**Project Background:**

1. **RESTful API**

***API:***

Computer applications use APIs (Application Programming Interface) to communicate with web servers. Generally speaking, an API exposes a set of data and functions to ease interactions between computer applications and allow them to exchange data and functions.

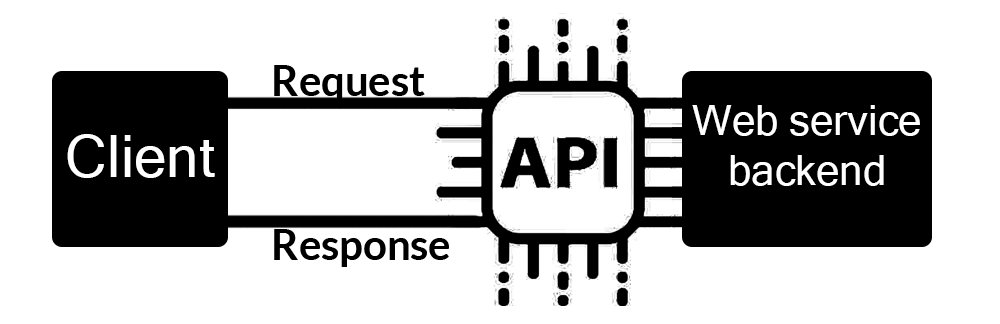


Figure (3-1) – Application Programming Interface

As shown in figure (3-1) a web API is the face of web services directly interacting with clients (receiving their inputs and responding back with requested data or functions).

***REST:***

REST (Representational State Transfer) is basically a technical description of how the world wide web works, it shows how the web can achieve its great scale.

***REST API:***

A REST API (Application Programming Interface) is a type of server that enables a client, whether it’s a user-operated or automated to access resources that model a system’s data and functions.

A REST architecture or style is mostly applied to the design of APIs for modern web services, and a web API conforming to the REST API architecture is called a “RESTful” API.

***REST API Requests:***

A REST API request is a very simple concept because it follow the HTTP (Hypertext Transfer Protocol) which is a protocol used for information systems.

**-Every REST API is made up of the following:**

1. **The Endpoint (or route):**

is the URI (Uniform Resource Identifier) you request for

You specify a resource in the URI of your request for example: http://mypetsore.com/pet/124

The /pet/124 part is a resource (path) located in *mypetstore.com* server also called root-endpoint.

Any colons (:) on the path denotes a variable for example http://mypetstore.com/users/:username/pets here you replace :username with the actual username of the user.

The final part of an endpoint is query parameters they’re technically not a part of the REST architecture but you’ll see them a lot example is: http://mypetstore.com/users/:username/pets/pet?name=${name}

1. **The method:**

The method is the type of request you send to the server the following are the types of request methods:

* GET: the request is used to get a resource from a server (read operation).
* POST: the request is used to create a new resource on the server (Create operation).
* PUT and PATCH: are used to update an entity on the server (Update operation).
* DELETE: the request is used to delete a resource from a server (Delete operation).

1. **The headers:**

The headers are used to provide information for both the client and the server, it can be used for many purposes, such as authentication and providing information about the body content.

Headers are a property-value pairs that are separated by colons for example:

"Content-Type: application/json".

This header tells the server to expect a data type with a JSON format.

1. **The data (or “the body”):**

The data or body contains the information or data that you want to send to the server it’s only used with POST, PUT, PATCH or DELETE requests.

1. **CI/CD (Continuous integration/Continuous deployment):**
2. **Continuous Integration:**

Is a software development practice where members of teams tend to integrate their work frequently, usually each person integrates at least daily and each integration is verified by automated testing to detect errors as quick as possible – leading to multiple integrations per day with lesser defects and allows teams to develop software more quickly.

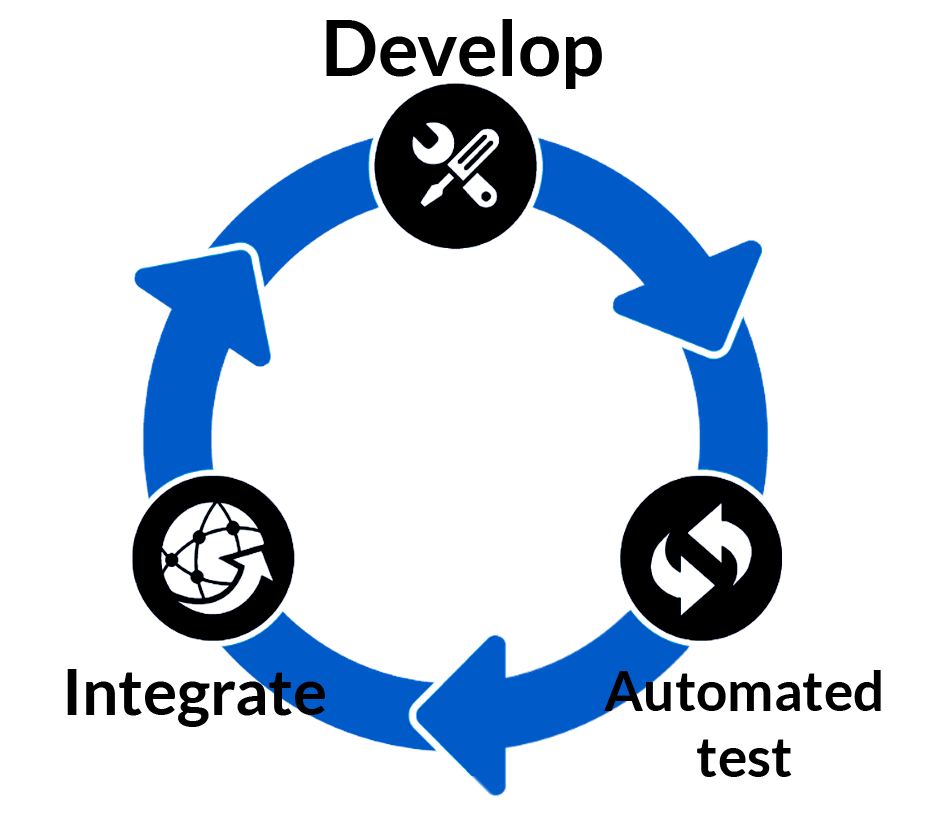


Figure (3-2) – Continuous integration

1. **Continuous Deployment:**

Jez Humble defines continuous delivery as: "Continuous Delivery is the ability to get changes of all types—including new features, configuration changes, bug fixes, and experiments—into production, or into the hands of users, safely and quickly in a sustainable way."

Continuous deployment build on continuous integration and takes it a step further, the continuous integration phase provides the first feedback to developers it checks out the code from the repository ,compiles it ,runs automated tests and verifies its quality if anything fails the execution is stopped and the developer should fix the integration build.

To understand continuous deployment better, let’s take this scenario:

You’re responsible for an email-client product, let’s say the user comes to you with a new requirement they want to sort emails by name you decide that the development will take one week, so when can the user expect the new feature?

Usually after the development is done, you’ll hand over the new feature to the QA team (Quality Assurance team) and then to the operations team which takes lots of additional time therefore even though the development is done the user will receive it after days or possibly months after the development.

The continuous Deployment approach addresses this issue by automating manual tasks so the user can receive the new feature as soon as it’s implemented.

**Benefits of continuous deployment:**

**a- Fast Delivery:** Time to market is significantly reduced as customers can use the product or the new features as soon as they’re developed.

**b- Fast Feedback:** If the developer created a bug in the code, and then it went into production the same day it’ll take significantly less time to fix.

**c- Low risk releases:** Releasing on daily basis means the process gets more repeatable thus a lot safer.

1. **Team Collaboration – DevOps:**

DevOps is a term that is usually associated with CI/CD (Continuous Integration/Continuous Delivery) practices, the term DevOps was introduced in 2007 and it represents the combination of Development (Dev) and Operations (Ops).

DevOps is a set of practices that reduces the barrier between developers who want to innovate and deliver as fast as possible, and Operations who want to guarantee the quality and stability of the product.

DevOps is an extension of agile processes (scrum, extreme programming, and so on) which reduce the delivery times but are often hindered and slowed because of the non-inclusion of Ops in the same teams.

In short DevOps combines Software Development and Information Technology Operations to shorten the development life-cycle and provide continuous delivery with high software quality and stability.

**There are several main key elements to facilitate the collaboration between developers and operations teams:**

a- More frequent application deployments with continuous integration and delivery.

b-The implementation and automation of integration test.

c- the implementation of a way to collect feedback from customers.

d- A way of monitoring the applications progress and infrastructure.

**The DevOps approach is based on three axes:**

***a- Collaboration:***

Collaboration is the key essence of DevOps since teams are no longer separated (one team of developer, another for operations, and one team for testers, and so on) these teams are brought together by a multidisciplinary team that have the same objective (delivering products and features as fast as possible).

***b- Processes:***

To expect fast development the team must follow development processes from agile methodologies to allow for better functionality and faster feedback. These processes should be integrated into the development workflow with **Continuous integration** and the deployment workflow with **Continuous delivery.**

**DevOps process phases:**

* The planning and prioritization of functionalities.
* Development.
* Continuous integration and delivery.
* Continuous deployment.
* Continuous monitoring.

***c- Tools:***

The choice of tools is very important in DevOps, when team were separated into Developers team and Operations team each team had their specific set of tools (Deployment and testing tools for developers, and infrastructure tools for operations team).

With DevOps the tools used must be usable and exploitable by all members of the DevOps team.

Developers should integrate with monitoring tools used by Ops to detect performance and security issues.

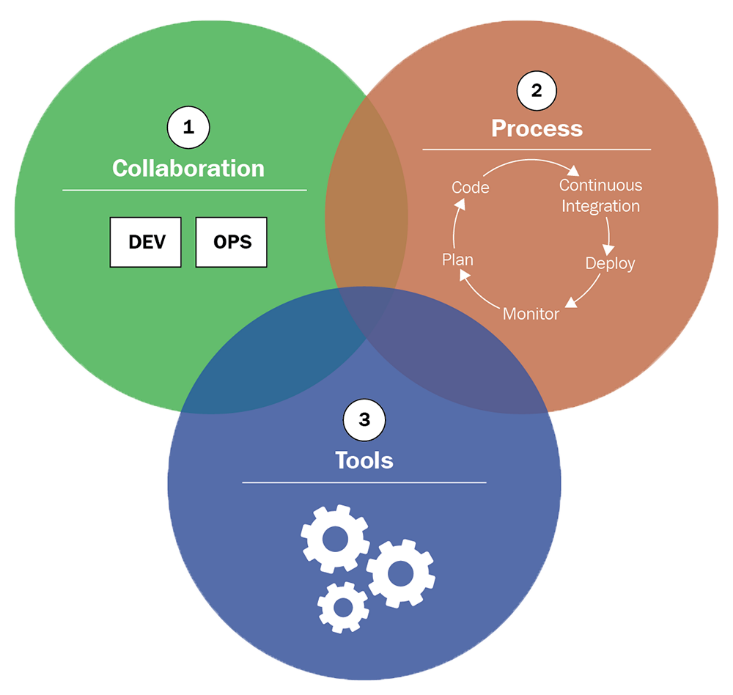


Figure (3-3) – The DevOps culture

1. **Software testing and test automation:**

* **Software Testing:**

Is defined as an activity to check if the actual result matches the expected result and ensuring the software system is **defects free**.

**Types of software testing:**

**a-Functional testing:** Is a type of software testing whereby the system is tested against its functional requirements, functions are tested by feeding them input and examining the output.

**b-Non-functional testing:** Is a type of software testing to check the non-functional aspects of a software system (such as performance, usability, reliability, etc.).

**c-Maintenance testing:** is a type of software testing that is done to either identify software problems, or to confirm that maintenance measures have been effective.

* **Test Automation:**

Is using a software other than the software being tested to control the execution of tests and the comparisons of the test’s outcomes.

**The Benefits of automation testing:**

* **Speed:** Automating tests is significantly faster than manual testing.
* **Efficiency:** Having a test automation tool will increase efficiency by reducing the time it takes for you to run your tests, thus giving you more time for test planning and thinking up new tests.
* **Accuracy and Precision:** Human attention span is limited and after running a few tests it can be lost, a test tool will run the tests and check the results perfectly every time.
* **Resource Reduction:** Reduces the number of people needed to run the tests and focusing all the resources into test planning.
* **Simulation and Emulation:** Test tools are sometimes used to replace hardware or software that would normally be the interface to your product. The test tool can simulate the interface (example: **GUI testing**).

* **Relentlessness:** Since test tools never tire up, they’ll keep going with the same constant energy and attention span.

1. **Version Control:**

AVCS or a (Version Control System) is a system that is capable of recording changes made to a file or chunks of files over a specific period of time in a way that it allows to go back in time and recall a specific version of that file.

A more formal explanation would be that “a version control system is a software package that when initiated will monitor your files for changes and allow you to tag the changes at different levels so that you can revisit those tagged stages whenever needed”.

**Why do we need version control?**

* Enables you to maintain multiple versions of the same file under the same name thus avoiding undesirable clustering of files with small differences.
* Works as a failsafe plan meaning that if your files got deleted or lost in any way you can always retrieve them back.
* Makes collaboration lots easier since multiple users can work on the same file and preview each other’s edits.

By using version control you have the power to play with the flow of changes happening to your files, whenever you write a considerable amount of changes to the existing content of the file you can mark those changes with mark that enables you to revisit them later, and in case something goes wrong you can revert back to those changes.

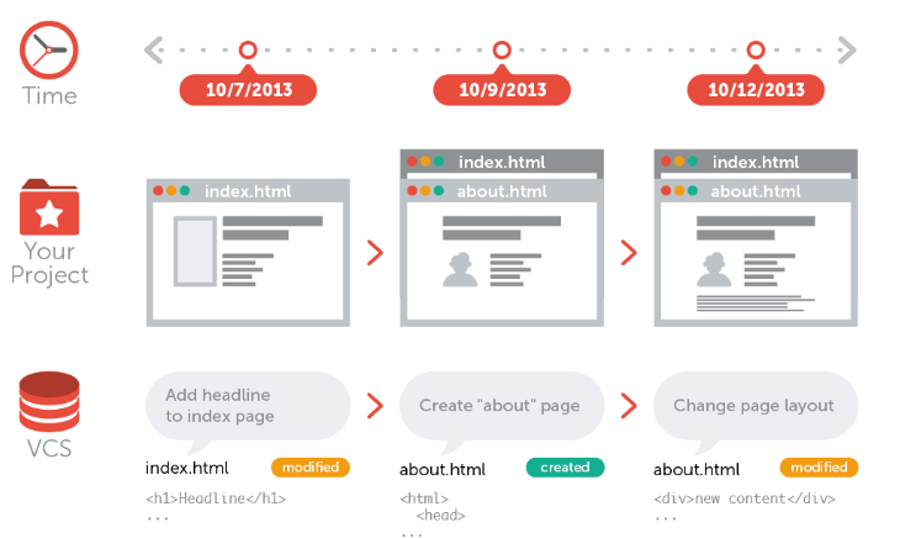


Figure (3-4) – Version Control System

Figure (3-4) shows the flow of a web page design while using a version control system as you can observe the flow goes from the left to the right on different periods of time and you can always roll back to a previous version of the project or file.

**Types of version control system:**

There are three types of version control available that are classified on their mode of operation:

1. **Local version control system:**

This type basically works by keeping patch sets (the difference between the files content) using a special format in the version tracker that is then stored into the computer’s local hard drive.

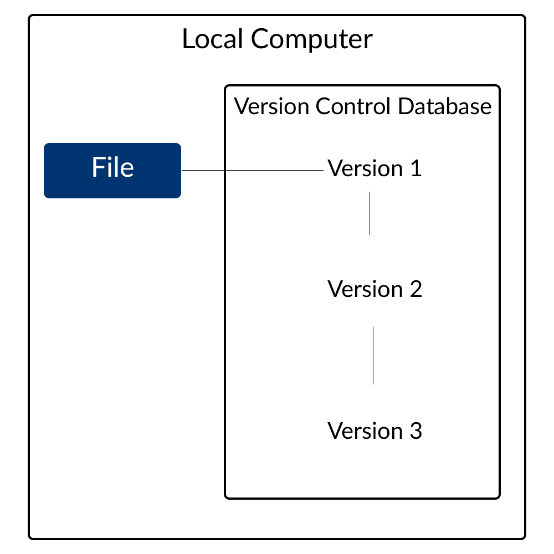


Figure (3-5) – Local Version Control System

1. **Centralized version control system:**

As it appears the local version control system doesn’t exactly satisfy in collaborative workspaces (people were not able to work collaboratively on the same project) and it was very limiting.

So, it was solved by keeping the files in a server that everybody has the access to hence was the birth of centralized version control systems that we’re using till now.

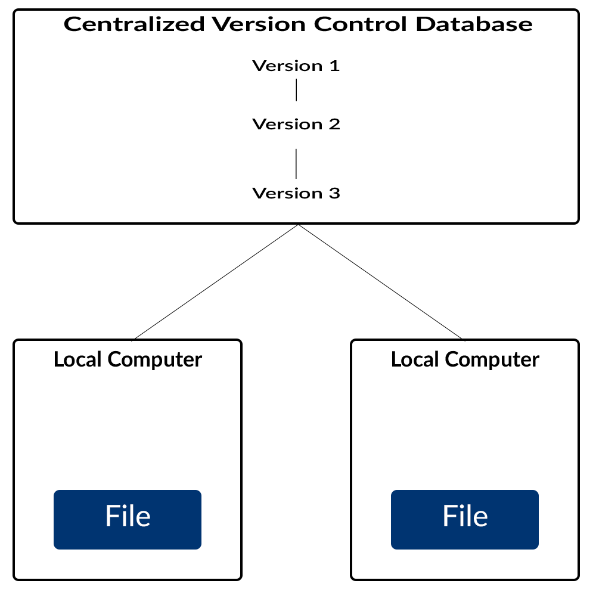


Figure (3-6) – Centralized Version Control System

1. **Distributed version control system:**

Using centralized version control systems might’ve solved the collaboration issue but it also has the disadvantage of losing your previous versions in a scenario if the server got corrupt or your files in it got deleted for any reason.

That’s where distributed version control systems come in handy by combining both local and centralized systems where you keep a copy of the project and its version tracker (containing the difference between files in project) both on the server and locally on your computer.

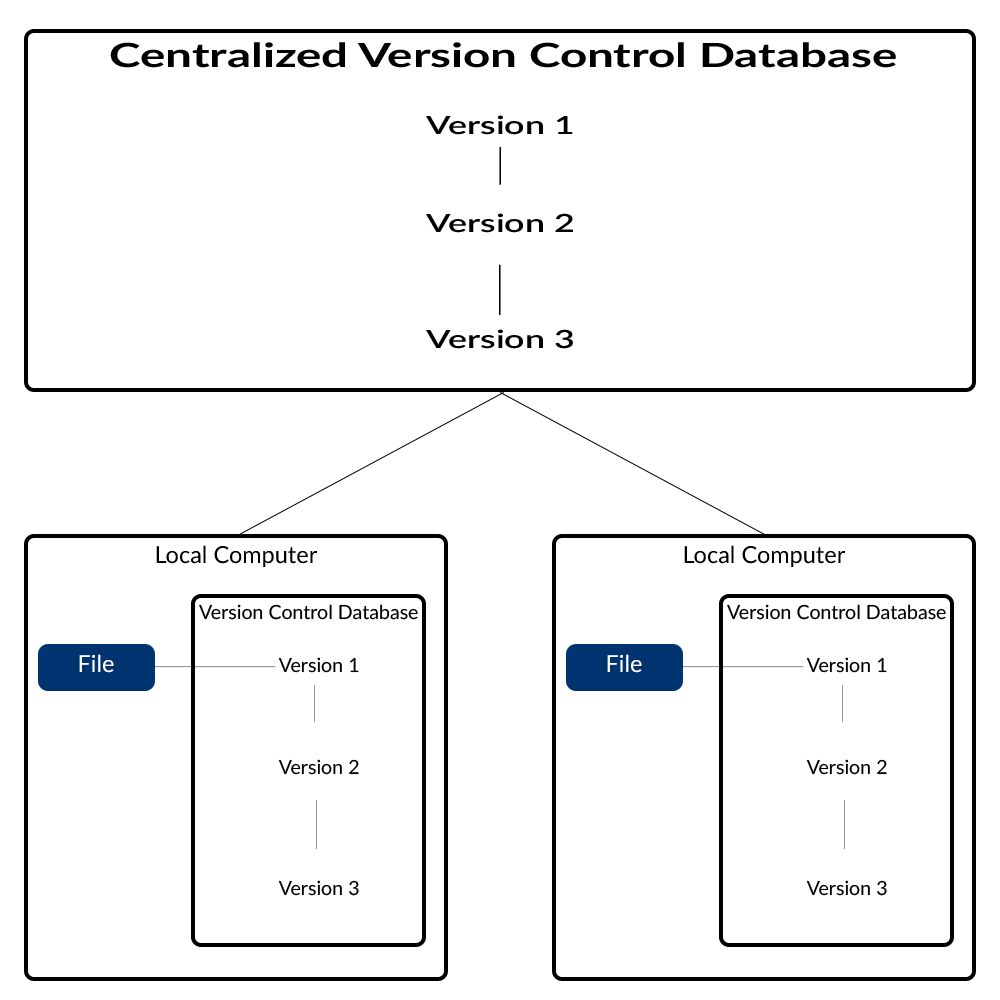


Figure (3-7) – Distributed Version Control System

**References:**

* REST API: [*https://www.smashingmagazine.com/2018/01/understanding-using-rest-api/*](https://www.smashingmagazine.com/2018/01/understanding-using-rest-api/) , **REST API Design rule book** by **Mark Massé.**
* CI/CD: **Continuous delivery with docker and Jenkins** by **Rafał Leszko**
* DevOps: **Learning DevOps** by **Mikael Krief.**
* Software Testing And Test Automation: **Software Testing, Second Edition** by **Ron Patton.**
* Version Control: **GIT: Version Control For Everyone** by **Ravishankar Somasundaram.**