

Switch Over Instructions

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

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

Server Setup Script

```
#!/usr/bin/env bash  
#####  
# Comprehensive Domain Join & Configuration Script  
# - Prioritizes Network/Proxy setup before package installation.  
# - Prompts for Cockpit and Docker CE/Podman installation.  
# - Correctly installs EPEL for AlmaLinux 10 from URL.  
# - Incorporates user-provided procedural steps with SSSD refinement.  
# - Automatically prefixes AD username with "ent_".  
# - Adds Fail2ban and a custom MOTD for system stats.  
# - Optional Development Stack (PHP, Node, MariaDB, Nginx, Caddy, PostgreSQL).  
# - Automated Tmux environment with session persistence.  
# - Colorized output for better readability.  
#####  
  
#-----  
# 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,.localdomain" # Base no_proxy entries
NO_PROXY_CUSTOM="172.30.0.0/20,172.26.21.0/24"      # Your custom NO_PROXY CIDRs


TIMEZONE="America/Port_of_Spain"     # Your desired timezone


# AD Group for Sudoers (Raw name, spaces are okay here. Script will escape.)
AD_SUDO_GROUP_RAW_NAME="ICT Staff SG"


#-----
# 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


set -euo pipefail
LOG_FILE="/var/log/setup-domain-$(date +%Y%m%d-%H%M%S).log"
exec >>(tee -a "$LOG_FILE") 2>&1 # Log stdout and stderr to file and console
echo -e "${GREEN}=== Script started at $(date) by $(whoami) ===${NC}"
echo -e "${GREEN}=== Logging to ${LOG_FILE} ===${NC}"


#-----
# PRELIMINARY CHECKS & INPUT
#-----

[[ $EUID -ne 0 ]] && { echo -e "${RED}ERROR: This script must be run as root or with
sudo.${NC}"; exit 1; }

```

```

PKG_MANAGER=""

if command -v dnf &>/dev/null; then PKG_MANAGER="dnf";
elif command -v yum &>/dev/null; then PKG_MANAGER="yum"; # Fallback for CentOS 7
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}"; exit 1; fi

echo -e "${BLUE}INFO:${NC} Detected package manager: ${PURPLE}$PKG_MANAGER${NC}"


source /etc/os-release
OS_ID_LOWER=$(echo "$ID" | tr '[:upper:]' '[:lower:]')
OS_VER="${VERSION_ID%.*}"
echo -e "${BLUE}INFO:${NC} Detected OS: ${PURPLE}$PRETTY_NAME${NC} (ID: $ID, Version:
$VERSION_ID)"


echo -e "\n${CYAN}--- User Input ---${NC}"
read -rp "$(echo -e "${CYAN}Enter the SERVICE part of the hostname (e.g., mydns-it-cl2-l):
${NC}")" SERVICE_NAME_PART
if [[ ! "$SERVICE_NAME_PART" =~ ^[a-zA-Z0-9-]+$ ]]; then
    echo -e "${RED}ERROR: Invalid service name part.${NC}"; exit 1; fi
TARGET_HOSTNAME_FQDN="${SERVICE_NAME_PART}.${DOMAIN_FQDN}"
TARGET_HOSTNAME_FQDN_LC=$(echo "$TARGET_HOSTNAME_FQDN" | tr '[:upper:]' '[:lower:]')


read -rp "$(echo -e "${CYAN}Enter your AD username SUFFIX (the part after 'ent_'):
${NC}")" AD_USER_SUFFIX
if [[ -z "$AD_USER_SUFFIX" ]]; then echo -e "${RED}ERROR: AD username suffix cannot be
empty.${NC}"; exit 1; fi
AD_USER_FOR_JOIN="ent_${AD_USER_SUFFIX}"
echo -e "${BLUE}INFO:${NC} Using full AD username: ${PURPLE}${AD_USER_FOR_JOIN}${NC}"


read -rsp "$(echo -e "${CYAN}Enter AD password for '${AD_USER_FOR_JOIN}': ${NC}")"
AD_PASSWORD
echo
if [[ -z "$AD_PASSWORD" ]]; then echo -e "${RED}ERROR: AD Password cannot be empty.${NC}";
exit 1; fi


NO_PROXY_FULL="${NO_PROXY_INITIAL},${DOMAIN_FQDN,,},${DOMAIN_FQDN,,},${DC_DNS_IP},${NTP_S
ERVER},${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" | awk -F, '{for(i=1;i<=NF;i++)arr[$i]}END{for(k in
arr)printf "%s,",k}' | sed 's/,,$//')

#-----
# SCRIPT FUNCTIONS
#-----

log_step() { echo -e "\n${GREEN}--- [STEP $1] $2 ---${NC}"; }

change_hostname() {
    log_step "1/16" "Setting Hostname to ${TARGET_HOSTNAME_FQDN_LC}"
    if ! hostnamectl set-hostname "$TARGET_HOSTNAME_FQDN_LC"; then
        echo -e "${RED}ERROR: Failed to set hostname.${NC}"; return 1; fi
    echo -e "${BLUE}INFO:${NC}   Hostname set to: ${PURPLE}${(hostnamectl hostname)}${NC}"
}

configure_proxy() {
    log_step "2/16" "Configuring Proxy Settings"
    if [[ -z "$HTTP_PROXY_URL" ]]; then
        echo -e "${YELLOW}WARN: HTTP_PROXY_URL is not set. Skipping proxy
configuration.${NC}"
        unset http_proxy https_proxy ftp_proxy no_proxy HTTP_PROXY HTTPS_PROXY FTP_PROXY
NO_PROXY
        [[ "$PKG_MANAGER" == "dnf" || "$PKG_MANAGER" == "yum" ]] && sed -i '/^proxy=/d'
/etc/$PKG_MANAGER/$PKG_MANAGER.conf 2>/dev/null
        [[ "$PKG_MANAGER" == "apt" ]] && rm -f /etc/apt/apt.conf.d/80scriptproxy
2>/dev/null
        return
    fi
    cat > /etc/environment <<EOF
http_proxy="${HTTP_PROXY_URL}"
https_proxy="${HTTP_PROXY_URL}"
ftp_proxy="${HTTP_PROXY_URL}"
no_proxy="${NO_PROXY_FULL}"
HTTP_PROXY="${HTTP_PROXY_URL}"
HTTPS_PROXY="${HTTP_PROXY_URL}"
FTP_PROXY="${HTTP_PROXY_URL}"
NO_PROXY="${NO_PROXY_FULL}"
EOF
    echo -e "${BLUE}INFO:${NC}   /etc/environment configured."

```

```

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

if [[ "$PKG_MANAGER" == "dnf" || "$PKG_MANAGER" == "yum" ]]; then
    sed -i '/^proxy=/d' /etc/$PKG_MANAGER/$PKG_MANAGER.conf 2>/dev/null
    echo "proxy=${HTTP_PROXY_URL}" >> /etc/$PKG_MANAGER/$PKG_MANAGER.conf
    if ! grep -q "^fastestmirror=" /etc/$PKG_MANAGER/$PKG_MANAGER.conf; then echo
"fastestmirror=1" >> /etc/$PKG_MANAGER/$PKG_MANAGER.conf;
    else sed -i 's/^fastestmirror=.*fastestmirror=1/'
/etc/$PKG_MANAGER/$PKG_MANAGER.conf; fi
    echo -e "${BLUE}INFO:${NC} □ $PKG_MANAGER proxy and fastestmirror configured."
elif [[ "$PKG_MANAGER" == "apt" ]]; then
    cat > /etc/apt/apt.conf.d/80scriptproxy <<EOF
Acquire::http::Proxy "${HTTP_PROXY_URL}";
Acquire::https::Proxy "${HTTP_PROXY_URL}";
Acquire::ftp::Proxy "${HTTP_PROXY_URL}";
EOF
    echo -e "${BLUE}INFO:${NC} □ APT proxy configured."
fi
}

install_packages() {
    log_step "3/16" "Installing Required Packages"
    # Clean up known repo files from previous failed runs to prevent blocking core package
installation
    rm -f /etc/yum.repos.d/MariaDB.repo /etc/yum.repos.d/caddy.repo
/etc/apt/sources.list.d/mariadb.sources /etc/apt/sources.list.d/caddy-stable.list
/etc/apt/sources.list.d/pgdg.list

    local common_pkgs="nano curl wget htop btop net-tools git zip unzip tar tmux chrony
open-vm-tools traceroute ncd"
    local dnf_base_pkgs="realmd sssd oddjob oddjob-mkhomedir adcli samba-common-tools
authselect dnf-plugins-core"
    local yum_base_pkgs="realmd sssd oddjob oddjob-mkhomedir adcli samba-common-tools
authconfig yum-utils" # For CentOS 7
    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"
```

```
if [[ "$PKG_MANAGER" == "dnf" || "$PKG_MANAGER" == "yum" ]]; then  
    echo -e "${BLUE}INFO:${NC} Ensuring EPEL repository is installed..."  
    # 1. Enable CRB/PowerTools repo first, as EPEL often depends on it.  
    if [[ "$OS_ID_LOWER" =~ ^(almalinux|rhel|centos|rocky)$ ]]; then  
        if [[ "$OS_VER" -ge 9 ]]; then  
            dnf config-manager --set-enabled crb || echo -e "${YELLOW}WARN: Failed to  
enable CRB repository.${NC}"  
        elif [[ "$OS_VER" -eq 8 ]]; then  
            dnf config-manager --set-enabled powertools || dnf config-manager --set-  
enabled PowerTools || echo -e "${YELLOW}WARN: Failed to enable PowerTools repo.${NC}"  
        fi  
    fi  
  
    # 2. Install EPEL Release package.  
    if ! $PKG_MANAGER -y list installed epel-release &>/dev/null; then  
        # For AlmaLinux 10, try direct URL first.  
        if [[ "$OS_ID_LOWER" == "almalinux" && "$OS_VER" -eq 10 ]]; then  
            echo -e "${BLUE}INFO:${NC} Attempting to install EPEL for AlmaLinux 10  
directly from URL..."  
            local epel_rpm_url="https://dl.fedoraproject.org/pub/epel/epel-release-  
latest-10.noarch.rpm"  
            if ! dnf -y install "${epel_rpm_url}"; then  
                echo -e "${YELLOW}WARN: Failed to install EPEL from URL. Some packages  
may not be available.${NC}"  
            else  
                echo -e "${BLUE}INFO:${NC} □ EPEL release installed for AlmaLinux 10  
from URL."  
            fi  
        else # For other DNF/YUM-based systems  
            if ! $PKG_MANAGER -y install epel-release; then  
                echo -e "${YELLOW}WARN: Could not install 'epel-release' package.  
Some packages may be unavailable.${NC}"  
            else  
                echo -e "${BLUE}INFO:${NC} □ 'epel-release' package installed."  
            fi  
        fi  
    fi  
fi
```

```

else
    echo -e "${BLUE}INFO:${NC} EPEL repository is already installed."
fi
# --- End EPEL Setup ---

local pkgs_to_install
if [[ "$PKG_MANAGER" == "dnf" ]]; then pkgs_to_install="${dnf_base_pkgs}
${common_pkgs}";
else pkgs_to_install="${yum_base_pkgs} ${common_pkgs}"; fi # yum

echo -e "${BLUE}INFO:${NC} Installing main packages for $ID ($PKG_MANAGER)..."
if ! $PKG_MANAGER -y install ${pkgs_to_install}; then
    echo -e "${RED}ERROR: Failed to install packages. Check errors above.${NC}";
exit 1;
fi

elif [[ "$PKG_MANAGER" == "apt" ]]; then
    export DEBIAN_FRONTEND=noninteractive
    echo -e "${BLUE}INFO:${NC} Updating package lists for APT..."
    if ! apt-get update -qq; then echo -e "${RED}ERROR: apt-get update failed.${NC}";
exit 1; fi
    local pkgs_to_install="${apt_base_pkgs} ${common_pkgs}"
    echo -e "${BLUE}INFO:${NC} Installing main packages for $ID (APT)..."
    if ! apt-get -y install ${pkgs_to_install}; then
        echo -e "${RED}ERROR: Failed to install APT packages. Check errors
above.${NC}"; exit 1;
    fi
fi

echo -e "${BLUE}INFO:${NC} □ Required packages installed."
}

configure_dns_and_hosts() {
    log_step "4/16" "Configuring DNS and /etc/hosts"
    unlink /etc/resolv.conf 2>/dev/null || echo -e "${BLUE}INFO:${NC} /etc/resolv.conf not
a symlink or unlinking failed."
    cp /etc/resolv.conf /etc/resolv.conf.scriptbak-$(date +%Y%m%d-%H%M%S) 2>/dev/null ||
true
    cat > /etc/resolv.conf <<EOF
search ${DOMAIN_FQDN,,}

```



```

sleep 3; chronyc sources -v || echo -e "${YELLOW}WARN: chronyc sources failed.${NC}"
chronyc makestep || echo -e "${YELLOW}WARN: chronyc makestep failed.${NC}"
chronyc waitsync 30 0.5 5 || echo -e "${YELLOW}WARN: chronyc waitsync timeout.${NC}"
echo -e "${BLUE}INFO:${NC} Time sync status:"; chronyc tracking || echo -e
"${YELLOW}WARN: chronyc tracking failed.${NC}"
echo -e "${BLUE}INFO:${NC} □ Time configuration attempted."
}

join_ad_domain() {
    log_step "6/16" "Joining Active Directory Domain: ${DOMAIN_FQDN}"
    if ! command -v realm &> /dev/null; then echo -e "${RED}ERROR: realm command not
found.${NC}"; exit 1; fi
    if realm list | grep -q -i "domain-name: ${DOMAIN_FQDN}"; then
        echo -e "${BLUE}INFO:${NC} □ Already joined to domain ${DOMAIN_FQDN}." ; return 0;
    fi
    realm discover "${DOMAIN_FQDN,,}" || echo -e "${YELLOW}WARN: Domain discovery
failed.${NC}"
    ping -c 2 "${DOMAIN_FQDN,,}" || echo -e "${YELLOW}WARN: Ping to ${DOMAIN_FQDN,,}
failed.${NC}"
    echo -e "${BLUE}INFO:${NC} Joining domain as '${AD_USER_FOR_JOIN}'..."
    if echo "${AD_PASSWORD}" | realm join -v -U "${AD_USER_FOR_JOIN}" "${DOMAIN_FQDN,,}";
then
        echo -e "${BLUE}INFO:${NC} □ Successfully joined domain ${DOMAIN_FQDN}."; unset
AD_PASSWORD
    else
        echo -e "${RED}ERROR: Domain join failed. Check logs and credentials.${NC}"; unset
AD_PASSWORD; exit 1; fi
}

configure_sssd_mkhomedir() {
    log_step "7/16" "Configuring SSSD and Automatic Home Directory Creation"
    local sssd_conf="/etc/sss/sss.conf"
    if [[ ! -f "$sss_conf" ]]; then
        echo -e "${YELLOW}WARN: ${sss_conf} not found. Domain join might have failed.
Skipping SSSD config.${NC}"
        return 1
    fi

    echo -e "${BLUE}INFO:${NC} Optimizing SSSD configuration process..."

```

```

# The new section to be added or to replace the old one
local new_domain_section
new_domain_section=$(printf '%s\n' \
    "[domain/${DOMAIN_FQDN,,}]" \
    "id_provider = ad" \
    "access_provider = ad" \
    "ad_domain = ${DOMAIN_FQDN,,}" \
    "krb5_realm = ${DOMAIN_FQDN^^}" \
    "realmd_tags = manages-system joined-with-adcli" \
    "cache_credentials = True" \
    "krb5_store_password_if_offline = True" \
    "ldap_id_mapping = True" \
    "use_fully_qualified_names = False" \
    "fallback_homedir = /home/%u" \
    "default_shell = /bin/bash" \
    "override_homedir = /home/%u" \
    "create_homedir = true" \
    "ad_gpo_access_control = permissive"
)
cp "$sssd_conf" "${sssd_conf}.bak-$(date +%s)"
local temp_conf; temp_conf=$(mktemp)

# Remove the old domain section entirely
awk -v domain_name="${DOMAIN_FQDN,,}" '
    BEGIN { in_section_to_remove = 0 }
    $0 ~ "^[\\domain\\\\/" domain_name "\\]" { in_section_to_remove = 1; next }
    in_section_to_remove && /^s*[\\/] { in_section_to_remove = 0 }
    !in_section_to_remove { print }
' "$sssd_conf" > "$temp_conf"

# Append the new, complete domain section
echo -e "\n${new_domain_section}" >> "$temp_conf"

if [[ -s "$temp_conf" ]]; then
    mv "$temp_conf" "$sssd_conf"
    chmod 0600 "$sssd_conf"
    echo -e "${BLUE}INFO:${NC} sssd.conf updated. Restarting service..."
    systemctl restart sssd || echo -e "${YELLOW}WARN: Failed to restart SSSD after
config change.${NC}"

```

```

else
    echo -e "${RED}ERROR: Generated temporary sssd.conf was empty. No changes
made.${NC}"
    rm -f "$temp_conf"; return 1; fi

echo -e "${BLUE}INFO:${NC} Enabling automatic home directory creation..."
if [[ "$PKG_MANAGER" == "dnf" ]]; then
    if command -v authselect &>/dev/null; then
        if authselect check && authselect current | grep -q "with-mkhomedir"; then
            echo -e "${BLUE}INFO:${NC} authselect already has mkhomedir enabled."
        else
            authselect select sssd with-mkhomedir --force || echo -e "${YELLOW}WARN:
authselect command failed.${NC}"
            echo -e "${BLUE}INFO:${NC} □ authselect configured for SSSD with
mkhomedir."
        fi
    else echo -e "${YELLOW}WARN: authselect command not found.${NC}"; fi
elif [[ "$PKG_MANAGER" == "yum" ]]; then # CentOS 7
    authconfig --enablesssd --enablesssdauth --enablemkhomedir --update
    echo -e "${BLUE}INFO:${NC} □ authconfig updated for SSSD and mkhomedir on CentOS
7."
elif [[ "$PKG_MANAGER" == "apt" ]]; then
    if command -v pam-auth-update &>/dev/null; then
        if ! grep -q "pam_mkhomedir.so" /etc/pam.d/common-session; then
            pam-auth-update --enable mkhomedir || echo -e "${YELLOW}WARN: pam-auth-
update --enable mkhomedir failed.${NC}"
            echo -e "${BLUE}INFO:${NC} □ pam-auth-update attempted for mkhomedir."
        else echo -e "${BLUE}INFO:${NC} mkhomedir already configured in PAM
(Ubuntu)."; fi
    else echo -e "${YELLOW}WARN: pam-auth-update command not found.${NC}"; fi
fi
echo -e "${BLUE}INFO:${NC} □ SSSD configuration and mkhomedir setup attempted."
}

configure_sudoers() {
    log_step "8/16" "Configuring Sudoers for AD Group: ${AD_SUDO_GROUP_RAW_NAME}"
    if [[ -z "$AD_SUDO_GROUP_RAW_NAME" ]]; then
        echo -e "${YELLOW}WARN: AD_SUDO_GROUP_RAW_NAME empty. Skipping sudoers.${NC}";
    return; fi
}

```

```

local sudoers_file="/etc/sudoers.d/90-domain-${DOMAIN_NETBIOS,,}-admins"
local group_name_escaped=$(echo "$AD_SUDO_GROUP_RAW_NAME" | sed 's/ /\ /g')
local sudo_entry_line="%${DOMAIN_NETBIOS}\\\${group_name_escaped} ALL=(ALL:ALL) ALL"
echo -e "${BLUE}INFO:${NC} Creating sudoers entry: ${sudo_entry_line}"
echo "${sudo_entry_line}" > "${sudoers_file}" && chmod 0440 "${sudoers_file}"
if visudo -cf "${sudoers_file}"; then echo -e "${BLUE}INFO:${NC} [Sudoers file
${sudoers_file} validated.";
    else echo -e "${RED}ERROR: Sudoers file syntax error!${NC}"; rm -f "${sudoers_file}";
return 1; fi
}

install_dev_stack() {
    # This function is now more resilient. Failures in one component won't stop others.
    set +e # Temporarily disable exit-on-error for this function

    log_step "9/16" "Installing Development Stack (Optional)"
    read -rp "$(echo -e "${CYAN}Install Development Stack (Webservers, Databases, etc)?
[y/N]: ${NC}")" install_choice
    install_choice=$(echo "$install_choice" | tr '[:upper:]' '[:lower:]')
    if [[ "$install_choice" != "y" ]]; then
        echo -e "${BLUE}INFO:${NC} Skipping Development Stack installation."
        set -e
        return
    fi

    # --- PHP Installation ---
    read -rp "$(echo -e "${CYAN}Install PHP? [y/N]: ${NC}")" php_install_choice
    if [[ "$(echo "$php_install_choice" | tr '[:upper:]' '[:lower:]')" == "y" ]]; then
        local php_versions=("8.1" "8.2" "8.3" "8.4")
        read -rp "$(echo -e "${CYAN}Which PHP version? (${php_versions[*]}): ${NC}")"
php_choice
        if [[ " ${php_versions[*]} " =~ " ${php_choice} " ]]; then
            echo -e "${BLUE}INFO:${NC} Installing PHP ${php_choice}..."
            local php_modules="php php-cli php-fpm php-mysqlnd php-gd php-xml php-mbstring
php-intl php-zip php-curl php-opcache php-pgsql"
            local php_install_success=false
            if [[ "$PKG_MANAGER" == "dnf" || "$PKG_MANAGER" == "yum" ]]; then
                $PKG_MANAGER -y install https://rpms.remirepo.net/enterprise/remi-release-
$(rpm -E %rhel).rpm

```

```

        if [[ "$PKG_MANAGER" == "dnf" ]]; then dnf module reset php -y && dnf
module enable php:remi-${php_choice} -y;
        else yum-config-manager --enable remi-php${php_choice};//.; fi

    if $PKG_MANAGER -y install ${php_modules}; then
        php_install_success=true
    else
        echo -e "${YELLOW}WARN: Failed to install PHP ${php_choice} from Remi
repository. Trying system default.${NC}"
        if [[ "$PKG_MANAGER" == "dnf" ]]; then dnf module reset php -y; fi
        if $PKG_MANAGER -y install php php-cli php-fpm php-mysqlnd; then
php_install_success=true;
        else echo -e "${RED}ERROR: Fallback installation of system PHP also
failed.${NC}"; fi
        fi
    elif [[ "$PKG_MANAGER" == "apt" ]]; then
        add-apt-repository ppa:ondrej/php -y && apt-get update -qq
        local apt_php_modules="php${php_choice} php${php_choice}-cli
php${php_choice}-fpm php${php_choice}-mysql php${php_choice}-gd php${php_choice}-xml
php${php_choice}-mbstring php${php_choice}-intl php${php_choice}-zip php${php_choice}-curl
php${php_choice}-opcache php${php_choice}-pgsql"
        if apt-get -y install $apt_php_modules; then php_install_success=true; fi
    fi

    if [[ "$php_install_success" == true ]] && php -v &>/dev/null; then
        echo -e "${BLUE}INFO:${NC} □ PHP installed successfully."
        echo -e "${BLUE}INFO:${NC} Installing Composer..."
        curl -sS https://getcomposer.org/installer -o /tmp/composer-setup.php
        if php /tmp/composer-setup.php --install-dir=/usr/local/bin --
filename=composer; then
            echo -e "${BLUE}INFO:${NC} □ Composer installed."
        else echo -e "${RED}ERROR: Failed to install Composer.${NC}"; fi
        rm /tmp/composer-setup.php
    else echo -e "${RED}ERROR: PHP installation failed. Skipping Composer.${NC}";
fi

    else echo -e "${YELLOW}WARN: Invalid PHP version. Skipping PHP and Composer
installation.${NC}"; fi
fi

```

```

# --- Node.js Installation ---
read -rp "$(echo -e "${CYAN}Install Node.js? [y/N]: ${NC})" node_install_choice
if [[ "$(echo "$node_install_choice" | tr '[:upper:]' '[:lower:]')" == "y" ]]; then
    local node_versions=("20" "22")
    read -rp "$(echo -e "${CYAN}Which Node.js LTS version? (${node_versions[*]}):
${NC})" node_choice
    if [[ " ${node_versions[*]} " =~ " ${node_choice} " ]]; then
        echo -e "${BLUE}INFO:${NC} Installing Node.js ${node_choice}.x..."
        if curl -fsSL https://rpm.nodesource.com/setup_${node_choice}.x | bash - ||
curl -fsSL https://deb.nodesource.com/setup_${node_choice}.x | bash - ; then
            if $PKG_MANAGER install -y nodejs; then echo -e "${BLUE}INFO:${NC} □
Node.js ${node_choice} and npm installed.";
                else echo -e "${RED}ERROR: Failed to install Node.js package.${NC}"; fi
            else echo -e "${RED}ERROR: Failed to setup NodeSource repository.${NC}"; fi
            else echo -e "${YELLOW}WARN: Invalid Node.js version. Skipping.${NC}"; fi
        fi

# --- Web Servers ---
read -rp "$(echo -e "${CYAN}Install Nginx? [y/N]: ${NC})" nginx_choice
if [[ "$(echo "$nginx_choice" | tr '[:upper:]' '[:lower:]')" == "y" ]]; then
    if $PKG_MANAGER -y install nginx && systemctl enable --now nginx; then echo -e
"${BLUE}INFO:${NC} □ Nginx installed and started.";
        else echo -e "${RED}ERROR: Nginx installation failed.${NC}"; fi
    fi

read -rp "$(echo -e "${CYAN}Install Caddy? [y/N]: ${NC})" caddy_choice
if [[ "$(echo "$caddy_choice" | tr '[:upper:]' '[:lower:]')" == "y" ]]; then
    if [[ "$PKG_MANAGER" == "dnf" || "$PKG_MANAGER" == "yum" ]]; then
        dnf -y install 'dnf-command(copr)' && dnf copr enable @caddy/caddy -y
    elif [[ "$PKG_MANAGER" == "apt" ]]; then
        apt-get install -y debian-keyring debian-archive-keyring apt-transport-https
        curl -1sLf 'https://dl.cloudsmith.io/public/caddy/stable/gpg.key' | gpg --
dearmor -o /usr/share/keyrings/caddy-stable-archive-keyring.gpg
        curl -1sLf 'https://dl.cloudsmith.io/public/caddy/stable/debian.deb.txt' >
/etc/apt/sources.list.d/caddy-stable.list
        apt-get update -qq
    fi
    if $PKG_MANAGER -y install caddy && systemctl enable --now caddy; then echo -e
"${BLUE}INFO:${NC} □ Caddy installed and started.";
        else echo -e "${RED}ERROR: Caddy installation failed.${NC}"; fi

```

```

fi

# --- Databases ---

read -rp "$(echo -e "${CYAN}Install MariaDB 10.11 LTS Server? [y/N]: ${NC})"
mariadb_choice
if [[ "$(echo "$mariadb_choice" | tr '[:upper:]' '[:lower:]')" == "y" ]] && ! command
-v mysql &>/dev/null; then
    echo -e "${BLUE}INFO:${NC} Installing MariaDB 10.11 LTS..."
    curl -Ls https://downloads.mariadb.com/MariaDB/mariadb_repo_setup | bash -s -- --
mariadb-server-version="mariadb-10.11"
    if $PKG_MANAGER -y install MariaDB-server MariaDB-client && systemctl enable --now
mariadb; then
        echo -e "${BLUE}INFO:${NC} □ MariaDB 10.11 installed. Configuring..."
        mysql -e "ALTER USER 'root'@'localhost' IDENTIFIED VIA unix_socket;" || echo -
e "${YELLOW}WARN: Could not set passwordless root login for MariaDB.${NC}"
        echo -e "${YELLOW}IMPORTANT: Run 'sudo mariadb-secure-installation' to secure
your database.${NC}"
    else echo -e "${RED}ERROR: MariaDB installation failed.${NC}"; fi
    elif [[ "$(echo "$mariadb_choice" | tr '[:upper:]' '[:lower:]')" == "y" ]]; then echo
-e "${YELLOW}WARN: MySQL client detected. Skipping MariaDB install.${NC}"; fi

read -rp "$(echo -e "${CYAN}Install PostgreSQL Server? [y/N]: ${NC})" pg_choice
if [[ "$(echo "$pg_choice" | tr '[:upper:]' '[:lower:]')" == "y" ]] && ! command -v
psql &>/dev/null; then
    local pg_versions=("16" "15" "14"); read -rp "$(echo -e "${CYAN}PostgreSQL
version? (${pg_versions[*]}): ${NC})" pg_ver_choice
    if [[ " ${pg_versions[*]} " =~ " ${pg_ver_choice} " ]]; then
        echo -e "${BLUE}INFO:${NC} Installing PostgreSQL ${pg_ver_choice}..."
        if [[ "$PKG_MANAGER" == "dnf" || "$PKG_MANAGER" == "yum" ]]; then
            $PKG_MANAGER -y install
"https://download.postgresql.org/pub/repos/yum/reposrums/EL-$(rpm -E %rhel)-x86_64/pgdg-
redhat-repo-latest.noarch.rpm"
            $PKG_MANAGER -y install postgresql${pg_ver_choice}-server
/usr/pgsql-${pg_ver_choice}/bin/postgresql-${pg_ver_choice}-setup initdb
systemctl enable --now postgresql-${pg_ver_choice}
        elif [[ "$PKG_MANAGER" == "apt" ]]; then
            echo "deb http://apt.postgresql.org/pub/repos/apt/ $(lsb_release -cs)-pgdg
main" > /etc/apt/sources.list.d/pgdg.list
            curl -s https://www.postgresql.org/media/keys/ACCC4CF8.asc | gpg --dearmor

```

```

> /etc/apt/trusted.gpg.d/postgresql.gpg
    apt-get update -qq && apt-get -y install postgresql-${pg_ver_choice}
fi
    echo -e "${BLUE}INFO:${NC}  PostgreSQL ${pg_ver_choice} installed."
    else echo -e "${YELLOW}WARN: Invalid PostgreSQL version. Skipping.${NC}"; fi
    elif [[ "$(echo "$pg_choice" | tr '[:upper:]' '[:lower:]')" == "y" ]]; then echo -e
"${YELLOW}WARN: PostgreSQL client detected. Skipping install.${NC}"; fi

    set -e # Re-enable exit-on-error after this function
}

install_cockpit() {
    set +e
    log_step "10/16" "Installing Cockpit Web Console (Optional)"
    read -rp "$(echo -e "${CYAN}Install Cockpit web console? [y/N]: ${NC})"
install_choice
    install_choice=$(echo "$install_choice" | tr '[:upper:]' '[:lower:]')
    if [[ "$install_choice" != "y" ]]; then echo -e "${BLUE}INFO:${NC} Skipping Cockpit.";
set -e; return; fi

    echo -e "${BLUE}INFO:${NC} Installing Cockpit..."
    local pkgs="cockpit cockpit-podman pcp python3-pcp"; if [[ "$PKG_MANAGER" == "apt" ]];
then pkgs="cockpit cockpit-podman"; fi
    if ! $PKG_MANAGER -y install $pkgs; then echo -e "${RED}ERROR: Failed to install
Cockpit packages.${NC}"; set -e; return; fi
    [[ "$PKG_MANAGER" == "dnf" || "$PKG_MANAGER" == "yum" ]] && systemctl enable --now
pmcd pmlogger || echo -e "${YELLOW}WARN: Could not enable PCP services.${NC}"

    if [[ -n "$HTTP_PROXY_URL" ]]; then
        mkdir -p /etc/systemd/system/cockpit.service.d
        echo -e "[Service]\nEnvironment=\"HTTP_PROXY=${HTTP_PROXY_URL}\"
\"HTTPS_PROXY=${HTTP_PROXY_URL}\" \"NO_PROXY=${NO_PROXY_FULL}\"" >
/etc/systemd/system/cockpit.service.d/proxy.conf
        systemctl daemon-reload
        echo -e "${BLUE}INFO:${NC}  Cockpit proxy configured."
    fi

    if systemctl is-active --quiet firewalld; then
        firewall-cmd --add-service=cockpit --permanent && firewall-cmd --reload

```

```

        echo -e "${BLUE}INFO:${NC} □ Firewall rule added for Cockpit (firewalld)."
    elif command -v ufw &>/dev/null && ufw status | grep -q "Status: active"; then
        ufw allow 9090/tcp
        echo -e "${BLUE}INFO:${NC} □ Firewall rule added for Cockpit on port 9090 (UFW)."
    else echo -e "${YELLOW}WARN: Firewall not active/managed. Manual config for port 9090
needed.${NC}"; fi

    local cockpit_service="cockpit.socket"; [[ "$PKG_MANAGER" == "apt" ]] &&
cockpit_service="cockpit"

    if ! systemctl enable --now "$cockpit_service"; then echo -e "${RED}ERROR: Failed to
start/enable ${cockpit_service}.${NC}"; set -e; return; fi

    echo -e "${BLUE}INFO:${NC} □ Cockpit installation complete."
    echo -e "${CYAN}Access Cockpit at:${NC}"
    hostname -I 2>/dev/null | xargs -n1 | grep -v '^127' | while read -r ip; do echo -e
" ${GREEN}https://${ip}:9090${NC}"; done
    set -e
}

install_container_runtime() {
    log_step "11/16" "Installing Container Runtime"
    read -rp "$(echo -e "${CYAN}Install a container runtime? [y/N]: ${NC})" install_choice
    install_choice=$(echo "$install_choice" | tr '[:upper:]' '[:lower:]')
    [[ "$install_choice" != "y" ]] && echo -e "${BLUE}INFO:${NC} Skipping container
runtime." && return

    local runtime_selection
    while true; do
        read -rp "$(echo -e "${CYAN}Runtime? 'docker' (Docker CE), 'podman', or 'none':
${NC})" runtime_selection
        runtime_selection=$(echo "$runtime_selection" | tr '[:upper:]' '[:lower:]')
        if [[ "$runtime_selection" =~ ^(docker|podman|none)$ ]]; then break;
        else echo -e "${YELLOW}Invalid. Enter 'docker', 'podman', or 'none'.${NC}"; fi
    done
    if [[ "$runtime_selection" == "none" ]]; then echo -e "${BLUE}INFO:${NC} No runtime
selected."; return; fi

    if [[ "$runtime_selection" == "podman" ]]; then
        echo -e "${BLUE}INFO:${NC} Installing Podman..."
    fi
}

```

```

if $PKG_MANAGER -y install podman podman-docker; then
    if [[ -n "$HTTP_PROXY_URL" ]]; then
        mkdir -p /etc/containers/containers.conf.d; cat >
/etc/containers/containers.conf.d/01-proxy.conf <<EOF
[engine]
env = [ "HTTP_PROXY=${HTTP_PROXY_URL}", "HTTPS_PROXY=${HTTP_PROXY_URL}",
"NO_PROXY=${NO_PROXY_FULL}" ]
EOF
        fi
        echo -e "${BLUE}INFO:${NC} Podman installed successfully."
    else echo -e "${RED}ERROR: Podman installation failed.${NC}"; fi
elif [[ "$runtime_selection" == "docker" ]]; then
    echo -e "${BLUE}INFO:${NC} Installing Docker CE..."
    if [[ "$PKG_MANAGER" == "dnf" || "$PKG_MANAGER" == "yum" ]]; then
        if [[ "$OS_ID_LOWER" == "almalinux" && "$OS_VER" -eq 10 ]]; then
            echo -e "${YELLOW}WARN: Docker CE on AlmaLinux $OS_VER might conflict or need
'--allowerasing'.${NC}"
            local REPLY_DOCKER_AL10; read -rp "Proceed with Docker CE on AlmaLinux
$OS_VER? (y/N): " -n 1 REPLY_DOCKER_AL10 && echo
            if [[ ! "$REPLY_DOCKER_AL10" =~ ^[Yy]$ ]]; then echo "INFO: Skipping Docker CE
for AL$OS_VER."; return; fi
            fi
            dnf config-manager --add-repo https://download.docker.com/linux/centos/docker-
ce.repo || { echo "ERR: Add Docker repo failed."; return 1; }
            if ! dnf -y install docker-ce docker-ce-cli containerd.io docker-buildx-plugin
docker-compose-plugin; then
                echo -e "${RED}ERR: Docker CE install failed. Try with --allowerasing.${NC}";
return 1; fi
            elif [[ "$PKG_MANAGER" == "apt" ]]; then
                install -m 0755 -d /etc/apt/keyrings
                curl -fsSL https://download.docker.com/linux/ubuntu/gpg | gpg --dearmor -o
/etc/apt/keyrings/docker.gpg || { echo "ERR: GPG key failed."; return 1; }
                chmod a+r /etc/apt/keyrings/docker.gpg
                echo "deb [arch=$(dpkg --print-architecture) signed-
by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu $(lsb_release -
cs) stable" > /etc/apt/sources.list.d/docker.list
                apt-get update -qq
                if ! apt-get -y install docker-ce docker-ce-cli containerd.io docker-buildx-plugin
docker-compose-plugin; then

```

```

        echo -e "${RED}ERR: Docker CE install failed.${NC}"; return 1; fi
    fi
    # Common Docker post-install steps
    if [[ -n "$HTTP_PROXY_URL" ]]; then
        mkdir -p /etc/systemd/system/docker.service.d; cat
>/etc/systemd/system/docker.service.d/http-proxy.conf <<EOF
[Service]
Environment="HTTP_PROXY=${HTTP_PROXY_URL}" "HTTPS_PROXY=${HTTP_PROXY_URL}"
"NO_PROXY=${NO_PROXY_FULL}"
EOF
        mkdir -p /etc/docker; cat >/etc/docker/daemon.json <<EOF
{ "proxies": { "http-proxy": "${HTTP_PROXY_URL}", "https-proxy": "${HTTP_PROXY_URL}", "no-
proxy": "${NO_PROXY_FULL}" } }
EOF
    fi
    systemctl daemon-reload && systemctl enable --now docker || { echo -e "${RED}ERR:
Docker service failed.${NC}"; return 1; }
    if [[ -n "${SUDO_USER:-}" ]] && id -u "$SUDO_USER" &>/dev/null && ! id -nG
"$SUDO_USER" | grep -qw docker; then
        usermod -aG docker "$SUDO_USER"; echo -e "${BLUE}INFO:${NC} Added $SUDO_USER to
docker group."; fi
    echo -e "${BLUE}INFO:${NC}   Docker CE installed."
fi
}

install_fail2ban() {
    set +e
    log_step "12/16" "Installing Fail2ban (Optional)"
    read -rp "$(echo -e "${CYAN}Install and configure Fail2ban for SSH? [y/N]: ${NC})"
install_choice
    install_choice=$(echo "$install_choice" | tr '[:upper:]' '[:lower:]')
    if [[ "$install_choice" != "y" ]]; then echo -e "${BLUE}INFO:${NC} Skipping Fail2ban
installation."; set -e; return; fi

    echo -e "${BLUE}INFO:${NC} Installing Fail2ban..."
    if ! $PKG_MANAGER -y install fail2ban; then echo -e "${RED}ERROR: Fail2ban install
failed.${NC}"; set -e; return; fi

    echo -e "${BLUE}INFO:${NC} Configuring SSH jail for Fail2ban..."

```

```

cat > /etc/fail2ban/jail.local <<EOF
[sshd]
enabled = true
port = ssh
filter = sshd
logpath = /var/log/auth.log
maxretry = 3
bantime = 3600
EOF

if [[ "$PKG_MANAGER" == "dnf" || "$PKG_MANAGER" == "yum" ]]; then
    sed -i 's|/var/log/auth.log|/var/log/secure|' /etc/fail2ban/jail.local
fi

if ! systemctl enable --now fail2ban; then echo -e "${RED}ERROR: Failed to
start/enable Fail2ban.${NC}"; set -e; return; fi

echo -e "${BLUE}INFO:${NC} □ Fail2ban installed and SSH jail configured."
sleep 2
fail2ban-client status sshd || echo -e "${YELLOW}WARN: Could not get fail2ban
status.${NC}"
set -e
}

setup_tmux() {
    set +e
    log_step "13/16" "Setting up Automated Tmux Environment (Optional)"
    read -rp "$(echo -e "${CYAN}Setup persistent Tmux environment? [y/N]: ${NC})"
    tmux_choice
    if [[ "$(echo "$tmux_choice" | tr '[:upper:]' '[:lower:]')" != "y" ]]; then echo -e
"${BLUE}INFO:${NC} Skipping Tmux setup."; set -e; return; fi

    echo -e "${BLUE}INFO:${NC} Installing Tmux Plugin Manager (TPM)..."
    git clone https://github.com/tmux-plugins/tpm /etc/tmux/plugins/tpm || echo -e
"${YELLOW}WARN: Could not clone TPM. Maybe it exists?${NC}"

    echo -e "${BLUE}INFO:${NC} Creating global tmux configuration..."
    cat > /etc/tmux.conf <<'EOF'

# List of plugins
set -g @plugin 'tmux-plugins/tpm'

```

```

set -g @plugin 'tmux-plugins/tmux-sensible'
set -g @plugin 'tmux-plugins/tmux-resurrect'
set -g @plugin 'tmux-plugins/tmux-continuum'
# Restore session on tmux start
set -g @continuum-restore 'on'
# Initialize TMUX plugin manager (keep this line at the very bottom of tmux.conf)
run '/etc/tmux/plugins/tpm/tpm'
EOF

    echo -e "${BLUE}INFO:${NC} Creating tmux auto-start script..."
    cat > /etc/profile.d/98-tmux-autostart.sh <<'EOF'
#!/bin/bash
if [[ $- == *i* ]] && [[ -z "$TMUX" ]] && command -v tmux &>/dev/null; then
    tmux attach-session -t main || tmux new-session -s main
fi
EOF

    chmod +x /etc/profile.d/98-tmux-autostart.sh
    echo -e "${BLUE}INFO:${NC}  Tmux configured. It will start automatically on next SSH
login."
    echo -e "${YELLOW}To fetch plugins, start tmux and press 'prefix + I' (Ctrl+b, then
Shift+i).${NC}"
    set -e
}

setup_motd() {
    log_step "14/16" "Setting Up Dynamic MOTD (Message of the Day)"
    echo -e "${BLUE}INFO:${NC} Creating a dynamic message of the day..."
    # Disable default MOTD generation if they exist
    chmod -x /etc/update-motd.d/* &>/dev/null || true
    # Disable printing MOTD via pam_motd.so which can interfere
    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'
#!/bin/bash
# Only run for interactive shells on login, not inside tmux
if [[ $- != *i* ]] || [[ -n "$TMUX" ]]; then
    return
fi

```

```

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

echo ""
# Display Hostname and OS
HOSTNAME=$(hostname -f)
OS_INFO=$(grep PRETTY_NAME /etc/os-release | cut -d'"' -f2)
echo -e "Hostname: ${GREEN}$HOSTNAME${NC}"
echo -e "OS:      ${YELLOW}$OS_INFO${NC}"
echo ""

# Display Network Information
echo -e "${BLUE}Network Information:${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 "${BLUE}System Usage:${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')"
```

EOF

```

    chmod +x /etc/profile.d/99-custom-motd.sh
    echo -e "${BLUE}INFO:${NC}  □ Dynamic MOTD script created. It will be displayed on next SSH login."
```

```

}

final_summary() {
    log_step "15/16" "Final Setup Summary"
    echo "Hostname: $(hostnamectl hostname 2>/dev/null || echo "N/A")"
    local domain_name_info=$(realm list 2>/dev/null | grep 'domain-name:' | awk '{print $2}')
    echo "Joined Domain: ${domain_name_info:-'Not Joined or Error'}"
    echo "Current Time: $(date)"
    echo "Timezone: $(timedatectl status 2>/dev/null | grep 'Time zone' | awk '{print $3,$4,$5}' || echo "N/A")"
    echo "NTP Synchronized: $(timedatectl status 2>/dev/null | grep 'NTP synchronized' | awk '{print $3}' || echo "N/A")"
    echo -e "Log file: ${PURPLE}$LOG_FILE${NC}"
    echo -e
    "${YELLOW}===== ${NC}"
    echo -e "${YELLOW}IMPORTANT: Review log for errors. Reboot may be required.${NC}"
    echo -e
    "${YELLOW}===== ${NC}"
}

system_updates_interactive() {
    log_step "16/16" "System Updates (Optional)"
    read -rp "$(echo -e "${CYAN}Perform full system updates now? [y/N]: ${NC})")"
    update_choice
    update_choice=$(echo "$update_choice" | tr '[:upper:]' '[:lower:]')
    if [[ "$update_choice" == "y" ]]; then
        echo -e "${BLUE}INFO:${NC} Starting system update..."
        if ! $PKG_MANAGER -y upgrade; then echo -e "${RED}ERROR: System upgrade failed.${NC}";
        else
            echo -e "${BLUE}INFO:${NC} System updates installed."
            read -rp "$(echo -e "${CYAN}Reboot now? [y/N]: ${NC})")" r; r=$(echo "$r"|tr '[:upper:]' '[:lower:]')
            if [[ "$r" == "y" ]]; then echo "Rebooting..."; sleep 5; reboot; fi
        fi
    else echo -e "${BLUE}INFO:${NC} Skipping updates."; fi
}

```

```

#-----
# MAIN EXECUTION FLOW
#-----

echo -e "\n${GREEN}--- Starting Main Execution Sequence ---${NC}"

change_hostname
configure_proxy
install_packages
configure_dns_and_hosts
configure_time
join_ad_domain
configure_sssd_mkhomedir
configure_sudoers


# Run optional steps in a way that doesn't halt the script on failure
(install_dev_stack) || echo -e "${RED}ERROR during Development Stack installation. Check
log. Continuing script...${NC}"
(install_cockpit) || echo -e "${RED}ERROR during Cockpit installation. Check log.
Continuing script...${NC}"
(install_container_runtime) || echo -e "${RED}ERROR during Container Runtime installation.
Check log. Continuing script...${NC}"
(install_fail2ban) || echo -e "${RED}ERROR during Fail2ban installation. Check log.
Continuing script...${NC}"
(setup_tmux) || echo -e "${RED}ERROR during Tmux setup. Check log. Continuing
script...${NC}"
(setup_motd) || echo -e "${RED}ERROR during MOTD setup. Check log. Continuing
script...${NC}"


final_summary
system_updates_interactive


echo -e "${GREEN}=== Script finished at $(date) ===${NC}"
exit 0

```

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/sss/sss.conf
```

Input at the end of the file

```
ad_gpo_access_control = permissive
```

Your "/etc/sss/sss.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.

```
[sss]
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: %MYDNS\ ICT\ Staff\ SG 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



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 certificate 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 certificate 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 Chrome (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	<p>Google Chrome (Ubuntu)</p> <pre>sudo apt install curl software-properties-common apt-transport-https ca-certificates -y</pre> <pre>curl -fSsL https://dl.google.com/linux/linux_signing_key.pub gpg --dearmor sudo tee /usr/share/keyrings/google-chrome.gpg > /dev/null</pre> <pre>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</pre> <pre>sudo apt update</pre> <pre>sudo apt -y install google-chrome-stable</pre>
---	---

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*****