LESSON PLANS

SL 4

USAREUR-AF

LESSON PLAN OUTLINES — SL 4 SPECIALIST TRACKS

USAREUR-AF Operational Data Team — C2DAO

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

26 MARCH 2026

DRAFT — UNOFFICIAL — NOT FOR OPERATIONAL USE

LESSON PLAN OUTLINES — SL 4 SPECIALIST TRACKS

USAREUR-AF OPERATIONAL DATA TEAM — C2DAO

Covers: SL 4G (ORSA) | SL 4H (AI Engineer) | SL 4M (MLE) | SL 4J (PM) | SL 4K (KM) | SL 4L (SWE) | SL 4N (UX Designer) | SL 4O (Platform Engineer) **Version:** 1.2 — March 2026 (updated: specialist tracks renumbered G–O; WFF tracks are SL 4A–F, prereq SL 3)

Abbreviated LP outlines for SL 4 specialist courses. Instructors at SL 4 level must have deep domain SME background — these outlines supplement SME knowledge, not replace it. Expand using `LP_TEMPLATE.md` as needed.

PART G — SL 4G: ORSA SPECIALIST

Duration: 5 days (40 hours) | **T:I ratio:** 4:1 | **Instructor req:** FA49 or equivalent; SL 4G certified or C2DAO ORSA SME

Block 1 — ORSA Role on MSS and Analytical Product Standards

Hours: 1.0 | **Method:** Brief | **Day:** 1 | **Time:** 0800–0900

Purpose: Establishes what it means to be an ORSA analyst on MSS and what the non-negotiable product standards are. Product standards (uncertainty communication, documentation, reproducibility) must be established before any modeling work.

TLO: The trainee will describe the ORSA analyst role on MSS and state the three non-negotiable product standards: uncertainty characterization, documented assumptions, and reproducibility.

Key Delivery Notes: - The ORSA role on MSS is not about building dashboards — it is about building models and delivering commander decision products. Everything else is infrastructure. - Non-negotiable standards: (1) Every estimate has a confidence interval. (2) Every model has documented assumptions. (3) All simulations are reproducible (seed set). - Foundry data model review: ORSA accesses data through Code Workspace, not Pipeline Builder.

Assessment: Evaluator will ask about each standard during the practical exercise brief.

New doctrine content (March 2026): MOE/MOP assessment framework for analytical product evaluation, CARVER scoring methodology for target/risk prioritization, and force ratio computation standards. Instructor should reference these in Blocks 1, 13, and 16 (wargame data architecture).

Block 2 — Code Workspace Setup

Hours: 2.0 | **Method:** Lab | **Day:** 1 | **Time:** 0900–1100

Purpose: A broken workspace on Day 4 costs the trainee their modeling time. Set up and verify the workspace on Day 1, not later.

TLO: The trainee will configure a Python/R Code Workspace with required packages, verify Foundry dataset connectivity via Spark/pandas read, and successfully execute a test write transaction to a curated output dataset.

Key Delivery Notes: - Package installation: `pip install statsmodels scipy pandas numpy matplotlib seaborn` (Python) or `install.packages(c("forecast", "lpSolve"))` (R). Foundry workspaces may not persist installed packages across session restarts — build an install cell at the top of

every notebook. - Verify connectivity: `spark.read.table("ri.foundry.main.dataset.[RID]")` or pandas via the Foundry API. Confirm schema appears. - Write transaction test: `transaction = dataset.start_transaction()` → write → `transaction.commit()`. This must work before any model outputs are written. Test it today.

Assessment: Evaluator confirms workspace is configured at start of Practical Exercise (Day 5).

Block 3 — Foundry Dataset Connectivity

Hours: 0.75 | **Method:** Lab | **Day:** 1 | **Time:** 1115–1200

Key Delivery Notes: - Three access patterns: Spark DataFrame (large datasets), pandas (smaller datasets via `.toPandas()`), direct Foundry REST API. For SL 4G, Spark → pandas conversion is the primary pattern. - Schema inspection: `df.printSchema()` or `df.dtypes`. Confirm column names and types match the Ontology definition.

Block 4 — Writing Outputs to Foundry

Hours: 2.0 | **Method:** Lab | **Day:** 1 | **Time:** 1300–1500

Purpose: Model outputs that don't return to Foundry can't be integrated with Ontology visualizations or downstream applications. The write transaction pattern must be mastered on Day 1.

Key Delivery Notes: - Transaction pattern: `with dataset.transaction() as t: t.write(df)`. The transaction must commit — an uncommitted transaction leaves no data. - Output schema: define the output schema explicitly before writing. Column names must follow C2DAO naming convention. Output dataset goes in the project's Datasets folder. - Test: after writing, open the output dataset in Foundry and confirm row count, column names, and a sample of values.

Block 5 — Data Profiling in Code Workspace

Hours: 1.75 | **Method:** Lab | **Day:** 1 | **Time:** 1515–1700

Key Delivery Notes: - Null analysis: `df.isnull().sum() / len(df)`. Columns with >30% nulls require an explicit handling decision before modeling. - Outlier detection: IQR or z-score. Flag outliers before modeling — do not silently include them. - Feature distributions: histograms for numerics, value counts for categoricals. This is the "know your data" step before any model.

Block 6 — Regression Modeling

Hours: 2.0 | **Method:** Lab | **Day:** 2 | **Time:** 0830–1030

Purpose: Regression is the foundational ORSA tool on MSS. The standard is not just "fit a model" — it is "fit, validate, and document a model that an FA49 evaluator can reproduce."

TLO: The trainee will build a linear regression model, run residual analysis and cross-validation, document assumptions and validation statistics, and write output to Foundry.

Key Delivery Notes: - Feature selection rationale must be documented — not just "I chose these because the model fit well." - Validation statistics required: R^2, RMSE, MAE, residual plot inspection, VIF for multicollinearity. - Set `random_state` before train/test split and before any stochastic operation. Reproducibility standard applies here too.

Assessment: Evaluated in Practical Exercise Tasks 2 and 6.

Block 7 — Classification Models

Hours: 1.25 | **Method:** Lab | **Day:** 2 | **Time:** 1045–1200

Key Delivery Notes: - Logistic regression and decision tree. Calibration check is required — Army ORSA products that present probabilities must be calibrated. - Cross-validation: at minimum k=5 fold. Report mean and std of validation metric. - Calibration: `sklearn.calibration.calibration_curve`. If the curve deviates substantially from the diagonal, apply Platt scaling or isotonic regression.

Block 8 — Model Validation Standards

Hours: 2.0 | **Method:** Lab | **Day:** 2 | **Time:** 1300–1500

Key Delivery Notes: - Residual analysis for regression: residuals vs. fitted plot, QQ plot for normality. Flag if heteroscedasticity or non-normality is present — document the finding. - USAREUR-AF validation standard: every model must have a documented model validation section before it is presented to a commander.

Block 9 — Practice Build (Regression to Foundry Output)

Hours: 1.75 | **Method:** Lab | **Day:** 2 | **Time:** 1515–1700

Key Delivery Notes: Second dataset, full cycle: profile → regression → validate → write to Foundry → build Quiver visualization of output. No instructor assistance.

Block 10 — Time Series: Stationarity and Model Identification

Hours: 2.0 | **Method:** Lab | **Day:** 3 | **Time:** 0830–1030

TLO: The trainee will test a time series for stationarity, plot ACF/PACF, identify plausible ARIMA order parameters, and document the model selection rationale.

Key Delivery Notes: - ADF test for stationarity. If non-stationary: first difference and retest. - ACF/PACF plots: trainees must explain what they see in the plots, not just run the code. "I see a cutoff at lag 2 in the PACF, suggesting AR(2)" — this is the level of explanation the evaluator expects. - Document the selection: "Selected ARIMA(2,1,1) based on: ADF test requiring 1 difference (d=1), PACF cutoff at 2 suggesting p=2, ACF gradual decay suggesting q≤1."

Block 11 — ARIMA/SARIMA Build

Hours: 1.25 | **Method:** Lab | **Day:** 3 | **Time:** 1045–1200

Key Delivery Notes: - 90% confidence intervals are required — not optional. The evaluator will look for them. - Out-of-sample forecast: at minimum 6 periods forward. Plot with historical data and confidence bounds.

Block 12 — Monte Carlo Simulation

Hours: 2.0 | **Method:** Lab | **Day:** 3 | **Time:** 1300–1500

TLO: Given a COA comparison scenario, the trainee will build a Monte Carlo simulation with ≥1,000 trials, distribution selection rationale, seed set for reproducibility, and probability at defined operational thresholds.

Key Delivery Notes: - Minimum 1,000 trials. 100 trials produce unstable distributions — the evaluator knows this and will test for it. - Distribution selection: `np.random.normal()`, `np.random.triangular()`, `np.random.uniform()` — justify the choice from the data or from operational knowledge. - Probability at threshold: "probability of <80% readiness at D+30" — count trials below threshold / total trials. - Set `np.random.seed()` before any stochastic operations.

Block 13 — Sensitivity Analysis and Logistics Risk

Hours: 1.75 | **Method:** Lab | **Day:** 3 | **Time:** 1515–1700

Key Delivery Notes: - Sensitivity analysis: vary each input parameter ±10% and observe output change. Identify which inputs drive the most output variance. - Logistics stockage risk: Poisson distribution for demand arrivals is a common model. Show the application to Class IX stockage.

Blocks 14–17 — Optimization, Wargame Data, Quiver/Contour

Hours: 7.5 total | **Days:** Day 4

Block 14 (LP): `scipy.optimize.linprog` for resource allocation. Document constraints explicitly. Sensitivity analysis on binding constraints.

Block 15 (Scheduling Optimization): Formulate maintenance scheduling as LP or ILP. Operational constraints from the mission calendar.

Block 16 (Wargame Data Architecture): Collection template design. Aggregation pipeline for exercise data. Post-exercise analysis pipeline.

Block 17 (Quiver/Contour for ORSA): Forecast dashboard with uncertainty bounds. COA comparison visualization. The uncertainty bounds must be visible — not buried.

Blocks 18–21 — Commander Brief Standards and Practical Exercise

Day: 5

Block 18–19 (Communicating Uncertainty): The brief posture is the hardest skill. Translate quantitative output to operational language. Three prohibitions: (1) point estimates without bounds, (2) unqualified predictions ("will"), (3) methods-paper language.

Block 20 (Scenario Brief): Provide scenario. 45 minutes planning time before build.

Block 21 (Practical Exercise): 4-hour build: regression + time series forecast + commander brief from provided readiness dataset.

Go standard: Pass 5 of 6 tasks. Commander brief reviewed by FA49 evaluator. Brief without uncertainty characterization fails that element — hard standard.

PART H — SL 4H: AI ENGINEER

Duration: 5 days (40 hours) | **T:I ratio:** 4:1 | **Instructor req:** AIP Logic authoring experience; C2DAO AI SME designation

MANDATORY: Block 1 (AI Safety Seminar, 2.0 hours) cannot be skipped, rescheduled, or made up independently.

Block 1 — AI Safety Seminar (MANDATORY)

Hours: 2.0 | **Method:** Seminar | **Day:** 1 | **Time:** 0800–1000

Purpose: The most consequential block in the course. AI systems that produce incorrect outputs affect operational decisions. Trainees who treat this as a compliance checkbox build workflows that fail in production in ways that matter.

TLO: The trainee will complete an AIP Authorization Checklist for a sample workflow, identify at least 5 prohibited use cases by category, and describe the human-in-the-loop requirement for Ontology write operations.

Key Delivery Notes: - Cover the Army CIO AI policy (April 2024) — prohibited use categories, human oversight requirements, OPSEC implications of AI outputs. - Operational context: AI outputs may be used in briefings, decision products, and Ontology records that commanders rely on. An incorrect extraction in a SITREP summary is a readiness reporting error. - Human-in-the-loop: NO AIP Logic workflow may write to production Ontology Objects without a tested human checkpoint. This is a non-negotiable design requirement, not a policy preference. - Walk through TM-40H Appendix B (Prohibited Use Cases) line by line.

Assessment: AIP Authorization Checklist completion is evaluated in Practical Exercise Task 6.

New doctrine content (March 2026): ADP 3-13 AI/ML framing for operational AI products, PED-to-pipeline mapping for intelligence-AI integration, and UDRA governance framework for AI data access. Instructor should reference these in Blocks 1, 5–8 (Ontology write integration), and 16 (AI Output Validation Framework).

Block 2 — AIP Platform Architecture

Hours: 1.75 | **Method:** Lecture | **Day:** 1 | **Time:** 1015–1200

Key Delivery Notes: - Full stack: AIP Logic (workflow authoring), Agent Studio (agent orchestration), Code Workspaces (Python transforms), LLM endpoints (model API). - AIP Logic is the workflow layer — it orchestrates the other components. Understanding the architecture prevents the common error of trying to do everything in the prompt.

Blocks 3–4 — AIP Logic: First Workflow and Conditional Chains

Hours: 3.75 | **Method:** Lab | **Day:** 1 | **Time:** 1300–1700

Block 3 TLO: Author a first AIP Logic workflow with a prompt template, input configuration, output binding, and run a test — observing the structured JSON output.

Block 4 TLO: Build conditional chains with looping and error handling — workflows that behave differently based on intermediate outputs.

Key Delivery Notes (combined): - Prompt template: explicit context for military terminology. LLMs don't know what "DODAAC" or "MTOE" mean without definition. - Structured JSON output: require the workflow to output JSON, not prose. This is what enables Ontology write integration. - Error handling: configure what happens when the LLM fails to produce structured output. Routes to human review queue, not silent failure.

Blocks 5–8 — Advanced AIP Logic and Python Transforms

Day: 2

Block 5: Multi-step chains with parallel branches and Ontology write Actions. **Block 6:** Python transforms — extracting and formatting Ontology data for AIP context. Military terminology definitions must be explicit. **Block 7:** Context management — chunking strategies for large datasets that exceed context windows. **Block 8:** Ontology write integration with human review queue. All objects begin as `status = Draft`. **This design is non-negotiable.**

Blocks 9–12 — RAG Architecture

Day: 3

Block 9: Semantic search setup, retrieval from Ontology Objects, context construction. **Block 10:** Full RAG pipeline: retrieval → context formatting → LLM → JSON output → Ontology write with review gate. **Block 11:** RAG quality evaluation: retrieval relevance scoring, grounding failure detection, hallucination patterns on operational data. **Block 12:** End-to-end workflow practice with instructor coaching.

Blocks 13–16 — Agent Studio and Evaluation

Day: 4

Block 13: Agent Studio architecture, tool definition, tool-use authorization (what can the agent do?), memory scope. **Block 14:** Testing Agent Studio — confirming authorization controls prevent out-of-scope tool use. **Block 15:** Red-teaming: adversarial prompt testing. For every workflow, ask: "How could this produce a dangerous or misleading output?" **Block 16:** AI Output Validation Framework; complete AIP Authorization Checklist for a practice workflow.

Blocks 17–20 — Deployment and Practical Exercise

Day: 5

Block 17: Production deployment: pipeline scheduling, monitoring, failure alerting, rollback procedures. **Block 18–19:** Scenario brief; planning time; authorization checklist guidance. **Block 20:** Practical Exercise (4 hours). Build an AIP Logic workflow extracting structured SITREP data, routing to human review queue, writing to Ontology via Action.

Hard No-Go: Workflow writes to production Objects without human checkpoint.

PART C — SL 4M: ML ENGINEER

Duration: 5 days (40 hours) | **T:I ratio:** 4:1 | **Instructor req:** ML production experience; C2DAO MLE SME designation

Block 1 — MLE Role and Model Governance Overview

Hours: 1.0 | **Method:** Brief | **Day:** 1

Purpose: Read TM-40M Chapter 9 (governance) first. The governance destination makes the technical work purposeful. Every model must eventually produce a model card.

TLO: The trainee will describe the MLE role on MSS and state the required components of a model governance document (model card).

Blocks 2–5 — Workspace Setup, Foundry Write Pattern, Feature Engineering

Day: 1

Block 2 (GPU Workspace Setup): GPU allocation, package management, Foundry connectivity. Write transaction test on Day 1 — this must work before modeling begins.

Block 3 (Foundry Write Pattern): Transaction-based output writes. Test this on Day 1. Trainees who skip this lose evaluation time on Day 5.

Block 4 (Feature Engineering Principles — Lecture): Null handling strategies (impute vs. drop vs. sentinel), encoding (one-hot vs. ordinal), scaling (standard vs. min-max), leakage detection.

Block 5 (Feature Engineering Practice): Apply to provided dataset. Document each feature decision in writing.

Blocks 6–9 — Feature Pipeline Build

Day: 2

Block 6: Full feature pipeline: raw → feature matrix. Every step documented. **Block 7:** Leakage audit. Can any feature be derived from the label? Run the procedure from TM-40M Ch 3. This is evaluated. **Block 8:** Output to Foundry curated dataset via write transaction. Verify schema. **Block 9:** Experiment setup: train/test split, baseline model (majority class), cross-validation configuration.

Blocks 10–13 — Model Training and Evaluation

Day: 3

Block 10: Training: scikit-learn or PyTorch in Code Workspace. Cross-validation. Hyperparameter tuning. **Block 11:** Evaluation: accuracy, precision/recall, ROC-AUC against USAREUR-AF thresholds. Calibration check is required. **Block 12:** Model comparison — train two models, select winner, document rationale including trade-offs. **Block 13:** Experiment tracking in Foundry model registry. Version each model.

Blocks 14–17 — Deployment and MLOps

Day: 4

Block 14: Serving endpoint deployment. Latency and throughput baseline. **Block 15:** Connect to Ontology via Actions that invoke inference and write predictions to Object properties. **Block 16:** Monitoring pipeline: data drift detection (PSI or KS test), threshold definition from validation set baseline, alert routing. **Block 17:** Operational patterns: readiness prediction, logistics demand forecasting, anomaly detection.

Blocks 18–21 — Governance and Practical Exercise

Day: 5

Block 18: Model governance document (model card) completion. Hard requirement: assumptions, training data, limitations, intended use restrictions, responsible AI declaration. **Block 19:** Deployment approval workflow; C2DAO governance for deployed models. **Block 20:** Scenario brief, planning time. **Block 21:** Practical Exercise (4 hours). Feature pipeline → train → evaluate → deploy → monitoring → governance doc.

Hard No-Go: Model calibration not performed. Governance document missing required sections. Drift pipeline fails to detect evaluator-seeded drift event.

PART J — SL 4J: PROGRAM MANAGER

Duration: 3 days (24 hours) | **T:I ratio:** 6:1 | **Instructor req:** Program management background; SL 3 certified; GFEBS/IMS proficiency

Block 1 — Program Data Architecture

Hours: 1.5 | **Method:** Lab | **Day:** 1

New doctrine content (March 2026): DDOF friction matrix for identifying integration friction points in program pipelines, roles and PM oversight mapping for DDOF-aligned governance, and portfolio health metrics framework. Instructor should reference these in Blocks 1, 5, and 8–9.

Purpose: The four Object Types (Program, Milestone, Resource, Risk) must be designed on paper before touching Ontology Manager. A poorly designed PM data model means rebuilding — which takes a day the course doesn't have.

Key Delivery Notes: - Design exercise on paper first: 4 Object Types, their properties, and the Link Types connecting them. 15 minutes on paper. Then build. - Program is the parent Object. Milestones, Resources, and Risks all link back to Program. - The data-as-of timestamp is not a column — it is a pattern. Every pipeline output that feeds a PM dashboard must include a computed `data_as_of_date` column using `CURRENT_DATE()`.

Block 2 — Ontology: PM Object Types, Link Types, Actions

Hours: 1.75 | **Method:** Lab | **Day:** 1

Key Delivery Notes: - Link Types: `Program → Milestone` (ONE_TO_MANY), `Program → Risk` (ONE_TO_MANY). Create both. - Action configuration: `UpdateMilestoneStatus` Action with parameters for `new_status` (GREEN/AMBER/RED) and `comments`. Access restricted to Editor role.

Block 3 — IMS Pipeline

Hours: 2.0 | **Method:** Lab | **Day:** 1

Purpose: The IMS pipeline is the heart of the PM data product. DATEDIFF for milestone variance and the RAG status computed column are the key deliverables.

Key Delivery Notes: - IMS Excel exports have dates as text. CAST before DATEDIFF — this is the most common pipeline failure. - DATEDIFF: `variance_days = DATEDIFF('day', planned_completion, actual_completion)`. Positive = late. Negative = ahead. - RAG status: `CASE WHEN variance_days > 30 THEN 'RED' WHEN variance_days > 0 THEN 'AMBER' ELSE 'GREEN' END AS milestone_rag`. - Data-as-of timestamp: `CURRENT_DATE() AS data_as_of_date`. Add to every pipeline output.

Block 4 — GFEBS Obligation Pipeline

Hours: 2.0 | **Method:** Lab | **Day:** 1

Key Delivery Notes: - Configure in Append mode before the first run. Add `CURRENT_DATE() AS snapshot_date`. Run twice. Verify two distinct snapshot dates in the output.

Block 5 — Workshop Milestone Dashboard

Hours: 2.0 | **Method:** Lab | **Day:** 2

Key Delivery Notes: - RAG conditional formatting: table rows colored by `milestone_rag`. Configure RED, AMBER, GREEN color rules. - Data-as-of widget: a text or metric widget displaying the most recent `data_as_of_date` value. This is a hard No-Go if absent on the evaluation.

Blocks 6–9 — Quiver, Contour, Reporting, Checklist

Day: 2

Block 6 (Quiver): Obligation rate chart with quarterly target reference line. Q2 target = 50%. **Block 7 (Contour):** Portfolio health matrix. Sort by `overall_status` ascending (RED at top). **Block 8 (Reporting):** Scheduled pipeline refresh. Build failure notification. PDF export procedure. **Block 9 (PM Dashboard Standards Checklist):** Walk every item. This is the evaluator's checklist.

Blocks 10–11 — Practice Run and Practical Exercise

Day: 3

Block 10 (Practice Run, 3.25 hrs — UNGRADED): Full stack from a different dataset. Instructor coaching available. This is instruction, not optional review.

Block 11 (Practical Exercise, 4 hrs): Full stack from provided IMS and GFEBS data. 7 tasks. Go standard: 6 of 7. Hard No-Go: dashboard missing data-as-of timestamp.

PART K — SL 4K: KNOWLEDGE MANAGER

Duration: 3 days (24 hours) | **T:I ratio:** 6:1 | **Instructor req:** KM background; SL 3 certified; AIP Logic configuration proficiency

Block 1 — KM Role on MSS

Hours: 1.0 | **Method:** Brief | **Day:** 1

New doctrine content (March 2026): FM 6-0 KM 5-step process mapped to MSS knowledge workflows, developmental domains framework for KM capability maturity, and Critical Knowledge Items (CKI) identification methodology. Instructor should reference these in Blocks 1, 4–5, and 9–10.

Key Delivery Notes: - The operational problem: institutional knowledge walks out at every PCS cycle. MSS is the solution — structured capture systems that survive personnel turbulence. - Design principle: "Build for the person who replaces you in 12 months." Every KM system must be operable by someone who was not there when it was built.

Block 2 — Knowledge Object Types

Hours: 2.0 | **Method:** Lab | **Day:** 1

Key Delivery Notes: - Five Object Types: Document, Lesson, AAR, SOP, ExpertiseProfile. Design on paper first. - Link Types: `Lesson → AAR` (Lesson is extracted from AARs), `Lesson → Unit` (which unit generated the lesson), `SOP → Unit` (which unit owns it).

Block 3 — AAR Submission Form

Hours: 0.75 | **Method:** Lab | **Day:** 1

Key Delivery Notes: - Required fields: unit, date, event type, location, what happened (text), key lesson, classification marking. - Field validation: required fields must produce an error if empty. Confirm the validation fires before submission.

Block 4 — Lessons Learned Pipeline

Hours: 2.0 | **Method:** Lab | **Day:** 1

Key Delivery Notes: - Tagging taxonomy: unit, event type (exercise, operation, transition), echelon, TTP category, classification level. Design the taxonomy before building the pipeline. - Distribution routing: `IF classification = 'SECRET', route to classified-only queue`. Routing logic depends on the tagging.

Block 5 — AIP Logic: Summarization with Human Review Gate

Hours: 1.75 | **Method:** Lab | **Day:** 1

Purpose: AIP summarization turns a 10-page AAR into a 3-paragraph lesson summary. The human review gate turns a draft into a vetted lesson.

Key Delivery Notes: - Workflow output: AIP-generated lessons ALWAYS begin with `status = Draft`. A Draft lesson requires KM review before `status = Published`. No exceptions. - The evaluator will specifically attempt to trigger a workflow that writes directly to Published status. Design the workflow so this is impossible by construction.

Blocks 6–10 — Knowledge Browser, Version Control, Expertise, PCS Transfer, Prompt Iteration

Day: 2

Block 6 (Knowledge Browser): Search by keyword, filter by tag/unit/date. Drill-down from search result to lesson full text.

Block 7 (SOP Version Control): Version tagging. Review notification workflow when SOP review date is passed.

Block 8 (ExpertiseProfile + Privacy Act): Skills taxonomy. Privacy Act authorities (TM-40K Section 8-1) apply to skills databases — the evaluator will ask about this.

Block 9 (PCS Transfer): Key person dependency analysis. Transfer package must name specific Foundry projects, Object Types, pipelines, data quality status, and required contacts. Generic template = No-Go.

Block 10 (AIP Prompt Iteration Lab): Test against 5 provided documents. Score extraction quality. Revise prompt. Retest. Repeat. The prompt matters more than the workflow configuration.

Blocks 11–13 — PCS Package Review and Practical Exercise

Day: 3

Block 11 (PCS Draft Review, 2.25 hrs): Each trainee presents draft PCS package to instructor. Instructor reviews against TM-40K Ch 9 completeness criteria. Revise based on feedback. **This block is instruction, not optional preparation.**

Block 12: Practical exercise scenario brief.

Block 13 (Practical Exercise, 4 hrs): 6 tasks: Ontology design, AAR form, lessons pipeline, AIP workflow with review gate, knowledge browser, PCS transfer package.

Hard No-Go: AIP workflow auto-publishes without human review gate.

PART L — SL 4L: SOFTWARE ENGINEER

Duration: 5 days (40 hours) | **T:I ratio:** 4:1 | **Instructor req:** OSDK/Platform SDK experience; SL 4L certified; C2DAO SWE SME designation

Block 1 — SWE Role and OSDK Architecture

Hours: 1.0 | **Method:** Lecture | **Day:** 1

Purpose: SL 4L SWEs are building production software on Foundry. Security must be established as a first principle before any code is written. Read TM-40L Chapter 9 (Security) before Day 1.

Key Delivery Notes: - The 5-layer data stack: raw data → pipelines → Ontology → OSDK → applications. SWEs live at layer 4–5. - OSDK architecture vs. REST: OSDK is type-generated from the Ontology definition. It is not a generic REST client. Type generation means changes to the Ontology break the OSDK client — trainees must understand this dependency. - Security principle established on Day 1: NO hardcoded credentials, tokens, or secrets in code. EVER. This is the hardest No-Go in the course.

Blocks 2–5 — OSDK Fundamentals

Day: 1

Block 2 (OSDK Setup): Auth architecture, client initialization, token handling. First single-object query. Verify authentication before anything else.

Block 3 (Filtering and Sorting): Query predicates, sort expressions, field selection. Build on Block 2's working client.

Block 4 (Pagination with ResourceIterator): This is an evaluated skill. If you write a query that retrieves page 1 and stops, the evaluator will use an Object set requiring 2+ pages. Use `ResourceIterator` and iterate all results. Every time.

Block 5 (Link Traversal): Query related Objects across Link Types. Join-like patterns without leaving the Ontology layer.

Blocks 6–9 — OSDK Advanced

Day: 2

Block 6 (Action Execution): Async response pattern — OSDK Action execution returns a task ID, not a result. Poll for completion with task ID.

Block 7 (Error Handling and Retry): Action execution failures, timeout patterns, structured error response.

Block 8 (Object Subscriptions): Real-time change notifications via WebSocket. Connecting subscriptions to application state for live-updating UIs.

Block 9 (Bulk Operations): Batch queries, bulk Action submissions. The pattern: never loop over Objects making individual API calls. Use bulk patterns. The evaluator tests with 200+ Objects.

Blocks 10–13 — Platform SDK and TypeScript FOO

Day: 3

Block 10 (Platform SDK): Dataset read/write operations, write transactions, file resources, branch management. Different layer than OSDK — Platform SDK operates on datasets, not Ontology Objects.

Block 11 (Platform SDK Exercise): Build a dataset integration using the read/write transaction pattern.

Block 12 (TypeScript FOO): Repository structure, computed property implementation, function registration. FOO extends the Ontology with computed properties that Foundry evaluates.

Block 13 (FOO Bulk Query Patterns): N+1 queries are the most common FOO performance failure. If your FOO makes 1 API call per Object, it will time out at 200+ Objects. Use bulk query patterns from TM-40L Chapter 5.

Blocks 14–17 — Action Validators and Slate

Day: 4

Block 14 (Action Validators): Multi-condition validation with cross-field logic (`if status = DEPLOYED, location must be non-null`). Clear error message for each validation condition.

Block 15 (Validator Testing): Build a test suite with minimum 8 cases (4 valid, 4 invalid). Each test case paired with expected error message. The evaluator runs this test suite against the trainee's actual code — all 8 must pass.

Block 16 (Slate Applications): Application structure, Foundry API integration, widget binding, initial data load.

Block 17 (Slate State Management): When a user triggers an Action that updates an Object, the UI must refresh. Configure state variable tied to Action completion, not a timer. This is the hardest Slate skill.

Blocks 18–21 — CI/CD, Security, and Practical Exercise

Day: 5

Block 18 (CI/CD Lecture): PR workflow, automated testing integration, C2DAO deployment checklist. Code review is non-negotiable before any deployment.

Block 19 (Security Lecture): Token handling (Foundry Secrets API, not hardcoded), input sanitization (injection risk on all user inputs), OPSEC (no architecture details in commit messages).

Block 20 (Scenario Brief): Provide scenario; planning time.

Block 21 (Practical Exercise, 4 hrs): OSDK paginated query → Action validator with test suite → TypeScript FOO → Slate application with error states → deployment checklist.

Hard No-Go: Any hardcoded credential in application code. Validator test suite not fully passing (all 8 cases required).

PART G — SL 4N: UI/UX DESIGNER

Duration: 5 days (40 hours) | **T:I ratio:** 4:1 | **Instructor req:** Soldier Centered Design practitioner; SL 4N certified or C2DAO UX SME designation

Block 1 — Designer Role on MSS and SCD Methodology

Hours: 1.0 | **Method:** Lecture | **Day:** 1 | **Time:** 0800–0900

Purpose: The Designer is the voice of the user on the product team. This block establishes the SCD methodology and the balanced team model (PM + Designer + SWE) before any design work begins. Commercial UX assumptions must be replaced with operational design thinking on Day 1.

TLO: The trainee will describe the Designer's role on MSS, distinguish SCD from commercial UX practice, and state the balanced team model — identifying which decisions belong to the Designer and which belong to the PM and SWE.

Key Delivery Notes: - SCD is not commercial UX with a military skin. Operational tempo, classification constraints, DDIL, and cross-rank dynamics fundamentally change what "good design" means. - Balanced team model: PM owns the backlog, SWE owns the implementation, Designer owns user understanding and design specification. The boundary is about primary responsibility, not capability. - Designer boundaries: backlog prioritization → SL 4J; code implementation → SL 4L/SL 3; infrastructure → SL 4O; data modeling → SL 3/SL 4H.

Assessment: Role understanding evaluated during Practical Exercise design presentation (Day 5).

Block 2 — User Research Planning

Hours: 2.0 | **Method:** Studio | **Day:** 1 | **Time:** 0900–1100

Purpose: User research in operational environments cannot be improvised. Research plans, interview guides, and contextual inquiry protocols must be prepared before accessing users — access windows are limited by operational tempo.

TLO: The trainee will produce a user research plan including research questions, target user population, interview guide with SCD semi-structured interview questions, and a contextual inquiry protocol.

Key Delivery Notes: - Research access is the constraint: operational users have limited availability. Plan the research session before requesting access — a wasted session cannot be rescheduled easily. - Interview guide: open-ended questions about current workflows, pain points, and decision-making

processes. Avoid leading questions. Avoid jargon that assumes the user understands the technology. - Contextual inquiry: observe the user in their actual work environment (TOC, office, field). Document the environment conditions — lighting, noise, screen size, classification level — these are design constraints.

Assessment: Research plan quality evaluated in synthesis exercise (Block 4).

Block 3 — Practice Interview Session

Hours: 0.75 | **Method:** Lab | **Day:** 1 | **Time:** 1115–1200

Key Delivery Notes: - Paired exercise: trainees interview each other using their prepared SCD semi-structured interview guides. Instructor observes and provides feedback on technique. - Common errors: asking leading questions, taking verbatim notes instead of capturing insights, failing to probe on "why" behind stated preferences. - Rank dynamics: in operational settings, junior soldiers may defer to senior users. SCD technique must account for this — use individual interviews before group sessions.

Block 4 — Research Synthesis: Affinity Diagramming and Persona Creation

Hours: 2.0 | **Method:** Studio | **Day:** 1 | **Time:** 1300–1500

Purpose: Raw interview data must be synthesized into actionable design artifacts. Affinity diagramming clusters observations into themes. Personas translate themes into representative user profiles that drive design decisions.

TLO: The trainee will synthesize interview data into an affinity diagram, derive at least 3 insight themes, and produce a persona including role, rank range, operational context, key tasks, and pain points.

Key Delivery Notes: - Affinity diagramming: write each observation on a separate note, cluster into groups, name each group. The group names become the insight themes. - Persona includes: role, rank range, unit type, operational context (TOC, field, garrison), top 3 tasks, top 3 pain points, technology comfort level, classification environment. - A persona based on assumptions is not a persona — it is a guess. Personas must trace back to research data.

Assessment: Persona checkpoint conducted at start of Day 2.

Block 5 — OPSEC in User Research

Hours: 1.75 | **Method:** Lecture + Discussion | **Day:** 1 | **Time:** 1515–1700

Key Delivery Notes: - Classification considerations for research artifacts: interview notes, personas, and affinity diagrams may contain operational details. Mark and handle appropriately. - Rank dynamics in research sessions: a 1SG observing a usability test changes the results. Control for observer effects. -

Research artifact storage: store on approved systems at appropriate classification. Do not store operational research data on personal devices or unclassified design tools.

Block 6 — Information Architecture

Hours: 2.0 | **Method:** Lecture + Studio | **Day:** 2 | **Time:** 0830–1030

Purpose: Military users are information-dense decision-makers. The Designer's job is not to simplify — it is to structure. The decision-first hierarchy determines what appears on screen and in what order.

TLO: The trainee will design an information architecture for an MSS dashboard that passes the "glance, scan, commit" test at 2-second, 10-second, and 30-second levels — with a documented decision-first hierarchy.

Key Delivery Notes: - Decision-first hierarchy: What decision does the user need to make? → What information supports that decision? → What is the priority order? → How should it be spatially organized? → What interactions let the user drill deeper? Start here, not with a widget palette. - Glance test (2 sec): user identifies overall status (green/amber/red) without reading. Scan test (10 sec): user identifies which areas need attention. Commit test (30 sec): user drills into details for a decision. If a design fails any level, iterate. - Information density: avoid "one metric per screen" (too sparse) and "wall of numbers with no hierarchy" (too dense). All critical metrics visible simultaneously with detail on demand.

Assessment: Dashboard design critique at start of Day 3.

Block 7 — Visual Design for Military Applications

Hours: 1.25 | **Method:** Studio | **Day:** 2 | **Time:** 1045–1200

Key Delivery Notes: - Color is a safety system: classification banners, status indicators, and threat levels all use color as primary encoding. The Designer ensures the overall color system is coherent — status green does not clash with classification green. - Redundant encoding: every color-encoded meaning must have a redundant non-color indicator (icon, text label, pattern). WCAG 2.1 AA contrast ratios are the minimum standard. - Typography: legible at field conditions (variable lighting, distance from screen). Standard font sizes, not decorative type.

Block 8 — Workshop Layout Design

Hours: 2.0 | **Method:** Lab | **Day:** 2 | **Time:** 1300–1500

TLO: The trainee will produce a Workshop layout specification with widget selection, data binding documentation, filter logic, and responsive behavior — using the MSS grid system and standard layout patterns.

Key Delivery Notes: - Workshop layout patterns: command dashboard (summary metrics with drill-down), data entry form (structured input with validation), drill-down explorer (list → detail view). Select the pattern that matches the user's decision workflow. - Widget catalog walkthrough: select widgets based on the information architecture, not based on visual appeal. A chart widget is appropriate when the user needs to see a trend. A table widget is appropriate when the user needs to compare specific values. - Data binding: document which Ontology Object properties bind to which widgets. This becomes the handoff spec for the SL 4L SWE or SL 3 builder.

Block 9 — Design Exercise: Command Dashboard

Hours: 1.75 | **Method:** Studio | **Day:** 2 | **Time:** 1515–1700

Key Delivery Notes: - Design a command dashboard for a specific WFF scenario using Workshop patterns and MSS visual standards. - Apply the full design chain: decision-first hierarchy → information architecture → widget selection → layout → visual design → data binding documentation. - Peer critique at start of Day 3: each trainee presents their dashboard for structured feedback.

Block 10 — Interaction Design: State Mapping

Hours: 2.0 | **Method:** Lecture + Studio | **Day:** 3 | **Time:** 0830–1030

TLO: The trainee will design a complete interaction specification covering default, loading, empty, error, and success states for a data entry workflow — with documented edge cases.

Key Delivery Notes: - Five states every interaction must address: default (initial view), loading (data being fetched), empty (no data available), error (something failed), success (action completed). If any state is undesigned, the user will encounter an uncontrolled experience. - Action flows: what happens when the user clicks a button? Document the sequence: click → loading state → API call → success state OR error state → next step. Every branch must be designed. - Edge cases: what happens when the user double-clicks? What happens when the network drops mid-action? What happens when the session expires? Design for these — they happen in operational environments.

Block 11 — Prototyping

Hours: 1.25 | **Method:** Studio | **Day:** 3 | **Time:** 1045–1200

Key Delivery Notes: - Low-fi sketches first, wireframes second, high-fidelity prototype last. Do not start in high fidelity — it wastes time when the information architecture is wrong. - Rapid iteration: sketch → critique → revise → repeat. Design tool proficiency matters for speed, but the thinking matters more than

the tool. - Prototype fidelity matches the question being asked: paper sketch for "is this the right layout?" wireframe for "is this the right interaction flow?" high-fi for "do users understand this interface?"

Block 12 — Interactive Prototype Build

Hours: 2.0 | **Method:** Lab | **Day:** 3 | **Time:** 1300–1500

TLO: The trainee will build a clickable interactive prototype of a data entry workflow with all five states designed (default, loading, empty, error, success) — ready for usability testing on Day 4.

Key Delivery Notes: - The prototype must be testable — a user should be able to walk through the primary task flow without the designer explaining what to do. - All five states must be represented in the prototype, even if simplified. An error state shown as a red text banner is sufficient — the design intent must be visible. - Prototype checkpoint at start of Day 4: confirm all trainees have a testable prototype.

Block 13 — Design Handoff Specifications

Hours: 1.75 | **Method:** Studio | **Day:** 3 | **Time:** 1515–1700

Purpose: A design that cannot be implemented without follow-up clarification is not a finished design. The handoff package is what the SL 4L SWE or SL 3 builder uses to build.

TLO: The trainee will produce a design-to-development handoff package including layout specification, data binding documentation, interaction specification, visual style reference, and accessibility requirements.

Key Delivery Notes: - Handoff package contents: annotated mockups, widget-to-data binding map, interaction specification (all states), filter logic, responsive behavior notes, accessibility requirements. - The test: can a SL 4L SWE implement this without asking the Designer a single clarifying question? If not, the handoff is incomplete. - Implementation-ready specification format: consistent layout across all projects. Use the C2DAO design handoff template.

Block 14 — Accessibility: Section 508 and WCAG 2.1 AA

Hours: 2.0 | **Method:** Lecture + Lab | **Day:** 4 | **Time:** 0830–1030

TLO: The trainee will complete an accessibility checklist covering contrast ratios, keyboard navigation, text alternatives, touch targets, and screen reader compatibility — and identify at least 3 accessibility issues in their own prototype.

Key Delivery Notes: - Section 508 requirements and WCAG 2.1 AA criteria are the minimum standard — not a stretch goal. - Automated testing tools: run automated scans first to catch the low-hanging fruit (contrast ratios, missing alt text). Manual testing catches what automated tools miss (keyboard navigation flow, screen reader experience). - Color alone cannot convey meaning: every status color must have a redundant text or icon indicator. This is the most common accessibility failure in MSS applications.

Assessment: Accessibility audit evaluated in Practical Exercise (Day 5).

Block 15 — Accessibility Audit

Hours: 1.25 | **Method:** Lab | **Day:** 4 | **Time:** 1045–1200

Key Delivery Notes: - Audit your own prototype against the MSS accessibility checklist. Identify issues. Fix what you can before the usability test in Block 16. - Document each issue with: location, severity, WCAG criterion violated, and proposed fix. This documentation becomes part of the handoff package.

Block 16 — Usability Testing

Hours: 2.0 | **Method:** Lab | **Day:** 4 | **Time:** 1300–1500

TLO: The trainee will execute a usability test with 5+ representative users (paired trainees serving as test participants), documenting task completion rates, error patterns, and severity-rated findings.

Key Delivery Notes: - Think-aloud protocol: ask users to verbalize their thought process as they navigate. "I'm looking for the readiness status... I expected it to be on the left... I'm clicking this filter but nothing happened." This reveals mental model mismatches. - Task completion rate: for each test task, record whether the user completed it successfully, with difficulty, or failed. Success rate below 80% on a primary task indicates a design problem. - Severity rating for findings: Critical (blocks task completion), Major (causes significant confusion), Minor (noticeable but workaround exists), Cosmetic (visual polish).

Assessment: Usability findings evaluated in Practical Exercise design presentation (Day 5).

Block 17 — Usability Findings Synthesis

Hours: 1.75 | **Method:** Studio | **Day:** 4 | **Time:** 1515–1700

Key Delivery Notes: - Pattern identification: look for findings that multiple users experienced. A single user's confusion may be an outlier — the same confusion across 3 of 5 users is a design problem. - Design recommendations: for each Critical or Major finding, propose a design change and sketch the revised interaction. Do not just identify problems — propose solutions. - Prioritization: fix Critical findings before Major findings. If time is limited, a design that works correctly is better than a design that looks polished but fails usability.

Block 18 — Design Governance

Hours: 1.0 | **Method:** Lecture | **Day:** 5 | **Time:** 0800–0900

Key Delivery Notes: - Pattern libraries: reusable design patterns documented for consistency across MSS applications. The Designer's job is to contribute patterns, not just consume them. - Design review process: every design goes through structured review before handoff — accessibility check, visual standards check, interaction completeness check. - Cross-track coordination: the Designer's role in sprint ceremonies. Design review happens before implementation begins, not after.

Blocks 19–22 — Practical Exercise and Design Presentation

Day: 5

Blocks 19–21 (EX_40N Practical Exercise, ~5 hrs): Design an MSS application from user research through handoff — including research plan, persona, information architecture, Workshop layout specification, interaction design with all states, accessibility audit, usability test findings, and design-to-development handoff package.

Block 22 (Design Presentation, 1.25 hrs): Each trainee presents their design, research findings, and handoff package. Evaluator assesses design rationale, SCD methodology application, and handoff package completeness.

Go standard: Pass 7 of 9 learning objectives. Design presentation demonstrates clear connection from research findings to design decisions. Handoff package is implementation-ready.

Hard No-Go: Design with no user research basis. Accessibility checklist not completed. Handoff package missing data binding documentation.

Block 23 — Post-Test and Course Evaluation

Hours: 0.5 | **Method:** Evaluation | **Day:** 5 | **Time:** 1630–1700

Post-test: EXAM_TM40N_POST administered. SL 5N pathway discussion for designers moving to enterprise design systems.

PART H — SL 4O: PLATFORM ENGINEER

Duration: 5 days (40 hours) | **T:I ratio:** 4:1 | **Instructor req:** Kubernetes operational experience; SL 4O certified or C2DAO Platform SME designation

Block 1 — Platform Engineer Role and IDP Concept

Hours: 1.0 | **Method:** Lecture | **Day:** 1 | **Time:** 0800–0900

Purpose: The Platform Engineer builds the floor that every other track stands on. When the platform fails, every application fails. This block establishes the platform-as-product mindset and the Internal Developer Platform (IDP) concept before any infrastructure work begins.

TLO: The trainee will describe the Platform Engineer's role on MSS, distinguish the platform-as-product mindset from the infrastructure mindset, and identify the platform services required to support MSS application teams.

Key Delivery Notes: - Platform-as-product: "I configure servers" → "I build self-service capabilities for application teams." The Platform Engineer's primary users are application developers (SL 4L, SL 3), not end users. - Platform failures are simultaneous, cross-cutting, and high-blast-radius. A SWE's application failure affects one application. A platform failure affects every application. - Platform Engineer boundaries: owns infrastructure around Foundry (K8s clusters, CI/CD, container registries, monitoring stacks, deployment toolchain), NOT Foundry itself. - Read TM-40O Chapter 8 (Platform Security and Compliance) — security requirements affect every infrastructure decision. Read before the labs, not after.

Assessment: Platform mindset evaluated throughout course; IDP concept assessed in Practical Exercise architecture (Day 5).

Block 2 — Kubernetes Fundamentals

Hours: 2.0 | **Method:** Lab | **Day:** 1 | **Time:** 0900–1100

Purpose: Kubernetes is the runtime for MSS applications outside Foundry. Trainees must be able to deploy, expose, and manage a workload before any advanced topics are introduced.

TLO: The trainee will deploy a workload to a Kubernetes cluster using declarative manifests, verify cluster connectivity, and demonstrate kubectl operations for pod management, log viewing, and event inspection.

Key Delivery Notes: - Kubernetes is a declarative system: you describe what you want in YAML, and the platform reconciles reality to match. Understand this contract or you will fight the system. - Pods are cattle, not pets. They are created, destroyed, and replaced automatically. Do not SSH into pods. Do not store state in pods. - kubectl operations: `kubectl apply -f`, `kubectl get pods`, `kubectl logs`, `kubectl describe pod`, `kubectl delete pod`. These must be second nature by end of Day 1.

Block 3 — Deployments, Services, and ConfigMaps

Hours: 0.75 | **Method:** Lab | **Day:** 1 | **Time:** 1115–1200

Key Delivery Notes: - Deployments manage pod replicas declaratively. Services provide stable networking endpoints. ConfigMaps separate configuration from container images. - Declarative workload management: define the desired state in YAML, apply it, and let the controller reconcile. Do not imperatively manage pods. - Labels are the API: Services find pods by label selectors. A consistent labeling strategy is foundational. Label every resource with `app`, `env`, and `team` at minimum.

Block 4 — Resource Management and Health Checks

Hours: 2.0 | **Method:** Lab | **Day:** 1 | **Time:** 1300–1500

TLO: The trainee will configure resource requests and limits, implement liveness/readiness/startup probes, and demonstrate understanding of pod lifecycle management.

Key Delivery Notes: - Resource requests: what the pod needs to start. Resource limits: the maximum the pod is allowed to use. Set both — a pod without limits can consume the entire node. - Health checks: liveness probe (is the container alive?), readiness probe (is the container ready to receive traffic?), startup probe (has the container finished starting?). An application without health checks is invisible to the platform. - Pod lifecycle: Pending → Running → Succeeded/Failed. Understand what causes a pod to stay in Pending (insufficient resources) or CrashLoopBackOff (container fails health check repeatedly).

Block 5 — Namespaces, Resource Quotas, and Isolation

Hours: 1.75 | **Method:** Lab | **Day:** 1 | **Time:** 1515–1700

Key Delivery Notes: - Namespaces provide logical isolation. One namespace per application per environment is the standard pattern. - Resource quotas: prevent any single team from consuming the entire cluster's resources. Set CPU and memory quotas per namespace. - LimitRanges: set default resource requests and limits for pods in a namespace. Prevents pods from deploying without resource specifications. - Resource governance: quotas and LimitRanges are platform policy, not suggestions. They protect the cluster from noisy neighbors.

Block 6 — Infrastructure as Code and Configuration Templating

Hours: 2.0 | **Method:** Lecture + Lab | **Day:** 2 | **Time:** 0830–1030

Purpose: If a change is not in Git, it did not happen. If it happened and is not in Git, it is drift — and drift kills. IaC principles must be established before any infrastructure is built that will persist beyond training.

TLO: The trainee will apply IaC principles (declarative configuration, idempotency, environment parity) and use configuration templating (ytt/Helm) to manage environment-specific settings without duplicating manifests.

Key Delivery Notes: - IaC benefits: every change has a commit, an author, and a review. Environments are reproducible. Rollback is `git revert`. - Idempotency: applying the same configuration twice produces the same result. If your IaC is not idempotent, you cannot safely re-apply it. - Templating: use ytt or Helm to parameterize environment-specific values (namespace name, resource limits, image tags). Do not maintain separate YAML files for each environment.

Block 7 — GitOps Setup

Hours: 1.25 | **Method:** Lab | **Day:** 2 | **Time:** 1045–1200

TLO: The trainee will configure a GitOps controller, deploy an application by committing to Git, and observe the controller reconcile desired state to actual state.

Key Delivery Notes: - GitOps loop: push to Git → GitOps controller detects change → controller reconciles cluster state to match Git → cluster state matches desired state. The cluster is always a reflection of the Git repo. - Deployment by commit: the trainee changes a configuration value in Git, commits, and observes the GitOps controller apply the change to the cluster. No `kubectl apply` needed after initial setup. - Verify: after the controller reconciles, confirm the cluster state matches the Git configuration.

Block 8 — Drift Detection

Hours: 2.0 | **Method:** Lab | **Day:** 2 | **Time:** 1300–1500

TLO: The trainee will manually modify cluster state, observe the GitOps controller detect and revert the drift, and configure drift alerts.

Key Delivery Notes: - Drift scenario: manually `kubectl edit` a deployment to change the replica count. The GitOps controller should detect the deviation and revert to the Git-defined state. If it does not, the GitOps configuration is incomplete. - Drift alerts: configure alerts that fire when drift is detected. Persistent drift may indicate a GitOps controller failure or a configuration that the controller cannot

reconcile. - Operational principle: if someone can change the cluster without going through Git, the GitOps guarantee is broken. Network policies and RBAC should limit direct cluster modification to emergency break-glass scenarios.

Block 9 — Network Policies

Hours: 1.75 | **Method:** Lab | **Day:** 2 | **Time:** 1515–1700

Key Delivery Notes: - Default deny: start with a network policy that denies all ingress and egress for the namespace. Then add explicit allow rules for required traffic. - Pod-to-pod communication: allow only the traffic that the application requires. If Service A needs to talk to Service B, create an allow rule. If it does not, deny by default. - Pod-to-external communication: control what can leave the cluster. Production workloads should not have unrestricted egress. - Testing: after applying network policies, verify that allowed traffic works AND that denied traffic is actually blocked. A network policy that does not block is worse than no policy — it creates a false sense of security.

Block 10 — Container Hardening

Hours: 2.0 | **Method:** Lecture + Lab | **Day:** 3 | **Time:** 0830–1030

TLO: The trainee will harden a container image starting from an Iron Bank base image, applying multi-stage build, non-root execution, and capability dropping — producing a container that passes a vulnerability scan.

Key Delivery Notes: - Iron Bank base images: DoD-approved hardened base images. Start from Iron Bank, not from Docker Hub. Docker Hub images are not authorized for production MSS workloads. - Multi-stage build: build stage installs build tools and compiles. Production stage copies only the compiled artifact. Build tools do not ship in the production image. - Non-root execution: containers run as a non-root user. `USER 1000` in the Dockerfile. A container running as root that is compromised gives the attacker root on the node. - Capability dropping: drop all Linux capabilities, add back only what the application requires. `securityContext.capabilities.drop: ["ALL"]`.

Block 11 — Container Scanning and Image Management

Hours: 1.25 | **Method:** Lab | **Day:** 3 | **Time:** 1045–1200

Key Delivery Notes: - Vulnerability scanning: scan every image before deployment. Block deployment of images with CRITICAL or HIGH vulnerabilities that have a fix available. - Digest pinning: reference images by SHA256 digest, not by tag. Tags are mutable — someone can push a new image to the same

tag. Digests are immutable. - Image signing: sign images in the CI pipeline. Admission controllers reject unsigned images. This ensures only CI-built images deploy to the cluster.

Block 12 — CI/CD Pipeline Design

Hours: 2.0 | **Method:** Lecture + Lab | **Day:** 3 | **Time:** 1300–1500

TLO: The trainee will design and build a CI/CD pipeline from source to deployment with defined stages, security gates, and artifact management.

Key Delivery Notes: - Pipeline stages: (1) build, (2) unit test, (3) security scan, (4) image build, (5) image scan, (6) deploy to staging, (7) integration test, (8) promote to production (manual gate). - Artifact management: built artifacts (container images, Helm charts) are stored in an artifact repository with version tagging. Production deployments reference artifacts from the repository, not from the CI workspace. - Pipeline-as-code: the pipeline definition is in Git alongside the application code. Changes to the pipeline go through the same review process as application changes.

Block 13 — CI/CD Security Gates

Hours: 1.75 | **Method:** Lab | **Day:** 3 | **Time:** 1515–1700

TLO: The trainee will add secrets detection, SCA, SAST, and image scanning gates to the CI/CD pipeline — and demonstrate that a gate blocks deployment when a vulnerability is detected.

Key Delivery Notes: - Secrets detection: scan for hardcoded credentials, API keys, and tokens in the codebase. Block the pipeline if detected. This is the most critical gate. - SCA (Software Composition Analysis): scan dependencies for known vulnerabilities. Block on CRITICAL/HIGH with available fix. - SAST (Static Application Security Testing): scan source code for common vulnerability patterns. Block on CRITICAL findings. - Gate blocking demonstration: intentionally introduce a vulnerability and confirm the pipeline blocks. A security gate that does not block is security theater.

Block 14 — Deployment Strategies

Hours: 2.0 | **Method:** Lecture + Lab | **Day:** 4 | **Time:** 0830–1030

TLO: The trainee will implement rolling update and blue/green deployment strategies, execute a rollback from each strategy, and describe when each strategy is appropriate.

Key Delivery Notes: - Rolling update: gradually replace old pods with new pods. Zero downtime if health checks are configured correctly. Rollback: `kubectl rollout undo`. Appropriate for most routine updates. - Blue/green: deploy the new version alongside the old version. Switch traffic when the new version is verified. Rollback: switch traffic back. Appropriate for high-risk changes where instant rollback

is required. - Canary (awareness): deploy the new version to a small percentage of traffic. Monitor for errors. Promote or rollback based on metrics. - Rollback practice: every trainee must execute a rollback from both rolling update and blue/green. In production, the rollback must be muscle memory, not a procedure to look up under pressure.

Block 15 — Monitoring and Alerting

Hours: 1.25 | **Method:** Lab | **Day:** 4 | **Time:** 1045–1200

Key Delivery Notes: - Health dashboards: cluster resource utilization (CPU, memory, disk), pod status summary, deployment status. One-glance cluster health. - Resource utilization metrics: track per-namespace consumption against quotas. Approaching quota limits should trigger an alert before pods fail to schedule. - Actionable alerting: every alert must have a documented response action. An alert that fires with no response procedure is noise. Noise trains operators to ignore alerts.

Block 16 — Air-Gapped Deployment: Bundle Creation

Hours: 2.0 | **Method:** Lecture + Lab | **Day:** 4 | **Time:** 1300–1500

Purpose: USAREUR-AF operates in environments where network connectivity to external registries is unavailable. Application deployment must work across an air gap — bundled artifacts transferred physically or through approved cross-domain solutions.

TLO: The trainee will package an application for air-gapped deployment: bundle container images, Helm charts, configuration, and dependencies for transfer across an air gap using imgpkg or equivalent tooling.

Key Delivery Notes: - Bundle creation: `imgpkg push` packages container images and configuration into a single relocatable bundle. The bundle contains everything needed to deploy — no external network access required at the target. - Internal registry setup: configure a container registry on the target cluster. Import the bundle into the internal registry. - Dependency mirroring: all container image dependencies must be included in the bundle. A missing dependency means a failed deployment with no way to pull it. - Certificate management: internal registries require TLS certificates. Configure the cluster to trust the internal registry's certificate.

Block 17 — Air-Gapped Deployment Exercise

Hours: 1.75 | **Method:** Lab | **Day:** 4 | **Time:** 1515–1700

TLO: The trainee will deploy an application to a simulated air-gapped cluster using bundled artifacts only — with no external network access.

Key Delivery Notes: - Simulated air gap: network policies block all egress from the training namespace. The only container images available are those imported into the internal registry from the bundle. - Success criteria: application deploys, health checks pass, the application serves traffic. If any dependency is missing from the bundle, the deployment fails — this is the lesson. - Document the procedure: air-gapped deployment is a repeatable procedure. Document every step so a different operator can execute it.

Block 18 — Platform Security and Compliance

Hours: 1.0 | **Method:** Lecture | **Day:** 5 | **Time:** 0800–0900

Key Delivery Notes: - RMF/ATO from the infrastructure perspective: the platform is part of the authorization boundary. STIG compliance, access control, and audit logging are platform responsibilities. - Access control: RBAC scoped to namespaces. Principle of least privilege. Break-glass procedures for emergency access with audit trail. - Audit logging: all kubectl operations, all API server access, all admission controller decisions are logged. Logs are forwarded to a centralized log store. Retention per classification policy.

Blocks 19–22 — Practical Exercise and Platform Demonstration

Day: 5

Blocks 19–21 (EX_40O Practical Exercise, ~5 hrs): Build and secure a platform environment: Kubernetes deployment with resource management and health checks, GitOps workflow with drift detection, CI/CD pipeline with security gates, container hardening, and air-gapped deployment.

Block 22 (Platform Demonstration, 1.25 hrs): Each trainee presents their environment, pipeline, and security posture. Evaluator assesses infrastructure design, security controls, and operational procedures.

Go standard: Pass 7 of 9 learning objectives. GitOps workflow detects and reverts drift. CI/CD pipeline blocks on a security gate violation. Air-gapped deployment succeeds.

Hard No-Go: Container running as root in production deployment. CI/CD pipeline with no security gates. Air-gapped deployment fails due to missing dependency.

Block 23 — Post-Test and Course Evaluation

Hours: 0.5 | **Method:** Evaluation | **Day:** 5 | **Time:** 1630–1700

Post-test: EXAM_TM40O_POST administered. SL 5O pathway discussion for platform engineers moving to fleet management and SRE.
