Enterprise-Grade Bash Script for Automated Secure Linux Server Provisioning with Monitoring & Logging

Keywords used: bash script, server automation, shell scripting, Linux hardening, cron job, logging, DevOps, CI/CD, monitoring, secure server, firewall, fail2ban, uptime, load average, security patch, SSH hardening, logrotate, systemd, SELinux, infrastructure automation, tech giant-level scripting, cloud deployment.

Objective

This Bash script securely provisions **a Linux server end to end automatically, hardens the security, installs monitoring, sets the firewall rules, enables automatic log rotation, schedules cron jobs** for reporting, and incorporates alert notification via mailx. Cloud-based infrastructure, CI/CD pipelines, and bare metal are appropriate uses.

---

provision_server.sh – Ultra Professional Bash Script

```
#!/bin/bash


#==============================#

#  Enterprise Linux Provisioner #

#  Author : Z. M. Sultana       #

#  Version: 1.0.0               #

#==============================#


set -euo pipefail

IFS=$'\n\t'
```

```
#==============================#
# Constants & Configurations   #
#==============================#

HOSTNAME="secure-node-$(hostname)"
LOG_FILE="/var/log/provision.log"
ADMIN_EMAIL="admin@example.com"
CRON_LOG="/var/log/cron_health.log"
SECURITY_PATCH_LOG="/var/log/patch_status.log"

# Colors for UI
GREEN='\033[0;32m'
RED='\033[0;31m'
NC='\033[0m' # No Color

#==============================#
# Logging Utility              #
#==============================#

log() {
    echo -e "$(date '+%Y-%m-%d %H:%M:%S') | $1" | tee -a "$LOG_FILE"
}

#==============================#
# Root Privilege Check         #
```

```
#==============================#

check_root() {
    if [[ "$EUID" -ne 0 ]]; then
        echo -e "${RED}ERROR: This script must be run as root.${NC}"
        exit 1
    fi
}

#==============================#
# Hostname Setup               #
#==============================#

setup_hostname() {
    log "Setting hostname to $HOSTNAME"
    hostnamectl set-hostname "$HOSTNAME"
}

#==============================#
# System Update & Patching     #
#==============================#

apply_security_patches() {
    log "Updating system and applying security patches..."
    apt-get update -y && apt-get upgrade -y
```

```
    apt-get install unattended-upgrades -y
    dpkg-reconfigure -plow unattended-upgrades
    echo "Security patches applied on $(date)" >> "$SECURITY_PATCH_LOG"
}

#================================#
# Essential Packages           #
#================================#

install_packages() {
    log "Installing essential tools..."
    apt-get install -y curl wget git ufw fail2ban mailutils net-tools htop logrotate
}

#================================#
# Firewall Setup               #
#================================#

setup_firewall() {
    log "Configuring UFW firewall..."
    ufw allow OpenSSH
    ufw allow http
    ufw allow https
    ufw enable
    ufw status verbose | tee -a "$LOG_FILE"
```

```
}

#==============================#
# Fail2Ban Setup               #
#==============================#

configure_fail2ban() {
    log "Setting up Fail2Ban for SSH brute force protection..."
    systemctl enable fail2ban
    systemctl start fail2ban
    fail2ban-client status sshd >> "$LOG_FILE"
}

#==============================#
# SSH Hardening                #
#==============================#

harden_ssh() {
    log "Hardening SSH configuration..."
    sed -i 's/^#Port 22/Port 2222/' /etc/ssh/sshd_config
    sed -i 's/^PermitRootLogin yes/PermitRootLogin no/' /etc/ssh/sshd_config
    sed -i 's/^#PasswordAuthentication yes/PasswordAuthentication no/' /etc/ssh/sshd_config
    systemctl restart sshd
    log "SSH now runs on port 2222 and root login is disabled."
}
```

```
#==============================#
# Log Rotation Configuration   #
#==============================#

setup_logrotate() {
    log "Configuring logrotate for custom logs..."
    cat <<EOF > /etc/logrotate.d/provision
$LOG_FILE {
    daily
    rotate 14
    compress
    missingok
    notifempty
}
EOF
}

#==============================#
# Cron Job for Health Report   #
#==============================#

setup_cron_health_monitor() {
    log "Creating cron job for system health monitoring..."
    cat <<'EOF' > /usr/local/bin/system_health_check.sh
```

```
#!/bin/bash
REPORT="/var/log/cron_health.log"
echo "===== System Health Report: $(date) =====" > $REPORT
echo "Uptime: $(uptime)" >> $REPORT
echo "Disk Usage:" >> $REPORT
df -h >> $REPORT
echo "Memory Usage:" >> $REPORT
free -h >> $REPORT
echo "Logged-in Users:" >> $REPORT
w >> $REPORT
mail -s "Daily Health Report - $(hostname)" admin@example.com < $REPORT
EOF
    chmod +x /usr/local/bin/system_health_check.sh
    echo "0 7 * * * root /usr/local/bin/system_health_check.sh" >> /etc/crontab
}

#==============================#
# SELinux Status (if exists)   #
#==============================#

check_selinux() {
    if command -v sestatus &> /dev/null; then
        log "Checking SELinux status..."
        sestatus | tee -a "$LOG_FILE"
    else
```

```
        log "SELinux not installed or not applicable on this system."
    fi
}

#==============================#
# Main Execution Block         #
#==============================#

main() {
    check_root
    log "========== Starting Server Provisioning =========="
    setup_hostname
    apply_security_patches
    install_packages
    setup_firewall
    configure_fail2ban
    harden_ssh
    setup_logrotate
    setup_cron_health_monitor
    check_selinux
    log "========== Provisioning Complete Successfully =========="
    echo -e "${GREEN}Server has been securely provisioned and automated.${NC}"
}

main "$@"
```

Advanced Concepts Covered

Topic	Included
Secure Shell (SSH) Hardening	Yes
UFW Firewall Rule Automation	Yes
Fail2Ban Configuration	Yes
Daily Cron Job Reports	Yes
Logrotate Integration	Yes
Email Alerts via Mailx	Yes
Systemd and Unattended-Upgrades	Yes
Root Permission Validation	Yes
Production-Ready Logging	Yes
Cloud/CI/CD Friendly	Yes