ARM Lab SOP Checklist

Author: Seth Cole
Purpose: A process-only checklist for building an ARM (JSON) lab with modular templates, safe deployment habits, and clean repo structure.
Last updated: January 21, 2026

Scope note: This document focuses on workflow and repeatability. It intentionally avoids template implementation details.

1) Create the project workspace (OneDrive)

- [ ] Create a new folder named **arm-network-lab** (or a name consistent with your portfolio).
- [ ] Add the standard repo structure: **templates/**, **scripts/**, **docs/**, plus a **README.md** and **.gitignore**.
- [ ] Keep anything that could contain secrets out of source control (parameter files should not include secrets).

2) Install tools (one-time setup)

- [ ] Install **VS Code**.
- [ ] Install **PowerShell 7** (recommended).
- [ ] Install the **Az PowerShell** module and verify you can sign in and select a subscription.
- [ ] VS Code extensions (recommended): ARM Tools, PowerShell, GitLens (optional), EditorConfig (optional).

3) Initialize the repo locally

- [ ] Open the project folder in VS Code.
- [ ] Create **.gitignore** early (exclude local settings, temp files, and anything sensitive).
- [ ] Create your baseline parameter file (e.g., **parameters.dev.json**) early so you do not hardcode values in templates.

4) Decide the execution model (before writing templates)

- [ ] Define a default region for the lab and keep it consistent across deployments (e.g., **westus**).
- [ ] Choose a resource group naming convention for lab runs (and document it).

- [ ] Set the default deployment mode to **Incremental** for safety; treat **Complete** as destructive-by-default.
- [ ] Record these decisions briefly in **docs/decisions.md**.

5) Build templates in deployment order

- [ ] Start with the first component template (e.g., network). Validate and deploy it independently.
- [ ] Re-deploy the same template/parameters to confirm idempotency (no unexpected changes on re-run).
- [ ] Add the next component template (e.g., security). Repeat validate → deploy → re-deploy.
- [ ] Create the orchestration template (**main.json**) after at least one child template is proven deployable.

6) Add scripts for repeatability (minimal, not fancy)

- [ ] Create **scripts/deploy.ps1** to run the deployment with the chosen mode and parameter file.
- [ ] Create **scripts/whatif.ps1** to run What-If using the same template/parameter set.
- [ ] Create **scripts/cleanup.ps1** to remove the lab environment (typically by removing the resource group).
- [ ] Parameterize RG name, region, and parameter file path; do not embed secrets in scripts.

7) Validation loop (your default workflow)

- [ ] Run local/static checks (editor validation) when you make changes.
- [ ] Run What-If before a real deploy after any meaningful change.
- [ ] Deploy the change.
- [ ] Deploy again to confirm stable, repeatable behavior.
- [ ] Capture outcomes and common errors in **docs/troubleshooting.md**.

8) README build-out (after the system works)

- [ ] Write the README after templates deploy cleanly and repeatably.
- [ ] Include: objective, what the lab proves, repo structure, how to run (what-if → deploy → cleanup), and deployment modes (high-level).
- [ ] Explain the modular design choice (parent + child templates by component).

- [ ] Add the required section near the end: **Post-Provisioning Automation (Out of Scope for This Lab)**.

9) Git + GitHub workflow (after first working deployment)

- [ ] Initialize git, then commit in small, meaningful increments (structure → first working component → scripts → orchestration → docs).
- [ ] Create the GitHub repo, add the remote, and push.
- [ ] Optionally use GitHub Issues as your backlog for next improvements.

10) Done criteria (definition of finished)

- [ ] What-If runs successfully and is easy to repeat.
- [ ] Deployment runs successfully from a script and is easy to repeat.
- [ ] Re-running the deployment does not introduce unexpected changes.
- [ ] Cleanup removes the environment reliably.
- [ ] README communicates intent and responsibility boundaries clearly.

Notes: Keep this checklist in your repo (or OneDrive project folder) and update it only when your workflow changes.