Yar CLI

```
Yar – Fleet Bootstrapper (local ↔ Cluster)

Usage:
  yar <object> <verb> [flags]

Objects & verbs:

fleet
  up [env]            Hoist the fleet (bootstrap Colima/VPN/DNS; start services)
  down [env]          Bring the fleet to port (stop services)
  destroy [env]       Remove the fleet
  restart [env]       Restart fleet, apply config
  update              Update yar and pack catalog
  status              Show fleet status (planned)

config
  get [-o fmt]        Show global config (~/.config/yar/config.yaml)
  edit                Edit global config

project
  init [-o fmt]       Guided setup (creates ./yar.yaml)
  get  [-o fmt]       Show project config
  edit [-o fmt]       Edit project config

pack
  list                List available service packs
  install <name>      Install a pack (e.g. redis)
  remove  <name>      Uninstall a pack  (alias: uninstall)

template
  build [--env <e>]   Produce deploy assets (Helm chart/subcharts/values or Compose)
  render [--env <e>]  Render manifests offline (Kubernetes/Compose)
  publish             Publish charts/subcharts to artifact repo

secret
  set <key> <val> [--env <e>] [--store <s>]   Set a secret
  get <key>           Inspect a secret (redacted)
  delete <key>        Delete a secret (redacted)
  list                List required secrets

hosts
  set <name> <ip>     Manually configure a host
  get <name>          Inspect a host block
  delete <name>       Delete a host block
  list                List yar-managed host blocks

doctor
  run                 Checks for VPN, DNS, hosts, clusters, secrets
  # aliases: `yar sound`, `yar repair`

Other:
  version             Print yar version
  help [object]       Show help for an object

Global flags:
  -h, --help          Show help
  -v, --verbose       Verbose output
  -o, --output <fmt>  Output format (yaml|json|table)

Ergonomic aliases:
  yar hoist [env]     → yar fleet up [env]
  yar dock [env]      → yar fleet down [env]
  yar scuttle [env]   → yar fleet destroy [env] (danger)
  yar swab            → yar doctor run --fix-cache (or your cleanup action)
```

Yar CLI Examples

```
# Edit configuration
yar config edit


# Create project manifest
yar project init

# Configure project manifest
yar project edit


# Deploy and start project
yar fleet up
# or: yar hoist

# Halt project
yar fleet down
# or: yar dock

# Stop and delete services
yar fleet destroy
# or: yar scuttle


# Local: generate prod artifacts without committing
yar template build --env prod

# Package and push from CI (same command works locally if you want)
yar template build --env prod --package --push oci://ghcr.io/acme/charts
# (alt: yar template publish --env prod --to oci://ghcr.io/acme/charts)

# Update only env values (no chart template changes)
yar template build --env staging --values-only

# Produce artifacts and lock versions for reproducibility
yar template build --env prod --lock
```

Yar Configuration

```
# ~/.config/yar/config.yaml
# Machine-wide defaults and provider wiring for yar.

# Which local container runtime yar should manage for "compose" clusters
container: colima       # options: colima | docker | nerdctl

# VPN provider (required MacOS and Windows)
vpn:
  provider: openvpn
  configPath: ~/.config/yar/vpn/client.ovpn

# How yar resolves service FQDNs on your box
hosts:
  mode: etc             # options: etc | kubedns
  suffix: ""            # optional, e.g. ".local"

# Optional: default docker network yar expects for Compose
network:
  name: yar-net
  cidr: 172.16.34.0/23   # network CIDR block

# Secret provider registry (names referenced by project `environments[*].secrets`)
# Providers: pass, keychain, github, azure, gcp, aws, hashicorp
secrets:
  pass:                 # local dev – maps {{ var }} to `pass` (or keychain)
    provider: pass      # local pass provider
    store: default

  github:
    provider: github
    organization: quay
    # yar will assume an ESO ClusterSecretStore named "github" exists,
    # bootstrapped with a K8s Secret holding the GitHub App creds.
    clusterSecretStore: github
    bootstrapSecretName: github-app-creds

  vault:
    provider: azure
    organization: quay
    vaultName: quay-vault
    tenantId: "<guid>"   # optional if resolvable from env
    clientId: "<guid>"   # only if you want yar to pull directly

# Cluster registry (names referenced by project `environments[*].cluster`)
clusters:
  local:
    provider: k8s       # or compose
    context: local      # kube context name (kubectl config get-contexts)
    namespace: default

  dev:
    provider: k8s
    context: aks08-dev-eus
    namespace: default

  qa:
    provider: k8s
    context: aks06-qa-eus
    namespace: default

  prod:
    provider: k8s
    context: aks04-prod-eus
    namespace: default
```

Yar Custom Service Packs

```
# ~/.config/yar/packs/<pack>/
    schema.json
    meta.json
    templates/
      helm/
        charts/
          <pack>/
            Chart.yaml
            README.md
            templates/
              deployment.yaml
              service.yaml
              configmap.yaml
              virtualService.yaml
      docker/
        docker-compose.yaml
```

Yar Project Configuration

```
# ./yar.yaml

project: ai-agents-backend

environments:
  # default build/render target if not specified
  local:
    cluster: local
    secrets: pass    # local secret provider (e.g., pass, keychain)

  dev:
    cluster: dev
    secrets: github  # matches a ClusterSecretStore "github"

  qa:
    cluster: qa
    secrets: github

  prod:
    cluster: prod
    secrets: github

services:
  - name: redis
    namespace: ai-agents-redis
    pack: redis
    params:
      passwordRef: redis_pass

  - name: kafka
    namespace: ai-agents-kafka
    pack: kafka
    params:
      passwordRef: kafka_pass

  - name: app
    pack: app
    requires: [redis, kafka]  # explicit for ordering
    replicas: 3
    ingress:
      host: api.ai-agents-backend
    env:
      # talk to mesh by stable FQDNs (same local + k8s)
      REDIS_HOST: redis.ai-agents-redis
      KAFKA_HOST: kafka.ai-agents-kafka
      LOG_LEVEL: debug

      # legacy/out-of-band secret reference (provider+key form)
      LEGACY_SECRET:
        provider: vault
        key: some_key
```