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 Linux-based infrastructure using **rootless Podman** and **systemd**
- Fully containerized services for **file sharing**, **email**, **SSO**, and **logging**
- Automated provisioning with **Terraform** and **Ansible**
- Written policy templates and **audit-ready artifacts**
- CMMC control mappings and SSP guidance

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.

☁ 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'll 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.

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
    regexp: '^PermitRootLogin'
    line: 'PermitRootLogin no'

- name: Disable password authentication
  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, Gitea, and more.

Keycloak Deployment (Secure Container)

```
docker run -d --name keycloak \
  -p 8080:8080 \
  -e KEYCLOAK_ADMIN=admin \
  -e KEYCLOAK_ADMIN_PASSWORD=supersecurepw \
  quay.io/keycloak/keycloak:24.0.2 \
  start --optimized
```

> In production, place behind a reverse proxy and run as a systemd-managed container with secure TLS.

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 via Docker

Run this container behind a reverse proxy (e.g., NGINX Proxy Manager):

```
docker run -d \\
  --name nextcloud-aio-mastercontainer \\
  --restart always \\
  -p 8080:8080 \\
  -v nextcloud_aio_mastercontainer:/mnt/docker-aio-config \\
```

✉ 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
Duplicity	GPG-encrypted full/diff backups

For simplicity and reliability, we'll use **Restic**.

Setting Up Restic

Install Restic:

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 – OpenCMMC FOSS Infrastructure

This deployment guide provides a step-by-step, modular walkthrough for provisioning and configuring a fully CMMC Level 2–aligned FOSS infrastructure stack for small to medium defense contractors.

Technologies Used:

- Terraform for infrastructure provisioning
- Ansible for operating system hardening and service configuration
- Podman for secure, rootless container deployment
- Keycloak, Smallstep CA, Tailscale, Mailcow, Nextcloud, Wazuh

The end result is a fully operational, Zero Trust–compliant environment accessible securely by:

- Windows, macOS, Linux endpoints
- Android and iOS mobile devices
- Remote and on-prem users

Deployment Phases

Each phase maps directly to key CMMC practices across Access Control (AC), Configuration Management (CM), System Integrity (SI), and Audit & Accountability (AU).

Phase	Focus
Phase 0	Planning and scoping CMMC-aligned infrastructure
Phase 1	Provisioning secure hosts using Terraform
Phase 2	Operating system hardening using Ansible
Phase 3	Secure application services deployed with Podman

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 Service Deployment

In this phase, we deploy core application services using **Podman** — a rootless, secure container runtime. Podman replaces Docker and integrates cleanly with systemd for persistent service management.

> **Note:** As of this phase, Nextcloud is deployed via a standalone hardened container using the **Nextcloud All-in-One (AIO)** model. It is **not** managed via `podman-compose`, but follows secure Docker deployment practices described in Phase 4: File Collaboration Services.

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:

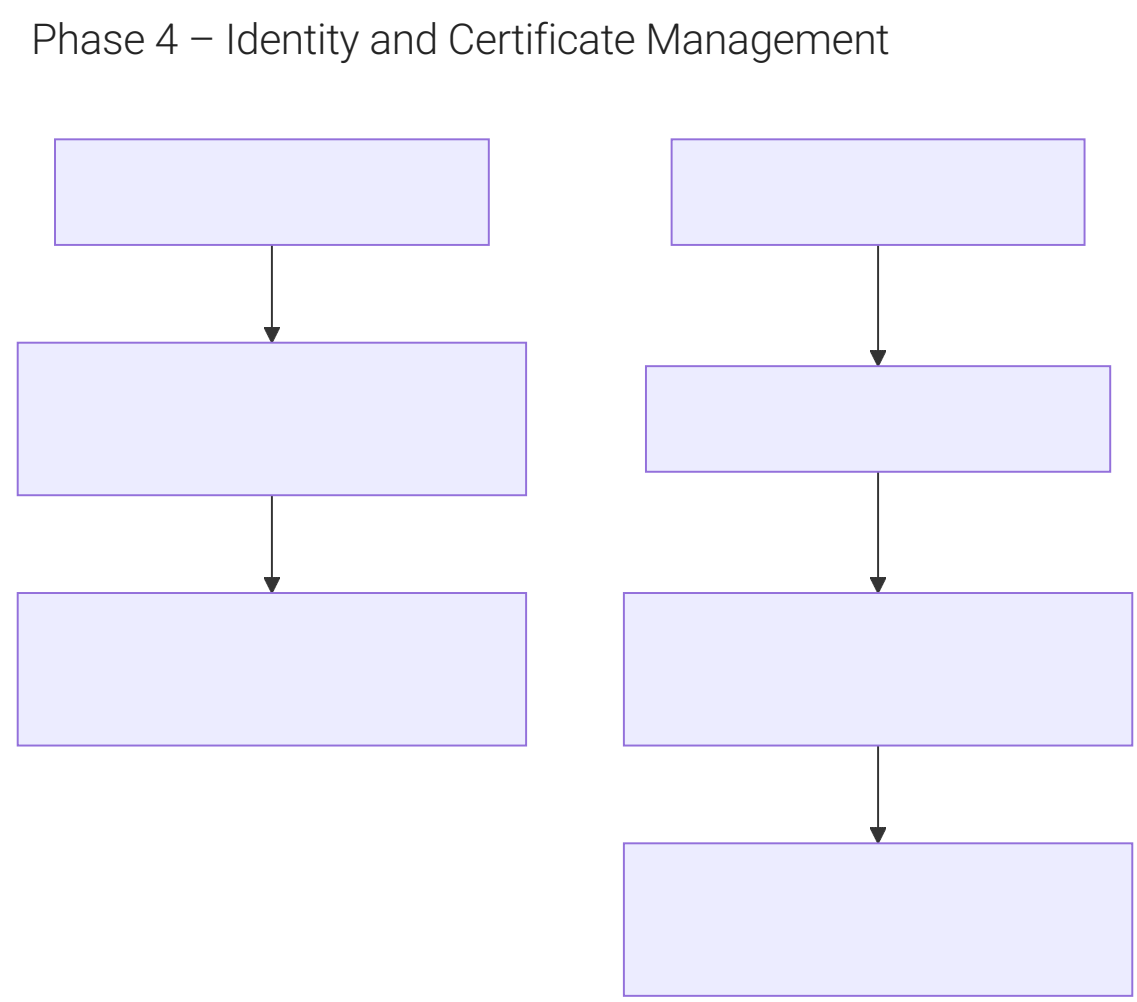
- Full-text search and OnlyOffice document editing
- File-level audit logging
- SAML SSO integration via Keycloak
- Team Folder access control
- Built-in virus scanning with ClamAV
- Scheduled backup tools
- Status endpoint for uptime monitoring

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.

Appendix A: Acronyms and Abbreviations

Acronym	Meaning
AC	Access Control
AU	Audit and Accountability
CA	Security Assessment
CM	Configuration Management
CP	Contingency Planning
IA	Identification and Authentication
IR	Incident Response
MA	Maintenance
MP	Media Protection
PE	Physical Protection
PL	Planning
PM	Program Management
PS	Personnel Security
RA	Risk Assessment
SA	System and Services Acquisition
SC	System and Communications Protection

Appendix B: Reference Resources

CMMC and NIST

- CMMC Model v2.0 Documentation
- NIST SP 800-171 Rev. 2
- NIST SP 800-172 (Advanced Controls)

Tools Used in This Guide

- Podman
- Keycloak
- Nextcloud
- Mailcow
- Tailscale
- Wazuh
- Auditd
- Restic
- Terraform
- Ansible

Policy & Template Resources

- ProjectSpectrum Document Library
- Open Source Policy Templates (GitHub)
- CMMC-AB Resources