# Digital Slide Archive on Ubuntu 24.04: local install from scratch

This guide walks through a complete local install of the **Digital Slide Archive (DSA)** on Ubuntu 24.04 using the `devops/ver5` Docker Compose stack. It assumes Docker is already installed and working.

**Result:** a DSA 5 instance at `http://localhost:8080`, with Girder, HistomicsUI, large_image, DICOMweb, Slicer CLI Web, the worker, MongoDB, RabbitMQ, and Redis all running in Docker.

### What was used for this walkthrough

| Item | Value |
|---|---|
| Host OS | Ubuntu 24.04.5 LTS (kernel 6.8.0-147-generic, x86_64) |
| Docker Engine | 29.9.2 |
| Docker Compose | v5.6.1 |
| Repository | `https://github.com/digitalSlideArchive/digital_slide_archive` |
| Checked-out commit | `2e16b9a` (Merge pull request #538 from DigitalSlideArchive/update-trivy) |
| Install directory | `~/devel/digital-slide-archive` |
| DSA version reported by the API | `5.0.23.dev5+g70eb94dcd` |
| Web address | `http://localhost:8080` |

---

## 1. Check Docker and the host

Run these commands first. They confirm the OS, the Docker engine, the Compose plugin, and that your user can talk to the Docker daemon.

Verify Docker and Ubuntu

```
ubuntu@dsa-host:~$ lsb_release -ds
Ubuntu 24.04.5 LTS

ubuntu@dsa-host:~$ uname -srm
Linux 6.8.0-147-generic x86_64

ubuntu@dsa-host:~$ docker --version
Docker version 29.9.2, build a82e528

ubuntu@dsa-host:~$ docker compose version
Docker Compose version v5.6.1

ubuntu@dsa-host:~$ docker info --format 'Server {{.ServerVersion}} | storage {{.Driver}} | root {{.DockerRootDir}}'
Server 29.9.2 | storage overlayfs | root /var/lib/docker
```

```
lsb_release -ds
uname -srm
docker --version
docker compose version
docker info --format 'Server {{.ServerVersion}} | storage {{.Driver}} | root {{.DockerRootDir}}'
```

Expected: Ubuntu 24.04.x, Docker 2x or newer, and a Compose v2+ plugin.

### If you get "permission denied" on the Docker socket

A freshly installed Docker daemon only lets members of the `docker` group use it. If `docker info` fails with `permission denied while trying to connect to the Docker API at unix:///var/run/docker.sock`, add yourself to the group and start a fresh login session:

```
sudo usermod -aG docker "$USER"
# log out and back in, or run:
newgrp docker
```

Verify again with `docker info`. You should not need `sudo` for the rest of this guide.

### Hardware and disk

- 2 CPU cores and 8 GB RAM is enough to boot the stack and click around.
- Plan on roughly 8–12 GB of disk for the container images plus the database and assetstore.
- The first image pull is the slow part. Later starts are fast because the images are cached.

---

## 2. Clone the repository

DSA keeps its deployment files in the same repository as the application. Clone it into `~/devel`:



```
mkdir -p ~/devel
cd ~/devel
git clone https://github.com/digitalSlideArchive/digital_slide_archive.git digital-slide-archive
cd digital-slide-archive
git log --oneline -1
```

The result is `~/devel/digital-slide-archive`. This guide was written against commit `2e16b9a`.

---

## 3. Look at the `devops/ver5` stack

The version 5 Compose stack lives in `devops/ver5`:

The devops/ver5 compose stack

```
$ cd /home/ubuntu/devel/digital-slide-archive/devops/ver5 && ls -la
total 96
drwxrwxr-x  5 ubuntu docker  4096 Nov  2 14:15 .
drwxrwxr-x 12 ubuntu docker  4096 Nov  2 14:15 ..
drwxrwxr-x  2 ubuntu docker  4096 Nov  2 14:15 assetstore
drwxrwxr-x  2 ubuntu docker  4096 Nov  2 14:15 db
-rw-rw-r--  1 ubuntu docker  8413 Nov  2 14:15 docker-compose.yml
drwxrwxr-x  2 ubuntu docker  4096 Nov  2 14:15 logs
-rwxrwxr-x  1 ubuntu docker 31492 Nov  2 14:15 provision.py
-rw-rw-r--  1 ubuntu docker  4121 Nov  2 14:15 provision.yaml
-rw-rw-r--  1 ubuntu docker    50 Nov  2 14:15 rabbitmq.advanced.config
-rw-rw-r--  1 ubuntu docker    59 Nov  2 14:15 rabbitmq.conf
-rwxrwxr-x  1 ubuntu docker  3475 Nov  2 14:15 start_dockerlog.py
-rwxrwxr-x  1 ubuntu docker  3878 Nov  2 14:15 start_girder.sh
-rwxrwxr-x  1 ubuntu docker  1804 Nov  2 14:15 start_redis.sh
-rwxrwxr-x  1 ubuntu docker  2191 Nov  2 14:15 start_worker.sh

$ cd /home/ubuntu/devel/digital-slide-archive/devops/ver5 && grep -n 'DSA_PORT\|DSA_USER\|8080' docker-compose.yml | head -20
24:    # Set DSA_USER to a user id that is part of the docker group (e.g.,
25:    # `DSA_USER=$(id -u):$(id -g)`).  This makes files in assetstores and logs
28:      DSA_USER: ${DSA_USER:-}
57:    # Set DSA_PORT to expose the interface on another port (default 8080).
59:      - "${DSA_PORT:-8080}:8080"
81:      test: ["CMD", "curl", "-f", "http://localhost:8080/api/v1/system/version"]
88:    # Set DSA_USER to your user id (e.g., `DSA_USER=$(id -u):$(id -g)`)
90:    user: ${DSA_USER:-PLEASE SET DSA_USER}
145:    # Set DSA_USER to a user id that is part of the docker group (e.g.,
146:    # `DSA_USER=$(id -u):$(id -g)`).  This provides permissions to manage
149:      DSA_USER: ${DSA_USER:-}
208:    user: ${DSA_USER:-PLEASE SET DSA_USER}
```

```
cd ~/devel/digital-slide-archive/devops/ver5
ls -la
```

Important files and directories:

| Path | Purpose |
| --- | --- |
| `docker-compose.yml` | Defines the `girder`, `worker`, `mongodb`, `rabbitmq`, `redis`, and `logging` services |
| `provision.py`, `provision.yaml` | Creates the initial admin user, collections, and plugin settings |
| `assetstore/` | Bind-mounted assetstore for uploaded files |
| `db/` | Bind-mounted MongoDB data directory |
| `logs/` | Bind-mounted application logs |
| `start_girder.sh`, `start_worker.sh` | Container entry points that honor `DSA_USER` |

Two details in `docker-compose.yml` matter for a local install:

- Girder is published on `${DSA_PORT:-8080}:8080`. The default is `http://localhost:8080`.
- `DSA_USER` controls the UID/GID used for files written to `assetstore/`, `db/`, and `logs/`. If it is omitted, files can end up owned by `root`.

---

## 4. Create the start script

Create a small `start.sh` next to the Compose file. It always brings the stack down first, then starts it detached as your user.

Create the start script

```
$ cat start.sh
#!/usr/bin/env bash
# Start (or restart) the Digital Slide Archive from this directory.
# Usage: ./start.sh
set -euo pipefail
cd "$(dirname "$0")"

# Stop anything already running so the stack starts from a clean state.
docker compose down

# Run the containers as the current user so database, assetstore, and log
# files are owned by you instead of root.
DSA_USER="$(id -u):$(id -g)" docker compose up -d

docker compose ps

$ chmod +x start.sh
```

```
cat > start.sh <<'EOF'
#!/usr/bin/env bash
# Start (or restart) the Digital Slide Archive from this directory.
# Usage: ./start.sh
set -euo pipefail
cd "$(dirname "$0")"

# Stop anything already running so the stack starts from a clean state.
docker compose down

# Run the containers as the current user so database, assetstore, and log
# files are owned by you instead of root.
DSA_USER="$(id -u):$(id -g)" docker compose up -d

docker compose ps
EOF

chmod +x start.sh
```

Why these two lines:

- `docker compose down` makes the script safe to run repeatedly. It removes the containers and network but keeps the bind-mounted data.
- `DSA_USER="$(id -u):$(id -g)"` passes your host UID/GID into the containers so the bind-mounted files stay owned by you.
- `up -d` runs in the background. The command returns before Girder has finished starting.

---

## 5. Pull the images

Pull once before the first start. This avoids waiting for a build and makes the first boot much more predictable.

Pull the container images

```
$ cd /home/ubuntu/devel/digital-slide-archive/devops/ver5 && docker compose pull
 Image redis Pulling
 Image dsarchive/dsa_common_5 Pulling
 Image mongo:latest Pulling
 Image rabbitmq:latest Pulling
 Image python:3.13-slim Pulling
 Image python:3.13-slim Pulled
 Image redis Pulled
 Image mongo:latest Pulled
 Image rabbitmq:latest Pulled
 Image dsarchive/dsa_common_5 Pulled
```

```
docker compose pull
```

The stack uses:

- `dsarchive/dsa_common_5` for both `girder` and `worker`
- `mongo:latest`
- `rabbitmq:latest`
- `redis`
- `python:3.13-slim` for the container-log collector

The DSA image is the large one. Expect several minutes and several GB on a first pull; on later runs this step is quick because the layers are cached.

---

## 6. Start the stack

Run the script:

Start the stack

```
$ ./start.sh
 Network ver5_default Creating
 Network ver5_default Creating
 Network ver5_default Created
 Network ver5_default Created
 Container ver5-rabbitmq-1 Creating
 Container ver5-redis-1 Creating
 Container ver5-mongodb-1 Creating
 Container ver5-redis-1 Created
 Container ver5-rabbitmq-1 Created
 Container ver5-worker-1 Creating
 Container ver5-mongodb-1 Created
 Container ver5-girder-1 Creating
 Container ver5-worker-1 Created
 Container ver5-girder-1 Created
 Container ver5-logging-1 Creating
 Container ver5-logging-1 Created
 Container ver5-redis-1 Starting
 Container ver5-rabbitmq-1 Starting
 Container ver5-mongodb-1 Starting
 Container ver5-redis-1 Started
 Container ver5-rabbitmq-1 Started
 Container ver5-worker-1 Starting
 Container ver5-mongodb-1 Started
 Container ver5-girder-1 Starting
 Container ver5-worker-1 Started
 Container ver5-girder-1 Started
 Container ver5-logging-1 Starting
 Container ver5-logging-1 Started
NAME             IMAGE                    COMMAND                  SERVICE    CREATED         STATUS                            PORTS
ver5-girder-1    dsarchive/dsa_common_5   "tini -- /opt/digita…"   girder     4 seconds ago   Up 1 second (health: starting)    0.0.0.0:8080-
>8080/tcp, [::]:8080->8080/tcp
ver5-logging-1   python:3.13-slim         "bash -c 'export PYT…"   logging    3 seconds ago   Up Less than a second
ver5-mongodb-1   mongo:latest             "docker-entrypoint.s…"   mongodb    5 seconds ago   Up 1 second (health: starting)    27017/tcp
ver5-rabbitmq-1  rabbitmq:latest          "docker-entrypoint.s…"   rabbitmq   5 seconds ago   Up 2 seconds (health: starting)   4369/tcp, 5671-
5672/tcp, 15691-15692/tcp, 25672/tcp
ver5-redis-1     redis                    "/usr/start_redis.sh"    redis      5 seconds ago   Up 2 seconds (health: starting)   6379/tcp
ver5-worker-1    dsarchive/dsa_common_5   "tini -- /opt/digita…"   worker     4 seconds ago   Up 1 second (health: starting)
```

```
./start.sh
```

You will see Compose create the network and the six containers:

| Container | Role |
|---|---|
| `ver5-girder-1` | Web application and REST API (port 8080) |
| `ver5-worker-1` | Girder Worker / Celery worker for analysis jobs |
| `ver5-mongodb-1` | Database |
| `ver5-rabbitmq-1` | Broker for the worker |
| `ver5-redis-1` | Cache and notification backend |
| `ver5-logging-1` | Collects container logs into `logs/` |

On the first run, Girder also creates the admin user and the default collections. Give it a minute or two.

> **A note about health status:** `docker compose ps` may report `health: starting` for Girder for several minutes because the healthcheck interval in the Compose file is five minutes. The application can already be serving requests while Docker still shows `starting`. Confirm readiness with the API check below rather than waiting on the label.

---

## 7. Verify that DSA is up

Check the containers and ask the API for its version:




```
docker compose ps
curl -s http://localhost:8080/api/v1/system/version
```

A healthy API response looks like:

```
{"release": "5.0.23.dev5+g70eb94dcd", "serverStartDate": "2026-11-02T14:15:45.347235+00:00"}
```

If the `curl` command hangs or refuses the connection, give Girder another minute and check its log:

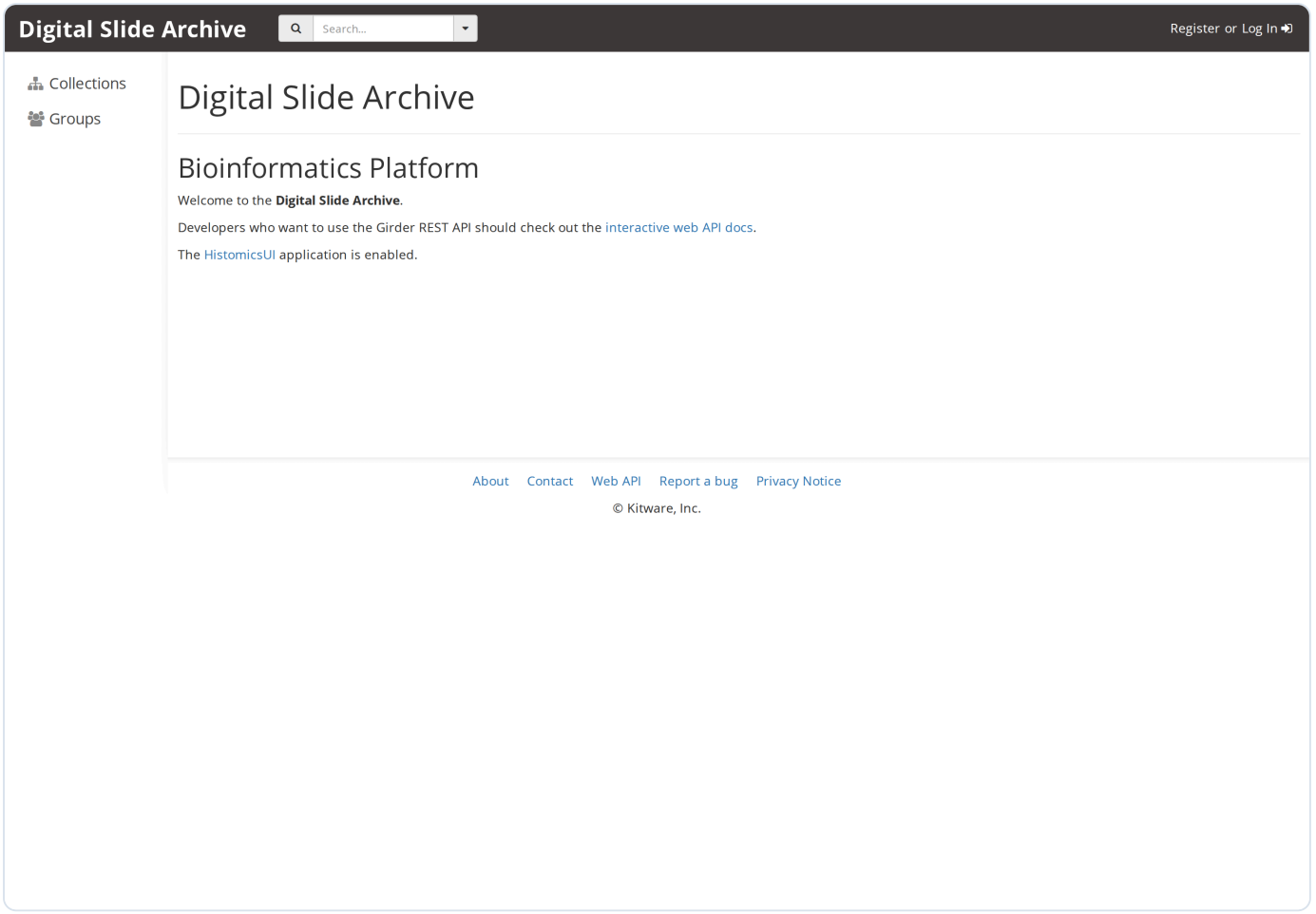
```
docker compose logs --tail=80 girder
```

Look for `==== Starting Girder ====` and the plugin list. A healthy first boot loads `large_image`, `large_image_annotation`, `histomicsui`, `dicomweb`, `slicer_cli_web`, `worker`, and the other DSA plugins.

---

## 8. Open DSA in a browser

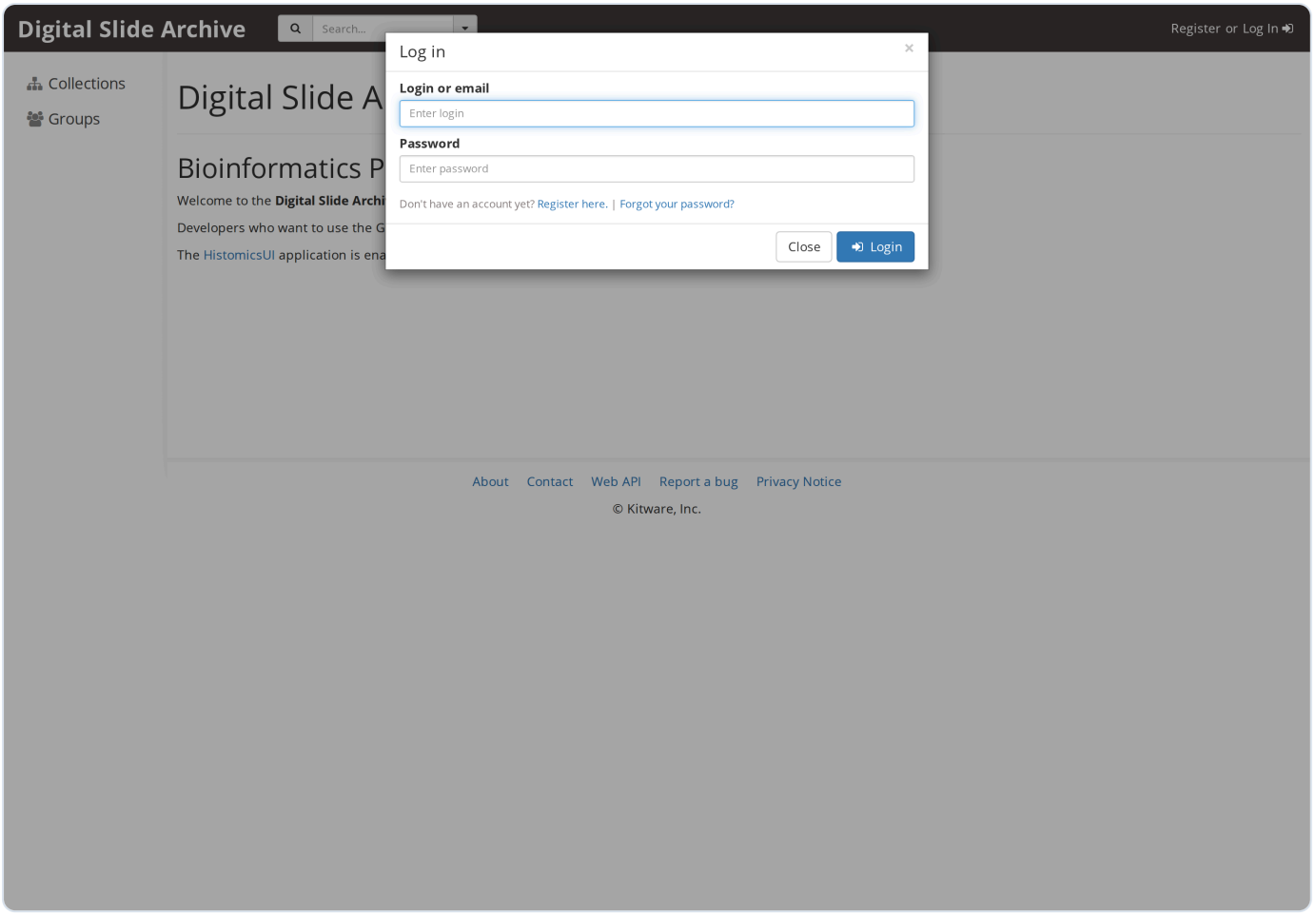
Browse to **http://localhost:8080**.



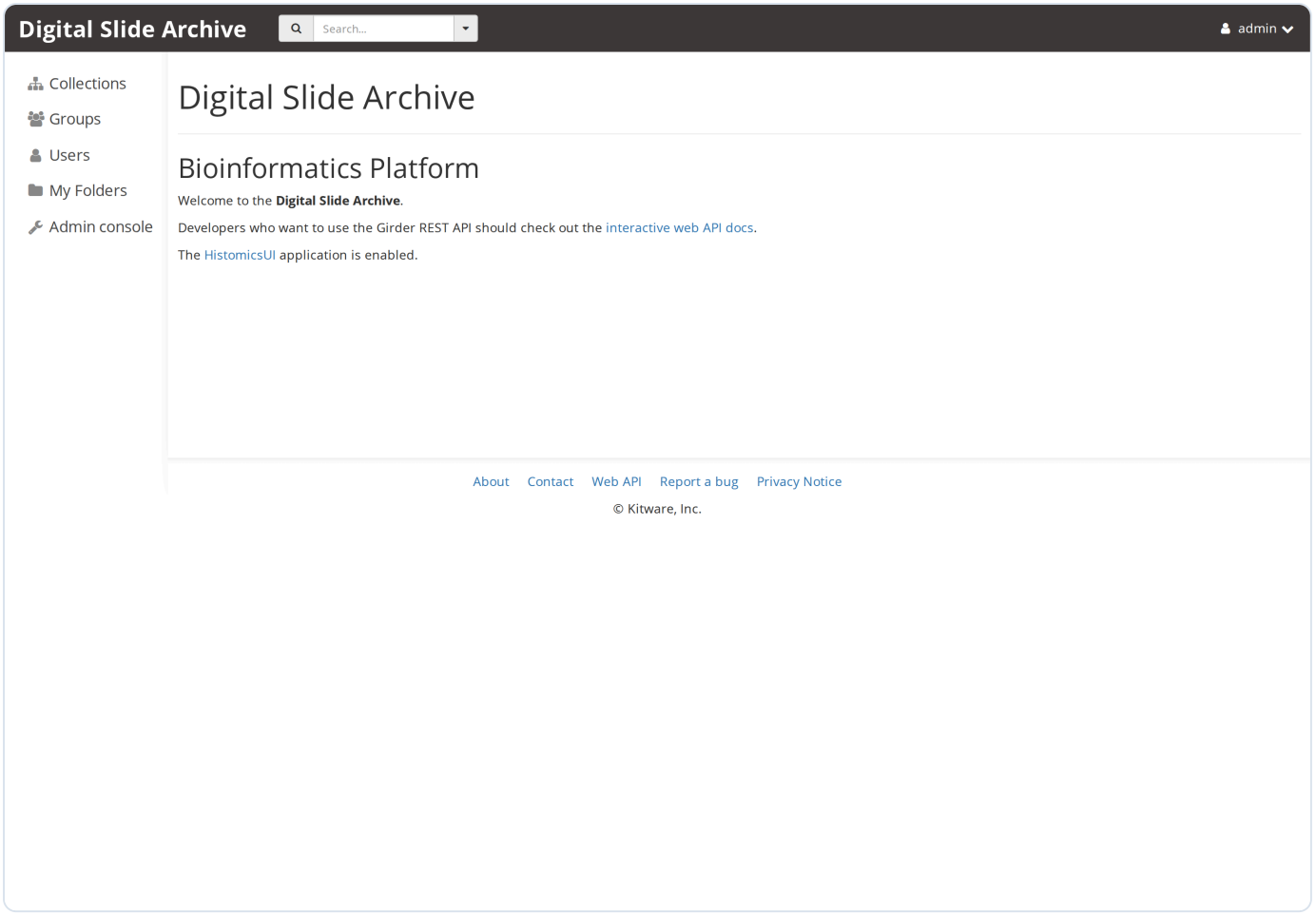

The stock `devops/ver5` provisioning creates one admin account:

| Username | Password |
|---|---|
| `admin` | `password` |

Click **Log In**:



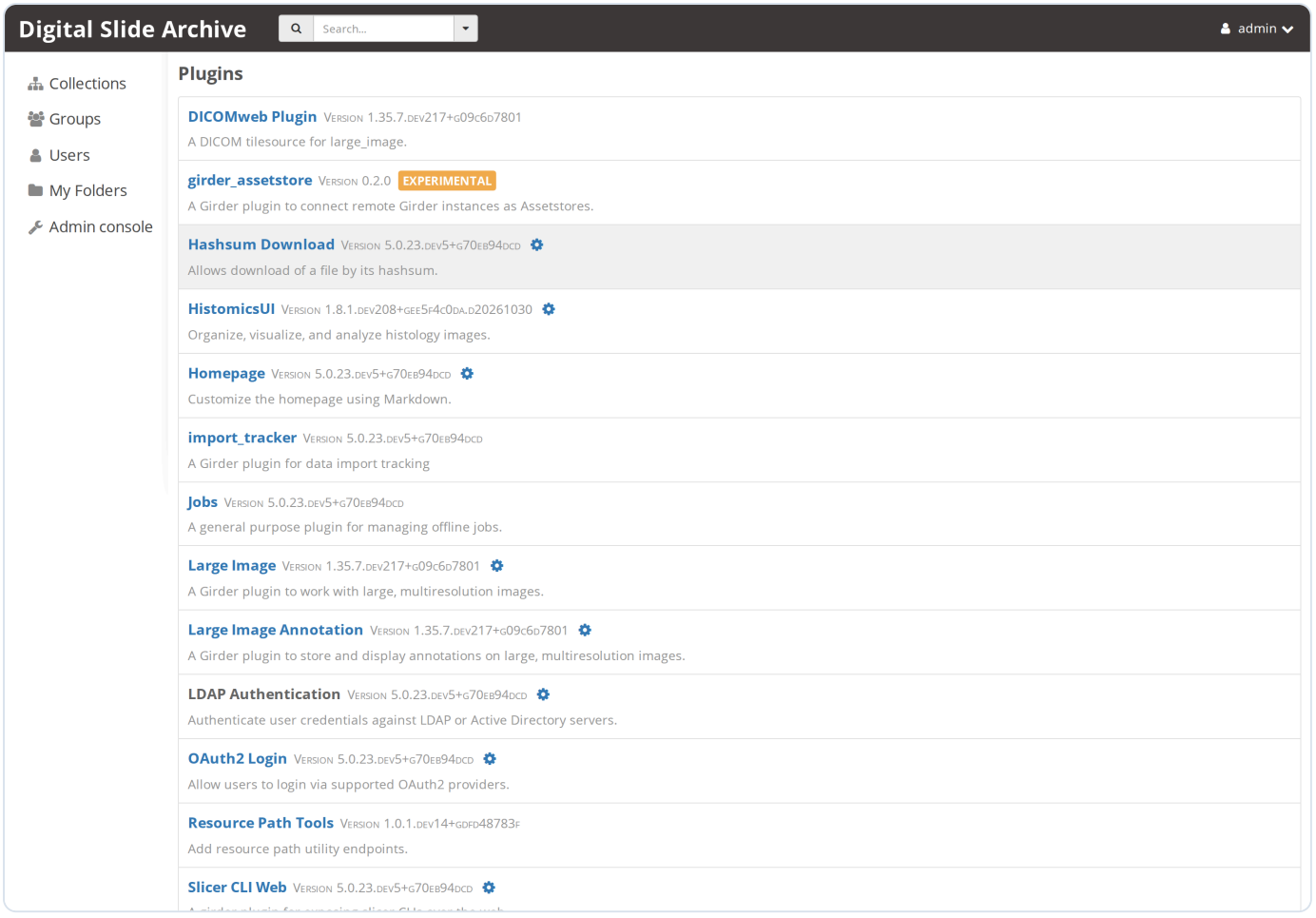

After signing in, the navigation bar shows the admin console, collections, groups, and users:




> Change the default admin password before putting this instance on a shared network. Use **My account → Password** in the web UI.

## Verify the plugins

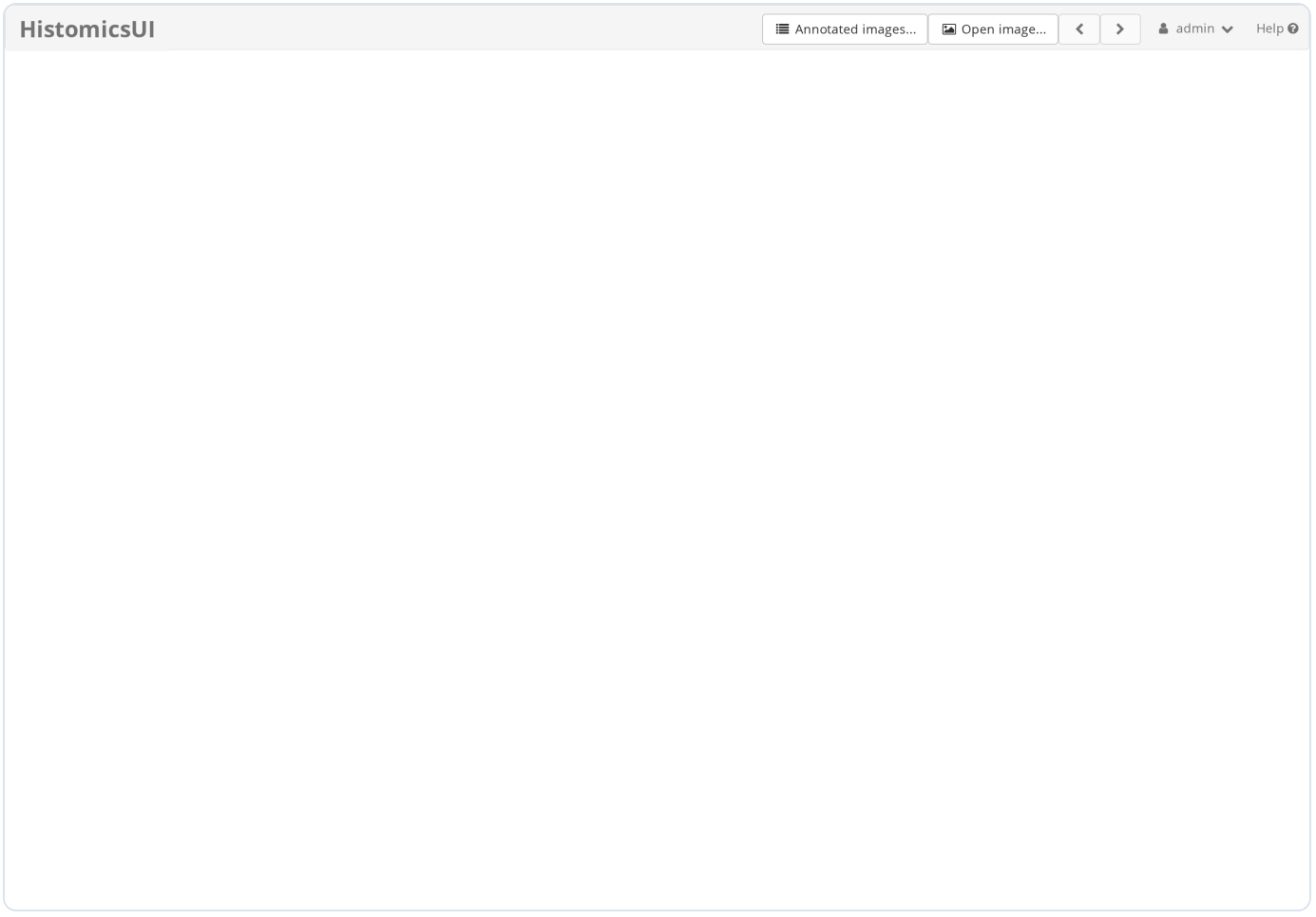
Open **Admin console → Plugins** (or go directly to `http://localhost:8080/#plugins`). This page shows the installed plugin set and the DSA version:




You should see `HistomicsUI`, `large_image`, `DICOMweb`, `Slicer CLI Web`, `Worker`, and the rest of the DSA plugin stack. The DSA release is shown at the top of the page.

## Open HistomicsUI

HistomicsUI is the digital-slide workspace and is the entry point for slide viewing and analysis:




Use **Open image…** after uploading a slide to open the viewer. For this install check, simply seeing the authenticated HistomicsUI landing page is enough to confirm that the web application, authentication, and plugin routing are working.

---

## 9. Day-to-day commands

Run these from `~/devel/digital-slide-archive/devops/ver5`.

| Task | Command |
|---|---|
| Start or restart everything | `./start.sh` |
| Start in the foreground (see all logs) | `DSA_USER=$(id -u):$(id -g) docker compose up` |
| Stop the stack | `docker compose down` |
| Stop and remove containers/network | `docker compose down` |
| Show status | `docker compose ps` |
| Follow Girder logs | `docker compose logs -f girder` |
| Follow worker logs | `docker compose logs -f worker` |
| Open a shell in Girder | `docker compose exec girder bash` |
| Check the API version | `curl -s http://localhost:8080/api/v1/system/version` |
| Use a different host port | `DSA_PORT=8081 ./start.sh` |

### Where the data lives

- Uploaded files: `devops/ver5/assetstore/`
- MongoDB data: `devops/ver5/db/`
- Application logs: `devops/ver5/logs/`

Because these are bind mounts, `docker compose down` does **not** delete them. To reset to a completely fresh instance, stop the stack and remove those three directories:

```
docker compose down
rm -rf assetstore/* db/* logs/*
./start.sh
```

This is destructive. Take a backup first if the instance has data you care about.

---

## 10. Troubleshooting

**`permission denied ... /var/run/docker.sock`** Your shell is not in the `docker` group yet. Run `sudo usermod -aG docker "$USER"`, then log out and back in (or `newgrp docker`), and retry.

**`Bind for 0.0.0.0:8080 failed: port is already allocated`** Something else is using port 8080. Start DSA on another port with `DSA_PORT=8081 ./start.sh`, then browse to `http://localhost:8081`.

**Girder stays on `health: starting`** This is expected for a while because the healthcheck interval is five minutes. Check `curl -s http://localhost:8080/api/v1/system/version` and `docker compose logs --tail=100 girder`. If the API responds, the instance is usable.

**The first start seems to hang** The first boot runs database provisioning, creates the admin user, and configures plugins. Watch `docker compose logs -f girder` and give it a few minutes. Later starts are much faster.

**MongoDB exits or complains about memory maps** The Compose file limits Mongo connections to reduce shared-memory demand. On a small host, make sure the machine has enough RAM and that `vm.max_map_count` is at least the Docker default (`262144`). Check with `sysctl vm.max_map_count`.

**Containers start but files in `assetstore/`, `db/`, or `logs/` are owned by root** You started the stack without `DSA_USER`. Use the provided `start.sh`, or pass `DSA_USER=$(id -u):$(id -g)` explicitly.

**I need to start completely over** Stop the stack, remove the bind-mounted data as shown above, and run `./start.sh`. Pulled images remain cached.

---

## 11. What this gives us for the DICOM plugin work

The stack already includes:

- `dicomweb` — DSA's DICOMweb plugin
- `large_image` and `large_image_annotation` — tile serving and annotation support
- `HistomicsUI` — the React/Vue front end where an inline viewer would live
- A running Girder REST API and worker for ingestion and processing

That means the new inline DICOM viewer can be developed against this instance without reinstalling anything. The likely integration points are the HistomicsUI item page, the DICOMweb endpoints exposed by the `dicomweb` plugin, and the existing `large_image` tile stack.

---

# Screenshot index

| Screenshot | What it shows |
|---|---|
| `01-prerequisites.png` | Ubuntu, kernel, Docker, and Compose versions |
| `02-clone-repo.png` | Repository clone and checked-out commit |
| `03-ver5-directory.png` | The `devops/ver5` stack and its Compose settings |
| `04-start-script.png` | The `start.sh` script |
| `05-pull-images.png` | Pulling the five container images |
| `06-start-containers.png` | Compose creating and starting the six containers |
| `07-container-health.png` | `docker compose ps` and the DSA version API |
| `10-dsa-landing.png` | DSA landing page at `localhost:8080` |
| `11-dsa-login.png` | Login dialog |
| `12-dsa-home-admin.png` | Signed-in home page with admin navigation |
| `13-dsa-plugins.png` | Installed plugins and DSA release |
| `14-histomicsui.png` | HistomicsUI landing page |