CICLO 3

[FORMACIÓN POR CICLOS]
Desarrollo de
SOFTWARE

Gestión de la configuración Git

UNIVERSIDAD DE ANTIOQUIA
Facultad de Ingeniería

Gestión de la configuración

- Conjunto de actividades diseñadas para identificar y definir los elementos en el sistema que cambia y controlar el cambio de los artefactos a lo largo del ciclo de vida del desarrollo.
- Establece mecanismos para gestionar las diferentes versiones de los ítems para auditar e informar los cambios.
- El propósito es establecer la integridad del producto *software* a través del proceso de *software*.
- Los cambios no controlados en un desarrollo de *software* hacen que se fracase en el proyecto.

Gestión de la configuración

"El arte de coordinar el desarrollo de *software* para minimizar la confusión se denomina gestión de configuración. La gestión de configuración es el arte de identificar, organizar y controlar las modificaciones que sufre el *software* que construye un equipo de programación. La meta es maximizar la productividad minimizando los errores". Roger S. Pressman

Gestión de la configuración

- El control de versiones es una de las tareas fundamentales para la administración de un proyecto de desarrollo de *software* en general.
- Surge de la necesidad de mantener y llevar control del código que vamos programando, conservando sus distintos estados.
- Es absolutamente necesario para el trabajo en equipo, pero resulta útil incluso a desarrolladores independientes.

Git

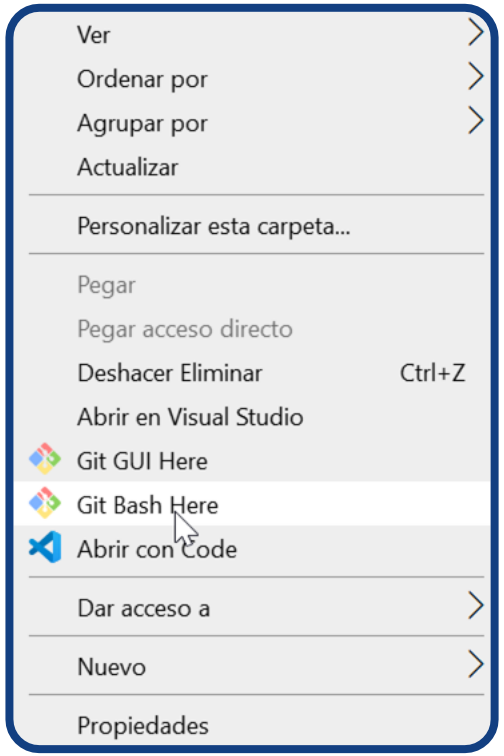
- Es un sistema de control de versiones distribuido, de código abierto, creado por Linus Torvalds.
- Cada desarrollador tiene el historial completo de su repositorio de código de manera local.
- Es multiplataforma, es decir, se pueden usar y crear repositorios locales en todos los sistemas operativos más comunes.

Git

- Es un *software* para el control de versiones.
- Es eficiente y confiable en el versionamiento del código fuente.
- Su propósito es llevar el registro de los cambios en los artefactos durante el ciclo de vida del desarrollo del producto.
- Es un *software* libre distribuido bajo los términos de la licencia pública general GNU.
- Físicamente no es más que una carpeta del sistema de archivos con código fuente de una aplicación.

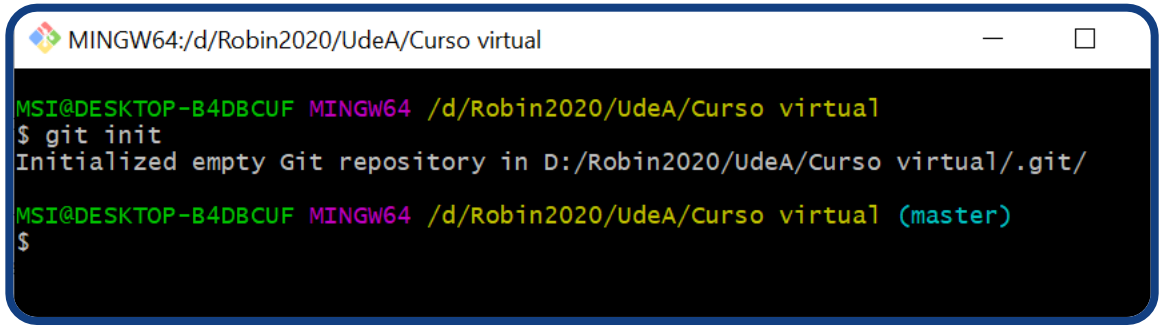
Git Bash en la práctica

- Descargue Git Bash: https://gitforwindows.org/ Instalar con la configuración por defecto.
- En un directorio de trabajo haga clic alterno y seleccione Git Bash:

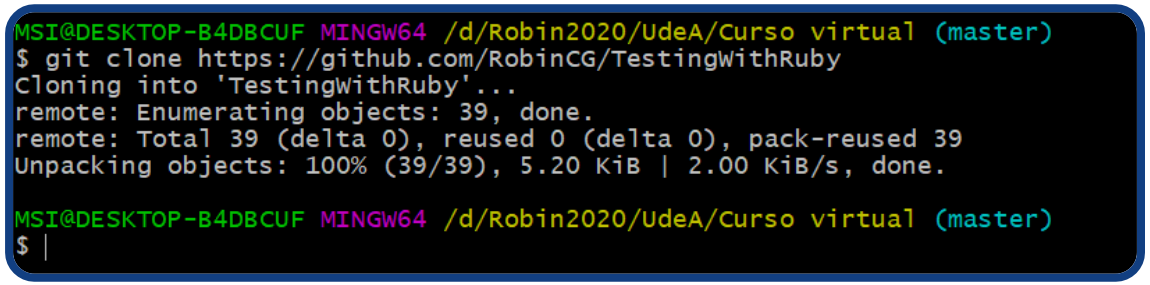


Git *Bash en la práctica

- Creamos el directorio de trabajo con el comando **git init**:



- Clonamos cualquier repositorio desde Github con el comando **git clone** "URL":



***La definición de Bash la puedes encontrar en el material de apoyo llamado glosario**

Git Bash en la práctica

- Listamos el contenido del directorio con el comando **ls**:

```
MSI@DESKTOP-B4DBCUF MINGW64 /d/Robin2020/UdeA/Curso virtual (master)
$ ls
'~$Azure DevOps - Gestion de la configuracion Video 4 - copia.pptx'
'2020-00095-DEVOPS_Estructura MOOC_V1_Karenina.docx'
'2020-00095-DEVOPS_Estructura MOOC_V2_Karenina.docx'
 2020-00095-DEVOPS_Generalidades_V2_Karenina.docx
 2020-00095-DEVOPS_Semana1_V2_Karenina.docx
'Azure DevOps - Aplicacion Cultura DevOps Video 2.pptx'
'Azure DevOps - Contenido para el video Herramientas e Introduccion.pptx'
'Azure DevOps - Gestion de la configuracion Video 4 - copia.pptx'
'Azure DevOps - Introduccion Video 1.pptx'
'Azure DevOps - Pilares DevOps Video 3.pptx'
'Complementar temas DevOps.txt'
 DevOps/
'DevOps para Dummy.pdf'
'Documento muy bueno de DevOps.pdf'
 TestingWithRuby/
```

- Nos ubicamos en el nuevo directorio con el comando **cd**:

```
MSI@DESKTOP-B4DBCUF MINGW64 /d/Robin2020/UdeA/Curso virtual (master)
$ cd TestingWithRuby/
```

- Identificamos en qué directorio se encuentra con **pwd**:

```
MSI@DESKTOP-B4DBCUF MINGW64 /d/Robin2020/UdeA/Curso virtual/TestingWithRuby (master)
$ pwd
/d/Robin2020/UdeA/Curso virtual/TestingWithRuby
```

Git Bash en la práctica

- Podemos usar el comando **git status** para identificar qué se modificó:

```
MSI@DESKTOP-B4DBCUF MINGW64 /d/Robin2020/UdeA/Curso virtual/TestingWithRuby (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   README.md
```

- Para crear una rama de trabajo usamos el comando **git checkout –b** <Nombre_rama>:

```
MSI@DESKTOP-B4DBCUF MINGW64 /d/Robin2020/UdeA/Curso virtual/TestingWithRuby (master)
$ git checkout -b feature-MyWork
Switched to a new branch 'feature-MyWork'
```

- Para adicionar todos los cambios usamos el comando **git add**:
- Para confirmar los cambios usamos git commit –m “Practica de DevOps”:

```
MSI@DESKTOP-B4DBCUF MINGW64 /d/Robin2020/UdeA/Curso virtual/TestingWithRuby (feature-MyWork)
$ git commit -m "Practica de DevOps"
[feature-MyWork befadc7] Practica de DevOps
 1 file changed, 1 insertion(+)
```

Git Bash en la práctica

- Podemos usar el comando **push** para publicar el nuevo cambio que se encuentra localmente hacia el servidor remoto de GitHub:
- git push origin <Nombre_feature>:

```
MSI@DESKTOP-B4DBCUF MINGW64 /d/Robin2020/UdeA/Curso virtual/TestingWithRuby (feature-MyWork)
$ git push origin feature-MyWork
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 356 bytes | 356.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
remote:
remote: Create a pull request for 'feature-MyWork' on GitHub by visiting:
remote:      https://github.com/RobinCG/TestingWithRuby/pull/new/feature-MyWork
remote:
To https://github.com/RobinCG/TestingWithRuby
 * [new branch]      feature-MyWork -> feature-MyWork
```