Supabase CLI Local stack and deploys

START HERE

`supabase init`
Create supabase/ folder

`supabase start`
Run local stack

`supabase status`
Show local URLs and keys

`supabase migration new <name>`
Empty migration file

`supabase db reset`
Rebuild local DB

`supabase login`
Authenticate CLI

`supabase link --project-ref <ref>`
Connect to a project

`supabase db push`
Apply migrations remotely

Local stack supabase start → `db diff` → **Migration** supabase/ migrations → `db push` → **Linked project** remote database → `gen types` → **Typed client** your app

CONCEPTS

The local stack runs in Docker containers: start Docker first, and expect the first start to pull images.

Most db and migration commands take --local (default), --linked or --db-url <url> to pick the target.

LOCAL STACK

`supabase init`
Create supabase/ config folder

`supabase start`
Start local Supabase containers
Prints API URL, Studio URL and keys when ready.

`supabase status`
Show URLs and keys of the local stack

`supabase stop`
Stop the local containers
Data is backed up to a volume unless --no-backup is given.

`supabase stop --no-backup`
Stop and wipe local data

`supabase start -x <services>`
Start without some services
Comma-separated, e.g. to skip imgproxy or studio.

`supabase status -o env`
Print status as env vars

LOGIN & LINK

`supabase login`
Log in with an access token

`supabase login --token <token>`
Log in non-interactively

`supabase link --project-ref <ref>`
Link folder to a hosted project

`supabase projects list`
List your projects

`supabase projects api-keys --project-ref <ref>`
Show project API keys

`supabase orgs list`
List organizations

DATABASE

`supabase db diff -f <name>`
Write local schema changes as migration
Compares the local database with your migrations.

`supabase db diff --linked`
Diff against the linked project

`supabase db push`
Apply pending migrations to linked DB

`supabase db push --dry-run`
Show what would be pushed

`supabase db pull`
Fetch remote schema as a migration

`supabase db reset`
Recreate local DB from migrations and seed
Local only by default; --linked resets the remote, so use with care.

`supabase db reset --no-seed`
Reset without running seed.sql

`supabase db dump -f <file>`
Dump schema to a file

`supabase db dump --data-only -f <file>`
Dump data only

`supabase db lint`
Check schema for errors

`supabase db push --include-seed`
Also run seed on push

MIGRATIONS

`supabase migration new <name>`
Create an empty timestamped migration

`supabase migration list`
Compare local and remote migrations

`supabase migration up`
Apply pending migrations locally

`supabase migration repair <version> --status applied`
Fix remote migration history
Use --status reverted to mark one as not applied.

`supabase migration squash`
Merge migrations into one

`supabase migration fetch`
Download remote migration history

MIGRATIONS (cont.)

`supabase migration down --last <n>`
Revert the last n local migrations

EDGE FUNCTIONS

`supabase functions new <name>`
Scaffold a function

`supabase functions serve`
Serve functions locally

`supabase functions deploy <name>`
Deploy one function

`supabase functions deploy`
Deploy all functions

`supabase functions serve --env-file <file>`
Serve with env vars from a file

`supabase functions list`
List deployed functions

`supabase functions delete <name>`
Remove a deployed function

`supabase functions deploy --no-verify-jwt`
Deploy without JWT check
Makes the function publicly callable.

SECRETS

`supabase secrets set <NAME>=<value>`
Set a function secret

`supabase secrets set --env-file <file>`
Set secrets from a file

`supabase secrets list`
List secret names

`supabase secrets unset <NAME>`
Remove a secret

TYPES & TESTS

`supabase gen types --lang typescript --local`
TypeScript types from local DB

`supabase gen types --lang typescript --linked`
TypeScript types from linked project

`supabase gen types --lang typescript --project-id <ref>`
Types by project ref

TYPES & TESTS (cont.)

`supabase gen types -s <schema>`
Limit to a schema

`supabase test db`
Run pgTAP tests

`supabase test new <name>`
Create a pgTAP test

BRANCHES & INSPECT

`supabase branches create <name>`
Create a preview branch

`supabase branches list`
List preview branches

`supabase inspect db <report>`
Run a database diagnostics report
Reports include bloat, locks, outliers and long-running-queries.

`supabase config push`
Push local config to the project

`supabase seed buckets`
Create storage buckets from config

RECIPES

First local setup

```
supabase init
supabase start
supabase status
```

Schema change to production

```
supabase migration new
add_posts
# edit the migration SQL
supabase db reset
supabase db push --dry-run
supabase db push
```

Capture Studio edits as migration

```
supabase db diff -f <name>
supabase db reset
supabase migration list
```

Ship a function

```
supabase functions new hello
supabase functions serve
supabase secrets set
API_KEY=<value>
supabase functions deploy hello
```

HELP

`supabase <command> --help`
Usage for a command

`supabase --debug <command>`
Verbose output for troubleshooting