Ruby on Rails Ruby web framework

START HERE

`rails new <app>`
Create an app

`bin/rails server`
Run it on :3000

`bin/rails generate scaffold <Name> <field:type>`
Generate a CRUD resource

`bin/rails db:migrate`
Apply migrations

`bin/rails console`
Poke at your app

`bin/rails routes`
List all routes

`bin/rails test`
Run the tests

`bin/rails destroy <generator> <args>`
Undo a generate

generate files + migration → *edit* → **migration** db/migrate → *db:migrate* → **schema** db/schema.rb → *routes + test* → **server** localhost:3000

CONCEPTS

Inside an app use bin/rails: it runs the version pinned in the Gemfile, not a global one.

Short aliases: s server, c console, g generate, d destroy, db dbconsole, t test.

Commit db/schema.rb with your migrations; teammates load it with db:schema:load or db:prepare.

CREATE & RUN

`rails new <app>`
Create a new app
Default database is SQLite.

`rails new <app> --database=postgresql`
New app using PostgreSQL

`bin/rails server`
Start the server (Puma)

`bin/rails server -p <port>`
Use another port

`bin/rails server -b <ip>`
Bind to an IP address
Default is localhost; use 0.0.0.0 inside containers.

`bin/rails server -e <env>`
Run in another environment

`rails new --help`
List all --skip-* options

`rails new <app> --skip-active-storage`
Skip Active Storage
Also drops Action Mailbox and Action Text.

`bin/rails about`
Rails, Ruby and DB versions

GENERATORS

`bin/rails generate`
List available generators

`bin/rails generate scaffold <Name> <field:type>`
Model, controller, views, routes

`bin/rails generate model <Name> <field:type>`
Model and migration
Field type defaults to string.

`bin/rails generate controller <Name> <action>`
Controller with views and routes

`bin/rails generate resource <Name> <field:type>`
Model, routes, empty controller
No views; good for APIs.

GENERATORS (cont.)

`bin/rails generate migration <Name>`
Empty migration

`bin/rails generate model <Name> password:digest`
Model with has_secure_password

`bin/rails generate task <name>`
Rake task in lib/tasks

`bin/rails generate <generator> --pretend`
Preview, write nothing

`bin/rails generate <generator> --force`
Overwrite existing files

`bin/rails generate <generator> --skip`
Keep files that exist

`bin/rails generate <generator> --help`
Options of one generator

`bin/rails destroy <generator> <args>`
Remove what generate made
Does not roll back the migration; run db:rollback first.

DATABASE

`bin/rails db:migrate`
Run pending migrations

`bin/rails db:rollback`
Undo the last migration
Add STEP=<n> to undo several.

`bin/rails db:migrate:status`
Show each migration up or down

`bin/rails db:create`
Create the database

`bin/rails db:seed`
Load db/seeds.rb

`bin/rails db:setup`
Create, load schema, seed
Does not drop an existing database first.

`bin/rails db:reset`
Drop, recreate and seed

`bin/rails db:prepare`
Set up, or migrate if it exists

`bin/rails db:drop`
Delete the database

`bin/rails db:migrate:redo`
Roll back one and re-run

`bin/rails db:migrate VERSION=<n>`
Migrate to a version

DATABASE (cont.)

`bin/rails db:migrate:up VERSION=<n>`
Run one migration's up

`bin/rails db:migrate:down VERSION=<n>`
Run one migration's down

`bin/rails db:schema:load`
Load schema without migrations

`bin/rails db:schema:dump`
Write db/schema.rb

`bin/rails db:version`
Show current schema version

`bin/rails db:seed:replant`
Truncate tables, then reseed

`bin/rails db:fixtures:load`
Load test fixtures

`bin/rails dbconsole`
Open psql, mysql or sqlite3

Rebuild the database

Task	Drops	Loads	Seeds
`db:setup`	no	schema	yes
`db:reset`	yes	schema	yes
`db:prepare`	no	if new	if new

CONSOLE & RUNNER

`bin/rails console`
Interactive shell with the app loaded

`bin/rails console --sandbox`
Roll back all changes on exit

`bin/rails console -e <env>`
Console in another environment

`bin/rails runner "<code>"`
Run Ruby in the app context

`bin/rails runner <file.rb>`
Run a script in the app

INSPECT

`bin/rails routes`
List all routes

`bin/rails routes -g <pattern>`
Filter by name, path or action

`bin/rails routes -c <controller>`
Routes of one controller

`bin/rails routes -E`
Expanded output with source line

`bin/rails routes -u`
Show unused routes

`bin/rails stats`
Lines of code and test ratio

`bin/rails notes`
Find TODO, FIXME, OPTIMIZE

INSPECT (cont.)

`bin/rails middleware`
Rack middleware stack

`bin/rails zeitwerk:check`
Check autoloading structure

TEST

`bin/rails test`
Run tests, not system tests

`bin/rails test <file>:<line>`
Run the test at a line

`bin/rails test <file1> <file2>`
Run several files or dirs

`bin/rails test -n /<regex>/`
Tests matching a name

`bin/rails test:system`
Run system tests

HOUSEKEEPING

`bin/rails credentials:edit`
Edit encrypted credentials
Uses $VISUAL or $EDITOR.

`bin/rails credentials:show`
Print decrypted credentials

`bin/rails secret`
Generate a random secret

`bin/rails tmp:clear`
Clear cache, sockets, screenshots

`bin/rails assets:precompile`
Compile assets for production

RECIPES

New app with a CRUD resource

```
rails new blog --
database=postgresql
cd blog
bin/rails db:create
bin/rails g scaffold Post
title:string body:text
bin/rails db:migrate
bin/rails server
```

Add a column

```
bin/rails g migration
AddPublishedToPosts
published:boolean
bin/rails db:migrate
# wrong? undo it
bin/rails db:rollback
```

Try code without side effects

```
bin/rails console --sandbox
# Post.create!(title: "x")
# exit rolls everything back
```

Find and run one test

```
bin/rails routes -g post
bin/rails test test/models/
post_test.rb:12
```