Universidad Autónoma de Sinaloa

ADMINISTRACIÓN DE SISTEMAS

Nombre de la práctica:
Tarea 2: Automatización y Gestión del Servidor DHCP

Grupo:

3-02

Alumno:

Montes Vázquez

Adrián Tadeo

Profesor:

Herman Geovany Ayala Zuñiga

Repositorio de GitHub:

https://github.com/ADNTD1/Administracion-de-Sistemas-T2

CONTROL DE VERSIONES:

V	Fecha	Descripción del Cambio	Autor
1.0	**2026-02-08**	**Creación de la estructura base del documento en el repo**	**Alumno**
1.1	**2026-02-08**	**creacion del readme**	**Alumno**
1.2	**2026-02-09**	**Creacion del archivo de los entregables.**	**Alumno**

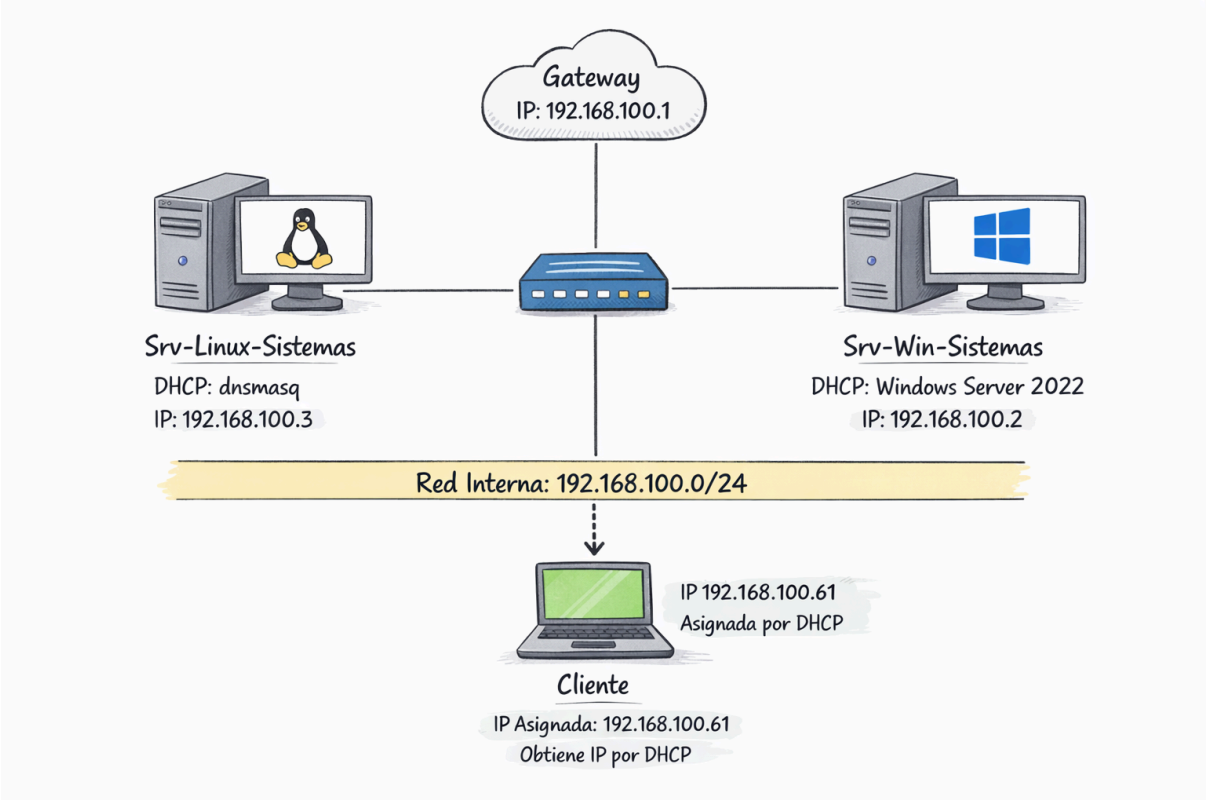
2. Introducción y Arquitectura de la Práctica

2.1 Objetivo

El objetivo de esta práctica es diseñar e implementar una solución automatizada mediante scripts en Bash y PowerShell para instalar, configurar y monitorear un servidor DHCP en entornos Linux (NixOs) y Windows seever.

La solución permite gestionar de forma dinámica el direccionamiento IP de una red interna, garantizando que los clientes reciban correctamente los parámetros de red, tales como dirección IP, puerta de enlace y tiempo de concesión, dns y gateway.

DIAGRAMA DE TOPOLOGIA DE RED:



MANUAL DE USUARIO:

Requisitos de Linux:

-Tener habilitado el servicio DHCP desde configuration.nix

-Usuario con privilegios root

-Interfaz de red confiurada desde el hipervisor

Requisitos de Windows Server:

-Ip estatica configurada

-tener rol de administrador

-interfaz de red configurada desde el hipervisor.

Instrucciones de ejecución:

Linux:

-Posicionarse en el directorio donde se encuenta el script

ejecutarlo con: `sudo bash dhcp_linux_nixos.sh`

Windows:

-Posicionarse en el directorio donde se encuentra el script

ejecutarlo con: `.\dhcp_windows.ps1`

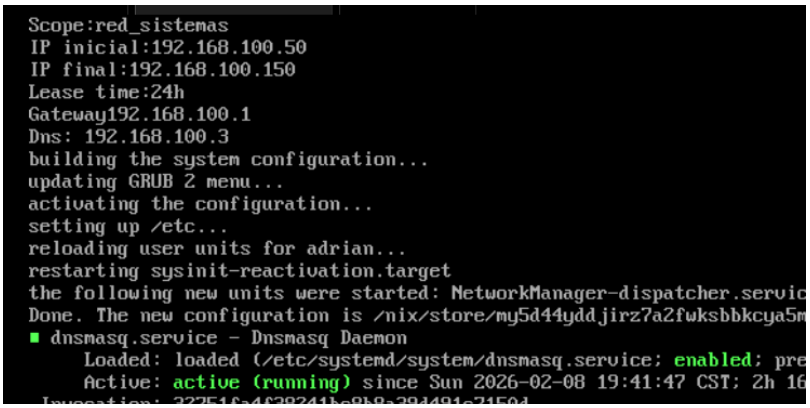
Flujo de Interacción

Durante la ejecución, los scripts solicitan al administrador:

Nombre del ámbito (Scope).
Dirección IP inicial.
Dirección IP final.
Puerta de enlace (Gateway).

Servidor DNS .
Tiempo de concesión (Lease Time).

La ejecucion de este script se mira de esta manera con la terminal pidiendote los parametros necesarios:



Explicacion del script:

El script solicita el Scope y se valida si existe, si si existe se solicitav la ip inicial que dentro hace uso de una funcion que esta en otro archivo la cual valida su formato haciendo uso de regex, si es valida pregunta la siguiente ip, despues se solicita el Lease time y dentro se convierte a un formato de segundos para que se pueda aplicar a la configuracion, despues de eso se imprime en pantalla el status del servicio, en la captura se observa que esta Active (running).

Con esa salida en pantalla se puede saber que el script se ejcuto correctamente.

4. Bitácora de Desarrollo y Configuración

Explicación del Script

-Verificación de instalación:
El script valida la existencia del servicio DHCP antes de instalarlo, garantizando idempotencia.
-Configuración dinámica:
Se solicitan parámetros al usuario para crear el scope y las opciones DHCP.
Activación del servicio:
El ámbito DHCP se activa explícitamente para permitir la asignación de IPs.

-Monitoreo:
Scripts independientes consultan el estado del servicio y las concesiones activas.

Validaciones:
Se palicaron validaciones para los rangos de la ips, por ejemplo si la ip inicial es 192.168.100.50 y la segunda es 192.168.40.150 no se puede y manda mensaje de error por que los rangos no coinciden, estas validaciones tambien funcionan para las ip reservadas por el sistema como el localhost o la 127.0.0.0

EVIDENCIAS DE CONFIGURACION:

Linux:

Archivo de configuracion del sistema operativo (configuration.nix)

```
#Servicio de DHCP en la interfaz ens37

  services.dnsmasq = {
        enable = true;


        settings = {
          interface = "ens37";
          bind-interfaces = true;
```

Salida de los servicios después de haber descargado el Feature en windows:

```
PS C:\Users\Administrator\Desktop\Scripts> Get-DhcpServerv4Scope

ScopeId         SubnetMask      Name           State    StartRange      EndRange        LeaseDuration
-------         ----------      ----           -----    ----------      --------        -------------
192.168.100.0   255.255.255.0   Scope-Red-S... Active   192.168.100.50  192.168.100.150 8.00:00:00
```

PRUEBAS DE FUNCIONAMIENTO;

Entrada	Salida Esperada	Salida Obtenida
Cliente solicita IP	**IP dentro del rango**	**192.168.100.X**
Scope activo	**Servicio DHCP operativo**	**Confirmado**

Release / Renew	**Nueva concesión**	**Exitosa**

WIndows - Linux Cliente:
Prueba de la asignacion de la IP solicitada desde el cliente (NIxOS):

```
[adrian@nixos:~/Escritorio/Scripts]$ nano test_dhcp.sh

[adrian@nixos:~/Escritorio/Scripts]$ ./test_dhcp.sh
eliminando la ip...
El dispositivo «ens37» ha sido desconectado correctamente.
Renovando ip...
El dispositivo «ens37» se activó correctamente con «ddf1bcf5-0895-34c3-8020-e5ca65794620».

nueva ip:
Ip: 192.168.100.61/24
gateway: 192.168.100.1
```

Prueba del monitoreo de red:

```
Conseciones activas:

IPAddress      ClientId          HostName AddressState LeaseExpiryTime
---------      --------          -------- ------------ ---------------
192.168.100.61 00-0c-29-7a-0c-6a nixos    Active       2/17/2026 2:34:04 PM
```

Linux Server - Linux Cliente:

Prueba de la asignacion de la ip a traves del servicio:

```
[adrian@nixos:~]$ ip route
default via 192.168.100.1 dev ens37 proto dhcp src 192.168.100.61 metric 100
192.168.100.0/24 dev ens37 proto kernel scope link src 192.168.100.61 metric 100
```

Prueba del monitor de red:

```
[adrian@Srv-Linux-Sisremas:~/Scripts/DHCP]$ ./dhcp_monitor.sh
estado del servicio:
active
concesiones activas:
IP: 192.168.100.61 Host: nixos

[adrian@Srv-Linux-Sisremas:~/Scripts/DHCP]$
```

Nueva distro:

Script para

```
 GNU nano 8.1                                                ./dhcp_linux.sh
#!/usr/bin/env bash
set -euo pipefail

# ===== Constants (RHEL 10 / KEA DHCP) =====
PKG="kea"
SVC="kea-dhcp4"
CONF="/etc/kea/kea-dhcp4.conf"
LEASES="/var/lib/kea/kea-leases4.csv"

die() { echo "ERROR: $*" >&2; exit 1; }
ok()  { echo "OK: $*"; }
info(){ echo "INFO: $*"; }

need_root() {
  [[ "$(id -u)" -eq 0 ]] || die "Ejecuta como root (sudo)."
}

have_cmd() { command -v "$1" >/dev/null 2>&1; }

# ===== IP utils =====
valid_ipv4() {
  local ip="${1:-}"
  [[ -n "$ip" ]] || return 1
  [[ "$ip" =~ ^([0-9]{1,3}\.){3}[0-9]{1,3}$ ]] || return 1
  local a b c d
  IFS='.' read -r a b c d <<<"$ip"
  for o in "$a" "$b" "$c" "$d"; do
    [[ "$o" =~ ^[0-9]+$ ]] || return 1
    (( o >= 0 && o <= 255 )) || return 1
  done
  return 0
}

ip_to_int() {
  local a b c d
  IFS='.' read -r a b c d <<<"$1"
  echo $(( (a<<24) + (b<<16) + (c<<8) + d ))
}

int_to_ip() {
  local n="$1"
  echo "$(( (n>>24)&255 )).$(( (n>>16)&255 )).$(( (n>>8)&255 )).$(( n&255 ))"
}

mask_from_ips() {
  echo "255.255.255.0" # Simplificado para Red Sistemas
```

```
  mask_from_ips() {
    echo "255.255.255.0" # Simplificado para Red Sistemas
  }

  netmask_to_prefix() {
    echo 24
  }

  net_of_ip_mask() {
    local ip="$1" mask="$2"
    local ipi mi
    ipi="$(ip_to_int "$ip")"
    mi="$(ip_to_int "$mask")"
    echo $(( ipi & mi ))
  }

  read_ip_required() {
    local label="$1" ip
    while true; do
      read -r -p "$label: " ip
      valid_ipv4 "$ip" && { echo "$ip"; return 0; }
      echo "IP invalida."
    done
  }

  read_ip_optional() {
    local label="$1" ip
    read -r -p "$label (Enter para omitir): " ip
    echo "$ip"
  }

  # ===== System / install utils =====
  is_installed() {
    rpm -q "$PKG" >/dev/null 2>&1
  }

  set_server_ip() {
    local iface="$1" ip="$2" prefix="$3"
    info "Configurando $iface con la IP del servidor: $ip/$prefix"

    # En lugar de borrar 'Wired', buscamos cualquier conexión activa en la interfaz y la modificamos
    local con_name
    con_name=$(nmcli -t -f NAME,DEVICE con show --active | grep ":$iface$" | cut -d: -f1 || echo "")
  -
  ^G Help        ^O Write Out   ^F Where Is    ^K Cut         ^T Execute     ^C Location    M-U Undo       M-A Set Mark   M-] To Bracket
```

```
write_conf() {
  local iface="$1" mask="$2" start="$3" end="$4" lease_s="$5" gw="$6" dns1="$7" dns2="$8"
  local prefix netaddr pool_start
  prefix="$(netmask_to_prefix)"
  netaddr="$(int_to_ip "$(net_of_ip_mask "$start" "$mask")")"
  pool_start="$(int_to_ip $(( $(ip_to_int "$start") + 1 )) )"

  info "Generando configuracion Kea (incluyendo parámetro 'id')..."

  cat >"$CONF" <<EOF
{
"Dhcp4": {
    "interfaces-config": { "interfaces": [ "$iface" ] },
    "lease-database": { "type": "memfile", "persist": true, "name": "$LEASES" },
    "valid-lifetime": $lease_s,
    "subnet4": [{
        "id": 1,
        "subnet": "$netaddr/$prefix",
        "pools": [ { "pool": "$pool_start - $end" } ],
        "option-data": [
            { "name": "routers", "data": "$gw" },
            { "name": "domain-name-servers", "data": "${dns1}${dns2:+, $dns2}" }
        ]
    }]
}
}
EOF
}

do_configure() {
  is_installed || die "Instala primero con -Install"

  read -r -p "Interfaz LAN (ej: ens34): " iface
  ip_start="$(read_ip_required "IP del Servidor (Inicial)")"
  ip_end="$(read_ip_required "IP Final del rango")"

  read -r -p "Horas de concesión (default 24): " h
  lease_s=$(( ${h:-24} * 3600 ))

  gw="$(read_ip_optional "Gateway")"
  dns1="$(read_ip_optional "DNS 1")"
  dns2="$(read_ip_optional "DNS 2")"
```

```
do_configure() {
  is_installed || die "Instala primero con -Install"

  read -r -p "Interfaz LAN (ej: ens34): " iface
  ip_start="$(read_ip_required "IP del Servidor (Inicial)")"
  ip_end="$(read_ip_required "IP Final del rango")"

  read -r -p "Horas de concesión (default 24): " h
  lease_s=$(( ${h:-24} * 3600 ))

  gw="$(read_ip_optional "Gateway")"
  dns1="$(read_ip_optional "DNS 1")"
  dns2="$(read_ip_optional "DNS 2")"

  set_server_ip "$iface" "$ip_start" "24"
  write_conf "$iface" "255.255.255.0" "$ip_start" "$ip_end" "$lease_s" "$gw" "$dns1" "$dns2"

  systemctl enable --now "$SVC"
  systemctl restart "$SVC"
  ok "Servidor configurado en $iface ($ip_start)."
}

main() {
  need_root
  case "${1:-}" in
    -Install)   dnf -y install kea ;;
    -Configure) do_configure ;;
    -Monitor)   [[ -f "$LEASES" ]] && column -s, -t "$LEASES" || echo "Sin leases." ;;
    *)          echo "Uso: sudo bash $0 {-Install|-Configure|-Monitor}" ;;
  esac
}

main "$@"
```

6. Conclusiones y Referencias

Lecciones Aprendidas

-Importancia de la idempotencia en scripts de automatización.
-Diferencias entre administración declarativa (NixOS) e imperativa (Windows).
-Manejo de validaciones internas del servicio DHCP en Windows Server.
-Uso de scripts de monitoreo para diagnóstico en tiempo real.

Conclusión:
Con esta practica se practica mucho el uso de Bash y Shell, ademas de comprender como se interfactua desde un cliente a un servidor DHCP, El hecho de configurar Nix Os desde ls configuracion del sistema fue mas comlpeja que solo descargar un paquete como se configura en windows, pero el manejo de todo es mas libre que el de windows con instrucciones menos complejas para el desarrollo de los scripts en este sistema operativo.
Complicaciones:
Al momento de configurar todo en linux, no podra inciarse por el manejador de paquetes que tiene NIxOS, el cual se maneja por un archivo con apartado de imports, este archivo tiene que rebuikdearse con cada cambio y por ende daba errorr y aveces duplicaba el contenido del mismo impidiendo que inicializara el servicio, con todas estas falla se opto por cabiar de distro y con ello se pudo realzar satisfacgoriamente

BIBLIOGRAFIA:

DHCP Server Overview (Windows Server)
https://learn.microsoft.com/en-us/windows-server/networking/technologies/dhcp/

DHCP Server Cmdlets in Windows PowerShell
https://learn.microsoft.com/en-us/powershell/module/dhcpserver/

NixOS
NixOS Manual (Stable)
https://nixos.org/manual/nixos/stable/

IAs utilizadas:
ChatGPT: para la configuracion de NIxOS y para la creacion de los scripts, ademas de informacion relacionada con errores del dns en la configuracion de el servicio en Windows