COURSE SYLLABUS

SL 4O

USAREUR-AF

COURSE SYLLABUS — SL 4O: PLATFORM ENGINEER

Maven Smart System (MSS) — USAREUR-AF

HEADQUARTERS
UNITED STATES ARMY EUROPE AND AFRICA
(USAREUR-AF)
Wiesbaden, Germany

DRAFT — NOT FOR OFFICIAL USE. FOR TRAINING PLANNING PURPOSES ONLY.

26 MARCH 2026

DRAFT — UNOFFICIAL — NOT FOR OPERATIONAL USE

COURSE SYLLABUS — SL 4O: PLATFORM ENGINEER

MAVEN SMART SYSTEM (MSS) — USAREUR-AF

Field	Detail
Level	SL 4O — Platform Engineer Specialist Track
Duration	5 days (40 hours)
Prerequisites	SL 1, SL 2, SL 3 (all Go evaluations on file — **REQUIRED**, not recommended); Linux systems administration proficiency (intermediate or higher); familiarity with containers (Docker) and version control (Git)
Audience	Platform engineers responsible for infrastructure, CI/CD, container security, and deployment operations on MSS
Format	Instructor-led lecture + lab + practical exercise
Location	MSS Training Environment (Kubernetes cluster access required; CLI tools: kubectl, helm, git)

BLUF: SL 4O teaches platform engineers to build, operate, and secure the infrastructure layer that MSS applications run on — Kubernetes cluster management, CI/CD pipeline design, container security and hardening, GitOps workflows, and air-gapped deployment. SL 4O is for engineers who build the platform that application developers (SL 4L) build on. If the requirement is application development, not infrastructure, SL 4L is the correct track.

LEARNING OBJECTIVES

#	Objective
1	Describe the Internal Developer Platform (IDP) concept and identify the platform services required to support MSS application teams
2	Deploy a workload to a Kubernetes cluster using declarative manifests with resource requests, limits, and health checks

#	Objective
3	Implement namespace-level isolation with resource quotas, network policies, and RBAC
4	Build a GitOps workflow where Git is the single source of truth for cluster state, with drift detection and automated reconciliation
5	Harden a container image: start from an Iron Bank base, apply multi-stage build, run as non-root, scan for vulnerabilities, pin by SHA256 digest
6	Design and implement a CI/CD pipeline with security gates at every stage (secrets detection, SCA, SAST, image scan, policy admission)
7	Execute a deployment using rolling update and blue/green strategies, including rollback procedures
8	Package an application for air-gapped deployment: bundle container images, configuration, and dependencies for transfer across an air gap
9	Configure platform monitoring with health checks, resource utilization dashboards, and actionable alerting

PRE-COURSE CHECKLIST

Complete **10+ duty days before Day 1:**

- ☐ Request **Kubernetes cluster access** from C2DAO — training cluster with namespace-admin privileges; distinct from standard MSS Builder access
- ☐ Configure your workstation: kubectl, helm, git, and a container runtime (Docker or Podman); bring your configured laptop
- ☐ Read TM-40O, Chapter 1 (Introduction, platform engineer role, the platform-as-product concept) — 20 min
- ☐ Read TM-40O, Chapter 8 (Platform Security and Compliance) — security requirements affect every infrastructure decision; read before the labs, not after
- ☐ Complete the Linux prerequisites self-assessment (provided in training package): file permissions, process management, networking basics, systemd

DAILY SCHEDULE

Day 1 — Platform Fundamentals and Kubernetes

Time	Block	Method	Content
0800–0900	1	Lecture	Platform Engineer role on MSS; platform-as-product mindset; IDP concept; DevEx principles
0900–1100	2	Lab	Kubernetes fundamentals: cluster architecture, kubectl operations, deploying a pod, viewing logs and events
1100–1115	—	Break	
1115–1200	3	Lab	Kubernetes: Deployments, Services, and ConfigMaps — declarative workload management
1200–1300	—	Lunch	
1300–1500	4	Lab	Kubernetes: resource requests and limits, health checks (liveness, readiness, startup probes), pod lifecycle
1500–1515	—	Break	
1515–1700	5	Lab	Kubernetes: namespaces, resource quotas, LimitRanges — isolation and resource governance

Evening reading: TM-40O, Chapter 3 (Kubernetes for MSS) — full chapter; the labs build on this foundation.

Day 2 — Infrastructure as Code and GitOps

Time	Block	Method	Content
0800–0830	—	Review	Day 1 questions; Kubernetes checkpoint — confirm all trainees can deploy, expose, and manage a workload
0830–1030	6	Lecture + Lab	IaC principles: declarative configuration, idempotency, environment parity; configuration templating with ytt/Helm
1030–1045	—	Break	

Time	Block	Method	Content
1045–1200	7	Lab	GitOps setup: configure a GitOps controller; deploy an application by committing to Git; observe reconciliation
1200–1300	—	Lunch	
1300–1500	8	Lab	Drift detection: manually modify cluster state; observe GitOps controller revert the change; configure drift alerts
1500–1515	—	Break	
1515–1700	9	Lab	Network policies: default deny, explicit allow rules; test pod-to-pod and pod-to-external communication controls

Evening reading: TM-40O, Chapter 4 (Infrastructure as Code) — full chapter; focus on GitOps workflow and templating.

Day 3 — Container Security and CI/CD

Time	Block	Method	Content
0800–0830	—	Review	Day 2 questions; GitOps checkpoint — confirm all trainees have a working GitOps workflow
0830–1030	10	Lecture + Lab	Container hardening: Iron Bank base images, multi-stage builds, non-root execution, capability dropping
1030–1045	—	Break	
1045–1200	11	Lab	Container scanning: vulnerability scan, digest pinning, image signing; block deployment of unscanned images
1200–1300	—	Lunch	
1300–1500	12	Lecture + Lab	CI/CD pipeline design: pipeline stages, security gates, artifact management; build a pipeline from source to deployment
1500–1515	—	Break	
1515–1700	13	Lab	CI/CD security gates: add secrets detection, SCA, SAST, and image scanning gates to the pipeline; test gate blocking

Evening reading: TM-40O, Chapter 5 (Container Security) and Chapter 6 (CI/CD Pipeline Design) — focus on security gates.

Day 4 — Deployment Strategies and Air-Gapped Operations

Time	Block	Method	Content
0800–0830	—	Review	Day 3 questions; pipeline checkpoint — confirm all trainees have a working pipeline with security gates
0830–1030	14	Lecture + Lab	Deployment strategies: rolling update, blue/green, canary; implement each strategy and practice rollback
1030–1045	—	Break	
1045–1200	15	Lab	Monitoring and alerting: configure health dashboards, resource utilization metrics, and actionable alerts
1200–1300	—	Lunch	
1300–1500	16	Lecture + Lab	Air-gapped deployment: bundle creation (imgpkg), internal registry setup, dependency mirroring, certificate management
1500–1515	—	Break	
1515–1700	17	Lab	Air-gapped exercise: deploy an application to a simulated air-gapped cluster using bundled artifacts only — no external network access

Evening reading: TM-40O, Chapter 7 (Air-Gapped and DDIL Operations) — full chapter.

Day 5 — Platform Security, Compliance, and Practical Exercise

Time	Block	Method	Content
0800–0900	18	Lecture	Platform security: RMF/ATO from infrastructure perspective, STIG compliance, access control, audit logging, break-glass procedures
0900–1100	19	Exercise	**EX_40O Practical Exercise** begins — build and secure a platform environment from cluster to deployment (see EX_40O for full task list)
1100–1115	—	Break	

Time	Block	Method	Content
1115–1200	20	Exercise	EX_40O continued
1200–1300	—	Lunch	
1300–1500	21	Exercise	EX_40O continued — air-gapped deployment and security validation
1500–1515	—	Break	
1515–1630	22	Presentation	Platform demonstrations: each trainee presents their environment, pipeline, and security posture
1630–1700	23	Evaluation	Post-test (EXAM_TM40O_POST); course evaluation; SL 5O pathway discussion

POST-COURSE

- **Evaluation:** EXAM_TM40O_POST administered Day 5 (see schedule)
- **Proficiency exercise:** EX_40O evaluated against Go/No-Go criteria
- **Continuation:** SL 5O — Advanced Platform Engineer (fleet management, SRE, RMF automation, DevEx engineering)
- **Cross-track recommendation:** Attend SL 4L (Software Engineer) overview session to understand the application developer's perspective on platform services; review TM-50L Chapter 6 (DevSecOps) for the application-side view of CI/CD