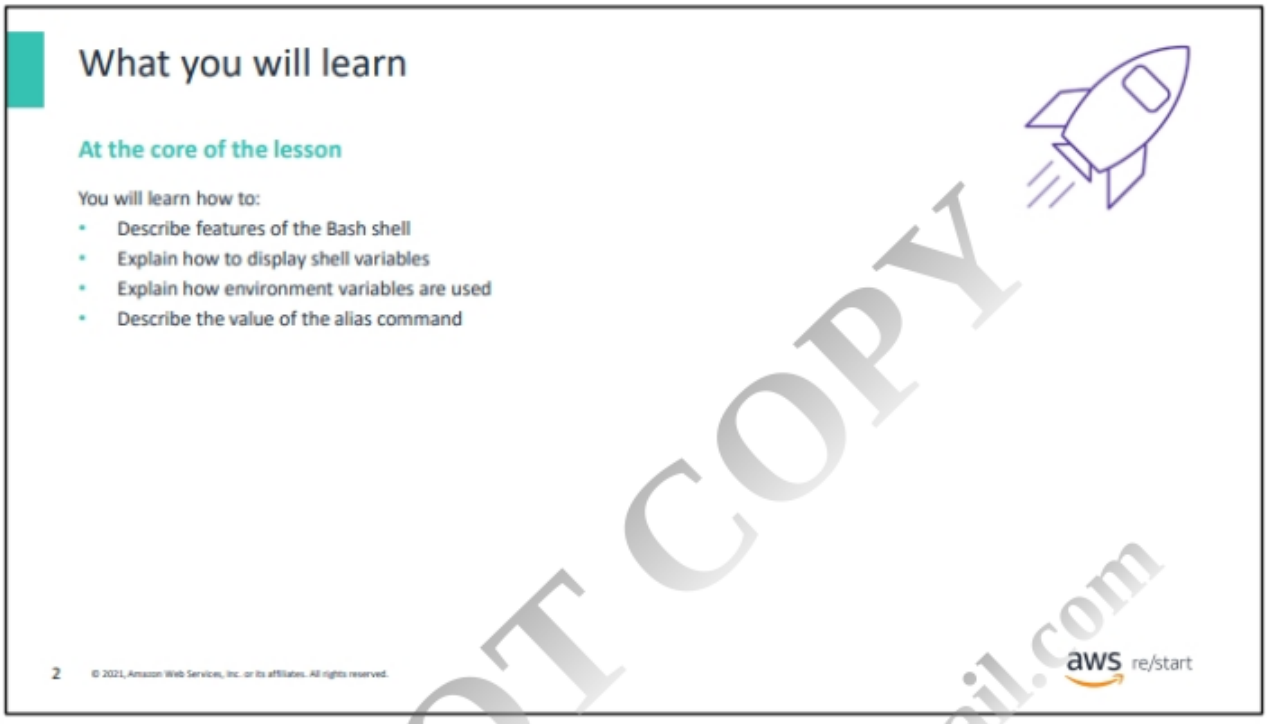
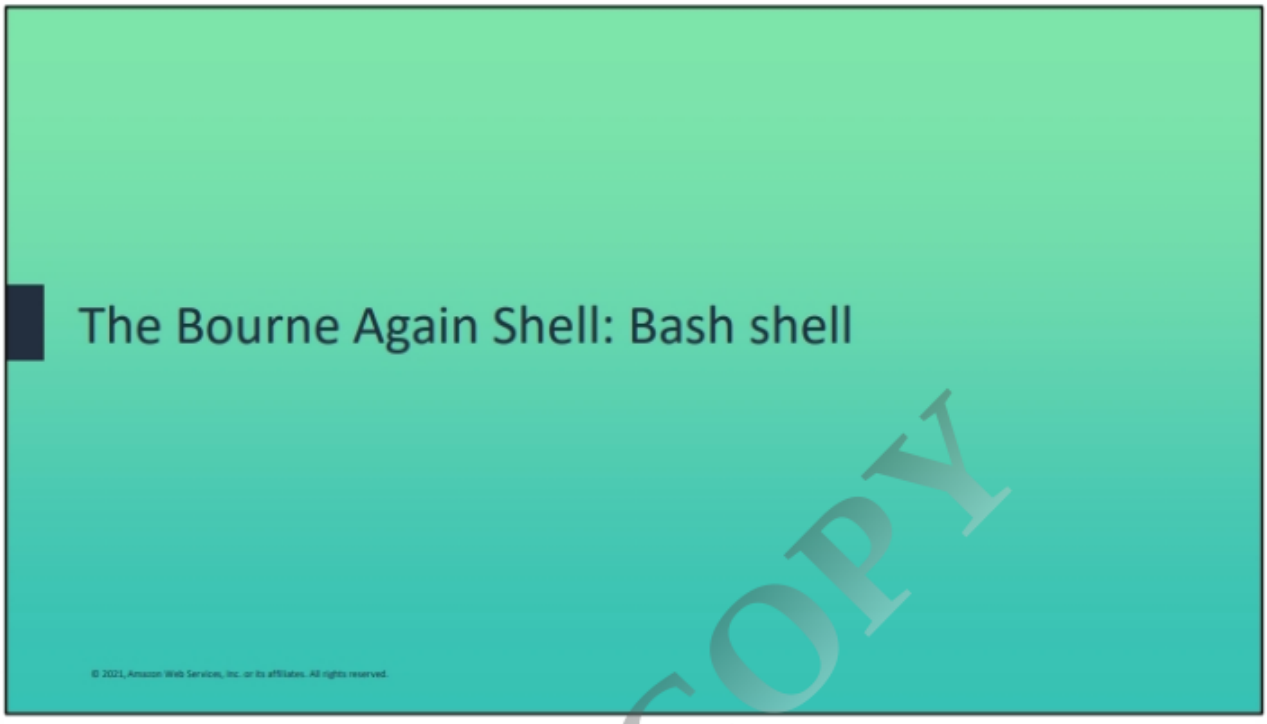
Welcome to the Bash Shell.



You will learn how to:

- Describe features of the Bash shell
- Explain how to display shell variables
- Explain how environment variables are used
- Describe the value of the alias command



This section defines a Linux shell and introduces the Bash shell.

The Linux shell

What is a shell?

- A shell accepts and interprets commands.
- A shell is an environment in which commands, programs, and shell scripts are run.
- There are many types of Linux shells available. Bash is one of them, and this section discusses it further.

aws re/start

What is a shell?

The primary purpose of a shell is to allow the user to interact with the computer operating system. It has two different functions. One is a program and the other is a command interpreter. As a program shell, it provides the interface for utilities and programs. As a command interpreter, a shell accepts and interprets the commands you enter into the **command line interface (CLI)** or **terminal.**

The Bourne Again Shell: Bash

What is the Bourne Again Shell (Bash)?

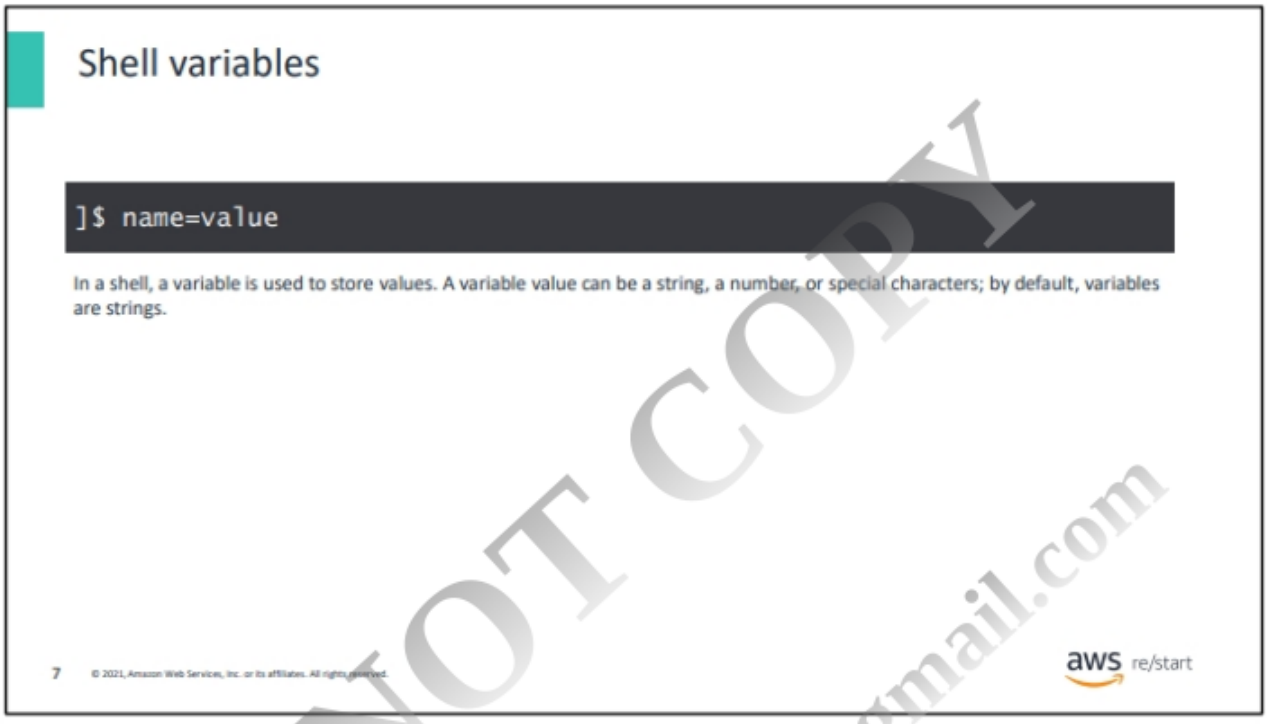
- Bash is the default shell in Linux.
- It offers an efficient environment for interacting with the operating system and scripting.

aws re/start

What is Bash?

Bash is a programming language for running commands. Bash is the default shell in Linux operating systems. It is widely used, so some familiarity with Bash is expected in many systems or development roles.

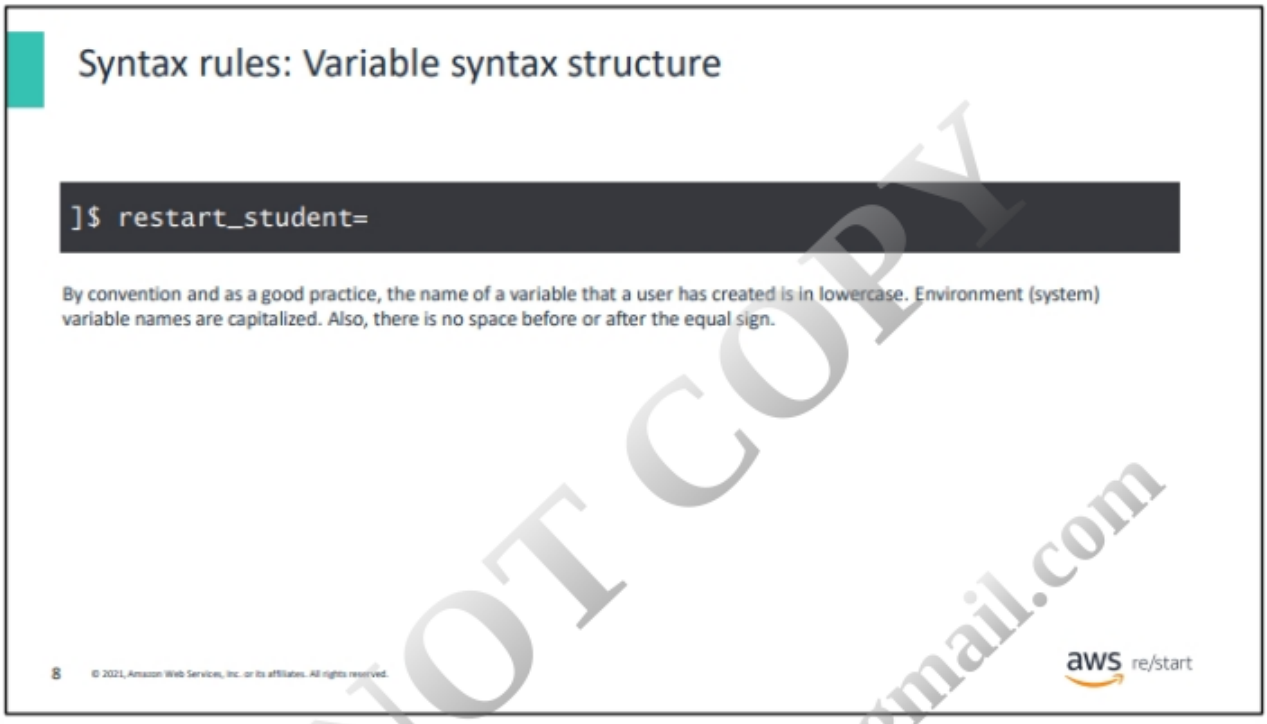
This section explains what Bash shell variables are, how to name them, what the rules are for writing them, and how to assign a value to them.



In a shell, a variable is used to store values. Variables can be a name, a number, or special characters; by default, variables are strings. Variables are character strings to which you can assign a value; a value can be assigned as a number, text, file name, device, or other data type. Variables are a symbolic label for a portion of memory used to assign values and read and manipulate contents.

Scripts or other commands can call shell variables. The values that these shell variables represent are then substituted into the script or command.

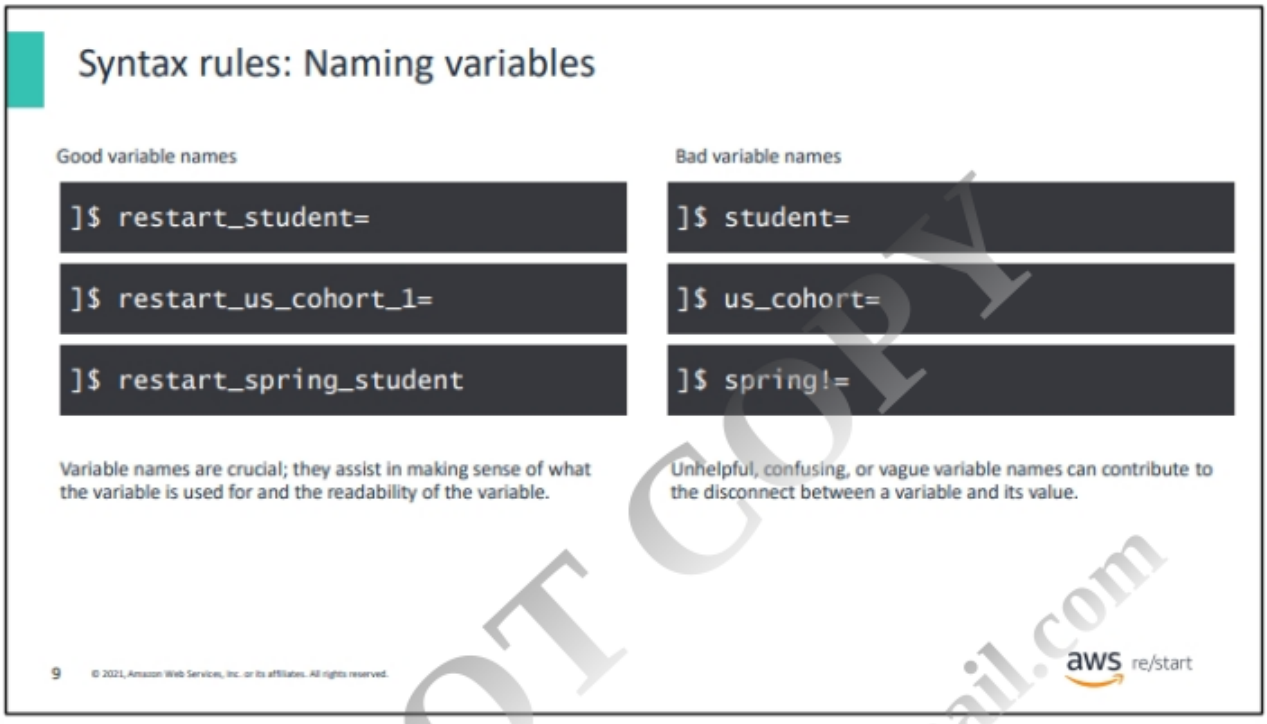
The next slide provides a few shell syntax rules for creating variables; you will look at these rules.



There are rules for defining or creating variables in the shell.

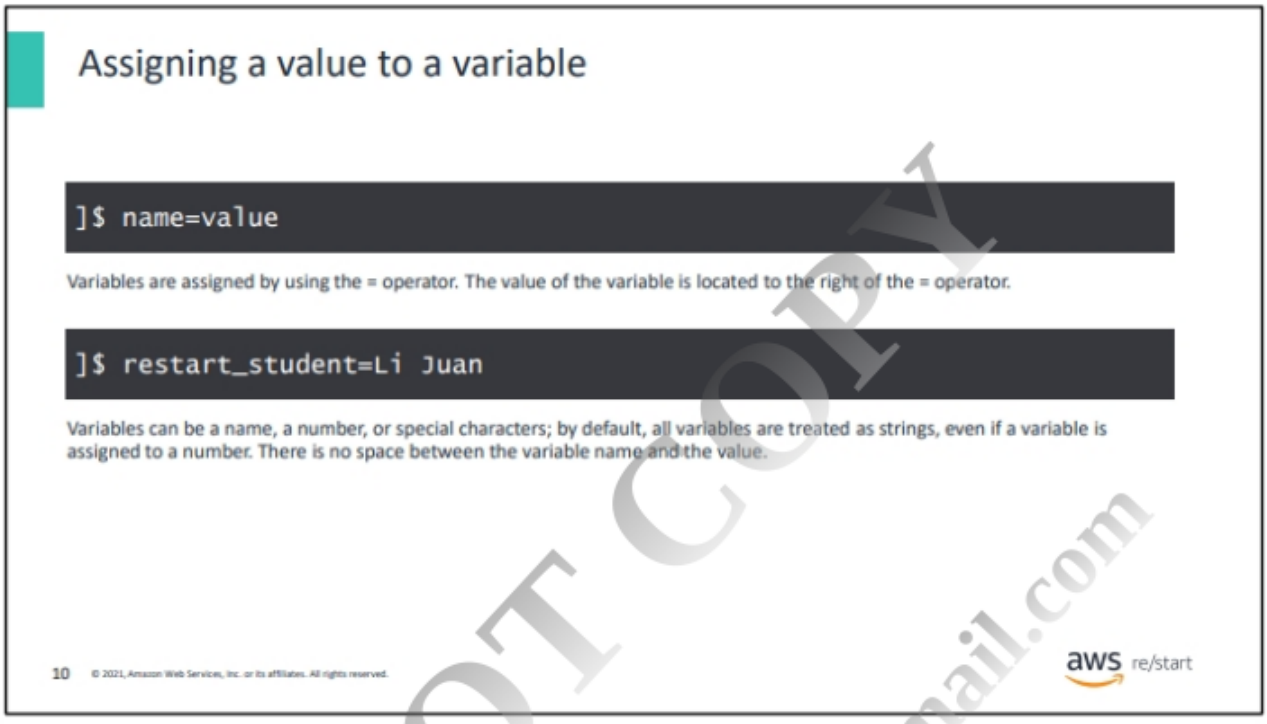
When defining a variable, the variable name must be prefixed with the dollar ($) symbol.

The variable must contain no spaces or special characters within the variable name. A variable name can contain only letters (a to z or A to Z), numbers (0 to 9), or the underscore character (_), and they are usually capitalized (e.g. VARIABLE).



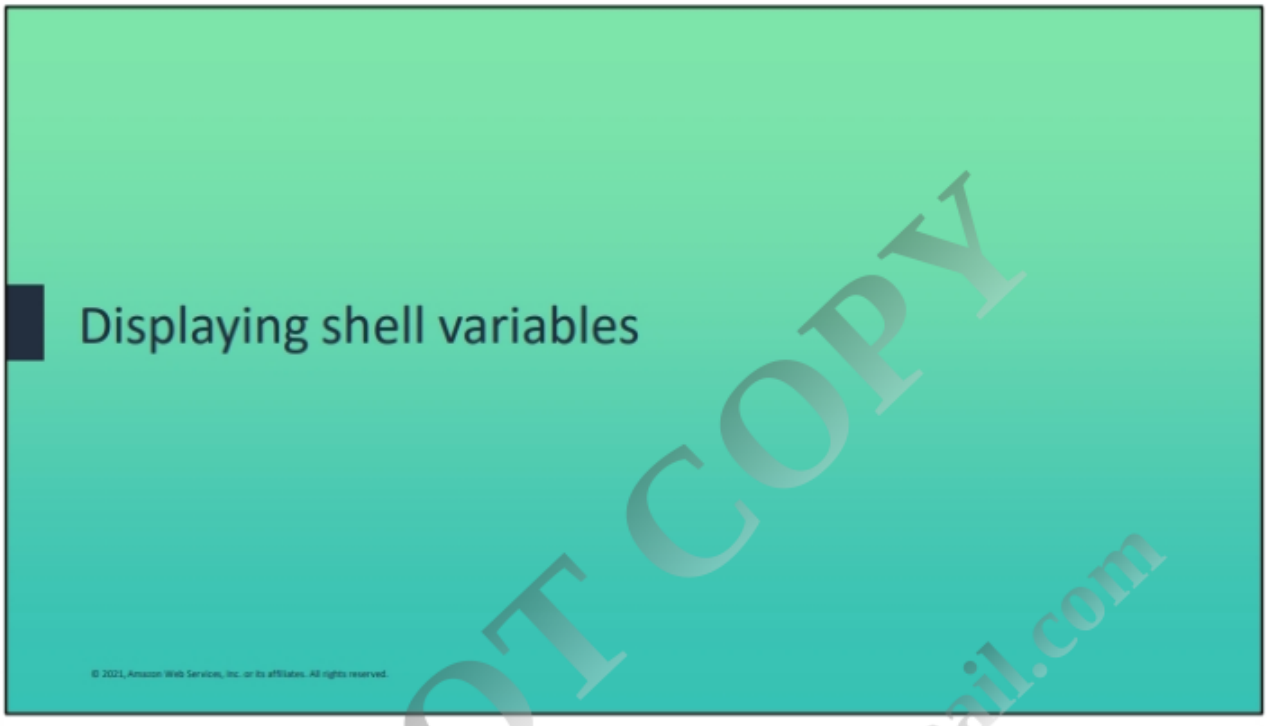
Naming variables in Bash can be difficult, but if you name variables properly, it is useful. Good variable names are crucial; they assist in making sense of what the variable is used for and the readability of the variable. It is good practice to be consistent in your naming pattern. Unhelpful, confusing, or vague variable names can contribute to the disconnect between a variable and its value.

The variable names on this slide are examples of a few good and a few bad variable names.



A value can be assigned as a number, text, file name, device, or other data type.

Again, as discussed earlier, a variable is used to store values. Variables can be a name, a number, or special characters; by default, all variables are treated as strings, even if a variable is assigned to a number. Variables are assigned by using the = operator. There is no space between the variable name and the value.



In this section, you learn how to display variables using the echo command.

Displaying shell variables

Display Bash variables by using the echo command:

```
]$ echo $VARIABLE_NAME
```

or

```
]$ echo $(VARIABLE_NAME)
```

```
[root@server00 ~]# echo $USER
root
[root@server00 ~]# echo $HOME
/root
[root@server00 ~]# echo $SHELL
/bin/bash
```

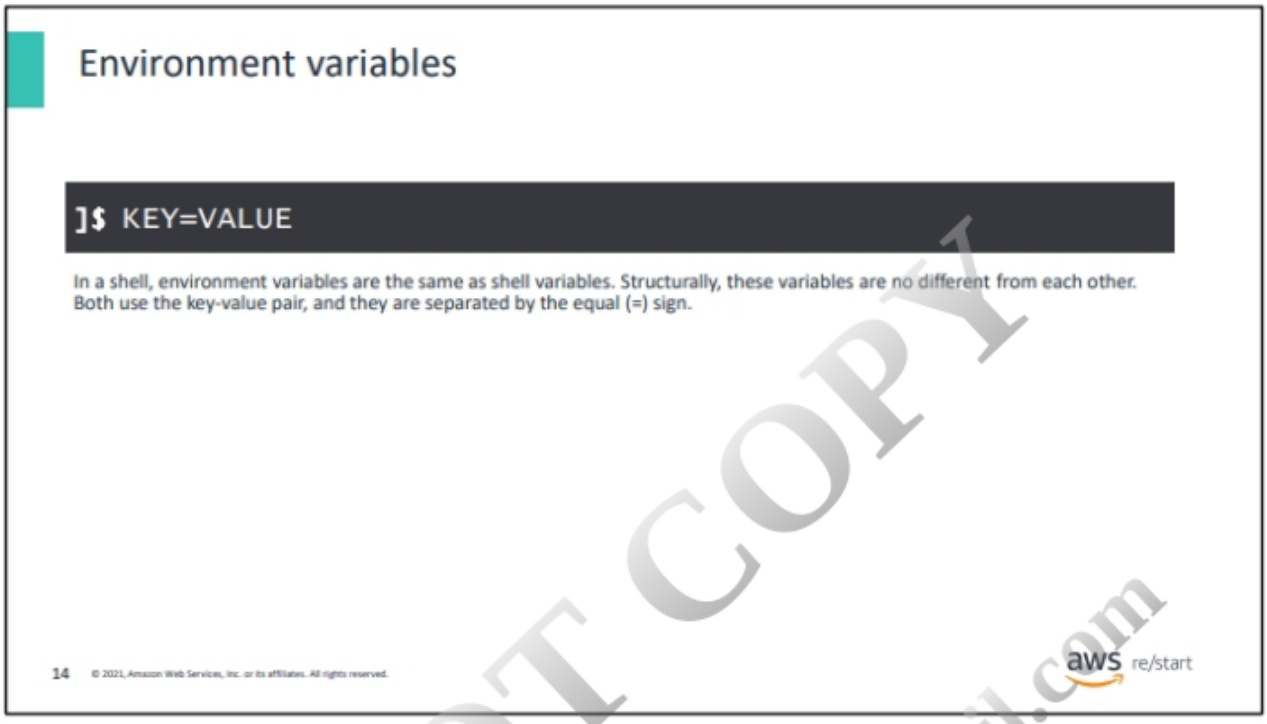
You can also use the echo command to view the output from environment variables.

12

aws re/start

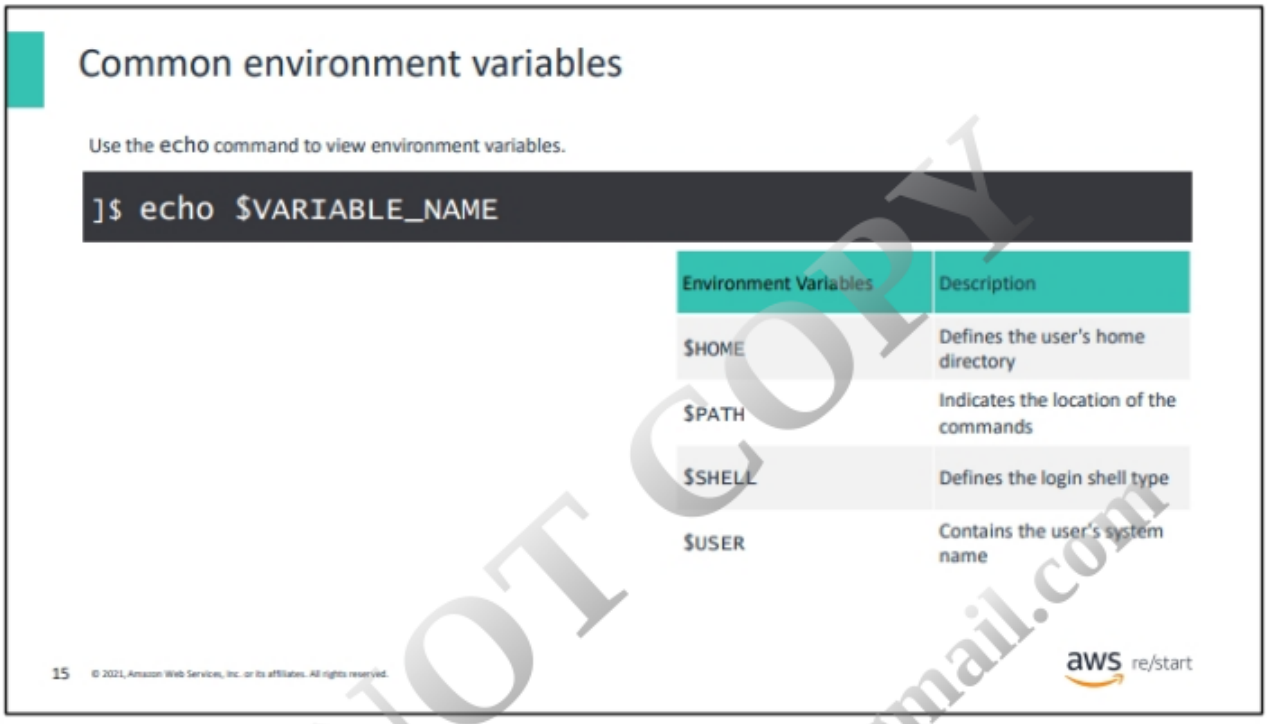
To display the value of a variable, use the echo $VARIABLE_NAME. Also use the echo command to view the output from environment variables or system-wide variables.

In this section, you review environment variables.

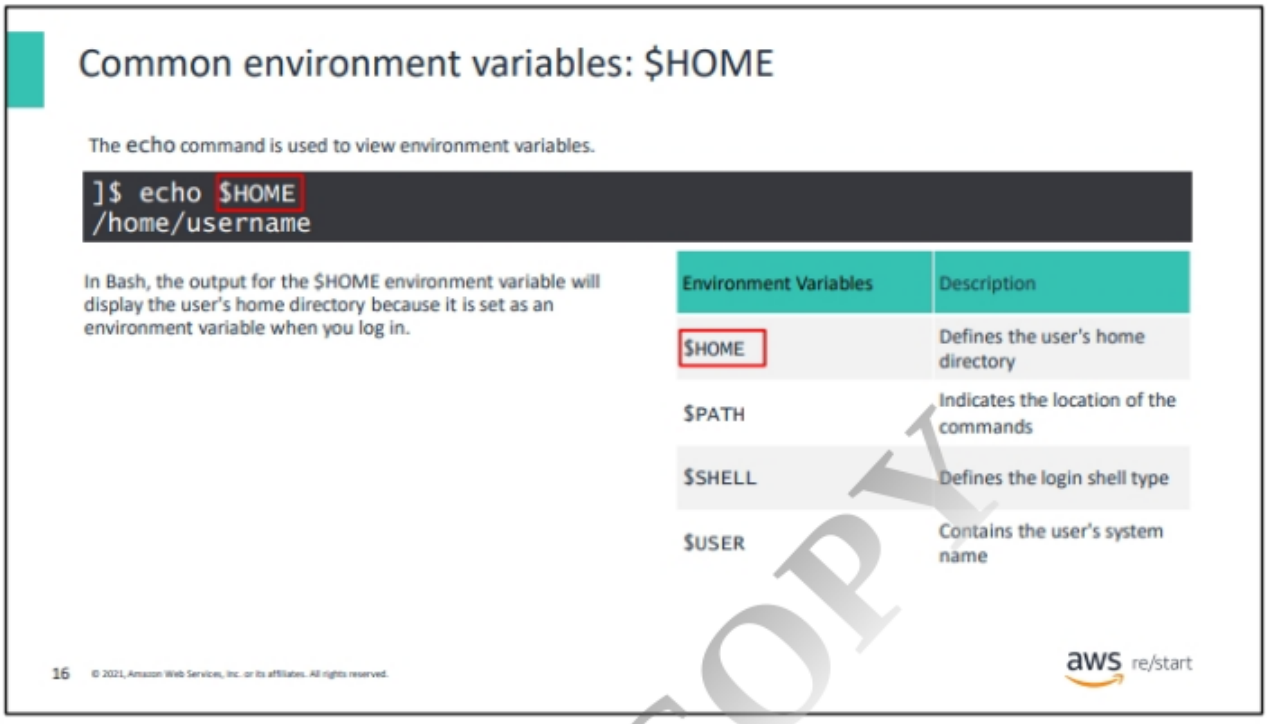


Environment variables are structurally the same as shell variables; they are no different from each other. Both use the key-value pair, and they are separated by the equal (=) sign.

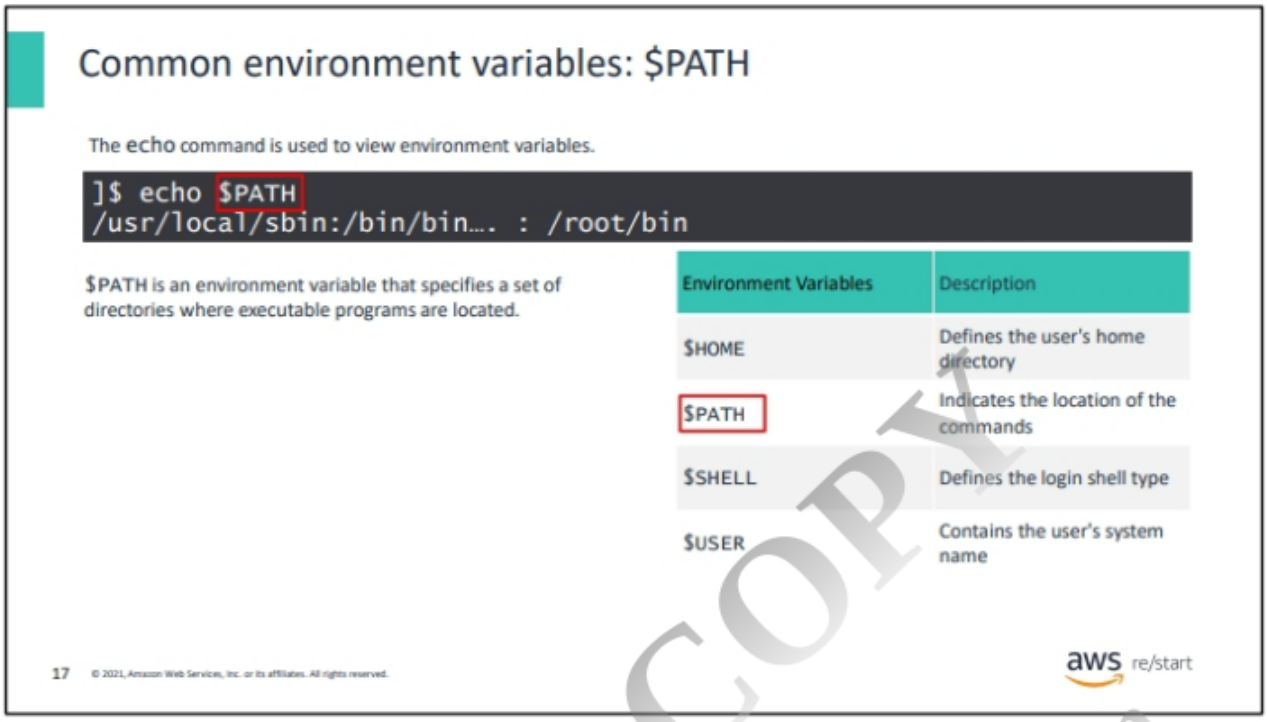
Environment variables are system wide, and all child processes and shells inherit them. With environment variables, you can pass information about the current operating environment to a program running. Finally, applications and daemons reference environment variables as needed.



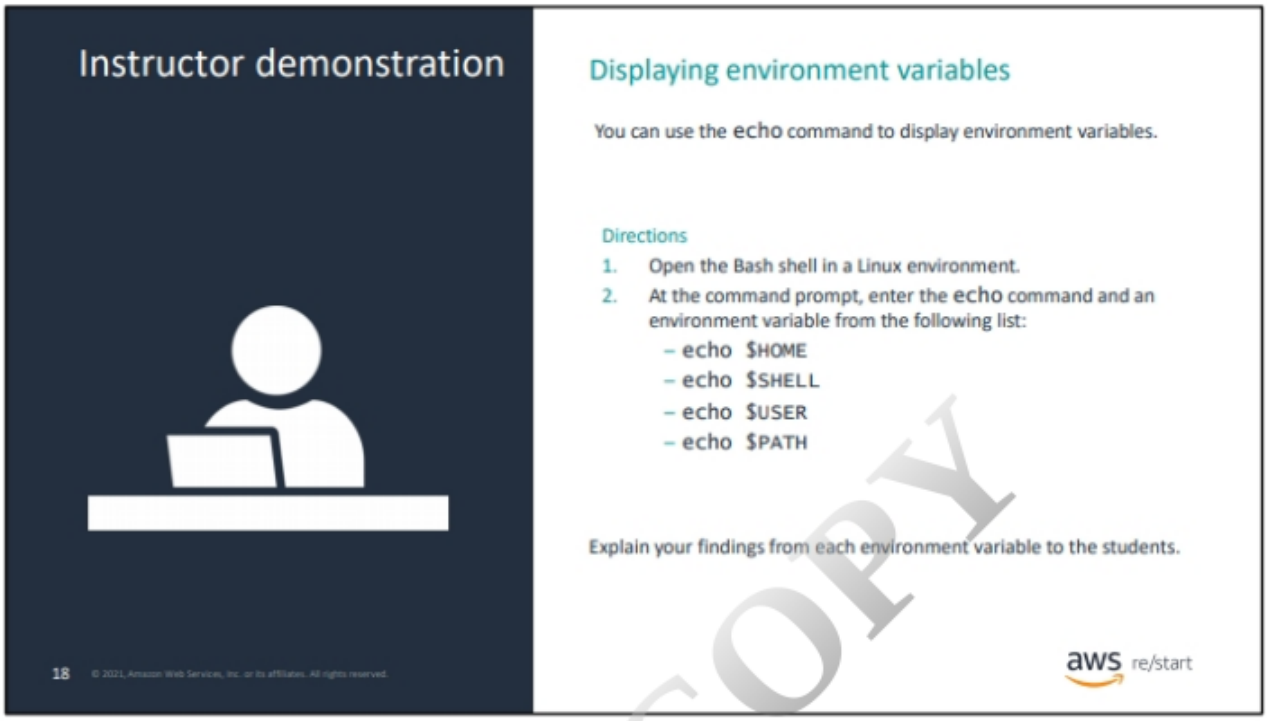
The table on this slide displays a list of a few common environment variables you may encounter when using Linux.



In Bash, for example, the output for the $HOME environment variable will display the user's home directory.



$PATH is an environment variable that specifies a set of directories where executable programs are located. In general, each running process or user session has its own PATH setting.



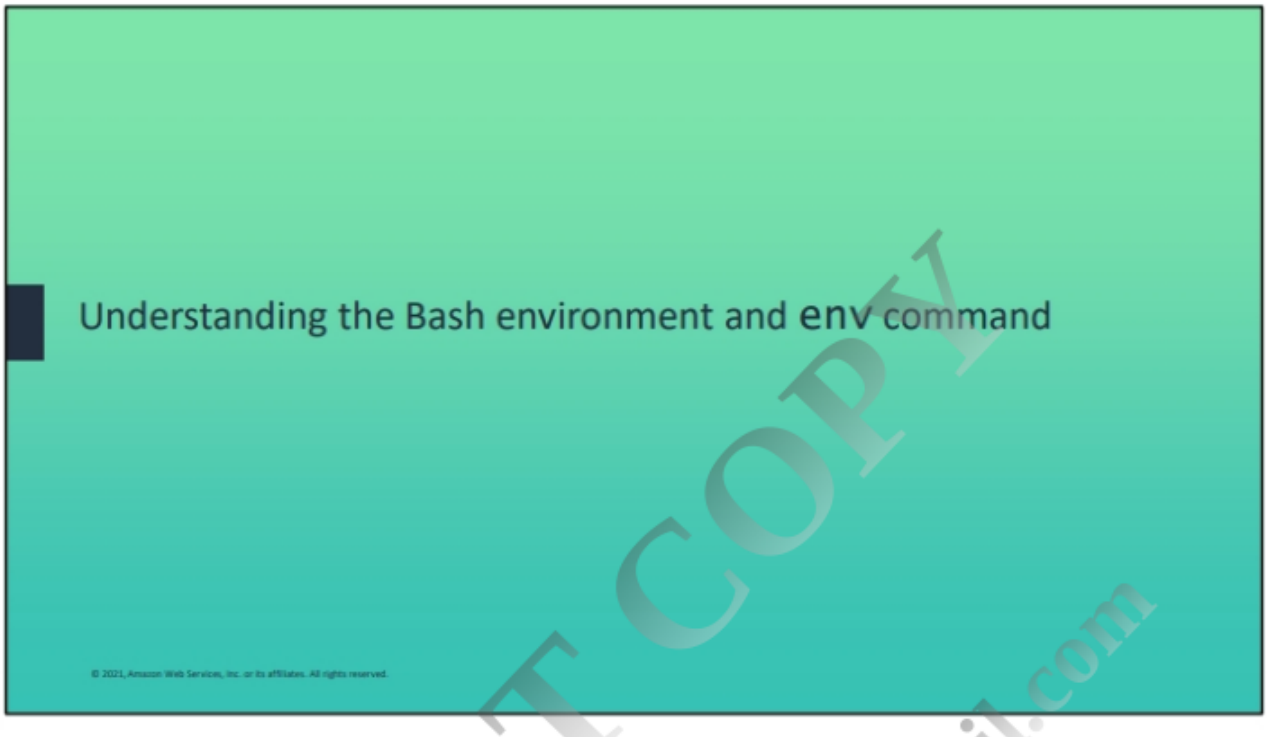
For this demonstration:

1. Open the Bash shell in a Linux environment.

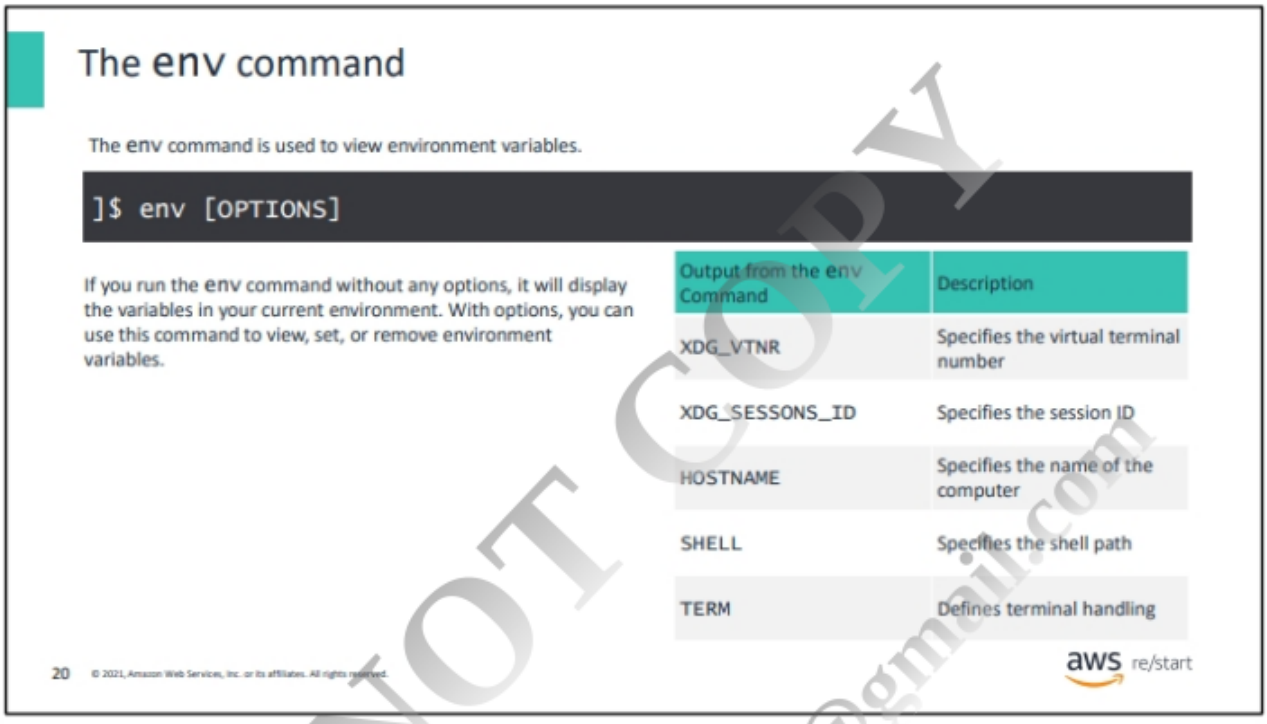
2. At the command prompt, enter the echo command and an environment variable from the following list:

- echo $HOME
- echo $SHELL
- echo $USER
- echo $PATH

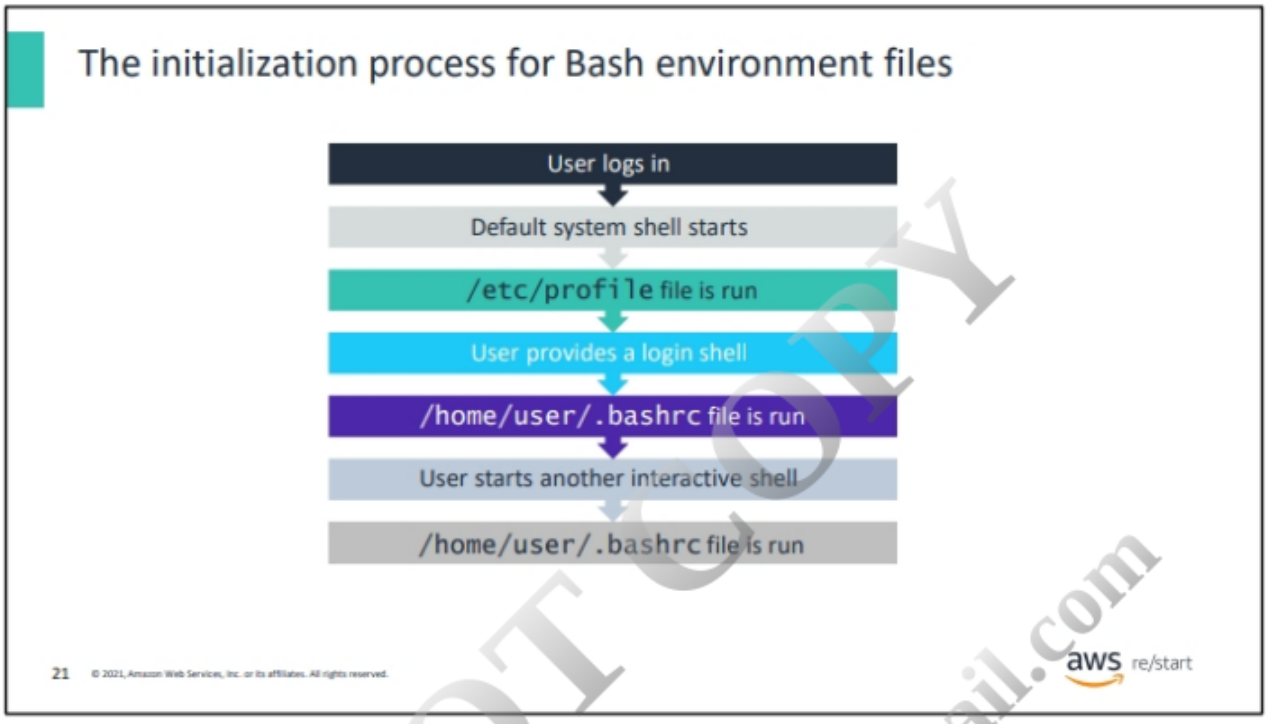
Explain your findings from each environment variable.



This section discusses the Bash environment and the env command.



The env command is a shell command for Linux. You use this command to print a list of environment variables or run another utility in an altered environment.



During the initialization process of Bash environment files, two different initialization shell files are invoked.

The /etc/profile file contains system-wide environment configurations and startup scripts for login setup. The command line prompt is set within this file. All configurations that you want to apply to all system users' environments should be added to this file.

When you log in, Bash reads the /etc/profile instructions. The /etc/profile file usually sets the shell variables PATH, USER, HOSTNAME, etc.

The /etc/bashrc file contains system-wide functions and aliases, including other configurations that apply to all system users.

Instructor demonstration

Displaying the Bash environment files

You will view the files that are used to set a user's configuration during system startup.

Directions

Steps to view the `.bashrc`

1. Open the Bash shell in a Linux environment.
2. At the command prompt, enter the `ls -a` command to view hidden files.
3. Enter the `cat .bashrc` command to display the content of the file.

Steps to view the `/etc/profile`

1. At the command prompt, enter `cd /etc/profile`
2. Enter `ls` to see the `profile` file.
3. Enter the `cat .profile` command to view the content of the file.

Explain the content of the Bash environment files to the students.

aws re/start

22

For this demonstration:

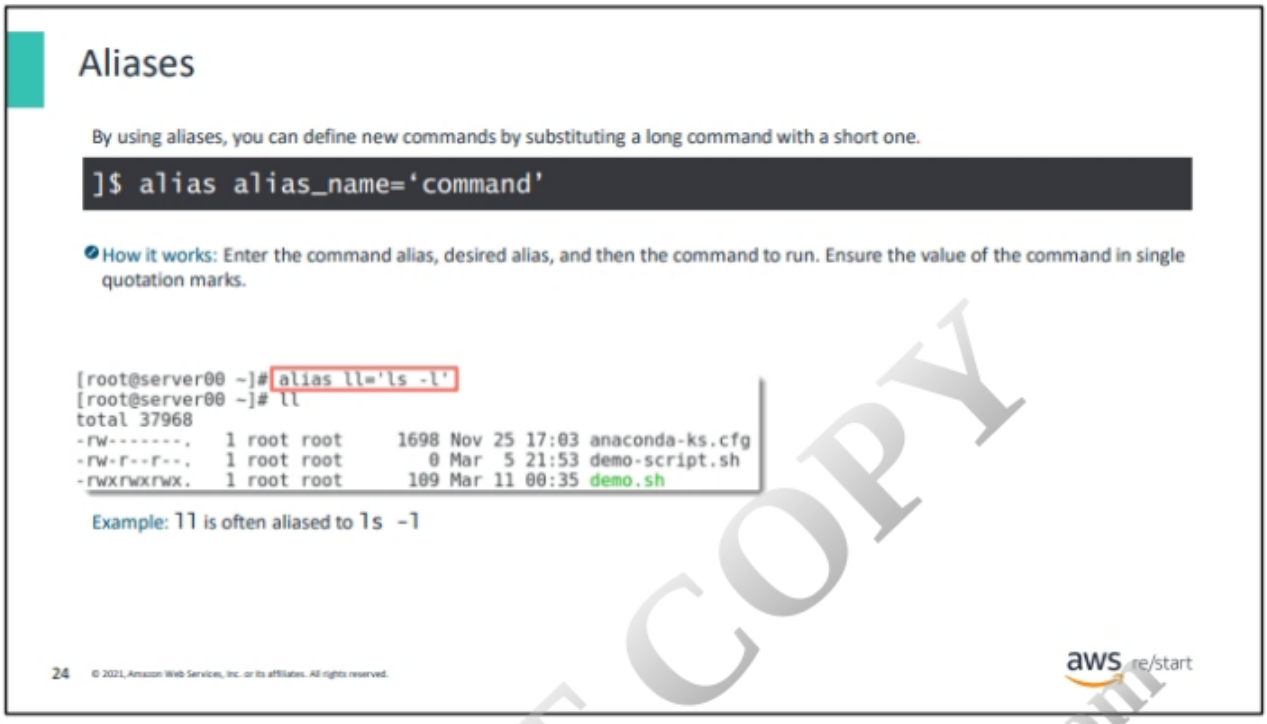
Steps to view the `.bashrc`

1. Open the Bash shell in a Linux environment.
2. At the command prompt, enter the `ls -a` command to view hidden files.
3. Enter the `cat .bashrc` command to display the content of the file.

Steps to view the `/etc/profile`

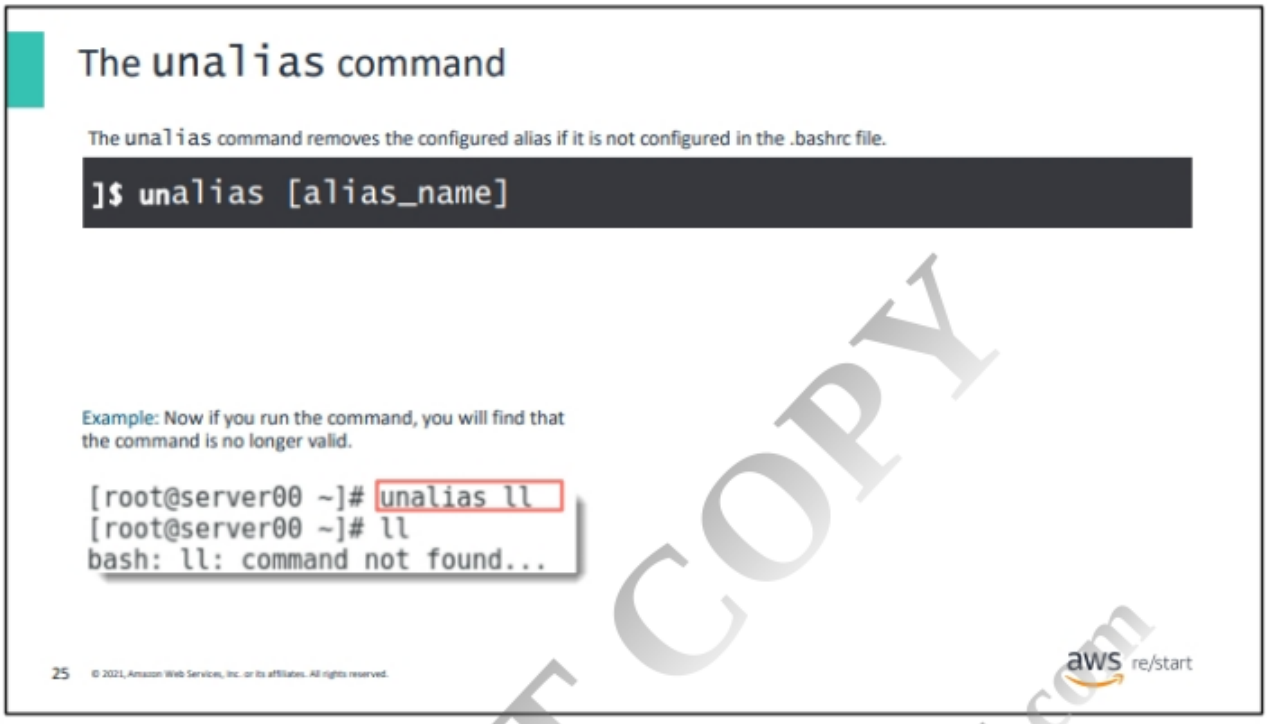
1. At the command prompt, enter `cd /etc/profile`
2. Enter `ls` to see the `profile` file.
3. Enter the `cat .profile` command to view the content of the file.

This section describes the value of the alias command.



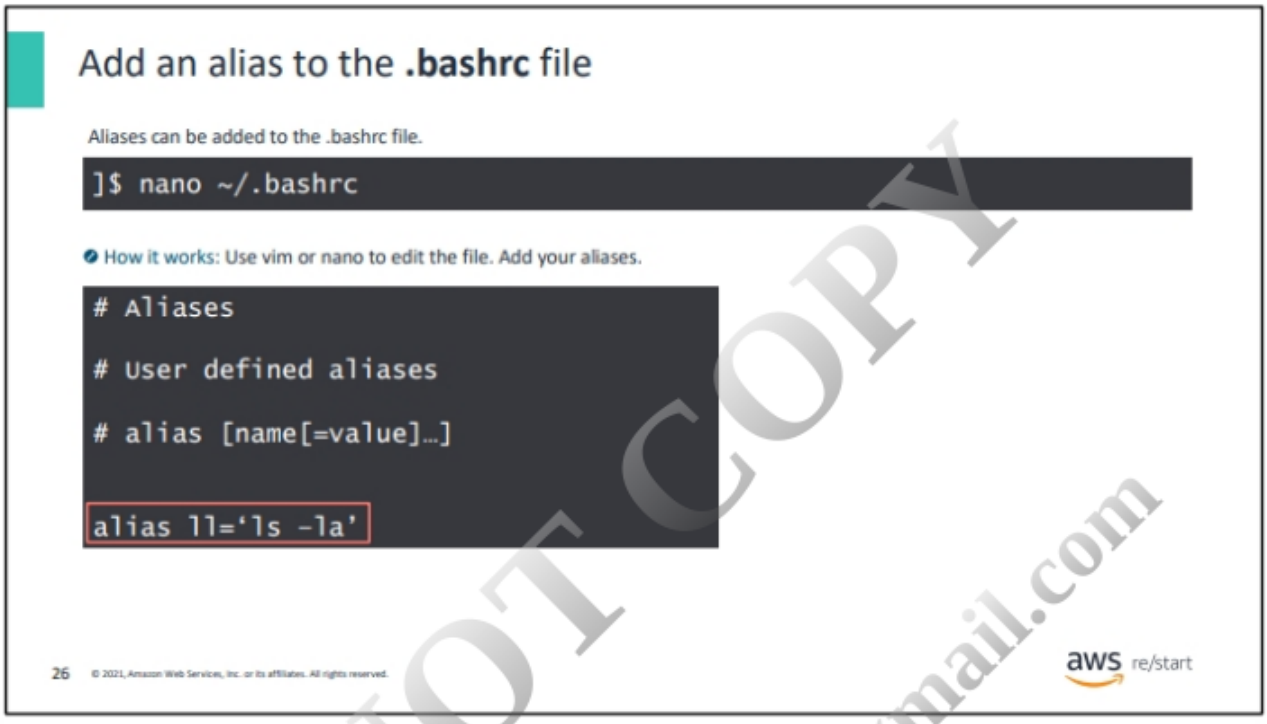
By using aliases, you can define new commands by substituting a long command with a short one. Aliases can be set temporarily in the current shell, but it is more common to set them in the user's **.bashrc** file so that they are permanent. In the example, **ll** is often substituted or aliased to **ls -l**.

Extra: In many distributions, you can create an alias of destructive commands—such as **rm**, **cp**, and **mv**—with the **-i** interactive option.



The unalias command removes the configured alias if it is not configured in the .bashrc file.

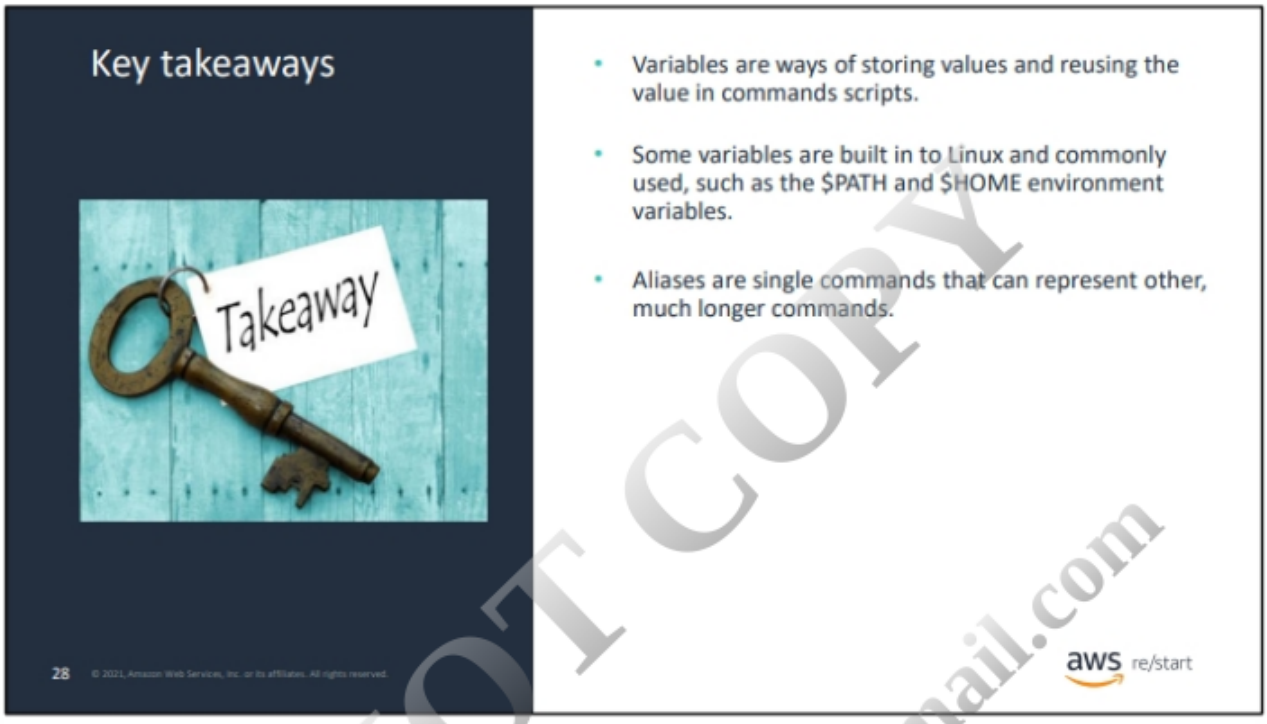
In the example, the unalias command is used to remove the new alias that was created earlier.



The **.bashrc** file is stored in the home directory of each user. As mentioned earlier, the .bashrc file is used to store configurations specific to the user. When creating an alias, remember that after you create it, the alias is applied to the .bashrc file.

1. Some commands used for aliases (other than the commands that were listed previously) include those that return the following:
   - The name of the user who is running the application
   - A user's preferences when they play a game (for example, sound and music levels)
   - A value that is commonly used when running calculations (pi, the fixed cost of product)

2. The env command displays the environment variable. You use this command to display your current environment.



Key takeaways include the following:

- Variables are ways of storing values and reusing the value in commands scripts.
- Some variables are built in to Linux and commonly used, such as the $PATH and $HOME environment variables.
- Aliases are single commands that can represent other, much longer commands.