SCOPE

ON-PREM DOCUMENTATION

Manual Swarm Install (RHEL 9)

Version	3.99.1
Channel	on-prem
Mode	production
Revision	49f2343
Generated	2026-10-20

Confidential · For authorized Scope AR customers and partners

Contents

Manual Swarm Install (client-owned RHEL 9 VM)

- 0. Set variables
- 1. Prerequisites, sizing, and ports
  - Sizing
  - Update the base OS and install helpers
  - Open the required network ports
- 2. Install Docker and initialize Swarm
- 3. Install scopearctl
- 4. Create the directory tree
- 5. Generate the Traefik TLS certificate (self-signed)
- 6. Generate the swarm secrets
  - Apply ownership and permissions
- 7. Write the swarm configuration
  - Enabling optional components
- 8. Get the container images
  - 8a. Online (registry login)
  - 8b. Air-gapped (load from a saved tarball)
- 9. SELinux and host tuning
- 10. Deploy the stack
- 11. (Optional) Run as systemd services
  - Deploy-on-boot unit
  - Config UI unit (optional)
- 12. DNS / hosts entries and verification
  - Point the hostnames at the VM
  - Verify the stack is up
  - Find the generated CMS admin password

Appendix: what this guide intentionally omits vs. the packer build

Samples

- `scopear_swarm_env_sample.yaml`

Manual Swarm Install (client-owned RHEL 9 VM)

This guide installs and runs the full ScopeAR swarm stack on a client-owned Linux VM, step by step. It reproduces what the automated packer build does in `packer/scripts/swarm-setup.sh`, but as manual, copy-pasteable commands you can run on your own machine.

Every command below can be copied and pasted in order. Read section 0 first to set the few variables the rest of the guide depends on.

> Target OS: **RHEL 9 / Rocky Linux 9 / AlmaLinux 9** (uses `dnf`, `firewalld`, and SELinux). Run everything as a normal user that has `sudo`.

---

0. Set variables

Set these once in your shell. Every later step references them.

```
# The login user that will own /opt/scopear (defaults to the current user).
export SWARM_USER="$(whoami)"

# The scopearctl release version to install (ask Scope AR for the current one).
export VERSION="<SCOPEAR_VERSION>"      # e.g. 1.2.3
export CHANNEL="stable"

# Docker Hub credentials with access to the private docker.io/scopear images.
# Only needed for the ONLINE image path (section 8a). Leave blank for air-gapped.
export DOCKERHUB_USER="<your-dockerhub-username>"
export DOCKERHUB_TOKEN="<your-dockerhub-token-or-password>"
```

---

1. Prerequisites, sizing, and ports

Sizing

The packer reference build uses an 8 vCPU / large-memory instance with a 250 GB disk. For a client VM, provision at least:

- 8+ CPU cores
- 32 GB RAM (16 GB minimum, more if you enable AI/Superset)
- 200 GB+ free disk (the container images alone are tens of GB)

Update the base OS and install helpers

```
sudo dnf -y update
sudo dnf install -y epel-release
sudo dnf install -y htop curl openssl
```

Open the required network ports

The stack serves traffic on the following ports. Open them in `firewalld` (adjust if you use a cloud security group or external firewall instead):

PORT	PROTOCOL	PURPOSE
80	TCP	HTTP web traffic (Traefik ingress)
443	TCP	HTTPS web traffic (Traefik ingress)
8443	TCP	TCP tunnel service
3478	TCP & UDP	TURN service (coturn, if enabled)
24000	TCP & UDP	User presence service (zeus)
49152-50000	UDP	TURN relay ports (coturn, if enabled)
8883	TCP	IoT MQTT (only if IoT is enabled)
8080	TCP	Config UI (optional, section 11)

```
sudo firewall-cmd --permanent --add-port=80/tcp
sudo firewall-cmd --permanent --add-port=443/tcp
sudo firewall-cmd --permanent --add-port=8443/tcp
sudo firewall-cmd --permanent --add-port=3478/tcp
sudo firewall-cmd --permanent --add-port=3478/udp
sudo firewall-cmd --permanent --add-port=24000/tcp
sudo firewall-cmd --permanent --add-port=24000/udp
sudo firewall-cmd --permanent --add-port=49152-50000/udp
# Optional services:
sudo firewall-cmd --permanent --add-port=8883/tcp   # IoT MQTT
sudo firewall-cmd --permanent --add-port=8080/tcp   # Config UI
sudo firewall-cmd --reload
```

---

2. Install Docker and initialize Swarm

```
sudo dnf install -y dnf-plugins-core
sudo dnf config-manager --add-repo https://download.docker.com/linux/rhel/docker-ce.repo
sudo dnf install -y docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-
compose-plugin

sudo systemctl enable docker
sudo systemctl start docker

# Allow your user to run docker without sudo (re-login or run `newgrp docker`).
sudo usermod -aG docker "${SWARM_USER}"

# Initialize the single-node swarm.
sudo docker swarm init || echo "Swarm already initialized"
```

After `usermod`, open a new shell (or run `newgrp docker`) so group membership takes effect before continuing.

---

3. Install scopearctl

`scopearctl` is the CLI that renders the swarm manifests and deploys the stack.

```
curl https://release.scopear.com/scopearctl/install -sSfL | sudo bash -s -- \
  --nodocs \
  --installdir=/usr/sbin \
  --version="${VERSION}" \
  --channel="${CHANNEL}"

# Record the installed version/channel (matches the packer build).
echo "${VERSION}" | sudo tee /etc/scopear-version
echo "${CHANNEL}" | sudo tee /etc/scopear-channel

# Confirm it runs.
scopearctl version || scopearctl --help
```

---

4. Create the directory tree

These are all the directories the stack bind-mounts data and config into.

```
sudo mkdir -p /opt/scopear/ui/dist
sudo mkdir -p /opt/scopear/add-ons
sudo mkdir -p /opt/scopear/secrets
sudo mkdir -p /opt/scopear/systemd
sudo mkdir -p /opt/scopear/data/mosquitto/config
sudo mkdir -p /opt/scopear/data/mosquitto/certs
sudo mkdir -p /opt/scopear/data/iot/certs
sudo mkdir -p /opt/scopear/data/cms
sudo mkdir -p /opt/scopear/data/mysql
sudo mkdir -p /opt/scopear/data/mongodb
sudo mkdir -p /opt/scopear/data/redis
sudo mkdir -p /opt/scopear/data/minio
sudo mkdir -p /opt/scopear/data/nginx
sudo mkdir -p /opt/scopear/data/postgres
sudo mkdir -p /opt/scopear/data/traefik
sudo mkdir -p /opt/scopear/data/ai
sudo mkdir -p /opt/scopear/traefik-tls
sudo mkdir -p /opt/scopear/__gen__
```

5. Generate the Traefik TLS certificate (self-signed)

Traefik terminates TLS for all `*.swarm.localdomain` hostnames. This generates a long-lived self-signed certificate with the correct Subject Alternative Names. (If you have your own CA-signed wildcard cert for your hostnames, you can skip the generation and place your cert/key at the two paths shown at the end.)

```
sudo tee /opt/scopear/traefik-tls/openssl.cnf >/dev/null <<'EOF'
[req]
default_bits = 2048
prompt = no
default_md = sha256
distinguished_name = dn
x509_extensions = v3_req

[dn]
C = US
ST = California
L = San Francisco
O = Scope AR
OU = DevOps
CN = *.swarm.localdomain

[v3_req]
subjectAltName = @alt_names

[alt_names]
DNS.1 = swarm.localdomain
DNS.2 = *.swarm.localdomain
DNS.3 = cms.swarm.localdomain
DNS.4 = create.swarm.localdomain
DNS.5 = gwl.swarm.localdomain
DNS.6 = worklink.swarm.localdomain
DNS.7 = minio.swarm.localdomain
DNS.8 = minio1.swarm.localdomain
DNS.9 = analytics.swarm.localdomain
DNS.10 = worklink-ai.swarm.localdomain
DNS.11 = iot.swarm.localdomain
DNS.12 = traefik.swarm.localdomain
DNS.13 = dtool.swarm.localdomain
DNS.14 = localhost
EOF

sudo openssl req -x509 -nodes -newkey rsa:2048 \
  -days 8250 \
  -keyout /opt/scopear/traefik-tls/tls.key \
  -out /opt/scopear/traefik-tls/tls.crt \
  -config /opt/scopear/traefik-tls/openssl.cnf \
  -extensions v3_req

sudo chmod 0644 /opt/scopear/traefik-tls/tls.crt
sudo chmod 0600 /opt/scopear/traefik-tls/tls.key
```

```
# Verify the SANs were applied.
openssl x509 -in /opt/scopear/traefik-tls/tls.crt -noout -subject -ext subjectAltName
```

---

6. Generate the swarm secrets

`scopearctl generate swarm-secrets` writes randomly generated passwords/tokens into `/opt/scopear/secrets`, plus the AIStor MinIO license file (`minio.license`) if it is not already present. When `/opt/scopear/traefik-tls/tls.crt` and `tls.key` already exist (from step 5), it also copies them into the nginx SSL secrets and builds the Zeus PKCS#12 encryption bundle — the same steps the AMI packer build runs.

```
# Generate random secret files; wire nginx/zeus from traefik-tls when present.
sudo scopearctl generate swarm-secrets -o /opt/scopear
```

If you generated secrets before creating TLS, re-run the command after step 5 to populate any missing `nginx_ssl_*` and `zeus_encryption_*` files. Existing secret files are never overwritten.

```
sudo scopearctl generate swarm-secrets -o /opt/scopear
```

To wire only missing nginx/zeus secrets from Traefik TLS:

```
sudo scopearctl generate swarm-secrets -o /opt/scopear --wire-tls-only
```

The command prints byte sizes for the wired files; they must be non-zero. If you see `traefik-tls/tls.crt not found`, create TLS under `/opt/scopear/traefik-tls/` first (see step 5), then re-run.

To replace a broken secret, delete that file first, then re-run generate.

Apply ownership and permissions

PEM-style files (`*_key`, `*_crt`, `*_cert`) and `minio.license` are bind-mounted into containers running as non-root, so they must be world-readable (0644). All other secret files stay private (0600).

If you are not using `generate swarm-secrets` for the license, copy `packer/files/minio.license` to `/opt/scopear/secrets/minio.license` before deploy.

```
sudo chown -R "${SWARM_USER}:${SWARM_USER}" /opt/scopear/secrets

sudo find /opt/scopear/secrets -type f \( -name '*_key' -o -name '*_crt' -o -name
'*_cert' -o -name 'minio.license' \) -exec chmod 0644 {} \;
sudo find /opt/scopear/secrets -type f ! -name '*_key' ! -name '*_crt' ! -name '*_cert' !
-name 'minio.license' -exec chmod 0600 {} \;
```

7. Write the swarm configuration

This is the deployment manifest `scopearctl` reads. The default below enables the core stack: **web (CMS), comms (zeus), and WLC2 (Worklink Create)**. AI, IoT, Superset, and coturn are disabled by default.

```
sudo tee /opt/scopear/swarm-config.yaml >/dev/null <<'EOF'
apiVersion: swarm.scopear.com/v1
kind: ScopeAR
metadata:
  creationTimestamp: null
  name: scopear-swarm
  namespace: scopear
spec:
  ai:
    config:
      airbrake: {}
      mongodb: {}
      storage: {}
      true_openai_queries: null
    enabled: false
  channel: stable
  comms:
    coturn:
      enabled: false
      resources: {}
    zeus:
      config:
        redis:
          data_volume: /opt/scopear/data/redis
          resources: {}
      enabled: true
      resources: {}
  hostnames:
    - swarm.localdomain
  iot:
    config:
      cert_volume: /opt/scopear/data/iot/certs
      mqtt:
        cert_volume: /opt/scopear/data/mosquitto/certs
        port: 8883
    enabled: false
    resources: {}
  log_level: info
  on_prem: true
  replicas: 1
  secrets: {}
  secrets_path: /opt/scopear/secrets
  traefik:
    data_volume: /opt/scopear/data/traefik
    tls:
      provider: static
      static_host_cert_path: /opt/scopear/traefik-tls/tls.crt
      static_host_key_path: /opt/scopear/traefik-tls/tls.key
```

```
  superset:
    config:
      cms_db_user: superset
      redis_celery_db: 3
      redis_results_db: 2
    enabled: false
    postgres:
      data_volume: /opt/scopear/data/postgres
      resources: {}
    resources: {}
  ui_password_file: /opt/scopear/.scopear-ui-password
  ui_username: scopear-onprem
  web:
    cms:
      data_volume: /opt/scopear/data/cms
      replicas: 1
      resources: {}
    config:
      admin: {}
      airbrake: {}
      coturn:
        enabled: false
      hubspot: {}
      ldap: {}
      log_level: info
      metabase: {}
      mysql:
        database: scopear
        host: mysql
        port: 3306
        username: scopear_user
      open_id: {}
      redis:
        host: redis
        port: 6379
      sentry: {}
      smtp:
        enabled: false
      storage:
        default_bucket: cms
      twilio: {}
      zeus:
        enabled: false
    minio:
      console_enabled: false
      data_volume: /opt/scopear/data/minio
      replicas: 1
      resources: {}
```

```
    mysql:
      data_volume: /opt/scopear/data/mysql
      enabled: true
      resources: {}
    redis:
      data_volume: /opt/scopear/data/redis
      replicas: 1
      resources: {}
    sidekiq:
      replicas: 1
      resources: {}
  wlc2:
    api:
      resources: {}
    client:
      resources: {}
    config: null
    enabled: true
    model_converter:
      resources: {}
    mongodb:
      data_volume: /opt/scopear/data/mongodb
      replicas: null
      resources: {}
    project_packager:
      resources: {}
    resources: {}
    webapp:
      resources: {}

EOF
```

Enabling optional components

To turn on an optional component, set its `enabled: true` in `/opt/scopear/swarm-config.yaml` and re-run the deploy (section 10):

- **AI / Worklink AI**: `spec.ai.enabled: true`
- **IoT (MQTT)**: `spec.iot.enabled: true` (also requires IoT/mosquitto certs; see the IoT note in section 9, and open port 8883).
- **Superset analytics**: `spec.superset.enabled: true`
- **coturn (TURN relay)**: `spec.comms.coturn.enabled: true` (open ports 3478 and 49152-50000/udp).

If you use hostnames other than `swarm.localdomain`, update `spec.hostnames` and regenerate the TLS cert (section 5) with matching SANs.

8. Get the container images

The ScopeAR images live in private `docker.io/scopear` repositories. Choose one of the two paths below.

8a. Online (registry login)

If the VM has internet access and you have Docker Hub credentials with access to the private repos, log in and pass `--with-registry-auth` on deploy so swarm nodes can pull private images.

```
echo "${DOCKERHUB_TOKEN}" | sudo docker login docker.io -u "${DOCKERHUB_USER}" --password-stdin
```

8b. Air-gapped (load from a saved tarball)

If the VM has **no** registry access, pull and save the images on a connected machine, transfer the tarball, and load it on the VM.

On a machine WITH internet access and registry access, create the image list and save them. The authoritative image list ships with scopearctl, but you can recreate it from the versions Scope AR provides:

```
# image-list.txt: one image[:tag] per line, e.g.
cat > image-list.txt <<'EOF'
docker.io/scopear/scope-cms:3.43.7_0faee7f.1392-onprem
docker.io/scopear/docker-wlc2-api:2.48.9_24ca95f.3773
docker.io/scopear/docker-wlc2-client:2.48.9_24ca95f.3773
docker.io/scopear/docker-wlc2-mc:2.48.9_24ca95f.3773
docker.io/scopear/docker-wlc2-pp:2.48.9_24ca95f.3773
docker.io/scopear/docker-wlc2-webapp:2.48.9_24ca95f.3773
docker.io/scopear/docker-diagnostic-tools:1.3.9_11ce459.240
docker.io/scopear/docker-iot-server:1.1.3_0fbd998.177
docker.io/scopear/iot-simulator:1.0.10_4615540.148
docker.io/scopear/docker-zeus:3.2.29_c849562.130
docker.io/scopear/docker-db:1.5.16_ddb76b9.108
docker.io/scopear/superset:1.4.0_a495f7c.158
docker.io/scopear/getworklink:1.14.1_593a521.127
docker.io/eclipse-mosquitto:2.0.22
docker.io/redis:8.8.0-alpine
quay.io/minio/aistor/minio:RELEASE.2026-06-06T02-44-06Z
docker.io/scopear/docker-mongodb:1.0.2_d9852c1.10
docker.io/mongo:7.0.25-jammy
docker.io/postgres:17.10
docker.io/coturn/coturn:4-alpine3.24
docker.io/qdrant/qdrant:v1.15.1
docker.io/traefik:v3.7.5
EOF

# Pull every image, then save them all into one tarball.
while read -r img; do
  [ -z "$img" ] && continue
  docker pull "$img"
done < image-list.txt

docker save -o scopear-images.tar $(tr '\n' ' ' < image-list.txt)
```

> The exact image versions change per release. Confirm the current tags with Scope AR (they correspond to `manifests/_images.yaml` for your version) before saving the tarball.

Transfer `scopear-images.tar` to the VM (scp, USB, etc.), then **on the VM**:

```
sudo docker load -i scopear-images.tar
sudo docker image ls | grep scopear
```

No `docker login` is required for the air-gapped path.

Loaded images appear as `scopear/scope-cms:tag` (without the `docker.io/` prefix). Ensure `spec.stripPublicRegistry: true` in `swarm-config.yaml` so generated compose files reference the same names. This is the default for `on_prem: true` installs.

```
grep stripPublicRegistry /opt/scopear/swarm-config.yaml
# should show: stripPublicRegistry: true
```

If your config predates this default, add it under `spec:` and redeploy.

---

9. SELinux and host tuning

The containers load kernel modules and mmap files, which SELinux blocks by default. Enable the required booleans and restore the contexts on `/opt/scopear`.

```
sudo setsebool -P domain_kernel_load_modules on
sudo setsebool -P domain_can_mmap_files on

sudo chown -R "${SWARM_USER}:${SWARM_USER}" /opt/scopear
sudo chmod 0750 /opt/scopear/__gen__
sudo restorecon -r /opt/scopear
```

IoT only: if you set `spec.iot.enabled: true`, you must also place the IoT CA and mosquitto server certs before deploying: `iot_ca_cert` / `iot_ca_key` under `/opt/scopear/secrets`, and the mosquitto `server.crt` / `server.key` / `ca.crt` under `/opt/scopear/data/mosquitto/certs` (then `sudo chown 1883:1883 /opt/scopear/data/mosquitto`). Ask Scope AR for the IoT certificate bundle, or generate a CA + server cert with openssl.

---

10. Deploy the stack

This is the command the automated build runs on first boot. It renders the manifests from your config and runs `docker stack deploy` for each enabled component.

Air-gapped (pre-loaded images):

```
sudo scopearctl swarm deploy \
  -f /opt/scopear/swarm-config.yaml \
  --host=unix:///var/run/docker.sock
```

Do **not** pass `--with-registry-auth` when images were loaded from a tarball; that flag tells swarm to pull from the registry and will fail without registry access.

Online (registry pull):

```
sudo scopearctl swarm deploy \
  -f /opt/scopear/swarm-config.yaml \
  --host=unix:///var/run/docker.sock \
  --with-registry-auth
```

The first deploy pulls (or loads) many images and can take a while. Re-run the same command (with or without `--with-registry-auth` as appropriate) any time you change `swarm-config.yaml`.

---

11. (Optional) Run as systemd services

To make the deploy run automatically on boot and (optionally) run the config UI, install the same systemd units the packer build uses.

Deploy-on-boot unit

```
sudo tee /etc/systemd/system/scopear-swarm-deploy.service >/dev/null <<'EOF'
[Unit]
Description=ScopeAR initial Swarm stack deployment
Documentation=file:///opt/scopear/swarm-config.yaml
After=network-online.target docker.service
Wants=network-online.target
Requires=docker.service

[Service]
Type=oneshot
User=root
WorkingDirectory=/opt/scopear
Environment=DOCKER_HOST=unix:///var/run/docker.sock
TimeoutStartSec=0
ExecStart=/usr/sbin/scopearctl swarm deploy -f /opt/scopear/swarm-config.yaml --
host=unix:///var/run/docker.sock
RemainAfterExit=yes
StandardOutput=journal
StandardError=journal
Restart=on-failure
RestartSec=30

[Install]
WantedBy=multi-user.target
EOF

sudo cp /etc/systemd/system/scopear-swarm-deploy.service /opt/scopear/systemd/
sudo systemctl daemon-reload
sudo systemctl enable scopear-swarm-deploy.service
# Start it now (or it will run on next boot):
sudo systemctl start scopear-swarm-deploy.service
sudo journalctl -u scopear-swarm-deploy.service -f
```

Config UI unit (optional)

The browser-based config UI (`scopearctl ui`) serves its assets from `/opt/scopear/ui/dist`. Those static assets are **not** part of the public `scopearctl` release; they are produced by Scope AR's internal build. If you do not have the `ui/dist` assets, skip this unit and manage the stack with the CLI deploy in section 10. If Scope AR provided you a `ui/dist` bundle, place it at `/opt/scopear/ui/dist` first, then:

```
sudo tee /etc/systemd/system/scopear-ui.service >/dev/null <<'EOF'
[Unit]
Description=ScopeAR Swarm Configuration UI
After=network.target docker.service scopear-swarm-deploy.service
Requires=docker.service scopear-swarm-deploy.service

[Service]
Type=simple
User=root
WorkingDirectory=/opt/scopear
Environment="DOCKER_HOST=unix:///var/run/docker.sock"
ExecStart=/usr/sbin/scopearctl ui --port 8080 --config /opt/scopear/swarm-config.yaml
Restart=always
RestartSec=10
StandardOutput=journal
StandardError=journal
NoNewPrivileges=false
PrivateTmp=true
ProtectSystem=strict
ReadWritePaths=/opt/scopear

[Install]
WantedBy=multi-user.target
EOF

sudo cp /etc/systemd/system/scopear-ui.service /opt/scopear/systemd/
sudo systemctl daemon-reload
sudo systemctl enable --now scopear-ui.service
```

The UI then listens on `http://<VM-IP>:8080`.

12. DNS / hosts entries and verification

Point the hostnames at the VM

The stack routes by hostname (`*.swarm.localdomain`). On every client device that needs access, map the hostnames to the VM's IP address. On Linux/macOS edit `/etc/hosts`; on Windows edit `C:\Windows\System32\drivers\etc\hosts`. Replace `<VM-IP>` with the VM's reachable IP:

```
<VM-IP> swarm.localdomain
<VM-IP> cms.swarm.localdomain
<VM-IP> create.swarm.localdomain
<VM-IP> gwl.swarm.localdomain
<VM-IP> worklink.swarm.localdomain
<VM-IP> worklink-ai.swarm.localdomain
<VM-IP> minio.swarm.localdomain
<VM-IP> analytics.swarm.localdomain
<VM-IP> iot.swarm.localdomain
<VM-IP> traefik.swarm.localdomain
<VM-IP> dtool.swarm.localdomain
```

For more than a couple of devices, add these as A records on your internal DNS instead of editing each `hosts` file.

Verify the stack is up

```
# Stacks deployed (expect: web, comms, plus any optional ones).
sudo docker stack ls

# All services and their replica counts (look for N/N, not 0/N).
sudo docker service ls

# Follow a specific service's logs if something is stuck.
sudo docker service logs -f web_cms
```

Then open `https://cms.swarm.localdomain` in a browser and accept the self-signed certificate warning. The CMS is the main entry point.

Find the generated CMS admin password

`scopearctl generate swarm-secrets` created a random CMS password. Read it from the secrets directory:

```
sudo cat /opt/scopear/secrets/cms_password
```

Appendix: what this guide intentionally omits vs. the packer build

The automated `packer/scripts/swarm-setup.sh` includes steps that are specific to building a reusable AWS AMI and are **not** needed for a manual client install. This guide deliberately leaves them out:

- Waiting on `cloud-init` and disabling it for future boots.

- SSH/PAM hardening edits, setting the `ec2-user` password, and `hostnamectl set-hostname swarm.localdomain`.
- Copying the repo's bundled test certs (`example.*`, `rootCA.pem`, IoT/mosquitto certs) — this guide generates a self-signed TLS cert on-box instead.
- Bundling the prebuilt config UI (`ui/dist`) — the CLI deploy in section 10 is the supported manual path; the UI is optional (section 11).
- AMI cleanup (removing temp files, ZeroTier, audit tweaks).

Everything required to get the stack running is included above.

Samples

The following sample manifests are also included as standalone YAML files under `samples/` in the documentation package.

`scopear_swarm_env_sample.yaml`

```
# Generated by hack/generate-docs-samples — do not edit by hand.
# Kitchen-sink example with SetDefaults() applied. Delete or change values as needed.
# Docker Swarm on-premises ScopeAR environment (channel: on-prem).
apiVersion: swarm.scopear.com/v1
kind: ScopeAR
metadata:
  name: scopear-swarm
  namespace: scopear
spec:
  ai:
    ai:
      resources: {}
    config:
      airbrake: {}
      helicone_api_url: https://oai.hconeai.com/v1
      mongodb:
        database: ai
        host: mongodb
        port: 27017
      qdrant_url: http://qdrant:6333
      storage:
        minio_server_address: minio.swarm.localdomain
        minio_server_port: 443
        minio_server_use_ssl: true
        provider: minio
      true_openai_queries: false
    data_volume: /opt/scopear/data/ai
    enabled: false
  channel: on-prem
  comms:
    coturn:
      enabled: false
      resources: {}
      turn_max_port: 50000
      turn_min_port: 49152
      turn_password: Augmenttheworld2025!
      turn_port: 3478
      turn_username: scopear
    zeus:
      config:
        redis:
          data_volume: /opt/scopear/data/redis
          resources: {}
      enabled: true
      resources: {}
  gwl:
    android_package_name: com.scopear.RemoteAR
    deferred_deep_linking_firebase_domain: scopear.page.link
```

```
    ios_app_store_id: "1099950714"
    microsoft_store_id: 9nblggh4v339
    replicas: 1
    resources: {}
  hostnames:
    - scopear.example.com
  iot:
    config:
      cert_volume: /opt/scopear/data/iot/certs
      mqtt:
        cert_volume: /opt/scopear/data/mosquitto/certs
        port: 30883
    enabled: false
    resources: {}
  logLevel: info
  on-prem: true
  secrets: {}
  secrets_path: /opt/scopear/secrets
  stripPublicRegistry: true
  superset:
    config:
      cms_db_user: superset
      mcp_server_enabled: false
      redis_celery_db: 3
      redis_results_db: 2
    enabled: false
    postgres:
      data_volume: /opt/scopear/data/postgres
      resources: {}
    resources: {}
  traefik:
    data_volume: /opt/scopear/data/traefik
    tls:
      acme_storage: /etc/traefik/acme.json
      provider: static
  web:
    cms:
      data_volume: /opt/scopear/data/cms
      replicas: 1
      resources: {}
    config:
      admin: {}
      airbrake: {}
      auth_method: db
      coturn:
        enable_multi_region: false
        enabled: false
        label: us-east-1
```

```
        tunnel_external_host: proxy-aws-us.scopear.com
        tunnel_external_port: 443
        turn_external_host: proxy-aws-us.scopear.com
        turn_external_port: 3478
        turn_protocol: udp
      hubspot: {}
      ldap: {}
      log_level: info
      metabase: {}
      mysql: {}
      open_id: {}
      redis: {}
      sentry: {}
      smtp:
        auth_type: plain
        email_format: all
        enable_starttls_auto: true
        openssl_verify_mode: none
      storage:
        file_storage: s3
      twilio: {}
      zeus:
        enabled: false
        external_host: master.scopear.com
        external_port: 24000
    minio:
      console_enabled: true
      data_volume: /opt/scopear/data/minio
      resources: {}
    mysql:
      data_volume: /opt/scopear/data/mysql
      resources: {}
    nginx:
      resources: {}
    redis:
      data_volume: /opt/scopear/data/redis
      resources: {}
    sidekiq:
      replicas: 1
      resources: {}
  wlc2:
    api:
      replicas: 1
      resources: {}
    client:
      replicas: 1
      resources: {}
    config:
```

```
    cad_pipeline:
      processing_mode: none
      query_time: 10000
    mongodb: {}
    storage:
      bucket_name: wlc2
  model_converter:
    replicas: 1
    resources: {}
  mongodb:
    data_volume: /opt/scopear/data/mongodb
    database: wlc2
    enabled: true
    replicas: null
    resources: {}
  project_packager:
    replicas: 1
    resources: {}
  resources: {}
  webapp:
    replicas: 1
    resources: {}
```