INSTALL & UPGRADE

```
# One-liner install
$ curl -fsSL https://raw.githubusercontent.com/desingh-rajan/tstack-kit/main/install.sh | sh

# From source
$ git clone https://github.com/desingh-rajan/tstack-kit  &&  cd tstack-kit/packages/cli
$ deno task install

# Upgrade
$ tstack upgrade            # latest
$ tstack upgrade 1.4.0      # specific version
```

Requires: Deno 2.6.4+ | PostgreSQL 16+ | Git

GLOBAL FLAGS

`--help, -h`	Show help
`--version, -v`	Show version
`--force, -f`	Overwrite / skip confirmation
`--dir, -d <path>`	Target directory

UTILITY COMMANDS

`tstack list`	List tracked projects
`tstack list --status all`	Include destroyed
`tstack versions`	Latest 10 CLI versions
`tstack templates`	List starter templates
`tstack upgrade [ver]`	Upgrade CLI

`--status: created | creating | destroyed | destroying | all`

TSTACK CREATE WORKSPACE <NAME>

```
$ tstack create workspace my-shop
$ tstack create workspace my-shop --with-api --with-admin-ui
$ tstack create workspace my-shop --skip-store
$ tstack create workspace my-shop --scope=core
$ tstack create workspace my-shop --github-org my-org --visibility private
```

Component Include / Exclude

`--with-api`	Include API starter
`--with-admin-ui`	Include Admin UI
`--with-store`	Include storefront
`--skip-api`	Exclude API
`--skip-admin-ui`	Exclude Admin UI
`--skip-store`	Exclude storefront

Cannot mix `--with-*` and `--skip-*`. No flags = all 3.

Workspace & GitHub Options

`--namespace <n>`	Custom namespace
`--scope=<s>`	core | listing | commerce
`--no-git`	Skip Git init
`--github-org <org>`	Create repos under org
`--github-token <t>`	PAT (or $GITHUB_TOKEN)
`--visibility <v>`	private | public (default)
`--skip-remote`	Skip GitHub remote

TSTACK CREATE <TYPE> <NAME> — TYPES: API | ADMIN-UI | STORE

```
$ tstack create api my-shop-api
$ tstack create admin-ui my-shop-admin
$ tstack create store my-shop-store
$ tstack create api my-api --scope=core --skip-db-setup
$ tstack create api my-api --dir ~/projects --force --latest
```

Common Flags

`--latest`	Fetch latest dep versions
`--force, -f`	Overwrite existing
`--dir <path>`	Target directory
`--skip-db-setup`	Skip DB creation

Scope Flags (api & workspace)

`--scope=core`	users, articles, site_settings, enquiries
`--scope=listing`	core + brands, categories, products, variants
`--scope=commerce`	listing + addresses, carts, orders, payments ★

★ default. Legacy: `--skip-listing=core`, `--skip-commerce=listing`

TSTACK SCAFFOLD <ENTITYNAME>

Generates a complete MVC entity (up to 8 files). Entity name can be PascalCase, snake_case, or kebab-case.

```
$ tstack scaffold BlogPost
$ tstack scaffold BlogPost --only-api
$ tstack scaffold BlogPost --only-admin-ui
$ tstack scaffold BlogPost --skip-tests --skip-auth
$ tstack scaffold BlogPost --dir ./my-shop-api --force
```

Flags

`--force, -f`	Overwrite existing files
`--dir <path>`	Target directory
`--skip-admin`	No admin routes / tests
`--skip-tests`	No test files
`--skip-auth`	No auth middleware
`--skip-validation`	No Zod schemas
`--skip-admin-ui`	Skip Admin UI scaffold
`--only-api`	API files only
`--only-admin-ui`	Admin UI files only

Generated Files (8 per entity)

`<e>.model.ts`	Drizzle schema
`<e>.dto.ts`	Zod validation
`<e>.service.ts`	Business logic (BaseService)
`<e>.controller.ts`	HTTP handlers
`<e>.route.ts`	Public API routes
`<e>.admin.route.ts`	Admin panel routes
`<e>.test.ts`	API tests
`<e>.admin.test.ts`	Admin route tests

Naming Convention

Input	Folder	Route	Table
BlogPost	blog_posts/	/blog-posts	blog_posts

TSTACK DESTROY [TYPE] <NAME>

```
$ tstack destroy workspace my-shop
$ tstack destroy workspace my-shop --delete-remote
$ tstack destroy api my-shop-api
$ tstack destroy my-shop-api     # auto-detect
$ tstack destroy my-shop -f      # skip confirmation
```

`--force, -f`	Skip confirmation
`--delete-remote`	Also delete GitHub repos
`--skip-remote`	Skip remote deletion

Lifecycle: KV lookup → Confirm → Delete dir → Drop 3 DBs → Remove from KV

CLI DATA & PATHS

`~/.tstack/`	CLI install directory
`~/.tstack/projects.db`	Deno KV project store
`~/.tonystack/config.json`	Sudo pw, DB defaults
`<proj>_dev`	Development database
`<proj>_test`	Test DB (auto-cleaned)
`<proj>_prod`	Production database

TSTACK INFRA — GENERATES KAMAL DEPLOY CONFIGS

```
$ tstack infra
$ tstack infra --registry ghcr --domain myapp.com --staging --github-username myuser
$ tstack infra --registry dockerhub --docker-username me --domain myapp.com
$ tstack infra --registry ecr --aws-account-id 123456789 --domain myapp.com
```

Flags

Flag	Default	Description
`--registry`	ghcr	ghcr | dockerhub | ecr
`--domain`	yourdomain.com	Production domain
`--staging-domain`	staging.<d>	Staging domain
`--staging`	false	Enable staging env
`--path-prefix`	ts-be/api	API URL path prefix
`--ssh-user`	root	Deploy SSH user
`--db-accessory`	false	Postgres as accessory
`--github-username`	--	Required for GHCR
`--docker-username`	--	Required for Docker Hub
`--aws-account-id`	--	Required for ECR
`--aws-region`	us-east-1	AWS region for ECR

Generated Files

`config/deploy.yml`	Production Kamal config
`config/deploy.staging.yml`	Staging (if --staging)
`.kamal/secrets`	Env vars template
`.github/workflows/deploy.yml`	CI/CD pipeline
`Dockerfile`	Production Dockerfile

Production URLs

Storefront	`https://yourdomain.com`
API	`https://yourdomain.com/ts-be/api/*`
Admin UI	`https://admin.yourdomain.com`

CI/CD: push `main` = production | push `staging` branch = staging

DENO TASKS: API STARTER

Task	Description	Task	Description
deno task dev	Start API with --watch (port 8000)	deno task db:generate	Generate migration from schema
deno task start	Production start	deno task db:seed	Seed superadmin + users + settings
deno task routes	Print all registered routes	deno task seed:ecommerce	Seed sample ecommerce data
deno task setup	migrate + seed (first run)	deno task db:studio	Open Drizzle Studio (:4983)
deno task test	Full test suite	deno task check	fmt --check + lint
deno task test:watch	Tests in watch mode	deno task fmt	Auto-format source
deno task db:create	Create databases	deno task lint	Run linter
deno task db:migrate	Run Drizzle migrations	Aliases: migrate:run = db:migrate | migrate:generate = db:generate	

DENO TASKS: CLI

deno task dev	Run CLI in dev mode
deno task test	Tests + auto cleanup
deno task test:watch	Watch mode
deno task test:coverage	With coverage report
deno task test:db	With real DB tests
deno task install	Install CLI globally
deno task cleanup:test-dbs	Clean all test DBs

DENO TASKS: ADMIN LIBRARY

deno task test	Run test suite
deno task test:watch	Watch mode
deno task test:coverage	With coverage report
deno task cleanup:test-db	Clean test DB

DENO TASKS: ADMIN UI & STOREFRONT

Task	Description
deno task dev	Vite dev server (Admin :5173 / Store :5174)
deno task build	Production build
deno task start	Serve built app
deno task update	Update Fresh framework
deno task check	fmt + lint + type check

Root: `deno task bump` — bump kit version across all packages

QUICK START (5 MINUTES)

```
$ curl -fsSL https://raw.githubusercontent.com/desingh-rajan/tstack-kit/main/install.sh | sh
$ tstack create workspace my-shop

$ cd my-shop/my-shop-api
$ cp .env.example .env       # set DATABASE_URL and JWT_SECRET
$ deno task db:migrate && deno task db:seed

# Run in 3 separate terminals:
$ deno task dev                              # API   localhost:8000
$ cd ../my-shop-admin-ui && deno task dev   # Admin localhost:5173
$ cd ../my-shop-store    && deno task dev   # Store localhost:5174

# Add a new entity:
$ cd my-shop/my-shop-api
$ tstack scaffold BlogPost
$ deno task db:generate && deno task db:migrate
```

WORKSPACE STRUCTURE

```
my-shop/
  my-shop-api/           # Hono API    :8000
    src/
      entities/          # model/dto/svc/ctrl/route
      middleware/        # auth, error, cors, timing
      config/            # db, env, app config
    migrations/          # Drizzle SQL files
    scripts/             # db:create, seed, migrate
    tests/               # integration tests
  my-shop-admin-ui/      # Fresh admin :5173
    routes/ islands/ entities/
    config/entities/ components/ lib/
  my-shop-store/         # Fresh store :5174
    routes/ islands/ components/ lib/
```

Default Ports

API	:8000	Admin UI	:5173
Storefront	:5174	Drizzle Studio	:4983

ENTITY SCOPES

Scope	Entities Included
core	users, articles, site_settings, enquiries
listing	core + brands, categories, products, product_images, product_variants, variant_options
commerce	listing + addresses, carts, orders, payments (default)

TECH STACK

Runtime	Deno 2.6.4+	Validation	Zod
API	Hono	Auth	JWT access + refresh
ORM	Drizzle ORM	Payments	Razorpay
Database	PostgreSQL 16+	Storage	AWS S3
Frontend	Fresh 2 + Preact	Email	Resend | SES | SMTP
Styling	Tailwind CSS	Deploy	Kamal + Docker
CI/CD	GitHub Actions	JSR	@tstack/admin

AUTH ROUTES

POST	/auth/login	Login → JWT pair
POST	/auth/register	Register new user
POST	/auth/refresh	Refresh access token
POST	/auth/forgot-password	Request password reset
POST	/auth/reset-password	Reset with token
GET	/auth/google/callback	Google OAuth callback

ENTITY ROUTES (PER ENTITY)

Public API — /api/<entity>		
GET	/api/<e>	List (paginated)
GET	/api/<e>/:id	Get by ID
POST	/api/<e>	Create record
PUT	/api/<e>/:id	Update record
DELETE	/api/<e>/:id	Delete record
Admin API — /ts-admin/<entity>		
GET	/ts-admin/<e>	List (admin)
GET	/ts-admin/<e>/:id	Get by ID (admin)
POST	/ts-admin/<e>	Create (admin)
POST	/ts-admin/<e>/bulk-delete	Bulk delete
PUT	/ts-admin/<e>/:id	Update (admin)
DELETE	/ts-admin/<e>/:id	Delete (admin)

BASESERVICE & BASECONTROLLER

Lifecycle Hooks

beforeCreate	Runs before INSERT
afterCreate	Runs after INSERT
beforeUpdate	Runs before UPDATE
afterUpdate	Runs after UPDATE
beforeDelete	Runs before DELETE
afterDelete	Runs after DELETE

Auth Options

requireAuth: true	Require valid JWT
ownershipCheck	(record, userId) => boolean
roles	["admin", "superadmin"]

TESTING

BDD-style (describe/it), real PostgreSQL, no mocks. Three separate databases: _dev / _test / _prod.

```
$ cd my-shop-api
$ deno task test              # full test suite
$ deno task test:watch        # watch mode

$ cd packages/cli
$ deno task test              # CLI tests
$ deno task test:coverage     # with coverage

$ cd packages/admin
$ deno task test              # admin library tests
```

Test Patterns

- Uses `app.request()` — no HTTP server needed
- Auth helper: `createTestUser("admin")`
- `ENVIRONMENT=test` auto-set during runs
- storefront-starter & admin-ui-starter: no test suite

Test Env Vars

`TSTACK_CLI_TEST=true`	Destructive CLI tests
`TSTACK_TEST_DB=true`	Real DB test suite
`TSTACK_TEST_GITHUB=true`	GitHub API tests

DEPLOYMENT (KAMAL)

```
$ cd my-shop-api
$ tstack infra                        # 1. generate deploy configs
$ kamal setup                         # 2. first-time server setup
$ kamal deploy                        # 3. deploy to production
$ kamal deploy -d staging             # deploy to staging
$ kamal app logs                      # view logs
$ kamal app logs -f                   # tail logs
$ kamal app exec -i bash              # SSH into container
$ kamal rollback                      # rollback last deploy
$ kamal app exec "deno task db:migrate"  # run migration on server
```

PRE-PUSH CHECKLIST & COMMON WORKFLOWS

Pre-Push (mandatory every time)

```
$ deno fmt --check && deno lint
$ cd packages/cli   && deno task test
$ cd packages/admin && deno task test
$ cd packages/api-starter && deno task test
```

> deno check has pre-existing failures in storefront-starter and admin-ui-starter — use fmt + lint only as gate.

Common Workflows

```
# new project
$ tstack create workspace my-app
$ cd my-app/my-app-api
$ cp .env.example .env
$ deno task setup && deno task dev

# add entity
$ tstack scaffold BlogPost
$ deno task db:generate && deno task db:migrate

# destroy
$ tstack destroy workspace my-app --delete-remote -f

# deploy
$ tstack infra && kamal setup && git push origin main

# upgrade CLI
$ tstack versions && tstack upgrade
```

CORE ENV VARS

Variable	Default	Description
ENVIRONMENT	development	development | test | production
PORT	8000	API server port
DATABASE_URL	--	PostgreSQL URL (required)
ALLOWED_ORIGINS	localhost:3000	CORS origins, comma-separated
LOG_LEVEL	info	debug | info | warn | error

Service URLs

APP_URL	http://localhost:8000	Backend API base URL
STOREFRONT_URL	http://localhost:5174	Storefront base URL
ADMIN_URL	http://localhost:5173	Admin UI base URL

JWT

JWT_SECRET	Signing secret (required)
JWT_ISSUER	tonystack (default)
JWT_EXPIRY	7d — values: 1h | 7d | 30d

Seed Credentials

SUPERADMIN_EMAIL	Admin login email (required)
SUPERADMIN_PASSWORD	Admin password (required)
ALPHA_EMAIL	Test user email
ALPHA_PASSWORD	Test user password

Priority: System env > .env.{ENVIRONMENT}.local > .env

EMAIL ENV VARS

Variable	Default	Description
EMAIL_PROVIDER	auto	resend | ses | smtp | auto
EMAIL_FROM	--	Sender email address
RESEND_API_KEY	--	Resend API key (re_xxx)
SMTP_HOST	--	SMTP server hostname
SMTP_PORT	--	SMTP port (usually 587)
SMTP_USER	--	SMTP username
SMTP_PASS	--	SMTP password
SES_ACCESS_KEY_ID	--	AWS SES access key
SES_SECRET_ACCESS_KEY	--	AWS SES secret key

GOOGLE OAUTH

GOOGLE_CLIENT_ID	OAuth Client ID
GOOGLE_CLIENT_SECRET	OAuth Client Secret
GOOGLE_CALLBACK_URL	{APP_URL}/auth/google/callback

PAYMENTS (RAZORPAY)

RAZORPAY_KEY_ID	rzp_test_* or rzp_live_*
RAZORPAY_KEY_SECRET	API Key Secret
RAZORPAY_WEBHOOK_SECRET	Webhook validation secret

AWS S3 (FILE UPLOADS)

AWS_ACCESS_KEY_ID	AWS access key
AWS_SECRET_ACCESS_KEY	AWS secret key
AWS_REGION	ap-south-1 (default)
S3_BUCKET_NAME	Target S3 bucket
S3_PREFIX	Key prefix for uploads

DEPLOY & DEBUG

KAMAL_REGISTRY_USERNAME	Container registry user
KAMAL_REGISTRY_PASSWORD	Registry password / token
DEPLOY_SSH_KEY	SSH private key (base64)
GITHUB_TOKEN	PAT: repo + delete_repo scopes
DEBUG=true	Full stack traces in CLI