ThreatWerk - Helm Deployment Guide

Deploying ThreatWerk on EKS with Helm

July 2026

1 Overview

This guide walks you through deploying ThreatWerk on an existing EKS cluster using the Helm chart. This is the primary deployment method for AWS Marketplace customers.

What you receive on purchase: A Helm chart package containing the ThreatWerk application (backend + frontend) and container images delivered via ECR.

Time to deploy: Under 30 minutes if your EKS cluster and RDS instance are already provisioned. If you need to set up AWS infrastructure first, see the companion *AWS Setup Guide*.

Prerequisites at a glance:

- EKS cluster (v1.27+) with the AWS Load Balancer Controller installed
- RDS PostgreSQL 16+ instance with a database and application user created
- AWS Secrets Manager secret containing `DATABASE_URL`, `JWT_SECRET`, `ENCRYPTION_KEY`
- IAM role configured for IRSA with Secrets Manager + License Manager permissions
- ACM certificate for your domain
- DNS record you can point to the ALB

If any of these are missing, follow the *AWS Setup Guide* first, then return here.

2 Step 1: Create the Namespace

```
kubectl create namespace threatwerk
```

3 Step 2: Create the Service Account

Create a Kubernetes service account annotated with your IAM role ARN. This enables IRSA - the backend pod assumes this role to access Secrets Manager and License Manager.

```
kubectl apply -f - <<EOF
apiVersion: v1
kind: ServiceAccount
metadata:
  name: threatwerk
  namespace: threatwerk
  annotations:
    eks.amazonaws.com/role-arn: arn:aws:iam::ACCOUNT:role/ThreatWerkPodRole
EOF
```

Replace `ACCOUNT` with your AWS account ID.

Alternatively, let the Helm chart create the service account by setting `serviceAccount.create: true` in your values file (see next step).

4 Step 3: Configure Helm Values

Create a `values-production.yaml` file. This is your deployment-specific configuration that overrides the chart defaults. Image repositories and tags are pre-configured in the Marketplace chart - you do not need to set them.

```
backend:
  replicaCount: 2
  resources:
    requests: { cpu: 100m, memory: 128Mi }
    limits:   { memory: 512Mi }
  env:
    APP_URL: "https://threatwerk.yourcompany.com"
    AWS_REGION: "eu-central-1"
    SECRETS_MANAGER_ARN: >-
      arn:aws:secretsmanager:REGION:ACCOUNT:secret:threatwerk-production-secrets-XXXXXX

    # Email notifications (optional - remove if not using)
    # NOTIFY_BACKEND: "ses"
    # NOTIFY_FROM: "noreply@yourcompany.com"
    # SES_REGION: "eu-west-1"

frontend:
  replicaCount: 1
  resources:
    requests: { cpu: 50m, memory: 32Mi }
    limits:   { memory: 64Mi }

ingress:
  enabled: true
  className: alb
```

```
  host: threatwerk.yourcompany.com
  annotations:
    alb.ingress.kubernetes.io/certificate-arn: >-
      arn:aws:acm:REGION:ACCOUNT:certificate/CERT_ID
    alb.ingress.kubernetes.io/scheme: internal
    alb.ingress.kubernetes.io/ssl-policy: >-
      ELBSecurityPolicy-TLS13-1-2-2021-06
    alb.ingress.kubernetes.io/target-type: instance
    alb.ingress.kubernetes.io/listen-ports: '[{"HTTPS":443}]'
    alb.ingress.kubernetes.io/healthcheck-path: /health
    alb.ingress.kubernetes.io/load-balancer-attributes: >-
      idle_timeout.timeout_seconds=180
    alb.ingress.kubernetes.io/security-groups: >-
      sg-XXXXXXXXXXXXXXXX

serviceAccount:
  create: false
  name: "threatwerk"
```

Replace all placeholder values (`ACCOUNT`, `REGION`, `CERT_ID`, security group ID, domain, secret ARN).

4.1 Values Reference

4.1.1 Required Values

Value	Description
`backend.env.APP_URL`	Public URL of your ThreatWerk instance
`backend.env.AWS_REGION`	AWS region for Secrets Manager and License Manager
`backend.env.SECRETS_MANAGER_ARN`	Full ARN of your Secrets Manager secret
`ingress.host`	Your domain name
`ingress.annotations.alb.../certificate-arn`	ACM certificate ARN
`serviceAccount.name`	Name of the IRSA-annotated service account

4.1.2 Optional Values

Value	Default	Description
`backend.replicaCount`	`2`	Backend pod replicas (2+ recommended)
`backend.env.NOTIFY_BACKEND`	unset	`ses` or `smtp` to enable email
`backend.env.NOTIFY_FROM`	unset	Sender email address
`backend.env.SES_REGION`	unset	AWS region for SES
`backend.env.SES_ROLE_ARN`	unset	Cross-account SES role (if applicable)
`backend.env.INTEL_POLL_INTERVAL`	`12h`	How often intel feeds are polled
`backend.env.INTEL_NVD_LOOKBACK_DAYS`	`90`	NVD history on first run
`backend.env.INTEL_RETENTION_DAYS`	`365`	How long intel entries are kept
`backend.env.DB_MAX_OPEN_CONNS`	`20`	Max open database connections per pod
`backend.env.DB_MAX_IDLE_CONNS`	`5`	Max idle database connections per pod
`backend.env.LOG_LEVEL`	`info`	`debug`, `info`, `warn`, `error`
`backend.env.RATE_LIMIT_BURST`	`60`	Rate limiter burst size
`backend.env.RATE_LIMIT_RPS`	`10`	Sustained requests per second
`ingress.annotations.alb.../scheme`	`internal`	`internal` or `internet-facing`
`frontend.replicaCount`	`1`	Frontend pod replicas

4.1.3 Secrets Manager Keys

The JSON object in your Secrets Manager secret must contain:

Key	Required	Description
`DATABASE_URL`	Yes	PostgreSQL connection string with `sslmode=require`
`JWT_SECRET`	Yes	HMAC signing key for session tokens (min 32 chars)
`ENCRYPTION_KEY`	Yes	AES-GCM key for credential encryption (min 32 chars)

Intel feed API keys (NVD, GHSA, OTX) are configured through the admin UI after installation, not in Secrets Manager.

4.2 Ingress Configuration

The chart creates an ALB via the AWS Load Balancer Controller. Key annotations:

Annotation	Purpose	Recommended
`certificate-arn`	TLS certificate from ACM	Required
`scheme`	`internal` (VPN only) or `internet-facing`	`internal` for corporate
`ssl-policy`	TLS version policy	`ELBSecurityPolicy-TLS13-1-2-2021-06`
`target-type`	How targets are registered	`instance`
`listen-ports`	ALB listener ports	`[{"HTTPS":443}]`
`healthcheck-path`	Health check endpoint	`/health`
`security-groups`	Security group IDs for the ALB	Your VPC security group(s)
`idle_timeout.timeout_seconds`	Connection idle timeout	`180` (WebSocket needs this)

The idle timeout must be at least 180 seconds for real-time collaboration (WebSocket connections).

5 Step 4: Install with Helm

```
helm install threatwerk ./helm/threatwerk \
  --namespace threatwerk \
  --values values-production.yaml
```

If you received the chart as a `.tgz` package:

```
helm install threatwerk threatwerk-26.06.tgz \
  --namespace threatwerk \
  --values values-production.yaml
```

6 Step 5: Configure DNS

Point your domain to the ALB that the ingress resource creates:

```
ALB_DNS=$(kubectl get ingress -n threatwerk threatwerk \
  -o jsonpath='{.status.loadBalancer.ingress[0].hostname}')

echo "Create a CNAME record: threatwerk.yourcompany.com -> ${ALB_DNS}"
```

If using Route 53, create an alias record. If using an external DNS provider, create a CNAME.

Note: the ALB takes 2-3 minutes to provision after Helm install. If the ingress shows no address, wait and retry.

7 Step 6: Verify

```
# Check pods are running
kubectl get pods -n threatwerk

# Check backend health
kubectl exec -n threatwerk deploy/threatwerk-backend -- \
  wget -qO- http://localhost:8080/health

# Check logs for successful startup
kubectl logs -n threatwerk deploy/threatwerk-backend --tail=50
```

Expected log output on successful startup:

```
secrets: resolved 3 keys from Secrets Manager (0 skipped — already set)
database connection established
migrations applied
license mode: AWS Marketplace (product SKU: ...)
ThreatWerk listening on :8080
```

If pods are in `CrashLoopBackOff`, check the Troubleshooting section below.

8 Step 7: Create the First Admin Account

Navigate to `https://threatwerk.yourcompany.com` in your browser. The first user to register automatically becomes the administrator.

Adding users after the first admin:

Users register by submitting their email on the login page, creating an access request. The admin approves it in the Admin panel.

- **With email configured:** An automated setup email is sent to the user.
- **Without email:** The admin gets a setup link on their clipboard to share manually (Slack, in person, etc.).

Email is optional - onboarding works either way.

9 Upgrading

9.1 Rolling Updates

Update the image tag in your values file, then:

```
helm upgrade threatwerk ./helm/threatwerk \
  --namespace threatwerk \
  --values values-production.yaml
```

The deployment strategy (`maxUnavailable: 0, maxSurge: 1`) ensures zero downtime.

9.2 Database Migrations

Migrations run automatically on container startup. When scaling horizontally, PostgreSQL advisory locks prevent concurrent execution - only one pod applies migrations, others wait.

9.3 Rollback

```
helm rollback threatwerk -n threatwerk
```

Database migrations are forward-only. If a rollback requires reverting schema changes, restore from an RDS snapshot taken before the upgrade.

9.4 Upgrade Procedure

1. Review the release notes
2. Create an RDS snapshot: `aws rds create-db-snapshot --db-instance-identifier threatwerk-db --db-snapshot-identifier pre-upgrade-$(date +%Y%m%d)`
3. Update the image tag in your values file
4. `helm upgrade threatwerk ./helm/threatwerk -n threatwerk -f values-production.yaml`
5. Monitor: `kubectl get pods -n threatwerk -w`
6. Verify: `curl -s https://threatwerk.yourcompany.com/health`

10 Scaling

10.1 Horizontal Scaling

```
kubectl scale deploy/threatwerk-backend -n threatwerk --replicas=4
```

Or update `backend.replicaCount` in your values file and run `helm upgrade`.

Cross-instance coordination uses PostgreSQL LISTEN/NOTIFY - no message broker needed. WebSocket sessions, collaboration, and notifications all work across replicas.

Database connection math: Each pod opens up to `DB_MAX_OPEN_CONNS` (default: 20). With N replicas, ensure RDS `max_connections` is at least `N x 20 + 10`.

10.2 Vertical Scaling

For larger teams (100+ concurrent users):

- Backend memory: increase to 1 Gi
- RDS: upgrade to `db.r6g.large` or larger
- Increase `DB_MAX_OPEN_CONNS` to 40

11 Operations

11.1 Health Endpoint

`GET /health` returns HTTP 200 when healthy. Used for ALB health checks and readiness probes.

11.2 Monitoring

Key log patterns to watch for:

Pattern	Meaning
`LICENSE EXPIRED`	License lapsed - all API calls blocked
`LICENSE WARNING`	License expiring within 30 days
`panic recovered`	Application crash (auto-recovered)
`slow request`	Request exceeded 2 seconds
`rate_limited`	Client hitting rate limits

11.3 Backup and Recovery

RDS handles automated daily backups. For point-in-time recovery:

1. Restore RDS from snapshot
2. Update `DATABASE_URL` in Secrets Manager
3. `kubectl rollout restart deploy/threatwerk-backend -n threatwerk`

11.4 Secret Rotation

Secret	Impact of rotation	Procedure
`JWT_SECRET`	Invalidates all sessions (users must re-login)	Update in Secrets Manager, restart pods
`ENCRYPTION_KEY`	Existing SBOM credentials become unreadable	Update, restart, re-enter SBOM source credentials
Database password	Connection failure until restart	Update in RDS, update `DATABASE_URL` in Secrets Manager, restart pods

Restart command:

```
kubectl rollout restart deploy/threatwerk-backend -n threatwerk
```

12 Troubleshooting

12.1 Backend pods not starting (CrashLoopBackOff)

```
kubectl logs -n threatwerk deploy/threatwerk-backend
```

Log message	Cause	Fix
`failed to resolve secrets`	IRSA not configured or ARN wrong	Check service account annotation and IAM role trust policy
`database not reachable`	RDS security group blocks access	Allow inbound 5432 from EKS node security group
`license validation failed`	License Manager permissions missing	Add License Manager actions to IAM policy
`JWT_SECRET must be at least 32 characters`	Secret value too short	Update value in Secrets Manager
`ENCRYPTION_KEY is required`	Key missing from secret JSON	Add the key to Secrets Manager

12.2 502 Bad Gateway from ALB

- Pods not ready - check target group health in EC2 console
- Health check path wrong - must be `/health`
- Port mismatch - backend listens on 8080

12.3 WebSocket connections failing

- ALB idle timeout too low - must be 180+ seconds
- Not using ALB (NLB does not support WebSocket upgrade)
- Security group blocking outbound from ALB to pods on 8080

12.4 Intel feeds not updating

- No outbound internet - check NAT Gateway and security group egress (443 outbound)
- NVD rate limiting - add `NVD_API_KEY` to Secrets Manager
- Check logs for `intel poll failed` messages

12.5 Ingress not getting an address

- AWS Load Balancer Controller not installed or not running
- Subnet tags missing (`kubernetes.io/role/elb=1` on public subnets)
- ACM certificate ARN invalid or not yet validated

13 Appendix: Ports and Protocols

Source	Destination	Port	Protocol	Purpose
User	ALB	443	HTTPS	Application access
ALB	Backend	8080	HTTP	API + WebSocket
ALB	Frontend	8080	HTTP	Static assets
Backend	RDS	5432	PostgreSQL/TLS	Database
Backend	Secrets Manager	443	HTTPS	Credential fetch
Backend	SES	443	HTTPS	Email delivery
Backend	S3	443	HTTPS	SBOM fetch
Backend	License Manager	443	HTTPS	License validation
Backend	Internet	443	HTTPS	Intel feeds

14 Appendix: Minimum Resource Requirements

Component	CPU	Memory	Notes
Backend (per replica)	100m	512 Mi	2+ replicas recommended
Frontend (per replica)	50m	64 Mi	1 replica sufficient
RDS	2 vCPU	2 GB	`db.t3.medium` or larger