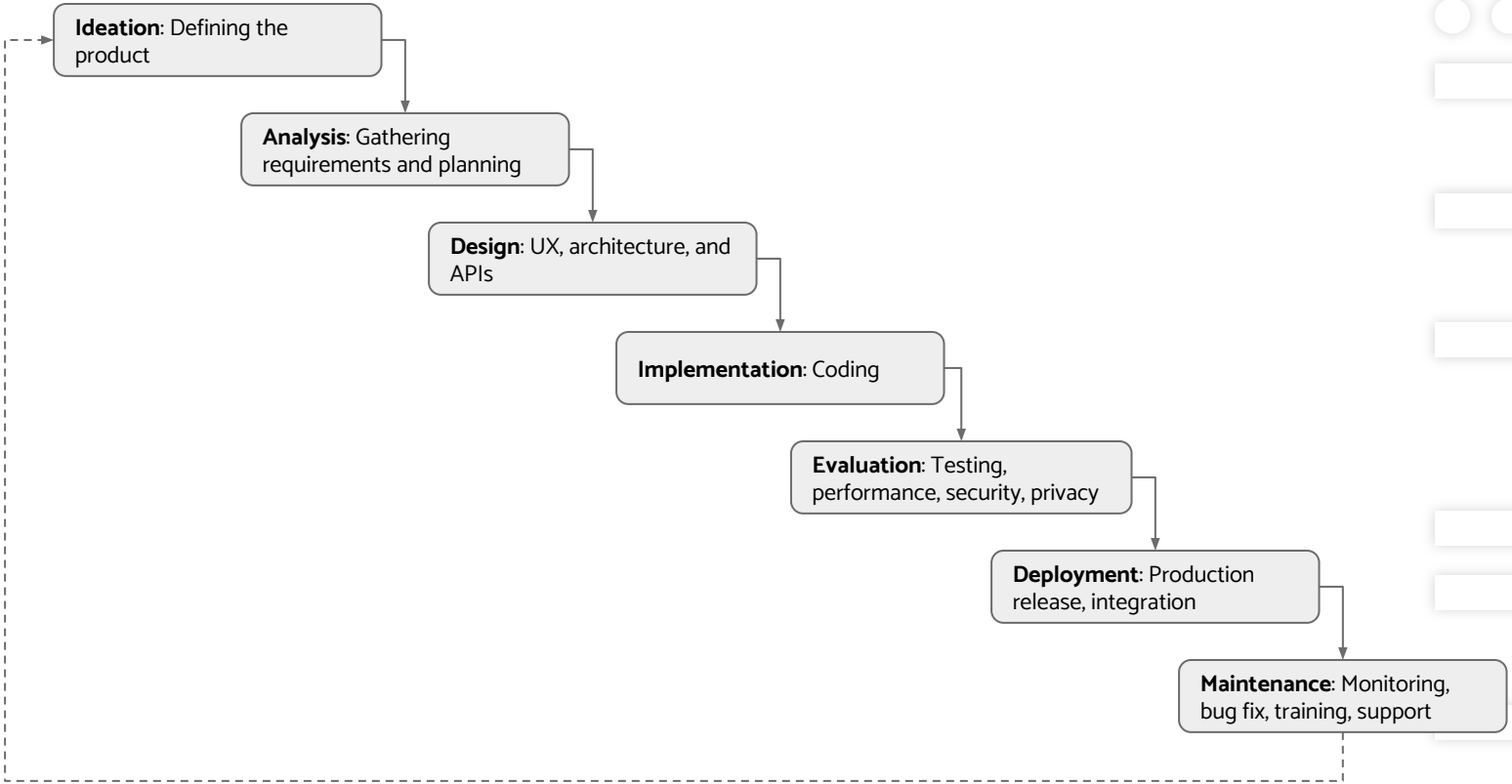
SOFTWARE IS BEYOND CODING

Developing software is a long journey; coding is only part of it. As a developer, you will be involved in many activities to build software, from ideation to design, testing, launch. (You will follow these in your final project.)

These activities resemble those in other areas. For example, take building a house, you first imagine the house at an abstract level, maybe draw it on paper. Then you talk to an architect and describe what kind of house you want, show the drawing, explain how many rooms, where each room should be, the windows, doors etc. The architect designs the final form of the house in a detailed plan. Next, the builders take this detailed plan and actually build the house; meanwhile certain tests (eg. structural tests, leaking tests, electrical circuitry tests etc. are performed). The house is then delivered for your use. Even after that, you still need to continue maintaining/repairing parts of it. Sometimes, you go over these activities again, for example if you want to build an extension on the house.

The software development process has similar activities. Let's first go over these activities and see the different approach that apply to these activities in various ways.

PHASES OF DEVELOPMENT



IDEATION: DEFINING THE PRODUCT

The first phase of development is all about defining the product. This is done for example by marketing analysis or based on needs of an organization. This phase usually ends with the high level requirements document.

For example:

- Build a mobile software application to calculate the taxes and submit a tax report.
- Build a web app to allow customers to search for and rent villas in coastal cities.

ANALYSIS: GATHERING REQUIREMENTS AND PLANNING

This is the phase where you start to understand the product in more detail and think about allocating resources to build the product.

Various types of discussions are held and requirements gathered and documented:

- **Functionality:** What tasks the software needs to perform
- **Environment**: What type of platforms it should run on (web, Android, iOS, watch, etc), will it be distributed or work on local clients.
- **Performance:** How fast the software should run, how scalable it should be (ie, how many users or operations it should respond to in a given time-frame)
- **Security, privacy:** What security is required, user data protection requirements, government regulations etc. etc.
- What are the predicted costs

Resources include people, engineers, UI designers, program managers and so on, to implement the rest of the development process, as well as a budget to compensate the costs (eg. location, software and hardware to buy etc)

DESIGN: UX, ARCHITECTURE, APIS

In this phase, two kinds of design happen (documented in great detail):

- **Product manager and UX designers** produce usage scenarios and screens are designed. End result is a very detailed documentation of every page, screen, message, with all fonts, colors, text, and behavior.
- **Software architect and engineers** produce design of software architecture: building blocks/components of the software (in terms of modules, libraries, servers) and how they interact with each other and external systems (ie. APIs).

The design should be reviewed and approved by multiple authorities before the implementation proceeds. This is critical to ensure there is no risk in security, privacy, and legal aspects. These are difficult to address during the implementation.

Designing the software early is crucial: it helps to discover potential problems, divide the work in parallel and sequential tasks for implementation, and provide visibility to the stakeholders that needs to be kept uptodate about the project.

IMPLEMENTATION: CODING

Implementation is the phase where software is written. Developers take the design and split it into coding tasks for building individual components of the software. This is often done in parallel by multiple developers working on different parts of the software. Designing individual components and their interfaces to each other and the external world upfront helps to make sure the end-product results in a coherent and cohesive system.

Implementation may involve more than pure coding:

- Setting up infrastructure; servers, configurations.
- Creating databases and other storage systems.
- Creating maintenance tools and documentation.

EVALUATION: TESTING, PERFORMANCE, SECURITY, PRIVACY

Once the software is built and components are integrated together, the software should be evaluated against the requirements collected in the analysis phase. This often done on a staging server that mimics the requirements of a production server.

- **Testing:** End-user and automated tests applied on the software to make sure it behaves as intended, it computes the correct results, responds to user actions correctly etc. etc.
- **Performance:** CPU, memory, and network usage of the software is measured to make sure it does fits into the original resource estimates and allocation. Battery usage is measured for mobile apps. Load tests are applied to make sure the software responds to requests in a timely fashion.
- **Security and privacy:** The final implementation subjected to security and privacy tests (eg. penetration tests) to make sure no sensitive data is leaked.

DEPLOYMENT: PRODUCTION RELEASE, INTEGRATION

In this phase, the software is “deployed” on the production environment.

- For server software, the code is placed on the backend/cloud servers and run from there.
- For desktop/mobile apps, the software is packaged and placed in the app store, for users to download.
- For libraries/SDKs, the software is packaged and distributed to the customers (for example, via a website or FTP server).

During deployment, the usage of the software is monitored for any issues or resource consumption problems that have not yet been identified, and patches applied as necessary.

MAINTENANCE: MONITORING, BUG FIX, TRAINING, SUPPORT

The journey does not end after the software is deployed. The production usage of the software needs to be monitored for any issues, i.e. bugs, security vulnerabilities in itself or third-party libraries it may be using. Memory leaks and edge case scenarios that have not been accounted for in the design.

It's critical that users are not blocked in any way from using the software effectively. The organization providing the service provides a support channel (eg. forums, call centers, web-chats, email etc. etc) to answer user questions and if needed direct the issues to the developer team to fix any bugs or improve the UX design.

WATERFALL VS AGILE: DIFFERENT APPROACHES TO THE DEVELOPMENT PROCESS

Based on the organization or the nature of the software, these phases are applied by various methodologies. A couple of these methodologies are 'Waterfall` and 'Agile.

Waterfall

In the waterfall approach, the entire software goes through a layered serious of design and engineering steps. Mission-critical systems, flight control systems, nuclear reactor control system and the like, where very careful design and thorough testing is required will perform stringent software engineering processes for each stage. Each stage may take months, or even years.

Agile

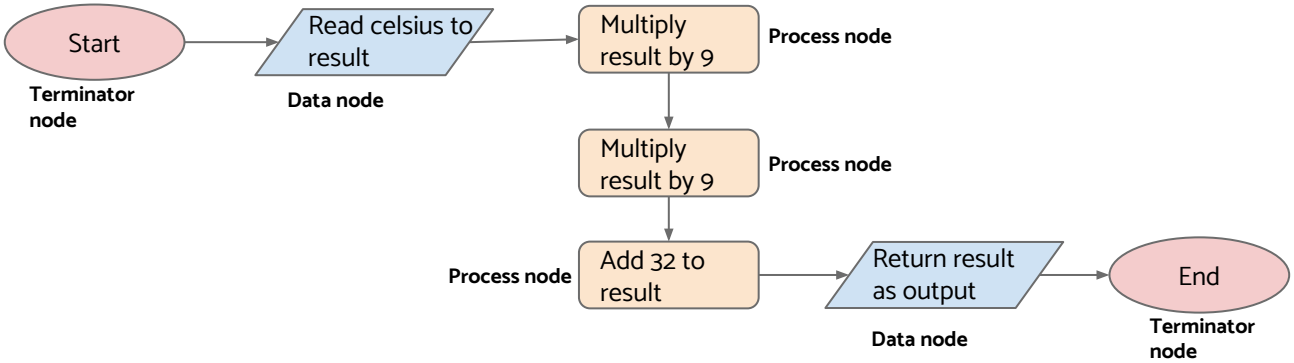
Using the agile approach, software is developed progressively; the phases are repeated in multiple cycles for new components or features of the software. For example, analysis-to-deployment is performed every 2 to 6 weeks for new features. At the end of each cycle, a functional version of the software is served to production users and a new cycle begins for another version.

Scrum is a framework to apply the agile approach. See https://www.scrum.org/resources/what-is-scrum for details.

TOOLS FOR DESIGN AND CODING

FLOWCHARTS: FOR DESIGNING A SEQUENCE OF COMPUTATIONS

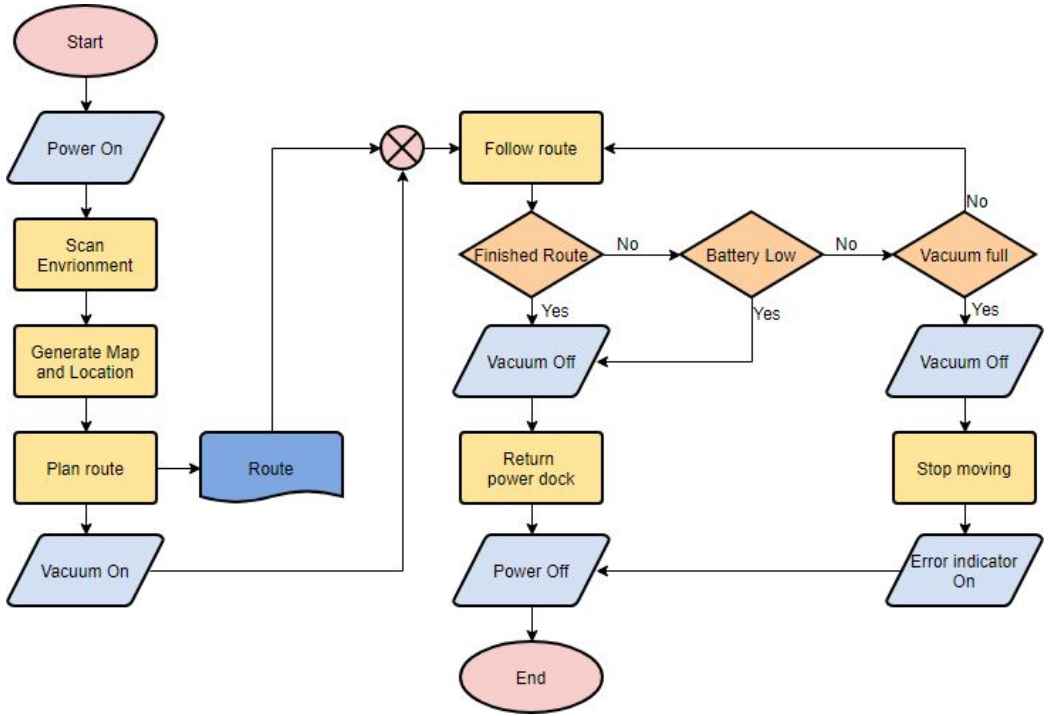
Flowchart are a graphical representation of a processes. A process can be an algorithm or any other sequence of events and measurements. Flowcharts are a good way of organizing the control and data flow at different levels of detail. For example, you can describe a logistics process or a computer program in a flowchart. The following example describes how we convert celsius to fahrenheit. You can find more examples in the links below.



Some pointers to flowchart tutorials:
https://online.visual-paradigm.com/diagrams/tutorials/flowchart-tutorial/
https://creately.com/blog/diagrams/flowchart-guide-flowchart-tutorial/amp/

FLOWCHARTS: DECISIONS AND LOOPS

The following example illustrates how an automated vacuum cleaner operates. This can be used as a starting point to design and develop the software for the vacuum cleaner. It is more accurate than a textual description, showing all decision points, data inputs, outputs etc.



IMPLEMENTATION: INTEGRATED DEVELOPMENT ENVIRONMENT - IDE

An IDE is a software application for developing software.

An IDE provides a collection of tools that provide a multitude of functionality to aid in the software development process.:

- Write code: Smart editor with syntax highlighting, autocomplete etc.
  - Tools for organizing and refactoring code.
- Compile, execute, and debug: Step through code line by line and observe variable' values
- Write and run tests
- Wide collection of related tools:
  - Source control to create branches, commits, push and pull code
  - Database tools to connect to and query/modify databases

For an overview of a popular IDE see the following Quick intro to PyCharm, an IDE for Python and other languages.

https://www.jetbrains.com/help/pycharm/quick-start-guide.html

EVALUATION: TESTING

Testing is a critical part of development. Software is written by humans, it is subject to mistakes. A test is a piece of code or process to verify that the program works correctly. The following is a brief summary of the common types of testing and their relationship.

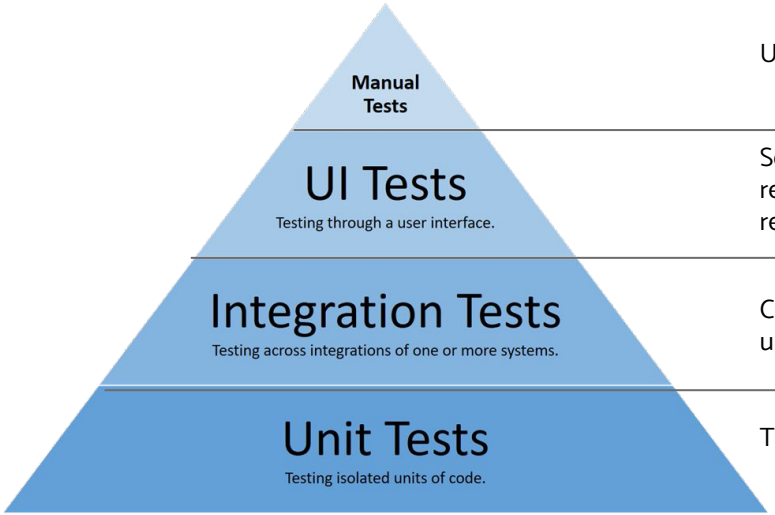
- A **unit test** is a piece of code that checks a function or class works correctly. Code files are usually accompanied by files containing their tests in the same directory. Everytime you update the code, you run the test to make sure the code still works correctly.

- An **integration test** invokes larger pieces of software, including multiple components, and verifies that these components works together correctly. For example, component A calls component B in the right way and component B returns the right form of response to component A.

- A **UI test** interacts with the user interface simulating the actions of a human user, ensuring the expected responses are given by the system.

- The first three categories above are automated. On the other hand, not everything can be automated. A **manual test** is carried out by a human by using the program and going over some usage scenarios and making sure that the program responds correctly. Manual tests are also used in destructive testing, to see if a user can break the interface.

EVALUATION: TESTING PYRAMID



Example: Bank app

Use the app to send money to someone else.

Send commands to app to fill in text boxes for amount and receiver account id, click at “Send” button; check success result.

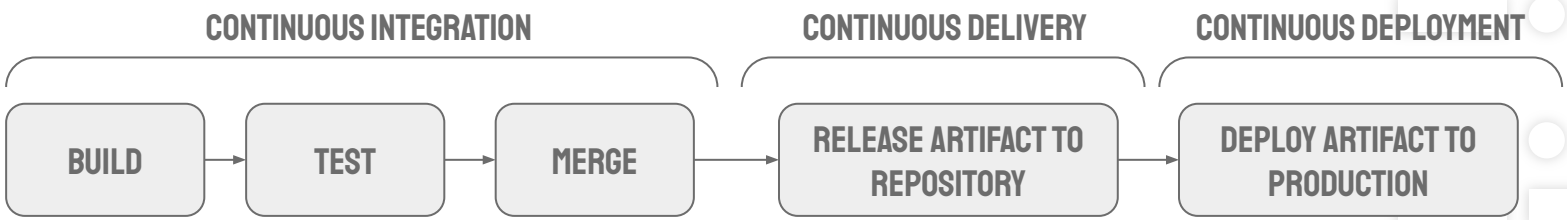
Check if (A) sending money to another bank and (B) updating database for account work together.

Test code to withdraw money from account.

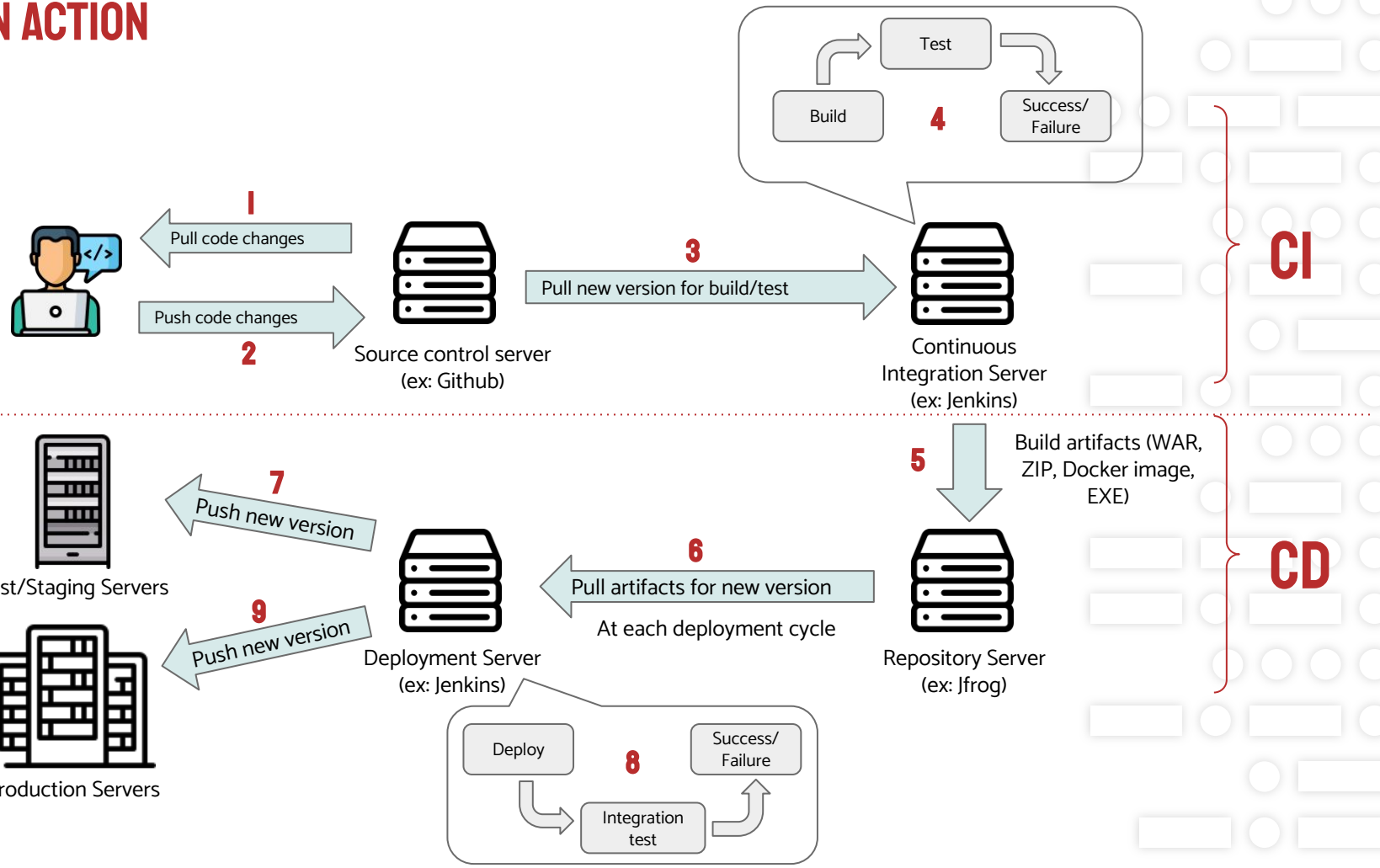
CI/CD: CONTINUOUS INTEGRATION AND DELIVERY

CI/CD: CONTINUOUS INTEGRATION/DELIVERY/DEPLOYMENT

CI/CD consists of processes and tools to automate the cycle of code/build/test/deploy. The tools include servers and local programs communicating with these servers. They integrate into daily life of developers and devops (developer-operations) engineers.



CI/CD IN ACTION



SOURCE VERSION CONTROL

Why do we need source control?

- Maintain a single source of truth for many developers (working in parallel on different parts of the software).
- Facilitate collaboration and accelerates release velocity.

Benefits:

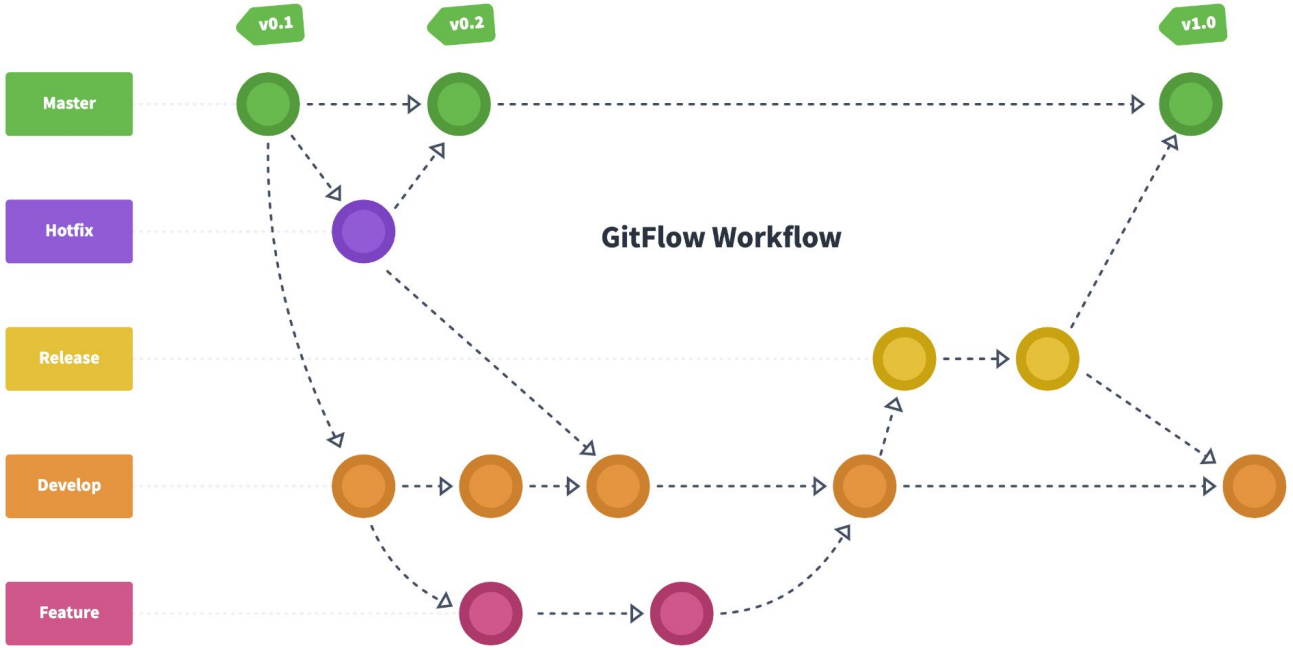
- Multiple developers can work on the same codebase.
- Developers can commit and merge code without conflicts.
- Developers can edit shared code without overwriting each other's work.

GIT: SOURCE CONTROL PROTOCOL

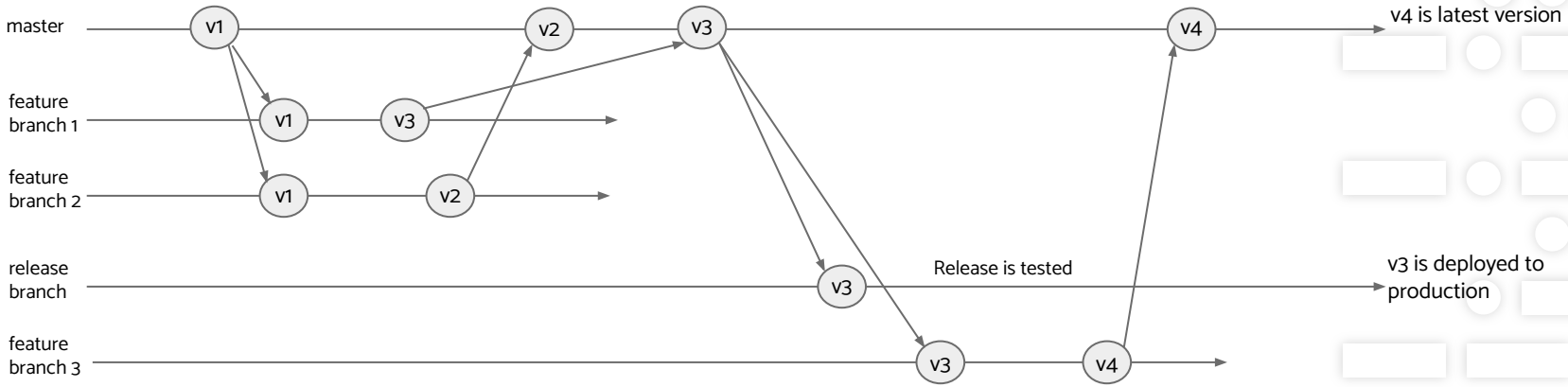
GIT is a protocol for interacting with source code repositories. Github and Bitbucket are known implementors.

You can use GIT tool to create or download a repository, submit new changes and download others' changes.

GIT: TYPES OF BRANCHES



GIT: WORKFLOW



GIT: CODELAB

Go to https://github.com/ and create a new account.

Checkout code: git clone ....

Create new branch: git branch <your_name>

Edit README.md file and see diffs: git diff

Add new file <your_name.md> and see diffs: git add <filename> and git diff

Commit your changes: git commit “Commit message”

Push your changes: git push

Create pull request from Github page and get it reviewed

Merge others’ changes: git merge origin/master