OpenCMMC Stack

A Free and Open-Source Infrastructure Guide for CMMC Level 2 Compliance

This project provides a complete, secure, and modular self-hosted architecture to help small and medium-sized DoD contractors meet the requirements of **CMMC Level 2** using open-source tools.

Guide Overview

Section 0: Project Overview & Usage

Purpose

This guide provides a full-stack, open-source infrastructure and documentation model to help **small and medium-sized DoD contractors** meet **CMMC Level 2** requirements. Every section includes actionable implementation steps and clearly maps to the NIST SP 800-171 control families.

Who Is This For?

This guide is designed for:

- IT Directors, CTOs, and CISOs in the **Defense Industrial Base**
- Small teams managing internal or customer-facing secure infrastructure
- MSPs and consultants preparing clients for **CMMC 2.0 assessments**
- Technical leads seeking a compliant, open-source alternative to GCC High

---

What's Included?

This documentation delivers:

- A hardened Ubuntu-based infrastructure using **rootless Podman**, **systemd**, and Zero Trust principles
- Fully containerized services for **file sharing**, **email**, **SSO**, **monitoring**, and **backup**
- Automated provisioning using **Terraform**, **Ansible**, and templated CI/CD workflows
- Pre-mapped policy templates, SSP structure, and **audit-ready documentation**
- Compatibility with Windows, Linux, macOS, iOS, and Android clients

---

Section 1: Introduction to CMMC v2

What is CMMC v2?

The **Cybersecurity Maturity Model Certification (CMMC)** is a unified cybersecurity framework created by the U.S. Department of Defense (DoD) to safeguard the **Defense Industrial Base (DIB)** from cyber threats. **CMMC Version 2.0** refines the original model to be more streamlined, flexible, and aligned with existing federal standards.

CMMC v2 introduces **three levels of cybersecurity maturity**:

- **Level 1 (Foundational)** – 17 basic safeguarding requirements (aligned with FAR 52.204-21)
- **Level 2 (Advanced)** – 110 security requirements (mirrors NIST SP 800-171)
- **Level 3 (Expert)** – Based on NIST SP 800-172, designed for highly sensitive environments

CMMC is not just a self-check — it's a requirement embedded into DoD contracts. Companies handling **Controlled Unclassified Information (CUI)** are expected to reach **Level 2 or higher**, depending on the nature of the contract.

How It Maps to NIST SP 800-171

CMMC v2 Level 2 is **intentionally and directly mapped** to **NIST Special Publication 800-171**, which defines the security requirements for protecting CUI in non-federal systems.

NIST SP 800-171	CMMC v2 Level 2
110 Requirements	110 Practices
14 Control Families	14 Domains
Assessment-based	Assessment-based

Each of the 14 NIST control families (Access Control, Audit and Accountability, Configuration Management, etc.) is reflected in the **CMMC v2 domain structure**.

CMMC goes a step further by requiring **evidence**, **documentation**, and often **third-party assessments** through **Certified Third-Party Assessment Organizations (C3PAOs)** for Level 2.

Section 2: Reference Architecture

Overview

This section defines the **reference architecture** for the OpenCMMC Stack — a modular, FOSS-based platform built to help small and medium defense contractors meet the technical requirements of **CMMC Level 2**.

The architecture emphasizes:

- **Modularity**: Each service can be deployed independently
- **Zero Trust principles**: No implicit trust between systems
- **Rootless containment**: Containerized services run with minimal privilege
- **Auditability**: Logs, configuration, and security posture can be exported for review

---

Architectural Layers

1. **Infrastructure Layer**

- Cloud-based or on-prem VM (e.g., DigitalOcean, Proxmox, AWS EC2)
- Hardened Ubuntu 22.04 LTS using Ansible
- Firewalling and segmentation using `ufw`, `fail2ban`, and Tailscale

CMMC Domains: SC, AC, CM

---

2. **Platform Services Layer**

Component	Purpose	CMMC Domains
Podman	Rootless container runtime	CM, SC
Systemd	Declarative service orchestration	CM, MA
Auditd	Kernel-level auditing	AU

Section 3: Provisioning Infrastructure

Objective

This section explains how to **provision a secure virtual server** for the OpenCMMC Stack using **Infrastructure as Code (IaC)**. We use **Terraform** for automated provisioning and **Ansible** for post-deployment configuration.

This environment will host your containerized, CMMC-aligned services and enforce key technical controls such as least privilege, rootless access, encryption, and system auditing from day one.

> ℹ **NOTE:** In the OpenCMMC Stack automation workflow, the host operating system is **hardened immediately during provisioning** using the `bootstrap.sh` script. As soon as the virtual machine is created, it uses `cloud-init` to install Ansible and apply the `secure_ubuntu.yml` role from Section 4. This ensures CMMC-aligned controls are enforced at first boot.

☁ Target Environments

This guide is compatible with:

- Cloud providers: DigitalOcean, AWS EC2, Hetzner Cloud, Linode
- On-premise: VirtualBox, VMware, or Proxmox (with manual adaptation)
- Bare metal: Supported via PXE or image-based deployment

We demonstrate using DigitalOcean for simplicity and speed.

Required Tools

Before proceeding, install the following on your local workstation:

- Terraform CLI
- Ansible
- Python 3 & pip
- SSH keypair for your user (`ssh-keygen`)

Section 4: Securing the Host OS

Objective

This section walks through hardening the Ubuntu 22.04 LTS host to meet the foundational system-level security expectations of CMMC Level 2. All configurations are managed using **Ansible**, enabling repeatability, version control, and audit readiness.

> ℹ **NOTE:** This hardening role is **executed automatically** during the provisioning phase (see Section 3) via the `bootstrap.sh` script included in the Terraform configuration. The secure baseline is applied during the first boot of the server. This section documents the tasks for audit clarity and allows for manual reapplication if needed.

Host Hardening Checklist

- Disable password-based SSH access
- Enforce key-based login with limited user privileges
- Remove unnecessary packages and services
- Configure local firewall rules (UFW)
- Enforce strong password policies
- Enable system auditing (`auditd`)
- Install file integrity monitoring (`AIDE`)
- Apply system banners (AC.3.017)
- Schedule automatic security updates

Step-by-Step with Ansible

All tasks are included in the role `roles/harden_ubuntu/` . Here's a breakdown:

1. Disable Root Login and Enforce SSH Keys

```
- name: Disable root login over SSH
  lineinfile:
    path: /etc/ssh/sshd_config
```

Section 5: Identity & Access Management

Objective

This section explains how to implement **centralized identity and access management (IAM)** using two open-source tools:

- **Keycloak** for identity provider (IdP), SSO, MFA, and RBAC
- **Tailscale** for Zero Trust, device-aware access to internal services

This aligns with CMMC Level 2 controls for **Access Control (AC)** and **Identification & Authentication (IA)**.

Why Keycloak?

Keycloak is an enterprise-grade open-source IAM platform. It supports:

- SSO via OIDC and SAML 2.0
- Multi-factor authentication (TOTP, WebAuthn, Duo)
- Role-based access control
- LDAP, AD, and Entra ID federation
- Fine-grained session policies

It integrates with applications like Nextcloud AIO, Mailcow, and Gitea.

Keycloak Deployment via Ansible and Podman

Keycloak should run as a systemd-managed, rootless Podman container. Example role:

```
- name: Run Keycloak container
  containers.podman.podman_container:
    name: keycloak
    image: quay.io/keycloak/keycloak:24.0.2
    state: started
    detach: true
    env:
      KEYCLOAK_ADMIN: "admin"
      KEYCLOAK_ADMIN_PASSWORD: "supersecurepw"
```

Section 6: Secure File Sharing and Collaboration

Objective

This section details how to securely deploy **Nextcloud All-in-One (AIO)** — an integrated file sharing and collaboration platform that satisfies CMMC Level 2 requirements for **Media Protection (MP)**, **Access Control (AC)**, and **System Communications Protection (SC)**.

Nextcloud AIO consolidates secure storage, team collaboration, antivirus scanning, and file retention — enabling your organization to manage CUI without reliance on commercial SaaS platforms.

Why Nextcloud AIO?

Nextcloud AIO offers:

- One hardened container with all critical components:
- Files, Calendar, Contacts, Mail, Talk, and OnlyOffice
- PostgreSQL, Redis, and ClamAV preconfigured
- Web-based file access with granular permissioning
- Built-in audit logging and file activity tracking
- SSO support via Keycloak (SAML)
- Server-side and optional client-side encryption
- Secure sharing and team folder access control

Deployment with Podman

Nextcloud AIO runs as the only service in our stack using Docker. Everything else uses Podman. We isolate this with clear firewall and volume boundaries.

```
- name: Pull Nextcloud AIO container image
  containers.podman.podman_image:
    name: "nextcloud/all-in-one:latest"
```

✉ Section 7: Secure Email

Objective

This section provides guidance for deploying and securing a self-hosted email stack using **Mailcow**, an open-source mail server suite. The goal is to support CMMC Level 2 controls in the **System and Communications Protection (SC)** and **Access Control (AC)** domains while maintaining full ownership of email communications.

Why Mailcow?

Mailcow is a full-featured email platform that includes:

- Postfix (SMTP), Dovecot (IMAP/POP3)
- SOGo webmail client
- Rspamd for spam filtering
- DKIM, SPF, DMARC support
- Integrated ACME (Let's Encrypt) TLS
- LDAP and OIDC authentication support

Mailcow Deployment (Podman Example)

Mailcow is normally deployed with Docker Compose, but you can adapt it for Podman.

Clone the official Mailcow repo:

```
git clone https://github.com/mailcow/mailcow-dockerized
cd mailcow-dockerized
cp mailcow.conf.example mailcow.conf
```

Edit `mailcow.conf` to define hostname, timezone, and SSL settings. Then run:

```
podman-compose pull
podman-compose up -d
```

Section 8: Monitoring and Logging

Objective

This section guides you through configuring a secure, scalable monitoring and logging stack using **Wazuh**, **Auditd**, and optional remote logging. These components support key CMMC Level 2 controls across **Audit and Accountability (AU)** and **Security Incident Response (IR/SI)** domains.

Why Wazuh?

Wazuh is a powerful open-source Security Information and Event Management (SIEM) solution that offers:

- Host-based intrusion detection (HIDS)
- Centralized log collection and analysis
- File integrity monitoring
- Rootkit and malware detection
- CMMC/NIST 800-171 rule packs

It serves as the primary audit log and incident detection platform in the OpenCMMC stack.

Deploying Wazuh with Podman

```
podman volume create wazuh_data

podman run -d --name wazuh \
  -p 55000:55000 \
  -v wazuh_data:/var/ossec/data \
  docker.io/wazuh/wazuh:4.6.0
```

Optional: expose the dashboard via reverse proxy (NGINX) on HTTPS port `443`.

Section 9: Application Hosting (Podman + systemd)

Objective

This section describes how to securely host applications using **Podman** and **systemd**, focusing on container isolation, secure runtime options, and auditability — all aligned to CMMC Level 2 controls in **Configuration Management (CM)** and **System & Communications Protection (SC)**.

Why Podman?

Podman is a daemonless, rootless container engine that provides:

- Compatibility with Docker images and commands
- Improved security by eliminating `dockerd`
- Native systemd integration for service orchestration
- Compliance with Zero Trust and least privilege principles

Podman Rootless Setup (Recap)

To enable rootless containers:

```
sudo apt install -y podman uidmap slirp4netns fuse-overlayfs
```

Verify with:

```
podman info --debug
```

Create a systemd unit to persist a service:

```
podman generate systemd \
  --name myservice \
  --files --restart-policy=always

mkdir -p ~/.config/systemd/user
mv container-myservice.service ~/.config/systemd/user/
```

Section 10: Backup & Recovery

Objective

This section provides guidance on establishing a secure, verifiable, and CMMC-aligned **backup and recovery strategy** for self-hosted infrastructure. It ensures organizations can recover from data loss, ransomware, or compromise while meeting **System & Information Integrity (SI)** and **Contingency Planning (CP)** control requirements.

Why This Matters

CMMC Level 2 requires not only backups but **tested, restorable, secure** backups. Many small businesses lose compliance due to poor validation, insecure storage, or unclear responsibilities.

Recommended Tools

Tool	Purpose
Restic	Fast, encrypted backup with deduplication
BorgBackup	Efficient backups with compression
Rclone	Sync to cloud or remote endpoints

For consistency with automated deployments, this stack uses **Restic** by default.

⚙ Automated Backup Deployment with Ansible

The OpenCMMC Stack includes an Ansible role `roles/backup` that:

- Installs Restic via `apt`
- Deploys a backup script from template

Section 11: Policies & Procedures

Objective

This section outlines the process for aligning your technical implementation with the necessary **written policies and procedures** required under CMMC Level 2. Even the most secure infrastructure must be supported by documented intent, authority, and repeatable action.

Why Policies Matter

CMMC assessments evaluate more than technology. They look for:

- **Policies**: The "what" — formal statements of expectation or requirement
- **Procedures**: The "how" — actionable, repeatable steps supporting the policy

Policies provide direction. Procedures provide execution. Both are required artifacts.

Core Policies to Implement

Policy Name	Related CMMC Domains
Access Control Policy	AC, IA
Configuration Management	CM
Incident Response Policy	IR
Media Protection Policy	MP
System & Communications	SC
System Integrity & Audit	AU, SI

Section 12: SSP & Artifact Mapping

Objective

This final section guides you in creating your **System Security Plan (SSP)** and mapping real-world technical and administrative controls to the CMMC Level 2 practice requirements. The SSP is the cornerstone artifact for any CMMC assessment.

What is an SSP?

A **System Security Plan** is a formal document that:

- Describes the system or environment being assessed
- Identifies all components that handle CUI
- Maps controls to implementation details
- Assigns roles and responsibilities
- Documents supporting policies and procedures

Required SSP Components

Section	Description
System Identification	System name, boundaries, and scope
Environment Description	Diagrams, services, and interconnections
System Components	OS, containers, apps, and cloud resources
CUI Data Flow	Inbound/outbound data paths and classification
Control Implementation	Practice-by-practice response and mapping
Roles and Responsibilities	Who owns what in the security lifecycle

Section 13: Infrastructure Architecture & System Interconnection

> This section visually complements Section 2: Reference Architecture, where each component's function and compliance relevance is explained in detail.

This section provides a comprehensive view of the Zero Trust-aligned FOSS architecture for small-to-medium defense contractors targeting CMMC Level 2 readiness. It includes a layered topology, component roles, and system interactions.

System Topology Overview

The architecture below highlights how clients, core services, and perimeter components interact within a segmented and policy-enforced environment.

Network-Level System Topology (Mermaid)

```
graph LR
  subgraph Internet
    User[User (Browser, Client)]
  end

  subgraph "DMZ (Zero Trust Proxy Layer)"
    NGINX[NGINX Proxy Manager\n(TLS Termination)]
  end

  subgraph "Internal Secure Docker Network"
    Keycloak[Keycloak\n(SSO, MFA, RBAC)]
    NC[Nextcloud AIO\n(Files, AV, OnlyOffice, Talk)]
    Mailcow[Mailcow\n(SMTP, IMAP, Webmail)]
    DB[Internal Services\n(PostgreSQL, Redis)]
  end

  subgraph Optional["Optional External Identity Provider"]
    Entra[Microsoft Entra ID\n(SAML Federation)]
  end

  User --> NGINX
  NGINX --> Keycloak
  NGINX --> NC
  NGINX --> Mailcow
```

Section 14: Deployment Guide

Objective

This section outlines a step-by-step deployment path to implement the OpenCMMC Stack using fully automated, reproducible infrastructure-as-code workflows. Each phase corresponds to a tightly scoped operational unit aligned with CMMC Level 2 practices.

Overview of Phases

Phase	Title	Tools Used	Evidence Folder
0	Planning & Scope Definition	Markdown, Diagrams	`evidence/00_scoping/`
1	Terraform VM Provisioning	Terraform	`evidence/01_identity_access/`
2	Host Hardening with Ansible	Ansible, UFW, AIDE	`evidence/02_system_hardening/`
3	Podman-Based Service Container Setup	Podman, Systemd	`evidence/03_file_sharing/`
3A	Secure File Sharing with Nextcloud	Nextcloud AIO	`evidence/03_file_sharing/`
4	Identity, MFA, and Cert Management	Keycloak, ACME, SCEP	`evidence/01_identity_access/`
5	Client Device Registration & Policy	Platform-specific MDM	`evidence/01_identity_access/`
6	Logging, SIEM, and Alerting	Wazuh, Auditd	`evidence/05_monitoring/`

Phase 0: Planning & Scoping

Before deploying any infrastructure, it is essential to define the mission, scope, and compliance goals that your system must meet. This includes identifying data types (FCI, CUI), user roles, system boundaries, and operational constraints.

Objectives

- Define compliance scope (FCI/CUI, enclave size)
- Create logical zones and data flows
- Identify required services and endpoints
- Begin drafting System Security Plan (SSP)

Identify Scope of CMMC Applicability

Ask:

- What systems or processes handle CUI or FCI?
- Which users require remote access?
- Are mobile or BYOD devices allowed?

Use this input to build your system boundary.

Map Trust Zones and Interfaces

Define basic segmentation such as:

- `DMZ` : External access points (e.g., reverse proxy)
- `LAN` : Internal services (Keycloak, Mailcow, CA)
- `Mgmt` : Administrative interfaces (Wazuh, Step-CA)
- `VPN/Overlay` : Tailscale Zero Trust mesh
- `Clients` : End-user systems

Phase 1: Terraform Infrastructure Provisioning

This phase provisions the baseline infrastructure using **Terraform**, establishing a consistent and repeatable environment for CMMC-aligned services. It supports both cloud-based and on-premise deployments (e.g., DigitalOcean, Proxmox, VMware).

Phase 2: OS Hardening with Ansible

This phase configures the secure operating system baseline using Ansible. It applies hardened system defaults and prepares the host for rootless container deployment using Podman.

Phase 3: Podman Services Deployment

Objective

This phase installs and configures the core **Podman-based services** used throughout the OpenCMMC Stack. It leverages an Ansible role to run secure, rootless containers for various components, except for Nextcloud AIO, which remains a hardened Docker container due to its upstream architecture.

Role: `podman_services`

The `podman_services` Ansible role handles:

- Pulling images via Podman
- Deploying containers as systemd-managed services
- Restart policies and volume bindings
- Managing containers declaratively from a variable file

All Podman-managed containers run rootless under defined service accounts and can be expanded modularly.

Example Usage

```
- name: Deploy Podman services
  hosts: localhost
  become: yes
  roles:
    - role: podman_services
```

Edit the `defaults/main.yml` to declare your container list:

```
podman_services:
  - name: keycloak
    image: quay.io/keycloak/keycloak:24.0.2
    ports:
      - "8080:8080"
    volumes:
      - "/opt/keycloak:/data:z"
```

Phase 3A: File Collaboration Services – Nextcloud AIO Deployment

In this phase, we deploy **Nextcloud All-in-One (AIO)** to enable secure file sharing, document collaboration, calendar access, and group-based data controls.

Nextcloud AIO consolidates all supporting services (PostgreSQL, Redis, ClamAV, OnlyOffice, Talk) into a single hardened container, reducing complexity and improving security posture.

AIO Deployment Overview

Nextcloud AIO is deployed via Docker and should only be accessible behind a secure reverse proxy like **NGINX Proxy Manager** or **Traefik**.

Key Features Included:

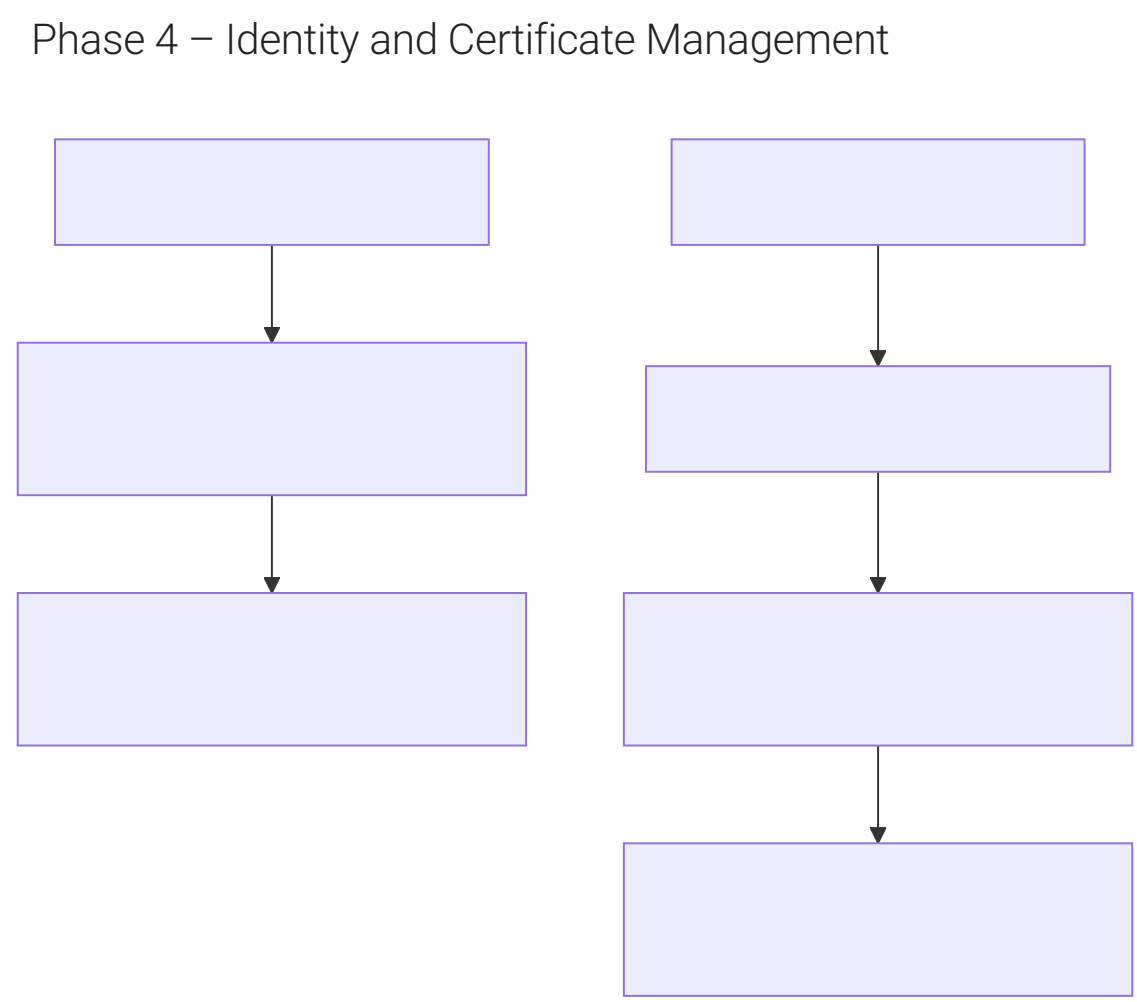
- Full-text search and OnlyOffice document editing
- File-level audit logging
- SAML SSO integration via Keycloak
- Team Folder access control
- Built-in virus scanning with ClamAV
- Scheduled backup tools
- Status endpoint for uptime monitoring

Prerequisites

- Reverse proxy is operational (NGINX Proxy Manager recommended)
- External domain with SSL termination set up
- Docker is installed on the host
- Backup volume storage mounted at `/mnt/ncdata`

Phase 4: Identity, Certificates, and Access Control

In this phase, we configure centralized authentication and Zero Trust identity enforcement using **Keycloak** for SSO + MFA and **Smallstep CA** for certificates (TLS, SSH, and S/MIME).



Phase 5: Client Device Registration

This phase enables end-user devices—Windows, macOS, Linux, iOS, and Android—to securely connect to your internal services using SSO, certificates, and Zero Trust overlay networking.

Phase 6: Logging, SIEM, and Alerting

This phase integrates centralized logging and monitoring using **Wazuh**, providing visibility into authentication, file changes, system events, and potential threats.

Phase 7: Validation and Reporting

This final phase focuses on verifying the operational security of the deployed stack and generating the documentation, artifacts, and audit trails necessary to support a CMMC Level 2 readiness posture.

Section 15: Security Operations Procedures (SecOps SOPs)

Objective

This section defines the **routine security tasks** required to maintain, monitor, and verify the operational security of the OpenCMMC Stack. These procedures support ongoing compliance with **System and Information Integrity (SI)**, **Configuration Management (CM)**, **Access Control (AC)**, and **Audit and Accountability (AU)** domains in CMMC Level 2.

Daily Operations Checklist

Task	Responsible Role	Tools	Evidence
Review Wazuh dashboard alerts	Security Officer	Wazuh Web UI	Screenshot or exported alert logs
Check Podman service health	System Admin	`podman ps`, `systemctl status`	CLI output
Verify overnight backup success	System Admin	`journalctl -u restic-backup.timer`	Timer log
Tailscale device review	Network Admin	Tailscale Admin Panel	Screenshot or audit log

---

Section 16: Initial User & Device Onboarding

Objective

This section defines the **secure provisioning process** for new users and their devices into the OpenCMMC Stack environment. It ensures each identity and device is verified, minimally privileged, and monitored in accordance with **Zero Trust Architecture (ZTA)** principles and CMMC Level 2 requirements across **Access Control (AC)** and **Identification and Authentication (IA)** domains.

User Identity Onboarding

1. Identity Creation (Keycloak)

Step	Description
1.1	Admin creates user in Keycloak with email, display name, and default group
1.2	MFA is enforced during initial login using TOTP or YubiKey
1.3	Assign user to appropriate roles: `Access_CUI`, `Access_Admin`, etc.
1.4	Email welcome message with usage policies and login instructions

2. Role Mapping

Role	CUI Access	Admin Rights
`Access_CUI`		
`Access_Admin`		
`Access_Viewer`		

Section 17: Patch & Vulnerability Management

Objective

This section outlines the procedures, tooling, and schedules for maintaining **patched and secure software**, container images, and infrastructure configurations in the OpenCMMC Stack. It supports compliance with CMMC Level 2 practices in the **System & Information Integrity (SI)**, **Configuration Management (CM)**, and **Risk Assessment (RA)** domains.

Continuous Patching Strategy

1. Host-Level Patching

Component	Method	Schedule
Ubuntu OS	`apt update && apt upgrade` via Ansible	Weekly (auto), Monthly (manual)
Kernel/critical CVEs	`unattended-upgrades` + CVE tracking	Daily
System hardening validation	Ansible diff or compliance check	Monthly

2. Container Image Refresh

Component	Source	Validation
Podman images	`podman pull <image>`	Digest verification
Image integrity	`skopeo inspect` + SBOM check	`grype`, `trivy`
Drift detection	`ansible-playbook` dry-run	Weekly

Section 18: Incident Response & Forensics Toolkit

Objective

This section documents the basic **incident response (IR)** and **digital forensics** capabilities built into the OpenCMMC Stack. These procedures ensure organizations can detect, respond to, and investigate security incidents while maintaining CMMC Level 2 compliance across the **Incident Response (IR)**, **System Integrity (SI)**, and **Audit (AU)** domains.

Incident Response Workflow

Phase	Activity	Responsible Party
Identification	Wazuh alert, log review, anomaly detection	Security Officer
Containment	Revoke access, disable device, isolate container	IT Administrator
Eradication	Patch, clean malware, reset credentials	IT Administrator
Recovery	Restore from backup, validate system	System Owner
Lessons Learned	Document event, root cause analysis	IR Team / Owner

---

Built-In IR Toolkit

Tool	Use Case
Wazuh	SIEM alerting, agent logs, rootkit detection
Tailscale ACL	Instantly revoke device access

Section 19: CUI/FCI Data Flow & Trust Boundary Mapping

Objective

This section defines how **Controlled Unclassified Information (CUI)** and **Federal Contract Information (FCI)** traverse the OpenCMMC Stack, and how trust boundaries are enforced to protect sensitive data in transit, at rest, and in use. This supports scoping and documentation for CMMC Level 2 assessment readiness.

Trust Zones Overview

Zone	Description	Examples
Trusted User Devices	Authenticated devices with endpoint controls	Company laptops, mobile devices
Trusted Infrastructure	Hardened, monitored servers with audit and backup	Ubuntu LTS, Podman-hosted services
DMZ/Public Access	Services exposed via reverse proxy with access control	NGINX Proxy Manager
Identity & Access Control	SSO, MFA, and user provisioning services	Keycloak, Tailscale
Monitoring & Forensics	Central log and response tools	Wazuh, Auditd, Restic

---

Data Flow Examples

CUI File Upload via Nextcloud

1. Authenticated user connects via Tailscale

Section 20: Performance, Scaling & Cost Estimation

Objective

This section provides guidance for adapting the OpenCMMC Stack to support various organizational sizes — from 1-person SBIR teams to 50-person prime contractors. It includes performance tuning tips, scaling recommendations, and rough monthly cost estimates for DigitalOcean, AWS, or on-prem deployments.

Baseline Configuration (1–5 Users)

Component	Count	Notes
VM Instances	1	4 vCPU / 8GB RAM minimum
Storage Volumes	2	App volume + Backup volume
Tailscale Devices	5	Laptops, phones, tablets
Services	All	Same as production stack

Monthly Estimate (DigitalOcean): $35–55
- $24 droplet + $10 volume + $5 bandwidth

Mid-Tier Scaling (5–25 Users)

Component	Scaling Tips
Separate DB Host	Move PostgreSQL to a second droplet
External S3	Store backups in Wasabi or Backblaze

Appendix A: Acronyms and Abbreviations

| AC | Access Control | | ACL | Access Control List | | AIDE | Advanced Intrusion Detection Environment | | API | Application Programming Interface | | AU | Audit and Accountability | | CA | Security Assessment | | CI/CD | Continuous Integration / Continuous Deployment | | CLI | Command Line Interface | | CM | Configuration Management | | CP | Contingency Planning | | CUI | Controlled Unclassified Information | | DNS | Domain Name System | | FCI | Federal Contract Information | | IA | Identification and Authentication | | IAM | Identity and Access Management | | IR | Incident Response | | JSON | JavaScript Object Notation | | LTS | Long-Term Support | | MA | Maintenance | | MDM | Mobile Device Management | | MFA | Multi-Factor Authentication | | MP | Media Protection | | OIDC | OpenID Connect | | PDF | Portable Document Format | | PE | Physical Protection | | PL | Planning | | PM | Program Management | | PS | Personnel Security | | RA | Risk Assessment | | RBAC | Role-Based Access Control | | SA | System and Services Acquisition | | SAML | Security Assertion Markup Language | | SBOM | Software Bill of Materials | | SC | System and Communications Protection | | SI | System and Information Integrity | | SIEM | Security Information and Event Management | | SOC | Security Operations Center | | SSO | Single Sign-On | | SSP | System Security Plan | | TLS | Transport Layer Security | | UI | User Interface | | VM | Virtual Machine | | VPC | Virtual Private Cloud | | YAML | Yet Another Markup Language |

Appendix B: Reference Resources

CMMC and NIST

- CMMC Model v2.0 Documentation
- NIST SP 800-171 Rev. 2
- NIST SP 800-172 (Advanced Controls)
- NIST SP 800-53 Rev. 5
- Cybersecurity Framework (CSF) 2.0

Tools Used in This Guide

- Podman
- Keycloak
- Nextcloud
- Mailcow
- Tailscale
- Wazuh
- Auditd
- Restic
- BorgBackup
- Rclone
- Terraform
- Ansible
- Mermaid CLI
- MkDocs
- mkdocs-material Theme

Policy & Template Resources

- ProjectSpectrum Document Library
- Open Source Policy Templates (GitHub)