Server Synchronization Plan

BOEM Marine Sensitivity Toolkit — Internal/External Server Sync

Ben Best

2026-02-19

Table of contents

1 Motivation

The Marine Sensitivity Toolkit (MST) is a BOEM-funded project that delivers interactive web applications and documentation for assessing the sensitivity of marine ecosystems to offshore energy development. The project is actively developed by the lead contractor, Ben Best (EcoQuants), who maintains:

- **Interactive apps** (Composite Scores, Species Distribution) built with R Shiny
- **Project documentation** (marinesensitivity.org/docs) built with Quarto

The lead contractor will be out of the United States for an extended period to have his second child in his wife's home country of Italy. During this time, active development will continue on a server **external** to the BOEM network (the existing AWS-hosted development server), while BOEM's goal is to migrate production web services to a server **internal** to the BOEM network.

To bridge this gap, the **internal production server must initiate all synchronization** — pulling software updates (Docker images), data files, and static web content from the external development server, and pushing log files back for remote debugging. This document describes the architecture, synchronization strategy, implementation scripts, monitoring approach, and government compliance considerations for this arrangement.

2 Architecture Overview

The MST infrastructure spans three tiers: developer laptops, an external cloud server (AWS), and the target internal BOEM server. The following diagram shows the overall software architecture and data flow.

2.1 Current state (external only)

The existing external server on AWS (`100.25.173.0`) runs the full development stack defined in `docker-compose.yml`:

Service	Purpose	Port
caddy	reverse proxy + TLS + file server	80/443
rstudio	development IDE + Shiny Server	8787/3838
plumber	R API	8888
postgis	PostgreSQL + PostGIS spatial database	5432
pgadmin	database admin UI	8088
pgbkups	automated database backups	—
tile	pg_tileserv (vector tiles from PostGIS)	7800
tilecache	Varnish cache for tile	6081
titiler	cloud-optimized GeoTIFF tile server	8000
titilecache	Varnish cache for titiler	6082

Static web content is currently served via GitHub Pages:

- marinesensitivity.org — project homepage
- marinesensitivity.org/docs — project documentation

2.2 Target state (internal production)

The internal BOEM server runs a minimal production stack defined in `prod/docker-compose.yml`:

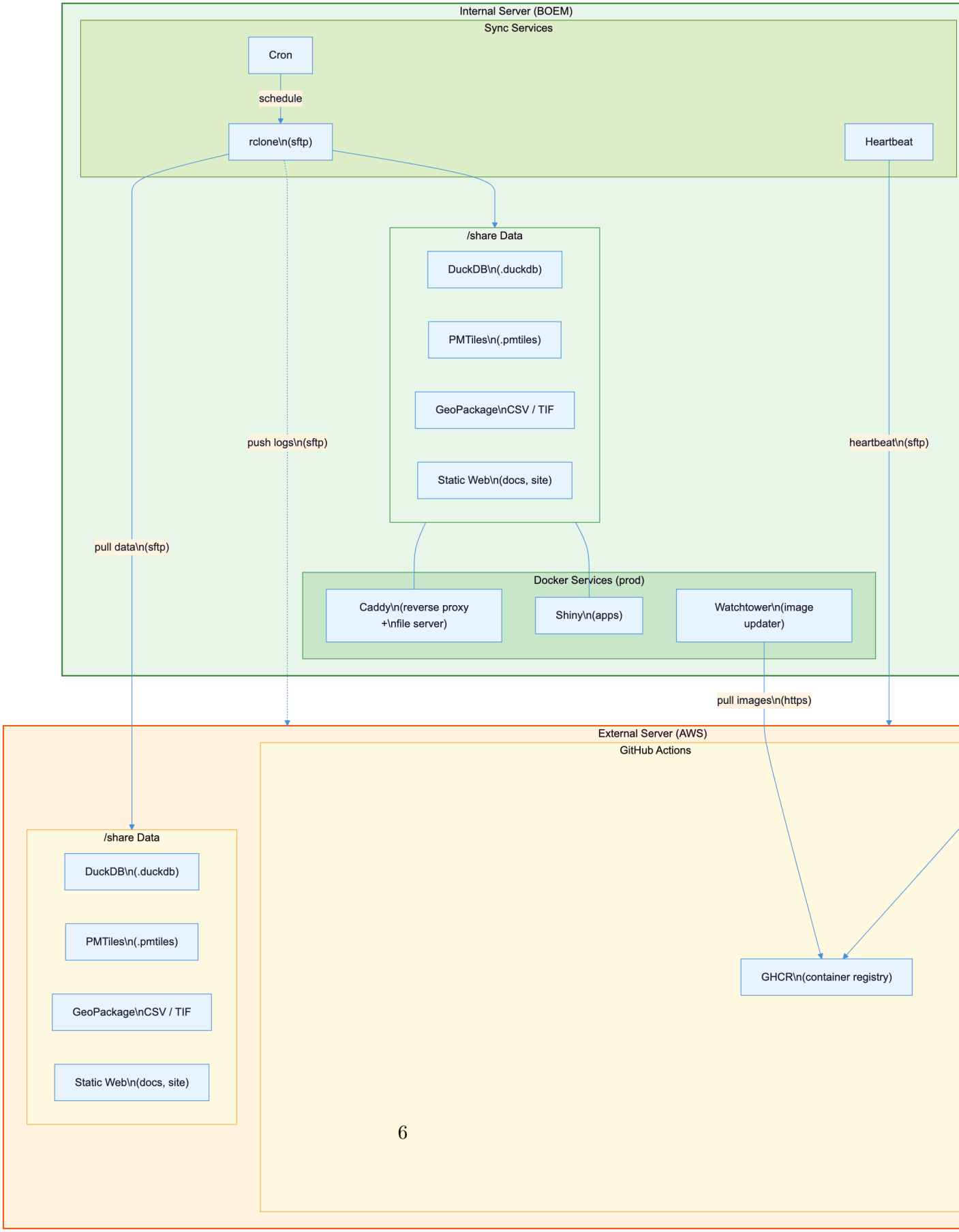

Figure 2.1: Overall server architecture showing developer laptop, external AWS server, and internal BOEM server with Docker services and data flows.

Service	Purpose	Port
caddy	reverse proxy + file server + docs	80/443
shiny	R Shiny applications	3838
watchtower	automatic Docker image updates	—

Services **not needed** on production:

- **rstudio** — development only; debugging happens on external server
- **postgis / pgadmin / pgbkups** — database migrated to DuckDB (file-based)
- **plumber** — API either proxied or migrated
- **tile / tilecache** — vector tiles migrated to PMTiles (file-based)
- **titiler / titilecache** — raster tiles served differently or proxied

The key architectural simplification is replacing the PostGIS + pg_tileserv stack with **PMTiles** (a single file format for vector tiles served directly by Caddy as static files) and replacing PostGIS-backed raster queries with **DuckDB** (an embedded analytical database read directly by Shiny apps).

3 Synchronization Overview

All synchronization is **initiated by the internal server** — it pulls updates from the external server and pushes logs back. No inbound connections to the BOEM network are required.

3.1 Sync summary

What	Direction	Tool	Schedule	Protocol
Data files (duckdb, gpkg, csv, tif)	internal pulls from external	rclone (sftp)	hourly	SSH
PMTiles	internal pulls from external	rclone (sftp)	hourly	SSH
Static website/docs	internal pulls from external	rclone (sftp)	hourly	SSH
Shiny app source	internal pulls from external	rclone (sftp)	hourly	SSH
Docker images	internal pulls from registry	Watchtower	every 6 hrs	HTTPS
Sync/Shiny/Docker logs	internal pushes to external	rclone (sftp)	hourly	SSH
Heartbeat	internal pushes to external	rclone (sftp)	every 5 min	SSH

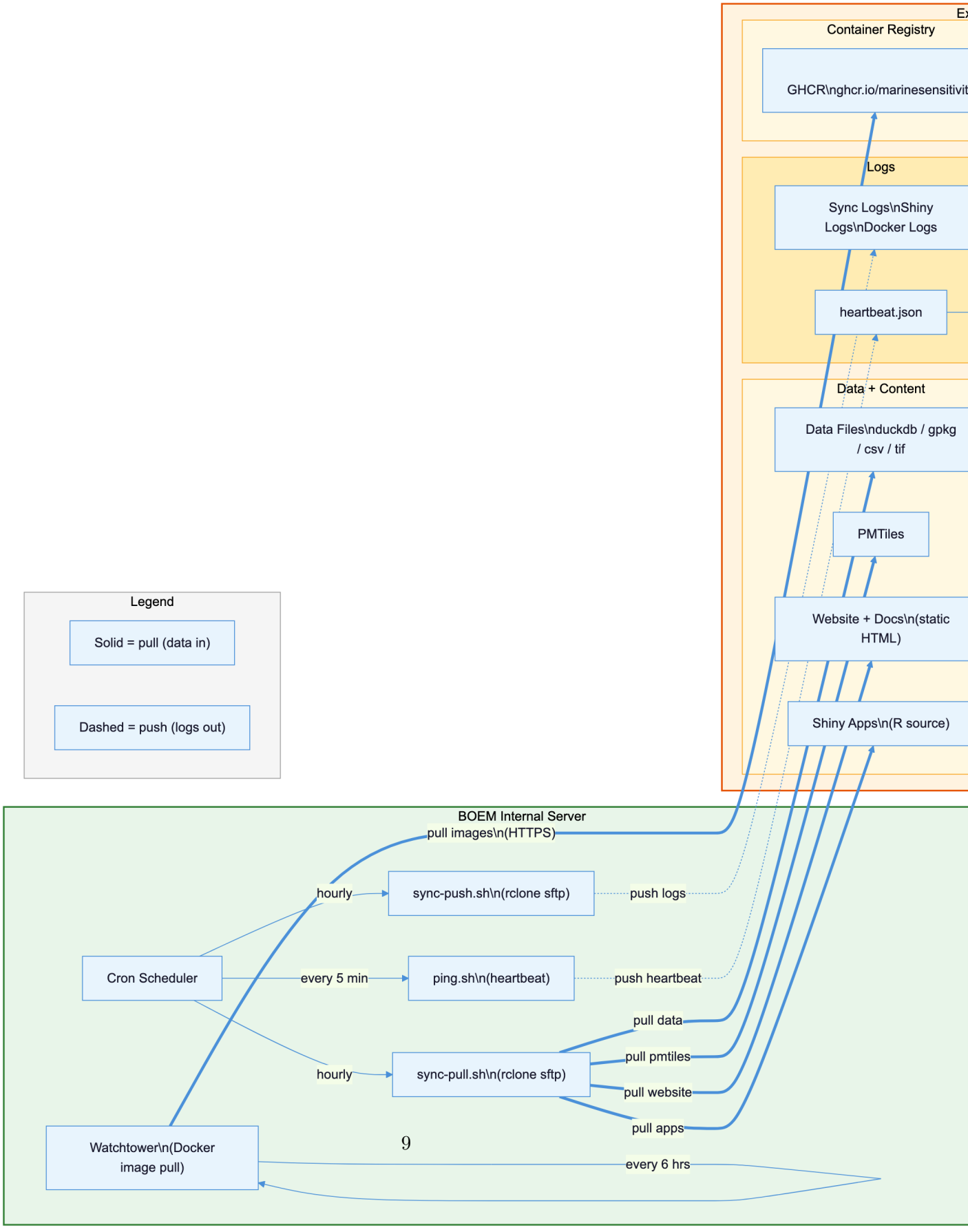

Figure 3.1: Synchronization flow between internal BOEM server and external AWS server. All connections are initiated by the internal server (outbound only).

4 Implementation

4.1 Prerequisites

On the **internal BOEM server** (Red Hat Linux):

```
# install rclone
sudo yum install -y rclone

# install docker and docker-compose (if not already)
sudo yum install -y docker docker-compose-plugin
sudo systemctl enable docker
sudo systemctl start docker

# create log directory
sudo mkdir -p /var/log/msens
sudo chown $USER:$USER /var/log/msens

# create data directories
sudo mkdir -p /share/data /share/public/www /share/shiny_apps /share/logs
```

4.2 SSH key setup

Generate an SSH key pair on the internal server for passwordless authentication to the external server:

```
# on internal server
ssh-keygen -t ed25519 -f ~/.ssh/msens_sync -N "" -C "msens-sync@boem"

# copy public key to external server (one-time manual step)
ssh-copy-id -i ~/.ssh/msens_sync.pub ubuntu@msens1.marinesensitivity.org
```

4.3 rclone configuration

Configure rclone on the internal server to connect to the external server via SFTP:

```
# create rclone config
rclone config

# or write directly:
cat >> ~/.config/rclone/rclone.conf <<'EOF'
[ext_dev]
type = sftp
host = msens1.marinesensitivity.org
user = ubuntu
key_file = /home/<user>/.ssh/msens_sync
shell_type = unix
EOF
```

Test the connection:

```
rclone lsd ext_dev:/share/data/derived/
```

4.4 Sync scripts

The sync scripts live in `prod/` and handle pulling data and pushing logs:

- **`prod/sync-pull.sh`** — pulls data files, PMTiles, static website, and Shiny apps from the external server
- **`prod/sync-push.sh`** — pushes sync logs, Shiny logs, and Docker logs to the external server for remote debugging
- **`prod/ping.sh`** — sends a heartbeat JSON with system health info every 5 minutes

Make scripts executable:

```
chmod +x prod/sync-pull.sh prod/sync-push.sh prod/ping.sh
```

4.5 Watchtower (Docker image sync)

Watchtower automatically monitors running Docker containers and pulls updated images from the container registry. The configuration is in `prod/watchtower.yml`.

Start Watchtower alongside the production stack:

```
cd /share/github/server/prod

# start production services
docker compose up -d

# start watchtower
docker compose -f watchtower.yml up -d
```

When the developer pushes a new Shiny image to GitHub Container Registry (GHCR), Watchtower will detect the update within 6 hours and automatically pull the new image, stop the old container, and start a new one.

4.6 Cron schedule

Add the following to the internal server's crontab (`crontab -e`):

```
# marine sensitivity toolkit sync jobs
# pull data and content from external server (hourly at :05)
5 * * * * /share/github/server/prod/sync-pull.sh

# push logs to external server (hourly at :35)
35 * * * * /share/github/server/prod/sync-push.sh

# heartbeat ping (every 5 minutes)
*/5 * * * * /share/github/server/prod/ping.sh
```

4.7 Monitoring and alerting

The heartbeat strategy uses a push model — the internal server pushes a `heartbeat.json` file to the external server every 5 minutes. The external server runs a monitor script that checks the heartbeat age and sends email alerts if it goes stale.

On the **external server**, add to crontab:

```
# check internal server heartbeat (every 10 minutes)
*/10 * * * * /share/github/server/prod/monitor-heartbeat.sh
```

The heartbeat JSON includes:

```
{
  "timestamp":    "2026-02-19T14:30:00Z",
  "hostname":     "boem-msens-prod",
  "uptime_since": "2026-02-01 08:00:00",
  "disk_free":    "142G",
  "mem_free":     "8.2G",
  "services": {
    "caddy": "running",
    "shiny": "running"
  }
}
```

If no heartbeat arrives for 15 minutes, the monitor sends an email alert with the last known service states.

5 Production Stack

The production Docker Compose configuration (`prod/docker-compose.yml`) runs only the essential services:

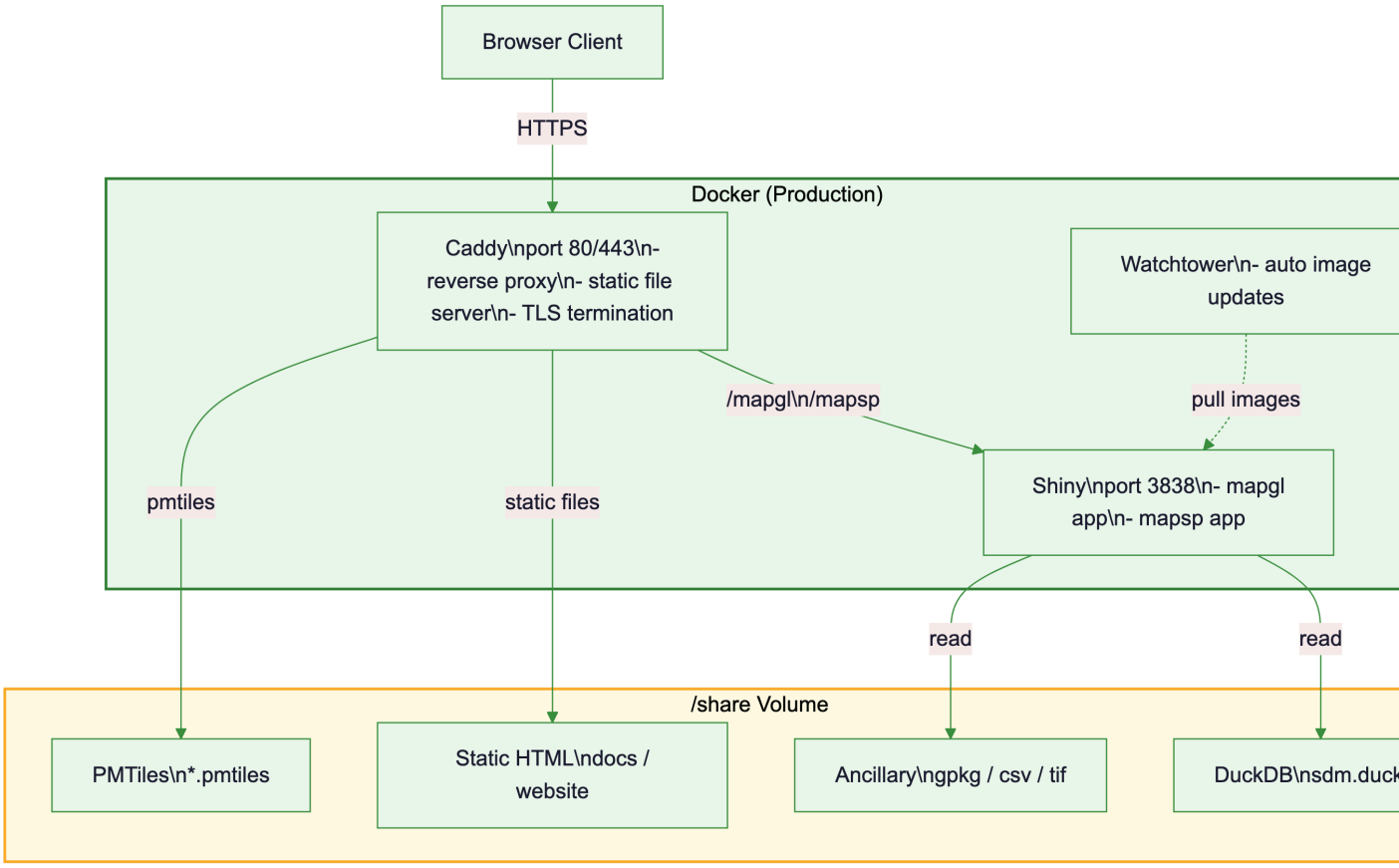

Figure 5.1: Production server Docker services and data flow. Caddy serves static content and proxies Shiny. Data files are mounted as volumes.

5.1 Deployment

```
# clone server repo on internal server
mkdir -p /share/github
cd /share/github
git clone https://github.com/MarineSensitivity/server.git
cd server/prod

# start services
docker compose up -d
docker compose -f watchtower.yml up -d

# verify
docker ps
curl -s http://localhost:3838 | head -5
```

5.2 Migrating static web content

The project website and documentation currently served by GitHub Pages can be served internally by Caddy as static files:

Content	Current URL	Internal path
Project homepage	marinesensitivity.org	/share/public/www/
Project documentation	marinesensitivity.org/docs	/share/public/www/docs/

The `sync-pull.sh` script automatically syncs these from the external server where they are built by GitHub Actions and deployed to `/share/public/www/`.

6 Government Compliance

This section reviews the relevant IT policy statutes and guidelines applicable to BOEM (Bureau of Ocean Energy Management), the Department of the Interior (DOI), and federal .gov requirements, and describes how the proposed synchronization arrangement is acceptable.

6.1 Federal Information Security Modernization Act (FISMA)

FISMA requires federal agencies to implement information security programs. The proposed arrangement is compliant because:

- **All connections are outbound from the BOEM network** — the internal server initiates all SSH/SFTP transfers to the external server. No inbound ports need to be opened on the BOEM firewall.
- **Data classification** — the MST data (species distribution models, sensitivity scores) is **publicly available scientific data** with no Controlled Unclassified Information (CUI) or Personally Identifiable Information (PII). All data will eventually be published openly.
- **Encryption in transit** — all transfers use SSH (SFTP) with Ed25519 key authentication, providing FIPS 140-2 compliant encryption.
- **Principle of least privilege** — the sync account on the external server has read-only access to data directories and write-only access to log directories.

6.2 NIST SP 800-53 (Security and Privacy Controls)

The NIST 800-53 framework applies to DOI systems. Relevant control families:

Control Family	Control	Compliance Approach
AC (Access Control)	AC-3, AC-6	SSH key-based auth, least privilege, no shared credentials
AU (Audit)	AU-2, AU-3	All sync operations logged with timestamps; logs pushed to external for review

Control Family	Control	Compliance Approach
CM (Configuration Mgmt)	CM-2, CM-3	Docker containers provide immutable, versioned configurations; Watchtower only updates from trusted GHCR registry
IA (Identification & Auth)	IA-2, IA-5	Ed25519 SSH keys (no passwords); keys stored with restricted permissions
SC (System & Comms Protection)	SC-8, SC-13	All data in transit encrypted via SSH; TLS for HTTPS
SI (System & Info Integrity)	SI-2, SI-3	Red Hat `yum` updates for OS patching; Docker images scanned upstream

6.3 DOI Information Security Policy

The DOI Departmental Manual Part 375 and DOI Cybersecurity Program require:

- **Authority to Operate (ATO)** — the internal server should be registered in the DOI System Inventory and receive an ATO. The low-risk nature of the data (public scientific information) supports a streamlined assessment.
- **Continuous monitoring** — the heartbeat mechanism and log synchronization provide continuous visibility into server health without requiring inbound network access.
- **Incident response** — if the external server is compromised, the internal server's pull-only model limits exposure: it only reads data files and Docker images from trusted sources (GHCR, known SFTP endpoint).

6.4 FedRAMP and Cloud Services

The external development server runs on **AWS**, which holds a FedRAMP authorization at the High impact level. This means:

- The cloud infrastructure meets federal security requirements
- AWS provides the physical and infrastructure security controls
- The MST application layer (Docker containers, data files) operates within this authorized environment

The internal server is **not a cloud service** — it is a BOEM-managed on-premises server. The synchronization only reads from the FedRAMP-authorized AWS environment.

6.5 Trusted Internet Connection (TIC) 3.0

TIC 3.0 guidance from CISA allows for more flexible network architectures than the original TIC model. The proposed arrangement:

- Uses **outbound-only connections** from the BOEM network, consistent with TIC guidance allowing authorized outbound traffic
- Does not require opening inbound ports or creating network exceptions
- SSH/SFTP traffic can be monitored by existing BOEM network security appliances (firewalls, IDS/IPS)

6.6 BOEM-Specific Considerations

- **Data sensitivity**: all MST data is **non-sensitive, publicly releasable** scientific information. Species distribution models, extinction risk scores, and sensitivity metrics are derived from publicly available datasets (AquaMaps, IUCN, NOAA, USFWS).
- **Contractor access**: the lead contractor maintains the external development server under the existing BOEM contract. Development artifacts are delivered to BOEM via the synchronization mechanism described here.
- **Open source**: all MST source code is published on GitHub under MIT license, consistent with federal open-source policies (M-16-21, Federal Source Code Policy).

6.7 Compliance summary

Requirement	Status	Notes
FISMA compliance	Compliant	outbound-only connections, encrypted, public data
NIST 800-53 controls	Addressed	see control mapping above
DOI security policy	Requires ATO	low-risk system; streamlined assessment recommended
FedRAMP (external)	Compliant	AWS holds FedRAMP High authorization
TIC 3.0	Compliant	outbound SSH/HTTPS only

Requirement	Status	Notes
Data classification	Low risk	public scientific data, no CUI/PII
Open source policy	Compliant	MIT-licensed on GitHub

7 Issues

The following items require clarification or action before full deployment:

7.1 1. Operating system compatibility

The internal BOEM server runs **Red Hat Enterprise Linux** (RHEL) with `yum`-based package management, while the external development server runs **Ubuntu** with `apt`. This should **not be a concern** because:

- All application services run inside Docker containers, which are OS-independent
- Only host-level tools differ: `rclone`, `docker`, `cron` — all available on both platforms
- Docker images built on Ubuntu will run identically on a RHEL host
- OS updates (via `yum` on RHEL, `apt` on Ubuntu) are independent of the containerized services

Action: confirm RHEL version and ensure Docker CE/EE is installed and running.

7.2 2. Internal web server access (firewall rules)

It appears that firewall rules on the BOEM network restrict server access to web content ports. To serve the MST applications internally:

- **Port 80 (HTTP)** and **port 443 (HTTPS)** need to be open for inbound traffic from BOEM users (or at least the BOEM internal network)
- On RHEL, this may involve `firewalld` (preferred) or `iptables`:

```
# firewalld (recommended on RHEL)
sudo firewall-cmd --permanent --add-service=http
sudo firewall-cmd --permanent --add-service=https
sudo firewall-cmd --reload

# or iptables
sudo iptables -A INPUT -p tcp --dport 80 -j ACCEPT
```

```
sudo iptables -A INPUT -p tcp --dport 443 -j ACCEPT
sudo iptables-save
```

- **Internal DNS**: BOEM IT will need to create internal DNS records pointing a hostname (e.g., `msens.boem.gov` or `marinesensitivity.boem.gov`) to the internal server's IP address
- If Caddy cannot obtain Let's Encrypt certificates (due to no external access), use internal CA certificates or configure Caddy for HTTP-only internally

Action: coordinate with BOEM IT to open ports 80/443 for internal traffic and configure internal DNS.

7.3 3. External access and domain management

The `marinesensitivity.org` domain is currently managed by the lead contractor through SquareSpace. Questions to resolve:

- **Keep marinesensitivity.org?** The domain is well-established and used in publications. Advantages: continuity, contractor can manage DNS. Disadvantages: not a .gov domain, may need to transition at contract end.
- **Use a BOEM.gov subdomain?** (e.g., `marinesensitivity.boem.gov` or `mst.boem.gov`). Advantages: clearly government-affiliated, managed by BOEM IT. Disadvantages: requires DOI IT approval, potential delays.
- **Hybrid approach?** Use `marinesensitivity.org` externally (public-facing, contractor-managed) and a `.boem.gov` subdomain internally. The internal server would serve the same content under both names.

Action: decide on domain strategy and coordinate with BOEM IT and SquareSpace as needed.

7.4 4. SSH outbound access from BOEM network

The synchronization depends on the internal server being able to make **outbound SSH connections** (port 22) to the external server. If the BOEM network restricts outbound SSH:

- Request a firewall exception for outbound SSH to the specific AWS IP (`100.25.173.0`)
- Alternative: use HTTPS-based sync (e.g., rclone with WebDAV or S3) if SSH is blocked
- Alternative: use an SSH tunnel over HTTPS (port 443) if only HTTPS outbound is allowed

Action: confirm outbound SSH (port 22) is permitted from the internal server to the external server IP.

7.5 5. Docker registry access

Watchtower needs **outbound HTTPS access** to GitHub Container Registry (`ghcr.io`) to pull updated Docker images. If HTTPS to external registries is restricted:

- Request firewall exception for `ghcr.io` (IP ranges published by GitHub)
- Alternative: manually transfer Docker images via `docker save` / `docker load` and rclone

Action: confirm outbound HTTPS to `ghcr.io` is permitted.

7.6 6. Data volume sizing

The MST data footprint should be estimated for disk provisioning on the internal server:

Data type	Estimated size	Growth rate
DuckDB (sdm.duckdb)	~5-10 GB	slow (model updates)
PMTiles	~1-2 GB	slow (boundary updates)
GeoPackage/CSV/TIF	~2-5 GB	slow
Static website/docs	~100 MB	moderate (doc updates)
Docker images	~3-5 GB	moderate (app updates)
Logs	~100 MB	steady
Total	**~15-25 GB**	

Action: confirm `/share` volume has sufficient capacity (recommend 50+ GB for headroom).

7.7 7. Backup strategy for internal server

While the external server has automated PostgreSQL backups, the internal production server should also have a backup plan:

- DuckDB files can be backed up by the regular sync (authoritative copy is on external)
- Docker volumes (Caddy certificates, config) should be included in any BOEM-managed backup system
- The `/share` directory should be on redundant storage or included in enterprise backup

Action: coordinate with BOEM IT on backup integration.

7.8 8. Caddy TLS certificates on internal network

Caddy automatically provisions TLS certificates via Let's Encrypt, but this requires:

- External DNS resolution (the domain must resolve publicly)
- HTTP challenge (port 80 accessible from the internet) or DNS challenge

If the internal server is not internet-accessible:

- Use Caddy's internal TLS for self-signed certificates
- Use a BOEM-managed internal CA
- Or serve HTTP-only if the server is only accessible from the BOEM intranet

Action: determine TLS strategy based on whether the internal server will be internet-facing.