Switch Over Instructions

Server Setup Script - copy and paste into terminal to execute the script

Script with options

```
#!/bin/bash

# --- Single, Robust Command to Mount, Copy, Execute, and Unmount ---
#
# This command is designed to be safely copied and pasted into any server terminal.
# It handles all the necessary steps to run the main setup script from your fileserver.
# It will prompt you to select which mode to run the main script in.
#
# Password for the mount command is included. Ensure this is run in a secure environment.

# --- Configuration ---
MOUNT_POINT="/mnt/fileserver"
SERVER_PATH="//172.16.21.16/fileserver2"
SCRIPT_SOURCE_PATH="${MOUNT_POINT}/General/IT FILES/script3.sh"
SCRIPT_DEST_PATH="/tmp/script3.sh"
MOUNT_USER="Cipher.m21"
MOUNT_PASS=")\1y;634'NJ%i+"

# --- Logic ---

# Ensure the mount point is unmounted on script exit (success or failure)
trap "echo 'Unmounting fileserver...'; sudo umount '${MOUNT_POINT}' &>/dev/null || true"
EXIT

# Check if already mounted. If not, create directory and mount.
if ! grep -qs "${MOUNT_POINT}" /proc/mounts; then
    echo "Mounting fileserver..."
    sudo mkdir -p "${MOUNT_POINT}"
    sudo mount -t cifs "${SERVER_PATH}" "${MOUNT_POINT}" -o
```

```
username="${MOUNT_USER}",password="${MOUNT_PASS}"
fi

echo "Copying script (overwriting if exists)..."
sudo cp "${SCRIPT_SOURCE_PATH}" "${SCRIPT_DEST_PATH}"

echo "Making script executable..."
sudo chmod +x "${SCRIPT_DEST_PATH}"

# --- Interactive Mode Selection ---
echo ""
echo "Please choose which setup to run:"
echo "  1) Full Setup (default)"
echo "  2) Development Stack Only (--dev)"
echo "  3) Security Hardening Only (--security)"
echo "  4) Shell & UX Setup Only (--shell)"
echo "  5) System Updates Only (--updates)"
read -rp "Enter your choice [1-5]: " run_choice

EXECUTION_FLAG="--full" # Default value

case "$run_choice" in
    2) EXECUTION_FLAG="--dev" ;;
    3) EXECUTION_FLAG="--security" ;;
    4) EXECUTION_FLAG="--shell" ;;
    5) EXECUTION_FLAG="--updates" ;;
    1) EXECUTION_FLAG="--full" ;;
    *) # Default to full for any other input
       echo "Invalid choice or no choice entered. Defaulting to Full Setup."
       EXECUTION_FLAG="--full"
       ;;
esac

echo "Executing the main setup script with flag: ${EXECUTION_FLAG}..."
sudo "${SCRIPT_DEST_PATH}" "${EXECUTION_FLAG}"

# The trap will handle the unmount automatically.
echo "Script execution finished. Unmounting is handled automatically."
```

Simple script

```
sudo mkdir -p /mnt/fileserver && \
sudo mount -t cifs //172.16.21.16/fileserver2 /mnt/fileserver -o
username="Cipher.m21",password=")\1y;634'NJ%i+" && \
cp /mnt/fileserver/General/IT\ FILES//script3.sh /tmp/ && \
sudo chmod +x /tmp/script3.sh && \
sudo /tmp/script3.sh --full && \
sudo umount /mnt/fileserver
```

Server Setup Script

```
#!/usr/bin/env bash
##########################
#
####
######################
#
# Comprehensive Domain Join & Configuration Script
#
# Version: 6.5 (Phoenix - The Final Cut)
# Last Modified: 2025-07-31
#
# Features
# [CRITICAL FIX] Zsh theme switching and plugin enabling is now fully robust.
# [CRITICAL FIX] Reboot prompt no longer hangs.
# [FIX] Enhanced Nano with persistent status bar and comprehensive syntax highlighting
from a dedicated repository.
# [FIX] Vi/Vim syntax highlighting is now guaranteed by installing a full vim package.
# [ENH] Added a pre-configured "Powerline" theme for Starship, showing date, time, and
hostname.
# [ENH] Added Powerlevel10k as a Zsh theme option with automatic installation and
configuration wizard setup.
# [CRITICAL FIX] Proxy variables are now exported immediately, fixing all subsequent
download failures.
# [CRITICAL FIX] Ctrl+C cancellation is now robust and reliably skips optional sections
```

```
without exiting the script.
#
################################
####################################
#---
# CONFIGURATION
# Adjust these variables for your environment!
#---
DOMAIN_FQDN="m21.gov.local"
DOMAIN_NETBIOS="M21" # NetBIOS name of your domain
DC_DNS_IP="172.16.21.161" # Your Domain Controller's IP (for DNS & domain ops)
NTP_SERVER="172.16.121.9" # Your dedicated NTP server IP
FILE_SERVER_IP="172.16.21.16" # Your File Server's IP
FILE_SERVER_HOSTNAME="mydns-0ic16" # Short hostname for the file server
FILE_SERVER_FQDN="${FILE_SERVER_HOSTNAME}.${DOMAIN_FQDN}" # FQDN for the file server

HTTP_PROXY_URL="http://172.40.4.14:8080/" # Set to "" if no proxy
NO_PROXY_INITIAL="127.0.0.1,localhost,localhost.localdomain" # Base no_proxy entries
NO_PROXY_CUSTOM="172.30.0.0/20,172.26.21.0/24,172.16.121.0/24,10.21.0.0/21" # Your custom
NO PROXY CIDRS
INSECURE_REGISTRIES='"172.16.121.119:5000", "docker-repo.mydns.gov.tt"' # Comma-separated,
quoted Docker insecure registries

TIMEZONE="America/Port_of_Spain" # Your desired timezone
AD_SUDO_GROUP_RAW_NAME="ICT Staff SG" # AD Group for Sudoers (Raw name, spaces are okay
here. Script will escape.)
#---
# INITIALIZE SCRIPT
#---
# Color Definitions
RED='\033[0;31m'
GREEN='\033[0;32m'
YELLOW='\033[0;33m'
BLUE='\033[0;34m'
PURPLE='\033[0;35m'
CYAN='\033[0;36m'
NC='\033[0m' # No Color
# Global flag for reboot
REBOOT_REQUIRED_FLAG=false
```

```
# Exit on error for most commands, but we will handle some manually.
set -o pipefail

LOG_FILE="/var/log/setup-domain-$(date +%Y-%m-%d_%H-%M-%S).log"
# Log to file, but keep stderr on the console to see errors immediately.
exec > >(tee -a "$LOG_FILE") 2>&1

echo -e "${GREEN}=== Script started at $(date --iso-8601=seconds) by $(whoami) ===${NC}"
echo -e "${GREEN}=== Logging to ${LOG_FILE} ===${NC}"
#---
# PRELIMINARY CHECKS & GLOBAL VARIABLES
#---
[[ $EUID -ne 0 ]] && { echo -e "${RED}ERROR: This script must be run as root or with
sudo.${NC}" >&2; exit 1; }
PKG_MANAGER=""
if command -v dnf &>/dev/null; then PKG_MANAGER="dnf";
elif command -v yum &>/dev/null; then PKG_MANAGER="yum";
elif command -v apt-get &>/dev/null; then PKG_MANAGER="apt";
else echo -e "${RED}ERROR: Neither DNF, YUM, nor APT package manager found. Exiting.${NC}"
>&2; exit 1; fi
source /etc/os-release
OS_ID_LOWER=$(echo "$ID" | tr '[:upper:]' '[:lower:]')
OS_VER="${VERSION_ID%%.*}"

HOSTNAME_VAR=$(hostname -f)

#---
# SCRIPT FUNCTIONS
#
log_step() { echo -e "\n${GREEN}--- [STEP $1 on ${PURPLE}${HOSTNAME_VAR}${NC}] $2 ---
${NC}"; }

#
# SECTION 0: CREDENTIAL GATHERING
#
gather_credentials() {
    log_step "0/X" "Gathering Credentials (Not Logged)"

    local creds_ok=false
```

```
    while ! $creds_ok; do
        ( # Start subshell for cancellable read
            trap 'echo -e "\n${RED}Credential entry cancelled. Exiting script.${NC}"; exit
1;' INT
            echo -e "\n${CYAN}--- Domain Credentials (will not be logged) ---${NC}"
            read -rp "$(echo -e "${CYAN}Enter the SERVICE part of the hostname (e.g.,
mydns-it-c12-1): ${NC}")" SERVICE_NAME_PART < /dev/tty
            if [[ ! "$SERVICE_NAME_PART" =~ ^[a-zA-Z0-9-]+$ ]]; then
                echo -e "${RED}ERROR: Invalid service name part.${NC}" >&2; exit 1;
            fi
            read -rp "$(echo -e "${CYAN}Enter your AD username SUFFIX (the part after
'ent_'): ${NC}")" AD_USER_SUFFIX < /dev/tty
            if [ -z "$AD_USER_SUFFIX" ]; then echo -e "${RED}ERROR: AD username suffix
cannot be empty.${NC}" >&2; exit 1; fi

            read -rsp "$(echo -e "${CYAN}Enter AD password for 'ent_${AD_USER_SUFFIX}':
${NC}")" AD_PASSWORD_TEMP < /dev/tty
            echo
            if [ -z "$AD_PASSWORD_TEMP" ]; then echo -e "${RED}ERROR: AD Password cannot
be empty.${NC}" >&2; exit 1; fi

            # Export variables from subshell to the main script via a temp file
            echo "export SERVICE_NAME_PART='${SERVICE_NAME_PART}'" > /tmp/creds.sh
            echo "export AD_USER_SUFFIX='${AD_USER_SUFFIX}'" >> /tmp/creds.sh
            echo "export AD_PASSWORD='${AD_PASSWORD_TEMP}'" >> /tmp/creds.sh
        )

        if [ $? -ne 0 ]; then exit 1; fi

        source /tmp/creds.sh
        rm /tmp/creds.sh
        creds_ok=true
    done

    TARGET_HOSTNAME_FQDN="${SERVICE_NAME_PART}.${DOMAIN_FQDN}"
    TARGET_HOSTNAME_FQDN_LC=$(echo "$TARGET_HOSTNAME_FQDN" | tr '[:upper:]' '[:lower:]')
    AD_USER_FOR_JOIN="ent_${AD_USER_SUFFIX}"
    echo -e "${BLUE}INFO:${NC} Using full AD username: ${PURPLE}${AD_USER_FOR_JOIN}${NC}"
```

```
NO_PROXY_FULL="${NO_PROXY_INITIAL},${DOMAIN_FQDN,,},.${DOMAIN_FQDN,,},${DC_DNS_IP},${NTP_SERVER},${FILE_SERVER_IP}"
    if [[ -n "$NO_PROXY_CUSTOM" ]]; then NO_PROXY_FULL="${NO_PROXY_FULL},${NO_PROXY_CUSTOM}"; fi
    NO_PROXY_FULL=$(echo "$NO_PROXY_FULL" | tr ',' '\n' | sort -u | tr '\n' ',' | sed 's/,$//')
}
#
#SECTION 1: CORE SYSTEM & NETWORK FUNCTIONS
#
change_hostname() {
    log_step "1/X" "Setting Hostname"
    echo -e "${BLUE}INFO:${NC} Setting hostname to ${PURPLE}${TARGET_HOSTNAME_FQDN_LC}${NC}"
    hostnamectl set-hostname "$TARGET_HOSTNAME_FQDN_LC"
    echo -e "${GREEN}SUCCESS:${NC} Hostname set to: ${PURPLE}$(hostnamectl hostname)${NC}"
}
configure_proxy() {
    log_step "2/X" "Configuring System-Wide Proxy"
    if [ -z "$HTTP_PROXY_URL" ]; then
        echo -e "${BLUE}INFO:${NC} HTTP_PROXY_URL is not set. Skipping proxy configuration."
        return
    fi

    echo -e "${BLUE}INFO:${NC} Applying proxy for current script session..."
    export http_proxy="${HTTP_PROXY_URL}"
    export https_proxy="${HTTP_PROXY_URL}"
    export ftp_proxy="${HTTP_PROXY_URL}"
    export no_proxy="${NO_PROXY_FULL}"
    export HTTP_PROXY="${HTTP_PROXY_URL}"
    export HTTPS_PROXY="${HTTP_PROXY_URL}"
    export FTP_PROXY="${HTTP_PROXY_URL}"
    export NO_PROXY="${NO_PROXY_FULL}"

    echo -e "${BLUE}INFO:${NC} Configuring proxy for future interactive shells (/etc/profile.d/proxy.sh)..."
    cat > /etc/profile.d/proxy.sh <<EOF
```

```
export http_proxy="${HTTP_PROXY_URL}"
export https_proxy="${HTTP_PROXY_URL}"
export ftp_proxy="${HTTP_PROXY_URL}"
export no_proxy="${NO_PROXY_FULL}"
export HTTP_PROXY="\${http_proxy}"
export HTTPS_PROXY="\${https_proxy}"
export FTP_PROXY="\${ftp_proxy}"
export NO_PROXY="\${no_proxy}"
EOF
    chmod +x /etc/profile.d/proxy.sh

    echo -e "${BLUE}INFO:${NC} Configuring system-wide environment file
(/etc/environment)..."
    sed -i '/^http_proxy=/d;/^https_proxy=/d;/^ftp_proxy=/d;/^no_proxy=/d'
/etc/environment
    sed -i '/^HTTP_PROXY=/d;/^HTTPS_PROXY=/d;/^FTP_PROXY=/d;/^NO_PROXY=/d'
/etc/environment

    echo "http_proxy=\"${HTTP_PROXY_URL}\"" >> /etc/environment
    echo "https_proxy=\"${HTTP_PROXY_URL}\"" >> /etc/environment
    echo "ftp_proxy=\"${HTTP_PROXY_URL}\"" >> /etc/environment
    echo "no_proxy=\"${NO_PROXY_FULL}\"" >> /etc/environment
    echo "HTTP_PROXY=\"${HTTP_PROXY_URL}\"" >> /etc/environment
    echo "HTTPS_PROXY=\"${HTTP_PROXY_URL}\"" >> /etc/environment
    echo "FTP_PROXY=\"${HTTP_PROXY_URL}\"" >> /etc/environment
    echo "NO_PROXY=\"${NO_PROXY_FULL}\"" >> /etc/environment

    echo -e "${BLUE}INFO:${NC} Configuring package manager proxy..."
    case "$PKG_MANAGER" in
        dnf|yum)
            if ! grep -q "proxy=" /etc/dnf/dnf.conf; then
                echo "proxy=${HTTP_PROXY_URL}" >> /etc/dnf/dnf.conf
            fi
            ;;
        apt)
            cat > /etc/apt/apt.conf.d/80proxy <<EOF
Acquire::http::proxy "${HTTP_PROXY_URL}";
Acquire::https::proxy "${HTTP_PROXY_URL}";
Acquire::ftp::proxy "${HTTP_PROXY_URL}";
```

```
EOF
            ;;
    esac
    echo -e "${GREEN}SUCCESS:${NC} System-wide proxy configured."
}
configure_dns_and_hosts() {
    log_step "3/X" "Configuring DNS and NetworkManager"
    echo -e "${BLUE}INFO:${NC} Configuring /etc/hosts file..."
    sed -i "/${DOMAIN_FQDN}/d" /etc/hosts
    cat >> /etc/hosts <<EOF
# AD Domain Configuration
${DC_DNS_IP}   ${DOMAIN_FQDN}
${FILE_SERVER_IP}  ${FILE_SERVER_FQDN} ${FILE_SERVER_HOSTNAME}
EOF
    echo -e "${BLUE}INFO:${NC} Configuring DNS via NetworkManager..."
    local conn
    conn=$(nmcli -t -f NAME,DEVICE connection show --active | grep -v "lo$" | head -n1 |
cut -d':' -f1)
    if [ -z "$conn" ]; then
        echo -e "${RED}ERROR:${NC} Could not find an active network connection to
configure." >&2
        return 1
    fi
    echo -e "${BLUE}INFO:${NC} Modifying connection: ${PURPLE}${conn}${NC}"
    nmcli connection modify "$conn" ipv4.dns "$DC_DNS_IP"
    nmcli connection modify "$conn" ipv4.ignore-auto-dns yes
    nmcli connection up "$conn"
    echo -e "${GREEN}SUCCESS:${NC} DNS configured to ${DC_DNS_IP} and /etc/hosts updated."
}
check_connectivity() {
    log_step "4/X" "Checking Network Connectivity"
    local has_error=0
    echo -e "${BLUE}INFO:${NC} Pinging Domain Controller (${DC_DNS_IP})..."
    if ! ping -c 3 "$DC_DNS_IP"; then
        echo -e "${RED}ERROR:${NC} Domain Controller is not reachable." >&2; has_error=1
    fi
    echo -e "${BLUE}INFO:${NC} Checking DNS resolution for ${DOMAIN_FQDN}..."
    if ! getent hosts "$DOMAIN_FQDN"; then
        echo -e "${RED}ERROR:${NC} Could not resolve domain FQDN." >&2; has_error=1
```

```
    fi
    if [ -n "$HTTP_PROXY_URL" ]; then
        echo -e "${BLUE}INFO:${NC} Testing connection to google.com via proxy..."
        if ! curl -s --head --connect-timeout 5 http://www.google.com | head -n 1 | grep
"200 OK" > /dev/null; then
            echo -e "${YELLOW}WARNING:${NC} Could not connect to the internet via proxy.
External repos may fail."
        fi
    fi
    if [ $has_error -eq 0 ]; then
        echo -e "${GREEN}SUCCESS:${NC} All connectivity checks passed."
    else
        echo -e "${RED}ERROR:${NC} One or more connectivity checks failed. Please review
the logs." >&2; exit 1
    fi
}
install_packages() {
    log_step "5/X" "Installing Core & Utility Packages"
    local common_pkgs="nano curl wget htop btop net-tools git zip unzip tar tmux chrony
open-vm-tools traceroute ncdu policycoreutils-python-utils logrotate tree bash-completion
bat jq fontconfig util-linux-user"
    local pkgs_to_install
    if [[ "$PKG_MANAGER" == "dnf" || "$PKG_MANAGER" == "yum" ]]; then
        common_pkgs+=" bind-utils dnf-utils vim-enhanced"
        echo -e "${BLUE}INFO:${NC} Ensuring core DNF plugins are installed..."
        $PKG_MANAGER -y install dnf-plugins-core
        echo -e "${BLUE}INFO:${NC} Enabling CRB/PowerTools repository..."
        if [[ "$OS_VER" -ge 9 ]]; then
            dnf config-manager --set-enabled crb -y
        else
            dnf config-manager --set-enabled powertools -y || dnf config-manager --set-
enabled PowerTools -y
        fi
        echo -e "${BLUE}INFO:${NC} Installing EPEL repository..."
        if ! $PKG_MANAGER -y install epel-release; then
            echo -e "${RED}ERROR: Failed to install EPEL repository. Cannot
continue.${NC}" >&2; exit 1;
        fi
        local dnf_base_pkgs="realmd sssd oddjob oddjob-mkhomedir adcli samba-common-tools
```

```
authselect"
        pkgs_to_install="${dnf_base_pkgs} ${common_pkgs}"
    elif [[ "$PKG_MANAGER" == "apt" ]]; then
        common_pkgs+=" dnsutils debian-goodies vim"
        [[ "$OS_ID_LOWER" == "ubuntu" ]] && common_pkgs=${common_pkgs/bat/batcat}
        local apt_base_pkgs="realmd sssd sssd-tools libnss-sss libpam-sss adcli samba-
common-bin oddjob oddjob-mkhomedir packagekit apt-transport-https ca-certificates
software-properties-common gnupg lsb-release"
        pkgs_to_install="${apt_base_pkgs} ${common_pkgs}"
        echo -e "${BLUE}INFO:${NC} Updating package lists for APT..."
        apt-get update -qq
    fi
    echo -e "${BLUE}INFO:${NC} Installing main packages..."
    if ! $PKG_MANAGER -y install ${pkgs_to_install}; then
        echo -e "${RED}ERROR: Package installation failed. This is often due to network,
proxy, or repository issues.${NC}" >&2; exit 1;
    fi
    if command -v batcat &>/dev/null && ! command -v bat &>/dev/null; then
        ln -sf /usr/bin/batcat /usr/local/bin/bat
    fi
    echo -e "${GREEN}SUCCESS:${NC} Core packages installed."
}
configure_time() {
    log_step "6/X" "Configuring System Time (NTP & Timezone)"
    echo -e "${BLUE}INFO:${NC} Setting timezone to ${TIMEZONE}..."
    timedatectl set-timezone "$TIMEZONE"
    echo -e "${BLUE}INFO:${NC} Configuring chrony to use NTP server ${NTP_SERVER}..."
    sed -i '/^pool/d' /etc/chrony.conf
    sed -i '/^server/d' /etc/chrony.conf
    echo "server ${NTP_SERVER} iburst" >> /etc/chrony.conf
    echo -e "${BLUE}INFO:${NC} Restarting and enabling chronyd service..."
    systemctl restart chronyd
    systemctl enable chronyd
    timedatectl set-ntp true
    echo -e "${GREEN}SUCCESS:${NC} System time configured."
}
#
#SECTION 2: DOMAIN & AUTHENTICATION FUNCTIONS
#
```

```
join_ad_domain() {
    log_step "7/X" "Joining Active Directory Domain"
    ( # Start subshell for cancellation
        trap 'echo -e "\n${YELLOW}Operation cancelled. Skipping Domain Join...${NC}"; exit
0;' INT

        if realm list | grep -q "$DOMAIN_FQDN"; then
            read -rp "$(echo -e "${YELLOW}WARNING:${NC} Server is already joined to
${DOMAIN_FQDN}. Action? ([S]kip, [R]e-join): ${NC}")" choice < /dev/tty
            case "$(echo "$choice" | tr '[:upper:]' '[:lower:]')" in
                r|re-join)
                    echo -e "${BLUE}INFO:${NC} Leaving the domain first..."
                    if ! realm leave; then
                        echo -e "${RED}ERROR:${NC} Failed to leave the domain. Please
check logs. Aborting re-join." >&2
                        exit 1
                    fi
                    echo -e "${GREEN}SUCCESS:${NC} Left the domain."
                    ;;
                *)
                    echo -e "${BLUE}INFO:${NC} Skipping domain join step."
                    exit 0
                    ;;
            esac
        fi

        echo -e "${BLUE}INFO:${NC} Attempting to join domain ${DOMAIN_FQDN} as user
${AD_USER_FOR_JOIN}..."
        if ! echo -n "$AD_PASSWORD" | realm join --user="$AD_USER_FOR_JOIN"
"$DOMAIN_FQDN"; then
            echo -e "${RED}ERROR:${NC} Failed to join the Active Directory domain." >&2
            echo -e "${YELLOW}Check username, password, and connectivity to the DC.${NC}"
>&2
            exit 1
        fi
        echo -e "${GREEN}SUCCESS:${NC} Successfully joined the domain."
    )
}
configure_sssd_mkhomedir() {
```

```
    log_step "8/X" "Configuring SSSD & Home Directories"
    local sssd_conf="/etc/sssd/sssd.conf"
    if [ ! -f "$sssd_conf" ]; then
        echo -e "${RED}ERROR:${NC} SSSD configuration file not found at ${sssd_conf}" >&2;
return 1;
    fi
    echo -e "${BLUE}INFO:${NC} Modifying ${sssd_conf}..."
    sed -i '/^use_fully_qualified_names/d' "$sssd_conf"
    sed -i "/\[domain\/${DOMAIN_FQDN,,}\]/a use_fully_qualified_names = False"
"$sssd_conf"
    sed -i '/^fallback_homedir/d' "$sssd_conf"
    sed -i "/\[domain\/${DOMAIN_FQDN,,}\]/a fallback_homedir = /home/%u" "$sssd_conf"
    echo -e "${BLUE}INFO:${NC} Enabling automatic home directory creation..."
    authselect enable-feature with-mkhomedir
    systemctl restart sssd oddjobd
    systemctl enable oddjobd
    echo -e "${GREEN}SUCCESS:${NC} SSSD and home directory creation configured."
}
configure_sudoers() {
    log_step "9/X" "Configuring Sudoers for AD Group"
    local escaped_group_name
    escaped_group_name=$(echo "$AD_SUDO_GROUP_RAW_NAME" | sed 's/ /\\ /g')
    local sudoer_file="/etc/sudoers.d/90-ad-admins"
    echo -e "${BLUE}INFO:${NC} Granting sudo rights to AD group
'${AD_SUDO_GROUP_RAW_NAME}'..."
    echo "\"%${escaped_group_name}\" ALL=(ALL) ALL" > "$sudoer_file"
    chmod 440 "$sudoer_file"
    echo -e "${GREEN}SUCCESS:${NC} Sudoers configured. Rule added to ${sudoer_file}."
}
#
#SECTION 3: SECURITY HARDENING FUNCTIONS
#
optimize_sshd() {
    log_step "10/X" "Optimizing SSH Daemon for Faster Logins"
    local sshd_config="/etc/ssh/sshd_config"
    echo -e "${BLUE}INFO:${NC} Setting 'UseDNS no' in ${sshd_config}..."
    if grep -q "^#\?UseDNS" "$sshd_config"; then
        sed -i 's/^#\?UseDNS.*/UseDNS no/' "$sshd_config"
    else
```

```
        echo "UseDNS no" >> "$sshd_config"
    fi
    systemctl restart sshd
    echo -e "${GREEN}SUCCESS:${NC} SSHD optimized for faster logins."
}
install_fail2ban() {
    log_step "11/X" "Installing Fail2ban (Optional, Ctrl+C to skip)"
    ( # Start subshell for cancellation
        trap 'echo -e "\n${YELLOW}Operation cancelled. Skipping Fail2ban...${NC}"; exit
0;' INT

        if [ -f /etc/fail2ban/jail.local ]; then
            read -rp "$(echo -e "${YELLOW}WARNING:${NC} Fail2ban configuration already
exists. Action? ([S]kip, [O]verwrite): ${NC}")" choice < /dev/tty
            if [[ "$(echo "$choice" | tr '[:upper:]' '[:lower:]')" != "o" ]]; then
                echo -e "${BLUE}INFO:${NC} Skipping Fail2ban setup."; exit 0;
            fi
        else
            read -rp "$(echo -e "${CYAN}Install and configure Fail2ban for SSH protection?
[y/N]: ${NC}")" choice < /dev/tty
            if [[ "$(echo "$choice" | tr '[:upper:]' '[:lower:]')" != "y" ]]; then
                echo -e "${BLUE}INFO:${NC} Skipping Fail2ban installation."; exit 0;
            fi
        fi

        echo -e "${BLUE}INFO:${NC} Installing Fail2ban..."
        $PKG_MANAGER -y install fail2ban
        echo -e "${BLUE}INFO:${NC} Creating local jail configuration for SSHD..."
        cat > /etc/fail2ban/jail.local <<EOF
[sshd]
enabled = true
port = ssh
filter = sshd
logpath = /var/log/auth.log
maxretry = 3
bantime = 3600
EOF
        systemctl enable --now fail2ban
        echo -e "${GREEN}SUCCESS:${NC} Fail2ban installed and enabled for SSHD."
```

```
    )
}
#
#SECTION 4: SHELL & USER EXPERIENCE FUNCTIONS
#
install_nano_syntax() {
    (
        trap 'echo -e "\n${YELLOW}Operation cancelled. Skipping extra Nano
syntax...${NC}"; exit 0;' INT
        echo -e "${BLUE}INFO:${NC} Installing enhanced syntax highlighting for Nano..."
        local nano_syntax_dir="/tmp/nanorc"
        if git clone https://github.com/scopatz/nanorc.git "$nano_syntax_dir"; then
            sudo cp -r ${nano_syntax_dir}/*.nanorc /usr/share/nano/
            rm -rf "$nano_syntax_dir"
            echo -e "${GREEN}SUCCESS:${NC} Enhanced Nano syntax installed."
        else
            echo -e "${RED}ERROR:${NC} Failed to download enhanced Nano syntax files."
        fi
    )
}
configure_nano() {
    log_step "12/X" "Configuring Nano Editor"
    echo -e "${BLUE}INFO:${NC} Applying system-wide Nano configuration..."
    cat > /etc/nanorc <<EOF
## Nano Editor Default Configuration
set linenumbers
set softwrap
set tabsize 4
set casesensitive
set constantshow # Always show line/col info

## Include all standard syntax definitions
include "/usr/share/nano/*.nanorc"
EOF
    install_nano_syntax
    echo -e "${GREEN}SUCCESS:${NC} Nano configured with defaults and syntax highlighting."
}
configure_vim() {
    log_step "12.1/X" "Configuring Vim/Vi"
```

```
    echo -e "${BLUE}INFO:${NC} Applying system-wide Vim configuration..."
    cat > /etc/vimrc <<EOF
" System-wide .vimrc file
syntax on
set background=dark
set number
set ruler
set showcmd
set incsearch
set wildmenu
EOF
    echo -e "${GREEN}SUCCESS:${NC} Vim configured with syntax highlighting."
}
enhance_bash() {
    log_step "13/X" "Enhancing Bash Experience (Optional, Ctrl+C to skip)"
    ( # Start subshell for cancellation
        trap 'echo -e "\n${YELLOW}Operation cancelled. Skipping Bash
enhancements...${NC}"; exit 0;' INT

        read -rp "$(echo -e "${CYAN}Enhance the Bash shell with a better prompt and
aliases? [y/N]: ${NC}")" choice < /dev/tty
        if [[ "$(echo "$choice" | tr '[:upper:]' '[:lower:]')" != "y" ]]; then
            echo -e "${BLUE}INFO:${NC} Skipping Bash enhancements."; exit 0;
        fi
        echo -e "${BLUE}INFO:${NC} Creating /etc/profile.d/enhanced_bash.sh..."
        cat > /etc/profile.d/enhanced_bash.sh <<'EOF'
# Custom Bash prompt
PS1='\[\e[32m\]\u@\h \[\e[33m\]\w\[\e[0m\]\n\$ '
# Useful Aliases
alias ls='ls --color=auto'
alias ll='ls -alF'
alias la='ls -A'
alias l='ls -CF'
alias grep='grep --color=auto'
alias ..='cd ..'
EOF
        echo -e "${GREEN}SUCCESS:${NC} Bash enhancements will be applied on next login."
    )
}
```

```
setup_tmux() {
    log_step "14/X" "Setting up Automated Tmux Environment (Optional, Ctrl+C to skip)"
    ( # Start subshell for cancellation
        trap 'echo -e "\n${YELLOW}Operation cancelled. Skipping Tmux setup...${NC}"; exit
0;' INT

        read -rp "$(echo -e "${CYAN}Set up a default Tmux configuration? [y/N]: ${NC}")"
choice < /dev/tty
        if [[ "$(echo "$choice" | tr '[:upper:]' '[:lower:]')" != "y" ]]; then
            echo -e "${BLUE}INFO:${NC} Skipping Tmux setup."; exit 0;
        fi
        echo -e "${BLUE}INFO:${NC} Creating system-wide /etc/tmux.conf..."
        cat > /etc/tmux.conf <<'EOF'
# Set prefix to Ctrl-a
set -g prefix C-a
unbind C-b
bind C-a send-prefix
# Enable mouse mode
set -g mouse on
# Improve status bar
set -g status-bg black
set -g status-fg white
set -g status-left '#[fg=green]#H'
set -g status-right '#[fg=yellow]%Y-%m-%d %H:%M'
EOF
        echo -e "${GREEN}SUCCESS:${NC} Default Tmux configuration created."
    )
}
setup_motd() {
    log_step "15/X" "Setting Up Dynamic MOTD (Optional, Ctrl+C to skip)"
    ( # Start subshell for cancellation
        trap 'echo -e "\n${YELLOW}Operation cancelled. Skipping MOTD setup...${NC}"; exit
0;' INT

        if [ -f /etc/profile.d/99-custom-motd.sh ]; then
            read -rp "$(echo -e "${YELLOW}WARNING:${NC} Custom MOTD script already exists.
Action? ([S]kip, [O]verwrite): ${NC}")" choice < /dev/tty
            if [[ "$(echo "$choice" | tr '[:upper:]' '[:lower:]')" != "o" ]]; then
                echo -e "${BLUE}INFO:${NC} Skipping MOTD setup."; exit 0;
```

```
            fi
        else
            read -rp "$(echo -e "${CYAN}Setup a dynamic MOTD (Message of the Day)? [y/N]:
${NC}")" choice < /dev/tty
            if [[ "$(echo "$choice" | tr '[:upper:]' '[:lower:]')" != "y" ]]; then
                echo -e "${BLUE}INFO:${NC} Skipping MOTD setup."; exit 0;
            fi
        fi

        echo -e "${BLUE}INFO:${NC} Creating a dynamic message of the day..."
        chmod -x /etc/update-motd.d/* &>/dev/null || true
        sed -i '/session\s\+optional\s\+pam_motd.so/s/^/#/' /etc/pam.d/sshd 2>/dev/null ||
true
        cat > /etc/profile.d/99-custom-motd.sh <<'EOF'
# This script runs for interactive shells to display a dynamic MOTD.
if [[ $- == *i* ]] && [[ "${SHLVL:-1}" -le 1 ]]; then
    RED='\033[0;31m'; GREEN='\033[0;32m'; YELLOW='\033[0;33m'; BLUE='\033[0;34m';
    PURPLE='\033[0;35m'; CYAN='\033[0;36m'; NC='\033[0m';
    echo -e "\nWelcome to ${GREEN}$(hostname -f)${NC}"
    echo -e "System time is: ${CYAN}$(date --iso-8601=seconds)${NC}\n"
    # Display OS Info
    if [ -f /etc/os-release ]; then
        OS_INFO=$(grep PRETTY_NAME /etc/os-release | cut -d'"' -f2)
        echo -e "${PURPLE}System Information${NC}"
        echo -e "  OS: ${YELLOW}${OS_INFO}${NC}"
        echo -e "  Uptime: $(uptime -p | sed 's/up //')\n"
    fi
    # Display Network Information
    echo -e "${PURPLE}Network Information${NC}"
    ip -4 addr | grep -oP '(?<=inet\s)\d+(\.\d+){3}/\d+\s.*\s\K\w+$' | while read -r dev;
    do
        ip_addr=$(ip -4 addr show dev "$dev" | grep -oP '(?<=inet\s)\d+(\.\d+){3}')
        if [[ -n "$ip_addr" ]]; then echo -e "  Interface ${GREEN}$dev${NC}:
${CYAN}$ip_addr${NC}"; fi
    done || echo -e "  ${RED}No active IPv4 interfaces found.${NC}"
    echo
    # Display System Usage
    echo -e "${PURPLE}System Usage${NC}"
    df -h / | awk '$NR==2 {print "  Disk (/): " $2 " total, " $3 " used (" $5 " full), "
```

```
$4 " free"}'
    free -h | awk '/^Mem:/ {print "  Memory:    " $2 " total, " $3 " used, " $7 "
available"}'
    echo -e "  CPU Load: $(uptime | awk -F'load average:' '{ print $2}' | sed 's/ //g')\n"
    # Display Installed Software Versions
    echo -e "${PURPLE}Installed Software${NC}"
    declare -A progs=(
        ["Docker"]="docker --version" ["Podman"]="podman --version"
        ["Nginx"]="nginx -v" ["Apache"]="httpd -v" ["PHP"]="php -v"
        ["Node"]="node -v" ["NPM"]="npm -v" ["Go"]="go version"
        ["Java"]="java -version" ["MySQL"]="mysql --version" ["PostgreSQL"]="psql --
version"
    )
    output=""
    for name in "${!progs[@]}"; do
        if command -v ${progs[$name]%% *} &>/dev/null; then
            version=$(${progs[$name]} 2>&1 | grep -oP '(\d+\.){1,}\d+' | head -n1)
            [[ -n "$version" ]] && output+="  ${GREEN}${name}${NC}:${CYAN}${version}${NC}"
        fi
    done
    echo -e "${output:-  None detected}\n"
fi
EOF
        chmod +x /etc/profile.d/99-custom-motd.sh
        echo -e "${GREEN}SUCCESS:${NC} Dynamic MOTD script created."
    )
}
install_nerd_fonts() {
    log_step "16.1/X" "Installing Nerd Fonts (Sub-step)"
    ( # Start subshell for cancellation
        trap 'echo -e "\n${YELLOW}Operation cancelled. Skipping Nerd Fonts...${NC}"; exit
124;' INT

        echo -e "${BLUE}INFO:${NC} Checking for Nerd Fonts..."
        local font_dir="/usr/local/share/fonts/FiraCodeNerdFont"
        if [ -d "$font_dir" ]; then
            echo -e "${BLUE}INFO:${NC} Nerd Font directory already exists. Skipping
download."
            exit 0
```

```
        fi

        read -rp "$(echo -e "${CYAN}Install FiraCode Nerd Font for Zsh themes? [y/N]:
${NC}")" choice < /dev/tty
        if [[ "$(echo "$choice" | tr '[:upper:]' '[:lower:]')" != "y" ]]; then
            echo -e "${BLUE}INFO:${NC} Skipping Nerd Font installation."; exit 0;
        fi
        mkdir -p "$font_dir"
        local tmp_zip="/tmp/FiraCode.zip"

        local retries=3; local count=0; local success=false
        until [ $count -ge $retries ]
        do
            echo -e "${BLUE}INFO:${NC} Attempting to download FiraCode Nerd Font (attempt
$((count+1))/${retries})..."
            ping -c 1 google.com &>/dev/null || true
            sleep 1
            curl --connect-timeout 20 -L "https://github.com/ryanoasis/nerd-
fonts/releases/download/v3.2.1/FiraCode.zip" -o "$tmp_zip"
            if [ $? -eq 0 ]; then success=true; break; fi
            count=$((count+1))
            echo -e "${YELLOW}WARNING:${NC} Download failed. Retrying in 5 seconds..."
            sleep 5
        done

        if ! $success; then
            echo -e "${RED}ERROR:${NC} Failed to download Nerd Fonts after $retries
attempts." >&2; rm -f "$tmp_zip"; exit 1;
        fi

        unzip -o "$tmp_zip" -d "$font_dir"; rm -f "$tmp_zip"
        echo -e "${BLUE}INFO:${NC} Rebuilding font cache..."; fc-cache -fv &>/dev/null
        echo -e "${GREEN}SUCCESS:${NC} Nerd Fonts installed."
    )
    return $?
}
configure_starship() {
    local user=$1
    local home_dir
```

```
    home_dir=$(eval echo ~$user)
    local config_dir="${home_dir}/.config"
    local starship_config="${config_dir}/starship.toml"

    echo -e "${BLUE}INFO:${NC} Creating Starship 'Powerline' config for ${user}..."
    sudo -u "$user" mkdir -p "$config_dir"
    sudo -u "$user" tee "$starship_config" > /dev/null <<'EOF'
# Starship "Powerline" configuration
# Shows: [USER@HOST] [DATE TIME] [DIRECTORY] [GIT] [CMD_DURATION]
# >>>

# A minimal left prompt
format = """$username$hostname$time$directory$git_branch$cmd_duration$character"""

# Move the directory to the second line
# format = """$username$hostname$time$directory$git_branch$cmd_duration$fill$character"""

[username]
style_user = "yellow bold"
style_root = "red bold"
format = "[$user]($style_user)@"
show_always = true

[hostname]
style = "green bold"
format = "[$hostname]($style) "
ssh_only = false
disabled = false

[time]
disabled = false
format = '[\[$time\]]($style) '
style = "blue bold"
time_format = "%Y-%m-%d %H:%M:%S"

[directory]
style = "cyan bold"
format = "[$path]($style) "
truncation_length = 4
```

```
[git_branch]
style = "bold purple"
format = "[$branch]($style) "

[cmd_duration]
min_time = 500
style = "bold italic yellow"
format = "[$duration]($style) "

[character]
success_symbol = "[>](bold green)"
error_symbol = "[x](bold red)"
EOF
}
install_zsh_omz() {
    log_step "16/X" "Installing Zsh & Oh My Zsh (Optional, Ctrl+C to skip)"
    ( # Start subshell for cancellation
        trap 'echo -e "\n${YELLOW}Operation cancelled. Skipping Zsh setup...${NC}"; exit
124;' INT

        local choice
        if command -v zsh &>/dev/null; then
            read -rp "$(echo -e "${CYAN}Zsh is already installed. Action? ([S]kip,
[R]econfigure): ${NC}")" choice < /dev/tty
            if [[ "$(echo "$choice" | tr '[:upper:]' '[:lower:]')" == "s" ]]; then
                echo -e "${BLUE}INFO:${NC} Skipping Zsh setup."; exit 0;
            fi
        else
            read -rp "$(echo -e "${CYAN}Install Zsh and Oh My Zsh? [y/N]: ${NC}")" choice
< /dev/tty
            if [[ "$(echo "$choice" | tr '[:upper:]' '[:lower:]')" != "y" ]]; then
                echo -e "${BLUE}INFO:${NC} Skipping Zsh installation."; exit 0;
            fi
        fi
        $PKG_MANAGER -y install zsh git

        install_nerd_fonts
        if [ $? -ne 0 ]; then
```

```
            echo -e "${YELLOW}WARNING:${NC} Nerd font installation failed or was skipped.
Zsh themes may not render correctly."
        fi

        local users_to_configure=()
        users_to_configure+=("root")
        if [[ -n "${SUDO_USER:-}" ]] && [[ "$SUDO_USER" != "root" ]]; then
            users_to_configure+=("$SUDO_USER")
        fi
        local prompt_choice
        read -rp "$(echo -e "${CYAN}Which Zsh prompt? ([1] Oh My Zsh (rkj-repos), [2]
Starship, [3] Powerlevel10k): ${NC}")" prompt_choice < /dev/tty
        for user in "${users_to_configure[@]}"; do
            echo -e "${BLUE}INFO:${NC} Configuring Zsh for user ${PURPLE}${user}${NC}..."
            local home_dir; home_dir=$(eval echo ~$user)
            local zsh_dir="${home_dir}/.oh-my-zsh"

            if [ ! -d "$zsh_dir" ]; then
                local installer_sh="/tmp/omz_install.sh"
                local retries=3; local count=0; local success=false
                echo -e "${BLUE}INFO:${NC} Downloading Oh My Zsh installer..."
                until [ $count -ge $retries ]; do
                    curl -fsSL
https://raw.githubusercontent.com/ohmyzsh/ohmyzsh/master/tools/install.sh -o
"$installer_sh"
                    if [ $? -eq 0 ]; then success=true; break; fi
                    count=$((count+1)); echo -e "${YELLOW}WARNING:${NC} Download failed.
Retrying..."; sleep 3
                done

                if $success; then
                    echo -e "${BLUE}INFO:${NC} Running Oh My Zsh installer for ${user}..."
                    sudo -u "$user" sh "$installer_sh" --unattended
                    rm "$installer_sh"
                else
                    echo -e "${RED}ERROR:${NC} Failed to download Oh My Zsh installer for
${user}." >&2; continue
                fi
            fi
```

```
            local custom_plugins_dir="${zsh_dir}/custom/plugins"
            if [ ! -d "${custom_plugins_dir}/zsh-autosuggestions" ]; then
                sudo -u "$user" git clone https://github.com/zsh-users/zsh-autosuggestions
"$custom_plugins_dir/zsh-autosuggestions"
            fi
            if [ ! -d "${custom_plugins_dir}/zsh-syntax-highlighting" ]; then
                sudo -u "$user" git clone https://github.com/zsh-users/zsh-syntax-
highlighting.git "$custom_plugins_dir/zsh-syntax-highlighting"
            fi
            local zshrc_file="${home_dir}/.zshrc"

            # FIX: Robustly set plugins
            sed -i 's/^plugins=(.*)$/plugins=(git docker npm nvm zsh-autosuggestions zsh-
syntax-highlighting)/' "$zshrc_file"

            # FIX: Clean up old theme settings before applying a new one
            sed -i '/^# Init Starship Prompt/d' "$zshrc_file"
            sed -i '/eval "$(starship init zsh)"/d' "$zshrc_file"

            case "$prompt_choice" in
                2) # Starship
                    if ! command -v starship &>/dev/null; then
                        local installer_sh="/tmp/starship_install.sh"
                        echo -e "${BLUE}INFO:${NC} Downloading Starship installer..."
                        if curl -sS https://starship.rs/install.sh -o "$installer_sh";
then
                            sh "$installer_sh" -y
                            rm "$installer_sh"
                        else
                             echo -e "${RED}ERROR:${NC} Failed to download starship
installer." >&2
                        fi
                    fi
                    echo -e '\n# Init Starship Prompt\neval "$(starship init zsh)"' >>
"$zshrc_file"
                    configure_starship "$user"
                    ;;
                3) # Powerlevel10k
```

```
                    local p10k_dir="${zsh_dir}/custom/themes/powerlevel10k"
                    if [ ! -d "$p10k_dir" ]; then
                        echo -e "${BLUE}INFO:${NC} Cloning Powerlevel10k theme..."
                        sudo -u "$user" git clone --depth=1
https://github.com/romkatv/powerlevel10k.git "$p10k_dir"
                    fi
                    echo -e "${BLUE}INFO:${NC} Setting Powerlevel10k theme in .zshrc..."
                    sed -i 's|^ZSH_THEME=.*|ZSH_THEME="powerlevel10k/powerlevel10k"|'
"$zshrc_file"
                    if ! grep -q 'POWERLEVEL9K_DISABLE_CONFIGURATION_WIZARD=true'
"$zshrc_file"; then
                         echo -e '\n# To customize prompt, run `p10k configure` or edit
~/.p10k.zsh.\n[[ ! -f ~/.p10k.zsh ]] && p10k configure' >> "$zshrc_file"
                    fi
                    ;;
                *) # Default to rkj-repos
                    echo -e "${BLUE}INFO:${NC} Setting ZSH_THEME to rkj-repos..."
                    sed -i 's|^ZSH_THEME=.*|ZSH_THEME="rkj-repos"|' "$zshrc_file"
                    ;;
            esac

            if ! grep -q 'setopt EXTENDED_HISTORY' "$zshrc_file"; then
                echo -e '\n# Custom settings by setup script\nsetopt
EXTENDED_HISTORY\nHIST_STAMPS="yyyy-mm-dd"\n' >> "$zshrc_file"
            fi

            if ! grep -q "alias ls='ls --color=auto'" "$zshrc_file"; then
                echo -e '\n# Color Aliases\nalias ls="ls --color=auto"\nalias grep="grep
--color=auto"' >> "$zshrc_file"
            fi

            local current_shell; current_shell=$(getent passwd "$user" | cut -d: -f7)
            if [[ "$current_shell" != "$(which zsh)" ]]; then
                echo -e "${BLUE}INFO:${NC} Changing shell for user ${user} to Zsh..."
                chsh -s "$(which zsh)" "$user"
                echo -e "${GREEN}SUCCESS:${NC} Shell changed."
            else
                echo -e "${BLUE}INFO:${NC} Shell for user ${user} is already zsh."
            fi
```

```
        done
    )
}
#
#SECTION 5: APPLICATION STACK FUNCTIONS
#
install_dev_stack() {
    log_step "18/X" "Installing Development Stack (Optional, Ctrl+C to skip)"
    ( # Start subshell for cancellation
        trap 'echo -e "\n${YELLOW}Operation cancelled. Skipping Dev Stack...${NC}"; exit
0;' INT

        read -rp "$(echo -e "${CYAN}Install a development stack (Nginx, PHP, Node.js)?
[y/N]: ${NC}")" choice < /dev/tty
        if [[ "$(echo "$choice" | tr '[:upper:]' '[:lower:]')" != "y" ]]; then
            echo -e "${BLUE}INFO:${NC} Skipping dev stack installation."; exit 0;
        fi
        echo -e "${BLUE}INFO:${NC} Installing Nginx..."
        $PKG_MANAGER -y install nginx
        systemctl enable nginx; systemctl start nginx
        echo -e "${BLUE}INFO:${NC} Installing Node.js (LTS)..."
        curl -fsSL https://rpm.nodesource.com/setup_lts.x | bash -
        $PKG_MANAGER -y install nodejs
        echo -e "${BLUE}INFO:${NC} Installing PHP..."
        if [[ "$PKG_MANAGER" == "dnf" || "$PKG_MANAGER" == "yum" ]]; then
            $PKG_MANAGER -y install http://rpms.remirepo.net/enterprise/remi-release-8.rpm
            $PKG_MANAGER -y module reset php; $PKG_MANAGER -y module install php:remi-8.1
            $PKG_MANAGER -y install php-cli php-fpm php-mysqlnd php-json php-gd php-
mbstring
        else # APT
            add-apt-repository ppa:ondrej/php -y; $PKG_MANAGER update
            $PKG_MANAGER -y install php8.1-cli php8.1-fpm php8.1-mysql php8.1-json php8.1-
gd php8.1-mbstring
        fi
        echo -e "${GREEN}SUCCESS:${NC} Development stack installed."
    )
}
install_cockpit() {
    log_step "19/X" "Installing Cockpit Web Console (Optional, Ctrl+C to skip)"
```

```
    ( # Start subshell for cancellation
        trap 'echo -e "\n${YELLOW}Operation cancelled. Skipping Cockpit...${NC}"; exit 0;'
INT

        read -rp "$(echo -e "${CYAN}Install the Cockpit web administration console? [y/N]:
${NC}")" choice < /dev/tty
        if [[ "$(echo "$choice" | tr '[:upper:]' '[:lower:]')" != "y" ]]; then
            echo -e "${BLUE}INFO:${NC} Skipping Cockpit installation."; exit 0;
        fi
        echo -e "${BLUE}INFO:${NC} Installing Cockpit..."
        $PKG_MANAGER -y install cockpit cockpit-storaged cockpit-networkmanager
        echo -e "${BLUE}INFO:${NC} Starting and enabling Cockpit socket..."
        systemctl enable --now cockpit.socket
        echo -e "${GREEN}SUCCESS:${NC} Cockpit is installed. Access it at
https://$(hostname -f):9090"
    )
}
install_container_runtime() {
    log_step "20/X" "Installing Container Runtime (Optional, Ctrl+C to skip)"
    ( # Start subshell for cancellation
        trap 'echo -e "\n${YELLOW}Operation cancelled. Skipping Container
Runtime...${NC}"; exit 0;' INT

        read -rp "$(echo -e "${CYAN}Install a container runtime? ([D]ocker, [P]odman,
[N]one): ${NC}")" choice < /dev/tty
        case "$(echo "$choice" | tr '[:upper:]' '[:lower:]')" in
            d|docker)
                echo -e "${BLUE}INFO:${NC} Installing Docker..."
                if [[ "$PKG_MANAGER" == "dnf" || "$PKG_MANAGER" == "yum" ]]; then
                    $PKG_MANAGER config-manager --add-repo
https://download.docker.com/linux/centos/docker-ce.repo
                    $PKG_MANAGER -y install docker-ce docker-ce-cli containerd.io
                else # APT
                    install -m 0755 -d /etc/apt/keyrings
                    curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o
/etc/apt/keyrings/docker.asc
                    chmod a+r /etc/apt/keyrings/docker.asc
                    echo "deb [arch=$(dpkg --print-architecture) signed-
by=/etc/apt/keyrings/docker.asc] https://download.docker.com/linux/ubuntu $(. /etc/os-
```

```
release && echo "$VERSION_CODENAME") stable" > /etc/apt/sources.list.d/docker.list
                    $PKG_MANAGER update
                    $PKG_MANAGER -y install docker-ce docker-ce-cli containerd.io
                fi
                systemctl enable --now docker
                if [[ -n "${SUDO_USER:-}" ]]; then
                    echo -e "${BLUE}INFO:${NC} Adding user ${SUDO_USER} to the docker
group..."
                    usermod -aG docker "$SUDO_USER"
                fi
                echo -e "${GREEN}SUCCESS:${NC} Docker installed."
                ;;
            p|podman)
                echo -e "${BLUE}INFO:${NC} Installing Podman..."
                $PKG_MANAGER -y install podman
                echo -e "${GREEN}SUCCESS:${NC} Podman installed."
                ;;
            *)
                echo -e "${BLUE}INFO:${NC} Skipping container runtime installation.";;
        esac
    )
}
#
#SECTION 6: UTILITY & FINALIZATION FUNCTIONS
#
setup_logrotate() {
    log_step "21/X" "Setting up Logrotate (Optional, Ctrl+C to skip)"
    ( # Start subshell for cancellation
        trap 'echo -e "\n${YELLOW}Operation cancelled. Skipping Logrotate setup...${NC}";
exit 0;' INT

        read -rp "$(echo -e "${CYAN}Configure log rotation for this script's log files?
[y/N]: ${NC}")" choice < /dev/tty
        if [[ "$(echo "$choice" | tr '[:upper:]' '[:lower:]')" != "y" ]]; then
            echo -e "${BLUE}INFO:${NC} Skipping logrotate setup."; exit 0;
        fi
        echo -e "${BLUE}INFO:${NC} Creating /etc/logrotate.d/setup-domain..."
        cat > /etc/logrotate.d/setup-domain <<EOF
/var/log/setup-domain-*.log {
```

```
    monthly
    rotate 4
    compress
    delaycompress
    missingok
    notifempty
    create 640 root root
}
EOF
        echo -e "${GREEN}SUCCESS:${NC} Logrotate configured."
    )
}
final_summary() {
    log_step "22/X" "Final Setup Summary"
    echo -e "${GREEN}================== SUMMARY ==================${NC}"
    echo -e "  Hostname: ${PURPLE}$(hostname -f)${NC}"
    echo -e "  IP Address: ${CYAN}$(hostname -I | awk '{print $1}')${NC}"
    echo -e "  Timezone: ${CYAN}${TIMEZONE}${NC}"
    echo -e "  Domain Membership: ${PURPLE}${DOMAIN_FQDN}${NC}"
    realm list | grep "configured: yes" &>/dev/null
    if [ $? -eq 0 ]; then
        echo -e "  Domain Join Status: ${GREEN}Success${NC}"
        echo -e "  Login with AD users as: ${CYAN}username${NC}"
        echo -e "  Sudo enabled for group: ${PURPLE}${AD_SUDO_GROUP_RAW_NAME}${NC}"
    else
        echo -e "  Domain Join Status: ${RED}Failed or Not Performed${NC}"
    fi
    echo -e "  Log File: ${YELLOW}${LOG_FILE}${NC}"
    echo -e "${GREEN}=============================================${NC}"
}
system_updates_interactive() {
    log_step "23/X" "System Updates (Optional, Ctrl+C to skip)"
    ( # Start subshell for cancellation
        trap 'echo -e "\n${YELLOW}Operation cancelled. Skipping System Updates...${NC}";
exit 0;' INT

        read -rp "$(echo -e "${CYAN}Check for and apply all available system updates?
[y/N]: ${NC}")" choice < /dev/tty
        if [[ "$(echo "$choice" | tr '[:upper:]' '[:lower:]')" != "y" ]]; then
```

```
            echo -e "${BLUE}INFO:${NC} Skipping system updates."; exit 0;
        fi
        echo -e "${BLUE}INFO:${NC} Checking for updates..."
        $PKG_MANAGER -y update
        echo -e "${GREEN}SUCCESS:${NC} System is up-to-date."

        echo -e "${BLUE}INFO:${NC} Checking if a reboot is required..."
        if [[ "$PKG_MANAGER" == "dnf" || "$PKG_MANAGER" == "yum" ]]; then
            if needs-restarting -r &>/dev/null; then
                # Using an external file to communicate back to the main script
                echo "true" > /tmp/reboot_required.flag
            fi
        elif [[ "$PKG_MANAGER" == "apt" ]]; then
            if [ -f /var/run/reboot-required ]; then
                echo "true" > /tmp/reboot_required.flag
            fi
        fi

        if [ -f /tmp/reboot_required.flag ]; then
            echo -e
"\n${YELLOW}################################################################################"
            echo -e "# WARNING: System updates have been installed that require a reboot."
            echo -e
"################################################################################${NC}"
        else
            echo -e "${GREEN}INFO:${NC} No reboot is required at this time."
        fi
    )
}
cleanup() {
    unset AD_PASSWORD
    rm -f /tmp/reboot_required.flag
    echo -e "${BLUE}INFO:${NC} Sensitive variables cleared from memory."
}
usage() {
    echo -e "${CYAN}Usage: $0 [OPTION]${NC}"
    echo "  --full      Run the complete end-to-end installation and configuration."
    echo "  --dev       Install the development stack (PHP, Node, Nginx, etc.)."
    echo "  --security  Apply security hardening (Fail2ban, SSH optimization)."
```

```
    echo "  --shell     Configure user experience (Bash, Zsh, Tmux, MOTD)."
    echo "  --updates   Check for and apply system updates."
    echo "  --help      Display this help message."
    echo
    echo -e "${YELLOW}If no option is provided, the script will run the full installation
interactively.${NC}"
}
#---
# MODULAR EXECUTION RUNNERS
#
run_full_install() {
    echo -e "${PURPLE}Starting Full System Setup...${NC}"
    gather_credentials
    # Core System & Network (Not cancellable - these are critical)
    change_hostname
    configure_proxy
    configure_dns_and_hosts
    check_connectivity
    install_packages
    configure_time
    # Domain & Auth (Not cancellable - these are critical)
    join_ad_domain
    configure_sssd_mkhomedir
    configure_sudoers
    # Security (Optional sections are now cancellable)
    optimize_sshd
    install_fail2ban
    #Shell & UX
    configure_nano
    configure_vim
    enhance_bash
    setup_tmux
    setup_motd
    install_zsh_omz
    # App Stacks
    install_dev_stack
    install_cockpit
    install_container_runtime
    # Utilities & Finalization
```

```
    setup_logrotate
    system_updates_interactive
    final_summary
}
run_dev_stack() {
    echo -e "${PURPLE}Starting Development Stack Setup...${NC}"
    install_packages
    install_dev_stack
    final_summary
}
run_security_hardening() {
    echo -e "${PURPLE}Starting Security Hardening...${NC}"
    install_packages
    optimize_sshd
    install_fail2ban
    final_summary
}
run_shell_ux() {
    echo -e "${PURPLE}Starting Shell & UX Setup...${NC}"
    install_packages
    configure_nano
    configure_vim
    enhance_bash
    setup_tmux
    setup_motd
    install_zsh_omz
    final_summary
}
#---
# MAIN EXECUTION FLOW
#---
main() {
    trap cleanup EXIT
    if [ $# -eq 0 ]; then
        run_full_install
    else
        case "$1" in
            --full) run_full_install ;;
            --dev) run_dev_stack ;;
```

```
            --security) run_security_hardening ;;
            --shell) run_shell_ux ;;
            --updates) system_updates_interactive ;;
            --help) usage ;;
            *)
                echo -e "${RED}Error: Invalid option '$1'${NC}" >&2
                usage
                exit 1
                ;;
        esac
    fi

    echo -e "\n${GREEN}========= Script finished at $(date --iso-8601=seconds)
=========${NC}"

    if [ -f /tmp/reboot_required.flag ]; then
        REBOOT_REQUIRED_FLAG=true
    fi

    if $REBOOT_REQUIRED_FLAG; then
        read -rp "$(echo -e "${YELLOW}A reboot is required to apply updates. Reboot now?
[y/N]: ${NC}")" choice < /dev/tty
        if [[ "$(echo "$choice" | tr '[:upper:]' '[:lower:]')" == "y" ]]; then
            echo -e "${RED}Rebooting now...${NC}"
            reboot
        else
            echo -e "${YELLOW}Please reboot the server manually to apply all
changes.${NC}"
        fi
    else
         echo -e "${GREEN}Script complete. No reboot required.${NC}"
    fi
}
# Only run main if the script is executed directly
if [[ "${BASH_SOURCE[0]}" == "${0}" ]]; then
    main "$@"
fi
```

master_script.sh - v42

```
#!/usr/bin/env bash
#
# MASTER INFRASTRUCTURE SETUP v42
# Fixes: Firewalld Defense-in-Depth, Docker JSON Parsing, SSSD Bulletproofing
#
################################################################################
# 1. CONFIGURATION
################################################################################
DOMAIN_FQDN="m21.gov.local"
DOMAIN_ALT="m21.gov.tt"
DOMAIN_SHORT="M21"
DC_DNS_IP="172.16.21.161"
NTP_SERVER="172.16.121.9"
TARGET_TIMEZONE="America/Port_of_Spain"

# File Server Info
FILE_SERVER_IP="172.16.21.16"
FILE_SERVER_NAME="fileserver2"

# Proxy
PROXY_URL="http://172.40.4.14:8080"
NO_PROXY_LIST="127.0.0.1,localhost,localhost.localdomain,${DOMAIN_FQDN},${DOMAIN_ALT},.${D
OMAIN_FQDN},.${DOMAIN_ALT},${DC_DNS_IP},172.30.0.0/20,172.26.21.0/24,10.21.0.0/21,172.16.1
21.0/24"

# Docker Settings
INSECURE_REGISTRIES='"172.16.121.119:5000", "docker-repo.msya.gov.tt"'

# AD Access Control
AD_SUDO_GROUP="ICT Staff SG M21"
ALLOWED_LOGIN_GROUP="ICT Staff SG M21"

# Share Credentials
SHARE_PATH="//172.16.21.16/fileserver2"
SHARE_USER="Cipher.m21"
SHARE_PASS=")\ly; 634'NJ%i+"
CERT_SOURCE_PATH="/General/IT FILES/prx/Gortt_certificate_V4.cer"
```

```
TARGET_CERT_NAME="GORTT_Root_Exp2029"

# Failsafe User
LOCAL_USER="pcsupport"
LOCAL_PASS="ProIT321*"

# LVM Settings
HOME_TARGET_SIZE="8G"

# Versions
PHP_VERSION="8.3"
JAVA_VERSION="21"
MARIADB_VERSION="10.11"

################################################################################
# 2. HELPER FUNCTIONS
################################################################################
set -e
RED='\033[0;31m'; GREEN='\033[0;32m'; YELLOW='\033[0;33m'; BLUE='\033[0;34m'; NC='\033[0m'

log() { echo -e "${BLUE}[$(date +'%H:%M:%S')] [INFO]${NC} $1"; }
step() { echo -e "\n${YELLOW}[$(date +'%H:%M:%S')] >>> $1${NC}"; }
success() { echo -e "${GREEN}[$(date +'%H:%M:%S')] [OK]${NC} $1"; }
error() { echo -e "${RED}[$(date +'%H:%M:%S')] [ERROR]${NC} $1"; }

run_retry() {
    local n=1; local max=3; local delay=2
    while true; do
        "$@" && return 0
        if [[ $n -lt $max ]]; then
            ((n++)); log "Command failed. Retrying ($n/$max)..."; sleep $delay
        else
            return 1
        fi
    done
}

################################################################################
# 3. PRE-FLIGHT CHECKS
```

```
################################################################################
detect_and_fix_os() {
    source /etc/os-release
    OS_ID=$(echo "$ID" | tr '[:upper:]' '[:lower:]')
    VERSION_MAJOR=$(echo "$VERSION_ID" | cut -d. -f1)

    if timeout 10s systemctl is-active --quiet packagekit.service 2>/dev/null; then
        timeout 15s systemctl stop packagekit.service || true
    fi

    if [[ "$OS_ID" == "centos" && "$VERSION_MAJOR" == "7" ]]; then
        PKG="yum"
        if grep -q "linux/rhel" /etc/yum.repos.d/docker-ce.repo 2>/dev/null; then rm -f
/etc/yum.repos.d/docker-ce.repo; fi
        if [ ! -f /etc/yum.repos.d/CentOS-Base.repo.backup ]; then
            cp /etc/yum.repos.d/CentOS-Base.repo /etc/yum.repos.d/CentOS-Base.repo.backup
2>/dev/null || true
            run_retry curl -o /etc/yum.repos.d/CentOS-Base.repo
https://el7.repo.almalinux.org/centos/CentOS-Base.repo
        fi
    elif [[ "$OS_ID" =~ (rhel|centos|almalinux|rocky) ]]; then PKG="dnf"
    elif [[ "$OS_ID" =~ (ubuntu|debian) ]]; then PKG="apt-get"
    else error "Unsupported OS: $OS_ID"; exit 1; fi
}

################################################################################
# 4. MODULES
################################################################################

mod_proxy() {
    step "Configuring System Proxy"
    cat > /etc/profile.d/proxy.sh <<EOF
export http_proxy="${PROXY_URL}"
export https_proxy="${PROXY_URL}"
export ftp_proxy="${PROXY_URL}"
export no_proxy="${NO_PROXY_LIST}"
export HTTP_PROXY="${PROXY_URL}"
export HTTPS_PROXY="${PROXY_URL}"
export FTP_PROXY="${PROXY_URL}"
```

```
export NO_PROXY="${NO_PROXY_LIST}"
EOF
    source /etc/profile.d/proxy.sh

    if [[ "$PKG" == "dnf" || "$PKG" == "yum" ]]; then
        CONF_FILE="/etc/dnf/dnf.conf"
        [[ ! -f "$CONF_FILE" ]] && CONF_FILE="/etc/yum.conf"
        grep -q "proxy=" "$CONF_FILE" 2>/dev/null || echo "proxy=${PROXY_URL}" >>
"$CONF_FILE"
        if ! grep -q "minrate" "$CONF_FILE" 2>/dev/null; then
            echo -e "timeout=60\nretries=10\nminrate=1" >> "$CONF_FILE"
        fi
    else
        echo -e "Acquire::http::Proxy \"${PROXY_URL}\";\nAcquire::https::Proxy
\"${PROXY_URL}\";" > /etc/apt/apt.conf.d/80proxy
    fi
}

mod_clock_fix() {
    step "Synchronizing System Clock"
    timedatectl set-timezone "$TARGET_TIMEZONE" || true
    timedatectl set-ntp true || true
    if systemctl list-unit-files | grep -q systemd-timesyncd; then
        timeout 30s systemctl restart systemd-timesyncd || true
    fi
}

mod_certs() {
    step "Installing Certificates"
    MNT="/mnt/share_certs_tmp"
    mkdir -p "$MNT"

    if ! command -v mount.cifs &>/dev/null; then
        if [[ "$PKG" == "apt-get" ]]; then run_retry apt-get update -qq >/dev/null 2>&1 ||
true; run_retry apt-get install -y cifs-utils
        else run_retry $PKG install -y cifs-utils; fi
    fi

    if mountpoint -q "$MNT"; then umount -l "$MNT"; fi
```

```
    if timeout 30s mount -t cifs "$SHARE_PATH" "$MNT" -o
username="$SHARE_USER",password="$SHARE_PASS",vers=3.0; then
        SOURCE_FULL="$MNT$CERT_SOURCE_PATH"
        TEMP_PEM="/tmp/${TARGET_CERT_NAME}_staging.pem"

        if [[ -f "$SOURCE_FULL" ]]; then
            if ! openssl x509 -inform der -in "$SOURCE_FULL" -out "$TEMP_PEM" 2>/dev/null;
then cp "$SOURCE_FULL" "$TEMP_PEM"; fi
            if [[ "$PKG" == "dnf" || "$PKG" == "yum" ]]; then
                cp "$TEMP_PEM" "/etc/pki/ca-trust/source/anchors/${TARGET_CERT_NAME}.pem"
                [[ "$VERSION_MAJOR" -lt 9 ]] && update-ca-trust force-enable 2>/dev/null
|| true
                update-ca-trust extract
            else
                cp "$TEMP_PEM" "/usr/local/share/ca-certificates/${TARGET_CERT_NAME}.crt"
                update-ca-certificates
            fi
        fi
        timeout 15s umount "$MNT" || true
    fi
    rmdir "$MNT" 2>/dev/null || true
}

mod_base_repos() {
    step "Configuring Base OS Repositories"
    if [[ "$PKG" == "dnf" || "$PKG" == "yum" ]]; then
        if ! rpm -q epel-release >/dev/null 2>&1; then run_retry $PKG install -y epel-
release; fi
        if [[ "$PKG" == "dnf" ]]; then
            if ! dnf repolist enabled 2>/dev/null | grep -E "crb|powertools" >/dev/null;
then
                run_retry $PKG install -y 'dnf-command(config-manager)'
                $PKG config-manager --set-enabled crb 2>/dev/null || $PKG config-manager
--set-enabled powertools 2>/dev/null || true
            fi
        fi
    else
        export DEBIAN_FRONTEND=noninteractive
```

```
        rm -f /etc/apt/sources.list.d/45drives.list
        apt-get update -qq || true
        run_retry apt-get install -y software-properties-common curl wget gnupg lsb-
release
    fi
}

mod_dev_repos() {
    step "Configuring Development Repositories"
    if [[ "$PKG" == "dnf" || "$PKG" == "yum" ]]; then
        if ! rpm -q remi-release >/dev/null 2>&1; then
            if [[ "$PKG" == "dnf" ]]; then run_retry $PKG install -y
"https://rpms.remirepo.net/enterprise/remi-release-${VERSION_MAJOR}.rpm"
            else run_retry $PKG install -y http://rpms.remirepo.net/enterprise/remi-
release-7.rpm yum-utils; fi
        fi
        if [[ ! -f /etc/yum.repos.d/docker-ce.repo ]]; then
            run_retry $PKG install -y yum-utils
            run_retry yum-config-manager --add-repo
https://download.docker.com/linux/centos/docker-ce.repo
        fi
        if [[ ! -f /etc/yum.repos.d/mariadb.repo ]]; then
            cat > /etc/yum.repos.d/mariadb.repo <<EOF
[mariadb]
name = MariaDB
baseurl = https://rpm.mariadb.org/${MARIADB_VERSION}/rhel/\$releasever/\$basearch
module_hotfixes=1
gpgkey=https://rpm.mariadb.org/RPM-GPG-KEY-MariaDB
gpgcheck=1
EOF
        fi
        $PKG remove -y podman buildah docker docker-client docker-common docker-engine
>/dev/null 2>&1 || true
    else
        if ! grep -q "ondrej/php" /etc/apt/sources.list.d/* 2>/dev/null; then run_retry
add-apt-repository -y ppa:ondrej/php; fi
        if [[ ! -f /etc/apt/sources.list.d/docker.list ]]; then
            install -m 0755 -d /etc/apt/keyrings
            run_retry curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o
```

```
/etc/apt/keyrings/docker.asc
            chmod a+r /etc/apt/keyrings/docker.asc
            echo "deb [arch=$(dpkg --print-architecture) signed-
by=/etc/apt/keyrings/docker.asc] https://download.docker.com/linux/ubuntu $(lsb_release -
cs) stable" > /etc/apt/sources.list.d/docker.list
        fi
        apt-get update -qq || true
    fi
}

mod_base_tools() {
    step "Installing Base System Tools"
    if [[ "$PKG" == "dnf" || "$PKG" == "yum" ]]; then
        PACKAGES="git curl wget nano unzip zip tar util-linux-user bind-utils net-tools
openssl policycoreutils-python-utils psmisc PackageKit pcp pcp-conf pcp-libs pcp-selinux"
        run_retry $PKG install -y $PACKAGES
    else
        PACKAGES="git curl wget nano unzip zip tar openssl net-tools dnsutils psmisc
packagekit pcp network-manager"
        run_retry apt-get install -y $PACKAGES
        timeout 30s systemctl enable --now NetworkManager || true
    fi
    systemctl unmask packagekit 2>/dev/null || true
    timeout 30s systemctl start packagekit 2>/dev/null || true
}

mod_network() {
    step "Configuring Network & DNS"
    if [[ "$PKG" == "apt-get" ]]; then
        if ls /etc/netplan/*.yaml >/dev/null 2>&1 && grep -q "addresses:"
/etc/netplan/*.yaml; then
            log "Static Netplan detected. Skipping wipe to prevent lockout."
        else
            mkdir -p /etc/netplan
            cat > /etc/netplan/01-network-manager-all.yaml <<EOF
network:
  version: 2
  renderer: NetworkManager
EOF
```

```
            netplan apply || true
        fi
    fi

    sed -i "/${DOMAIN_FQDN}/d; /${DOMAIN_ALT}/d; /${DC_DNS_IP}/d; /${FILE_SERVER_NAME}/d"
/etc/hosts
    cat >> /etc/hosts <<EOF
${DC_DNS_IP}    ${DOMAIN_FQDN} ${DOMAIN_ALT} ${DOMAIN_SHORT}
${FILE_SERVER_IP}    ${FILE_SERVER_NAME}.${DOMAIN_FQDN} ${FILE_SERVER_NAME}.${DOMAIN_ALT}
${FILE_SERVER_NAME}
EOF

    if [[ -L /etc/resolv.conf ]]; then rm -f /etc/resolv.conf; fi
    echo -e "search ${DOMAIN_FQDN} ${DOMAIN_ALT}\nnameserver ${DC_DNS_IP}" >
/etc/resolv.conf

    if command -v nmcli &>/dev/null; then
        TARGET_IFACE=$(ip -4 -o addr show | grep "172.16." | awk '{print $2}' | head -n1)
        if [[ -n "$TARGET_IFACE" ]]; then
            CONN=$(nmcli -t -f NAME,DEVICE con show --active | grep ":${TARGET_IFACE}" |
cut -d: -f1 | head -n1)
            if [[ -n "$CONN" ]]; then
                nmcli con mod "$CONN" ipv4.dns "$DC_DNS_IP" ipv4.dns-search
"${DOMAIN_FQDN},${DOMAIN_ALT}" ipv4.ignore-auto-dns yes
                timeout 15s nmcli con up "$CONN" >/dev/null 2>&1
            fi
        fi
    fi

    echo -e "net.ipv6.conf.all.disable_ipv6 = 1\nnet.ipv6.conf.default.disable_ipv6 = 1" >
/etc/sysctl.d/90-disable-ipv6.conf
    sysctl --system &>/dev/null || true

    if command -v systemctl &>/dev/null; then
        if [[ "$PKG" == "apt-get" ]]; then run_retry apt-get install -y chrony;
CHRONY_CONF="/etc/chrony/chrony.conf"
        else run_retry $PKG install -y chrony; CHRONY_CONF="/etc/chrony.conf"; fi

        if [[ -f "$CHRONY_CONF" ]]; then
```

```
            sed -i '/server/d; /pool/d' "$CHRONY_CONF" 2>/dev/null || true
            echo "server ${NTP_SERVER} iburst" >> "$CHRONY_CONF"
        fi
        timeout 30s systemctl restart chronyd 2>/dev/null || timeout 30s systemctl restart
chrony || true
    fi
}

mod_firewall() {
    step "Configuring Firewalld (Defense in Depth)"
    if [[ "$PKG" == "apt-get" ]]; then
        run_retry apt-get install -y firewalld
        systemctl disable ufw --now 2>/dev/null || true
    else
        run_retry $PKG install -y firewalld
    fi

    systemctl enable --now firewalld

    # Trust Docker Subnets (Fixes 502 Bad Gateway)
    firewall-cmd --permanent --zone=trusted --add-source=172.17.0.0/16
    firewall-cmd --permanent --zone=trusted --add-source=172.18.0.0/16
    firewall-cmd --permanent --zone=trusted --add-source=172.19.0.0/16
    firewall-cmd --permanent --zone=trusted --add-source=172.20.0.0/16
    firewall-cmd --permanent --zone=trusted --add-source=192.168.250.0/24

    # Web Ports
    firewall-cmd --permanent --add-service=http
    firewall-cmd --permanent --add-service=https

    # SSH Lockdown
    firewall-cmd --permanent --remove-service=ssh
    firewall-cmd --permanent --add-rich-rule='rule family="ipv4" source
address="10.21.0.0/21" service name="ssh" accept'
    firewall-cmd --permanent --add-rich-rule='rule family="ipv4" source
address="172.16.121.0/24" service name="ssh" accept'
    firewall-cmd --permanent --add-rich-rule='rule family="ipv4" source
address="172.16.21.0/24" service name="ssh" accept'
```

```
    firewall-cmd --reload
}

mod_resize_home() {
    step "LVM Home Resizer"
    if ! command -v lvs &>/dev/null; then return; fi
    if ! mountpoint -q /home; then return; fi
    HOME_DEV=$(findmnt -n -o SOURCE /home)
    if [[ "$HOME_DEV" != *"/mapper/"* ]]; then return; fi

    LV_NAME=$(lvs --noheadings -o lv_name "$HOME_DEV" | tr -d ' ')
    VG_NAME=$(lvs --noheadings -o vg_name "$HOME_DEV" | tr -d ' ')
    LV_PATH="/dev/$VG_NAME/$LV_NAME"
    ROOT_LV_PATH="/dev/$VG_NAME/root"
    MAPPER_PATH="/dev/mapper/${VG_NAME}-${LV_NAME}"

    CURRENT_SIZE=$(lvs --noheadings -o lv_size --units g "$LV_PATH" 2>/dev/null | tr -d 'g
' || lvs --noheadings -o L_SIZE --units g "$LV_PATH" | tr -d 'g ')
    if [[ ${CURRENT_SIZE%.*} -le 9 ]]; then return; fi

    tar czf /tmp/home_backup.tar.gz -C /home .
    fuser -km /home || true
    timeout 30s umount /home || timeout 15s umount -l /home || true

    lvremove -y "$LV_PATH"
    lvcreate -L "$HOME_TARGET_SIZE" -n "$LV_NAME" "$VG_NAME" -y
    mkfs.ext4 "$LV_PATH"

    sed -i '/\/home/d' /etc/fstab
    echo "$MAPPER_PATH /home ext4 defaults 0 0" >> /etc/fstab
    systemctl daemon-reload || true

    timeout 30s mount /home || true
    tar xzf /tmp/home_backup.tar.gz -C /home
    if command -v restorecon &>/dev/null; then restorecon -R /home; fi

    lvextend -l +100%FREE "$ROOT_LV_PATH"
    xfs_growfs / || resize2fs "$ROOT_LV_PATH" || true
    rm -f /tmp/home_backup.tar.gz
```

```
}

mod_domain_users() {
    step "Domain Join & User Setup"

    if ! timeout 15s id "$LOCAL_USER" &>/dev/null; then timeout 15s useradd -m -s
/bin/bash "$LOCAL_USER" || true; fi
    echo "$LOCAL_USER:$LOCAL_PASS" | chpasswd || true
    timeout 15s usermod -aG sudo "$LOCAL_USER" 2>/dev/null || timeout 15s usermod -aG
wheel "$LOCAL_USER" 2>/dev/null || true

    if [[ "$PKG" == "apt-get" ]]; then
        run_retry apt-get install -y realmd sssd sssd-tools libnss-sss libpam-sss adcli
packagekit
        if ! grep -q "pam_mkhomedir.so" /etc/pam.d/common-session; then
            echo "session optional pam_mkhomedir.so skel=/etc/skel umask=077" >>
/etc/pam.d/common-session
        fi
    else
        run_retry $PKG install -y realmd sssd oddjob oddjob-mkhomedir adcli samba-common-
tools
    fi

    if ! ping -c 1 -W 2 "$DOMAIN_FQDN" &>/dev/null; then error "DNS setup failed. Cannot
join domain."; return; fi

    if command -v update-crypto-policies &>/dev/null; then
        update-crypto-policies --set DEFAULT:AD-SUPPORT >/dev/null 2>&1 || true
    fi

    if ! timeout 15s realm list | grep -q "$DOMAIN_FQDN"; then
        echo -e "\n${YELLOW}Enter AD Admin Username (e.g., ent_joeld):${NC}"
        read -p "User: " JOIN_USER
        realm join --verbose --user="$JOIN_USER" "$DOMAIN_FQDN"
    else
        success "Already joined. Enforcing state..."
    fi

    SSSD_CONF="/etc/sssd/sssd.conf"
```

```
    if [[ -f "$SSSD_CONF" ]]; then
        timeout 15s systemctl stop sssd || true

        # 1. BULLETPROOF SERVICES LINE
        if grep -q "^services" "$SSSD_CONF"; then
            sed -i 's/^services.*/services = nss, pam, ssh/' "$SSSD_CONF"
        else
            sed -i '/\[sssd\]/a services = nss, pam, ssh' "$SSSD_CONF"
        fi

        # 2. BULLETPROOF ACCESS & GROUPS
        grep -q "access_provider" "$SSSD_CONF" && sed -i
's/access_provider.*/access_provider = simple/' "$SSSD_CONF" || sed -i '/\[domain/a
access_provider = simple' "$SSSD_CONF"
        grep -q "simple_allow_groups" "$SSSD_CONF" && sed -i
"s/simple_allow_groups.*/simple_allow_groups = ${ALLOWED_LOGIN_GROUP}/" "$SSSD_CONF" ||
sed -i "/access_provider = simple/a simple_allow_groups = ${ALLOWED_LOGIN_GROUP}"
"$SSSD_CONF"

        # 3. BULLETPROOF SSH KEY MAPPINGS
        sed -i '/ldap_user_ssh_public_key/d' "$SSSD_CONF"
        sed -i '/ldap_user_extra_attrs/d' "$SSSD_CONF"
        sed -i '/\[domain/a ldap_user_extra_attrs =
info:sshPublicKey\nldap_user_ssh_public_key = info' "$SSSD_CONF"

        # 4. BULLETPROOF NAMING
        sed -i 's/use_fully_qualified_names.*/use_fully_qualified_names = False/'
"$SSSD_CONF"
        sed -i 's/fallback_homedir.*/fallback_homedir = \/home\/%u/' "$SSSD_CONF"

        # 5. BULLETPROOF PERFORMANCE TWEAKS
        sed -i '/ignore_group_members/d' "$SSSD_CONF"
        sed -i '/subdomain_enumerate/d' "$SSSD_CONF"
        sed -i '/\[domain/a ignore_group_members = True\nsubdomain_enumerate = False'
"$SSSD_CONF"

        timeout 30s systemctl start sssd || true

        if command -v sss_cache &>/dev/null; then
```

```
            log "Flushing SSSD cache..."
            sss_cache -E || true
        fi

        log "Configuring SSH daemon for AD-based keys..."
        sed -i '/AuthorizedKeysCommand/d' /etc/ssh/sshd_config
        echo -e "\nAuthorizedKeysCommand
/usr/bin/sss_ssh_authorizedkeys\nAuthorizedKeysCommandUser nobody" >> /etc/ssh/sshd_config

        if systemctl list-unit-files | grep -q "^ssh.service"; then
            systemctl restart ssh || true
        else
            systemctl restart sshd || true
        fi
    fi
}

mod_docker() {
    step "Installing & Configuring Docker"

    if ! command -v docker &>/dev/null; then
        if [[ "$PKG" == "apt-get" ]]; then run_retry apt-get install -y docker-ce docker-
ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
        else run_retry $PKG install -y docker-ce docker-ce-cli containerd.io docker-
buildx-plugin docker-compose-plugin; fi
    fi

    # Daemon Config (NO PROXIES HERE, only Systemd)
    mkdir -p /etc/docker
    cat > /etc/docker/daemon.json <<EOF
{
  "insecure-registries": [ ${INSECURE_REGISTRIES} ]
}
EOF

    # Systemd Proxy (Universally Safe)
    mkdir -p /etc/systemd/system/docker.service.d
    cat > /etc/systemd/system/docker.service.d/http-proxy.conf <<EOF
[Service]
```

```
Environment="HTTP_PROXY=${PROXY_URL}"
Environment="HTTPS_PROXY=${PROXY_URL}"
Environment="NO_PROXY=${NO_PROXY_LIST}"
EOF

    systemctl daemon-reload || true
    timeout 30s systemctl enable --now docker || true
    timeout 60s systemctl restart docker || true

    timeout 15s usermod -aG docker root 2>/dev/null || true
    if timeout 15s id "$LOCAL_USER" &>/dev/null; then timeout 15s usermod -aG docker
"$LOCAL_USER" 2>/dev/null || true; fi

    # Client Proxy (camelCase)
    mkdir -p /root/.docker
    cat > /root/.docker/config.json <<EOF
{
  "proxies": {
    "default": {
      "httpProxy": "${PROXY_URL}",
      "httpsProxy": "${PROXY_URL}",
      "noProxy": "${NO_PROXY_LIST}"
    }
  }
}
EOF

    if timeout 15s id "$LOCAL_USER" &>/dev/null; then
        USER_HOME=$(eval echo ~$LOCAL_USER)
        mkdir -p "$USER_HOME/.docker"
        cp /root/.docker/config.json "$USER_HOME/.docker/config.json"
        chown -R "$LOCAL_USER:$LOCAL_USER" "$USER_HOME/.docker" || true
    fi
}

mod_lazydocker() {
    step "Installing LazyDocker"
    if ! command -v lazydocker &>/dev/null; then
        run_retry curl -sSL
```

```
https://raw.githubusercontent.com/jesseduffield/lazydocker/master/scripts/install_update_linux.sh | bash
    fi
}

mod_dev_stack() {
    step "Installing Dev Stack"
    if [[ "$PKG" == "dnf" || "$PKG" == "yum" ]]; then
        $PKG clean packages >/dev/null 2>&1 || true
        if [[ "$PKG" == "dnf" ]]; then
            $PKG module reset php -y || true
            $PKG module install -y php:remi-${PHP_VERSION}
        else
            yum-config-manager --enable remi-php83 || true
            $PKG install -y php php-cli php-fpm php-mysqlnd php-gd
        fi
        $PKG install -y java-${JAVA_VERSION}-openjdk nginx MariaDB-server MariaDB-client postgresql-server nodejs
    else
        run_retry apt-get install -y php${PHP_VERSION} php${PHP_VERSION}-{cli,fpm,mysql,gd,mbstring,xml,curl,zip}
        run_retry apt-get install -y openjdk-${JAVA_VERSION}-jdk nginx mariadb-server postgresql nodejs npm
    fi

    mod_docker
    mod_lazydocker

    if command -v php &>/dev/null; then
        find /etc/php* -name "php.ini" 2>/dev/null | while read -r INI_FILE; do
            sed -i '/^http_proxy/d; /^https_proxy/d' "$INI_FILE"
            echo -e "\n; Proxy Settings\nhttp_proxy = \"${PROXY_URL}\"\nhttps_proxy = \"${PROXY_URL}\"" >> "$INI_FILE"
            if grep -q "allow_url_fopen" "$INI_FILE"; then sed -i 's/^allow_url_fopen.*/allow_url_fopen = On/' "$INI_FILE"
            else echo "allow_url_fopen = On" >> "$INI_FILE"; fi
        done
        if systemctl list-unit-files | grep -q php-fpm; then timeout 30s systemctl restart php-fpm || true; fi
```

```
        if systemctl list-unit-files | grep -q php${PHP_VERSION}-fpm; then timeout 30s
systemctl restart php${PHP_VERSION}-fpm || true; fi
    fi
}

mod_cockpit() {
    step "Installing Cockpit"
    if [[ "$PKG" == "apt-get" ]]; then run_retry apt-get install -y cockpit cockpit-
storaged cockpit-pcp cockpit-packagekit
    else run_retry $PKG install -y cockpit cockpit-storaged cockpit-pcp 2>/dev/null ||
run_retry $PKG install -y cockpit; fi

    mkdir -p /etc/systemd/system/cockpit.service.d
    echo -e
"[Service]\nEnvironment=\"HTTP_PROXY=${PROXY_URL}\"\nEnvironment=\"HTTPS_PROXY=${PROXY_URL
}\"\nEnvironment=\"NO_PROXY=${NO_PROXY_LIST}\"" >
/etc/systemd/system/cockpit.service.d/proxy.conf
    systemctl daemon-reload || true
    timeout 30s systemctl enable --now cockpit.socket || true
}

mod_cleanup() {
    step "Final Cleanup"
    if command -v tmux &>/dev/null; then $PKG remove -y tmux 2>/dev/null || true; fi
    rm -f /etc/tmux.conf
    if [[ "$PKG" == "apt-get" ]]; then run_retry apt-get install -y zsh git fail2ban; else
run_retry $PKG install -y zsh git fail2ban; fi

    systemctl disable systemd-networkd-wait-online.service 2>/dev/null || true
    systemctl mask systemd-networkd-wait-online.service 2>/dev/null || true
    if [[ -f /etc/rc.d/rc.local ]]; then chmod +x /etc/rc.d/rc.local; fi
    if grep -q "172.16.21.16" /etc/fstab; then sed -i '/172.16.21.16/d' /etc/fstab; fi

    if systemctl is-failed sssd-nss.socket &>/dev/null; then
        systemctl reset-failed || true
        timeout 30s systemctl restart sssd || true
    fi

    ESCAPED_GROUP=$(echo "$AD_SUDO_GROUP" | sed 's/ /\\ /g')
```

```
    echo "%${ESCAPED_GROUP} ALL=(ALL) NOPASSWD: ALL" > "/etc/sudoers.d/10-ad-admins"
    chmod 440 "/etc/sudoers.d/10-ad-admins"

    cat > /etc/fail2ban/jail.local <<EOF
[sshd]
enabled = true
port = ssh
logpath = %(sshd_log)s
maxretry = 3
bantime = 3600
EOF
    timeout 30s systemctl enable --now fail2ban || true
}

################################################################################
# 5. EXECUTION
################################################################################
detect_and_fix_os

show_help() {
    echo "Usage: $0 [OPTION]"
    echo "  --basics        Safe: Proxy, Certs, Firewalld, Domain, Security."
    echo "  --full          Everything (Basics + Dev Stack + Cockpit)."
    echo "  --resize-home   Shrink /home to 8GB (BACKUP+DESTROY+RECREATE+RESTORE)."
    echo "  --dev-stack     Install Docker, LazyDocker, PHP, DBs only."
}

if [[ $# -eq 0 ]]; then show_help; exit 0; fi

while [[ "$#" -gt 0 ]]; do
    case $1 in
        --basics) mod_proxy; mod_clock_fix; mod_certs; mod_base_repos; mod_base_tools;
mod_network; mod_firewall; mod_domain_users; mod_cleanup ;;
        --full) mod_proxy; mod_clock_fix; mod_certs; mod_base_repos; mod_base_tools;
mod_network; mod_firewall; mod_domain_users; mod_dev_repos; mod_dev_stack; mod_cockpit;
mod_cleanup ;;
        --resize-home) mod_resize_home ;;
        --dev-stack) mod_proxy; mod_dev_repos; mod_dev_stack ;;
        *) echo "Unknown option: $1"; show_help; exit 1 ;;
```

```
    esac
    shift
done

echo -e "\n${GREEN}[$(date +'%H:%M:%S')] === Setup Complete ===${NC}"
```

DISABLE WAYLAND FOR SUPPORT MESH COMPATIBILITY

1	**From the initial boot after installation on the login screen**
2	**Select your account and go to the bottom right to click the gear icon**
3	**Select Gnome on Xorg**
4	**Input password to proceed with login**

Open Terminal

```
sudo -i
nano /etc/gdm/custom.conf
```

1	Go to the line #WaylandEnable=false and Delete the hashtag '#'
2	To exit: CTRL + 'X'
3	Select 'Y' for yes
4	To save: 'enter' key

```
sudo dnf update -y
sudo reboot
```

**********COMPLETED**********

CHANGE COMPUTER NAME

1	Open Terminal **PC Name example: MYDNS-IT-C12-L.M21.GOV.LOCAL**
2	sudo hostnamectl set-hostname mydns-it-c12-l.m21.gov.local

**********COMPLETED**********

TO JOIN THE DOMAIN

Open Terminal

```
sudo nano /etc/environment
```

Add the following line to the file:

```
http_proxy="http://172.40.4.14:8080/"
https_proxy="http://172.40.4.14:8080/"
ftp_proxy="http://172.40.4.14:8080/"
no_proxy=127.0.0.1,localhost,.localdomain,172.30.0.0/20,172.26.21.0/24
HTTP_PROXY="http://172.40.4.14:8080/"
HTTPS_PROXY="http://172.40.4.14:8080/"
FTP_PROXY="http://172.40.4.14:8080/"
NO_PROXY=127.0.0.1,localhost,.localdomain,172.30.0.0/20,172.26.21.0/24
```

1	To exit: CTRL + 'X'
2	Select 'Y' for yes
3	To save: 'enter' key
4	Log out and back in again

```
sudo nano /etc/dnf/dnf.conf
```

Add the following line to the file:

```
fastestmirror=1
```

1	To exit: CTRL + 'X'
2	Select 'Y' for yes
3	To save: 'enter' key

On Fedora

```
sudo dnf -y install epel-release && sudo dnf -y install realmd sssd oddjob oddjob-mkhomedir adcli samba-common-tools authselect nano curl wget htop btop net-tools git zip unzip tar freeipa-client tmux
```

On Ubuntu

```
sudo apt -y install realmd sssd sssd-tools libnss-sss libpam-sss adcli samba-common-bin oddjob oddjob-mkhomedir packagekit nano curl wget htop btop net-tools git zip unzip tar freeipa-client tmux
```

Fix DNS

```
sudo unlink /etc/resolv.conf
sudo nano /etc/resolv.conf
```

Input the IP Address and the Domain Name into file

```
search m21.gov.local
nameserver 172.16.21.161
```

1	To exit: CTRL + 'X'
2	Select 'Y' for yes
3	To save: 'enter' key

```
sudo nano /etc/hosts
```

Input the following lines into file

```
172.16.21.161 m21.gov.local M21.GOV.LOCAL
172.16.21.16 mydns-0ic16.m21.gov.local mydns-0ic16
```

1	To exit: CTRL + 'X'
2	Select 'Y' for yes
3	To save: 'enter' key

```
sudo realm discover M21.GOV.LOCAL
ping -c 4 M21.GOV.LOCAL
```

To stop ping: CTRL + 'C'

```
sudo realm join -U ent_username@M21.GOV.LOCAL m21.gov.local -v
```

Input Ent Account Password

To ensure that it was successful run the realm join code again and you should see "**Already joined to this domain**"

**********COMPLETED**********

GROUP POLICY CONFLICT RESOLVE (to login without wifi)

Open Terminal

```
sudo nano /etc/sssd/sssd.conf
```

Input at the end of the file

```
ad_gpo_access_control = permissive
```

Your "/etc/sssd/sssd.conf" should look like this. Make all necessary changes or copy and paste this into the file replacing everything. Can use CTRL + K to cut entire lines until the file is empty.

```
[sssd]
domains = m21.gov.local
config_file_version = 2
services = nss, pam

[nss]
homedir_substring = /home
```

```
[domain/m21.gov.local]
default_shell = /bin/bash
krb5_store_password_if_offline = True
cache_credentials = True
krb5_realm = M21.GOV.LOCAL
realmd_tags = manages-system joined-with-adcli
id_provider = ad
fallback_homedir = /home/%u
ad_domain = m21.gov.local
use_fully_qualified_names = False
ldap_id_mapping = True
access_provider = ad
ad_gpo_access_control = permissive
```

1	To exit: CTRL + 'X'
2	Select 'Y' for yes
3	To save: 'enter' key

On Fedora

```
sudo authselect select sssd with-mkhomedir
sudo systemctl restart sssd
```

On Ubuntu

```
sudo pam-auth-update --enable mkhomedir
sudo systemctl restart sssd
```

On CentOS 7

```
sudo authconfig --enablesssdauth --enablesssd --enablemkhomedir --updateall
sudo systemctl restart sssd
```

**********COMPLETED**********

TO MAKE AD ACCOUNT A SUDOER

Open Terminal

```
sudo nano /etc/sudoers.d/domain_admins
```

1	Input line : firstname.lastname ALL=(ALL) ALL
2	To allow all ICT Staff: %ICT\ Staff\ SG\ M21 ALL=(ALL:ALL) ALL
	cn=mydns ict staff sg,ou=security groups_m21,ou=mydns,dc=m21,dc=gov,dc=local
3	To exit: CTRL + 'X'
4	Select 'Y' for yes
5	To save: 'enter' key

**********COMPLETED**********

TO MOUNT SHARE DRIVE

1	Launch the Files app -> OTHER LOCATIONS -> Bottom of window to enter address
2	Input: smb://172.16.21.16/
3	Toggle on REGISTERED USER
4	Input: YOUR DOMAIN ACCOUNT USERNAME and PASSWORD
5	Domain: M21.GOV.LOCAL or 172.16.21.161

**********COMPLETED**********

TO ADD PRINTER

Open Terminal

HP Printers

```
dnf search hplip
sudo dnf install hplip hplip-gui -y
hp-setup
hp-setup ‘printer IP Address’
```

1	Select detected printer
2	Follow next prompt until the end

XEROX Printers

```
Open Terminal
wget
http://download.support.xerox.com/pub/drivers/CQ8580/drivers/linux/pt_BR/XeroxOfficev5Pkg-Linuxx86_64-5.20.661.4684.rpm
sudo dnf -y localinstall XeroxOfficev5Pkg-Linuxx86_64-5.20.661.4684.rpm
```

NOTE: DO NOT PRINT A TEST PAGE!! Print a regular text document to test

**********COMPLETED**********

TO REPLACE FEDORA LOGO

Download Image and rename as: MYDNS-Logo

MYDNS-Logo.png

1	Go to EXTENSION MANAGER -> SYSTEM EXTENSIONS -> BACKGROUND LOGO
2	Click on the gear icon to get the background settings
3	Go to LOGO -> Filename to attach the MYDNS-Logo.png file -> Filename (dark) to attach the MYDNS-Logo.png file
4	Scroll down to OPTIONS -> Toggle on Show for all backgrounds

**********COMPLETED**********

Adding Certificate File to Local Machine (Fedora)

Browse to **172.16.21.16>fileserver2>General>IT FILES>prx** and copy the **GORTT.pem** file to a folder on the local machine.

1. Navigate to the location of the certificate file in Terminal (or right click and open from the location)
2. Move the ceritficate file to the proper location with the following command:
   `sudo mv GORTT.pem /etc/pki/ca-trust/source/anchors/GORTT.pem`
3. Update trusted certificates with the following command:
   `sudo update-ca-trust`

[Adding Certificate File to Local Machine (Ubuntu)]

Browse to **172.16.21.16>fileserver2>General>IT FILES>prx** and copy the **GORTT.pem** file to a folder on the local machine.

```
sudo apt-get install -y ca-certificates
```

1. Navigate to the location of the certificate file in Terminal (or right click and open from the location)

```
openssl x509 -in GORTT.pem -out GORTT.crt
```

1. Move the ceritficate file to the proper location with the following command:

   ```
   sudo mv GORTT.crt /usr/local/share/ca-certificates
   ```

2. Update trusted certificates with the following command:

   ```
   sudo update-ca-certificates
   ```

HELPFUL APPS

1	Extension Manager flatpak install flathub com.mattjakeman.ExtensionManager
2	GNOME Tweaks (sudo dnf install gnome-tweaks)
3	OnlyOffice https://download.onlyoffice.com/install/desktop/editors/linux/onlyoffice-desktopeditors.x86_64.rpm sudo dnf -y localinstall onlyoffice-desktopeditors.x86_64.rpm

4	Element flatpak install flathub im.riot.Riot
5	Google Chome (Fedora) wget https://dl.google.com/linux/direct/google-chrome-stable_current_x86_64.rpm sudo dnf -y localinstall google-chrome-stable_current_x86_64.rpm
6	Google Chrome (Ubuntu) sudo apt install curl software-properties-common apt-transport-https ca-certificates -y curl -fSsL https://dl.google.com/linux/linux_signing_key.pub | gpg --dearmor | sudo tee /usr/share/keyrings/google-chrome.gpg > /dev/null echo deb [arch=amd64 signed-by= /usr/share/keyrings/google-chrome.gpg] http://dl.google.com/linux/chrome/deb/ stable main | sudo tee /etc/apt/sources.list.d/google-chrome.list sudo apt update sudo apt -y install google-chrome-stable

HELPFUL EXTENSIONS

1	Dash to Dock - Displays a dynamic centered Taskbar
2	Dash to Panel - Displays screen width static Taskbar
3	Vitals - displays the PC health at the top right
4	Desktop icons NG (Ding) - display anything saved to desktop
5	Clipboard History - enables clipboard history tool

**********COMPLETED**********

Revision #72
Created 2023-11-29 15:32:58 AST by Jamila Anthony
Updated 2026-04-20 23:45:55 AST by Joel Duncan