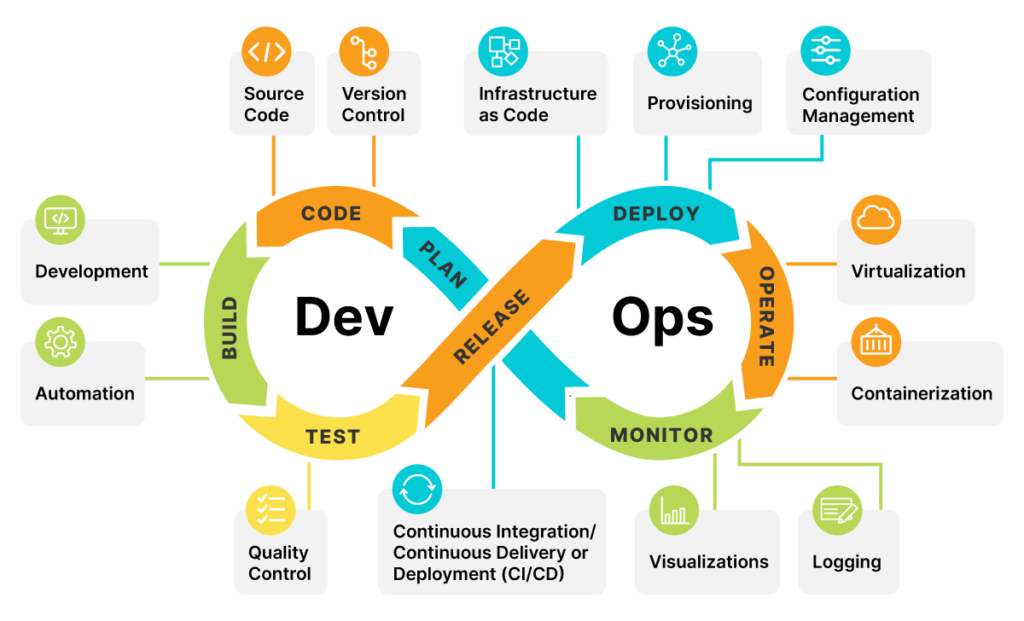
DevOps is a set of practices, tools, and a cultural philosophy that automate and integrate the processes between software development and IT teams.

DevOps combines development and operations to increase the efficiency, speed, and security of software development and delivery compared to traditional processes. A nimbler software development lifecycle results in a competitive advantage for businesses and their customers.

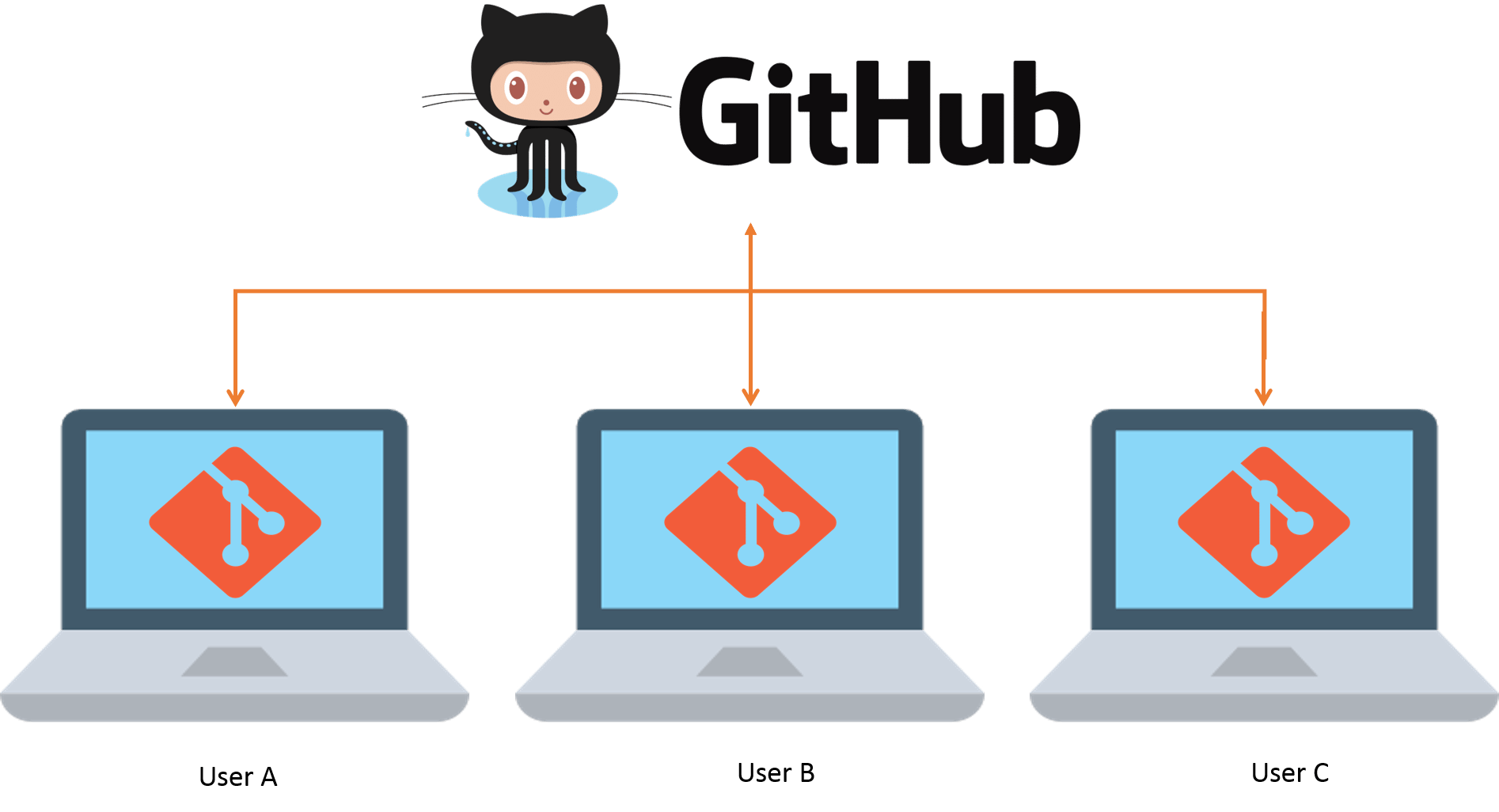


There are multiple stages/divisions at DevOps:

**SCM/VC:**

To deliver a product to customers, after the Business analysis and requirement gathering the actual development will start and in the development process, we may generate multiple files either code base (Python, Java, .net, Go, Java script, Node JS, Shell script, Powershell scripting, etc..) or config files in form of yaml, json, services.

To manage the code with different history versions automatically we can use Source code management (SCM) or version controller. Using this SCM/VC tools we can maintain different versions of each file and retrieve back the any version any time.



Example:

You are a developer in a team with group of contributors working together on different individual tasks, and after some days the client changed the specific requirement. So, you have modified your code as per the new changes. Later the client wants to get back to the requirement due to business needs.

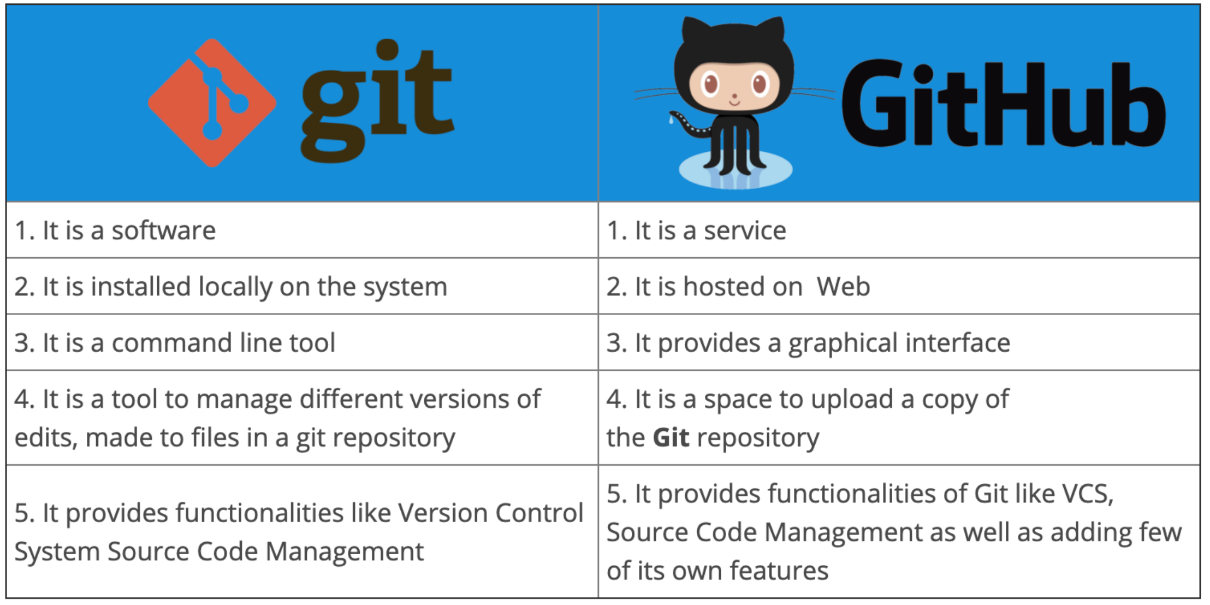
We have modified the code earlier as per the client changes and we don’t have the copy of old version. As we don’t have the old version source file, need to edit back the code. Incase if we have the old version easily get back the requirement without loosing extra effort and time.

And using the SCM/VC not only get back the source code, we can maintain different line of source code to enhance or add extra features, can create the branches.

There are two different types of SCM tools:

1. Centralized
2. Distributed

We can learn on each tool and cover more in-depth concepts when we work on SCM/VC.



**Build tool:**

When we complete the development, we are not going to deliver the code to clients. In the delivery process we need to make a final build, which is completely compressed packaged ready to deploy.



The above exe file is a final product from developers after their code development and build life cycle. The build tools are managing the life cycles or phases like compile, test, package, integration test and the below are the example of difference code base build tools: 

|  |  |
| --- | --- |
| **Source Code** | **Build Tool** |
| Java | Maven, Gradle, and ant. |
| Python | Pip, Pybuild and Maven with Jython plugin |
| .net | MSbuild, zip |
| Java Script | Npm, yarn, webpack |

**Containerization:**

Containerization involves placing a software component and its environment, dependencies, and configuration, into an isolated unit called a container.

In General, without the containerization, the deployment will be like giving the config files, build, and dependency files to the production environment.

Sometimes due to human environments the environment setup and config might be miss, So the application might not work same as in development environment.

With the Containerization concept, all the config and dependencies can be packed as a compact less size package, which is used to directly deploy into production environment without any human mistakes

. 

The below are the different containerization tools:

1. Docker
2. Kubernetes
3. Vagrant
4. Containerd
5. Rkt
6. Podman

**Infrastructure as code (IaC):**

It is used to automate the process of environment setup and config the environment, the first step is creating environment.

The steeping of environments for Development, Testing and Production takes more time and in case, if we want to setup multiple environments like 100 machines takes a huge amount of manual effort. To reduce the manual effort of the environment setup we can use Iac code.

The below are the different Iac example tools:

1. Terraform
2. Pulumi
3. Cloud formation.

After creating the setup might need to config the environment like pattern changes, services setup, users, groups authentication setup, installing or updating packages, etc.

The below are the config management tools:

1. Ansible
2. Salt
3. Chef
4. Puppet

**CI/CD**:

To organize all the above steps one by one automatically and release the final product to users need CI/CD tool. The process of the integration of each stage continuously called continuous integration.

And CD have two different approaches:

1. **Continuous Delivery**: It is a strategy in software development where code changes to an application are release with permission or approvals into the production environment.
2. **Continuous Deployment**: It is a strategy in software development where code changes to an application are released automatically into the production environment.

The **CI/CD** tools:

1. Jenkins
2. Bamboo
3. GitLab runners
4. Git Lab actions
5. Circle Ci
6. Cloud pipelines

**Monitoring:**

After successfully release the product we need to monitor the performance of the application, environment, logs continuously to avoid the downtime to customers.

1. Prometheus
2. Nagios
3. Grafana
4. Slack
5. Datadog
6. App Dynamics
7. Sematext Monitoring
8. Cloud watch from AWS