CCNA

ICND-1 Command Reference

Somenath Sinha

Nov 2018

Contents

Part I

ICND1 - Switches

Chapter 1

Basic Cisco Catalyst Switch Configuration

1.1 Port Addressing

The addressing of ports on a Cisco Catalyst Switch follows the format:

`interface-type Stack #/Card #/Port #`

Let us consider the **Catalyst 3750** switch. This switch isn't modular and thus can't have multiple cards, but it supports Stack-wise, which is a Cisco technology to combine multiple physical switches in a stack into a single logical switch. Thus, the middle number when referring to a particular interface (i.e., port on a physical switch) will **always be 0**. Thus, the Giga-bit port that's 16th port on the 3rd stack member of switches will be written as: `GigabitEthernet 3/0/16`, or **g3/0/16** for short.

For a **Cisco 2960** switch, there's no option for Stack-wise, and hence we don't need the first number. Then, the name for the interface becomes `GigabitEthernet 0/16` or **g0/16**.

1.2 Configuring a Management IP Address

We can connect to the switch physically via the console port, or through one of the interfaces using Ethernet. In case of the later, we have to set up a management IP address through which we can use telnet or SSH to control and configure the switch.

1.2.1 View how we're connected to a Switch

To view how we're presently connected to a switch, we can use the `show line` command:

```
sw1>show line
   Tty Typ     Tx/Rx    A Modem  Roty AccO AccI   Uses   Noise  Overruns   Int
*    0 CTY              -    -      -    -    -      0       0     0/0       -
     1 AUX   9600/9600  -    -      -    -    -      0       0     0/0       -
     2 VTY              -    -      -    -    -      1       0     0/0       -
     3 VTY              -    -      -    -    -      0       0     0/0       -
     4 VTY              -    -      -    -    -      0       0     0/0       -
```

```
    5 VTY              -    -      -    -    -      0       0     0/0        -
    6 VTY              -    -      -    -    -      0       0     0/0        -
```

The **asterisk (*)** at the beginning of the line shows that it's the port currently being used. The **CTY** refers to a console port, which is the port we're using. If we were using telnet or SSH, i.e., log in remotely to the switch/router, then one of the **VTY** (*Virtual Terminal*) would've been used.

1.2.2 Privileged mode, auto-completion and showing possible commands

The > sign at the end of the prompt signals that we're currently in *user mode*. To change the configuratin on the equipment we need to be in *Privileged mode*, which is indicated by a # prompt. To enter the privileged mode, we use the command `enable`. To leave the privileged mode, we use `disable`:

```
sw1>enable
sw1#disable
sw1>
```

To go into privileged mode, we need not type the entire command. Cisco **IOS** (*Internetwork Operating System*) can show all possible commands that can follow the current string on the present line on pressing the `?` key. For example:

```
sw1>e?
enable  ethernet  exit

sw1>en?
enable
```

This also means that we don't need to type out the entire command. We can simply use short-cuts for the commands, which are the minimum number of letters in the word of a command that is unique. Thus, in the case of the `enable`, the minimum characters required are 2 since there are 3 commands starting with 'e' but only 1 starting with 'en', i.e., `enable`.

```
sw1>en
sw1#
```

1.2.3 Configuring the Terminal

While we may be in privileged mode, we still need to tell the switch that we want to configure from the terminal. For this, we use the command `configure terminal`, or **conf t** for short. This would bring us into the **global config** mode:

```
sw1#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
sw1(config)#
```

To exit the global configuration mode, we use the `exit` command:

```
sw1(config)#exit
sw1#
*Nov 11 21:36:07.353: %SYS-5-CONFIG_I: Configured from console by console
sw1#
```

1.2.4 Interface Configuration Mode

To set the management IP address we need to go to the interface configuration port. On a Cisco Switch, unlike on a PC/Laptop, every interface gets associated to the IP address which is of the network it's connected to. Layer-2 Switches are generally blind to the IP addresses since they don't make forwarding decisions on the basis of IP addresses, but this IP Address is required to be associated with the switch to uniquely identify the switch on the network for remote access through telnet/SSH.

By default, on the switch, all the ports belong to a specific **VLAN*(Virtual LAN)***. Thus, we can set the IP address of the *default VLAN*, **vlan1**, to set up the management IP address of the switch. To enter the interface configuration for vlan1, we use: `interface vlan1`, or simply **int vlan1** :

```
sw1(config)#int vlan 1
sw1(config-if)#
```

The prompt changed to `(config-if)#` to show that we've now entered the *Interface config mode*.

1.2.5 Adding a Management IP

Just like with a PC, we need to set up both the IP address as well as the Subnet Mask. Let us consider the switch needs to have the IP **192.168.1.11/24**. We can't enter the value in suffix notation. The equivalent subnet mask is *255.255.255.0*. Thus, the command becomes:

```
sw1(config-if)#ip add 192.168.1.11 255.255.255.0
sw1(config-if)#no shutdown
sw1(config-if)#
*Nov 11 21:51:57.295: %LINK-3-UPDOWN: Interface Vlan1, changed state to up
*Nov 11 21:51:58.298: %LINEPROTO-5-UPDOWN: Line protocol on Interface Vlan1, changed
↪  state to up
sw1(config-if)#
```

Typically the interface `vlan1` (i.e, the 1st virtual LAN) is turned on by default, but sometimes it may have been administratively shut-down, which we can fix using `no shutdown` command, or for short **no shut**.

1.3 Configuring a Default Gateway

A default gateway is required for the switch to determine which path to take to exit it's own subnet and reach another network. While the IP address has to be set for an interface, the default gateway has to be set in global config mode:

```
sw1(config)#ip default-gateway 192.168.1.1
```

1.4 Setting Console and VTY Passwords

To set the password for the console port login, we need to enter its configuration, i.e., console line configuration, and then set the password. Let's say we want to set it to *cisco*. We can do this by:

```
sw1(config-line)#password cisco
sw1(config-line)#login
```

The `login` command enables the password check at login. Unless it's set, even if there's a valid password, the system won't ask for credentials while logging in. Further, to disable password check, we don't have to change/remove the password. we simply need to do `no login`. The login command doesn't work unless there's a password set:

```
sw1(config-line)#login
% Login disabled on line 0, until 'password' is set
```

1.4.1 Configuring a password for the VTY lines

We could set the password for each VTY line individually, but if we're setting them (or a subset of them) all to an identical config, we can use:

```
sw1(config)#line vty 0 5
sw1(config-line)#password cisco
sw1(config-line)#login
sw1(config-line)#end
sw1#
```

This is assuming we want to set the password for lines 0-5 and not 6 of the available VTY lines. Also notice that we jump levels till we end the configuration of the switch by using `end` command. Unlike the `exit` command, the `end` command doesn't just jump up to the parent config level, but ends the configuration all together and takes us out of the configuration mode all together.

1.5 Show Running Configuration

To see the current configuration we created (but not yet saved), i.e., the configuration that the device is currently using, we use the command:

```
sw1#sh run
Building configuration...

Current configuration : 2911 bytes
!
! Last configuration change at 22:48:16 UTC Sun Nov 11 2018
!
version 15.2
service timestamps debug datetime msec
service timestamps log datetime msec
no service password-encryption
service compress-config
!
hostname sw1
...
!
interface Vlan1
 ip address 192.168.1.11 255.255.255.0
```

```
!
ip default-gateway 192.168.1.1
ip forward-protocol nd
!
...
!
line con 0
 exec-timeout 0 0
 password cisco
 login
line aux 0
line vty 0 4
 password cisco
 login
line vty 5
 password cisco
 login
line vty 6
 login
!
!
end

sw1#
```

At the very bottom we have the details for the console and the VTY lines. It's completely normal for IOS to break down the configuration for 0-6 into ranges (0-4,5,6), even though we configured them together at the same time.

1.6 Checking for connectivity using Ping

The `ping` command sends **ICMP (*Internet Control Message Protocol*)** packets to the destination device and waits for *ICMP replies* from it to determine whether the host is up. It can help detect delays and packet drops, and whether the host is even connected i.e., reachable. We can use it by:

```
PC-1> ping 192.168.1.11
84 bytes from 192.168.1.11 icmp_seq=1 ttl=255 time=3.973 ms
84 bytes from 192.168.1.11 icmp_seq=2 ttl=255 time=3.969 ms
84 bytes from 192.168.1.11 icmp_seq=3 ttl=255 time=5.951 ms
84 bytes from 192.168.1.11 icmp_seq=4 ttl=255 time=3.472 ms
84 bytes from 192.168.1.11 icmp_seq=5 ttl=255 time=10.913 ms
```

Some variation of the ping command is available on all major OS (Operating Systems).

1.7 Enabling Telnet Access

Telnet is considered a non-secure method of accessing network equipment since data is sent in plain text and packet capture utils like wireshark can capture the traffic and get access to all the data. Thus **SSH (*Secure Shell*)** is considered a much better alternative.

1.7.1 Show line details

To see the details of the VTY/CTY line, we use:

```
sw1#sh line vty 0
   Tty Typ     Tx/Rx    A Modem  Roty AccO AccI   Uses   Noise  Overruns   Int
     2 VTY              -    -      -    -    -      1       0     0/0       -

Line 2, Location: "", Type: ""
Length: 24 lines, Width: 80 columns
Baud rate (TX/RX) is 9600/9600
Status: Ready, No Exit Banner
Capabilities: none
Modem state: Ready
Group codes:    0
Special Chars: Escape  Hold  Stop  Start  Disconnect  Activation
                ^^x    none   -     -       none
Timeouts:      Idle EXEC    Idle Session   Modem Answer  Session   Dispatch
               00:10:00        never                     none     not set
                            Idle Session Disconnect Warning
                              never
                            Login-sequence User Response
                             00:00:30
                            Autoselect Initial Wait
                              not set
Modem type is unknown.
Session limit is not set.
Time since activation: never
Editing is enabled.
History is enabled, history size is 20.
DNS resolution in show commands is enabled
Full user help is disabled
Allowed input transports are lat pad telnet rlogin nasi ssh.
Allowed output transports are lat pad telnet rlogin nasi ssh.
Preferred transport is lat.
Shell: enabled
Shell trace: off
No output characters are padded
No special data dispatching characters
```

The two lines of special interest here are:

```
Allowed input transports are lat pad telnet rlogin nasi ssh.
Allowed output transports are lat pad telnet rlogin nasi ssh.
```

The **Input Transport** refer to the methods via which we're allowed to login to the router whereas the *output transports* are the methods we can use to login to another device from the router. To see all of the transports methods available we use:

```
sw1#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
sw1(config)#line vty 0 6

sw1(config-line)#transport input ?
  all     All protocols
  lat     DEC LAT protocol
  nasi    NASI protocol
  none    No protocols
  pad     X.3 PAD
  rlogin  Unix rlogin protocol
  ssh     TCP/IP SSH protocol
  telnet  TCP/IP Telnet protocol
  udptn   UDPTN async via UDP protocol
```

To only allow telnet and SSH, we use:

```
sw1(config-line)#transport input telnet ssh
sw1(config-line)#end
sw1#sh line vty 0
...
Allowed input transports are telnet ssh.
Allowed output transports are lat pad telnet rlogin nasi ssh.
...
```

1.7.2 Logging in via telnet

To login via telnet from another device, and see which vty we're using, we use the command:

```
sw2>telnet 192.168.1.11
Trying 192.168.1.11 ... Open
...
sw1>sh line
   Tty Typ     Tx/Rx    A Modem  Roty AccO AccI   Uses   Noise  Overruns   Int
*    0 CTY              -    -      -    -    -      0       0     0/0       -
     1 AUX   9600/9600  -    -      -    -    -      0       0     0/0       -
*    2 VTY              -    -      -    -    -      2       0     0/0       -
     3 VTY              -    -      -    -    -      0       0     0/0       -
     4 VTY              -    -      -    -    -      0       0     0/0       -
...
sw1>
```

Here we can see we are using VTY 2 for the telnet.

1.7.3 Keyboard Shortcuts for Telnet

Short cut	Action
`Ctrl` + **A**	Jump to the start of the line
`Ctrl` + **E**	Jump to the end of the line
`Ctrl` + **W**	Delete previous word in the line
`Ctrl` + **X**	Delete all characters before cursor.
`Ctrl` + **K**	Delete all characters from cursor till end.
↑ **Arrow**	Delete all characters before cursor.
↓ **Arrow**	Delete all characters before cursor.
`Esc` + **B**	Move cursor back 1 word.
`Esc` + **F**	Move cursor forward 1 word.

1.7.4 History Buffer

The history buffer contains the last 20 commands used by default. This can be viewed with:

```
sw1#show history
  sh line
  conf t
  en
  conf t
  sh run
  sh ver | i up
  sh line status
  sh line
  sh controllers gigabitEthernet 0/0
  sh cont g 0/0 tab
  sh cont g 0/0 tabular
  sh diag
  sh line
  sh line vty 0
  conf t
  sh line vty 0
  sh ip int br
  sh run
  sh mem
  sh history
sw1#
```

The size of the history buffer can be changed in the line configuration mode with:

```
sw1#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
sw1(config)#line vty 0 6
sw1(config-line)#history size ?
  <0-256>  Size of history buffer

sw1(config-line)#history size 50
```

Now the history buffer will store the last 50 commands.

1.7.5 IP Interface Connection Overview

The `show ip interface brief` command shows us the connection status of the interface, the IP address assigned to each interface and more.

```
sw1#sh ip int brief
Interface              IP-Address      OK? Method Status                Protocol
GigabitEthernet0/0     unassigned      YES unset  up                    up
GigabitEthernet0/1     unassigned      YES unset  up                    up
GigabitEthernet0/2     unassigned      YES unset  up                    up
GigabitEthernet0/3     unassigned      YES unset  up                    up
Vlan1                  192.168.1.11    YES manual up                    up
```

1.8 Enabling SSH Access

Secure SHell (*SSH*), unlike Telnet requires both a user-name and a password for authentication. This user-name-password can live either on an external authentication server, or locally, i.e., in the memory of the router itself. In addition, the router also needs to know which domain it's a part of.

1.8.1 Setting up User-name, password and domain name

The `username` and `password` commands take care of creating an username and password, and can be used together on the same line. The domain name is set using `ip domain-name` command. These have to be set in the global config mode:

```
sw1(config)#username cisco password cisco
sw1(config)#ip domain-name somuvmnet.com
```

The domain name is used in the creation and self-signing of the digital certificate. A public-private key pair is used to enforce authentication. They can be generated using the `crypto key generate rsa`. Note that the mod size has to be $\geq$ 768 bits for SSH 2.

```
sw1(config)#crypto key generate rsa
The name for the keys will be: sw1.somuvmnet.com
Choose the size of the key modulus in the range of 360 to 4096 for your
  General Purpose Keys. Choosing a key modulus greater than 512 may take
  a few minutes.

How many bits in the modulus [512]: 1024
% Generating 1024 bit RSA keys, keys will be non-exportable...
[OK] (elapsed time was 0 seconds)

sw1(config)#
*Nov 12 01:57:14.624: %SSH-5-ENABLED: SSH 1.99 has been enabled
sw1(config)#
```

Now we have to choose to use SSH version 2, using:

```
sw1(config)#ip ssh version 2
```

Finally we need to tell the device where to find the username and password combination. In this case, it's going to be the device memory itself. Hence, we use the `login local` command in the *line config mode*.

```
sw1(config)#line vty 0 6
sw1(config-line)#login local
```

1.8.2 Connecting via SSH

Depending on the OS, just like the `ping` command, the input options may vary for the `ssh` command, but the premise remains the same. The following is the method to do this on an IOS terminal:

```
sw2#ssh -l cisco 192.168.1.11
...
Password:
...
sw1>exit

[Connection to 192.168.1.11 closed by foreign host]
sw2#
```

1.9 Viewing Version Information

The `show version` or `sh ver` command shows us some critical information about the device itself. This includes the version of the IOS that's being used by the hardware - a critical piece of information when reporting issues to Cisco's TAC (Technical Assistance Centre).

```
sw2#sh ver
Cisco IOS Software, ... Version 15.2(20170321:233949)
...
sw2 uptime is 6 hours, 33 minutes
...
Cisco IOSv () processor (revision 1.0) with 984321K/62464K bytes of memory.
...
1 Virtual Ethernet interface
4 Gigabit Ethernet interfaces
...

sw2#
```

The command also shows the uptime of the switch, i.e., how long it's been running. Further it shows teh total amount of RAM, which in this case is $984321 + 62464$ KB of memory, i.e., $= 1046785KB \approx 1GB$ of RAM. The number and types of interfaces is also displayed.

1.10 Viewing Current Configuration

To show the entirety of the running configuration, we use the `show running-configuration` command, or `sh run` for short. There's also the option of seeing specific lines in the configuration by keyword searching using output filters.

```
sw1#sh run
Building configuration...

Current configuration : 3169 bytes
!
! Last configuration change at 02:12:13 UTC Mon Nov 12 2018
!
version 15.2
...
hostname sw1
...
!
username cisco password 0 cisco
...
!

sw1#sh run | i username
username cisco password 0 cisco
sw1#
```

The second command above uses the i| filter that only shows lines including the matching keyword. There are several more criteria, some of which are:

Short cut	Action
`<command> | i`	Only show lines that **include** the keyword
`<command> | e`	Only show lines that **exclude** the keyword
`<command> | b`	**Begin** showing all lines from the first instance of the keyword
`<command> | s`	Only show the **section** containing the keyword
`<command> | c`	**Count** the number of lines matching the RegExp

Just like a normal pager in Linux, the output of the show run command is searchable in the format `/<keyword>` where upon typing the *keyword* after the / character and pressing enter, the screen jumps to the first line containing the keyword.

```
sw1#sh run
Building configuration...
...
username cisco password 0 cisco
no aaa new-model
!
/vty
filtering...
line vty 0 4
 password cisco
 login local
 history size 50
 transport input telnet ssh
line vty 5
 password cisco
 login local
 history size 50
 transport input telnet ssh
!
end

sw1#
```

1.11 MAC Address (CAM) Table Examination

The primary objective of a Layer 2 Switch is to learn which MAC addresses live off of which port and storing them in a table called Content Addressable Memory (CAM) or MAC Address table.

Then, when it gets a packet destined for a certain MAC Address, it sends it along the corresponding port it looks up from the CAM table. If the address is unknown, it floods all the ports but the port from which the packet was received, in an effort to learn which port is associated with that MAC address and store it in the table, to aid in future lookups.

The contents of the MAC address can be viewed with the `show mac address-table` command:

```
sw1#sh mac address-table
          Mac Address Table
-------------------------------------------

Vlan    Mac Address       Type        Ports
----    -----------       --------    -----
   1    0cee.bab5.5f00    DYNAMIC     Gi0/0
   1    0cee.bacd.5000    DYNAMIC     Gi0/1
   1    0cee.bacd.5001    DYNAMIC     Gi0/0
Total Mac Addresses for this criterion: 3
```

The reason there are multiple MAC addresses associated to a certain port is because that MAC address belongs to another switch through which multiple devices can be reachable. Since that switch connects us to those MAC addresses, i.e., is a direct path, the end-user MAC addresses are mapped to the port leading up to the switch.

While our MAC addresses can be obtained beyond switches, they can't be obtained for devices beyond a router. The dynamic entries in the MAC address table can be aged-out/timed-out if no traffic was seen from that port in a certain time (by default 5mins). To see the ageing time, we use:

```
sw1#show mac address-table aging-time
Global Aging Time:  300
Vlan    Aging Time
----    ----------
```

There can also be statically mapped MAC addresses that aren't learned but hard-coded into the switch.

1.11.1 Adding a Static MAC Address

We can add a static MAC Address mapping using:

```
sw1(config)#mac address-table static a820.6332.0087 vlan 1 interface gi 0/3
sw1(config)#end
sw1#sh mac address-table
          Mac Address Table
-------------------------------------------

Vlan    Mac Address       Type        Ports
----    -----------       --------    -----
   1    0cee.bab5.5f00    DYNAMIC     Gi0/0
   1    0cee.bacd.5000    DYNAMIC     Gi0/1
   1    0cee.bacd.5001    DYNAMIC     Gi0/0
   1    a820.6332.0087    STATIC      Gi0/3
Total Mac Addresses for this criterion: 4
sw1#
```

This can be handy when we're troubleshooting and trying to ensure that the MAC address of a device has been learned off of the right port.

1.12 Setting the Hostname

The hostname of a network device acts as the name of the device in the network, helping us easily identify the device. If we have multiple switches in a network, as is often the case, the hostname lets us quickly identify and distinguish the devices. We can change the hostname using:

```
sw(config)#hostname sw1
sw1(config)#end
```

The current hostname is displayed as the first word before the prompt.

1.13 Setting the Enable Password

Since the `Enable` command lets us configure the switch, we should set a password on it. We can do this using the `enable password` command:

```
sw1(config)#enable password cisco
sw1(config)#end
sw1#
*Nov 19 05:37:32.191: %SYS-5-CONFIG_I: Configured from console by console
sw1#disable
sw1>enable
Password:
sw1#
```

This creates a password that's stored as clear-text in the running config. Anyone who can view the running config will be able to see it:

```
sw1#sh run | i cisco
enable password cisco
```

To change this, we can store an encrypted password hash, using the `enable secret` command. This has several options in terms of encryption algorithms:

```
sw1(config)#enable secret ?
  0      Specifies an UNENCRYPTED password will follow
  5      Specifies a MD5 HASHED secret will follow
  8      Specifies a PBKDF2 HASHED secret will follow
  9      Specifies a SCRYPT HASHED secret will follow
  LINE   The UNENCRYPTED (cleartext) 'enable' secret
  level  Set exec level password
```

For example, the MD5 (Message Digest 5) algorithm when applied on a word will give us a 128-bit hash value, also called a *digest*. For example, the MD5 digest of 'cat' is '`D077F244DEF8A70E5EA758BD8352FCD8`'. When someone enters the password, the switch runs the MD5 hashing algorithm on it to get it's hash. If the digest matches that already stored in the memory for the password, then the user is granted access. There's no way to get the password directly from the hash, since the hash is like a fingerprint of the password instead of a scrambled up version of the password itself. We can use the `enable secret` command (which uses MD5 as the default algorithm) as:

```
sw1(config)#enable secret somu
sw1(config)#end
sw1#sh run | i enable
enable secret 5 $1$/Rxb$NofwEzG6FNEc2v9TiU1VA1
enable password cisco
```

The `5` before the hash `$1$/Rxb$NofwEzG6FNEc2v9TiU1VA1` tells us that it's a MD5 digest. There are other algorithms available as well:

Type	Algorithm	Description
5	**MD5**	Default algorithm; produces 128bit hash; lesser security than the hashes below.
8	**PBKDF2**	Uses a variant of **Password Based Key Derivation Function 2 (*PBKDF2*)**, that itself uses **SHA-256 (Secure Hashing Algorithm - 256)** to generate a 256-bit hash; more secure than MD5.
9	**SCRYPT**	Pronounced "S-Crypt"; Like PBKDF2, but requires even more computational resources to crack.

To change the algorithm type while setting the secret we can use `enable algorithm-type` :

```
sw1(config)#enable algorithm-type ?
  md5     Encode the password using the MD5 algorithm
  scrypt  Encode the password using the SCRYPT hashing algorithm
  sha256  Encode the password using the PBKDF2 hashing algorithm

sw1(config)#enable algorithm-type sha256 secret cisco
sw1(config)#end
sw1#sh run | i enable
enable secret 8 $8$srB.hFoYl8uTbI$.h2H8Ux2Ie4F/qDgysbS43/RWT1aapN4rlNIA3/0YUc
```

1.14 Setting the exec-timeout

The default behaviour of the switch is to log out the user after a certain period of inactivity, so that no one gains access if an admin forgets to logout. This can be defined through the `exec-timeout` command, followed by the number of minutes and seconds before time-out:

```
sw1#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
sw1(config)#line con 0
sw1(config-line)#exec-timeout 5 30
```

The above sets the execution time-out period to 5m 30s. To remove it, we can use either `no exec-timeout` or `exec-timeout 0 0`.

1.15 Encrypting Passwords on Switches

If we do a `show run` after `password` command, the password for logging in will be stored in clear text just like in the case of `enable password` command.

```
sw1#sh run | s line con
line con 0
 exec-timeout 0 0
 password cisco
 login
sw1#
```

To store an encrypted version of the password (i.e., **not** a hash), we can use *password encryption service*. We do this using the `service password-encryption` command:

```
sw1(config)#service password-encryption
sw1(config)#end
sw1#sh run | s line con
line con 0
 exec-timeout 0 0
 password 7 070C285F4D06
 login
sw1#
```

1.15.1 User Privileges

Cisco's IOS allows us to set granular access to commands using the privilege levels associated with a username. There can typically be 0-15 privilege levels. However, we typically only use two values: **0: Normal user** and **15: Administrator**. We can create a new user with a set privilege level using:

```
sw1(config)#username somu privilege 15 password cisco
sw1(config)#end
sw1#sh run | i somu
username somu privilege 15 password 0 cisco
sw1#
```

Again, if we want the password to be encrypted, we use:

```
sw1(config)#serv password-encryption
sw1(config)#username somu privilege 15 password cisco
sw1(config)#end
sw1#sh run | i somu
username somu privilege 15 password 7 05080F1C2243
```

These use level 7 encryption, which is very easy to decrypt. Hence, a better method is to use the `secret` command which uses hashing just like in the case of the `enable secret` command.

1.15.2 Storing password hashes for login

If we want to use an encrypted password for login or change the algorithm type used for login, we need to have a username. The command allows us to change the algorithm type as well as set an encrypted password by:

```
sw1(config)#username somu algorithm-type scrypt secret cisco
sw1(config)#end
sw1#sh run | i somu
username somu secret 9 $9$JFpvgWWbMj8jr2$cK3k0VWA1xMpGD5Ky6Vj4URxXGgtX4jkyML86D4vuuQ
```

1.15.3 Removing a user

To remove a user, we simply need to use the `no username` command:

```
sw1(config)#no username somu
This operation will remove all username related configurations with same name.Do you want
↪  to continue? [confirm]
sw1(config)#
```

1.16 Creating a Banner

A **banner** is a section of text displayed during a specific action, such as during login. There can be several types of banners:

```
sw1(config)#banner ?
  LINE            c banner-text c, where 'c' is a delimiting character
  config-save     Set message for saving configuration
  exec            Set EXEC process creation banner
  incoming        Set incoming terminal line banner
  login           Set login banner
  motd            Set Message of the Day banner
  prompt-timeout  Set Message for login authentication timeout
  slip-ppp        Set Message for SLIP/PPP
```

The banner is added by using the `banner` command, followed by the type of banner (shown in the list above) and finally, the **delimiter**. This delimiter signals the start and end of the banner to IOS and thus, has to be a character that doesn't appear in the banner. We can set the banner with:

```
sw1(config)#banner login `
Enter TEXT message.  End with the character '`'.
+-------------------------+
| sw1 :      [ Swtich 1 ] +------------>
| Authorized Access Only! +--------------->
+-------------------------+
`
sw1(config)#end
sw1#sh run | b banner login
banner login ^C
+-------------------------+
| sw1 :      [ Swtich 1 ] +------------>
| Authorized Access Only! +--------------->
+-------------------------+
^C
!
```

During the login sessions, the banner will look like:

```
sw1 con0 is now available
Press RETURN to get started.

+-------------------------+
| sw1 :      [ Swtich 1 ] +------------>
| Authorized Access Only! +--------------->
+-------------------------+

User Access Verification
Password:
```

1.17 Specifying Port Speed and Duplex

The speed and duplex settings are critical to ensure that data is sent and received properly, and are set per interface. Depending on the interface, the speed is used to set the transmit (Tx) and Receive (Rx) rates. The typical speed settings available are:

```
sw1(config)#int g 0/1
sw1(config-if)#speed ?
  10    Force 10 Mbps operation
  100   Force 100 Mbps operation
  1000  Force 1000 Mbps operation
  auto  Enable AUTO speed configuration
```

Duplex determines whether data can be sent and received at the same time. **Full-duplex** means that data can be both sent and received at the same time, while **half-duplex** means data can only be sent or received at a time. The half-duplex mode allows **CSMA/CD (Carrier Sense Multiple Access/Collision Detection)** to work, which isn't required in full-duplex.

```
sw1(config-if)#duplex ?
  auto  Enable AUTO duplex configuration
  full  Force full duplex operation
  half  Force half-duplex operation
```

Note that both speed and duplex settings can be set to auto-negotiate. This means the two ends of the link automatically determine the best settings. However, people may choose to hard-code the speed and duplex settings in case auto-negotiation fails. Typically, if we administer devices on both sides of the link, then we choose to hard-code the settings. However, if the end user device such as a laptop is connected, then we may use auto since we don't know what settings these devices are configured with.

1.17.1 MDI-X (Medium Dependent Interface Crossover)

If we consider an RJ45 connector, the pins 1,2,3 & 5 are used for Ethernet. Here, two wires each are used for transmit and receive. In case we connected two switches using a straight-through Ethernet cable, the transmit wires on both sides will be connected, instead of the transmit wires on one side connecting to the receive wires on the other, leading to a failure of communication.

If we enable **MDI-X** it allows the switch to auto-detect which wires should be used for transmit and which for receive. To turn it on, we use the `mdix auto` command. This only is available, of course, only if the switch itself supports it.

1.18 Saving the Configuration

The running configuration isn't stored in a non-volatile memory, i.e., the changes don't survive a power-cycle. To make the changes permanent, we need to copy them to the NVRAM, by saving them as the **startup-configuration**. For this we use the command : `copy run star`:

```
sw1#copy running-config startup-config
Destination filename [startup-config]?
Building configuration...
```

```
Compressed configuration from 2781 bytes to 1574 bytes[OK]
*Nov 19 08:21:24.796: %GRUB-5-CONFIG_WRITING: GRUB configuration is being updated on
↪  disk. Please wait...
*Nov 19 08:21:25.631: %GRUB-5-CONFIG_WRITTEN: GRUB configuration was written to disk
↪  successfully.
sw1#
```

In older versions of IOS, there was a `write memory` command that did the same thing. We may still be able to use the `write mem` command, (`wr` for short):

```
sw1#wr
Building configuration...
Compressed configuration from 2781 bytes to 1574 bytes[OK]
sw1#
*Nov 19 08:25:12.887: %GRUB-5-CONFIG_WRITING: GRUB configuration is being updated on
↪  disk. Please wait...
*Nov 19 08:25:13.693: %GRUB-5-CONFIG_WRITTEN: GRUB configuration was written to disk
↪  successfully.
sw1#
```

Cisco however, may get rid of the `wr` command in the future, and hence it's suggested to use the `copy run star` command.

Chapter 2

Virtual LANs (VLANs)

2.1 Virtual LANs (VLANs) : Introduction

When interconnecting devices on a switch, we may want some of those devices to be on different logical networks or subnets. This is done by assigning them to a different **VLAN** (***Virtual Local Area Network***).

For example, in an office building, there may be a dedicated VLAN for each department, i.e., a VLAN for the accounts team and one for the sales team so that the data for one department doesn't flow through the network of another team. We can assign different switch-ports to different VLANs. When we want traffic to flow between those VLANs, a router has to be used to route traffic from one subnet to another.

2.2 VLAN Theory

Let us consider a case where we have two floors in an office which has 2 departments: Sales and Accounting. Since we don't want traffic to intermix, we need separate switches on each floor on each department. Given that we have 2 floors with 2 department each, the total number of switches is 4. For a bigger office, with more floors and department each, we'd need many more devices.

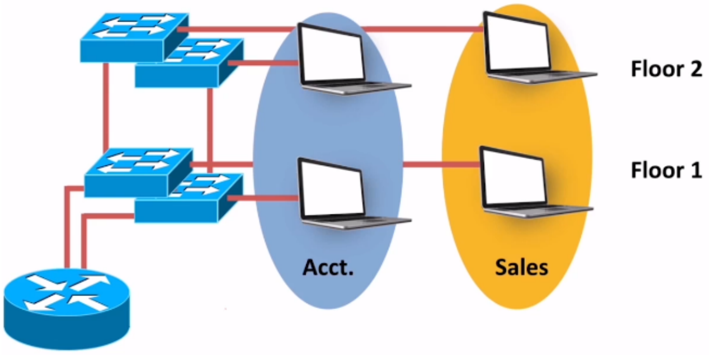

Figure 2.1: Without VLANs

Further, for a new department to be added, a new switch has to be installed on each floor for that department. A solution to this problem is given by the introduction of VLANs. While the devices are connected to the same physical network, they devices themselves are connected to different logical networks. Thus, instantly the total number of switches required falls dramatically.

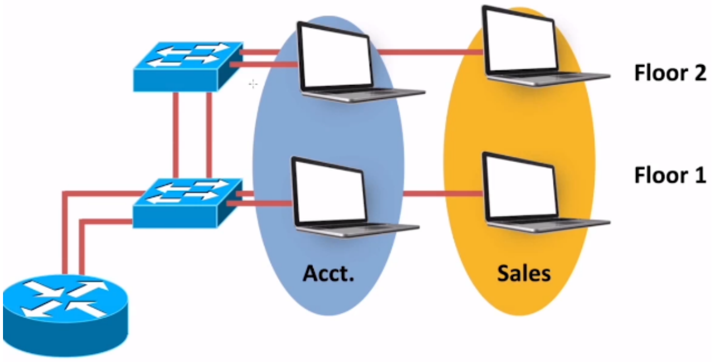

Figure 2.2: With VLANs

In either of the above cases, however, when data has to pass between floors, for example, when routing data between the Accounts and Sales subnets/broadcast domains/VLANs, there has to be one port dedicated on every switch and the router for each VLAN. However, if we use trunking, then these multiple ports can be combined into a single physical port that can carry data for multiple VLANs, while still maintaining separation.

The primary reason we may want different VLANs for different networks is because it allows us to divide the broadcast domains, which means that performance is instantly increased. Further, devices won't be able to perform a packet capture of unknown broadcast/multicast/unicast packets.

2.2.1 Packet Flow between VLANs

Let us consider the situation in diagram 2.3. We have PC$_1$ connected to VLAN$_{100}$, and PC$_2$ connected to VLAN$_{200}$, while the router is connected to the switch on a separate port, outside both VLANs, as a trunk (i.e., can carry data for multiple VLANs).

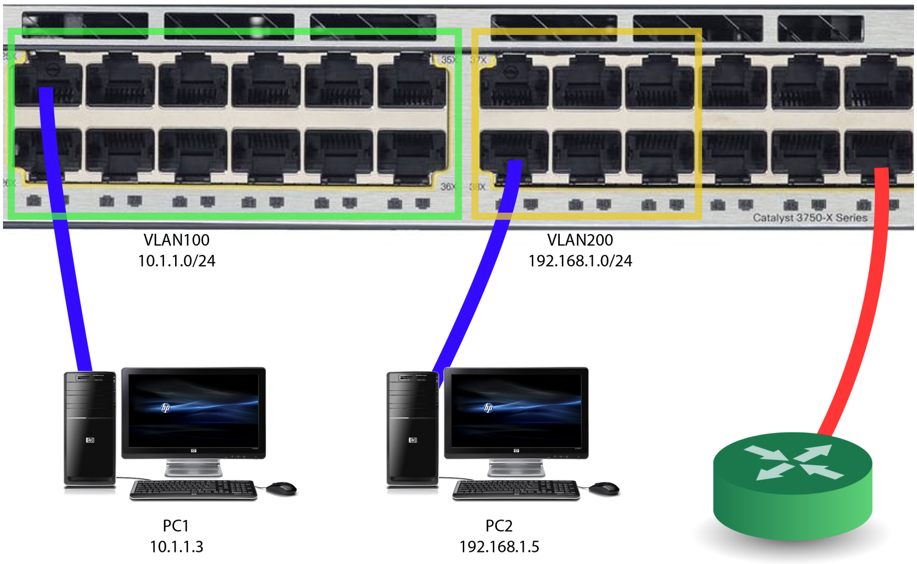

Figure 2.3: Routing between VLANs

When data has to go from VLAN$_{100}$ to VLAN$_{200}$, they can't directly pass since they're on both different subnets in different VLANs. Thus, a router is required to convey traffic from one VLAN to another.

So, data will go from the **ingress port** in VLAN$_{100}$, across the *switching fabric*, i.e., switching back-plane to the router through the trunk port. The router will discover that this packet is destined for the network of VLAN$_{200}$ (`192.168.1.0/24`), back across the trunk and then the switching fabric to reach the **egress port** in VLAN$_{200}$.

This kind of a configuration where a router is connected to a switch to manage data across VLANs is called *router on a stick* or **router on a trunk** connection. Some of the new Catalyst switches are Multilayer/Layer-3 Switches which don't need a router to perform the routing between VLANs.

2.3 VLAN Creation

2.3.1 Show existing VLANs

To see a brief overview of the existing VLANs, we use the `show vlan brief` command:

```
sw1#sh vlan brief

VLAN Name                             Status    Ports
---- -------------------------------- --------- -------------------------------
1    default                          active    Gi0/0, Gi0/1, Gi0/2, Gi0/3
1002 fddi-default                     act/unsup
1003 token-ring-default               act/unsup
1004 fddinet-default                  act/unsup
1005 trnet-default                    act/unsup
```

The fddi, token ring VLANs can be ignored. The main VLAN is the default, VLAN$_1$.

2.3.2 Creating a new VLAN

Let us say we want a couple of new VLANs - VLAN$_{100}$ for the accounts department and VLAn$_{200}$ for the sales department in an office. Just like the interface configuration mode, we also have a VLAN configuration mode. The command used to name a VLAN is `name`. We name the two VLANs using:

```
sw1(config)#vlan 100
sw1(config-vlan)#name ACCT
sw1(config-vlan)#exit
sw1(config)#vlan 200
sw1(config-vlan)#name SALES
sw1(config-vlan)#end
sw1#sh vlan br

VLAN Name                             Status    Ports
---- -------------------------------- --------- -------------------------------
1    default                          active    Gi0/0, Gi0/1, Gi0/2, Gi0/3
100  ACCT                             active
200  SALES                            active
...
```

We see that the VLANs have been created and now we can assign ports to it.

2.3.3 Deleting a VLAN

To delete a VLAN, we use the command `no vlan` followed by the VLAN number:

```
sw1(config)#vlan 300
sw1(config-vlan)#name TEST
sw1(config-vlan)#end
sw1#sh vlan br | i TEST
300  TEST                             active
sw1#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
sw1(config)#no vlan 300
sw1(config)#end
sw1#sh vlan br | i TEST
sw1#
```

Now, we may assume that the VLANs have been wiped, however, the information about VLANs are stored in a separate section of the memory, called the *flash*. We can view its contents using `show flash`. In it, we have a file called `vlan.dat`. To truly erase all the VLANs, we need to use the `delete flash:vlan.dat` command.

2.4 Assigning Ports to a VLAN

When a port is dedicated to a single VLAN, it's called an **access** port. Contrastingly, there can be **trunk** ports carrying data over multiple VLANs over a single interface. To assign a port to a VLAN, we need to declare it as an access port using the `switchport access` command. Let us consider we want to add the interface *gi0/0* to VLAN 100. We use:

```
sw1(config)#int gi0/0
sw1(config-if)#switchport access vlan 100
```

To put a range of interfaces all in the same VLAN, we use the *interface range* configuration mode instead of teh *interface* config mode. To put all ports between Gi0/1 and Gi0/3 in VLAN 300, we use:

```
sw1(config)#interface range gi0/1 - 2
sw1(config-if-range)#switchport access vlan 200
sw1(config-if-range)#end
sw1#sh vlan br

VLAN Name                             Status    Ports
---- -------------------------------- --------- -------------------------------
1    default                          active    Gi0/3
100  ACCT                             active    Gi0/0
200  SALES                            active    Gi0/1, Gi0/2
...
```

Chapter 3

Trunking

3.1 Trunking : Introduction

3.2 Trunking Theory

3.3 Trunking Modes

3.4 Creating Trunks

3.5 VLAN pruning