Ruby on Rails

The CLI, generators, migrations and database tasks.

01 Install & new app

```
curl https://mise.run | sh
eval "$(~/.local/bin/mise activate bash)"
mise use -g ruby@3
gem install rails
rails --version
rails new store && cd store
bin/rails server    # localhost:3000
```

Rails 8.1 needs Ruby 3.2 or newer. Inside an app, run bin/rails so the app's locked Rails version is used.

02 rails new options

`--api`	API-only app, lighter stack
`-d postgresql`	Database (default: SQLite)
`--minimal`	Skips mailers, storage, more
`--skip-test`	No Minitest files
`--skip-git`	No git init
`-m template.rb`	Apply an app template

Example: rails new myapp --api -d postgresql. -d is short for --database. rails new --help lists the rest.

03 Server & console

```
bin/rails server       # or: bin/rails s
bin/rails s -p 3001 -b 0.0.0.0
bin/rails s -e production
bin/rails console      # or: bin/rails c
bin/rails c --sandbox  # roll back on exit
bin/rails c -e production
```

-b 0.0.0.0 binds every interface. In the console, reload! picks up code changes without a restart.

04 Model generator

```
bin/rails generate model Post \
  title:string body:text user:references
bin/rails generate model User \
  email:string:uniq name:string:index
bin/rails generate model Product \
  'price:decimal{8,2}' stock:integer
bin/rails destroy model Post
```

Adds a model, migration and tests. references makes a foreign key; :index and :uniq add indexes.

05 Controllers & scaffolds

```
bin/rails generate controller Users index
bin/rails generate controller api/v1/Posts \
  index --skip-template-engine
bin/rails generate scaffold Post \
  title:string body:text published:boolean
bin/rails destroy scaffold Post
```

A controller comes with views, a helper and tests. A scaffold adds the model and full CRUD; --api skips the views.

06 Migration generator

```
bin/rails generate migration \
  AddEmailToUsers email:string
bin/rails generate migration \
  RemoveEmailFromUsers email:string
bin/rails generate migration \
  AddUserRefToArticles user:references
bin/rails generate migration \
  CreateJoinTableUserProject user project
```

These name patterns fill in the migration for you. With any other name, write the change method yourself.

07 Database commands

`db:create`	Create the database
`db:migrate`	Run pending migrations
`db:rollback`	Undo the last migration
`db:seed`	Load db/seeds.rb
`db:setup`	Create, load schema, seed
`db:reset`	Drop, then db:setup

Run each with bin/rails. STEP=3 rolls back three. db:reset loads the schema; it does not replay migrations.

08 Migration file

```
class AddSku < ActiveRecord::Migration[8.1]
  def change
    add_column :products, :sku, :string
    add_index :products, :sku, unique: true
  end
end
```

[8.1] is the Rails version the migration targets.

09 Migration methods

```
create_table :products do |t|
  t.string :name
  t.decimal :price, precision: 8, scale: 2
  t.references :category, foreign_key: true
  t.timestamps
end
rename_column :products, :desc, :description
remove_column :products, :sku, :string
```

t.timestamps adds created_at and updated_at. change_column cannot be rolled back automatically.

Ruby on Rails

Routes, controllers and Active Record models.

10 Resourceful routes

```
resources :posts   # the 7 RESTful routes
resources :posts, only: [:index, :show]
resources :users do
  resources :posts, shallow: true
end
resources :posts do
  member { post :publish }
  collection { get :archived }
end
```

shallow: true nests only index, new and create. List every route with bin/rails routes.

11 Custom routes

```
root "posts#index"
get "dashboard", to: "dashboard#index",
  as: :user_dashboard
get "posts/:id", to: "posts#show",
  constraints: { id: /\d+/ }
get "/old-path", to: redirect("/new-path")
```

as: names the helpers: user_dashboard_path. namespace :admin do puts routes under /admin.

12 Controller actions

```
def index
  @posts = Post.includes(:user)  # no N+1
  respond_to do |format|
    format.html
    format.json { render json: @posts }
  end
end
```

Actions are public methods of PostsController.

13 Create & strong params

```
def create
  @post = Post.new(post_params)
  if @post.save
    redirect_to @post, notice: "Created!"
  else
    render :new,
      status: :unprocessable_content
  end
end
private
def post_params
  params.expect(post: [:title, :body])
end
```

params.expect replaces require(:post).permit(...).

14 Filters

```
before_action :set_post, only: %i[show edit]
before_action :admin_only, if: :admin_area?
skip_before_action :require_login,
  only: %i[index show]
around_action :wrap_in_transaction  # yields
```

Also after_action. Subclasses inherit filters.

15 Associations

```
class Post < ApplicationRecord
  belongs_to :user     # needs posts.user_id
  has_many :comments, dependent: :destroy
  has_many :taggings
  has_many :tags, through: :taggings
end
```

belongs_to is required unless optional: true. Also has_one.

16 Validations

```
validates :email, presence: true,
  uniqueness: { case_sensitive: false }
validates :name, length: { minimum: 2 }
validates :price,
  numericality: { greater_than: 0 }
validates :role,
  inclusion: { in: %w[admin user guest] }
```

Run on save, create and update. Add if: or unless: for conditions, or validate :method for custom rules.

17 Queries & scopes

```
Post.find(1)                  # or raises
Post.find_by(slug: "hello")  # or nil
Post.where("views > ?", 100)
Post.where.not(status: "draft")
Post.order(created_at: :desc).limit(10)
scope :published,
  -> { where(published: true) }
Post.published.count
```

Scopes live in the model and chain like any query. Also sum, average and maximum.

18 Callbacks

```
before_validation :normalize_title
before_save :generate_slug
after_create :send_notification
after_save :notify_admin, if: :published?
def normalize_title
  self.title = title.strip.titleize
end
```

Keep complex logic out of callbacks; use a service object.