Linux Plus 2-for AWS and Decops

6 Aralık 2022 Salı 19:04

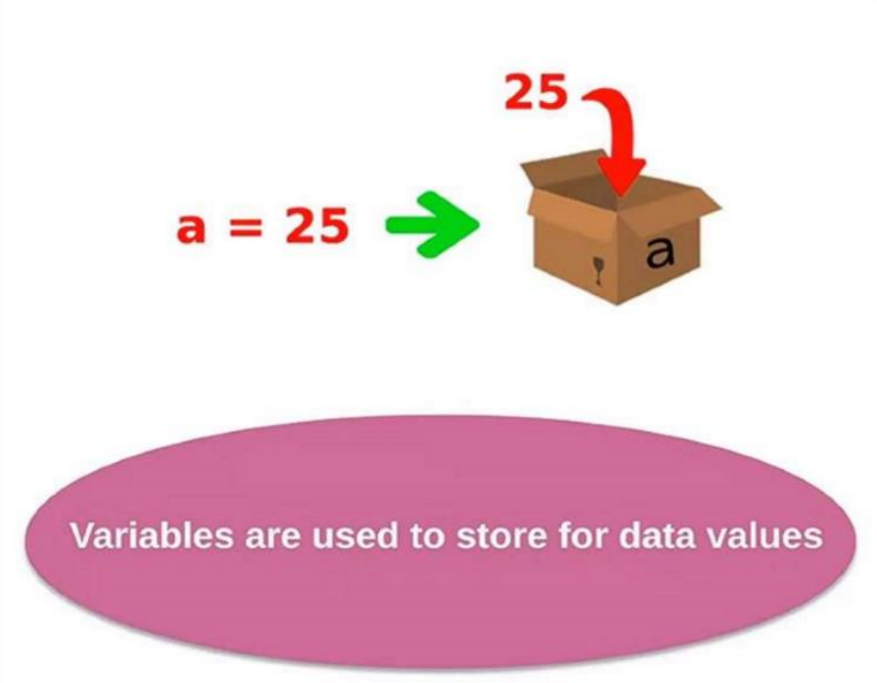
Environment Variables:

İlerleyen dönemlerde DevOps kısmında, infostructure bir araya getirirken birçok scirpt yazacağız ve tekrar eden değerleri defalarca kullanmak durumunda kalacağız. Örn, 40 tane EC2 ayağa kaldırdığını düşünün. Bunların bazıları için değerler ortak olacaktır.

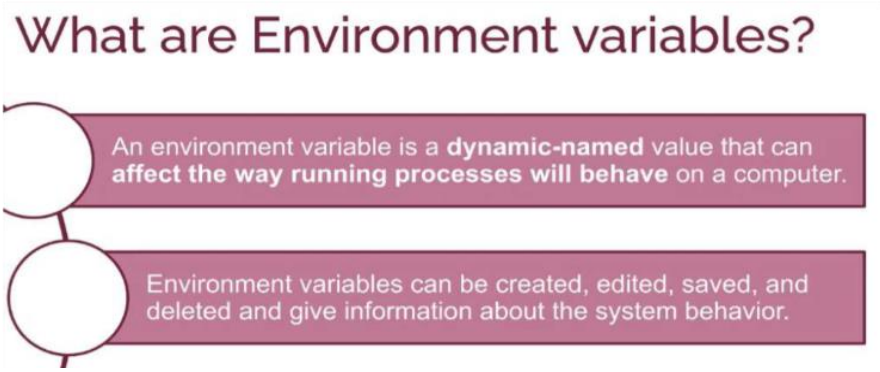
Değişen verileri ve defalarca tekrar eden verileri bir değişkene bağlıyoruz ve veri değişeceği zaman bu değişkeni değiştirdiğimizde metin içine değişikliği entegre etmiş oluyoruz. Örn, 15 tane EC2 için bir pem.key kullandık ve bir süre sonra güvenlik için bu pem.key i değiştirmemiz gerekti. Bu durumda pem.key i her EC2 için ayrı ayrı değiştirmiyoruz, bunu daha önce bağladığımız değişkeni değiştiriyoruz. (variable)

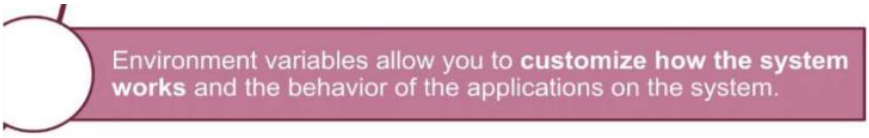
açtığımız 60 tane EC2 nun 15 tanesi development, 15 tanesi staging, 15 tanesi production ve diğer 15 tanesi test için olacak. Development ortamı için pem.key şu olsun, produciton için makinanın tipi şu olsun gibi veri değişikliklerini variables a atayarak otomasyon yaparken işimizi çok kolaylaştıracak.

Bu sayede örn gecenin bir vakti bir script koşulması gerektiğinde biz bunun otomasyonunu kurarak uyurken scriptin koşulmasını sağlayacağız.



Variables, sayı, düz metin, list, dictionary gibi veri tiplerini tutan bir depodur. Bu değerlerin değiştirlmesi, update edilmesi çok hızlı bir şekilde gerçekleşebilir.lı bir şekilde gerçekleştiği için işl





Enviroment Variable lara nasıl erişiyoruz?

"env" veya "printenv" komutuyla erişiyoruz:

Environment variable ları BÜYÜK HARFLE yazarız. PEP 8 gibi bir kuraldır bu.

Common Environment Variables

Biz hiçbir şey yapmasak da sistemin bizim istediğimiz gibi çalışması, cunfigürasyon ayarlarının yapılabilmesi için önceden tanımlanmış EV (Environment Variable) lerdir.

Variable	Description
PATH	This variable contains a **colon (:)-separated list of directories** in which your system looks for executable files.
USER	The username
HOME	**Default path** to the **user's home** directory
EDITOR	Path to the program which edits the content of files
UID	User's unique ID
TERM	Default terminal emulator
SHELL	Shell being used by the user
LANG	The current locales settings.

tek bir değişkene erişmek için echo $DEĞİŞKENİSMİ

Oluşturduğumuz bir komutun, dosya ve klasör adı girmeden çalışabilmesi, komut olarak çalışabilmesi için onun bulunduğu klasör adresinin PATH değişkeninde bulunması gerekir.

$PATH variable aralarında iki nokta bulunduran ve bu iki nokta aralarında komut dosyalarını klasör adreslerini barındıran bir variable dır.

directorlylerde komutları saklı tutar. yani ls komutu yazdığımızda ls komutu $PATH değişkeni içindeki directorylerinden birinde yer alıyordur.
ve girdiğimiz komutu soldan sağa doğru sırasıyla directoryde arar.

```
ubuntu@ip-172-31-95-146:~$ echo $PATH
/home/ubuntu/.vscode-server/bin/6261075646f055b99068d3688932416f2346dd3b/bin/remote-cli:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:
/usr/local/games:/snap/bin
```

PATH variable ı export edip (içini boşalttığımız zaman) ls kodunu çağırıp çalıştırmadı mesela:

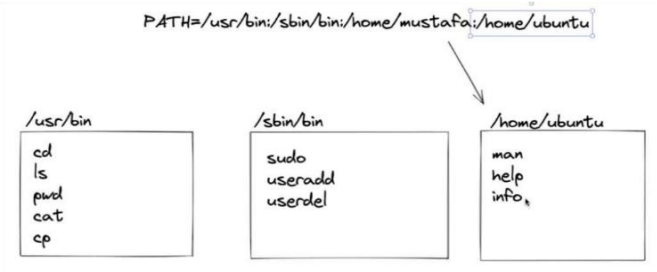
```
ubuntu@ip-172-31-95-146:~$ export PATH=""
ubuntu@ip-172-31-95-146:~$ echo $PATH

ubuntu@ip-172-31-95-146:~$ ls
bash: ls: No such file or directory
ubuntu@ip-172-31-95-146:~$
```

$PATH içerisine bir directory (örn, /home/ubuntu) eklemek istersek aşağıdaki komutla ekleyebiliriz:

export PATH="${PATH}:/home/ubuntu"

```
ubuntu@ip-172-31-95-146:~$ export PATH="${PATH}:/home/ubuntu"
ubuntu@ip-172-31-95-146:~$ echo $PATH
/home/ubuntu/.vscode-server/bin/6261075646f055b99068d3688932416f2346dd3b/
/usr/local/games:/snap/bin:/home/ubuntu
ubuntu@ip-172-31-95-146:~$
```



bir klasör altındaki komutların çalışabilmesi için $PATH içinde o klasörün listelenmesi gerekir. Aşağıdaki durumda /sbin/bin altındaki komutlar çalışmayacaktır:

yanyana dizilen bu directoryler iç içe değildir, sadece Linux bunlara soldan sağa doğru bir sırayla bakar.

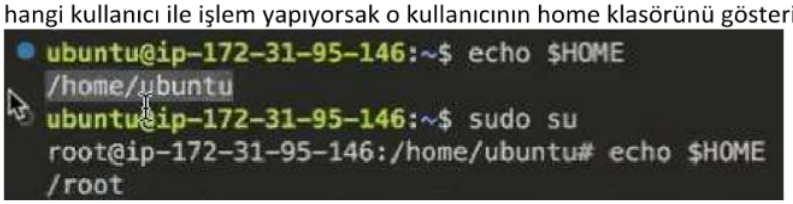
Örnek; şirkette birisi yeni bir dosya oluşturdu. Biz de DevOps çu olarak bu dosyayı $PATH değişkeni altına ekliyoruz ki o komut çalışsın.

USER variable:
hangi kullanıcı ile işlem yapıyorsak onu gösterir.

```
ubuntu@ip-172-31-95-146:~$ echo $USER
ubuntu
ubuntu@ip-172-31-95-146:~$ sudo su
root@ip-172-31-95-146:/home/ubuntu# echo $USER
root
```

HOME variable:

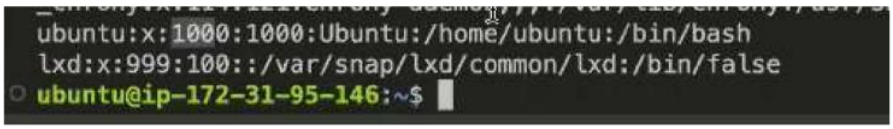
hangi kullanıcı ile işlem yapıyorsak o kullanıcının home klasörünü gösterir.



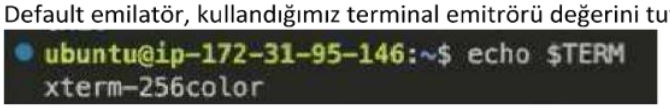
Bunlar otomasyonla ilgili scriptleri koşturacakken kullanacağız. Mesela /home/mustafa diye yazmayacağız da ${HOME}/şeklinde yazacağız ki, sonradan her distroda çalışabilsin.

EDITOR variable:
default bir editör atamamızı sağlayan variable dır. Örnekte default editörü vim olarak ayarladık:

UID variable:
Her kullanıcı Linux ta benzersiz bir id ye sahiptir. /etc/passwd de bütün sistem ve gerçek kullanıcılar tutulur.
Root default olarak 0 değerini alır. Gerçek kullanıcılar aksi ayarlanmadıkça 1000 değerinden başlar.

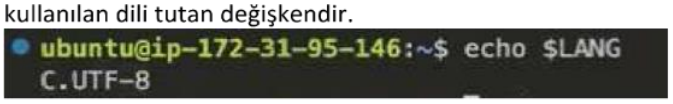


TERM variable:
Default emilatör, kullandığımız terminal emitrörü değerini tutuyor. çok kullandığımız bir şey değil.



SHELL variable:
kullanıcının default olarak kullandığı shell i tutar:

LANG variable:
kullanılan dili tutan değişkendir.

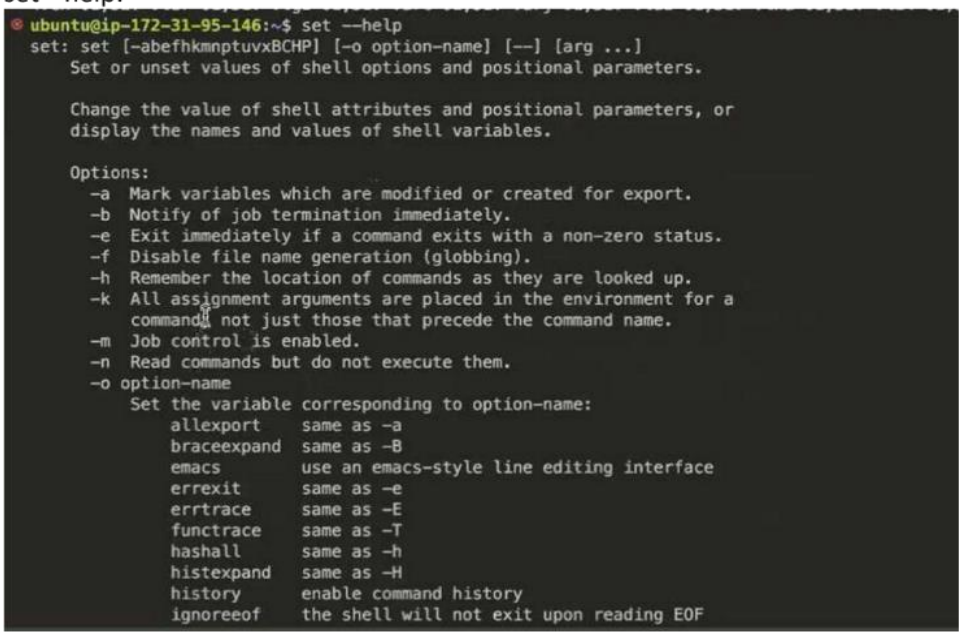


Variable lara ulaşma komutları:

Command	Description
env	The **env** command is a shell command used to display and manipulate environment variables. It is **used to list down** environment variables.
printenv	The command prints all or the specified environment variables.
set	The command sets or unsets shell variables. When used without an argument it will print a list of all variables including environment and shell variables, and shell functions.
unset	The command deletes shell and environment variables.
export	The command sets environment variables.

printenv ile $ kullanmadan tek bir variable çağırabiliriz.
env ile sistemde kayıtlı düm değişkenleri çağırırız.

set ve unset ile environment ve shell variable ları oluşturmak için kullanılır.
set --help:

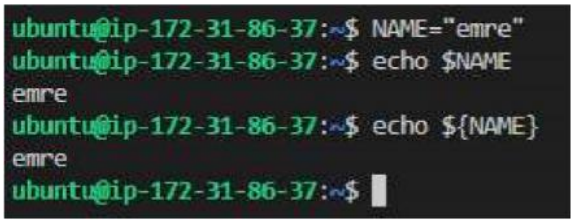


unset komutundan sonra variable ismini girerek o variable ı silebiliriz.

export komutu ile de shell değil environment variable oluştururuz.

Command	Description
echo $VARIABLE	To display value of a variable
env	Displays all environment variables
VARIABLE_NAME=variable_value	Create a new shell variable
unset	Remove a variable
export Variable=value	To set value of an environment variable

NAME isminde bir shell variable oluşturarak echo komutu ile çağırıyoruz:



= den sonra boşluk olmayacak.
env ile çağırırken shell variable ları getirmez.

export ile environment variable atayarak env ile çağıralım:

```
ubuntu@ip-172-31-86-37:~$ export NAME2="emre"
ubuntu@ip-172-31-86-37:~$ env
SHELL=/bin/bash
NAME2=emre
```

unset komutu ile de variable ı kaldırabiliriz:

```
ubuntu@ip-172-31-86-37:~$ unset NAME2
ubuntu@ip-172-31-86-37:~$ echo $NAME2

ubuntu@ip-172-31-86-37:~$ env
SHELL=/bin/bash
PWD=/home/ubuntu
```

shell variables ve enivronment variables farklıdır:
script içinde de shell variables a erişemeyiz.

echo ile ikisini de yazdırırız.

printenv ve echo ile variable çağırma:

```
clarusway@DESKTOP-UN6T2ES:~$ printenv USER
clarusway
clarusway@DESKTOP-UN6T2ES:~$ printenv HOME
/home/clarusway
clarusway@DESKTOP-UN6T2ES:~$ printenv UID
clarusway@DESKTOP-UN6T2ES:~$ echo $TERM
xterm-256color
clarusway@DESKTOP-UN6T2ES:~$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
```

Exercise:

Create a variable named **MYVAR** with the value of **"my value"**
Print value of the **MYVAR** variable to the screen
Assign **"new value"** to the **MYVAR** variable
Print value of the **MYVAR** variable to the screen
Delete **MYVAR** variable
Print value of the **MYVAR** variable to the screen

```
ubuntu@ip-172-31-86-37:~$ MYVAR=my_value
ubuntu@ip-172-31-86-37:~$ echo $MYVAR
my_value
ubuntu@ip-172-31-86-37:~$ MYVAR="new value"
ubuntu@ip-172-31-86-37:~$ echo $MYVAR
new value
ubuntu@ip-172-31-86-37:~$ unset MYVAR
ubuntu@ip-172-31-86-37:~$ echo $MYVAR

ubuntu@ip-172-31-86-37:~$
```

export ile yapınca printenv ile çağırabiliriz:

```
ubuntu@ip-172-31-86-37:~$ export MYVAR=my_value
ubuntu@ip-172-31-86-37:~$ printenv MYVAR
my_value
ubuntu@ip-172-31-86-37:~$ export MYVAR="new value"
ubuntu@ip-172-31-86-37:~$ printenv MYVAR
new value
ubuntu@ip-172-31-86-37:~$ unset MYVAR
ubuntu@ip-172-31-86-37:~$ printenv MYVAR
ubuntu@ip-172-31-86-37:~$
```

The PATH Variable

When we want the system to execute a command, we almost **never need to give the full path** to that command.

For instance, we know that the ls command is in the /bin directory (you can check with), yet we don't need to enter the /bin/ls command for the computer to list the content of the current directory.

This is maintained by the PATH environment variable. This variable **lists all directories** in the system **where executable files** can be **found**.

$PATH değişkenine eklenememiş olsaydı bir komutu çalıştırmak için her komut için tam adresini girmemiz gerekirdi.

Bir komut bulunamadı ise ilk bakacağımız yer $PATH değişkeni olacak.

```
clarusway@DESKTOP-UN6T2ES:~$ printenv PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
```

In this example, the /usr/local/sbin, /usr/local/bin, /usr/sbin, /usr/bin, /sbin and /bin directories are subsequently searched for the required program. The search will be stopped as soon as a match is found, even if not all directories in the path have been

PATH variable a klasör eklemek için export komutunu şu şekilde kullanıyoruz:

```
clarusway@DESKTOP-UN6T2ES:~$ export PATH=$PATH:/games/awesome
clarusway@DESKTOP-UN6T2ES:~$
```

Let's say you want to run that file called fun. You learned from running the find command that it's in a directory called /games/awesome. However, /games/awesome is not in your path, and you don't want to type the full path just to run the game. So you can add it to PATH variable with export command.

Quoting

Quoting is used to disable special treatment of certain characters and words, as well as to prevent parameter expansion and preserve what is quoted.

The bash shell knows rare, important characters. For example, $var is used to extend the value of the element.

```
echo "$PATH"
echo "$PS1"
```

Bazen değişkenleri ' ' veya " " ile çağırırız. Variable ı tek başına kullanmayacağımız zamanlarda; home klasörünün 5-6 klasör altındaki bir dosya yolunu çağırmak için, "" ve {} kullanırız:

```
ubuntu@ip-172-31-95-146:~$ VAR1=/home/ubuntu/mustafa/clarusway/dosya1^C
ubuntu@ip-172-31-95-146:~$ VAR1="${HOME}/mustafa/clarusway/dosya1"
```

{} gibi özel karakterleri özel anlamlarıyla kullanabilmek için string içinde " " yazmamız gerekir.

Ancak özel anlamlar içerek karakterleri özel anlamlarında arındırmak istiyorsak ' ' tek tırnak kullanıyoruz. Ancak " " kullanmak istiyoruz fakat bazı özel karakterlerin de anlamını yitirmesini istiyorsak \ ters slash kullanırız.

Double Quotes	• The double quote ("quote") protects everything enclosed between two double quote marks except $, `, " and \.	echo "$SHELL" echo "/etc/*.conf"
Single Quotes	• The single quote ('quote') protects everything enclosed between two single quote marks.	echo '$SHELL' echo '/etc/*.conf'
Backslash	• Use the backslash to change the special meaning of the characters or to escape special characters within the text such as quotation marks.	echo "Path is \$PATH"

```
ubuntu@ip-172-31-95-146:~$ echo "$SHELL"
/bin/bash
ubuntu@ip-172-31-95-146:~$ echo '$SHELL'
$SHELL
ubuntu@ip-172-31-95-146:~$ echo "\$SHELL is $SHELL"
$SHELL is /bin/bash
ubuntu@ip-172-31-95-146:~$
```

yukarıda çift tırnak içinde $SHELL kullandık. allta tek tırnak içinde $ işaretinin özel gücünü aldık. En altta ise \ işaretiyle ilk $ özel gücünü alırken, çift tırnak kullanarak ikinci $ işaretinin özel gücünü kullandık.

sudo command:
super user do

vekaleten okul müdürlüğü gibi düşünülebilir.
root user ın yapabileceği bazı ayrıcalıkları komutları kullanma imkanı verir.

Commands	Meaning
sudo -l	List available commands.

sudo command	Run command as root.
sudo -u root command	Run command as root.
sudo -u user command	Run command as user.
sudo su	Switch to the superuser account.
sudo su -	Switch to the superuser account with root's environment.
sudo su - username	Switch to the username's account with the username's environment.
sudo -s	Start a shell as root
sudo -u root -s	Same as above.
sudo -u user -s	Start a shell as user.