Whitepaper #02: The Disconnected Pipeline

Subtitle: *Solving Dependency Management & Containerization in Secure Facilities* **Author:** Dustin J. Ober, PMP

---

1. Executive Summary

The Bottom Line Up Front (BLUF): The single greatest bottleneck in sovereign AI development is not compute power, but **dependency management**. Modern software engineering relies on a "connected supply chain" (PyPI, npm, Docker Hub, Hugging Face). When this chain is broken by an air-gap or secure boundary, development velocity often drops to near zero.

The Solution: This brief outlines a standardized **"Disconnected DevOps"** pipeline. It moves away from ad-hoc file transfers ("burning a DVD with zip files") to a structured, automated system of **Internal Mirrors** and **Containerized Artifacts**.

The Outcome: By implementing local package repositories (e.g., Sonatype Nexus) and utilizing container-based transport (Docker/Apptainer), organizations can restore 90% of the "connected" developer experience while maintaining 100% Zero Trust compliance.

---

2. The Operational Challenge: Developing in the Dark

In the commercial world, a "Hello World" Python script takes 30 seconds to set up. In a secure facility, it can take 3 weeks.

2.1 The "Dependency Hell" Cascade

A developer requests the `pandas` library. Security approves it. The developer brings it in, only to find it requires `numpy`. They bring that in, only to find it requires `pytz`.

- **The Reality:** A standard AI environment (PyTorch + LangChain + Transformers) has over **300 transitive dependencies**.
- **The Failure Mode:** Manually moving these files one by one is impossible. It leads to "Shadow IT," where developers smuggle unauthorized libraries just to get their code to run.

2.2 The "It Works on My Machine" Crisis

Without a shared internet connection, every workstation drifts. Developer A has `v1.2` of a library; Developer B has `v1.3`.

- **Result:** Code written on one machine crashes on another. The lack of a centralized "Truth" (like PyPI) breaks collaboration.

---

3. The "Sneakernet" Protocol: Moving Assets Securely

The "Sneakernet"—physically moving data on portable media—is often ridiculed, but in closed systems, it is the only bandwidth available. The goal is to make it **robust**, not just a workaround.

3.1 The "Open Internet" Staging Area

You must establish a dedicated "Ingest Workstation" connected to the Open Internet. This machine does **not** contain sensitive internal data. Its only purpose is to fetch, build, and bundle public assets.

- **The "Bundle" Strategy:** Never move raw code. Move **Artifacts**.
  - *Bad:* Moving a folder of `.py` scripts.
  - *Good:* Moving a Docker Image (`.tar`) or a Python Wheel (`.whl`).

3.2 The Docker "Save/Load" Workflow

Containerization is the ultimate transport wrapper. It freezes the OS, the drivers, and the libraries into a single file.

Step 1: Build on the Open Internet

```
# On the Internet-connected machine
docker build -t my-ai-app:v1 .
```

Step 2: Export to Artifact

Docker provides a native command to flatten an image into a tarball.

```
docker save -o my-ai-app_v1.tar my-ai-app:v1
```

Step 3: Transfer & Scan

Move the `.tar` file to the transfer medium (CD/DVD/Diode). The security team scans *only this one file*. This is significantly faster than scanning thousands of loose source files.

Step 4: Hydrate in the Closed System

```
# On the Closed System machine
docker load -i my-ai-app_v1.tar
```

Result: The exact environment is restored, bit-for-bit. No missing libraries, no version conflicts.

---

4. Mirroring the World: The Local Repository Strategy

While "Sneakernet" works for moving massive artifacts (like Docker images), it is inefficient for granular dependency management. You cannot burn a DVD every time a developer needs a tiny helper library like `tqdm` or `requests`.

The solution is to establish a **Local Mirror**—an internal server that mimics the directory structure of public repositories like PyPI and Hugging Face.

4.1 The Infrastructure: Nexus or Artifactory

You need a "Binary Repository Manager." The two industry standards are **Sonatype Nexus** and **JFrog Artifactory**.

The Architecture:

```
┌─────────────────────────────────────────────────────────────────────┐
│                         OPEN INTERNET                               │
│  ┌───────────────────────────────────────────────────────────────┐  │
│  │ Nexus Instance (Proxy Mode)                                   │  │
│  │ - Proxies pypi.org                                            │  │
│  │ - Caches all packages downloaded by developers                │  │
│  └───────────────────────────────────────────────────────────────┘  │
└─────────────────────────────────────────────────────────────────────┘
                              │
                              │
                       [Export Script]
                    (Scheduled blob export)
                              │
                              │
                       ┌──────▼──────┐
                       │ DATA DIODE  │
                       └──────┬──────┘
                              │
                              │
┌─────────────────────────────▼───────────────────────────────────────┐
│                      CLOSED SYSTEM                                  │
│  ┌───────────────────────────────────────────────────────────────┐  │
│  │ Nexus Instance (Hosted Mode)                                  │  │
│  │ - Imports storage blob from Open Internet                     │  │
│  │ - Serves packages to internal developers                      │  │
│  └───────────────────────────────────────────────────────────────┘  │
└─────────────────────────────────────────────────────────────────────┘
```

4.2 Configuring the Client (The Developer Experience)

The goal is **Transparency**. The developer should type `pip install` and have it work, without knowing the internet is gone. We achieve this by modifying the global configuration.

For Python (pip):

Create or modify `~/.pip/pip.conf` (Linux) or `%APPDATA%\pip\pip.ini` (Windows):

```
[global]
index-url = https://nexus.internal.lab/repository/pypi-hosted/simple
trusted-host = nexus.internal.lab
```

Result: `pip` now ignores `pypi.org` and talks strictly to your internal server.

For Hugging Face (Local):

Hugging Face libraries (`transformers`) attempt to ping the internet by default. You must point them to your local cache or offline endpoint using environment variables in your `.bashrc` :

```
export HF_DATASETS_OFFLINE=1
export TRANSFORMERS_OFFLINE=1
export HF_HOME=/mnt/shared/models/huggingface
```

---

5. Advanced Containerization: Apptainer (Singularity)

While Docker is the standard for *building* containers, it is often forbidden in Closed Systems (especially HPC clusters) because the Docker daemon requires `root` privileges. If a container breaks out, the attacker gains root access to the host node.

The Solution: Apptainer (formerly Singularity).

5.1 The SIF Standard

Apptainer compresses an entire container into a Single Image File (`.sif`). Unlike Docker layers, a `.sif` file is a single, immutable artifact that can be cryptographically signed.

Feature	Docker	Apptainer
Format	Layered tarball	Single `.sif` file
Privileges	Requires `root` daemon	Rootless execution
Signing	Docker Content Trust	Native GPG signing
HPC Support	Limited	Native (MPI, SLURM)
GPU Passthrough	`--gpus` flag	`--nv` flag (simpler)

5.2 The "Docker-to-Apptainer" Pipeline

You do not need to rewrite your Dockerfiles. You simply convert them at the boundary.

Step 1: Save Docker Image (Open Internet)

```
docker save -o my-model.tar my-model:latest
```

Step 2: Build SIF (Transfer Boundary)

Use Apptainer to convert the Docker tarball into a secure SIF image.

```
apptainer build my-model.sif docker-archive://my-model.tar
```

Step 3: Execute Securely (Closed System)

Run the image as a standard user (no root required).

```
apptainer run --nv my-model.sif
```

Note: The `--nv` flag passes the NVIDIA GPU drivers from the host into the container automatically—a massive quality-of-life feature for AI workloads.

---

6. Security & Governance: The "Golden Image"

In a Closed System, you cannot "patch" vulnerabilities easily. Therefore, security shifts left—it must happen *before* the artifact enters the air-gap.

6.1 The Software Bill of Materials (SBOM)

Every container entering the Closed System must accompany an SBOM. This is a manifest listing every library (OS-level and Python-level) inside the image.

- **Tooling:** Use **Syft** or **Grype** to generate SBOMs during the build process.

```
syft my-model:latest -o cyclonedx-json > sbom.json
```

- **The Audit:** If a new CVE is discovered in `OpenSSL`, you query your central SBOM database to find exactly which offline containers are affected, rather than scanning terabytes of closed-system drives.

6.2 The "Golden Image" Strategy

Do not allow developers to bring in raw base images (like `ubuntu:latest`).

1. **Create a Base:** Security creates a "Hardened AI Base" (Ubuntu + CUDA + Python + Certs).
2. **Publish:** This image is available on the Closed System Nexus.
3. **Mandate:** All developer Dockerfiles must start with `FROM nexus.internal.lab/hardened-ai-base:v1`.

Example Hardened Base Dockerfile:

```
# hardened-ai-base:v1
FROM ubuntu:22.04

# Security hardening
RUN apt-get update && apt-get install -y --no-install-recommends \
    ca-certificates \
    curl \
    && rm -rf /var/lib/apt/lists/*

# CUDA runtime (pre-approved version)
COPY cuda-12.1-runtime.deb /tmp/
RUN dpkg -i /tmp/cuda-12.1-runtime.deb && rm /tmp/*.deb

# Python (pinned version)
RUN apt-get update && apt-get install -y python3.11 python3.11-venv

# Non-root user
RUN useradd -m aiuser
USER aiuser
WORKDIR /home/aiuser
```

---

7. Transfer Checklist

Before transferring assets across the air-gap, verify:

- [ ] **Artifacts Built:** All dependencies bundled (Docker `.tar` or Python `.whl`)
- [ ] **SBOM Generated:** `syft` or `grype` output attached
- [ ] **Hashes Computed:** SHA256 checksums for all files
- [ ] **Signatures Applied:** GPG signatures for Apptainer `.sif` images
- [ ] **Manifest Created:** List of all files with versions and hashes
- [ ] **Security Scan:** Antivirus and vulnerability scan completed
- [ ] **Approval Obtained:** Change control board sign-off

---

8. Conclusion

The "Disconnected Pipeline" is the circulatory system of a Sovereign AI capability. Without it, the hardware discussed in Whitepaper #01 is merely expensive metal.

By moving from ad-hoc file transfers to a structured architecture of **Internal Mirrors**, **Containerized Artifacts**, and **Apptainer Runtimes**, organizations can achieve a development velocity that rivals the commercial sector while adhering to the strictest security mandates. The result is a system that is secure by design, auditable by default, and resilient against supply chain attacks.

Key Takeaways:

1. **Bundle, Don't Scatter:** Move artifacts (`.tar`, `.whl`), not raw files.
2. **Mirror Everything:** Internal Nexus mirrors restore the `pip install` experience.
3. **Convert at the Boundary:** Build with Docker, deploy with Apptainer.
4. **SBOM Everything:** Know what's inside every container before it crosses.
5. **Golden Images:** Mandate hardened base images for all development.

Next in this Series:

- **Whitepaper #03:** *Private Knowledge Retrieval: Architecting Local RAG Systems.*

About the Author

Dustin J. Ober, PMP, M.Ed. *AI Developer & Technical Instructional Systems Designer*

Dustin J. Ober is a specialist in the intersection of Artificial Intelligence, Instructional Strategy, and secure systems architecture. With a background spanning over two decades in the United States Air Force and defense contracting, he focuses on deploying high-impact technical solutions within mission-critical environments.

Unlike traditional developers who focus solely on code, Dustin bridges the gap between **technical capability** and **operational reality**. His expertise lies in architecting "Sovereign AI" systems—designing offline, air-gapped inference pipelines that allow organizations to leverage state-of-the-art intelligence without compromising data security or compliance.

He holds a Master of Education in Instructional Design & Technology and is a certified Project Management Professional (PMP). He actively develops open-source tools for the AI community, focusing on DSPy implementation, neuro-symbolic logic, and verifiable agentic workflows.

Connect:

- **Web:** aiober.com
- **LinkedIn:** linkedin.com/in/dustinober
- **Email:** dustinober@me.com

Suggested Citation: Ober, D. J. (2025). *The Disconnected Pipeline: Solving Dependency Management & Containerization in Secure Facilities* (Whitepaper No. 02). AIOber Technical Insights.