

PowerShell

Introducción

Es el nuevo editor de línea de comandos para el sistema operativo Windows, sustituirá en un futuro al viejo editor de línea de comandos que nos venía acompañando desde MS-DOS.

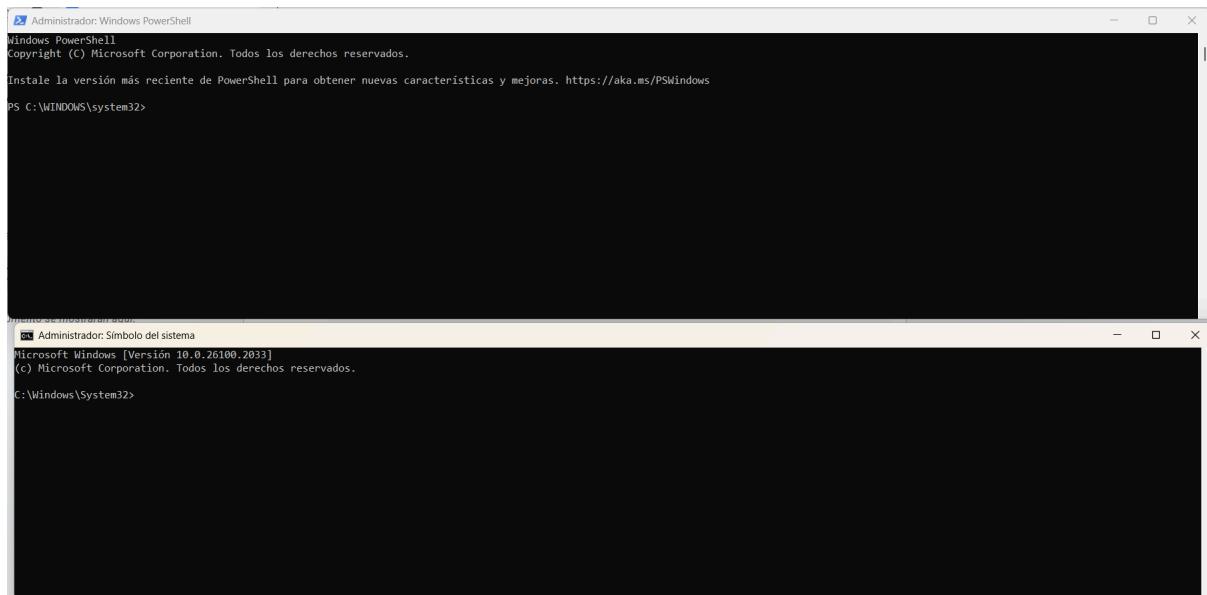


Foto arriba PowerShell abajo CMD

Los anteriores Batch Scripts de CMD puede ejecutarse normalmente como siempre, así como la mayoría de los comandos de este viejo interprete, algunas ventajas de este intérprete se mencionan a continuación:

- Permite el uso de los comandos más usuales de la terminal Bash de los sistemas Linux
- También permite la creación de nuevos comandos (**cmdlets**), solo que ahora se utiliza C# en vez de C, (puede usarse C++, sólo implica esfuerzos adicionales)
- Los Cmdlets tienen más versatilidad que los simples comandos de CMD o Linux.

El esfuerzo debe entonces dirigirse al aprendizaje de los nuevos CMDlets y el manejo de la sintaxis para esas nuevas instrucciones. Estos comandos son presentados en el formato *Verbo-Sustantivo* (de ahí el nombre) como por ejemplo “Get-Process” o “Get-Content”.

Otra mención especial es la orientación a Objetos que maneja, los CMDlets no solo ejecutan una acción sino que también tienen *Propiedades* y *Métodos*, por lo que para poder mostrar el contenido de estos objetos deberemos usar “tuberías” (*pipelines* |) para poder redirigir la entrada a los comandos especiales de formato

-ft Formato Tabla

-fl Formato Lista

-fw Formato Amplio, solo muestra una característica completa

Debido a que los CMDlets tienen Propiedades también se tendrá que usar una sintaxis similar a una “Query” propia de las bases de datos para poder filtrar los resultados que nos proporciona la ejecución de estos comandos.

Ejemplo:

```
Get-Process | Sort-Object -Property CPU -Descending | Select-Object -Property Name,CPU -First 10
```

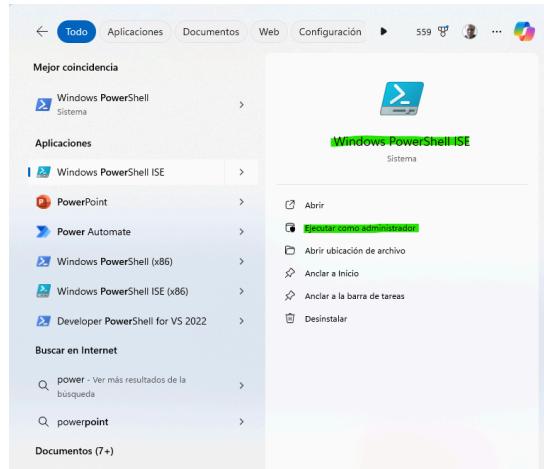
```
PS C:\WINDOWS\system32> Get-Process | Sort-Object -Property CPU -Descending | Select-Object -Property Name,CPU -First 10
Name          CPU
---          ---
System        17261.671875
svchost       3950.984375
WTabletServicePro 3254.3125
audiogd       2765.234375
mc-fw-host   1609.34375
svchost       1492.265625
WmiPrvSE     1433.859375
AsusWiFiSmartConnect 1173.078125
servicehost   1036.21875
svchost       906.625
PS C:\WINDOWS\system32>
```

el CMDlet busca todos los procesos, pedimos que se filtren en orden ascendente acorde al Campo CPU, y asu vez solo se muestren los campos Name y CPU de los primeros 10 Registros, el formato de salida por defecto es por tabla, pero podemos pedir que sea por lista

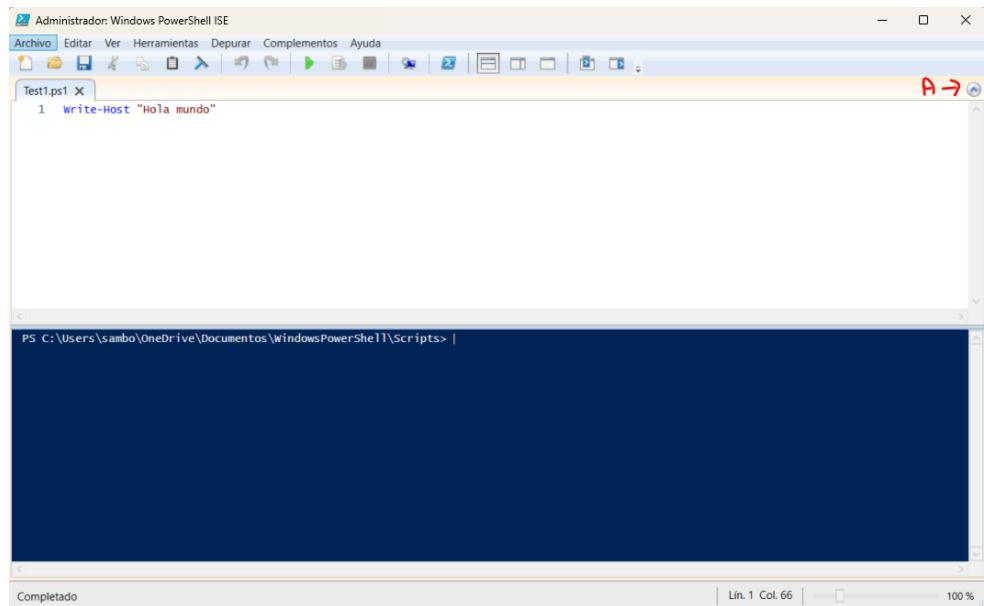
```
PS C:\WINDOWS\system32> Get-Process | Sort-Object -Property CPU -Descending | Select-Object -Property Name,CPU -First 10 | fl
Name : System
CPU : 17418.21875
Name : svchost
CPU : 4002.234375
Name : WTabletServicePro
CPU : 3279.34375
Name : audiogd
CPU : 2829.984375
Name : mc-fw-host
CPU : 1620.765625
Name : svchost
CPU : 1523.859375
Name : WmiPrvSE
CPU : 1445.421875
Name : msedge
CPU : 1203.859375
Name : AsusWiFiSmartConnect
CPU : 1192
Name : servicehost
CPU : 1094.3125
```

Scripting

Al ser un intérprete de comandos, también es posible escribir Scripts de automatización, cuenta con las instrucciones básicas de cualquier lenguaje estructurado, por seguridad Powershell tiene una política de restricción a la ejecución de Scripts, por lo que se recomienda usar la versión de diseño de Powershell y en modo de administrador:



Nos genera la siguiente ventana:



Usando el botón marcado en A podemos minimizar el área de edición de scripts, para ampliar el área del intérprete de Powershell, ésta ventana nos permite editar y ejecutar nuestros Scripts de manera “segura”, sin necesidad de cambiar las políticas de ejecución, resta mencionar que la mayoría de los viejos comando de CMD y Bash funcionaran en este interprete, solo basta movernos a una carpeta donde tengamos permisos de escritura y ejecución (justo como en Linux) para comenzar a experimentar la edición de nuestros primeros scripts (para ejecutar un script use la tecla F5 o los botones), a continuación se muestran las instrucciones de control más usuales:

If

La siguiente imagen muestra el uso de la instrucción:

A screenshot of the Windows PowerShell ISE interface. The script 'Test1.ps1' contains an 'if-else' block. Line 3 creates a variable '\$Respuesta="true"'. Line 4 starts an if-block. Line 5 contains the command 'Write-Host "Hola mundo"'. Line 6 ends the if-block. Line 7 starts an else-block. Line 8 contains the command 'Write-Host "Adios mundo"'. Line 9 ends the else-block. Line 11 contains a comment '#B'. Lines 12 and 13 contain the commands 'Write-Host "Opcion Falsa"' and 'Write-Host ""' respectively. Line 15 sets the variable '\$Respuesta="false"'. Line 16 starts another if-block. Line 17 contains the command 'Write-Host "Hola mundo"'. The status bar at the bottom shows 'PS C:\Users\sambo\OneDrive\Documentos\WindowsPowerShell\Scripts> .\Test1.ps1'. The output window shows the results of the script execution: 'Opcion Verdadera', 'Hola mundo', 'Opcion Falsa', and 'Adios mundo'.

En ‘A’ (línea 3) podemos ver la creación de un “contenedor” (también conocido como variable en otros lenguajes de programación), y en ‘B’ podemos ver el envío de un carácter de salto de línea.

La sentencia IF usa los operadores de comparación -le -lt -eq -gt -ge (menor o igual, menor, igual, mayor, mayor o igual, respectivamente), y engloba los grupos de sentencias a ejecutar en cada caso entre corchetes { }.

Switch

La instrucción de selección múltiple:

```
Test1.ps1 Test2_Switch.ps1 X
1 $respuesta= 3
2 Write-Host "Acorde al contenido de la variable respuesta es la eleccion"
3 Write-Host ""
4 Switch($respuesta){
5     1 {Write-Host "Seleccion 1"}
6     2 {Write-Host "Seleccion 2"}
7     3 {Write-Host "Seleccion 3"}
8     Default {Write-Host "Seleccion no en lista"}
9 }
```

```
PS C:\Users\sambo\OneDrive\Documentos\WindowsPowerShell\Scripts> .\Test2_Switch.ps1
Acorde al contenido de la variable respuesta es la eleccion
Seleccion 1

PS C:\Users\sambo\OneDrive\Documentos\WindowsPowerShell\Scripts> .\Test2_Switch.ps1
Acorde al contenido de la variable respuesta es la eleccion
Seleccion 2

PS C:\Users\sambo\OneDrive\Documentos\WindowsPowerShell\Scripts> .\Test2_Switch.ps1
Acorde al contenido de la variable respuesta es la eleccion
Seleccion no en lista

PS C:\Users\sambo\OneDrive\Documentos\WindowsPowerShell\Scripts> .\Test2_Switch.ps1
Acorde al contenido de la variable respuesta es la eleccion
Seleccion 3

PS C:\Users\sambo\OneDrive\Documentos\WindowsPowerShell\Scripts>
```

Las selecciones pueden incluir caracteres y se comparan con valores de tipo cadena encerrados entre “ ” (comillas). El bloque completo de la sentencia Switch está agrupado entre dos {corchetes} y cada opción elegible también.

For

```
Test1.ps1 Test2_Switch.ps1 Test3_For.ps1 X
1 For ($contador=1; $contador -lt 10; $contador++)
2 {
3     Write-Host "Contador es: $contador"
4 }
```

```
PS C:\Users\sambo\OneDrive\Documentos\WindowsPowerShell\Scripts> .\Test3_For.ps1
Contador es: 1
Contador es: 2
Contador es: 3
Contador es: 4
Contador es: 5
Contador es: 6
Contador es: 7
Contador es: 8
Contador es: 9

PS C:\Users\sambo\OneDrive\Documentos\WindowsPowerShell\Scripts>
```

Tiene una sintaxis igual que C y un funcionamiento similar

Do..While

```
Test1.ps1 Test2_Switch.ps1 Test3_For.ps1 Test4_DoWhile.ps1 X
1 $contador=2
2 Do
3 {
4     Write-Host "Contador es: $contador"
5     $contador++
6 } while ( $contador -lt 10 )
```

```
PS C:\Users\sambo\OneDrive\Documentos\WindowsPowerShell\Scripts> .\Test4_DoWhile.ps1
Contador es: 2
Contador es: 3
Contador es: 4
Contador es: 5
Contador es: 6
Contador es: 7
Contador es: 8
Contador es: 9

PS C:\Users\sambo\OneDrive\Documentos\WindowsPowerShell\Scripts>
```

Cuando deseamos que el bloque de código se ejecute una vez al menos.

Do...Until

The screenshot shows the Windows PowerShell ISE interface. The top tab bar has tabs for Test1.ps1, Test2_Switch.ps1, Test3_For.ps1, Test4_DoWhile.ps1, and Test5_DoUntil.ps1. The Test5_DoUntil.ps1 tab is active. The code in the editor is:

```
1 Write-Host "Si usamos -eq";Write-Host ""
2 $contador=2
3 Do
4 {
5     Write-Host "Contador es: $contador"
6     $contador++
7 } Until ($contador -eq 10)
8 Write-Host "Si usamos -lt";Write-Host ""
9 $contador=2
10 Do
11 {
12     Write-Host "Contador es: $contador"
13     $contador++
14 } Until ($contador -lt 10)
```

Red brackets labeled (A) and (B) highlight the two different loop structures. The output window below shows the execution of the script:

```
PS C:\Users\sambo\OneDrive\Documentos\WindowsPowerShell\Scripts> .\Test5_DoUntil.ps1
Si usamos -eq
Contador es: 2
Contador es: 3
Contador es: 4
Contador es: 5
Contador es: 6
Contador es: 7
Contador es: 8
Contador es: 9
Si usamos -lt
Contador es: 2
```

Una sutil diferencia a Do..While, observe las piezas de código (A) y (B), podría decirse que Do...Until es “excluyente” con la condición a diferencia de su contra parte.

While

The screenshot shows the Windows PowerShell ISE interface. The top tab bar has tabs for Test1.ps1, Test2_Switch.ps1, Test3_For.ps1, Test4_DoWhile.ps1, Test5_DoUntil.ps1, and Test6_Whileps1.ps1. The Test6_Whileps1.ps1 tab is active. The code in the editor is:

```
1 $contador=2
2 While ($contador -lt 10)
3 {
4     Write-Host "Contador es: $contador"
5     $contador+=2
6 }
```

The output window shows the execution of the script:

```
PS C:\Users\sambo\OneDrive\Documentos\WindowsPowerShell\Scripts> .\Test6_Whileps1.ps1
Contador es: 2
Contador es: 4
Contador es: 6
Contador es: 8
```

No hay mucho que decir de las instrucciones de control, salvo que su comportamiento es el esperado como en otros lenguajes de programación.

Primeros pasos con los CMD-Lets de Powershell

La manera más sencilla de conocer los comandos que ofrece el CLI (Command Line Interpreter) es experimentar con ellos, para poder encontrar información sobre alguna “acción” a realizar, utilizaremos el CMDLet *Get-Command* (note el uso de la notación CamelCase en los nombres de dichos comandos):

The screenshot shows the Windows PowerShell ISE interface. The command `Get-Command *process*` is run in the console. The output is a table showing various cmdlets related to processes:

CommandType	Name	Version	Source
Cmdlet	ConvertTo-ProcessMitigationPolicy	1.0.12	ProcessMitigations
Cmdlet	Debug-Process	3.1.0.0	Microsoft.PowerShell.Management
Cmdlet	Enter-PSToken	3.0.0.0	Microsoft.PowerShell.Core
Cmdlet	Exit-PSToken	3.0.0.0	Microsoft.PowerShell.Core
Cmdlet	Get-Process	3.1.0.0	Microsoft.PowerShell.Management
Cmdlet	Get-ProcessMitigation	1.0.12	ProcessMitigations
Cmdlet	Get-PSToken	3.0.0.0	Microsoft.PowerShell.Core
Cmdlet	Invoke-LapsPolicyProcessing	1.0.0.0	LAPS
Cmdlet	Set-ProcessMitigation	1.0.12	ProcessMitigations
Cmdlet	Start-Process	3.1.0.0	Microsoft.PowerShell.Management
Cmdlet	Stop-Process	3.1.0.0	Microsoft.PowerShell.Management
Cmdlet	Wait-Process	3.1.0.0	Microsoft.PowerShell.Management

En la figura anterior (A) muestra el uso de Get-Command seguido de las letras que deseamos buscar, en este caso *process* significa que no importa el número ni el contenido de los caracteres que estén antes de la palabra *process* y lo mismo para el final de la palabra, podemos usar una tubería | y aplicar el formato de lista a la información (el formato Tabla es el mostrado por defecto), dicha salida muestra las “propiedades” de los registros obtenidos en la búsqueda (D), los cuales pueden ser enviados a un Contenedor, para poder acceder a cada uno de los campos de cada registro:

```
PS C:\Users\sambo> $Salida=Get-Command *process*
PS C:\Users\sambo> $Salida.Name
ConvertTo-ProcessMitigationPolicy
Debug-Process
Enter-PSTHostProcess
Exit-PSTHostProcess
Get-Process
Get-ProcessMitigation
Get-PSHostProcessInfo
Invoke-LapsPolicyProcessing
Set-ProcessMitigation
Start-Process
Stop-Process
Wait-Process
PS C:\Users\sambo> $Salida.Version

Major Minor Build Revision
----- ----- ----- -----
1       0      12     -1
3       1      0      0
3       0      0      0
3       0      0      0
3       1      0      0
1       0      12     -1
3       0      0      0
1       0      0      0
1       0      12     -1
3       1      0      0
3       1      0      0
3       1      0      0
```

En el ejemplo se crea el contenedor \$Salida y se accede a cada uno de los campos usando el operador de acceso (.) y el nombre del campo que deseamos mostrar, algunos campos están formados a su vez por otros campos (“propiedades”) como en el caso de campo *Version*.

Tras usar el CMD-Let Get-Command *process*, podemos ubicar cual la instrucción sobre la que estamos interesados en usar:

CommandType	Name	Version	Source
Cmdlet	ConvertTo-ProcessMitigationPolicy	1.0.12	ProcessMitigations
Cmdlet	Debug-Process	3.1.0.0	Microsoft.PowerShell.Management
Cmdlet	Enter-PSTHostProcess	3.0.0.0	Microsoft.PowerShell.Core
Cmdlet	Exit-PSTHostProcess	3.0.0.0	Microsoft.PowerShell.Core
Cmdlet	Get-Process	3.1.0.0	Microsoft.PowerShell.Management
Cmdlet	Get-ProcessMitigation	1.0.12	ProcessMitigations
Cmdlet	Get-PSHostProcessInfo	3.0.0.0	Microsoft.PowerShell.Core
Cmdlet	Invoke-LapsPolicyProcessing	1.0.0.0	LAPS
Cmdlet	Set-ProcessMitigation	1.0.12	ProcessMitigations
Cmdlet	Start-Process	3.1.0.0	Microsoft.PowerShell.Management
Cmdlet	Stop-Process	3.1.0.0	Microsoft.PowerShell.Management
Cmdlet	Wait-Process	3.1.0.0	Microsoft.PowerShell.Management

Usando el CMD-Let *Get-Help <Nombre Buscado>* podemos desplegar la información sobre dicha instrucción:

```
PS C:\Users\sambo> Get-Help Get-Process

NOMBRE
  Get-Process

SINOPSIS
  Gets the processes that are running on the local computer or a remote computer.

SYNTAXIS
  Get-Process [[-Name] <System.String[]>] [-ComputerName <System.String[]>] [-FileVersionInfo] [-Module]
  [-CommonParameters]

  Get-Process [-ComputerName <System.String[]>] [-FileVersionInfo] -Id <System.Int32[]> [-Module]
  [-CommonParameters]

  Get-Process [-ComputerName <System.String[]>] [-FileVersionInfo] -InputObject <System.Diagnostics.Process[]>
  [-Module] [-CommonParameters]

  Get-Process -Id <System.Int32[]> -IncludeUserName [<CommonParameters>]

  Get-Process [[-Name] <System.String[]>] -IncludeUserName [<CommonParameters>]

  Get-Process -IncludeUserName -InputObject <System.Diagnostics.Process[]> [<CommonParameters>]
```

Muy similar a otras Terminales, también es posible obtener información sobre las Propiedades y Métodos relacionados a cada Contenedor, por ejemplo si redirigimos la salida de nuestro Contenedor hacia el CMDLet *Get-Member* podemos ver las estructuras que lo forman:

```
PS C:\Users\sambo> $Salida | Get-Member

TypeName: System.Management.Automation.CmdletInfo

Name          MemberType   Definition
---          -----
Equals        Method      bool Equals(System.Object obj) (A)
 GetHashCode  Method      int GetHashCode()
 GetType       Method      type GetType()
 ResolveParameter Method    System.Management.Automation.ParameterMetadata ResolveParameter(string name)
 ToString      Method      string ToString()
 CommandType   Property   System.Management.Automation.CommandTypes CommandType {get;}
 DefaultParameterSet Property string DefaultParameterSet {get;}
 Definition    Property   string Definition {get;}
 Helpfile     Property   string Helpfile {get;}
 ImplementingType Property  type ImplementingType {get;}
 Module       Property   psmoduleinfo Module {get;}
 ModuleName   Property   string ModuleName {get;}
 Name          Property   string Name {get;} (A)
 Noun         Property   string Noun {get;}
 Options      Property   System.Management.Automation.ScopedItemOptions Options {get;set;}
 OutputType   Property   System.Collections.ObjectModel.ReadOnlyCollection[System.Management.Automation.PS...
 Parameters    Property   System.Collections.Generic.Dictionary[string, System.Management.Automation.Paramet...
 ParameterSets Property   System.Collections.ObjectModel.ReadOnlyCollection[System.Management.Automation.Co...
 PSSnapin     Property   System.Management.Automation.PSSnapinInfo PSSnapin {get;}
 RemotingCapability Property System.Management.Automation.RemotingCapability RemotingCapability {get;}
 Source       Property   string Source {get;}
 Verb         Property   string Verb {get;}
 Version      Property   version Version {get;} (A)
 Visibility   Property   System.Management.Automation.SessionStateEntryVisibility Visibility {get;set;}
 DLL         ScriptProperty System.Object DLL {get=$this.ImplementingType.Assembly.Location;}
```

Podemos constar en (A) que Name y Version son efectivamente propiedades del contenedor así como los métodos (funciones) que operan sobre las propiedades (datos del contenedor), y asu vez podemos ver que la salida del CMDLet *Get-Member* genera otro conjunto de registros con sus respectivos campos (propiedades) y se muestra en formato Tabla por defecto. Como lector podrá pensar si es posible guardar la salida en otro contenedor y acceder a cada uno de los miembros de el.

También podemos ejecutar grupos de instrucciones que guarden algún orden o secuencia entre sus resultados, es decir probar líneas de scripts directamente en el interprete:

```
PS C:\Users\sambo> $handleCount=0
PS C:\Users\sambo> foreach($process in Get-Process){$handleCount += $process.Handles}
PS C:\Users\sambo> $handleCount
155295
PS C:\Users\sambo> |
```

Así podemos simplemente realizar un copia y pega de dichas instrucciones en el archivo Script ya que hemos verificado su funcionalidad y eso ayuda mucho al proceso de prueba y depuración de un Script.

Contenedores (variables)

Podemos crear Contenedores para almacenar nuestros Objetos, sin necesidad de expresar el tipo de dato manejan, los Objetos almacenados por sí sólos qué tipo de operaciones pueden realizarse:

```
PS C:\Users\sambo> $caracter1= "1"
PS C:\Users\sambo> $caracter2= "2"
PS C:\Users\sambo> $c=$caracter1 + $caracter2 } (A)
PS C:\Users\sambo> $c
12
PS C:\Users\sambo> $num1= 1
PS C:\Users\sambo> $num2= 2
PS C:\Users\sambo> $Cnum= $num1 + $num2 } (B)
PS C:\Users\sambo> $Cnum
3
PS C:\Users\sambo>
```

Los contenedores en (B) del tipo numérico saben que el símbolo + indica la operación suma y los contenedores en (A) saben que la operación a realizar es la concatenación de caracteres. También existen contenedores que tienen más propiedades y métodos en su estructura:

```
PS C:\Users\sambo> $fecha = Get-Date <- (A)
PS C:\Users\sambo> $fecha
viernes, 1 de noviembre de 2024 04:38:40 a. m. <- (B)

PS C:\Users\sambo> $fecha | Get-Member

    TypeName: System.DateTime
(C)
↓
Name        MemberType   Definition
----        -----      -----
Add         Method      datetime Add(timespan value)
AddDays     Method      datetime AddDays(double value)
AddHours    Method      datetime AddHours(double value)
AddMilliseconds Method    datetime AddMilliseconds(double value)
AddMinutes  Method      datetime AddMinutes(double value)
AddMonths   Method      datetime AddMonths(int months)
AddSeconds  Method      datetime AddSeconds(double value)
AddTicks    Method      datetime AddTicks(long value)
AddYears    Method      datetime AddYears(int value)
CompareTo   Method      int CompareTo(System.Object value), int CompareTo(datetime value), int IComparable.CompareTo(System.Object obj), int...
Equals     Method      bool Equals(System.Object value), bool Equals(datetime value), bool IEquatable[datetime].Equals(datetime other)
GetDateTimeFormats Method    string[] GetDateTimeFormats(), string[] GetDateTimeFormats(System.IFormatProvider provider), string[] GetDateTimeFor...
```

El contenedor \$fecha de tipo Date, se presenta en la salida estándar como en (B) (implica el uso de un método para mostrarse) y contiene a su vez más propiedades y métodos para interactuar con él, usando el operador de acceso (.) puede hacerse uso de los métodos de los objetos:

```
PS C:\Users\sambo> $fecha.GetType()
IsPublic IsSerial Name                                     BaseType
-----  -----  ----
True     True    DateTime                               System.ValueType

PS C:\Users\sambo>
```

Pueden guardar una colección de datos de un mismo tipo, en Formaciones (Arrays):

```
PS C:\Users\sambo> $MiArreglo="ejemplo1", "ejemplo2", "ejemplo3"
PS C:\Users\sambo> $MiArreglo
ejemplo1
ejemplo2
ejemplo3
PS C:\Users\sambo> $MiArreglo[1]
ejemplo2
PS C:\Users\sambo>
```

El Contendor mostrará todos los elementos o mediante el operador [] un elemento en particular, se observa que el índice menor es [0].

Es posible crear Formaciones (Arreglos) de varias dimensiones y Formaciones de Formaciones:

```
PS C:\Users\sambo> $MiArreglo="ejemplo1","ejemplo2","ejemplo3"
PS C:\Users\sambo> $MiArreglo
ejemplo1
ejemplo2
ejemplo3
PS C:\Users\sambo> $MiArreglo[1]
ejemplo2
PS C:\Users\sambo> $Arreglo2Dimension= ("EJEMPLO1", $MiArreglo[0]), ("EJEMPLO2", $MiArreglo[1])
PS C:\Users\sambo> $Arreglo2Dimension
EJEMPLO1
ejemplo1
EJEMPLO2
ejemplo2
PS C:\Users\sambo> $Arreglo2Dimension[1][0]
EJEMPLO2
PS C:\Users\sambo> $Arreglo2Dimension[1][1]
ejemplo2
```

Para acceder a los elementos se usan varios operadores de acceso especificando el nivel de profundidad del o de los datos que queremos acceder.

Hashes

Los arreglos tienen el inconveniente de que el acceso es mediante un índice numérico y que el agregar miembros adicionales o remover es complicado ya que la longitud del arreglo está “fija”, para evitar esas limitantes tenemos las contenedores de tablas hash:

```
[DBG]: PS C:\Users\sambo> $users = @{"john.doe" = "jdoe"; "jane.doe" = "jdoe1"}
[DBG]: PS C:\Users\sambo> $users ←(A) ↴
Name          Value
----          ----
john.doe      jdoe
jane.doe      jdoe1

[DBG]: PS C:\Users\sambo> $users["john.doe"] ←(B)
jdoe
[DBG]: PS C:\Users\sambo> $users.add("John.Smith", "jsmith") ←(C)
[DBG]: PS C:\Users\sambo> $users["jane.doe"] = "jadoe" ←(D)
[DBG]: PS C:\Users\sambo> $users.remove("jhon.doe") ←(E)
[DBG]: PS C:\Users\sambo> $users ←(F)

Name          Value
----          ----
jane.doe      jadoe
John.Smith    jsmith
john.doe      jdoe

[DBG]: PS C:\Users\sambo> |
```

Los cuales almacenan los datos como pares “Clave-Valor” (A), la búsqueda se realiza por valor no por índice (B), agregar nuevos elementos es más sencillo (C), modificarlos también (D) así como removerlos (E), observe las diferencias entre la tabla (F) y la (A).

Manipulación de cadenas

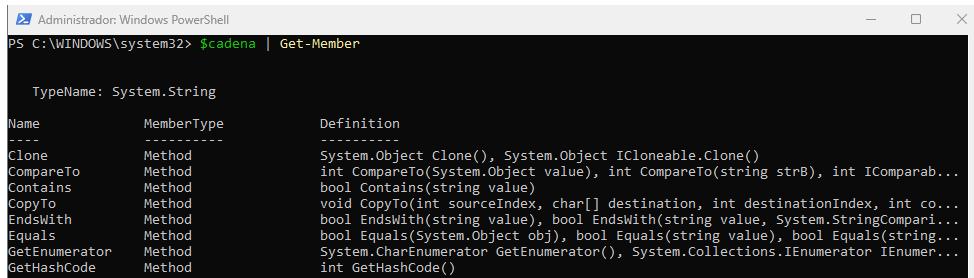
Métodos

Una tarea muy común es el formato de la salida de Powershell, esto debido a que no encaje o muestre correctamente lo que necesitamos. Mostraremos primero los métodos para la manipulación de objetos tipo cadena.

Creamos un contenedor de tipo cadena

```
PS C:\WINDOWS\system32> $cadena="PowerShell"
PS C:\WINDOWS\system32> $cadena.GetType()
IsPublic IsSerial Name                                     BaseType
-----  -----  -----
True     True    String                                  System.Object
```

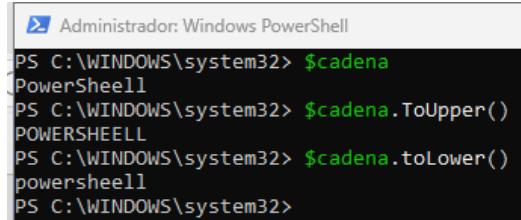
Y usamos una tubería para mostrar los métodos disponibles:



```
Administrator: Windows PowerShell
PS C:\WINDOWS\system32> $cadena | Get-Member

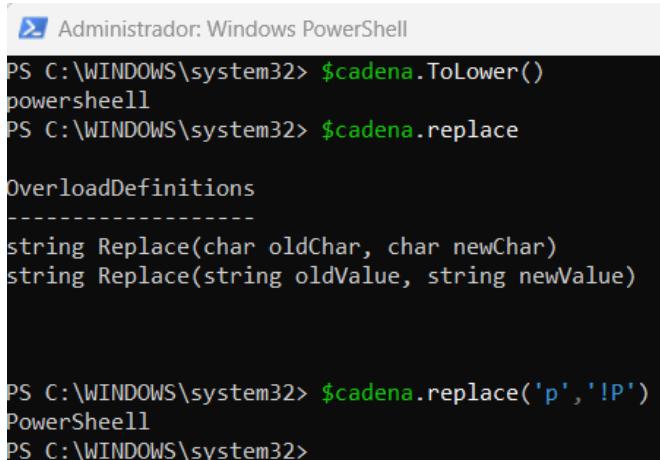
TypeName: System.String
Name      MemberType      Definition
----      ----          -----
Clone    Method          System.Object Clone(), System.Object ICloneable.Clone()
CompareTo Method          int CompareTo(System.Object value), int CompareTo(string strB), int IComparab...
Contains Method          bool Contains(string value)
CopyTo   Method          void CopyTo(int sourceIndex, char[] destination, int destinationIndex, int co...
EndsWith Method          bool EndsWith(string value), bool EndsWith(string value, System.StringCompar...
Equals   Method          bool Equals(System.Object obj), bool Equals(string value), bool Equals(string...
GetEnumerator Method        System.CharEnumerator GetEnumerator(), System.Collections.IEnumerator IEnumer...
GetHashCode Method        int GetHashCode()
```

El operador de acceso nos permite usar los métodos:



```
Administrator: Windows PowerShell
PS C:\WINDOWS\system32> $cadena
PowerSheell
PS C:\WINDOWS\system32> $cadena.ToUpper()
POWERSHEELL
PS C:\WINDOWS\system32> $cadena.ToLower()
powersheell
PS C:\WINDOWS\system32>
```

En el ejemplo los métodos `ToUpper()` y `ToLower()` convierte toda la cadena a mayúsculas o minúsculas respectivamente. Si queremos saber cómo usar un método del listado de los disponibles para un objeto, usamos la sintaxis `<Objeto>.<Método>` sin el parentesis:



```
Administrator: Windows PowerShell
PS C:\WINDOWS\system32> $cadena.ToLower()
powersheell
PS C:\WINDOWS\system32> $cadena.replace

OverloadDefinitions
-----
string Replace(char oldChar, char newChar)
string Replace(string oldValue, string newValue)

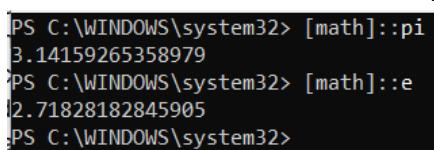
PS C:\WINDOWS\system32> $cadena.replace('p','!P')
PowerSheell
PS C:\WINDOWS\system32>
```

En la imagen superior podemos ver que `$cadena.replace` muestra la sintaxis básica de uso del método, el cual requiere dos parámetros (*valor_a_reemplazar, valor_que_reemplaza*) así es como cambiamos la 'p' por '!P' en el valor almacenado en el objeto `$cadena`.

Dar una explicación de cada uno de los métodos disponibles sería extenso, se recomienda usar la sintaxis de ayuda para obtener información sobre el método así como los recursos en línea adicionales que provee Microsoft. Nos conformamos con el ejemplo anterior de uso y con los ejemplos siguientes.

Manipulación Numérica

A parte de los métodos disponibles para cada objeto, también podemos contar con paquetes que nos proporcionan más métodos, uno muy útil es `[math]` por ejemplo contiene los valores de las constantes más comunes por ejemplo:



```
PS C:\WINDOWS\system32> [math]::pi
3.14159265358979
PS C:\WINDOWS\system32> [math]::e
2.71828182845905
PS C:\WINDOWS\system32>
```

Nos proporciona métodos de cálculo adicionales como la raíz cuadrada:

```
PS C:\WINDOWS\system32> [math]::sqrt("16")
4
PS C:\WINDOWS\system32> $raiz=16
PS C:\WINDOWS\system32> [math]::sqrt($raiz)
4
```

Usamos alguna de las siguientes líneas de código para obtener el listado de los métodos disponibles en el grupo [math]:

```
[math] |Get-Member -Static
Get-Member -InputObject $($[math]) -Static
```

The screenshot shows the Windows PowerShell interface with the title bar "Administrador: Windows PowerShell". The command PS C:\WINDOWS\system32> [math] |Get-Member -Static is run. The output lists various static methods of the System.Math class, such as Abs, Acos, Asin, Atan, Atan2, BigMul, Ceiling, Cos, Cosh, DivRem, Equals, and Exp, along with their definitions.

Name	MemberType	Definition
Abs	Method	static sbyte Abs(sbyte value), static int16 Abs(int16 value), static int Abs(int value), ...
Acos	Method	static double Acos(double d)
Asin	Method	static double Asin(double d)
Atan	Method	static double Atan(double d)
Atan2	Method	static double Atan2(double y, double x)
BigMul	Method	static long BigMul(int a, int b)
Ceiling	Method	static decimal Ceiling(decimal d), static double Ceiling(double a)
Cos	Method	static double Cos(double d)
Cosh	Method	static double Cosh(double value)
DivRem	Method	static int DivRem(int a, int b, [ref] int result), static long DivRem(long a, long b, [re...
Equals	Method	static bool Equals(System.Object objA, System.Object objB)
Exp	Method	static double Exp(double d)

Formato a Números

Recortar, redondear o conversiones entre sistemas numéricos requieren modificar la apariencia de un número, algunas operaciones se aplican al objeto \$número en la siguiente imagen:

The screenshot shows the Windows PowerShell interface with the title bar "Administrador: Windows PowerShell". The command PS C:\WINDOWS\system32> \$numero=[math]::pi is run, followed by PS C:\WINDOWS\system32> \$numero (A). Then, PS C:\WINDOWS\system32> "{0:N4}" -f \$numero (B) is run, resulting in 3.1416. Next, PS C:\WINDOWS\system32> \$numero (C) is run, showing the original value 3.14159265358979. Then, PS C:\WINDOWS\system32> \$numero="{0:N0}" -f \$numero (D) is run, resulting in 3. Finally, PS C:\WINDOWS\system32> "{0:X0}" -f \$numero #convertido a un hexadecimal (E) is run, resulting in 3. PS C:\WINDOWS\system32> \$numero=10 (F) is run, followed by PS C:\WINDOWS\system32> "{0:X0}" -f \$numero #convertido a un hexadecimal A is run, resulting in A.

En (A)Creamos un valor numérico flotante, (B)redondeamos a 4 decimales (C)el objeto no fue alterado, en (D)el objeto se vuelve entero (E)es convertido en Hexadecimal (F)enfatizado

Formato a medidas de Bytes

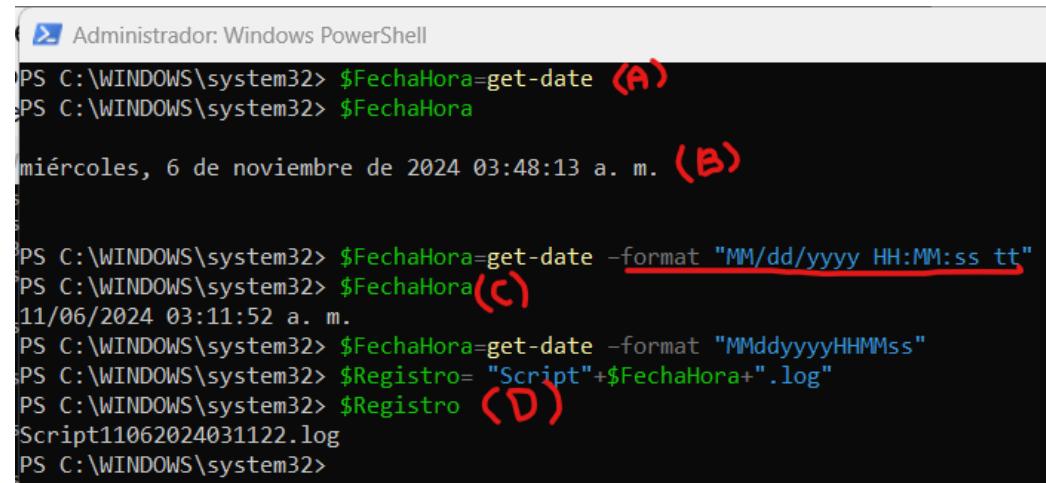
Esta opción es bastante útil cuando se desea trabajar con unidades de medidas en Bytes, observe el siguiente fragmento de código:

The screenshot shows the Windows PowerShell interface with the title bar "Administrador: Windows PowerShell". The command PS C:\WINDOWS\system32> \$Memoria= 16849174528 is run, followed by PS C:\WINDOWS\system32> \$Memoria / 1TB. This results in 0.0153242349624634. Then, PS C:\WINDOWS\system32> \$Memoria / 1GB is run, resulting in 15.6920166015625. PS C:\WINDOWS\system32> \$Memoria / 1MB is run, resulting in 16068.625. PS C:\WINDOWS\system32> \$Memoria / 1KB is run, resulting in 16454272. Finally, PS C:\WINDOWS\system32> \$Memoria is run, resulting in 16849174528.

El contenido del objeto \$Memoria se ha dividido en segmentos de 1TB (TeraByte), 1GB (GigaByte), 1 MB (MegaByte) y 1 KB(KiloByte). Poder calcular el espacio en la memoria o en el Disco Duro, es útil para poder administrar el espacio restante para almacenar y ejecutar programas.

Manipulación de Objetos tipo Fecha y Hora

Usualmente requerimos programar la ejecución de ciertos procesos en una determinada hora y fecha, o calcular eventos importantes futuros o pasados.

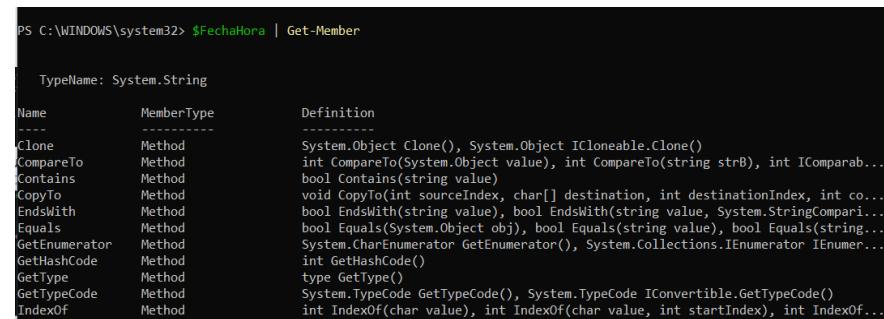


```
PS C:\WINDOWS\system32> $FechaHora=get-date (A)
PS C:\WINDOWS\system32> $FechaHora
miércoles, 6 de noviembre de 2024 03:48:13 a. m. (B)

PS C:\WINDOWS\system32> $FechaHora=get-date -format "MM/dd/yyyy HH:MM:ss tt"
PS C:\WINDOWS\system32> $FechaHora(C)
11/06/2024 03:11:52 a. m.
PS C:\WINDOWS\system32> $FechaHora=get-date -format "MMddyyyyHHMMss"
PS C:\WINDOWS\system32> $Registro= "Script"+$FechaHora+".log"
PS C:\WINDOWS\system32> $Registro (D)
Script11062024031122.log
PS C:\WINDOWS\system32>
```

El inciso (A) se crea un objeto tipo Fecha, y (B) muestra el formato de salida “estándar”, (C) muestra cómo modificar la salida de objeto, esto es útil para crear nombres de archivos de registro únicos para registrar acciones que se efectúan con cierta frecuencia.

El uso de una tubería junto al CMDLet Get-Member lista los métodos y propiedades del objeto \$FechaHora

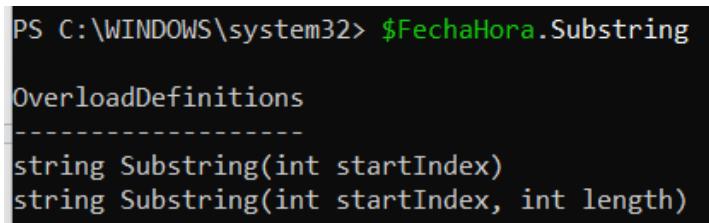


```
PS C:\WINDOWS\system32> $FechaHora | Get-Member

TypeName: System.String

Name      MemberType      Definition
----      ----
Clone    Method   System.Object Clone(), System.Object ICloneable.Clone()
CompareTo Method   int CompareTo(System.Object value), int CompareTo(string strB), int IComparab...
Contains Method   bool Contains(string value)
CopyTo    Method   void CopyTo(int sourceIndex, char[] destination, int destinationIndex, int co...
EndsWith  Method   bool EndsWith(string value), bool EndsWith(string value, System.StringCompar...
Equals   Method   bool Equals(System.Object obj), bool Equals(string value), bool Equals(string...
GetEnumerator Method   System.CharEnumerator GetEnumerator(), System.Collections.IEnumerable...
GetHashCode Method   int GetHashCode()
GetType   Method   type GetType()
GetTypeCode Method   System.TypeCode GetTypeCode(), System.TypeCode IConvertible.GetTypeCode()
IndexOf   Method   int IndexOf(char value), int IndexOf(char value, int startIndex), int IndexOf...
```

Es posible obtener información básica sobre el uso de los métodos escribiendo el nombre del objeto seguido del operador de acceso (.) y de nombre del método sin parentesis:



```
PS C:\WINDOWS\system32> $FechaHora.Substring

OverloadDefinitions
-----
string Substring(int startIndex)
string Substring(int startIndex, int length)
```

Para mayor información consulte la documentación de Powershell.

Conversión de tipos de datos

En ocasiones necesitaremos convertir explícitamente un tipo de dato a otro para su manipulación:

```
PS C:\WINDOWS\system32> $Raiz="256"
PS C:\WINDOWS\system32> [math]::sqrt($Raiz)
16
```

Podemos ver claramente que la operación raíz cuadrada se aplicó “automáticamente” al un valor de tipo cadena (String), no todas las conversiones se aplican de facto, por lo que sera necesario forzar a la conversión:

```
PS C:\WINDOWS\system32> $Dato="0x000D" (A)
PS C:\WINDOWS\system32> $Dato2="0x0002"
PS C:\WINDOWS\system32> $Dato2 + $Dato (B)
0x000213
PS C:\WINDOWS\system32> [byte]$Dato2 + $Dato (C)
15
PS C:\WINDOWS\system32> "{0:X0}" -f [byte]$Dato2 + $Dato (D)
213
```

Declaramos dos objetos (A) y (B), son del tipo cadenas (String), pero representan números hexadecimales deseamos conocer la suma de estos valores pero en (B) obtenemos un resultado inesperado al usar el operador suma, en (C) parece que tenemos el resultado correcto mostrado en decimal, un valor muy extraño nos otorga la expresión (D), el resultado que deseamos lo da la expresión:

```
PS C:\WINDOWS\system32> "{0:X0}" -f ([byte]$Dato2 + [byte]$Dato)
F
```

No debemos olvidar el uso de paréntesis para clarificar el orden de las operaciones aun en líneas como (C) y lo recomendable es asegurar que todos los objetos son del mismo tipo al realizar las operaciones, en resumen si vamos a concatenar que todos los objetos sean Cadenas o Caracteres, si necesitamos un valor numérico que todos los operandos pertenezcan a la misma base numérica.

Tuberías (PIPELINES)

El concepto de tubería consiste en redirigir la salida de un comando como entrada para otro comando, el símbolo ‘|’ es usualmente usado para designar una tubería, la cual luce como *comando | resultado-manipulación | OrdenaciónDeObjetos*, si hay necesidad de ordenar a elementos individuales dentro de la tubería se utiliza una variable de salida del tipo *\$_comando*, esto le indica a la tubería que evalúe los resultados de la tubería y sus atributos, la tubería ofrece una amplia variedad de usos; por ejemplo aprovecha la cualidad de *sort-object* para ordenar los objetos acorde a un atributo específico, *format-list* para formato de objetos en una lista, he incluso *select-object* también permite aprovechar los parámetros *-first* y *-last* junto a un número para indicar la cantidad de registros del conjunto a mostrar ya sea del inicio o del final de la tubería, otro popular comando es *where* el cual permite usar una expresión para seleccionar objetos que cumplan con ciertos criterios.

Ejemplo de uso:

```
PS C:\windows\system32> $services=Get-Service | where{$_.Name -like "*Event*"} | Sort-Object Name
PS C:\windows\system32> $services
Status   Name           DisplayName
-----   --
Running  AMD External Event... AMD External Events Utility
Running  EventLog        Registro de eventos de Windows
Running  EventSystem     Sistemas de eventos
```

Entender el funcionamiento de una tubería es más directo si utilizamos cada comando por separado uno a la vez:

```
PS C:\windows\system32> $services=Get-Service  
PS C:\windows\system32> $services ↓  
Status Name DisplayName  
---- -- --  
Stopped Aarsvc_6d92dfc Agent Activation Runtime_6d92dfc  
Stopped AJRouter Servicio de enrutador de AllJoyn  
Stopped ALG Servicio de puerta de enlace de niv...  
Running AMD Crash Defen... AMD Crash Defender Service  
Running AMD External Ev... AMD External Events Utility  
Stopped AppIDSvc Identidad de aplicación  
Running AppInfo Información de la aplicación  
Stopped AppReadiness Preparación de aplicaciones  
Running AppXSvc AppX Deployment Service (AppXSVC)
```

La variable de entorno \$services guarda en el formato por defecto tabla, el resultado de llamar al comando Get-Service, la lista está ordenada por defecto en orden alfabético acorde al atributo *Name*, la siguiente es algo “enredada”:

```
PS C:\windows\system32> $services=Get-Service | where{$_.Name -like "*Event*"}  
PS C:\windows\system32> $services ↓  
Status Name DisplayName  
---- -- --  
Running AMD External Ev... AMD External Events Utility  
Running *EventLog Registro de eventos de Windows  
Running EventSystem Sistema de eventos COM
```

La salida que se almacenó en el contenedor \$services, es utilizada por el comando *Where {<expresión>}*, donde *<expresión>* hace referencia al contenedor \$services mediante \$_ del cual evaluará los elementos del atributo *.Name* acorde al parámetro *-like* (“como” o “similar a”) “*<frase a comparar>*”, y a su vez *<frase a comparar>* utiliza “*” (Wild Cards) como comodines para indicar que puede haber cualquier número de caracteres antes de la *frase* y después de la misma también, el resultado se almacena en el contenedor \$services (variable de entorno), haciendo un pequeño cambio a la *frase* podemos apreciar mejor el orden en que se muestran los datos:

```
PS C:\windows\system32> $services=Get-Service | where{$_.Name -like "*E*n*t*"} | Sort-Object Name  
PS C:\windows\system32> $services ↓  
Status Name DisplayName  
---- -- --  
Running AMD External Ev... AMD External Events Utility  
Running AudioEndpointBu... Compilador de extremo de audio de W...  
Running BrokerInfrastru... Servicio de infraestructura de tare...  
Stopped ConsentUXUserSv... Servicio de usuario ConsentUX_6d92dfc  
Running CoreMessagingRe... CoreMessaging  
Stopped CredentialEnrol... CredentialEnrollmentManagerUserSvc_...  
Stopped DeviceInstall Servicio de instalación de dispositi...  
Running DisplayEnhancem... Servicio de mejora de visualización  
Stopped DmEnrollmentsSv... Servicio de inscripción de administ...  
Stopped EntAppsvc Servicio de administración de aplic...  
Running EventLog Registro de eventos de Windows  
Running EventSystem Sistema de eventos COM  
Stopped FrameServerMonitor Monitor del servidor de marco de la...
```

Se puede ver que efectivamente los registros del contenedor \$services ahora están ordenados en orden alfabético acorde al atributo *Name*.

También es posible usar las tuberías en un script, en la siguiente imagen podemos ver como cambia la presentación del formato de los datos de Tabla a Lista:

```

Untitled1.ps1* X
1 $largeFiles = get-childitem "c:\windows\system32\" | where{$_.Length -gt 20MB}
2 $count = $largeFiles.count
3 Write-host "There are $count Files over 20MB"
4 write-host "Files Over 20MB in c:\Windows\System32\ :"
5 $largefiles | select-object name,length,lastwritetime | format-table ←

```

```

PS C:\windows\system32> $largeFiles = get-childitem "c:\windows\system32\" | where{$_.Length -gt 20MB}
$Count = $largeFiles.Count
Write-host "There are $Count Files over 20MB"
Write-host "Files Over 20MB in c:\Windows\System32\ :"
$largefiles | select-object name,length,lastwritetime | format-table
There are 8 Files over 20MB
Files Over 20MB in c:\Windows\System32\ :

Name          Length LastWriteTime
----          ----- -----------
amdx64.so     110177296 22/04/2024 03:44:00 a. m.
and_comgr.dll 105400944 22/04/2024 04:21:16 a. m.
DXCaptureReplay.dll 29556736 30/04/2024 05:54:39 p. m.
edgehtml1.dll 27082752 07/11/2024 08:45:08 p. m.
Hydrogen.dll 25260032 07/05/2022 01:10:25 a. m.
MRT.exe       202035632 13/11/2024 06:11:36 a. m.
mshtm1.dll   23150592 13/11/2024 04:57:55 a. m.
OneDriveSetup.exe 50312608 07/05/2022 01:10:24 a. m.

PS C:\windows\system32> $largefiles | select-object name,length,lastwritetime | format-list ←

```

```

Name      : amdx64.so
Length    : 110177296
LastWriteTime : 22/04/2024 03:44:00 a. m.

Name      : and_comgr.dll
Length    : 105400944
LastWriteTime : 22/04/2024 04:21:16 a. m.

Name      : DXCaptureReplay.dll
Length    : 29556736
LastWriteTime : 30/04/2024 05:54:39 p. m.

Name      : edgehtml1.dll
Length    : 27082752
LastWriteTime : 07/11/2024 08:45:08 p. m.

Name      : Hydrogen.dll
Length    : 25260032
LastWriteTime : 07/05/2022 01:10:25 a. m.

```

Esto es útil para darle al usuario del script la opción de cambiar la forma de la presentación de los datos a una que mejor se adapte a sus necesidades.

Edición de ventanas simples

Existen dos formas de crear Ventanas para Powershell usar un editor para crear nuestra forma y exportarlo como un archivo XML y agregar en otro archivo el código de la lógica de los controles de la forma. La otra opción es usar la librería .NET Windows Forms (WinForms), usaremos esta última ya que se puede codificar directamente en un script de Powershell.

Ventana simple

El proceso de crear una ventana requiere seguir una lista de pasos (Observe la figura de abajo):

1. Crear un Script y cargar Windows Forms Assembly
línea 2: `Add-Type -assembly System.Windows.Forms`
2. Crear una ventana de diálogo con sus parámetros básicos
línea 4: `$Ventana_Principal=New-Object System.Windows.Forms.Form`
línea 8 a 9: `$Ventana_Principal.Text='GUI para mi Script de Powershell'`
 `$Ventana_Principal.Width=600`
 `$Ventana_Principal.Height=400`

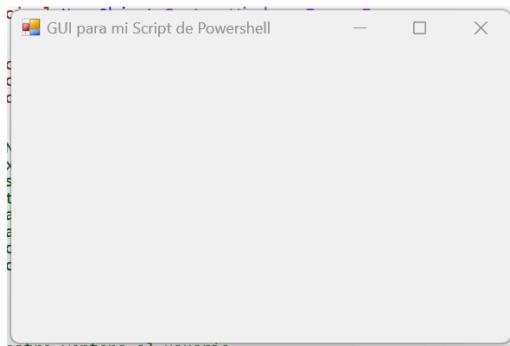
Podemos agregar en esta sección los demás elementos gráficos a la ventana, como etiquetas, botones o cuadros de diálogo por ejemplo.

3. Agregar los manejadores de los eventos por ejemplo botones
línea X: *no implementado en este ejemplo*
4. Mostrar la ventana al usuario
línea 13: `$Ventana_Principal.ShowDialog()`

Se muestra el código de ejemplo del listado anterior:

```
1 #Agregamo la libreria .Net para las formas (WinForms)
2 Add-Type -assembly System.Windows.Forms
3
4 #Creamos un objeto del tipo forma (formulario)
5 $Ventana_Principal=New-Object System.Windows.Forms.Form
6
7 #Configuramos la apariencia de nuestra Ventana
8 $Ventana_Principal.Text='GUI para mi Script de Powershell' #Titulo de la Ventana
9 $Ventana_Principal.Width=600 #Definimos el Ancho de la ventana
10 $Ventana_Principal.Height=400 #Así como la Altura de la misma
11
12 <#
13     Aquí se colocan los manejadores de los eventos, los cuales pueden
14     ser funciones que se encuentren en otro Script para facilitar
15     la lectura del código de la ventana
16 #>
17
18 #Mostramos nuestra ventana al usuario
19 $Ventana_Principal.ShowDialog()
```

La ejecución del Script anterior genera la siguiente ventana:



Etiqueta con formato simple

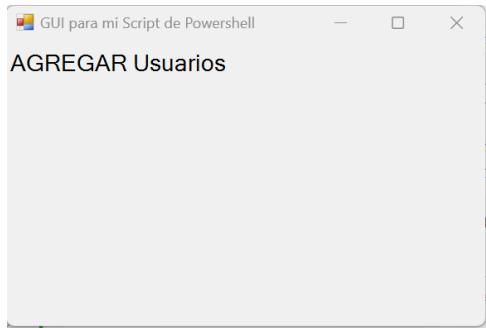
Seguido de la definición de la Ventana podemos agregar los elementos que deseamos incluir en la misma, así como los parámetros de apariencia. El código siguiente muestra en las líneas 13 a 19 la definición de una Etiqueta y en la línea 20 la instrucción para incluirla en la Ventana .

```

1 #Agregamos la libreria .Net para las formas (WinForms)
2 Add-Type -assembly System.Windows.Forms
3
4 #Creamos un objeto del tipo forma (formulario)
5 $Ventana_Principal=New-Object System.Windows.Forms.Form
6
7 #Configuramos la apariencia de nuestra Ventana
8 $Ventana_Principal.Text='GUI para mi Script de Powershell' #Titulo de la Ventana
9 $Ventana_Principal.Width=600 #Definimos el Ancho de la ventana
10 $Ventana_Principal.Height=400 #Así como la Altura de la misma
11
12 <# Sección de elementos básicos #>
13 $Etiquetal = New-Object System.Windows.Forms.Label #Creamos un objeto de tipo Etiqueta
14 $Etiquetal.Text = "AGREGAR Usuarios" #Escribimos en la Etiqueta
15 # Configuramos los parametros del Texto de la Etiqueta
16 $Etiquetal.Font = 'Microsoft Sans Serif,14'
17 # Indicamos la posición del Texto dentro de nuestra Etiqueta
18 $Etiquetal.Location = New-Object System.Drawing.Point(0,10)
19 $Etiquetal.AutoSize = $true #Habilitamos la opción para que la Etiqueta pueda ajustar automaticamente su tamaño
20 $Ventana_Principal.Controls.Add($Etiquetal) #Agregamos nuestra Etiqueta a la Ventana
21
22 <# Sección de manejadores de eventos #>
23
24
25 #Mostramos nuestra ventana al usuario
26 $Ventana_Principal.ShowDialog()

```

El resultado de la ejecución de este script se muestra en la siguiente figura:



Menú desplegable

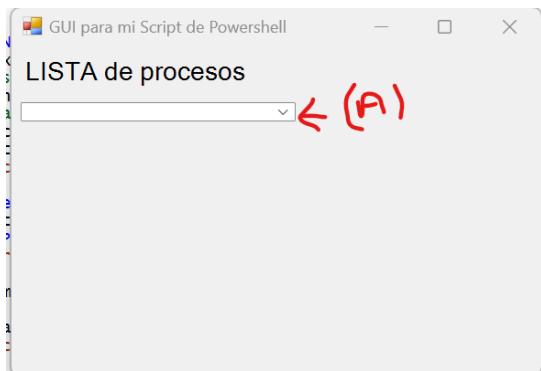
Seguido de la definición de la Ventana podemos agregar los elementos que deseamos incluir en la misma, así como los parámetros de apariencia. El código siguiente muestra en las líneas 13 a 19 la definición de una Etiqueta y en la línea 20 la instrucción para incluirla en la Ventana .

```

11 <# Sección de elementos básicos #>
12 $Etiquetal = New-Object System.Windows.Forms.Label #Creamos un objeto de tipo Etiqueta
13 $Etiquetal.Text = "LISTA de procesos" #Escribimos en la Etiqueta
14 # Configuramos los parametros del Texto de la Etiqueta
15 $Etiquetal.Font = 'Microsoft Sans Serif,14'
16 # Indicamos la posición del Texto dentro de nuestra Etiqueta
17 $Etiquetal.Location = New-Object System.Drawing.Point(10,10)
18 $Etiquetal.AutoSize = $true #Habilitamos la opción para que la Etiqueta pueda ajustar automaticamente su tamaño
19 $Ventana_Principal.Controls.Add($Etiquetal) #Agregamos nuestra Etiqueta a la Ventana
20
21 $ComboBox = New-Object System.Windows.Forms.ComboBox
22 $ComboBox.Width = 300
23 $Users = Get-Process | Select Name -First 10 #Query para listar los primeros 10 procesos solo por el nombre
24 Foreach ($User in $Users)
25 {
26     $ComboBox.Items.Add($User.Name);
27 }
28 $ComboBox.Location = New-Object System.Drawing.Point(10,60)
29 $Ventana_Principal.Controls.Add($ComboBox)
30
31 <# Sección de manejadores de eventos #>
32
33
34
35 #Mostramos nuestra ventana al usuario
36 $Ventana_Principal.ShowDialog()

```

Las modificaciones a la Ventana con este script se muestra en la siguiente figura:



Usar la pestaña (A) muestra una lista con los nombres de los primeros 10 procesos en ejecución.

Botón y el evento Click()

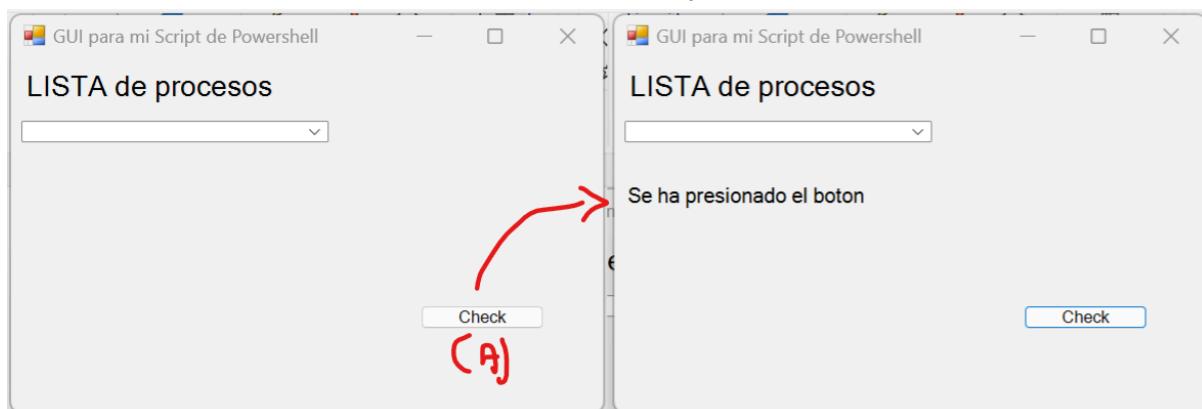
Creamos en las líneas 32 a la 36 un botón y lo agregamos a la ventana en la línea 37, en las líneas 47 a 51 agregamos lo que deseamos que haga ese botón al ser presionado.

```

52
33 $Button = New-Object System.Windows.Forms.Button
34 $Button.Location = New-Object System.Drawing.Size(400,240)
35 $Button.Size = New-Object System.Drawing.Size(120,23)
36 $Button.Text = "Check"
37 $Ventana_Principal.Controls.Add($Button)
38
39 $Etiqueta2 = New-Object System.Windows.Forms.Label
40 $Etiqueta2.Text = ""
41 $Etiqueta2.Location = New-Object System.Drawing.Point(10,120)
42 $Etiqueta2.AutoSize = $true
43 $Etiqueta2.Font = 'Microsoft Sans Serif,10'
44 $Ventana_Principal.Controls.Add($Etiqueta2)
45
46 <# Sección de manejadores de eventos #>
47 $Button.Add_Click(
48 {
49     $Etiqueta2.Text = 'Se ha presionado el boton'
50 }
51 )
52
53 #Mostramos nuestra ventana al usuario
54 $Ventana_Principal.ShowDialog()

```

Podemos apreciar la apariencia de la ventana antes y después de presionar el botón (A).



Un poco más sobre los operadores de comparación

En la sección de Scripting se mostraron indirectamente el uso de los operadores, los cuales no solo funcionan con números, también es posible comparar contenedores de hora y fecha, como muestra el siguiente ejemplo:

```
Untitled1.ps1* 
1 $fechavieja = Get-Date
2 Start-Sleep -seconds 2
3 $fechanueva= Get-Date
4 If ($fechanueva -gt $fechavieja) { Write-Host "Valor $fechanueva es mayor que $fechavieja" }

PS C:\windows\system32> $fechavieja = Get-Date
Start-Sleep -seconds 2
$fechanueva= Get-Date
If ($fechanueva -gt $fechavieja) { Write-Host "Valor $fechanueva es mayor que $fechavieja" }
Valor 11/22/2024 02:41:56 es mayor que 11/22/2024 02:41:54
```

Es bastante útil para determinar si hay diferencias de tiempo entre eventos, por ejemplo para determinar si es necesario actualizar algún proceso.

Aparte de los operadores de comparación usuales, contamos con algunos que permiten buscar elementos al momento de evaluar una expresión, estos operadores son:

-*contains* “<expresión>” busca por valores exactos en la expresión

```
Untitled1.ps1* 
1 $miarreglo = "Este", "es", "mi", "arreglo"
2 If ($miarreglo -contains "Este") { Write-Host "El arreglo contiene la palabra: Esto" }
3 If ($miarreglo -notcontains "eso") { Write-Host "El arreglo no contiene la palabra: eso" }

PS C:\windows\system32> $miarreglo = "Este", "es", "mi", "arreglo"
If ($miarreglo -contains "Este") { Write-Host "El arreglo contiene la palabra: Esto" }
If ($miarreglo -notcontains "eso") { Write-Host "El arreglo no contiene la palabra: eso" }
El arreglo contiene la palabra: Esto
El arreglo no contiene la palabra: eso
```

En la imagen podemos ver que -*contains* permite la comparación entre elementos, no solo el contenido completo del contenedor.

-*like* “<WildCard><expresión><WildCard>” permite evaluar caracteres antes y después de la expresión buscada, usado comúnmente con Wild Cards (“*” múltiples caracteres, “?” un solo carácter)

```
Untitled1.ps1* 
1 $miEjemplo = "Esto es un ejemplo de PowerShell."
2 Write-Host " " #salto de linea
3 Write-Host "-----Ejecucion----- "
4 If ($miEjemplo -like "*shell*") { Write-Host "La variable contiene una palabra
5 que es igual a shell" }
6 Write-Host " " #salto de linea
7 If ($miEjemplo -notlike "*Aquellos*") { Write-Host "La variable no contiene una palabra
8 una palabra que es igual a *Aquellos*" }

-----Ejecucion-----
If ($miEjemplo -like "*shell*") { Write-Host "La variable contiene una palabra
que es igual a shell" }
Write-Host " " #salto de linea
If ($miEjemplo -notlike "*Aquellos*") { Write-Host "La variable no contiene una palabra
una palabra que es igual a *Aquellos*" }

-----Ejecucion-----
La variable contiene una palabra
que es igual a shell

La variable no contiene una palabra
una palabra que es igual a *Aquellos*
```

En el ejemplo se ignoró completamente el hecho de que la coincidencia estaba en la palabra: PowerShell

-match “<regular expresión>” las “expresiones regulares” son cadenas formadas con operadores y contenedores, por ejemplo [“Hola \$NombreUsuario ” + \$Apellido]

```
Untitled1.ps1* X
1 $miejemplo = "La red ha caido."
2 Write-Host "" #salto de linea
3 If ($miejemplo -match "[a]") { Write-Host "La variable contiene a la palabra a. (Contiene tres a's)" }
4 Write-Host "" #salto de linea
5 Write-Host "----Ejecución----"
6 $matches
7 If ($miejemplo -notmatch "[U]") { Write-Host "La variable no contiene a la palabra a U. (No contiene a U)" }

Write-Host "----Ejecución----"
$matches
If ($miejemplo -notmatch "[U]") { Write-Host "La variable no contiene a la palabra a U. (No contiene a U)" }

La variable contiene a la palabra a. (Contiene tres a's)
----Ejecución----

Name           Value
---           ---
0             a
La variable no contiene a la palabra a U. (No contiene a U)
```

El uso de *-match* se verá más claramente en el futuro.

Podemos apreciar que ninguno de los operadores es “sensible a mayúsculas” por defecto, si deseamos agregar la opción debemos agregar una “c” al inicio de cada operador:

-ccontains -clike y *-cmatch*, para enfatizar la opción por defecto usamos *-icontains -ilike* *-imatch* esto para cuando queramos más claridad.

Nota: para evitar el “echo” de las instrucciones en pantalla, guarde los scripts como se indica en la sección de Shell Scripting casi al inicio de este documento.

Administración remota con Powershell

Trataremos sobre la ejecución de comandos en sistemas remotos (Linux o Windows) y la apertura y configuración de sesiones en sistemas remotos, para poder ejecutar varios comandos de manera paralela y esto nos permitirá automatizar procesos en varias máquinas a la vez. Partiremos en un ambiente con una conexión entre sistemas ya establecidas para información sobre cómo configurar el sistema para administración remota mediante Powershell consulte el **Anexo I**.

Comunicación Cliente-Servidor 1 a 1 (Linux o Windows)

Comenzamos la comunicación de nuestra máquina cliente hacia una sola máquina remota servidor, algunos cmdlets por sí solos cuentan con el parámetro que les permite ejecutarse en un sistema remoto, por ejemplo:

```
PS C:\Windows\System32> Get-Help Invoke-Command
NAME
    Invoke-Command

SYNTAX
    Invoke-Command [-ScriptBlock] <scriptblock> [-NoNewScope] [-InputObject <psobject>] [-ArgumentList <Object[]>] [<CommonParameters>]
    Invoke-Command [[-Session] <PSSession[]>] [-ScriptBlock] <scriptblock> [-ThrottleLimit <int>] [-AsJob] [-HideComputerName] [-JobName <string>] [-RemoteDebug] [-InputObject <psobject>] [-ArgumentList <Object[]>] [<CommonParameters>]
    Invoke-Command [[-Session] <PSSession[]>] [-FilePath] <string> [-ThrottleLimit <int>] [-AsJob] [-HideComputerName] [-JobName <string>] [-RemoteDebug] [-InputObject <psobject>] [-ArgumentList <Object[]>] [<CommonParameters>]
    Invoke-Command [[-ComputerName] <string[]>] [-ScriptBlock] <scriptblock> [-Credential <pscredential>] [-Port <int>] [-UseSSL] [-ConfigurationName <string>] [-ApplicationName <string>] [-ThrottleLimit <int>] [-AsJob] [-InDisconnectedSession] [-SessionName <string[]>] [-HideComputerName] [-JobName <string>] [-SessionOption <PSSessionOption>] [-Authentication {Default | Basic | Negotiate | NegotiateWithImplicitCredential | Credssp | Digest | Kerberos}] [-EnableNetworkAccess] [-RemoteDebug] [-InputObject <psobject>] [-ArgumentList <Object[]>] [-CertificateThumbprint <string>] [<CommonParameters>]
    Invoke-Command [[-ComputerName] <string[]>] [-FilePath] <string> [-Credential <pscredential>] [-Port <int>] [-UseSSL] [-ConfigurationName <string>] [-ApplicationName <string>] [-ThrottleLimit <int>] [-AsJob] [-InDisconnectedSession] [-SessionName <string[]>] [-HideComputerName] [-JobName <string>] [-SessionOption <PSSessionOption>] [-Authentication {Default | Basic | Negotiate | NegotiateWithImplicitCredential | Credssp | Digest | Kerberos}] [-EnableNetworkAccess] [-RemoteDebug] [-InputObject <psobject>] [-ArgumentList <Object[]>] [<CommonParameters>]
    Invoke-Command [[-ConnectionUri] <uri[]>] [-ScriptBlock] <scriptblock> [-Credential <pscredential>] [-ConfigurationName <string>] [-ThrottleLimit <int>] [-AsJob] [-InDisconnectedSession] [-HideComputerName] [-JobName <string>] [-AllowRedirection] [-SessionOption <PSSessionOption>] [-Authentication {Default | Basic | Negotiate | NegotiateWithImplicitCredential | Credssp | Digest | Kerberos}]
```

El cmdlet *Invoke-Command* cuenta con el parámetro *-ComputerName* el cual indica cuál máquina remota debe ejecutarse el comando, otros parámetros como *-Session* o *-HostName* también tienen el mismo efecto, solo que encuentran a la máquina remota mediante otra definición. Aprovechando mencionaremos que este comando (*Invoke-Command*) permite la ejecución de Cmdlet en sistemas remotos que no cuentan por sí solos con la propiedad de ejecutarse remotamente, por ejemplo el Cmdlet *Get-Date* al ejecutarse nos entrega la hora y fecha del sistema en el que tenemos la ventana de Powershell como muestra (A) en la siguiente figura:

```
PS C:\Windows\System32> Get-Date (A)
miércoles, 27 de noviembre de 2024 05:38:47 a. m.

PS C:\Windows\System32> invoke-Command -HostName raspberrypi -UserName rodrigo -ScriptBlock {Get-Date}
rodrigo@raspberrypi's password: (B)

miércoles, 27 de noviembre de 2024 05:39:07 a. m.

PS C:\Windows\System32>
```

Para su ejecución en un sistema remoto debemos incluirlo en la instrucción:

invoke-Command -HostName raspberrypi -UserName rodrigo -ScriptBlock {Get-Date}

Como lo indica (B), podemos ver que entre los símbolos de llaves "{}" va el comando que deseamos ejecutar en el sistema remoto (en este caso Get-Date), y no solo puede ir un solo comando podemos incluir varias instrucciones para que se ejecuten en el sistema remoto, como lo muestra la figura siguiente:

```
PS C:\Windows\System32> Get-Command | Measure-Object; Get-Date | Format-Table Local
Count : 1684
Average :
Sum :
Maximum :
Minimum :
StandardDeviation :
Property :

DisplayHint Date          Day DayOfWeek DayOfYear Hour Kind Millisecond Microsecond Nanosecond
----- ----          -- -- -- -- -- -- -- -- -- --
DateTime 27/11/2024 12:00:00 a. m. 27 Wednesday      332   5 Local        298       740      100

PS C:\Windows\System32> invoke-Command -HostName raspberrypi -UserName rodrigo -ScriptBlock {Get-Command | Measure-Object; Get-Date | Format-Table}
rodrigo@raspberrypi's password: Remoto
Count : 270
Average :
Sum :
Maximum :
Minimum :
Property :
PSComputerName : raspberrypi

DisplayHint Date          Day DayOfWeek DayOfYear Hour Kind Millisecond Minute Month
----- ----          -- -- -- -- -- -- -- --
DateTime 27/11/2024 0:00:00 27 Wednesday      332   5 Local        623      53       11

PS C:\Windows\System32>
```

Las diferencias ahora son más palpables, podemos ver que el sistema local cuenta con muchos más comandos para ejecutar que el sistema remoto, lo cual limitará cuanto se puede automatizar. Se puede observar que para cada ejecución en el sistema remoto se debe de incluir las credenciales lo cual será tedioso cuando debamos ingresar varios comandos, un método mejor consiste en ingresar en una sesión interactiva usando el comando:

Enter-PSSession -HostName raspberrypi -Username rodrigo -SSHTransport

Como muestra la siguiente figura:

A screenshot of a Windows PowerShell window. The session starts with `Enter-PSSession -HostName raspberrypi -Username rodrigo -SSHTransport`. A red arrow labeled (A) points to the password prompt. The user enters their password. Then, the command `Get-Process | Measure-Object | format-table` is run, indicated by a red arrow (B). The output shows a table of process statistics. Finally, the command `Get-Date | format-table` is run, indicated by a red arrow (C), showing the current date and time. A red bracket (D) groups the three arrows (A), (B), and (C).

```
PS C:\Users\sambo> Enter-PSSession -HostName raspberrypi -Username rodrigo -SSHTransport
rodrigo@raspberrypi's password: (A)
[rodrigo@raspberrypi]: PS /home/rodrigo> Get-Process | Measure-Object | format-table
Count Average Sum Maximum Minimum StandardDeviation Property
-----
166
[rodrigo@raspberrypi]: PS /home/rodrigo> Get-Date | format-table (C)
DisplayHint Date Day DayOfWeek DayOfYear Hour Kind Millisecond Minute Month
----- -----
DateTime 27/11/2024 0:00:00 27 Wednesday 332 7 Local 758 46 11
```

Podemos ver en (A) que solo necesitamos ingresar el Password una sola vez para iniciar una sesión interactiva en el sistema remoto, note que estamos en un CLI de Powershell, como muestran las instrucciones en (C) podemos entonces ingresar varias instrucciones y obtendremos el resultado sin que se nos pida el password en cada ejecución, en cierto sentido es igual a una sesión SSH típica de un sistema Linux, para terminar la interacción tecleamos *exit*.

La desventaja de usar sesiones interactivas o invoke es que la ejecución de instrucciones es en serie, es decir, hasta que no se responde el sistema remoto, no se puede avanzar a la siguiente instrucción, combinando los dos métodos es posible ejecutar los mismos comandos de manera paralela en varios sistemas remotos a la vez, esto se logra creando contenedores con los cuales interactuar, la siguiente figura muestra cómo hacerlo:

A screenshot of a Windows PowerShell window. It starts with `New-PSSession -HostName raspberrypi -UserName rodrigo -SSHTransport`, followed by a password prompt (A). The session object is stored in the variable `\$session` (B). The command `Invoke-Command -Session \$session -ScriptBlock {Get-Date}` is run, showing the date (C). Finally, `Invoke-Command -Session \$session -ScriptBlock {Get-Process | Measure-Object | Format-table}` is run, showing a table of process statistics. A red bracket (D) groups the three arrows (A), (B), and (C).

```
PS C:\Users\sambo> $session = New-PSSession -HostName raspberrypi -UserName rodrigo -SSHTransport
rodrigo@raspberrypi's password: (A)
PS C:\Users\sambo> $session (B)
Id Name Transport ComputerName ComputerType State ConfigurationName Availability
-- -- -- -- -- -- -- --
2 Runspace1 SSH raspberrypi RemoteMachine Opened DefaultShell Available

PS C:\Users\sambo> Invoke-Command -Session $session -ScriptBlock {Get-Date} (C)
miércoles, 27 de noviembre de 2024 08:05:16 a. m.

PS C:\Users\sambo> Invoke-Command -Session $session -ScriptBlock {Get-Process | Measure-Object | Format-table}

Count Average Sum Maximum Minimum StandardDeviation Property
-----
160
PS C:\Users\sambo> HostName (D)
```

En (A) creamos el contenedor con el comando *New-PSSession* (nos pide la contraseña solo una vez), el contenido de *\$session* se muestra en (B) atención especial a los atributos *Transport State Availability* que indican el tipo de protocolo usado para la conexión, el estado de la misma y si esta disponible a recibir comandos de Powershell (podría estar ocupada ejecutando procesos en segundo plano), podemos ver que al enviar comandos en (C) no nos pide nuevamente la clave de acceso, al final podemos constatar en (D) que seguimos en la máquina local.

A diferencia de un sistema Linux ejecutando Powershell, aparte del método de conexión por el protocolo SSH, existe el método de comunicación es mediante WinRM (creado por Microsoft), el cual requiere ser configurado en todas las máquinas que recibirán los comandos remotamente. Consulte el Anexo I para información sobre realizar la conexión. *Comunicación Cliente-Servidores 1 a Varios (Linux o Windows)*
 Requiere que establezcamos inicios de sesión previamente y los guardemos en un contenedor que ahora contendrá una lista de sesiones en vez de solo una.
 (construcción)

Anexo A

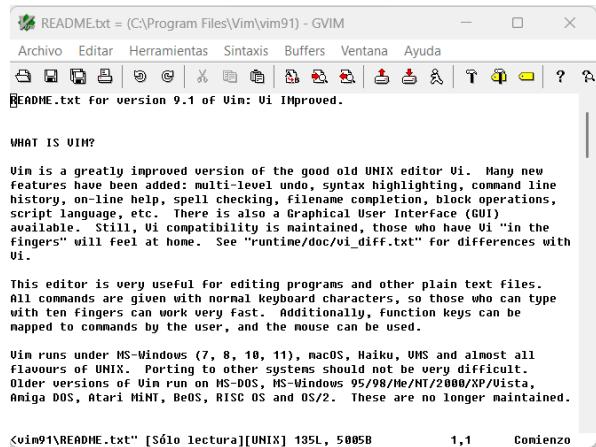
Instalación de Vim

En el explorador busque la página del proyecto VIM:

En la sección de descargas (Download), elija la opción más adecuada para su sistema:

Se muestra en (A) las opciones para sistemas de 32 y 64 bits respectivamente, este instalador colocará una versión GUI para usar el editor en Windows, aunque una instalación mínima es suficiente se recomienda la versión completa si se cuenta con espacio suficiente.

Se preguntará al final si se desea leer el archivo “readme” para conocer detalles sobre la versión que se ha instalado, puede ser una buena oportunidad para ver la GUI del editor:



Es posible utilizar la versión de Vim en la ventana de CMD y Powershell de Windows, en dichas ventanas simplemente teclee *vim* y con eso invocará al editor:



Listo ya puede usar Vim en sus terminales.

Anexo B

Instalación de MinGW

En el explorador buscamos MinGW y elegimos la opción de SourceForge.net:

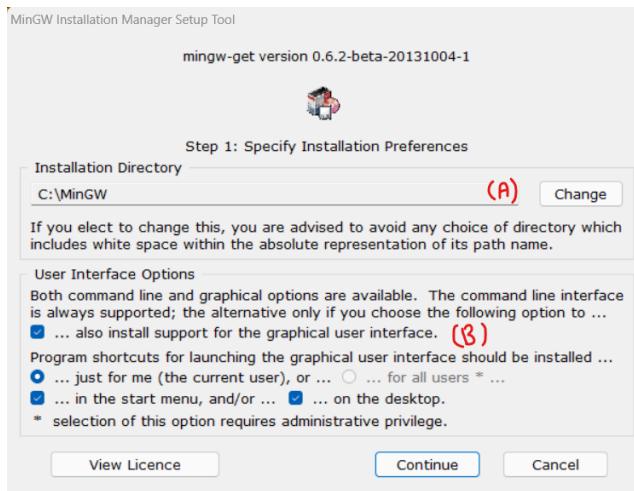
[MinGW - Minimalist GNU for Windows download | SourceForge.net](https://sourceforge.net/projects/mingw/)

A screenshot of a Google search results page. The search query "mingw" is entered in the search bar. The results show a link to the SourceForge project page for MinGW. The page title is "SourceForge - MinGW - Minimalist GNU for Windows download". The page content includes a brief description of MinGW as a native Windows port of the GNU Compiler Collection (GCC), with freely distributable import libraries and header files for building native Windows ... It also shows a rating of 4.2 stars from 210 reviews, and links to the installer and browse pages.

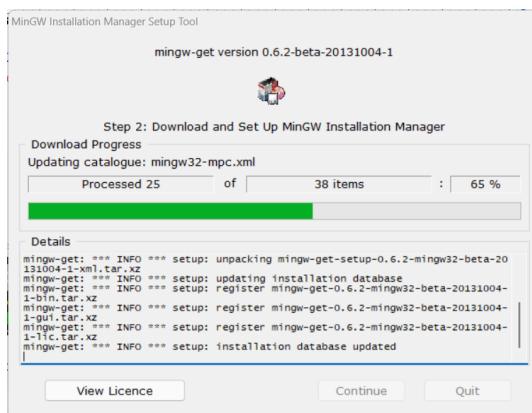
Descargamos el Instalador y lo ejecutamos en con permisos de administrador:



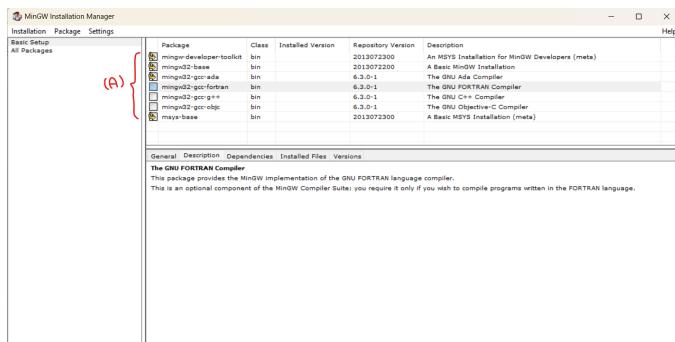
La siguiente figura muestra información sobre la instalación, (A) indica el directorio donde se estará el programa, (B) si queremos instalar la GUI o solo el conjunto de programas para el CLI (terminal).



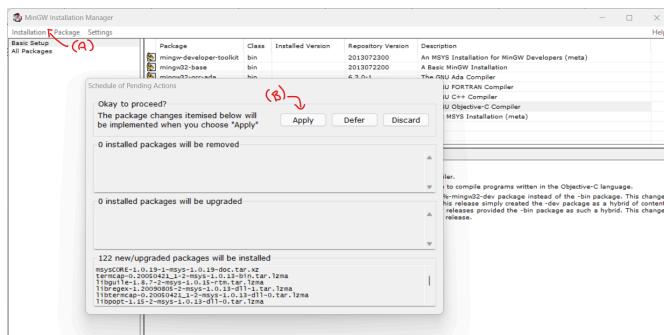
Comenzará a descargar los archivos de la instalación y los depositara en la carpeta antes mencionada:



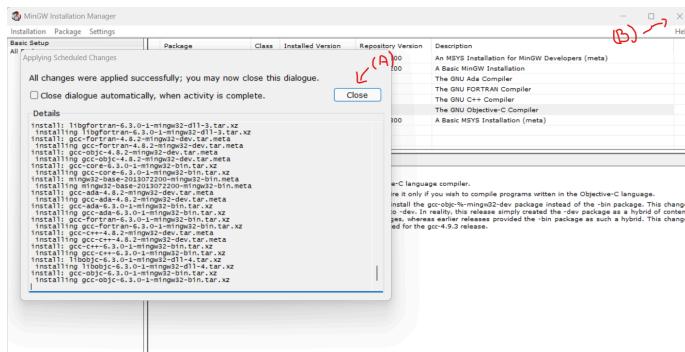
Al terminar nos pedirá que continuemos y nos indicará que debemos marcar los paquetes que instalarán (A), se recomienda elegirlos todos:



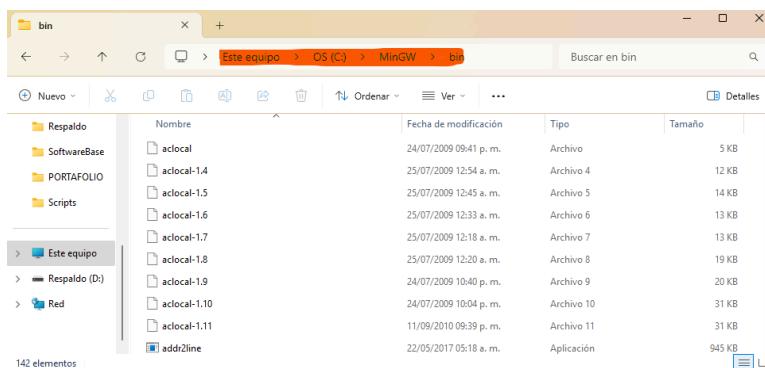
En el menú (A) elegimos la opción “apply changes” y en la ventana (B) presionamos el botón “Apply”, esperamos que termine la instalación:



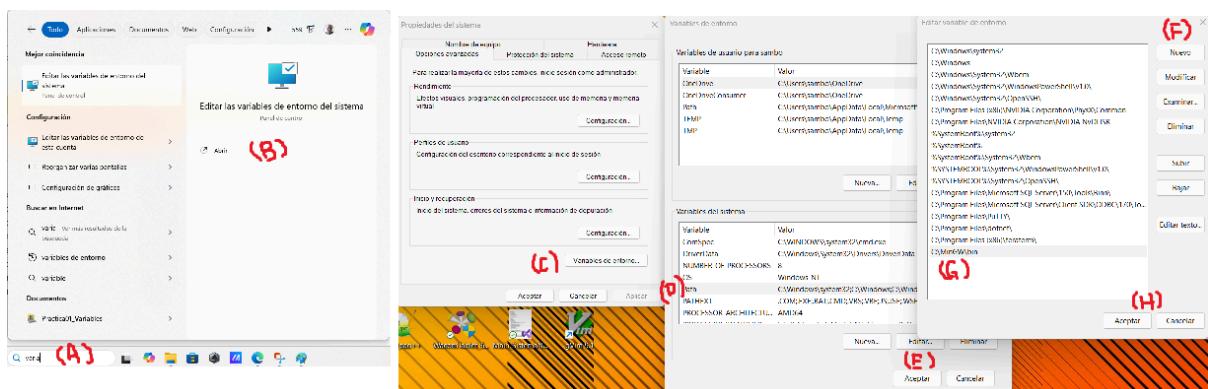
Una vez terminada la instalación cerramos el cuadro de diálogo (A) y la ventana (B):



En la ventana del Explorador de Archivos, nos ubicamos en la dirección donde se instalarón los archivos (indicada al inicio de la instalación), dentro de la carpeta *bin*:

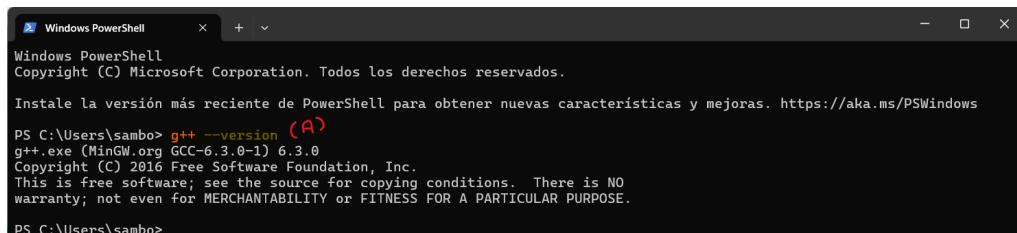


Y copiamos la dirección al portapapeles (clic con el botón derecho del ratón), esto para agregarla a la dirección de búsqueda de programas ejecutables en las variables de entorno de Windows:



En el cuadro de búsqueda de Windows, escribimos *Variables de Ambiente* (A), ejecutamos la aplicación (B), presionamos el botón *Variables de Entorno* (C), elegimos del cuadro de texto la opción *Path* (D), en la ventana el botón (F) y en (G) pegamos la dirección que copiamos del explorador de Windows y presionamos el botón (H) para terminar (damos click en Aceptar en las ventanas anteriores).

En la terminal de PowerShell tecleamos el comando `g++ --version`, para verificar la instalación:



```
Windows PowerShell
Copyright (C) Microsoft Corporation. Todos los derechos reservados.

Instale la versión más reciente de PowerShell para obtener nuevas características y mejoras. https://aka.ms/PSWindows
PS C:\Users\sambo> g++ --version (A)
g++.exe (MinGW.org GCC-6.3.0-1) 6.3.0
Copyright (C) 2016 Free Software Foundation, Inc.
This is free software; see the source for copying conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

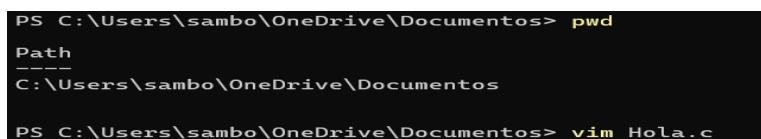
PS C:\Users\sambo>
```

Si nos indica la versión entonces hemos instalado el software correctamente.

Anexo C

Codificando y Compilando con Vim y MinGW en la terminal

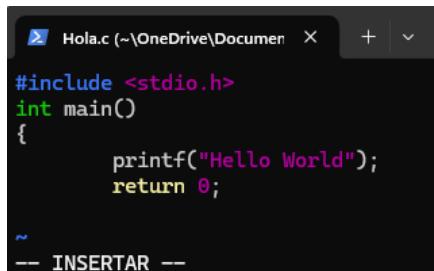
Nos movemos a una carpeta con permisos de edición y lanzamos el comando `vim <nombre>.c`



```
PS C:\Users\sambo\OneDrive\Documentos> pwd
Path
-----
C:\Users\sambo\OneDrive\Documentos

PS C:\Users\sambo\OneDrive\Documentos> vim Hola.c
```

Y escribimos el siguiente código:

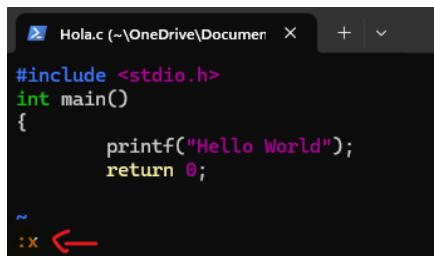


```
#include <stdio.h>
int main()
{
    printf("Hello World");
    return 0;
}

-- INSERTAR --
```

Para iniciar la edición presione la tecla 'i' y verifique que el la parte inferior indique INSERTAR.

Para salir y grabar presione la tecla 'esc' y cuando la barra inferior cambie, escriba ':x'



```
#include <stdio.h>
int main()
{
    printf("Hello World");
    return 0;
}

:x ↵
```

Presione la tecla Enter para terminar, en la Terminal de PowerShell escriba lo siguiente para compilar el código `gcc hola.c -o hola.exe`, después verifique la creación del archivo usando `ls Hola*` (o `dir Hola*`), ejecute el programa con `.\Hola.exe` :

```

Administrator: Windows PowerShell
PS C:\Users\sambo\OneDrive\Documentos> gcc Hola.c -o Hola.exe
PS C:\Users\sambo\OneDrive\Documentos> ls Hola*
Directorio: C:\Users\sambo\OneDrive\Documentos

Mode                LastWriteTime        Length Name
----                -----          ----  --
-a---l    02/11/2024 02:27 a. m.      75 Hola.c
-a---l    02/11/2024 02:31 a. m.  40766 Hola.exe

PS C:\Users\sambo\OneDrive\Documentos> .\Hola.exe
Hello World
PS C:\Users\sambo\OneDrive\Documentos>

```

Nota: no olvide ingresar a la terminal de PowerShell como administrador
 Para compilar programas codificados en lenguaje C++ escriba un archivo en Vim
Vim Hola.cpp y escriba lo siguiente:

```

Hola.cpp (~\OneDrive\Documentos) - VIM
#include <iostream>
using namespace std;
int main()
{
    cout << "Hola mundo" << endl;
    return 0;
}

```

Compile, liste y ejecute el archivo generado como se muestra en la figura:

```

Administrator: Windows PowerShell
PS C:\Users\sambo\OneDrive\Documentos> vim Hola.cpp
PS C:\Users\sambo\OneDrive\Documentos> g++ Hola.cpp -o Hola_C.exe
PS C:\Users\sambo\OneDrive\Documentos> dir Hola*
Directorio: C:\Users\sambo\OneDrive\Documentos

Mode                LastWriteTime        Length Name
----                -----          ----  --
-a---l    02/11/2024 02:27 a. m.      75 Hola.c
-a---l    02/11/2024 02:39 a. m.     193 Hola.cpp
-a---l    02/11/2024 02:31 a. m.  40766 Hola.exe
-a---l    02/11/2024 02:45 a. m.  44550 Hola_C.exe
-a---l    02/11/2024 02:40 a. m.  44550 Hola_Cmas.exe

PS C:\Users\sambo\OneDrive\Documentos> .\Hola_C.exe
Hola mundo
PS C:\Users\sambo\OneDrive\Documentos>

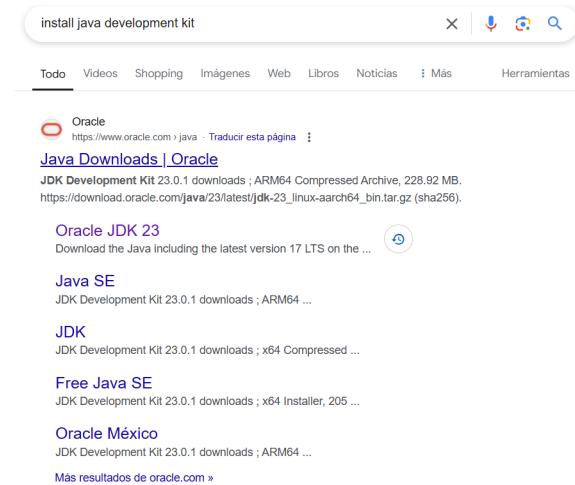
```

Note que esta vez se utilizó *dir Hola** para listar los archivos en lugar de *ls Hola**, ambos comandos realizan la misma función.

Anexo D

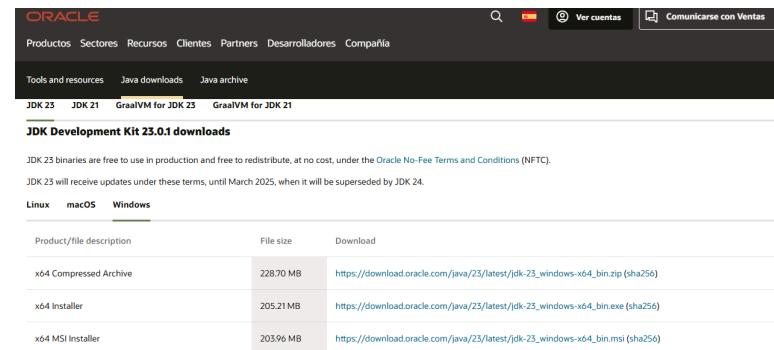
Instalando JAVA

Descargue el conjunto de herramientas para desarrolladores de la página de Oracle en su explorador escriba Instalar Java Development Kit



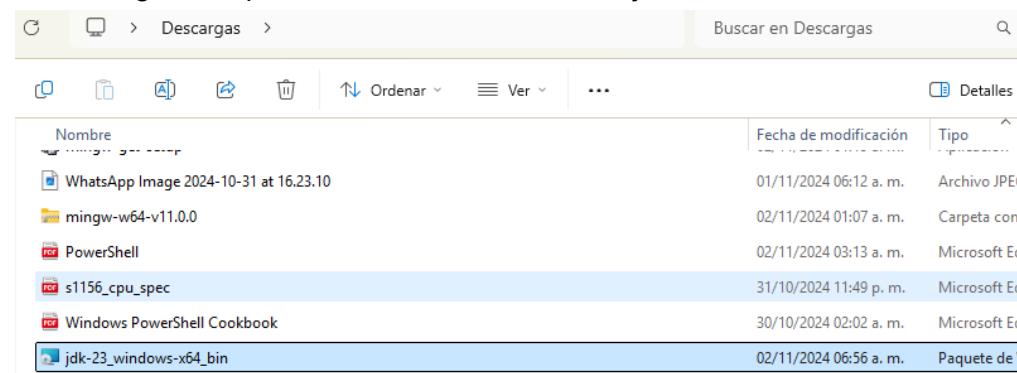
The screenshot shows the Oracle Java Downloads page. At the top, there's a search bar with the query "install java development kit". Below the search bar, a navigation bar includes links for "Todo", "Videos", "Shopping", "Imágenes", "Web", "Libros", "Noticias", "Más", and "Herramientas". The main content area features several sections: "Java Downloads | Oracle" with a link to "JDK Development Kit 23.0.1 downloads"; "Oracle JDK 23" with a link to "Download the Java including the latest version 17 LTS on the ..."; "Java SE" with a link to "JDK Development Kit 23.0.1 downloads"; "JDK" with a link to "JDK Development Kit 23.0.1 downloads"; "Free Java SE" with a link to "JDK Development Kit 23.0.1 downloads"; "Oracle México" with a link to "JDK Development Kit 23.0.1 downloads"; and a link to "Más resultados de oracle.com »".

Elija la opción Oracle JDK xx y elija la opción acorde a sus sistema operativo (en este caso Windows)



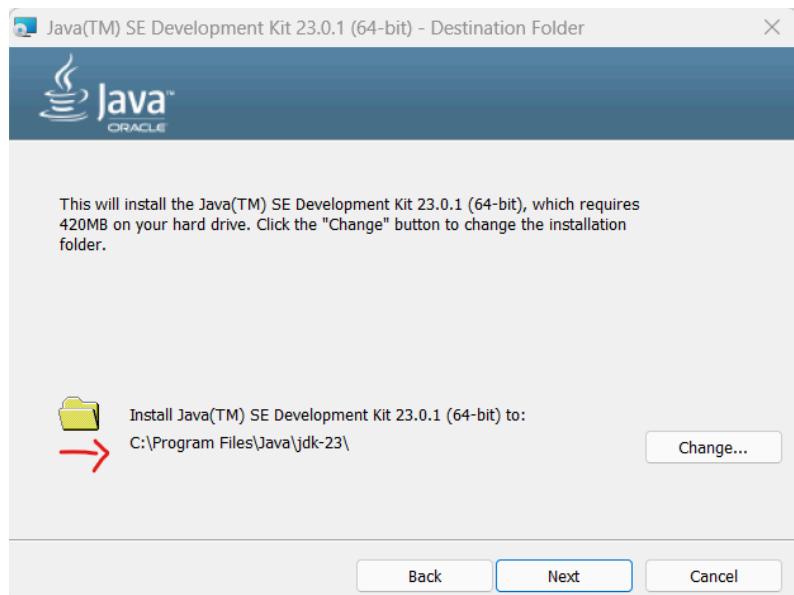
The screenshot shows the Oracle Java archive page for JDK 23.0.1. The top navigation bar includes links for "Productos", "Sectores", "Recursos", "Clientes", "Partners", "Desarrolladores", and "Compañía". Below the navigation bar, there are tabs for "Tools and resources", "Java downloads", and "Java archive". The "Java downloads" tab is selected. Under this tab, there are links for "JDK 23", "JDK 21", "GraalVM for JDK 23", and "GraalVM for JDK 21". The "JDK 23" section is expanded, showing three download options: "x64 Compressed Archive" (228.92 MB), "x64 Installer" (205.21 MB), and "x64 MSI Installer" (203.96 MB). Each option has a corresponding download link.

Se ha elegido la opción Instalador de MSI, así ejecutaremos el instalador:

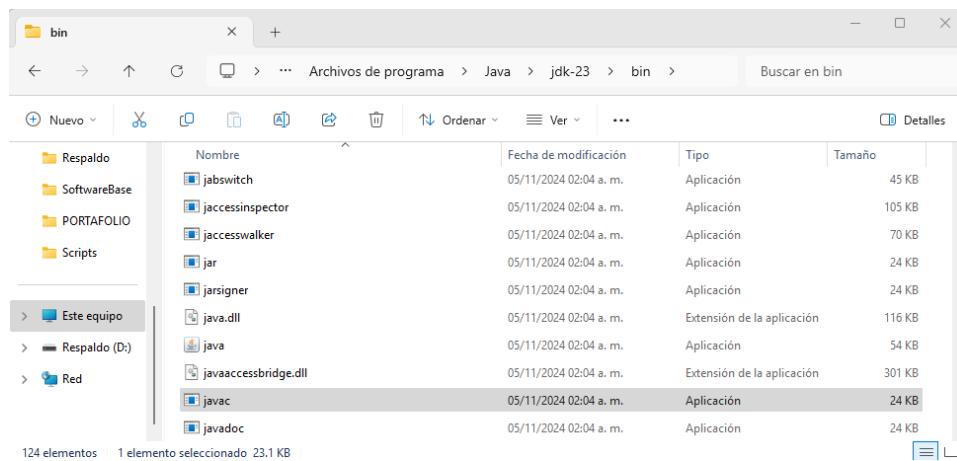


The screenshot shows a file explorer window titled "Descargas". The search bar contains "Buscar en Descargas". The file list includes several items: "WhatsApp Image 2024-10-31 at 16.23.10" (Archivo JPEG), "mingw-w64-v11.0.0" (Carpeta com), "PowerShell" (Microsoft Edge), "s1156_cpu_spec" (Microsoft Edge), "Windows PowerShell Cookbook" (Microsoft Edge), and "jdk-23_windows-x64_bin" (Paquete de).

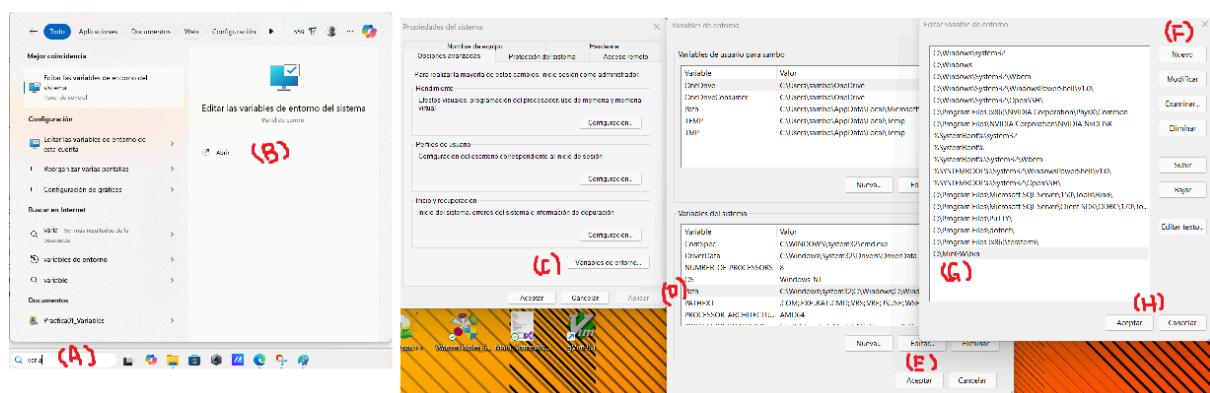
No olvide tomar nota de la ruta donde se instalarán las herramientas:



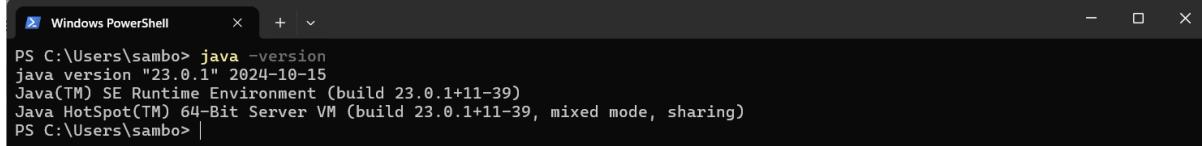
Tendrá que copiar la ruta de acceso en las variables de entorno de Windows:
C:\Program Files\Java\jdk-23\bin #Para este caso



En el cuadro de búsqueda de Windows, escribimos **Variables de Ambiente** (A), ejecutamos la aplicación (B), presionamos el botón **Variables de Entorno** (C), elegimos del cuadro de texto la opción **Path** (D), en la ventana el botón (F) y en (G) pegamos la dirección que copiamos del explorador de Windows y presionamos el botón (H) para terminar (damos clic en Aceptar en las ventanas anteriores).

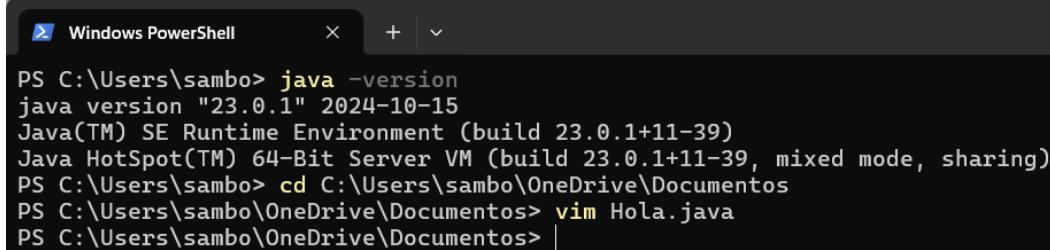


En la terminal de PowerShell tecleamos el comando `java -version`, para verificar la instalación:



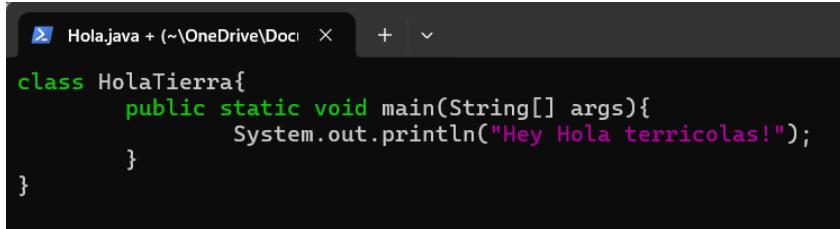
```
PS C:\Users\sambo> java -version
java version "23.0.1" 2024-10-15
Java(TM) SE Runtime Environment (build 23.0.1+11-39)
Java HotSpot(TM) 64-Bit Server VM (build 23.0.1+11-39, mixed mode, sharing)
PS C:\Users\sambo>
```

Ingrese a una carpeta en la que tenga permisos para crear y ejecutar archivos, cree el archivo `Hola.java`



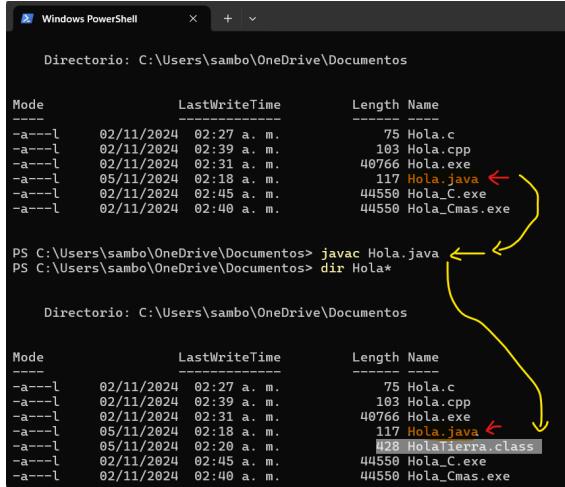
```
PS C:\Users\sambo> java -version
java version "23.0.1" 2024-10-15
Java(TM) SE Runtime Environment (build 23.0.1+11-39)
Java HotSpot(TM) 64-Bit Server VM (build 23.0.1+11-39, mixed mode, sharing)
PS C:\Users\sambo> cd C:\Users\sambo\OneDrive\Documentos
PS C:\Users\sambo\OneDrive\Documentos> vim Hola.java
PS C:\Users\sambo\OneDrive\Documentos>
```

Y escriba el siguiente fragmento de código:



```
PS C:\Users\sambo> vim Hola.java + (~\OneDrive\Doc
class HolaTierra{
    public static void main(String[] args){
        System.out.println("Hey Hola terricolas!");
    }
}
```

Guarde el archivo y compile con la siguiente instrucción:



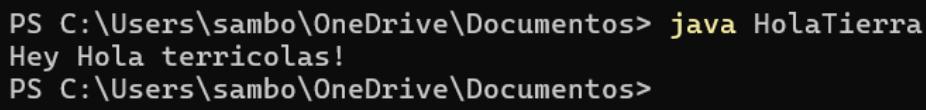
```
PS C:\Users\sambo> javac Hola.java
PS C:\Users\sambo> dir Hola*
Directorio: C:\Users\sambo\OneDrive\Documentos

Mode LastWriteTime Length Name
---- -- -- -
-a---l 02/11/2024 02:27 a.m. 75 Hola.c
-a---l 02/11/2024 02:39 a.m. 103 Hola.cpp
-a---l 02/11/2024 02:31 a.m. 40766 Hola.exe
-a---l 05/11/2024 02:18 a.m. 117 Hola.java ←
-a---l 02/11/2024 02:45 a.m. 44550 Hola_C.exe
-a---l 02/11/2024 02:40 a.m. 44550 Hola_Cmas.exe

PS C:\Users\sambo> javac Hola.java ←
PS C:\Users\sambo> dir Hola*
Directorio: C:\Users\sambo\OneDrive\Documentos

Mode LastWriteTime Length Name
---- -- -- -
-a---l 02/11/2024 02:27 a.m. 75 Hola.c
-a---l 02/11/2024 02:39 a.m. 103 Hola.cpp
-a---l 02/11/2024 02:31 a.m. 40766 Hola.exe
-a---l 05/11/2024 02:18 a.m. 117 Hola.java ←
-a---l 05/11/2024 02:20 a.m. 1128 HolaTierra.class
-a---l 02/11/2024 02:45 a.m. 44550 Hola_C.exe
-a---l 02/11/2024 02:40 a.m. 44550 Hola_Cmas.exe
```

Si no se listan errores, entonces la compilación fue exitosa, así que puede ejecutar su programa usando `java <nombreClase>` sin escribir la extensión:

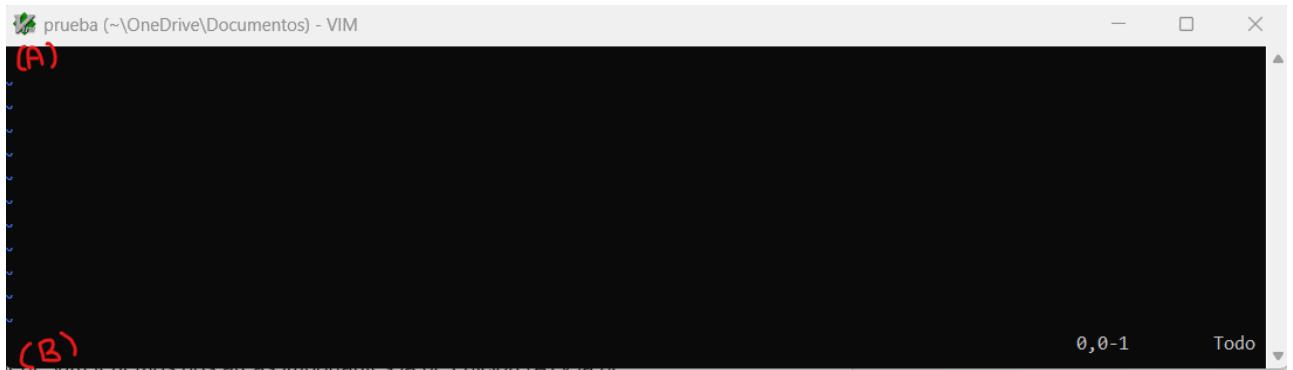


```
PS C:\Users\sambo> java HolaTierra
Hey Hola terricolas!
PS C:\Users\sambo>
```

Anexo E

Manejo básico del editor Vim

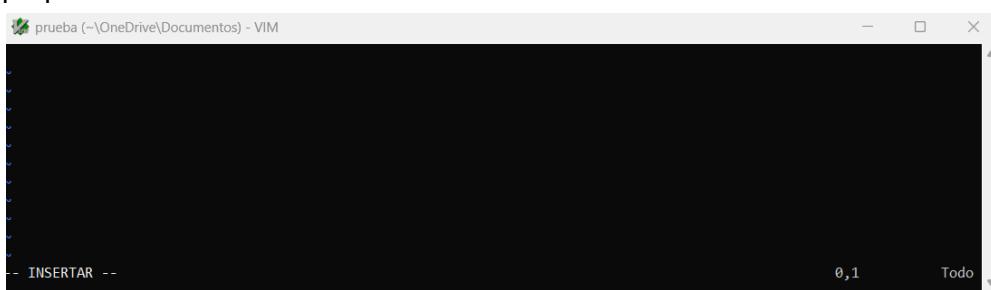
Al ingresar al editor de Vim tenemos dos áreas importantes la de edición (A) y la de comandos (B)



(A)

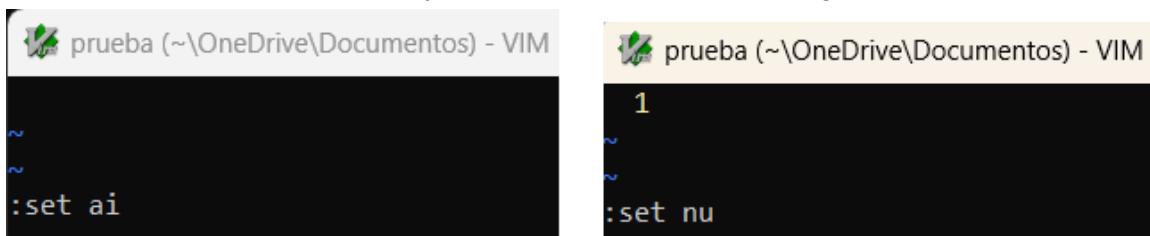
(B)

Para asegurar que estemos en la sección de comandos, presione la tecla <esc>, para comenzar la edición presione la tecla ‘i’, y vera como cambia la sección de comandos ahora mostrará la leyenda INSERTAR puede ingresar letras en la sección marcada por el cursor parpadeando:



-- INSERTAR --

Presionando la tecla <esc> y escribiendo :set ai, se activa la habilidad de auto indentación para la escritura de programas, usando el mismo procedimiento pero escribiendo set nu se activan los números de línea muy útil para ubicarnos en un programa.



:set ai

1

:set nu

Tenemos entonces dos tipos de comandos los que necesitan : y una orden, y los que son activados simplemente con una tecla, tenemos también dos modos modo edición y el modo comandos vinculados a sus respectivas áreas, para intercambiar entre modos usamos la tecla <esc> y alguna de las teclas de edición (usualmente ‘i’ para insertar).

Abriendo Múltiples Archivos en ventanas separadas:

Utilizando el parámetro -o y la siguiente sintaxis al usar el comando vim:

`vim -o <archivo1> <archivo2> ... <archivoN>`

Los diferentes archivos se mostrarán de manera horizontal:

```

Hola.c (~\OneDrive\Documentos) + 5
#include <stdio.h>
int main()
{
    printf("Hello World");
    return 0;
}

Hola.cpp
#include <iostream>
using namespace std;
int main()
{
    cout << "Hola mundo" << endl;
    return 0;
}

Hola.java
class HolaTierra{
    public static void main(String[] args){
        System.out.println("Hey Hola terricolas!");
    }
}

"Hello.java" 6L, 117B

```

Se puede cerrar cada archivo sin guardar los cambios usando el comando `:q!` , o podemos cerrar todos los archivos sin guardar cambios usando `:qa!`

Si deseamos abrir los archivos en posición vertical usamos:

`vim -O <archivo1> <archivo2> ... <archivoN>` #sintaxis igual pero con el parámetro `-O`

```

Hola.c (~\OneDrive\Documentos) + 5
#include <stdio.h>
int main()
{
    printf("Hello World");
    return 0;
}

Hola.cpp
#include <iostream>
using namespace std;
int main()
{
    cout << "Hola mundo" << endl;
    return 0;
}

Hola.java
class HolaTierra{
    public static void main(String[]
} args){
        System.out.println("Hey
        Hola terricolas!");
    }
}

"Hello.java" 6L, 117B

```

Para moverse entre las ventanas usamos la siguiente lista de comandos

(`Ctrl + w`) + `w` -> Cambiar a la siguiente ventana (Por orden de la creación).

(`Ctrl + w`) + `h` -> Moverse a la ventana izquierda.

(`Ctrl + w`) + `l` -> Moverse a la ventana derecha.

(`Ctrl + w`) + `j` -> Moverse a la ventana de abajo.

(`Ctrl + w`) + `k` -> Moverse a la ventana de arriba.

Note que todas las acciones comienzan con la combinación de teclas CTRL + W (presionar juntos soltar) y después seleccionar la tecla para moverse a donde queramos.

Note también que los comandos pueden no funcionar en todos los teclados.

Si queremos ver el mismo archivo en dos secciones diferente, podemos dividir la ventana usando el comando `:visp` o `:visplit` esto nos permitira examinar diferentes partes del código:

```

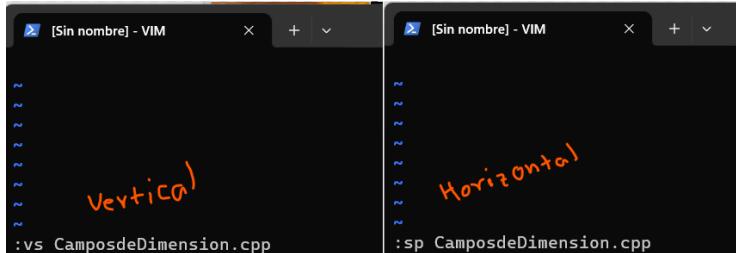
1 // practical.cpp : Este archivo contiene la función "main". 1 // practical.cpp : Este archivo contiene la función "main".
n". La ejecución del programa comienza y termina ahí. n". La ejecución del programa comienza y termina ahí.
2 // 2 //
3 3
4 #include <fstream> 4 #include <fstream>
5 #include <iostream> 5 #include <iostream>
6 #include <cstring> 6 #include <cstring>
7 using namespace std; 7 using namespace std;
8 char ARCHIVO[] = "CORREOS.DAT"; 8 char ARCHIVO[] = "CORREOS.DAT";
9 ←(E) 9
10 class correo 10 class correo
11 { 11 {
12 private: 12 private:
13     string nombre; 13     string nombre;
14     //string contenido; 14     //string contenido;
15 public: 15 public:
16     ~correo() 16     ~correo()
17     {} 17     {}
18     correo() :nombre("NULL") // contenido("NULL") 18     correo() :nombre("NULL") // contenido("NULL")
19     {} 19     {}
20     void pedirDatos() 20     void pedirDatos()
21     { 21     {
22         cout << "\nNombre: "; cin >> nombre; 22         cout << "\nNombre: "; cin >> nombre;
23         //cout << "\nContenido: "; cin >> contenido; 23         //cout << "\nContenido: "; cin >> contenido;
24     } 24     }
25     void mostrarDatos() 25     void mostrarDatos()
26     { 26     {
27         cout << "\nNombre: " << nombre; 27         cout << "\nNombre: " << nombre;
28     } 28     }

```

:vs (A) 2,1 Comienzo .\CamposdeDimension.cpp (B) 2,1 Comienzo (C) 2,1 Comienzo (D) 2,1 Comienzo (E)

En (A) podemos ver la invocación del comando, (B) y (C) Muestran que el nombre es el mismo archivo, (D) indica la ubicación de fila columna en la ventana y (E) lista los números de línea.

Si ya tenemos abierto Vim, y deseamos abrir otro archivo en una ventana dividida, agregamos el nombre del archivo al comando :vs <nombreArchivo> y lo abrirá en una ventana vertical, si usamos :sp <nombreArchivo> lo abrirá en una ventana Horizontal.



Podemos combinar los comandos :vsplit o :split para abrir las ventanas en el orden que nos interese:

```

// practical.cpp : Este archivo contiene la función "main". // practical.cpp : Este archivo contiene la función "main".
La ejecución del programa comienza y termina ahí. La ejecución del programa comienza y termina ahí.
// //

#include <fstream> #include <fstream>
#include <iostream> #include <iostream>
#include <cstring> #include <cstring>
using namespace std; using namespace std;
char ARCHIVO[] = "CORREOS.DAT"; char ARCHIVO[] = "CORREOS.DAT";

class correo class correo
{
private: private:
    string nombre; string nombre;
}

```

CamposdeDimension.cpp 1,1 Comienzo CamposdeDimension.cpp 2,1 Comienzo

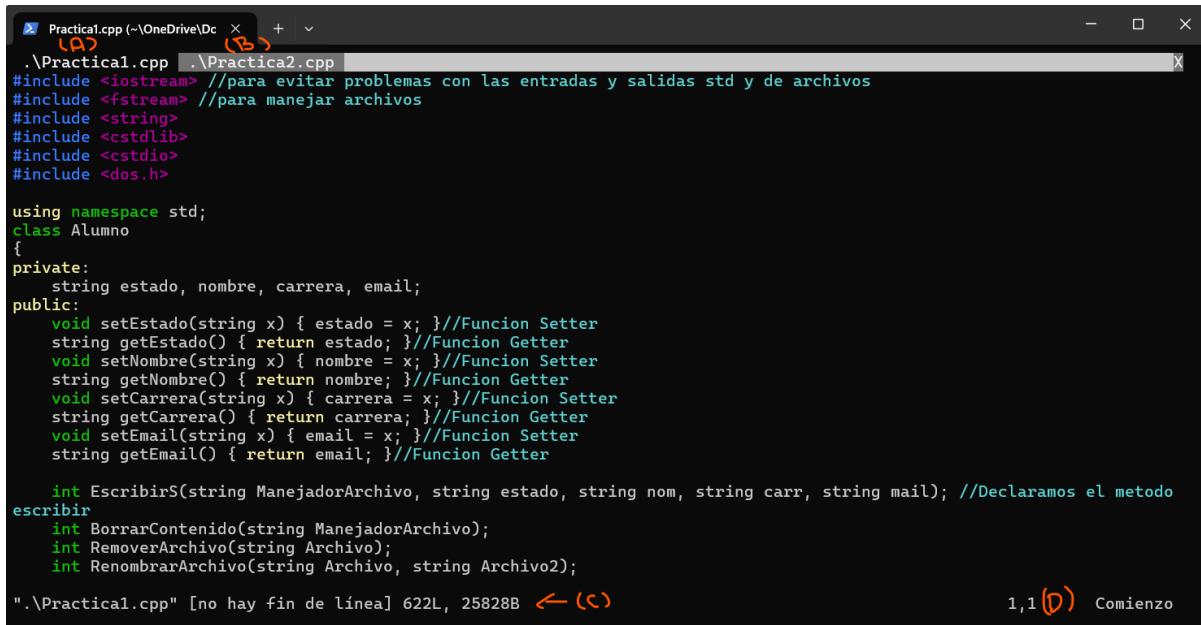
[Sin nombre] [+] CamposdeDimension.cpp" 121 lines --0%-- 2,0-1 Todo

Abriendo Múltiples Archivos en pestañas separadas:

Usando el parámetro -p y la sintaxis:

`vim -p <Archivo1> <Archivo2> ... <ArchivoN>`

Los archivos se muestran en diferentes pestañas en el editor:



```
Practica1.cpp (~\OneDrive\DC) + ~ 
.\Practica1.cpp ..\Practica2.cpp
#include <iostream> //para evitar problemas con las entradas y salidas std y de archivos
#include <fstream> //para manejar archivos
#include <string>
#include <cstdlib>
#include <cstdio>
#include <dos.h>

using namespace std;
class Alumno
{
private:
    string estado, nombre, carrera, email;
public:
    void setEstado(string x) { estado = x; }//Funcion Setter
    string getEstado() { return estado; }//Funcion Getter
    void setNombre(string x) { nombre = x; }//Funcion Setter
    string getNombre() { return nombre; }//Funcion Getter
    void setCarrera(string x) { carrera = x; }//Funcion Setter
    string getCarrera() { return carrera; }//Funcion Getter
    void setEmail(string x) { email = x; }//Funcion Setter
    string getEmail() { return email; }//Funcion Getter

    int EscribirS(string ManejadorArchivo, string estado, string nom, string carr, string mail); //Declaramos el metodo
escribir
    int BorrarContenido(string ManejadorArchivo);
    int RemoverArchivo(string Archivo);
    int RenombrarArchivo(string Archivo, string Archivo2);
};

.\Practica1.cpp" [no hay fin de linea] 622L, 25828B ← (C) 1,1 (D) Comienzo
```

En (A) es la pestaña del archivo 1 (B) la pestaña del archivo 2, (C) indica en cual archivo nos encontramos y (D) la ubicación dentro del archivo (podemos colocar números de línea con :set nu), para movernos hacia la derecha en las pestañas presionamos la tecla *ESC* para asegurar estar en modo comando y tecleamos *gt* para movernos hacia la derecha entre pestañas y *gT* para movernos hacia la izquierda, si usamos *2gt* nos moveremos 2 pestañas.

- :tabnew abre una pestaña nueva vacía o con el archivo especificado
- :tabclose Cierra la pestaña actual
- :tabonly Cierra todas las pestañas excepto la actual
- :tabs list Muestra todas las pestañas y sus ventanas
- :tabm 0 Mueve la pestaña actual a la primera
- :tabm Mueve la pestaña actual a la última
- :tabm {i} Mueve la pestaña actual a la posición i+1
- :tabm +{i} Mueve la pestaña actual hacia la derecha a la posición actual+i
- :tabm -{i} Mueve la pestaña actual hacia la izquierda a la posición actual-i
- :tabn Va a la siguiente pestaña
- :tabp Va a la pestaña anterior
- :tabfirst Va a la primera pestaña
- :tablast Va a la última pestaña

Anexo F

Máquinas Virtuales con Qemu

Usaremos el software Qemu para emular nuestras máquinas virtuales, en el explorador de internet, busque Qemu y ubique la sección Download, para la descarga del programa:

[Download QEMU - QEMU](https://www.qemu.org/download/#windows)

The screenshot shows the QEMU download page at <https://www.qemu.org/download/#windows>. At the top, there's a navigation bar with links for HOME, DOWNLOAD, SUPPORT, CONTRIBUTE, DOCS, WIKI, and BLOG. Below the navigation bar, there are four tabs: Source code, Linux, macOS, and Windows. The Windows tab is highlighted with a red border. To the left of the tabs is a large QEMU logo. Below the tabs, there's a section for MSYS2, which says "QEMU can be installed using MSYS2 also. MSYS2 uses pacman to manage packages. First, follow the [MSYS2 installation procedure](#). Then update the packages with pacman -Syu command. Now choose the proper command for your system as following:" followed by two bullet points: "For 64 bit Windows 7 or above (in MINGW64):" and "pacman -S mingw-w64-x86_64-qemu". Below that is another bullet point: "For 64 bit Windows 8.1 or above (in UCRT64):" followed by the command "pacman -S mingw-w64-x86_64-qemu". A red arrow labeled '(A)' points to the Windows tab.

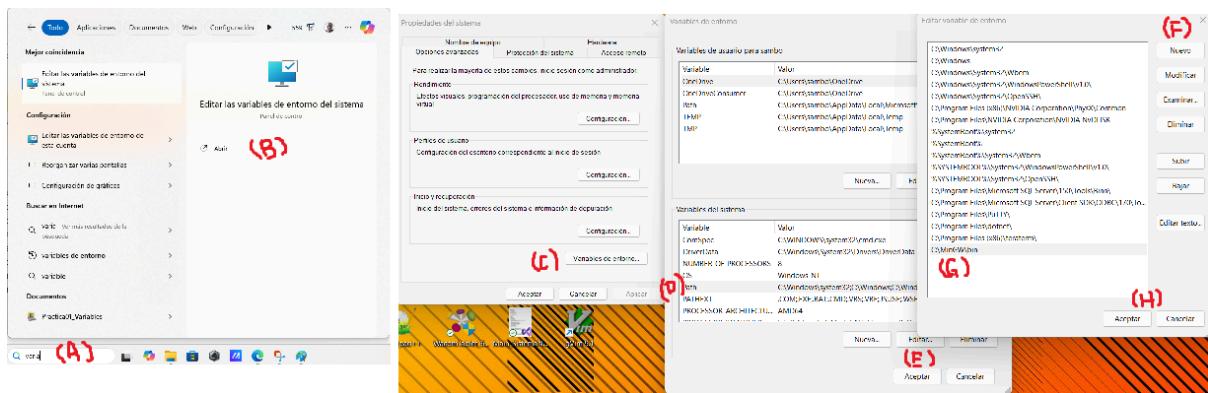
Como muestra (A) seleccione el instalador dependiendo del tipo de sistema que tenga (para este ejemplo 64x):

The screenshot shows the QEMU download page at <https://qemu.weinnetz.de/w64/>. It displays a list of files and directories. The directory structure includes 2812/, 2813/, 2814/, 2815/, 2816/, 2817/, 2818/, 2819/, 2820/, 2821/, 2822/, 2823/, 2824/, and old/. Below these are two files: qemu-w64-setup-20240903.exe and qemu-w64-setup-20240903.sha512. A red arrow labeled '(A)' points to the qemu-w64-setup-20240903.exe file.

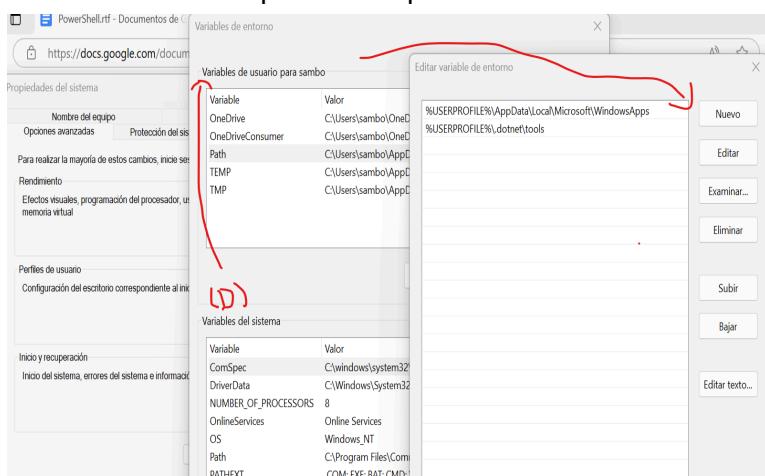
Nos redirecciona a una página con las descargas, elegimos el instalador de Windows (A), para evitar tener que compilar e instalar el software.

The screenshot shows the QEMU 9.1.0 (64bit) setup wizard. The first step, "Elegir lugar de instalación", asks to select the installation directory. The default path is C:\Program Files\qemu. Below it, it says "El programa de instalación instalará QEMU en el siguiente directorio. Para instalar en un directorio diferente, presione Examinar y seleccione otro directorio. Presione Instalar para comenzar la instalación." The second step, "Directorio de Destino", shows the same path. It also displays disk space information: "Espacio requerido: 1.1 GB" and "Espacio disponible: 345.9 GB". At the bottom are buttons for "Atrás", "Instalar", and "Cancelar". A red arrow labeled '(A)' points to the "Instalar" button.

No olvide tomar nota de la dirección de instalación del software, ya que necesitaremos agregarla a las variables de ambiente para completar la instalación, en el cuadro de búsqueda de Windows, escribimos *Variables de Ambiente* (A), ejecutamos la aplicación (B), presionamos el botón *Variables de Entorno* (C), elegimos del cuadro de texto la opción *Path* (D), en la ventana el botón (F) y en (G) pegamos la dirección y presionamos el botón (H) para terminar (damos clic en Aceptar en las ventanas anteriores).



Hacemos lo mismo para el bloque usuarios arriba del inciso (D):



Y pegamos la dirección del software Qemu, de esta manera podemos acceder al ejecutable desde cualquier localidad en el sistema. verificamos la instalación con el comando:

`qemu-system-x86_64 --version`

```
PS C:\Users\sambo\OneDrive\Documentos> qemu-system-x86_64 --version
QEMU emulator version 9.1.0 (v9.1.0-12064-gc658eefbf44)
Copyright (c) 2003-2024 Fabrice Bellard and the QEMU Project developers
PS C:\Users\sambo\OneDrive\Documentos>
```

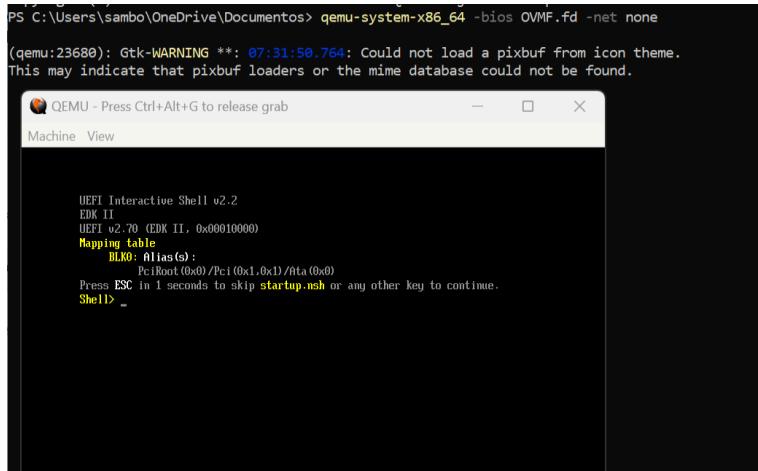
Para ejecutar una máquina virtual Qemu, usaremos el UEFI-BIOS de prueba, en su explorador de internet busque OVMF.fd download:

[common/OVMFfd at master · clearlinux/common · GitHub](https://github.com/clearlinux/common/blob/master/OVMFfd)

The screenshot shows a GitHub repository page for 'common/OVMFfd'. The repository has 6 issues, 23 forks, and 54 stars. The code tab is selected, showing a single file named 'OVMFfd'. A red arrow points to the 'Raw' download link at the bottom right of the file preview.

Descargue el archivo OVMF.fd usando el icono de descarga, posteriormente ejecutar Qemu con la ubicación del archivo de arranque:

```
qemu-system-x86_64 -bios OVMF.fd -net none
```

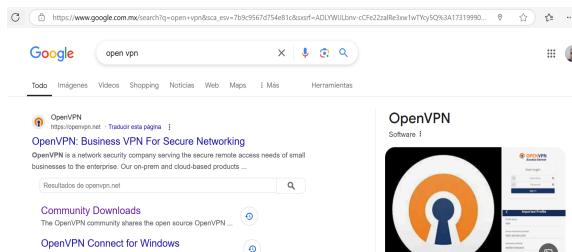


Generará una ventana de una VM ejecutando el Firmware UEFI-BIOS, para manipular cómodamente la ventana use la combinación de teclas *ctrl + alt + g*, esto libera el menú superior (la combinación de teclas se muestra en el marco superior de la ventana). para salir escriba el comando *reset -s* y enter.

Anexo G

Accediendo a nuestra máquina virtual mediante conexión ssh desde nuestro host

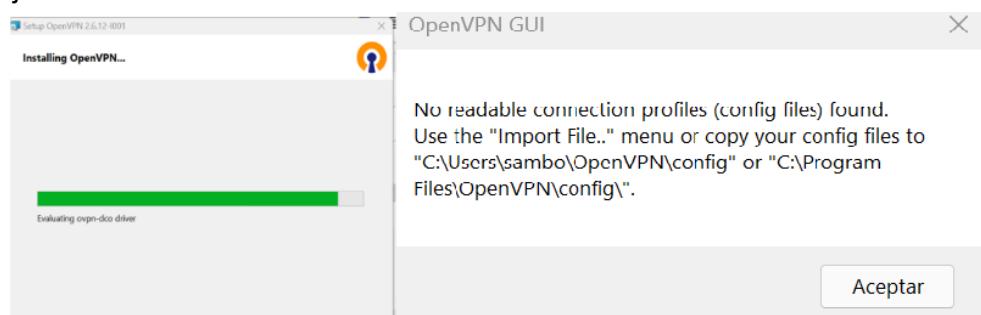
Necesitamos instalar el paquete “TAP-Win32 Virtual Ethernet Adapter” como lo especifica la documentación de Qemu, este forma parte del proyecto Open VPN, en el explorador buscamos:



Elegimos el enlace “Community Downloads”, y en la página descargamos el instalador adecuado, en este caso será el de Windows 64:

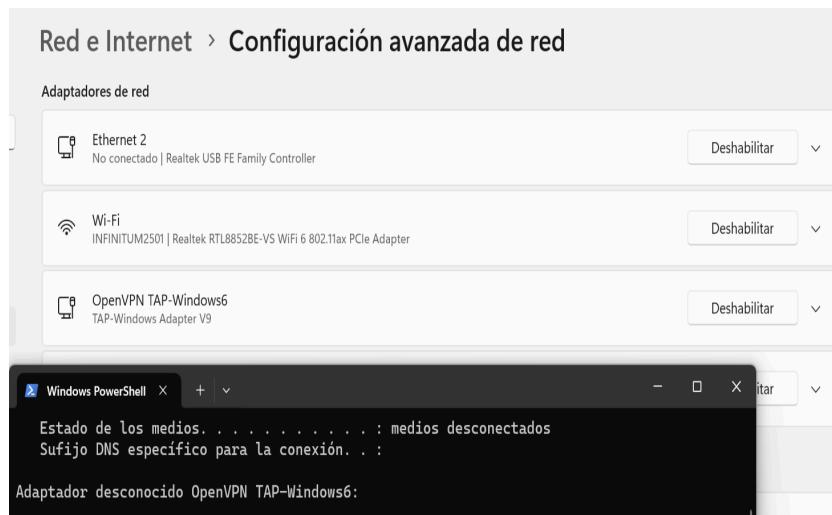
Windows ARM64 MSI installer GnuPG Signature OpenVPN-2.6.12-I001-arm64.msi

Ejecutamos el instalador:



El cual instalará el controlador TAP (Temporary Access Point), como no hay archivos de configuración de VPN instalados nos enviará un mensaje de error, lo ignoramos porque no requerimos la función por el momento.

Verificamos que nuestro controlador TAP se halla instalado correctamente:



Renombramos nuestro TAP a TAP9 usando la opción de la pestaña a un lado del botón deshabilitar, para que sea más sencillo conectar con Qemu.

La siguiente línea muestra cómo lanzar una máquina virtual qemu con un puerto de red para conectar el HOST a la VM mediante ssh.

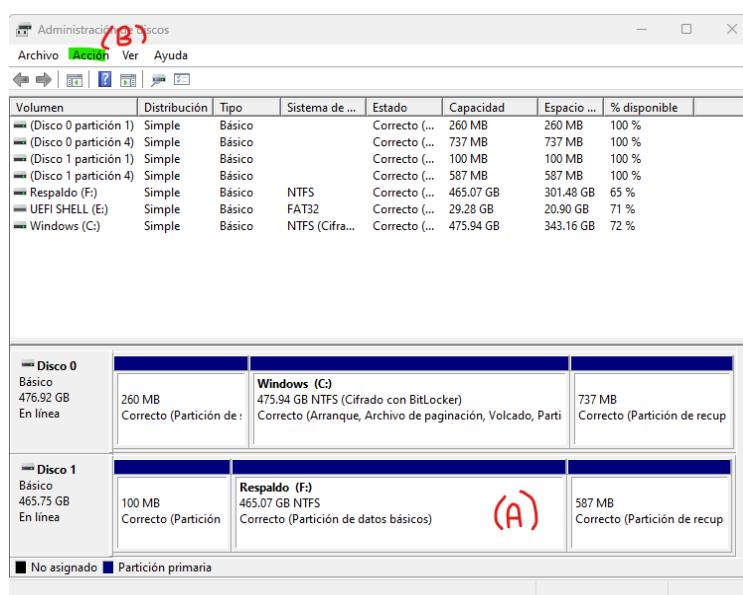
```
qemu-system-x86_64.exe -hda <Archivo.img> -m 2048 -netdev  
tap,id=mynet0,ifname=<tap0>,script=no,downscript=no -device  
e1000,netdev=mynet0,mac=52:55:00:d1:55:01
```

*Nota no está terminada esta sección

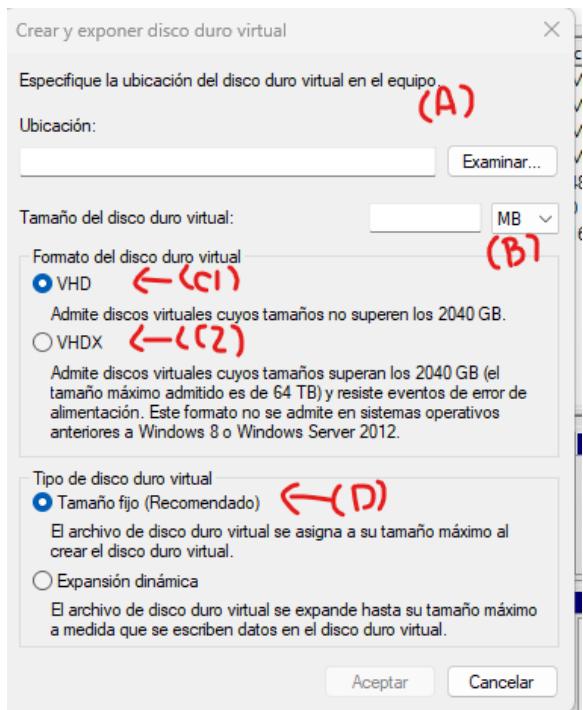
Anexo H

Agregando un disco USB virtual con FAT32 a nuestra máquina virtual

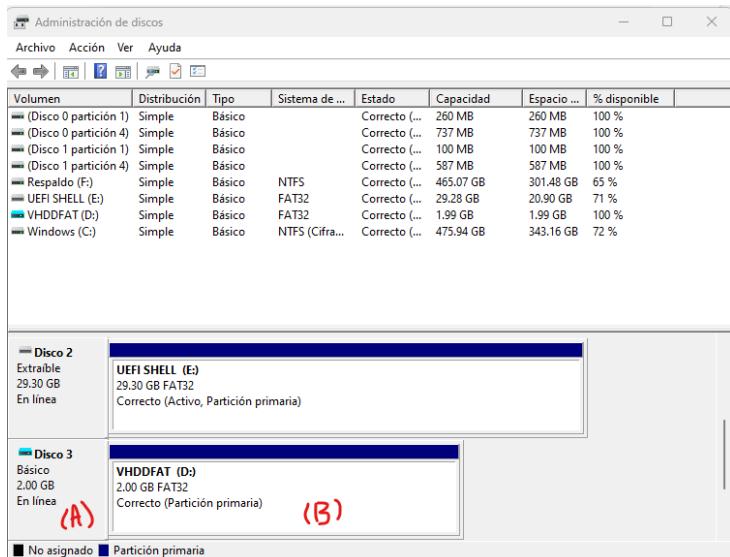
Crearemos un disco virtual para conectar con nuestra máquina virtual y poder ejecutar las aplicaciones UEFI que vayamos creando, usando la herramienta *disk management* (escriba la sentencia en buscador de la barra de herramientas de windows):



Seleccionamos el disco (A) y en el menú Accion->Crear VHD, nos envía a la ventana de configuración:

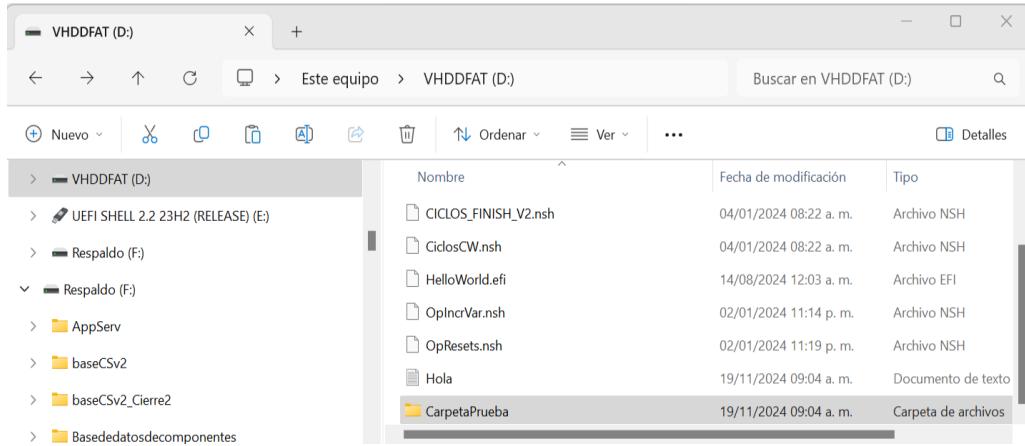


Donde especificamos en (A) la ubicación de nuestro disco virtual y en (B) el tamaño en Megas o en Gigas dependiendo de sufijo, seleccionamos (C1) para un disco de máximo 2Gb de espacio (use C2 si desea un disco más grande), importante indicar (D) para que el disco conserve su tamaño y respete el espacio asignado.

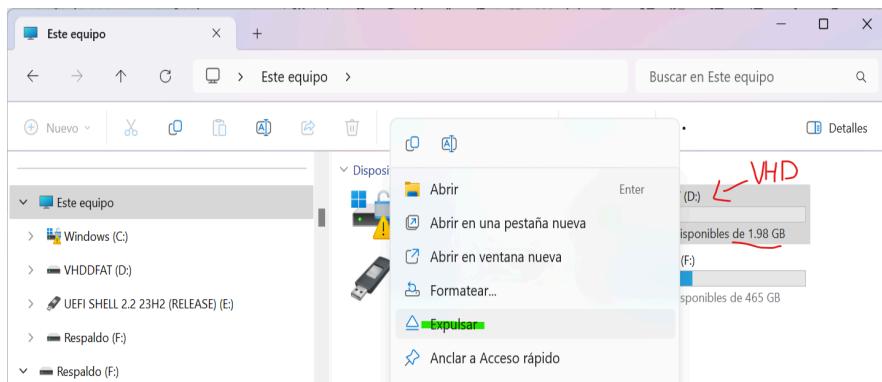


Cuando el disco se muestre, lo seleccionamos en (A) y hacemos click con el botón derecho para seleccionar “Nuevo Volumen”(Initialize Disk), importante escoger formato “MBR”, asignar nombre a la unidad y volumen, posteriormente en (B) usar la opción “Nuevo Volumen” ("New Simple Volume"), prestamos atención a seleccionar como Formato FAT32, continuar con la instalación.

En el explorador de archivos nos ubicamos en dentro del disco virtual copiamos unos archivos de prueba y creamos un archivo Hola.txt de prueba así como una carpeta de prueba:



Expulsamos el disco para poder redirigirlo a la Máquina Virtual:



Dentro de Powershell nos ubicamos a la dirección donde se encuentra el disco (A):

```
Administrator: Windows PowerShell
PS F:\VirtualMachines\RasbianOS_VM\RaspbianHDD> pwd
Path <- (A)
-----
F:\VirtualMachines\RasbianOS_VM\RaspbianHDD

PS F:\VirtualMachines\RasbianOS_VM\RaspbianHDD> ls

    Directorio: F:\VirtualMachines\RasbianOS_VM\RaspbianHDD

Mode          LastWriteTime      Length Name
----          -----        -----
-a---  11/10/2023 12:50 p. m.  3607101440 2022-07-01-raspberries-bullseye-i386.iso
-a---  19/11/2024 05:12 a. m.  4866834432 AntXLinux.img
-a---  15/11/2024 07:04 a. m.  4194304 OVMF.fd <- (D1)
-a---  12/11/2024 07:06 a. m.  7540375552 RaspberryPi.img
-a---  12/11/2024 04:49 a. m.  7538081792 RaspbianVMHDD.img
-a---  19/11/2024 08:28 a. m.  2147484160 VFAT32.vhd <- (D2) <- (C)
```

Listamos los archivos y vemos que se encuentra la imagen de UEFI para Qemu (D1) y nuestro disco virtual (D2), ejecutamos el comando (C) que se muestra más abajo:

`qemu-system-x86_64 -bios .\OVMF.fd -hda .\VFAT32.vhd -net none`

Lanzamos la máquina virtual, si todo salió correctamente veremos el disco como una unidad FS y podremos ver los archivos de prueba que copiamos anteriormente:

The screenshot shows a QEMU terminal window titled "QEMU - Press Ctrl+Alt+G to release grab". The terminal displays the following text:

```
UEFI v2.70 (EDK II, 0x00010000)
Mapping table
FS0: Alias(s) :HDD0a1::BLK1: ←
  PciRoot(0x0)/Pci(0x1,0x1)/Ata(0x0)/HD(1,MBR,0xA8066617,0x80,0x3FE800)
BLK0: Alias(s) :
  PciRoot(0x0)/Pci(0x1,0x1)/Ata(0x0)
BLK2: Alias(s) :
  PciRoot(0x0)/Pci(0x1,0x1)/Ata(0x0)
Press ESC in 4 seconds to skip startup.nsh or any other key to continue.
Shell> FS0:
FS0:> ls
Directory of: FS0:\

11/19/2024  15:15           10,549  NuVars
02/07/2024  12:47            5,591  ASM.EFI
01/10/2023  17:55             430  BorrarVar.nsh
01/04/2024  08:22            8,112  CICLOS_FINISH_V2.nsh
01/04/2024  08:22            8,112  CiclosCW.nsh
08/14/2024  00:03            8,896  HelloWorld.efi
01/02/2024  23:14             646  OpIncrVar.nsh
01/02/2024  23:19            1,056  OpResets.nsh
11/19/2024  09:04              19  Hola.txt ←
11/19/2024  09:04 <DIR>        4,096  CarpetaPrueba ←
                           9 File(s)    43,411 bytes
                           1 Dir(s)
```

Apagamos la máquina virtual con el comando `reset -s` para volver a “montar” la unidad virtual, simplemente accedemos a la ubicación del archivo dentro del explorador, así podemos agregar los archivos de nuestras prácticas para probarlos, sin necesidad de tener un hardware específico.

Anexo I

Configurando un sistema para conexión remota mediante Powershell

Powershell tiene soporte de Microsoft para algunos sistemas Linux, consulte la página oficial para mayor información. Para que una máquina reciba comandos remotos de Powershell necesita contar con el CLI (Command Line Interpreter), un método de conexión (WRM para sistemas con Windows y SSH para sistemas con Linux) y una cuenta de usuario con su password que tenga los permisos necesarios para ejecutar comandos de Powershell en el sistema remoto.

Linux

Como ejemplo del caso de Linux usaremos una Raspberry Pi, ingresamos a nuestro equipo remoto (en este caso una sesión SSH en la terminal de Powershell en nuestra Windows PC)

```

PS C:\Users\sambo> ssh rodrigo@raspberrypi <(A)
rodrigo@raspberrypi's password: <(B)
Linux raspberrypi 6.1.21-v8+ #1642 SMP PREEMPT Mon Apr 3 17:24:16 BST 2023 aarch64
The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Wed Nov 27 02:55:34 2024 from fe80::b836:9af7:5508:5435%eth0
rodrigo@raspberrypi:~ $ <(C)

```

Ingresamos en (A) el par Usuario-HostName y la clave para terminar registrados en nuestro sistema como indica (B). Una vez dentro de nuestro sistema remoto ejecutamos los siguientes pasos:

-Primero actualizamos el sistema:

`sudo apt update`

-Instalamos el software correspondiente que ayuda a la ejecución de Powershell

`sudo apt install wget libssl1.1 libunwind8`

-Creamos un directorio para la instalación de PowerShell

`sudo mkdir -p /opt/microsoft/powershell/7`

-Otorgamos permisos al directorio para la ejecución de programas

`sudo chmod +x /opt/microsoft/powershell/7/pwsh`

-Necesitamos instalar la aplicación Wget para descargar PowerShell

#para equipos de 32 bits

`wget -O /tmp/powershell.tar.gz`

<https://github.com/PowerShell/PowerShell/releases/download/v7.2.6/powershell-7.2.6-linux-arm32.tar.gz>

#para equipos de 64 bits

`$wget -O /tmp/powershell.tar.gz`

<https://github.com/PowerShell/PowerShell/releases/download/v7.2.6/powershell-7.2.6-linux-arm64.tar.gz>

Instale la versión adecuada a su sistema (32 o 64 bits)

`sudo tar zxf /tmp/powershell.tar.gz -C /opt/microsoft/powershell/7`

-Una vez instalado enlazamos el ejecutable de Powershell a las variables de entorno

`sudo ln -s /opt/microsoft/powershell/7/pwsh /usr/bin/pwsh`

-Eliminamos los archivos de instalación

`sudo rm /tmp/powershell.tar.gz`

-Ejecutamos Powershell para verificar que la instalación fue exitosa

`pwsh #salimos del CLI tecleando exit`

```

rodrigo@raspberrypi:~ $ pwsh <(E)
PowerShell 7.2.6
Copyright (c) Microsoft Corporation.

https://aka.ms/powershell
Type 'help' to get help.

A new PowerShell stable release is available: v7.4.6
Upgrade now, or check out the release page at:
https://aka.ms/PowerShell-Release?tag=v7.4.6

ps /home/rodrigo> exit <(S)
rodrigo@raspberrypi:~ $ <(S)

```

-Nos aseguramos de tener instalado los elementos de Cliente y Servidor para SSH en nuestro sistema remoto.

```
sudo apt-get install openssh-server
```

```
sudo apt-get install openssh-client
```

-Ingresamos al archivo de configuración sshd_config

```
sudo vim /etc/ssh/sshd_config
```

-Para agregar la siguiente instrucción en el apartado de Subsystems

```
Subsystem powershell /usr/bin/pwsh -sshs -NoLogo
```

```
#PidFile /var/run/sshd.pid
#MaxStartups 10:30:100
#PermitTunnel no
#ChrootDirectory none
#VersionAddendum none

# no default banner path
#Banner none

# Allow client to pass locale environment variables
AcceptEnv LANG LC_*

# override default of no subsystems
Subsystem      sftp      /usr/lib/openssh/sftp-server
Subsystem      powershell /usr/bin/pwsh -sshs -NoLogo
Subsystem      powershell /usr/bin/pwsh -sshs -NoLogo

# Example of overriding settings on a per-user basis
#Match User anoncvs
#       X11Forwarding no
```

Verifique que la línea PasswordAutentication este en Yes

```
# HostbasedAuthentication
#IgnoreUserKnownHosts no
# Don't read the user's ~/.rhosts and ~/.shosts files
#IgnoreRhosts yes

# To disable tunneled clear text passwords, change to no here!
#PasswordAuthentication yes
#PermitEmptyPasswords no

# Change to yes to enable challenge-response passwords (beware issues with
# some PAM modules and threads)
#ChallengeResponseAuthentication no

# Kerberos options
#KerberosAuthentication no
#KerberosOrLocalPasswd yes
#KerberosTicketCleanup yes
#KerberosGetAFSToken no

-- VISUAL --
```

27 58,1

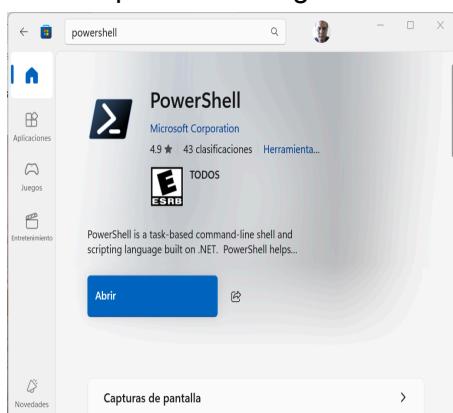
```
#PasswordAutentication yes
```

-Reinicie el archivo de configuración con los nuevos servicios

```
sudo systemctl restart sshd
```

Windows (*En construcción*)

Asegure tener instalado una versión de Powershell arriba de 4.0, en caso de que no cuente con una puede descargar PS desde la tienda de software de windows:



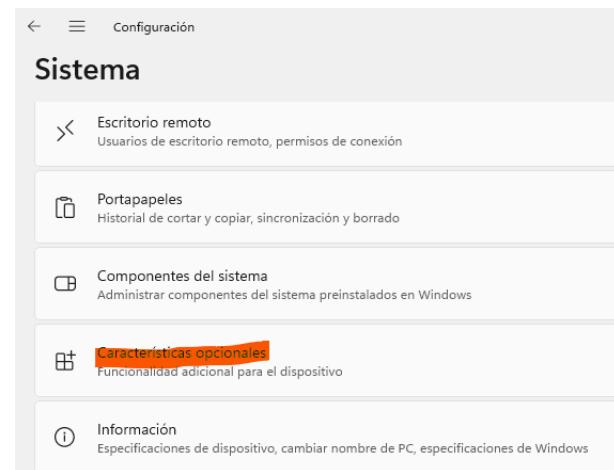
Una vez instalado, proceda con la configuración del equipo para habilitar la conexión **Utilice el protocolo de conexión provisto por Microsoft con WinRM para mayor seguridad, en caso de contar con una versión de sistema operativo Pro, Estudiante o Server, y no poder establecer un dominio utilice la opción SSH, la cual se describe a continuación no olvide deshabilitar la configuración al terminar.**

El siguiente proceso debe realizarse en cada máquina que se deseé habilitar en la RED.

-En la ventana de CONFIGURACION habilitar el modo DESARROLLADORES (PARA PROGRAMADORES)(1), se indica una advertencia aceptar elija SI para proceder, esto iniciará la instalación del componente adicional MODO DE DESARROLLADORES, (posteriormente Active la casilla de DETECCIÓN DE DISPOSITIVOS(2)).



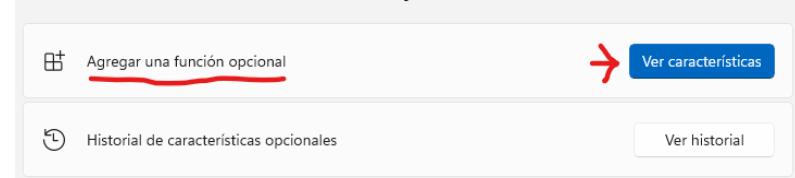
Tras ejecutar la acción nos envía al menú SISTEMA>CARACTERISTICAS OPCIONALES:



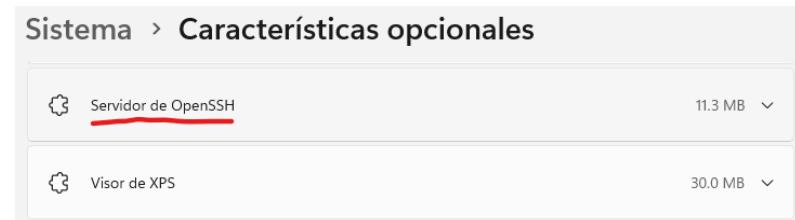
Este movimiento habilita la opción de ping en el sistema y la máquina puede ser localizada en una red.

Debemos instalar en CARACTERISTICAS DE OPCIONALES el servicio de Servidor SSH

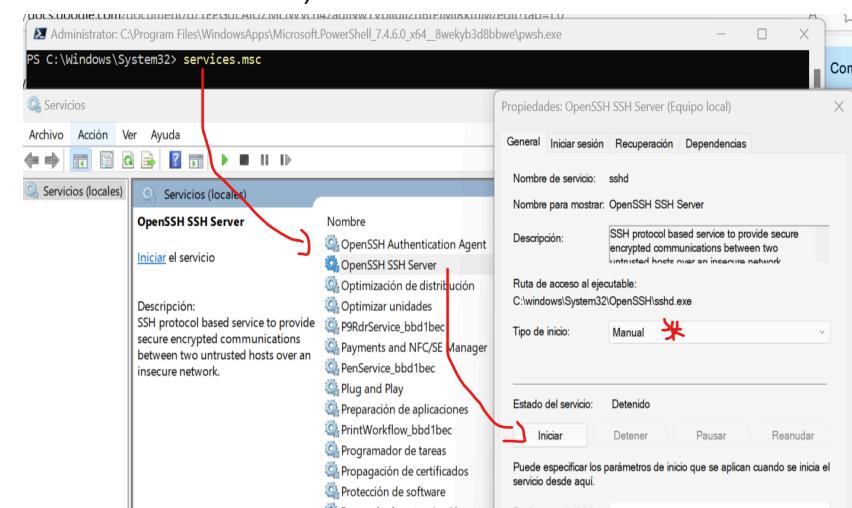
Sistema > Características opcionales



Busque e instale la opción OpenSSH Server, tras instalar aparecerá en la lista de características encontradas más debajo:



-Abra una ventana de Powershell con permisos de administrador y ejecute el comando: `services.msc` y busque el servicio **OpenSSH server** y activelo (nota, deje la casilla de “modo manual” intacta):



Una vez establecida la configuración de la conexión comprobamos que funcione correctamente. En una ventana de PowerShell con permisos de administrador, verifique en ambas maquinas sus direcciones IP:



Usando el comando `ping`, verificamos que el cliente pueda conectar al servidor:

```
[Administrator: C:\Program Files\WindowsApps\Microsoft.PowerShell_7.4.6.0_x64] PS C:\Windows\System32> ping 169.254.97.29

Haciendo ping a 169.254.97.29 con 32 bytes de datos:
Respuesta desde 169.254.97.29: bytes=32 tiempo=1ms TTL=128
Respuesta desde 169.254.97.29: bytes=32 tiempo=2ms TTL=128
Respuesta desde 169.254.97.29: bytes=32 tiempo=1ms TTL=128
Respuesta desde 169.254.97.29: bytes=32 tiempo=2ms TTL=128

Estadísticas de ping para 169.254.97.29:
    Paquetes: enviados = 4, recibidos = 4, perdidos = 0
                (0% perdidos),
Tiempos aproximados de ida y vuelta en milisegundos:
    Mínimo = 1ms, Máximo = 2ms, Media = 1ms
PS C:\Windows\System32>
```

y viceversa (Servidor a Cliente):

```
[Administrator: C:\Program Files\WindowsApps\Microsoft.PowerShell_7.4.6.0] PS C:\Windows\System32> ping 169.254.17.218

Haciendo ping a 169.254.17.218 con 32 bytes de datos:
Respuesta desde 169.254.17.218: bytes=32 tiempo=1ms TTL=128
Respuesta desde 169.254.17.218: bytes=32 tiempo=1ms TTL=128
Respuesta desde 169.254.17.218: bytes=32 tiempo=1ms TTL=128
Respuesta desde 169.254.17.218: bytes=32 tiempo=2ms TTL=128

Estadísticas de ping para 169.254.17.218:
    Paquetes: enviados = 4, recibidos = 4, perdidos = 0
                (0% perdidos),
Tiempos aproximados de ida y vuelta en milisegundos:
    Mínimo = 1ms, Máximo = 2ms, Media = 1ms
PS C:\Windows\System32>
```

Podemos entonces conectar a los equipos mediante una conexión de protocolo SSH:

Cliente al Servidor:

```
[Administrator: C:\Program Files\WindowsApps\Microsoft.PowerShell_7.4.6.0_x64] PS C:\Windows\System32> ssh RODRIGOTR\sambo@169.254.97.29
RODRIGOTR\sambo@169.254.97.29's password:
```

Servidor a Cliente:

```
[Administrator: C:\Program Files\WindowsApps\Microsoft.PowerShell_7.4.6.0_x64_8weky] PS C:\Windows\System32> ssh RODRIGOHP\samborms@169.254.17.218
RODRIGOHP\samborms@169.254.17.218's password:
```

Siempre y cuando ambos tengan habilitadas las opciones SSH Cliente y Servidor respectivamente. Nos ingresamos desde la máquina Cliente hacia la máquina Servidor con un usuario y contraseña que tenga permitido el acceso, al acceder nos encontraremos en la carpeta del usuario, buscamos por un archivo en esa carpeta (*dir Creado_desde_HP*), veremos que no existe, entonces lo crearemos con el comando (*mkdir Creado_desde_HP*):

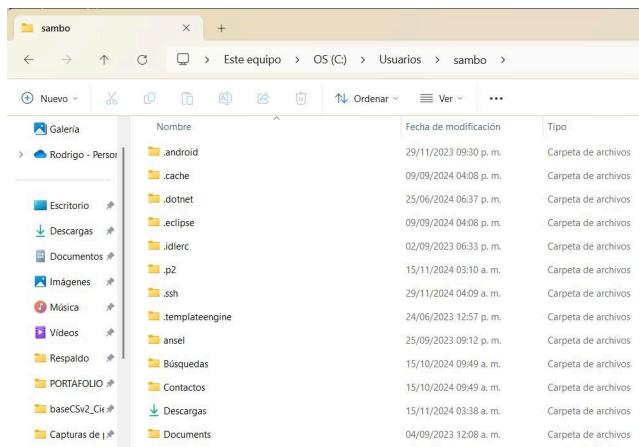
```
sambo@RODRIGOTR C:\Users\sambo>dir Creado_desde_HP
El volumen de la unidad C es OS
El número de serie del volumen es: C8FC-FCE5

Directorio de C:\Users\sambo

No se encuentra el archivo

sambo@RODRIGOTR C:\Users\sambo>mkdir Creado_desde_HP
```

Verificamos que no existe el archivo en la máquina servidor:



Tras crear el archivo verificamos su creación desde la ventana de consola de nuestra máquina Cliente:

```
Administrator: C:\WINDOWS\system32\conhost.exe

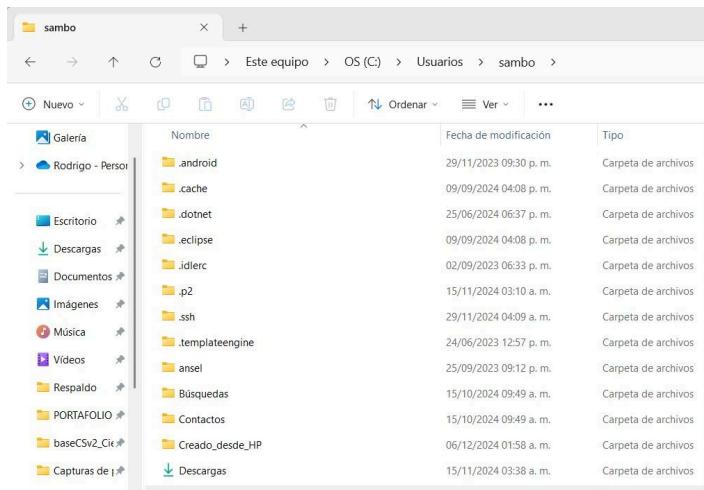
sambo@RODRIGOTR C:\Users\sambo>dir Creado_desde_HP
El volumen de la unidad C es OS
El n mero de serie del volumen es: C8FC-FCE5

Directorio de C:\Users\sambo\Creado_desde_HP

06/12/2024 01:58 a. m.    <DIR>      .
06/12/2024 01:58 a. m.    <DIR>      ..
          0 archivos            0 bytes
         2 dirs   344,079,392,768 bytes libres

sambo@RODRIGOTR C:\Users\sambo>
```

As  tambi n lo hacemos en la ventana del explorador de archivos de la m quina Servidor, (los archivos est n ordenados alfab ticamente, el archivo buscado esta en la parte inferior de la ventana) Nota: use el bot n de refrescar si no se aprecian los cambios:



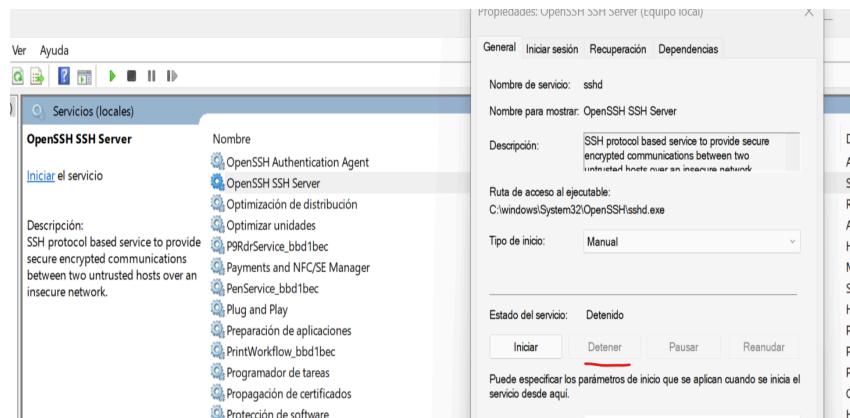
Desde la ventana de comandos de nuestro Cliente lo eliminamos:

```
sambo@RODRIGOTR C:\Users\sambo>rmdir Creado_desde_HP  
sambo@RODRIGOTR C:\Users\sambo>dir Creado_desde_HP  
El volumen de la unidad C es OS  
El n mero de serie del volumen es: C8FC-FCE5  
  
Directorio de C:\Users\sambo  
  
No se encuentra el archivo  
sambo@RODRIGOTR C:\Users\sambo>
```

Y ah  mismo comprobamos que fue eliminado (también revisamos la ventana del explorador de archivos en nuestra m quina Servidor y veremos que fue eliminado).

Importante al terminar las sesiones remotas no olvide deshabilitar las opciones de openSSH

-Desactive el servicio de la parte de server bot n *detener*:



Deshabilite la casilla de *Detecci n de dispositivos* y la de *Modo de desarrolladores (programadores)*



La configuraci n recomendada es habilitar Winrm

- Use `test-WsMan` #para verificar la conexi n en el sistema local
- `winrm quickconfig` #para configurar el sistema al protocolo Windows Remote
- `Get-Item wsman:\localhost\Client\TrustedHosts` #para verificar los Host agregados
- `Set-Item wsman:\localhost\client\trustedhosts -Value <NombreHost>` #Agregamos el Host que deseamos la lista

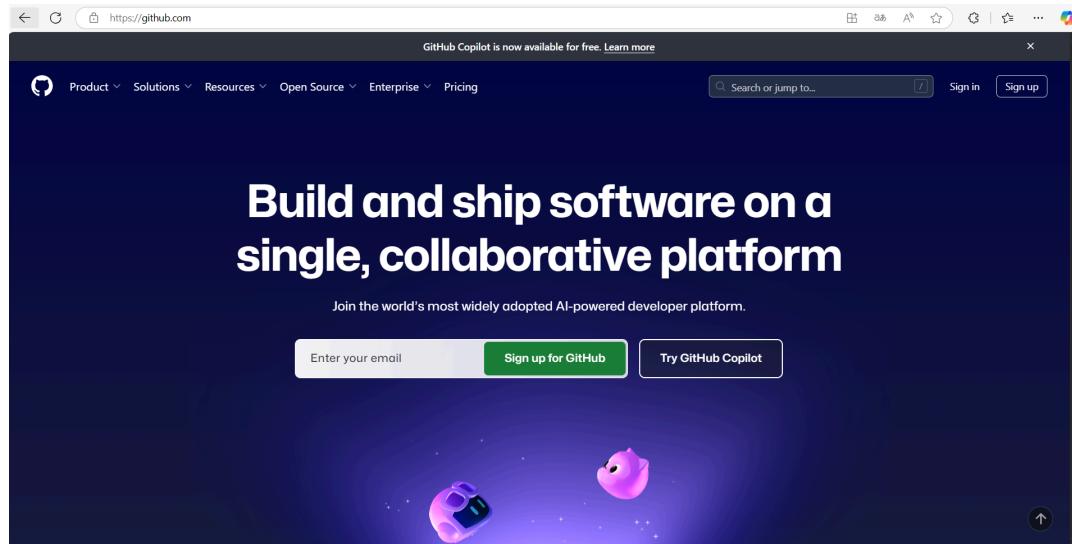
Las instrucciones se muestran en la siguiente figura:

Anexo H

GitHub

Es necesario tener un respaldo donde guardar nuestros scripts, así como un lugar donde poder compartir nuestro código, para que otros puedan usarlo, esto se logra mediante el servicio GitHub, a continuación se muestra cómo configurar una cuenta de GitHub.

Creamos una cuenta en el sitio de GitHub:



[GitHub · Build and ship software on a single, collaborative platform · GitHub](#)

Una vez creada podemos comenzar a construir los Repositorios de nuestros proyectos

A screenshot of a GitHub user profile page for "RodrigoSamboms". The profile picture is a green and white pixelated design. The user has 1 repository, 0 projects, 0 packages, and 0 stars. A "Follow" button is visible. The "Popular repositories" section shows one repository named "test1" (Public). The "Contribution activity" section shows a grid of commits for the year 2025, with a note that the user joined last month. A "Blanks or Blame" link is also present.

Creamos un repositorio de prueba como muestra de las tareas más comunes:

A screenshot of a GitHub user profile page for "RodrigoSamboms". The profile picture is a green and white pixelated design. The user has 1 repository, 0 projects, 0 packages, and 0 stars. A "Follow" button is visible. The "Popular repositories" section shows one repository named "test1" (Public). A context menu is open on the right side, listing options: "New repository", "Import repository", "New codespace", "New gist", "New organization", and "New project".

Seguimos las instrucciones de la página para crear un repositorio con nombre test1 (1), y de tipo público (*), para que cualquiera pueda acceder.

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere?
[Import a repository.](#)

Required fields are marked with an asterisk (*).

Owner * Repository name *



RodrigoTorresR

/

(1)

Great repository names are short and memorable. Need inspiration? How about [verbose-couscous](#) ?

Description (optional)

- Public Anyone on the internet can see this repository. You choose who can commit.
Private You choose who can see and commit to this repository.

Initialize this repository with:

Add a README file
This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore

.gitignore template: None

Creamos un archivo de prueba titulado README.md, con la opción *Add file*:

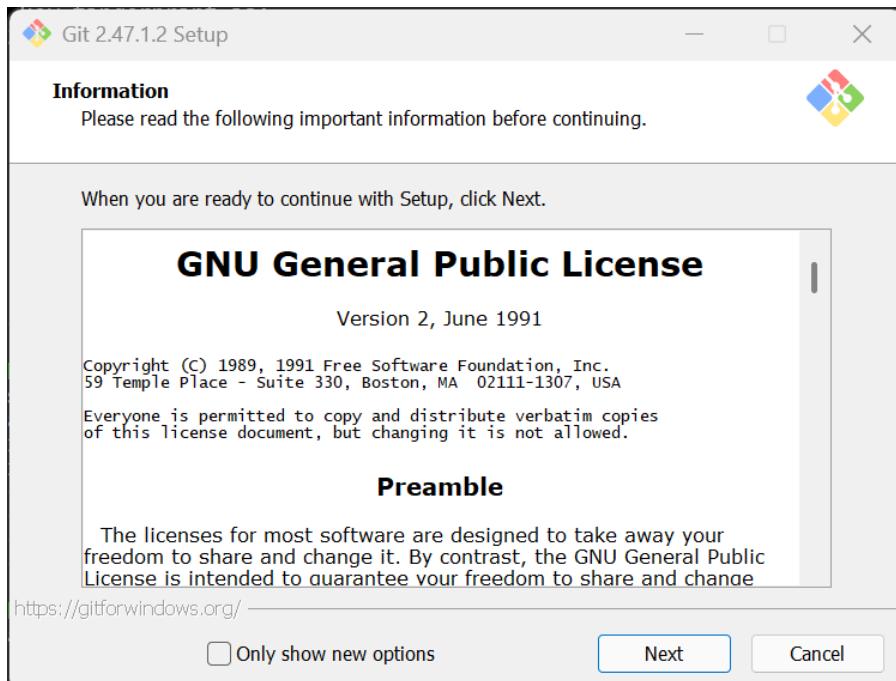
The screenshot shows a GitHub repository named "test1" which is public. The repository has 1 branch and 0 tags. A commit by "RodrigoSamboms" is shown with the message "Initial commit". In the code editor, a file named "README.md" is open. A red arrow points to the "README.md" file, and a red bracket labeled "(A)" encloses the entire code editor area. A context menu is open over the file, with a red arrow pointing to the "Create new file" option. The menu also includes "Upload files". The status bar at the bottom right shows "last month".

Podemos ver que hemos creado el archivo README.md y forma parte del repositorio (A), también podemos cargar archivos hacia nuestro repositorio. Es más común en la práctica usar comandos de CLI para realizar las tareas de GitHub, por lo que instalaremos el CLI de GitHub, GitCMD, el cual permitirá usar los comandos GitHub en otros editores de línea de texto así como software adicional para realizar las conexiones adicionales con nuestros repositorios. Para instalar Git CLI de la pagina:

[Git - Downloading Package](#)

The screenshot shows the official Git website at <https://git-scm.com/downloads/win>. The page is titled "git --distributed-even-if-your-workflow-isn't". It features a sidebar with links for "About", "Documentation", "Downloads" (selected), and "Community". The "Downloads" section is expanded, showing "GUI Clients" and "Logos". Below the sidebar, there's a note about the "Pro Git book". The main content area is titled "Download for Windows". It features a prominent orange button with the text "Click here to download" followed by "the latest (2.47.1(2)) 64-bit version of Git for Windows". Below this, it says "This is the most recent maintained build. It was released 16 days ago, on 2025-01-14.". Further down, there are sections for "Other Git for Windows downloads", "Standalone Installer", "32-bit Git for Windows Setup.", "64-bit Git for Windows Setup.", "Portable ("thumbdrive edition")", "32-bit Git for Windows Portable.", and "64-bit Git for Windows Portable.". At the bottom, there's information about the "winget tool" and the command "winget install --id Git.Git --source winget". A note states "The current source code release is version 2.48.1. If you want the newer version, you can build it from the [source code](#)".

Al descargar y ejecutar el archivo con permisos de administrador, seguimos las instrucciones del instalador:



Si no conoce el proceso de instalación de GIT CLI, recomendamos usar las opciones por Default. Una vez instalado ejecutamos la ventana de GITCMD, para generar las SSH Keys requeridas para poder realizar cambios en el repositorio, **nota:** el repositorio es público cualquiera puede descargar los archivos, pero no permite la edición, para ello debemos gestionar correctamente los permisos que permitan hacerlo.

A screenshot of a terminal window titled "MINGW64:/c/Users/sambo". The command entered is "ssh-keygen -t ed25519 -C "rodrigo.torres4234@alumnos.udg.mx"". The output shows the creation of a key pair, saving the private key to "/c/Users/sambo/.ssh/id_ed25519" and the public key to "/c/Users/sambo/.ssh/id_ed25519.pub". It also displays the key fingerprint and the raw key content in base64 format.

Generamos la SSH Key, verificamos que se haya creado correctamente y copiamos esta clave para agregarla a los usuarios que tendrán permisos de edición. En la página de inicio de nuestra cuenta de GitHub

Buscamos el apartado de Settings:

Y en la sección de SSH Keys agregamos la clave que fue creada anteriormente, así el usuario tendrá permiso para editar el repositorio. En una ventana con permisos de administrador de Powershell necesitamos activar el ssh-agent.

```

Administrator: C:\Program Files\WindowsApps\Microsoft.PowerShell_7.4.6.0_x64_8wekyb3d8bbwe\pwsh.exe
PowerShell 7.4.6

A new PowerShell stable release is available: v7.5.0
Upgrade now, or check out the release page at:
https://aka.ms/PowerShell-Release?tag=v7.5.0

PS C:\Windows\System32> Get-Service -Name ssh-agent | Set-Service -StartupType Manual
PS C:\Windows\System32> Start-Service ssh-agent
PS C:\Windows\System32>

```

En una ventana terminal windows sin permisos de administrador.

```

C:\Windows\System32\cmd.exe > + <
Microsoft Windows [Versión 10.0.26100.3037]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\sambo\Downloads>ssh-add c:/Users/sambo/.ssh/id_ed25519
Enter passphrase for c:/Users/sambo/.ssh/id_ed25519:
Identity added: c:/Users/sambo/.ssh/id_ed25519 (rodrigo.torres4234@alumnos.udg.mx)

C:\Users\sambo\Downloads>

```

Tras que hayamos creado un repositorio en github podemos clonarlo usando

The screenshot shows a GitHub repository page for 'test1'. On the right side, there is a 'Clone' section with three options: HTTPS, SSH, and GitHub CLI. The 'HTTPS' option is selected. A red arrow points to the 'SSH' option. Below the options, there is a note: 'Use a password-protected SSH key.' and a link to 'Open with GitHub Desktop'. There is also a 'Download ZIP' link.

El enlace que se describe, en este caso usamos SSH:

```
sambo@RodrigoHP MINGW64 ~/Documents/GitHub
$ git clone git@github.com:RodrigoSamboms/test1.git
Cloning into 'test1'...
The authenticity of host 'github.com (140.82.114.4)' can't be established.
ED25519 key fingerprint is SHA256:+DiY3wvvV6TuJJhbPZisF/zLDA0zPMsvHdkr4UvCoqu.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? y
Please type 'yes', 'no' or the fingerprint: yes
Warning: Permanently added 'github.com' (ED25519) to the list of known hosts.
Enter passphrase for key '/c/Users/sambo/.ssh/id_ed25519':
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (3/3), done.

sambo@RodrigoHP MINGW64 ~/Documents/GitHub
$ ls
test1/

sambo@RodrigoHP MINGW64 ~/Documents/GitHub
$ ls test1/
README.md

sambo@RodrigoHP MINGW64 ~/Documents/GitHub
$
```

Nos preguntará si deseamos conectar con un servidor cuya “fingerPrint” no pudo ser verificada, indicamos que “yes”, para clonar el repositorio. **Nota** el proceso de clonar un repositorio público no requiere habilitar conexión mediante SSH Keys.

También podemos realizar las mismas acciones en la terminal de PowerShell

```
(env) PS C:\Users\sambo\Documents\Programacion> dir
    Directory: C:\Users\sambo\Documents\Programacion

Mode                LastWriteTime         Length Name
----                -----          -----   -----
d----        18/02/2025 12:48 p. m.           C_languaje
d----        19/02/2025 04:58 a. m.           Python
d----        25/02/2025 06:42 a. m.          test1

(env) PS C:\Users\sambo\Documents\Programacion> git clone git@github.com:RodrigoSamboms/test1.git
Cloning into 'test1'...
remote: Enumerating objects: 6, done.
remote: Counting objects: 100% (6/6), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 6 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (6/6), done.

(env) PS C:\Users\sambo\Documents\Programacion> ls
    Directory: C:\Users\sambo\Documents\Programacion

Mode                LastWriteTime         Length Name
----                -----          -----   -----
d----        18/02/2025 12:48 p. m.           C_languaje
d----        19/02/2025 04:58 a. m.           Python
d----        25/02/2025 06:42 a. m.          test1

(env) PS C:\Users\sambo\Documents\Programacion> dir
```

Y dentro de un ambiente virtual de Python, lo cual nos brinda más flexibilidad.

Mostramos el contenido del archivo README.md antes de modificarlo y después de haberlo hecho, para poder mostrar el uso del comando `git status`

```
(env) PS C:\Users\sambo\Documents\Programacion\test1> vim .\README.md
(env) PS C:\Users\sambo\Documents\Programacion\test1> cat .\README.md
# test1
this is second commit

##sub commit to illustrate the use of github

(env) PS C:\Users\sambo\Documents\Programacion\test1> vim .\README.md
(env) PS C:\Users\sambo\Documents\Programacion\test1> cat .\README.md
# test1
this is second commit

##Realizamos un cambio

(env) PS C:\Users\sambo\Documents\Programacion\test1>
```

El cual podemos ver que nos indica si hubo cambios o no en la carpeta `test1` que contiene nuestro repositorio, también hemos añadido un nuevo archivo `index.html` :

Mode	LastWriteTime	Length	Name
-a---	25/02/2025 06:56 a. m.	1819	.index.html.un~
-a---	25/02/2025 06:51 a. m.	3330	.README.md.un~
-a---	25/02/2025 06:44 a. m.	81	}
-a---	25/02/2025 06:56 a. m.	18	index.html
-a---	25/02/2025 06:51 a. m.	60	README.md
-a---	25/02/2025 06:51 a. m.	81	README.md~

```
(env) PS C:\Users\sambo\Documents\Programacion\test1> cat .\index.html
<div>HELLO</div>
(env) PS C:\Users\sambo\Documents\Programacion\test1> git status
On branch main
Your branch is up to date with 'origin/main'.

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          (A)

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    .README.md.un~
    .index.html.un~
    README.md~
    index.html
    }

no changes added to commit (use "git add" and/or "git commit -a")
(env) PS C:\Users\sambo\Documents\Programacion\test1>
```

La sección (A) muestra los archivos que fueron modificados y la sección (B) archivos que han sido agregados. Agregamos los archivos con cambios para su actualización usando el comando `git add <parámetros>`, en este caso usamos `< . >` para indicar que se incluirá todos los archivos de la carpeta.

```
(env) PS C:\Users\sambo\Documents\Programacion\test1> git add .
(env) PS C:\Users\sambo\Documents\Programacion\test1> git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:  .README.md.un~
    new file:  .index.html.un~
    modified: README.md
    new file: README.md~
    new file: index.html
    new file:  }

(env) PS C:\Users\sambo\Documents\Programacion\test1>
```

Podemos ver que los archivos han cambiado de color debido a que los incluimos para la actualización del repositorio, están listos para ser etiquetados para la actualización del repositorio mediante el comando `git commit -m "<Mensaje justificando el cambio>"`

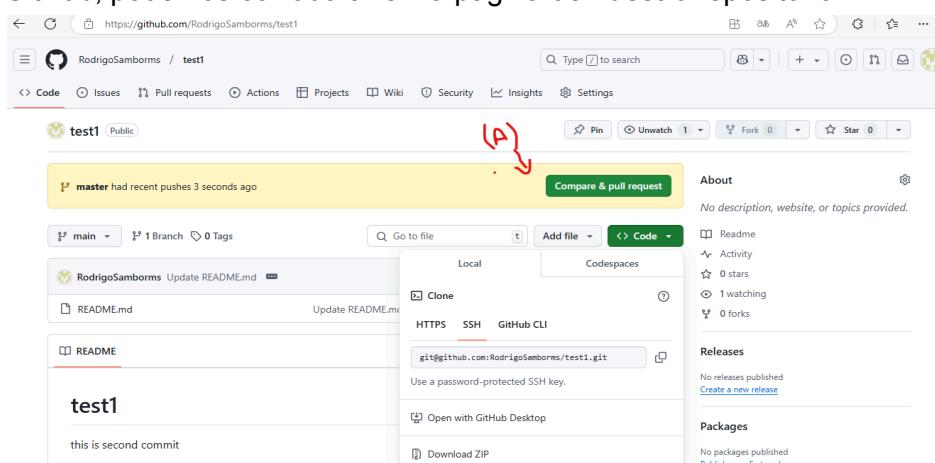
```
(env) PS C:\Users\sambo\Documents\Programacion\test1> git commit -m "added index.html file"
[main b384504] added index.html file
 6 files changed, 14 insertions(+)
 create mode 100644 .README.md.un~
 create mode 100644 .index.html.un~
 create mode 100644 README.md~
 create mode 100644 README.md~
 create mode 100644 index.html
 create mode 100644 }

(env) PS C:\Users\sambo\Documents\Programacion\test1> #necesita configurar su clave privada y publica para poder completar la actualizacion"
```

Para enviar los archivos, debemos especificar si el cambio será a la línea “Máster” o alguna “Branche” (ramificación), esto para un mejor control de versiones.

```
(env) PS C:\Users\sambo\Documents\Programacion\test1> git branch -M master
(env) PS C:\Users\sambo\Documents\Programacion\test1> git push origin master
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 8 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (7/7), 1.04 KiB | 176.00 KiB/s, done.
Total 7 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), done.
remote:
remote: Create a pull request for 'master' on GitHub by visiting:
remote:   https://github.com/RodrigoSamboms/test1/pull/new/master
remote:
To github.com:RodrigoSamboms/test1.git
 * [new branch]      master -> master
(env) PS C:\Users\sambo\Documents\Programacion\test1>
```

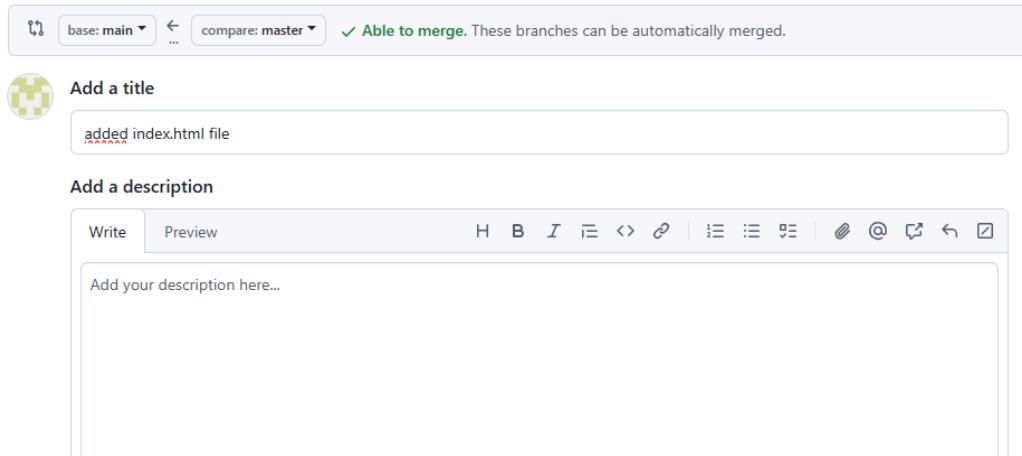
Utilizando el comando `git push origin master`, enviamos los cambios al repositorio de GitHub, podemos corroborar en la página de nuestro repositorio:



Presionando en (A) nos mostrará los comentarios de los cambios realizados:

Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also compare across forks. [Learn more about diff commits](#)



Podemos constatar que el mensaje es el mismo que escribirnos al usar *commit*

Ahora tenemos un proyecto nuevo con varios archivos de python:

```
Directory: C:\Users\sambo\Documents\Programacion\Python\Flask
Mode           LastWriteTime      Length Name
----           -----          ---- 
d----
```

(A)

```
25/02/2025 02:40 a. m.          58831 __pycache__
d----
```

(B)

```
19/02/2025 04:58 a. m.          523 env
d----
```

(C)

```
25/02/2025 04:58 a. m.          2269 instance
d----
```

(D)

```
25/02/2025 01:41 a. m.          1910 static
d----
```

(E)

```
25/02/2025 05:17 a. m.          1910 templates
-a---
```

(F)

```
25/02/2025 04:51 a. m.          58831 .app.py.un~
-a---
```

(G)

```
25/02/2025 05:21 a. m.          523 .Procfile.un~
-a---
```

(H)

```
25/02/2025 07:21 a. m.          2269 .README.md.un~
-a---
```

(I)

```
25/02/2025 04:51 a. m.          1910 app.py
-a---
```

(J)

```
25/02/2025 04:49 a. m.          1910 app.py~
-a---
```

(K)

```
25/02/2025 05:21 a. m.          22 Procfile
-a---
```

(L)

```
25/02/2025 07:21 a. m.          87 README.md
-a---
```

(M)

```
25/02/2025 05:53 a. m.          59 README.md~
-a---
```

(N)

```
25/02/2025 05:02 a. m.          446 requirements.txt
-a---
```

(O)

```
25/02/2025 03:04 a. m.          8192 test.db
-a---
```

(P)

```
25/02/2025 04:13 a. m.          709 update.html

(env) PS C:\Users\sambo\Documents\Programacion\Python\Flask> git init
Reinitialized existing Git repository in C:/Users/sambo/Documents/Programacion/Python/Flask/.git/
(env) PS C:\Users\sambo\Documents\Programacion\Python\Flask> git status
On branch main
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.un~
      modified:   README.md
      modified:   README.md~

no changes added to commit (use "git add" and/or "git commit -a")
(env) PS C:\Users\sambo\Documents\Programacion\Python\Flask> git add .
(env) PS C:\Users\sambo\Documents\Programacion\Python\Flask> git status
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    modified:   .README.md.un~
    modified:   README.md
    modified:   README.md~

(env) PS C:\Users\sambo\Documents\Programacion\Python\Flask>
```

- (A) nos muestra el contenido de la carpeta del proyecto, el cual marcamos para un nuevo repositorio con *git init*
- (B) revisamos con *git status* cuales archivos han sufrido cambios
- (C) Para este ejemplo agregamos todos los archivos a la actualización del repositorio con *git add*.
- (D) revisamos que los archivos hayan sido agregados mediante *git status* nuevamente

Si intentamos “subir” los cambios con `git push origin main`, veremos un error: **nota** este repositorio tiene como principal rama **main**

```
(env) PS C:\Users\sambo\Documents\Programacion\Python\Flask> git push origin main
info: please complete authentication in your browser...
remote: Repository not found.
fatal: repository 'https://github.com/RodrigoTorresR/TestFlask/' not found
(env) PS C:\Users\sambo\Documents\Programacion\Python\Flask> _
```

Debido a que no existe ningún repositorio con el nombre de “Flask”, una forma sencilla es ir a la página de nuestra cuenta de GitHub y crear el repositorio.

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Required fields are marked with an asterisk (*).

Owner * Repository name *

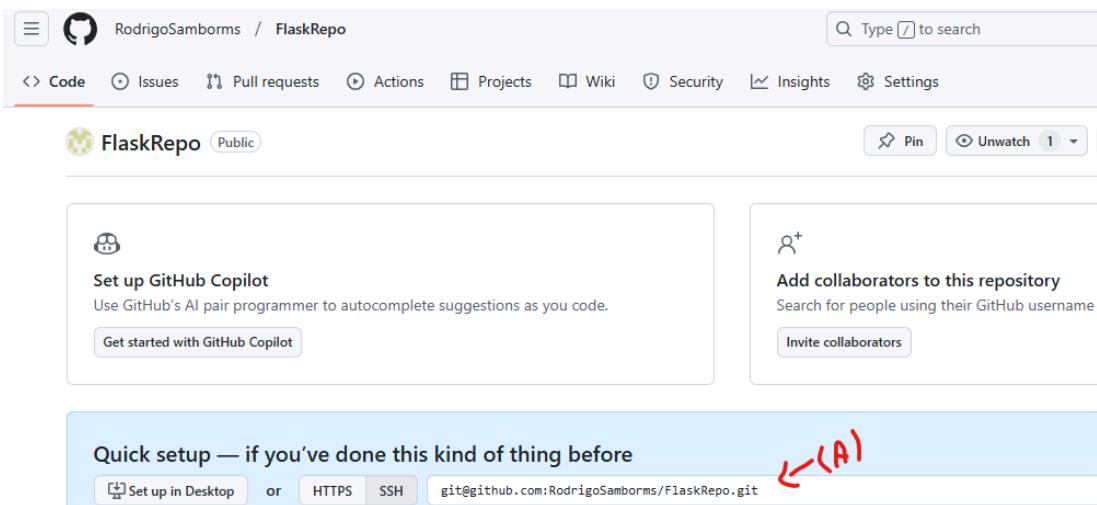
 RodrigoSamboms /
 FlaskRepo is available.

Great repository names are short and memorable. Need inspiration? How about [urban-adventure](#) ?

Description (optional)

Repositorio de PythonFlask

Con el nuevo repositorio “FlaskRepo”:



The screenshot shows the GitHub repository page for "FlaskRepo". The top navigation bar includes links for Code, Issues, Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings. The repository name "FlaskRepo" is displayed with a "Public" badge. Below the header, there are two main callout boxes: one for "Set up GitHub Copilot" and another for "Add collaborators to this repository". A red arrow labeled "(A)" points to the "Get started with GitHub Copilot" button. At the bottom, there is a "Quick setup — if you've done this kind of thing before" section with options for "Set up in Desktop" (using a QR code), "HTTPS", "SSH", and a copy link "git@github.com:RodrigoSamboms/FlaskRepo.git".

Regresamos a nuestra terminal para completar el proceso tomando la referencia (A):

```
(env) PS C:\Users\sambo\Documents\Programacion\Python\Flask> git remote add main git@github.com:RodrigoSamboms/FlaskRepo.git
(env) PS C:\Users\sambo\Documents\Programacion\Python\Flask> git remote -v
main    git@github.com:RodrigoSamboms/FlaskRepo.git (fetch) ← (B)
main    git@github.com:RodrigoSamboms/FlaskRepo.git (push)
origin  https://github.com/RodrigoTorresR/TestFlask (fetch)
origin  https://github.com/RodrigoTorresR/TestFlask (push)
(env) PS C:\Users\sambo\Documents\Programacion\Python\Flask> git push main ← (C)
Enumerating objects: 39, done.
Counting objects: 100% (39/39), done.
Delta compression using up to 8 threads
Compressing objects: 100% (33/33), done.
Writing objects: 100% (39/39), 16.56 KiB | 1.51 MiB/s, done.
Total 39 (delta 14), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (14/14), done.
To github.com:RodrigoSamboms/FlaskRepo.git
 * [new branch]      main -> main
(env) PS C:\Users\sambo\Documents\Programacion\Python\Flask>
```

- (A) indicamos con `git remote add main <referencia al repo>`
(B) revisamos cuales referencias se han hecho podemos ver que Main esta correctamente referenciado a un repositorio que si existe
(C) usamos `git push main` para enviar los archivos al repositorio y estos estaran en la “Rama” (branch) main (principal).

Verificamos en nuestra página del repositorio:

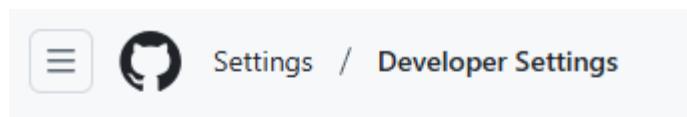
The screenshot shows a GitHub repository named 'FlaskRepo' (Public). The main branch is selected, showing 1 Branch and 0 Tags. There are 4 commits listed, all made by 'RodrigoSamboms' 2 hours ago, initialized the app. The commits include: __pycache__, instance, static/CSS, templates, .Procfile.un~, .README.md.un~, .app.py.un~, Procfile, README.md, README.md~, app.py, and app.py~.

Commit	Message	Date
RodrigoSamboms	initialize the app	0ea0e02 · 2 hours ago
__pycache__	init app	2 hours ago
instance	init app	2 hours ago
static/CSS	init app	2 hours ago
templates	init app	2 hours ago
.Procfile.un~	init app	2 hours ago
.README.md.un~	initialize the app	2 hours ago
.app.py.un~	init app	2 hours ago
Procfile	init app	2 hours ago
README.md	initialize the app	2 hours ago
README.md~	initialize the app	2 hours ago
app.py	init app	2 hours ago
app.py~	init app	2 hours ago

El repositorio fue cargado con los archivos correctamente.

Creando Tokens

En su cuenta de GitHub sección Settings\Developer Settings, encontrara la opción *Personal access tokens*



Escoja de la sección el token clasico:

A screenshot of the "Personal access tokens" page. It shows a button "Generate new token" with two options: "Generate new token (Beta)" and "Generate new token (classic)". A red arrow labeled "(A)" points to the "Generate new token" button. Another red arrow labeled "(B)" points to the "Generate new token (classic)" option. Below the button, there is a message: "No personal access token created. Need an API token for scripts or testing? Generate a personal access token for quick access to the GitHub API." A key icon is also present.

Agregue un nombre al token para su identificación, y muy importante (*) el tiempo de vida del token asi como (**) los permisos otorgados al usuario del token:

New personal access token (classic)

Personal access tokens (classic) function like ordinary OAuth access tokens. They can be used instead over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#).

Note

Provisional

What's this token for?

Expiration * ← (x)

Custom... ▾

26/02/2025

Select scopes

Scopes define the access for personal tokens. [Read more about OAuth scopes](#).

<input checked="" type="checkbox"/> repo	← (x) *	Full control of private repositories
<input checked="" type="checkbox"/> repo:status		Access commit status
<input checked="" type="checkbox"/> repo_deployment		Access deployment status
<input checked="" type="checkbox"/> public_repo		Access public repositories
<input checked="" type="checkbox"/> repo:invite		Access repository invitations
<input checked="" type="checkbox"/> security_events		Read and write security events
<input checked="" type="checkbox"/> workflow		Update GitHub Action workflows

Aseguramos con `git add .` y `git status`, que los archivos estén incluidos en la actualización.

Con el nuevo token creamos un punto de acceso mediante el comando:

```
git remote add main https://[TOKEN]@github.com/[REPO-OWNER]/[REPO-NAME]
```

```
error: remote origin already exists.  
(env) PS C:\Users\sambo\Documents\Programacion\Python\Flask> git remote add master https://ghp_CLPdXmdawnZhtG4gCxHb03uwF0Rson4VXzw3@github.com/RodrigoTorresR/TestFlask.git  
main  
git@github.com:RodrigoSamboms/FlaskRepo.git (fetch)  
main git@github.com:RodrigoSamboms/FlaskRepo.git (push)  
master https://ghp_CLPdXmdawnZhtG4gCxHb03uwF0Rson4VXzw3@github.com/RodrigoTorresR/TestFlask.git (fetch)  
master https://ghp_CLPdXmdawnZhtG4gCxHb03uwF0Rson4VXzw3@github.com/RodrigoTorresR/TestFlask.git (push)  
origin https://github.com/RodrigoTorresR/TestFlask (fetch)  
origin https://github.com/RodrigoTorresR/TestFlask (push)  
(env) PS C:\Users\sambo\Documents\Programacion\Python\Flask> git push master  
Enumerating objects: 39, done.  
Counting objects: 100% (39/39), done.  
Delta compression using up to 8 threads  
Compressing objects: 100% (33/33), done.  
Writing objects: 100% (39/39), 16.56 KiB | 1.66 MiB/s, done.  
Total 39 (delta 14), reused 0 (delta 0), pack-reused 0 (from 0)  
remote: Resolving deltas: 100% (14/14), done.  
To https://github.com/RodrigoTorresR/TestFlask.git  
 * [new branch] main -> main  
(env) PS C:\Users\sambo\Documents\Programacion\Python\Flask>
```

En este caso la sintaxis fue:

```
git remote add main
```

https://ghp_CLPdXmdawnZhtG4gCxHb03uwF0Rson4VXzw3@github.com/RodrigoTorresR/TestFlask.git

Revisamos con `git remote -v` que la entrada se haya creado, usamos `git push master` para subir los archivos a la “rama” “master”.

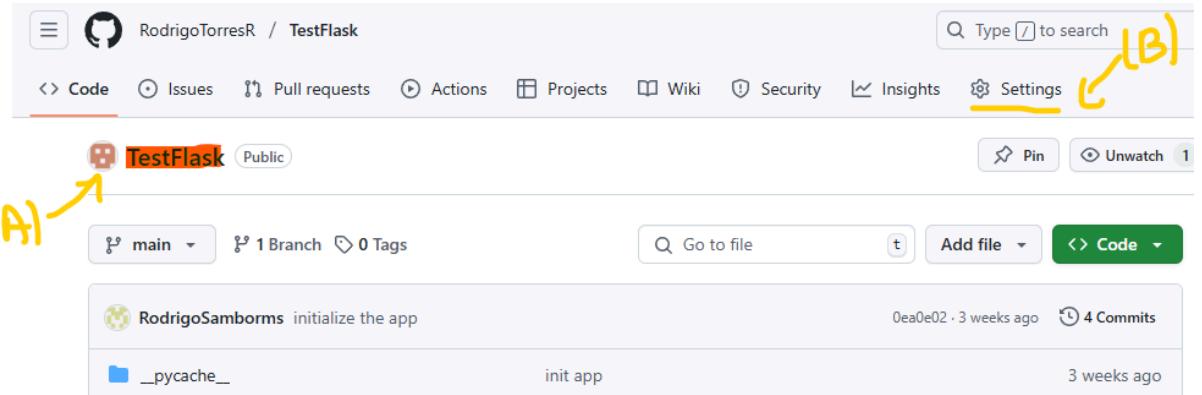
The screenshot shows the GitHub repository page for 'TestFlask'. At the top, there's a navigation bar with links for Code, Issues, Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings. Below the navigation bar, the repository name 'TestFlask' is displayed along with its status as 'Public'. There are buttons for Pin, Unwatch (with 1 subscriber), and Fork (with 0 forks). Underneath, there's a summary showing 'main' branch, '1 Branch', and '0 Tags'. A search bar and an 'Add file' button are also present. The main content area displays a list of commits from 'RodrigoSamboms'. The first commit is 'initialize the app' made 2 hours ago. Subsequent commits include 'init app' for files like '__pycache__', 'instance', 'static/CSS', 'templates', '.Profile.un~', '.README.md.un~', '.app.py.un~', and 'Procfile', all made 3 hours ago. On the right side of the page, there are sections for 'About', 'Tutorial de Flask', 'Readme', 'Activity', 'Stars', 'Watching', 'Forks', 'Releases', 'Create a new release', 'Packages', and 'Publish your first package'.

Corroboramos en nuestra página de GitHub que los archivos se hayan subido adecuadamente, **nota** hemos usado para el ejemplo con Token otra cuenta y asu vez hemos usado un Token con una vida útil de 1 día.

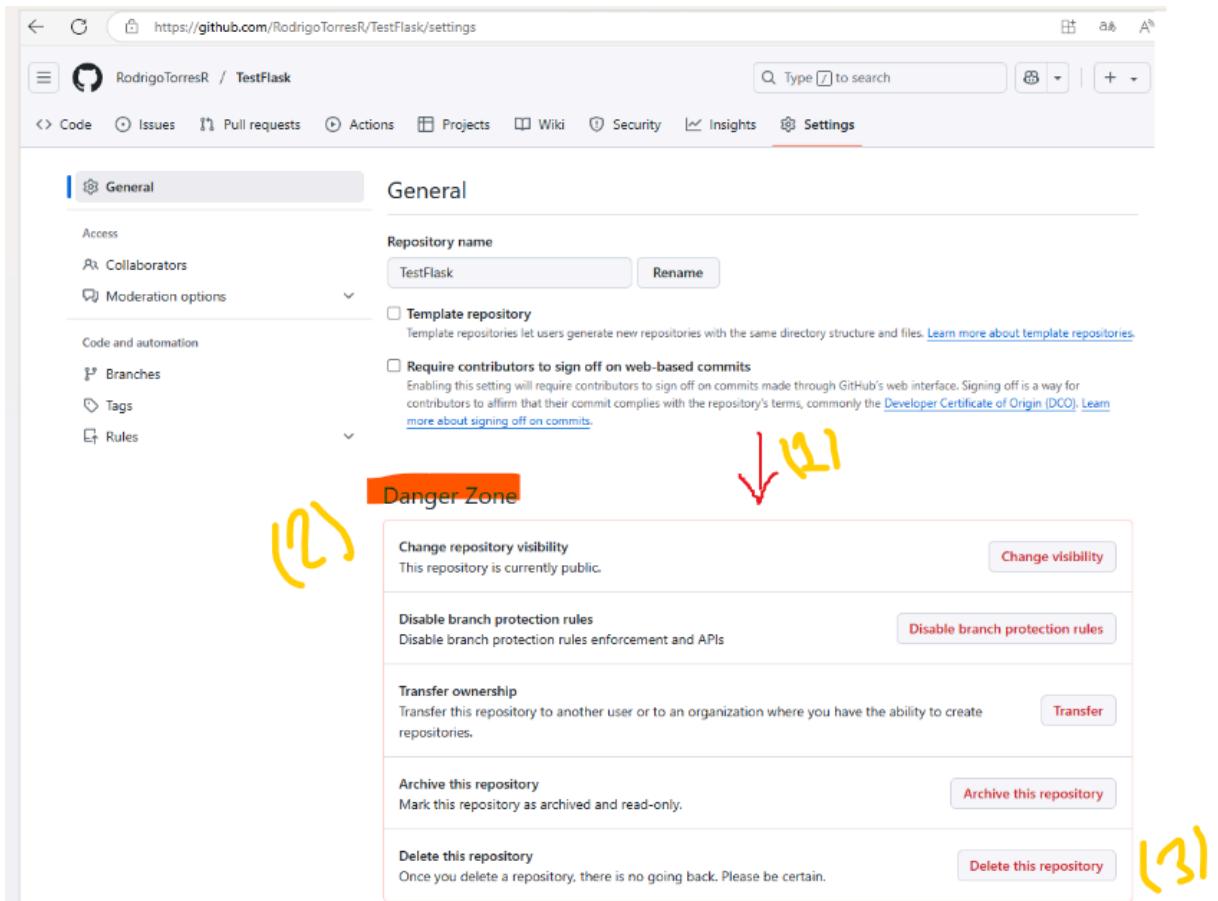
Removiendo repositorio de GitHub

En su cuenta de GitHub:

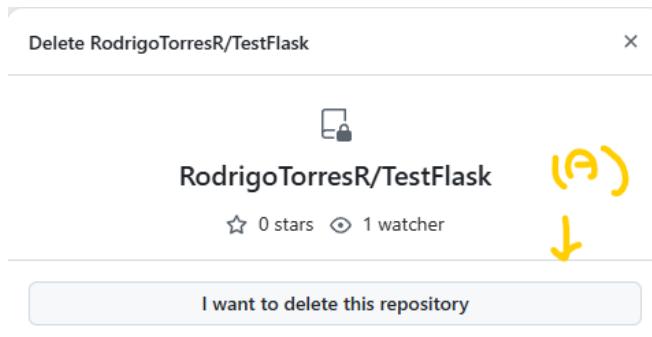
- Ubíquese en el repositorio que se desea remover (A) y presione Settings bajo el nombre del repositorio repositorio (B)



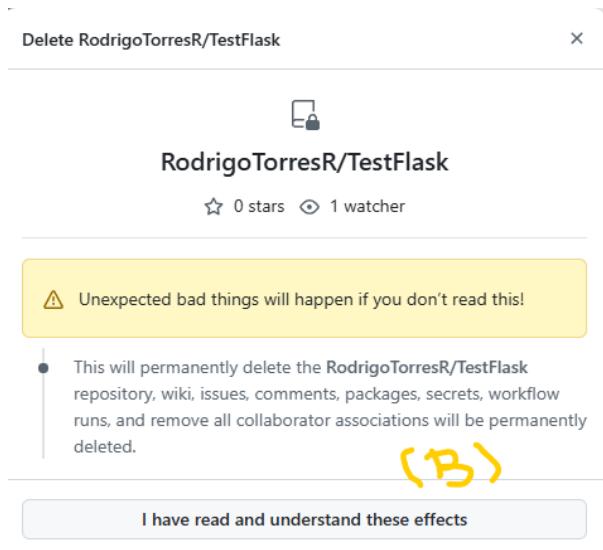
- Muévase hacia (1)Danger Zone section (2), presione Delete (3) para eliminar este repositorio



- Confirme que desea remover el repositorio (A)



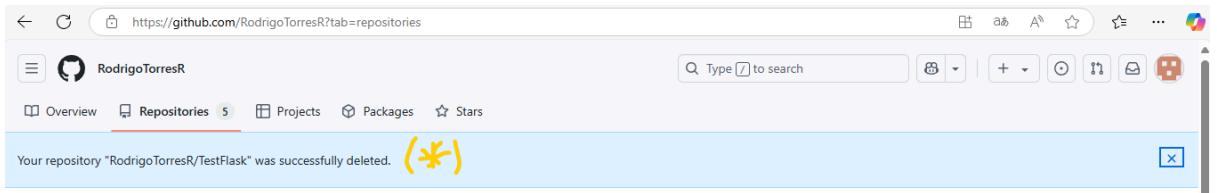
- Read the warnings and click I have read and understand these effects



- Teclee el nombre del repositorio que desea eliminar)1), presione “Delete this repository” (2)



- Confirme la eliminación del repositorio (*)



Removiendo repositorio local (clonado en la computadora local)

Simplemente elimine la carpeta y su contenido, vuelva a clonar el repositorio

```
PS C:\Users\sambo\Programacion> dir

Directory: C:\Users\sambo\Programacion

Mode                LastWriteTime         Length Name
----              -----        ----
d----       18/03/2025 12:26 a. m.          FlaskRepo
d----       17/03/2025 11:45 p. m.          Github
d----       06/03/2025 04:14 a. m.          Lenguaje_C
-a---      05/03/2025 01:56 a. m.      177588 CircularDoubleLinkedList.c.un~
-a---      05/03/2025 12:39 a. m.      29116 CircularDoubleLinkedList.c.un~
-a---      05/03/2025 01:56 a. m.      5754 CircularDoubleLinkedList.c
-a---      05/03/2025 01:50 a. m.      5754 CircularDoubleLinkedList.c~
-a---      05/03/2025 12:44 a. m.      40856 CircularDoubleLinkedList.exe
-a---      05/03/2025 12:32 a. m.      596 CircularDoubleLinkedList.c~

PS C:\Users\sambo\Programacion> Remove-Item -Recurse .\FlaskRepo\
PS C:\Users\sambo\Programacion> dir

Directory: C:\Users\sambo\Programacion

Mode                LastWriteTime         Length Name
----              -----        ----
d----       17/03/2025 11:45 p. m.          Github
d----       06/03/2025 04:14 a. m.          Lenguaje_C
-a---      05/03/2025 01:56 a. m.      177588 CircularDoubleLinkedList.c.un~
-a---      05/03/2025 12:39 a. m.      29116 CircularDoubleLinkedList.c.un~
-a---      05/03/2025 01:56 a. m.      5754 CircularDoubleLinkedList.c
-a---      05/03/2025 01:50 a. m.      5754 CircularDoubleLinkedList.c~
-a---      05/03/2025 12:44 a. m.      40856 CircularDoubleLinkedList.exe
-a---      05/03/2025 12:32 a. m.      596 CircularDoubleLinkedList.c~

PS C:\Users\sambo\Programacion> git clone https://github.com/RodrigoTorresR/FlaskRepo.git
Cloning into 'FlaskRepo'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 3 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (3/3), done.
PS C:\Users\sambo\Programacion> dir

Directory: C:\Users\sambo\Programacion

Mode                LastWriteTime         Length Name
----              -----        ----
d----       18/03/2025 12:28 a. m.          FlaskRepo
d----       17/03/2025 11:45 p. m.          Github
d----       06/03/2025 04:14 a. m.          Lenguaje_C
-a---      05/03/2025 01:56 a. m.      177588 CircularDoubleLinkedList.c.un~
-a---      05/03/2025 12:39 a. m.      29116 CircularDoubleLinkedList.c.un~
```

Verifique que los archivos se cargan nuevamente:

```

PS C:\Users\sambo\Programacion> dir

Directory: C:\Users\sambo\Programacion

Mode                LastWriteTime        Length Name
<--->
d----       18/03/2025 12:28 a. m.          FlaskRepo
d----       17/03/2025 11:45 p. m.          Github
d----       06/03/2025 04:14 a. m.          Lenguaje_C
-a---      05/03/2025 01:56 a. m.        177588 CircularDoubleLinkedList.c.un~
-a---      05/03/2025 12:39 a. m.        29116 .CircularDoubleLinkedList.c.un~
-a---      05/03/2025 01:56 a. m.         5754 CircularDoubleLinkedList.c
-a---      05/03/2025 01:50 a. m.         5754 CircularDoubleLinkedList.c~
-a---      05/03/2025 12:44 a. m.        40856 CircularDoubleLinkedList.exe
-a---      05/03/2025 12:32 a. m.         596 CircularDoubleLinkedList.c~

PS C:\Users\sambo\Programacion> dir .\FlaskRepo\

Directory: C:\Users\sambo\Programacion\FlaskRepo

Mode                LastWriteTime        Length Name
<--->
-a---      18/03/2025 12:28 a. m.          23 README.md

PS C:\Users\sambo\Programacion> cat .\FlaskRepo\README.md
Repositorio de prueba
PS C:\Users\sambo\Programacion>

```

Recuperando archivos específicos a nuestro repositorio local

Puede haber casos en los cuales nosotros perdamos algún archivo o varios de nuestro proyecto, como muestra la siguiente figura:

```

PS C:\Users\sambo\Programacion\Github\Python\DataStructures\LinkedQueue> Get-ChildItem -Path .\

Directory: C:\Users\sambo\Programacion\Github\Python\DataStructures\LinkedQueue

Mode                LastWriteTime        Length Name
<--->
-a---      19/03/2025 08:09 a. m.        5680 CircularQueue.py.un~
-a---      19/03/2025 07:17 p. m.        38173 LinkedQueue.py.un~
-a---      19/03/2025 08:09 a. m.        2102 CircularQueue.py
-a---      19/03/2025 08:08 a. m.        2103 CircularQueue.py~
-a---      19/03/2025 07:40 p. m.        1780 LinkedQueue.py
-a---      19/03/2025 08:09 a. m.        2102 LinkedQueue.py~

PS C:\Users\sambo\Programacion\Github\Python\DataStructures\LinkedQueue> Remove-Item .\linkedQueue.py
PS C:\Users\sambo\Programacion\Github\Python\DataStructures\LinkedQueue> Get-ChildItem -Path .\

Directory: C:\Users\sambo\Programacion\Github\Python\DataStructures\LinkedQueue

Mode                LastWriteTime        Length Name
<--->
-a---      19/03/2025 08:09 a. m.        5680 CircularQueue.py.un~
-a---      19/03/2025 07:17 p. m.        38173 LinkedQueue.py.un~
-a---      19/03/2025 08:09 a. m.        2102 CircularQueue.py
-a---      19/03/2025 08:08 a. m.        2103 CircularQueue.py~
-a---      19/03/2025 08:09 a. m.        2102 LinkedQueue.py~

```

Hemos removido el archivo “linkedQueue.py” intencionalmente, usando el comando Git status, podemos ver que se ha detectado un cambio en el repositorio:

```

PS C:\Users\sambo\Programacion\Github\Python\DataStructures\LinkedQueue> git status
On branch main
Your branch is up to date with 'origin/main'.

Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
    (use "git restore <file>..." to discard changes in working directory)
      deleted:    LinkedQueue.py

no changes added to commit (use "git add" and/or "git commit -a")

```

Utilizamos el comando `git log --oneline` para poder listar los commits que hemos hecho en nuestro repositorio:

```
PS C:\Users\sambo\Programacion\Github\Python\DataStructures\LinkedQueue> git log --oneline
21a1cfc (HEAD -> main, origin/main) agregando mas estructuras de datos
bffab61 Update README.md
b4b1a83 Agregamos una carpeta para los ejemplos de Structuras de Datos usando OPP
e544fa6 Creamos el archivo de requerimientos de paquetes para nuestra aplicacion de Python
d87f91f Creamos el archivo de código para el programa Matrix de Verificación
a4f4b0d Creamos la carpeta para el programa de Matrix de Verificación
```

Aquí podemos ver la importancia de escribir comentarios significativos que no indiquen que cambios se hicieron a nuestro repositorio, para nuestro caso buscamos el ID del commit en el cual nuestro archivo perdido está presente, y lo usaremos para recuperar el archivo eliminado:

```
PS C:\Users\sambo\Programacion\Github\Python\DataStructures\LinkedQueue> git checkout 21a1cfc .\
Updated 1 path from ce2d2f6
PS C:\Users\sambo\Programacion\Github\Python\DataStructures\LinkedQueue> Get-ChildItem -Path .\

Directory: C:\Users\sambo\Programacion\Github\Python\DataStructures\LinkedQueue

Mode          LastWriteTime      Length Name
----          <-----           ----- 
-a---  19/03/2025 08:09 a. m.    5680 .CircularQueue.py.un~
-a---  19/03/2025 07:17 p. m.    38173 .LinkedQueue.py.un~
-a---  19/03/2025 08:09 a. m.    2102 CircularQueue.py
-a---  19/03/2025 08:08 a. m.    2103 CircularQueue.py~
-a---  19/03/2025 07:43 p. m.    1780 LinkedQueue.py
-a---  19/03/2025 08:09 a. m.    2102 LinkedQueue.py~
```

El comando `git checkout <id commit> <path>` requiere que especifiquemos el identificador del commit y la ubicación en el repositorio local donde deseamos copiar los archivos recuperados, tras la ejecución verificamos que se haya recuperado el archivo con éxito.

Anexo I

Ambientes virtuales con Python 2.7

En la página de Python busca por una versión de python 2.7.11 (esta versión ya viene con PIP) la cual permite descargar la paquetería de ambientes virtuales (y usar el script de inicialización mencionado más adelante), se recomienda usar un instalador de Windows.

Python 2.7.11

Release Date: Dec. 5, 2015

Python 2.7.11 is a bugfix release of the Python 2.7 series.

Full ChangeLog

Files

Version	Operating System	Description	MD5 Sum	File Size	GPG
Gzipped source tarball	Source release		6bd076ec9e93f05dd63e47eb9c15728b	16.1 MB	SIG
XZ compressed source tarball	Source release		1dbc0cc48bdcd0399a8199d000f9fb232	11.7 MB	SIG
Mac OS X 32-bit/PPC Installer	macOS	for Mac OS X 10.5 and later	8d563a63b261fc3068c101471442b601	22.9 MB	SIG
Mac OS X 64-bit/32-bit Installer	macOS	for Mac OS X 10.6 and later	cac08bbfa09c5a5c0fe19f68440cf1f0	21.1 MB	SIG
Windows help file	Windows		0d8044f1d0197c3381be0789c2dfcc98	5.9 MB	SIG
Windows debug information files	Windows		b5ebe6703d69ee97d1d648d20df6ee55	23.2 MB	SIG
Windows debug information files for 64-bit binaries	Windows		34b3e9342b7a9dd58e0f20c6108e72e6	23.9 MB	SIG
Windows x86 MSI Installer	Windows		241bf8e097ab4e1047d9b4f59602095	17.8 MB	SIG

Tras la instalación vaya a la carpeta donde se ubican los archivos de Python 2.7:

```
PS C:\Python27> .\python.exe
Python 2.7.11 (v2.7.11:6d1b6a68f775, Dec  5 2015, 20:40:30) [MSC v.1500 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> exit()
PS C:\Python27>
```

Use el ejecutable de la carpeta para lanzar python 2 para verificar que funcione correctamente (.\\python.exe # asegúrese de estar en la carpeta de la instalación).

Creamos un perfil personalizado de PowerShell

La ubicación de nuestro archivo de perfil, se muestra tras la creación de nuestro perfil

```
PS C:\Users\sambo> New-Item -Type file -Force $profile

    Directorio: C:\Users\sambo\OneDrive\Documentos\WindowsPowerShell

Mode          LastWriteTime      Length Name
----          -----          ----  --
-a---        12/03/2025 03:34 a. m.       0 Microsoft.PowerShell_profile.ps1

PS C:\Users\sambo> cd C:\Users\sambo\OneDrive\Documentos\WindowsPowerShell
PS C:\Users\sambo\OneDrive\Documentos\WindowsPowerShell> dir

    Directorio: C:\Users\sambo\OneDrive\Documentos\WindowsPowerShell

Mode          LastWriteTime      Length Name
----          -----          ----  --
da---l      30/10/2024 01:25 a. m.      Scripts
-a---l      12/03/2025 03:34 a. m.       0 Microsoft.PowerShell_profile.ps1

PS C:\Users\sambo\OneDrive\Documentos\WindowsPowerShell> vim .\Microsoft.PowerShell_profile.ps1
```

(Note que para este caso esta ubicado en OneDrive, por lo que podría afectar otros equipos que tengan acceso al archivo compartido)

Usamos un editor de texto simple (para este caso Vim) para agregar nuestros “alias” para Python3 y Python2:

```
PS C:\> Microsoft.PowerShell_profile.ps1 <+>
Set-Alias python3 "C:\Users\sambo\AppData\Local\Programs\Python\Python313\python.exe"
Set-Alias python2 "C:\Python27\python.exe"
```

Agregamos la llamada al archivo ejecutable absoluta, para cada versión de Python. Antes de comenzar con la instalación de los paquetes, es recomendable verificar que Pip este actualizado, usamos `python2 -m pip list` (note que python2 es el alias si no cuenta con uno tendrá que llamar a python con la dirección absoluta).

```
PS C:\> python2 -m pip list
DEPRECATION: Python 2.7 will reach the end of its life on January 1st, 2020. Please upgrade your Python as Python 2.7 won't be maintained after that date. A future version of pip will drop support for Python 2.7. More details about Python 2 support in pip, can be found at https://pip.pypa.io/en/latest/development/release-process/#python-2-support
Package           Version
----           -----
pip            19.2.3
setuptools     41.2.0
WARNING: You are using pip version 19.2.3, however version 20.3.4 is available.
You should consider upgrading via the 'python -m pip install --upgrade pip' command.
PS C:\>
```

Como sugiere el programa actualice pip con `python2 -m pip install --upgrade pip`, y continue con la instalación de paquete `virtualenv`.

```
PS C:\> python2 -m pip install --upgrade pip
DEPRECATION: Python 2.7 will reach the end of its life on January 1st, 2020. Please upgrade your Python as Python 2.7 won't be maintained after that date. A future version of pip will drop support for Python 2.7. More details about Python 2 support in pip, can be found at https://pip.pypa.io/en/latest/development/release-process/#python-2-support
Collecting pip
  Downloading https://files.pythonhosted.org/packages/27/79/8a850fe3496446ff0d584327ae44e7500daf6764ca1a382d2d02789accf7
/pip-20.3.4-py2.py3-none-any.whl (1.5MB)
|#####| 1.5MB 297kB/s
Installing collected packages: pip
  Found existing installation: pip 19.2.3
    Uninstalling pip-19.2.3:
      Successfully uninstalled pip-19.2.3
Successfully installed pip-20.3.4
PS C:\> python2 -m pip install virtualenv
DEPRECATION: Python 2.7 reached the end of its life on January 1st, 2020. Please upgrade your Python as Python 2.7 is no longer maintained. pip 21.0 will drop support for Python 2.7 in January 2021. More details about Python 2 support in pip can be found at https://pip.pypa.io/en/latest/development/release-process/#python-2-support pip 21.0 will remove support for this functionality.
Collecting virtualenv
  Downloading virtualenv-20.15.1-py2.py3-none-any.whl (10.1 MB)
|#####| 10.1 MB 1.6 MB/s
```

Al finalizar la instalación nos indicará que el paquete no fue instalado dentro de la ruta (PATH), lo cual es intencional ya que deseamos tener ambientes separados entre Python2 y Python3

```
Installing collected packages: platformdirs, six, distlib, contextlib2, scandir, typing, pathlib2, singledispatch, zipp, importlib_resources, filelock, configparser, importlib_metadata, virtualenv
  WARNING: The script virtualenv.exe is installed in 'C:\Python27\Scripts' which is not on PATH.
  Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-warn-script-location.
Successfully installed configparser-4.0.2 contextlib2-0.6.0.post1 distlib-0.3.9 filelock-3.2.1 importlib-metadata-2.1.3 importlib_resources-3.3.1 pathlib2-2.3.7.post1 platformdirs-2.0.2 scandir-1.10.0 singledispatch-3.7.0 six-1.17.0 typing-3.10.0.0 virtualenv-20.15.1 zipp-1.2.0
PS C:\>
```

No olvide actualizar a Python 3 si se requiere.

Para lanzar un ambiente virtual, nos ubicamos en la carpeta (A) donde deseamos se active (para este caso `.Math`) y ejecutamos la instrucción `python2 -m virtualenv Math` (este paso solo hay que hacerlo una vez para crear los archivos necesarios).

```
PS C:\Users\sambo\Documents\Programacion\Python> cd .\Math\
PS C:\Users\sambo\Documents\Programacion\Python\Math> python2 -m virtualenv Math (A)
created virtual environment CPython2.7.18.final.0-64 in 565ms
  creator CPython2Windows(dest=C:\Users\sambo\Documents\Programacion\Python\Math\Math, clear=False, no_vcs_ignore=False, global=False)
  seeders FromAppData(download=False, pip=bundle, wheel=bundle, setuptools=bundle, via=copy, app_data_dir=C:\Users\sambo\AppData\Local\pypa\virtualenv)
    added seed packages: pip==20.3.4, setuptools==44.1.1, wheel==0.37.1
  activators NushellActivator,PythonActivator,FishActivator,BatchActivator,PowerShellActivator,BashActivator
PS C:\Users\sambo\Documents\Programacion\Python\Math> .\Scripts\activate.ps1 (B)
(Math) PS C:\Users\sambo\Documents\Programacion\Python\Math> dir
(C)
  Directorio: C:\Users\sambo\Documents\Programacion\Python\Math

Mode                LastWriteTime         Length Name
----                -----          ---- 
d----        12/03/2025  04:50 a. m.           Include
d----        12/03/2025  04:50 a. m.           Lib
d----        12/03/2025  04:50 a. m.           libs
d----        12/03/2025  04:51 a. m.           Math
d----        12/03/2025  04:50 a. m.           Scripts <
d----       11/03/2025  12:59 a. m.           share
-a---       11/03/2025  12:40 a. m.            42 .gitignore
-a---       11/03/2025  02:16 a. m.        47771 .plotting.py.un~
-a---       11/03/2025  05:33 a. m.        128080 .tareal.py.un~
-a---       11/03/2025  02:52 a. m.        32746 .trigonometric.py.un~
-a---       11/03/2025  12:40 a. m.          197 CACHEDIR.TAG
-a---       11/03/2025  02:16 a. m.        1057 plotting.py
```

Esto crea la carpeta `Scripts` (B) donde se ubica `activate.ps1` que activa nuestro ambiente virtual (C), dentro de la carpeta mencionada existen archivos para habilitar un ambiente virtual en otros intérpretes, por ejemplo el terminación `.bat` es para CMD y el `.sh` para Bash.

Archivo de Requerimientos para ambientes virtuales

El propósito de los ambientes virtuales es 1) permitir ejecutar varias versiones de Python en la misma computadora y 2) proveer un ambiente “homogéneo” para las dependencias de los paquetes y versiones específicas de Python que pueda requerir nuestro proyecto.

Una vez dentro de nuestro ambiente virtual, recomendamos verificar la versión de python que estamos usando para asegurar sea la correcta (A), listamos los paquetes instalados (B)

```
(MatrixVer) PS C:\Users\sambo\Programacion\Github\Python\MatrixVerificacion> python --version (A)
Python 2.7.18
(MatrixVer) PS C:\Users\sambo\Programacion\Github\Python\MatrixVerificacion> python -m pip list (B)
DEPRECATION: Python 2.7 reached the end of its life on January 1st, 2020. Please upgrade your Python as Python 2.7 is no longer maintained. pip 21.0 will drop support for Python 2.7 in January 2021. More details about Python 2 support in pip can be found at https://pip.pypa.io/en/latest/development/release-process/#python-2-support pip 21.0 will remove support for this functionality.
Package          Version
-----
pip              20.3.4
setuptools       44.1.1
wheel            0.37.1
(MatrixVer) PS C:\Users\sambo\Programacion\Github\Python\MatrixVerificacion>
```

Instalamos los paquetes que pueda requerir nuestra aplicación (para este caso requerimos numpy Matplot) :

```
(MatrixVer) PS C:\Users\sambo\Programacion\Github\Python\MatrixVerificacion> python -m pip install numpy
DEPRECATION: Python 2.7 reached the end of its life on January 1st, 2020. Please upgrade your Python as Python 2.7 is no longer maintained. pip 21.0 will drop support for Python 2.7 in January 2021. More details about Python 2 support in pip can be found at https://pip.pypa.io/en/latest/development/release-process/#python-2-support pip 21.0 will remove support for this functionality.
Collecting numpy
  Downloading numpy-1.16.6-cp27-cp27m-win32.whl (10.0 MB)
    #####| 10.0 MB 2.6 MB/s
Installing collected packages: numpy
Successfully installed numpy-1.16.6
(MatrixVer) PS C:\Users\sambo\Programacion\Github\Python\MatrixVerificacion> python -m pip install Matplotlib
DEPRECATION: Python 2.7 reached the end of its life on January 1st, 2020. Please upgrade your Python as Python 2.7 is no longer maintained. pip 21.0 will drop support for Python 2.7 in January 2021. More details about Python 2 support in pip can be found at https://pip.pypa.io/en/latest/development/release-process/#python-2-support pip 21.0 will remove support for this functionality.
Collecting Matplotlib
  Downloading matplotlib-2.2.5-cp27-cp27m-win32.whl (8.5 MB)
    #####| 8.5 MB 764 kB/s
Requirement already satisfied: numpy<=1.7.1 in c:\users\sambo\programacion\github\python\matrixverification\matrixver\lib\site-packages (from Matplotlib) (1.16.6)
Collecting six<=1.10
(MatrixVer) PS C:\Users\sambo\Programacion\Github\Python\MatrixVerificacion>
```

Verificamos que los paquetes se hayan instalado:

```
(MatrixVer) PS C:\Users\sambo\Programacion\Github\Python\MatrixVerificacion> python -m pip list
DEPRECATION: Python 2.7 reached the end of its life on January 1st, 2020. Please upgrade your Python as Python 2.7 is no longer maintained. pip 21.0 will drop support for Python 2.7 in January 2021. More details about Python 2 support in pip can be found at https://pip.pypa.io/en/latest/development/release-process/#python-2-support pip 21.0 will remove support for this functionality.
Package          Version
-----
backports.functools-lru-cache 1.6.6
cycler                  0.10.0
kiwisolver               1.1.0
matplotlib                2.2.5
numpy                    1.16.6
pip                      20.3.4
pyParsing                 2.4.7
python-dateutil           2.9.0.post0
pytz                     2025.1
setuptools                44.1.1
six                      1.17.0
wheel                    0.37.1
(MatrixVer) PS C:\Users\sambo\Programacion\Github\Python\MatrixVerificacion>
```

Creamos una “instantánea” de los paquetes instalados y la guardamos en un archivo llamado “requirements.txt”, con `python -m pip freeze > requirements.txt`, comprobamos el contenido del archivo:

```
(MatrixVer) PS C:\Users\sambo\Programacion\Github\Python\MatrixVerificacion> python -m pip freeze > requirements.txt
DEPRECATION: Python 2.7 reached the end of its life on January 1st, 2020. Please upgrade your Python as Python 2.7 is no longer maintained. pip 21.0 will drop support for Python 2.7 in January 2021. More details about Python 2 support in pip can be found at https://pip.pypa.io/en/latest/development/release-process/#python-2-support pip 21.0 will remove support for this functionality.
(MatrixVer) PS C:\Users\sambo\Programacion\Github\Python\MatrixVerificacion> cat ..\requirements.txt
backports.functools-lru-cache==1.6.6
cycler==0.10.0
kiwisolver==1.1.0
matplotlib==2.2.5
numpy==1.16.6
pyParsing==2.4.7
python-dateutil==2.9.0.post0
pytz==2025.1
six==1.17.0
(MatrixVer) PS C:\Users\sambo\Programacion\Github\Python\MatrixVerificacion>
```

Este archivo puede enviarse al control de versiones de GitHub para que sea cargado posteriormente por el usuario final (ya sea en un ambiente virtual o no), salimos del ambiente virtual y verificamos que no se instalaron los paquetes anteriores:

Anexo commandos

Copy-Item -Path .\Python\Math\ .\GitHub\PythonPracticas\ -Recurse

copiar un documento con todo su contenido

para agregar una carpeta en Github debemos asegurar que tiene por lo menos un archivo o crear la carpeta dentro de nuestro repositorio

echo "Texto " >> <path>\README.md por ejemplo

El siguiente comando instruye a GitHub que agregue todos los archivos dentro de la carpeta -*all* el parámetro -*f* indica que la acción es forzada

git add .\Math\ --all -f

si desea eliminar esta limitante, como indica más abajo deshabilite la regla en el archivo de configuración.

hint: Use -f if you really want to add them.

hint: Disable this message with "git config set advice.addIgnoredFile false"

Remover un directorio y su contenido

Remove-Item -Recurse .\Math\

Copiar un directorio con todos los archivos

Copy-Item -Path .\Math2\ .\Math\ -Recurse

```
C:\Python27\python.exe .\Programa1.py
```

Mostrar todos los archivos y directorio

Get-ChildItem -Path E:\music\Santana –Recurse

Mostrar todos los archivos incluyendo los ocultos

Get-ChildItem -Force

Mostrar todos los archivos y subdirectorios

Get-Childitem -Path E:\ -Recurse

Remover todo el contenido de un archivo incluyendo archivos ocultos (Debe tener permisos)

Remove-Item -Recurse -Force *