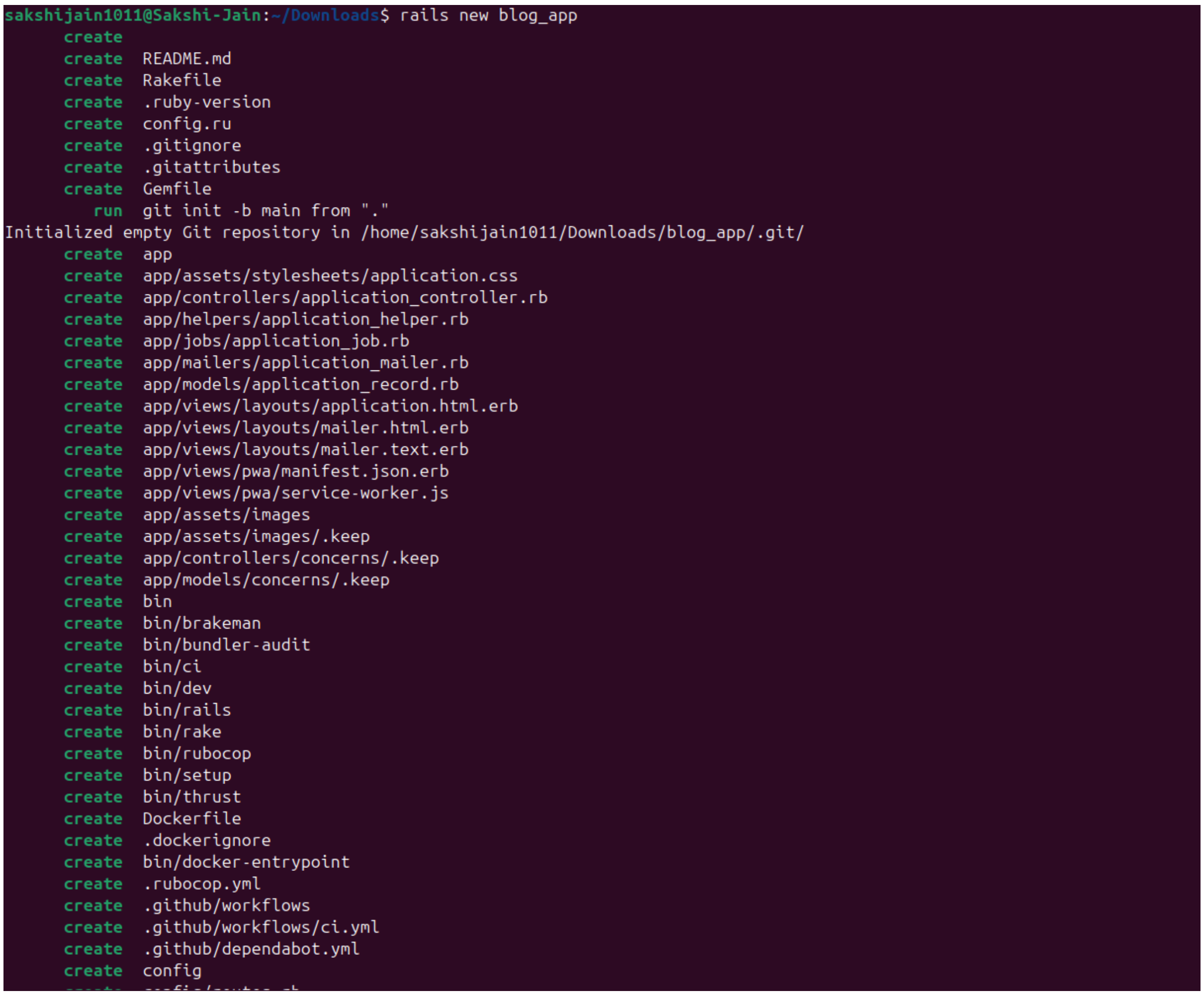
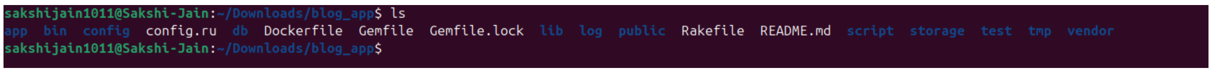
1. Introduction

- Create a new Rails app using rails new blog_app generator command. (5 mins)



```
sakshijain1011@Sakshi-Jain:~/Downloads$ rails new blog_app
      create
      create  README.md
      create  Rakefile
      create  .ruby-version
      create  config.ru
      create  .gitignore
      create  .gitattributes
      create  Gemfile
         run  git init -b main from "."
Initialized empty Git repository in /home/sakshijain1011/Downloads/blog_app/.git/
      create  app
      create  app/assets/stylesheets/application.css
      create  app/controllers/application_controller.rb
      create  app/helpers/application_helper.rb
      create  app/jobs/application_job.rb
      create  app/mailers/application_mailer.rb
      create  app/models/application_record.rb
      create  app/views/layouts/application.html.erb
      create  app/views/layouts/mailer.html.erb
      create  app/views/layouts/mailer.text.erb
      create  app/views/pwa/manifest.json.erb
      create  app/views/pwa/service-worker.js
      create  app/assets/images
      create  app/assets/images/.keep
      create  app/controllers/concerns/.keep
      create  app/models/concerns/.keep
      create  bin
      create  bin/brakeman
      create  bin/bundler-audit
      create  bin/ci
      create  bin/dev
      create  bin/rails
      create  bin/rake
      create  bin/rubocop
      create  bin/setup
      create  bin/thrust
      create  Dockerfile
      create  .dockerignore
      create  bin/docker-entrypoint
      create  .rubocop.yml
      create  .github/workflows
      create  .github/workflows/ci.yml
      create  .github/dependabot.yml
      create  config
```

- Identify what Rails generates by default (5 mins)



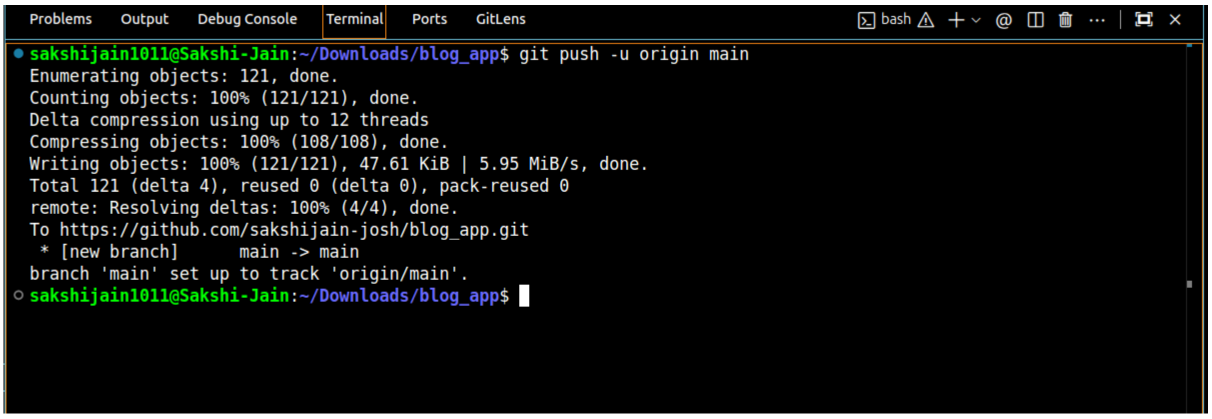
```
sakshijain1011@Sakshi-Jain:~/Downloads/blog_app$ ls
app  bin  config  config.ru  db  Dockerfile  Gemfile  Gemfile.lock  lib  log  public  Rakefile  README.md  script  storage  test  tmp  vendor
sakshijain1011@Sakshi-Jain:~/Downloads/blog_app$
```

Key folders/files and why they exist:

- app/ → main application code
- models/ → database-backed domain objects
- controllers/ → request handling
- views/ → HTML templates
- helpers/ → view helper methods
- jobs/, mailers/, channels/ → background/websocket/email
- config/
  - routes.rb → URL → controller mapping
  - database.yml → DB config
  - environment configs
- db/

    - schema.rb → DB structure snapshot
    - migrate/ → migrations
- Gemfile → dependencies
- bin/rails → Rails CLI entry

    - Push initial setup to a Git repository



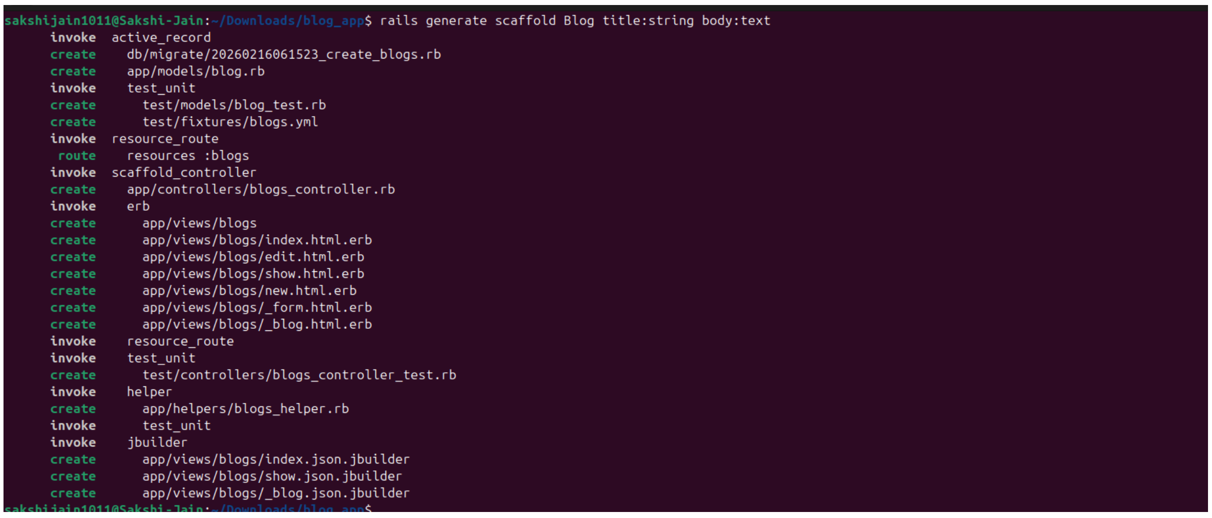
```
sakshijain1011@Sakshi-Jain:~/Downloads/blog_app$ git push -u origin main
Enumerating objects: 121, done.
Counting objects: 100% (121/121), done.
Delta compression using up to 12 threads
Compressing objects: 100% (108/108), done.
Writing objects: 100% (121/121), 47.61 KiB | 5.95 MiB/s, done.
Total 121 (delta 4), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (4/4), done.
To https://github.com/sakshijain-josh/blog_app.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
sakshijain1011@Sakshi-Jain:~/Downloads/blog_app$
```

    - Explore Rails Guides: Getting Started guide (15 mins)

Key ideas the guide is teaching: **Convention over configuration**
Rails assumes defaults so you write less code.

Folder Structure & Request Lifecycle

1. Use the rails scaffold command to create a blog app, with a blog model(title: string, body: text) (10 mins)

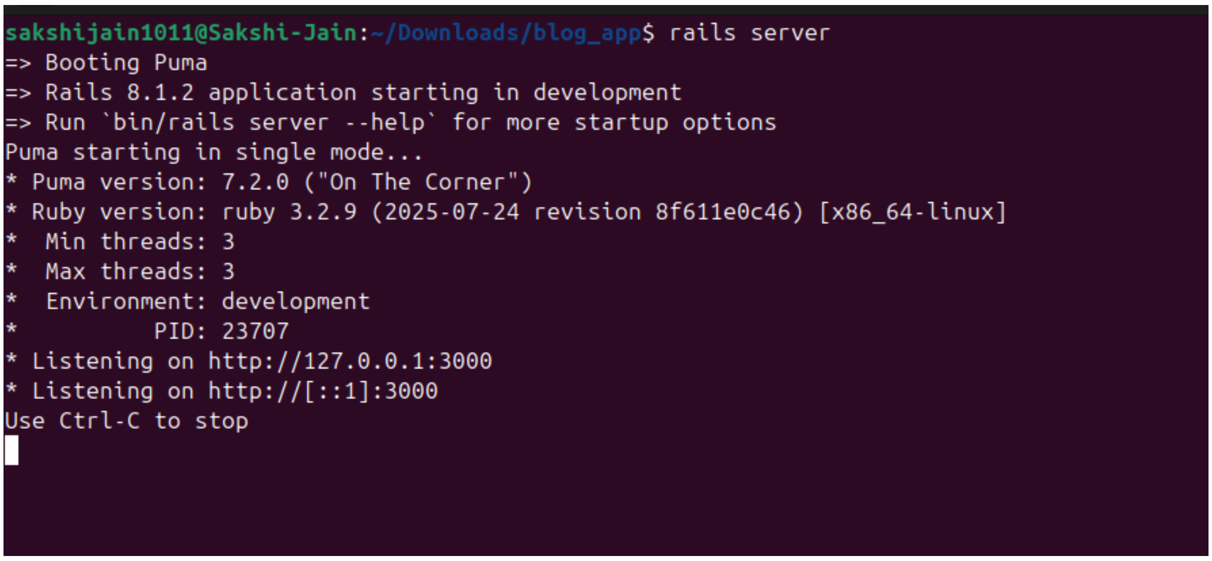


```
sakshijain1011@Sakshi-Jain:~/Downloads/blog_app$ rails generate scaffold Blog title:string body:text
      invoke  active_record
      create    db/migrate/20260216061523_create_blogs.rb
      create    app/models/blog.rb
      invoke    test_unit
      create      test/models/blog_test.rb
      create      test/fixtures/blogs.yml
      invoke  resource_route
       route    resources :blogs
      invoke  scaffold_controller
      create    app/controllers/blogs_controller.rb
      invoke    erb
      create      app/views/blogs
      create      app/views/blogs/index.html.erb
      create      app/views/blogs/edit.html.erb
      create      app/views/blogs/show.html.erb
      create      app/views/blogs/new.html.erb
      create      app/views/blogs/_form.html.erb
      create      app/views/blogs/_blog.html.erb
      invoke    resource_route
      invoke    test_unit
      create      test/controllers/blogs_controller_test.rb
      invoke    helper
      create      app/helpers/blogs_helper.rb
      invoke      test_unit
      invoke    jbuilder
      create      app/views/blogs/index.json.jbuilder
      create      app/views/blogs/show.json.jbuilder
      create      app/views/blogs/_blog.json.jbuilder
sakshijain1011@Sakshi-Jain:~/Downloads/blog_app$
```

2. Explore Rails Structure: identify the purpose of app/controllers, app/models, app/views, and config/routes.rb, db/migrate (10 mins)

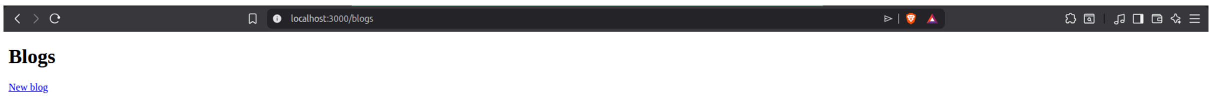
- **app/controllers**
  Handles incoming HTTP requests. Controllers decide what action to run, fetch data from models, and pass that data to views. They act as the coordinator between models and views and should contain request logic, not heavy business logic.
- **app/models**
  Represents application data and business rules. Models interact with the database using ActiveRecord, handle validations, associations, scopes, and callbacks. They are responsible for enforcing data integrity and domain logic.
- **app/views**
  Responsible for presentation. Views render HTML (or JSON) using data prepared by controllers. They should only focus on display and formatting, not business logic.
- **config/routes.rb**
  Defines how URLs map to controller actions. Routes are the entry point of the application and determine which controller and action handle each request.
- **db/migrate**
  Contains migration files that describe database schema changes. Migrations allow version-controlled, repeatable updates to the database structure and act as the source of truth for schema evolution.

3. Start the server and trace the request flow for GET /blogs, check logs. Identify route, controller action, model interaction, response type (10 mins)



```
sakshijain1011@Sakshi-Jain:~/Downloads/blog_app$ rails server
=> Booting Puma
=> Rails 8.1.2 application starting in development
=> Run `bin/rails server --help` for more startup options
Puma starting in single mode...
* Puma version: 7.2.0 ("On The Corner")
* Ruby version: ruby 3.2.9 (2025-07-24 revision 8f611e0c46) [x86_64-linux]
*  Min threads: 3
*  Max threads: 3
*  Environment: development
*          PID: 23707
* Listening on http://127.0.0.1:3000
* Listening on http://[::1]:3000
Use Ctrl-C to stop
```

Server running -

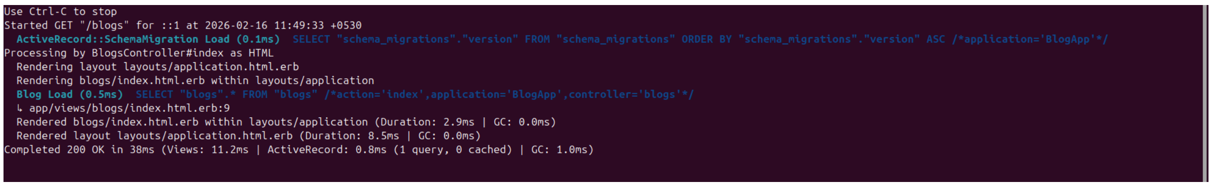


localhost:3000/blogs

Blogs

New blog

Request Trace:



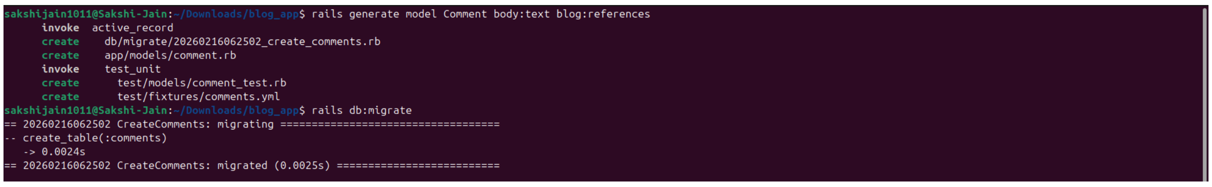
```
Use Ctrl-C to stop
Started GET "/blogs" for ::1 at 2026-02-16 11:49:33 +0530
  ActiveRecord::SchemaMigration Load (0.1ms)  SELECT "schema_migrations"."version" FROM "schema_migrations" ORDER BY "schema_migrations"."version" ASC /*application='BlogApp'*/
Processing by BlogsController#index as HTML
  Rendering layout layouts/application.html.erb
  Rendering blogs/index.html.erb within layouts/application
  Blog Load (0.5ms)  SELECT "blogs".* FROM "blogs" /*action='index',application='BlogApp',controller='blogs'*/
  ↳ app/views/blogs/index.html.erb:9
  Rendered blogs/index.html.erb within layouts/application (Duration: 2.9ms | GC: 0.0ms)
  Rendered layout layouts/application.html.erb (Duration: 8.5ms | GC: 0.0ms)
Completed 200 OK in 38ms (Views: 11.2ms | ActiveRecord: 0.8ms (1 query, 0 cached) | GC: 1.0ms)
```

Identifying route, controller action, model interaction, response type

GET /blogs
→ routes.rb matches BlogsController#index
→ controller calls Blog.all
→ model queries database
→ index.html.erb is rendered
→ HTML response returned

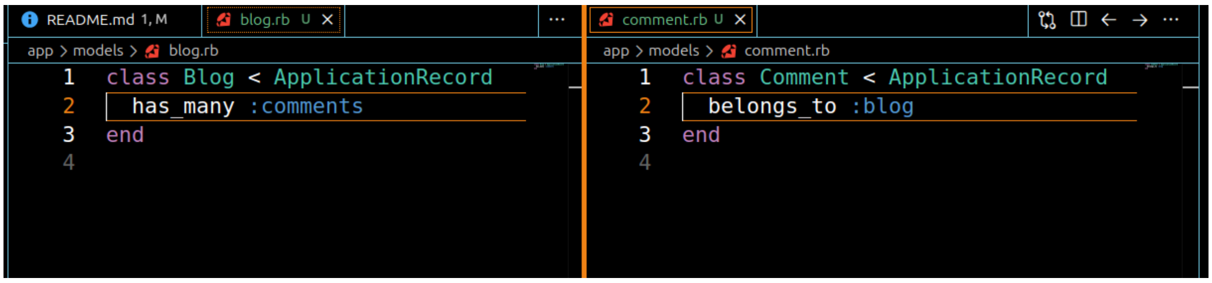
Models

1. Create a Comment model (10 mins)



```
sakshijain1011@Sakshi-Jain:~/Downloads/blog_app$ rails generate model Comment body:text blog:references
      invoke  active_record
      create    db/migrate/20260216062502_create_comments.rb
      create    app/models/comment.rb
      invoke    test_unit
      create      test/models/comment_test.rb
      create      test/fixtures/comments.yml
sakshijain1011@Sakshi-Jain:~/Downloads/blog_app$ rails db:migrate
== 20260216062502 CreateComments: migrating ===================================
-- create_table(:comments)
   -> 0.0024s
== 20260216062502 CreateComments: migrated (0.0025s) ==========================
```

Updated files -



app > models > blog.rb

```ruby
class Blog < ApplicationRecord
  has_many :comments
end
```

app > models > comment.rb

```ruby
class Comment < ApplicationRecord
  belongs_to :blog
end
```

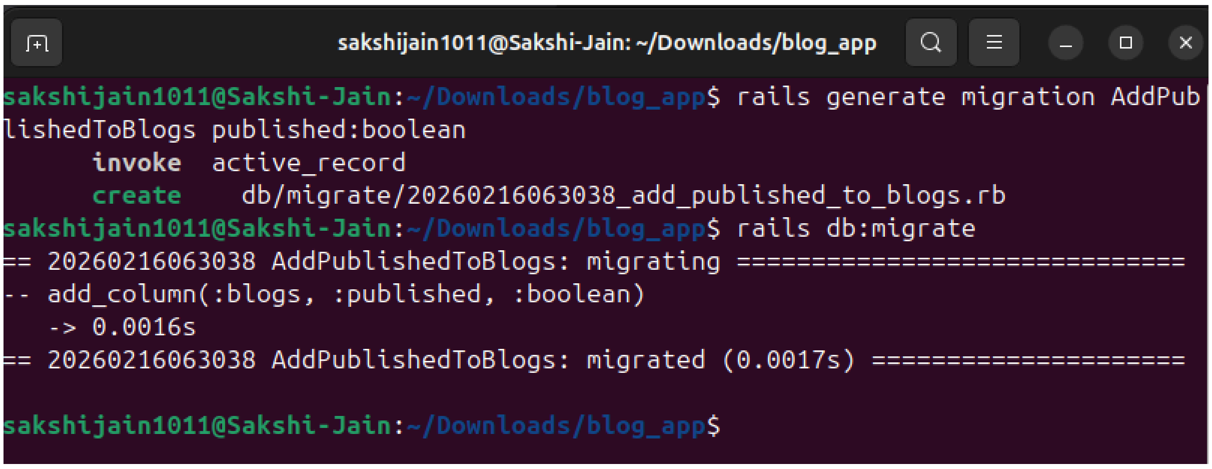
2. Explore what helper methods ActiveRecord adds automatically to both the model and the record

Rails adds:

- Blog.create
- Blog.find
- Blog.where
- blog.comments
- comment.blog
- validations + callbacks API

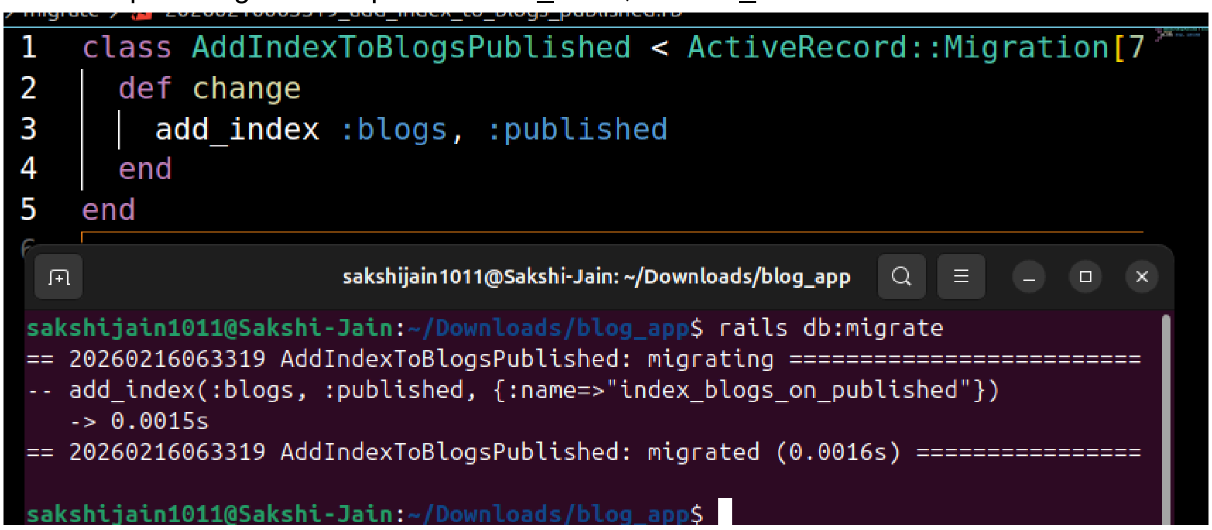
These come from ActiveRecord inheritance.

3. Add a published: boolean column using migration to the blog model (5 mins)



```
sakshijain1011@Sakshi-Jain:~/Downloads/blog_app$ rails generate migration AddPub
lishedToBlogs published:boolean
      invoke  active_record
      create    db/migrate/20260216063038_add_published_to_blogs.rb
sakshijain1011@Sakshi-Jain:~/Downloads/blog_app$ rails db:migrate
== 20260216063038 AddPublishedToBlogs: migrating ===============================
-- add_column(:blogs, :published, :boolean)
   -> 0.0016s
== 20260216063038 AddPublishedToBlogs: migrated (0.0017s) ======================

sakshijain1011@Sakshi-Jain:~/Downloads/blog_app$
```

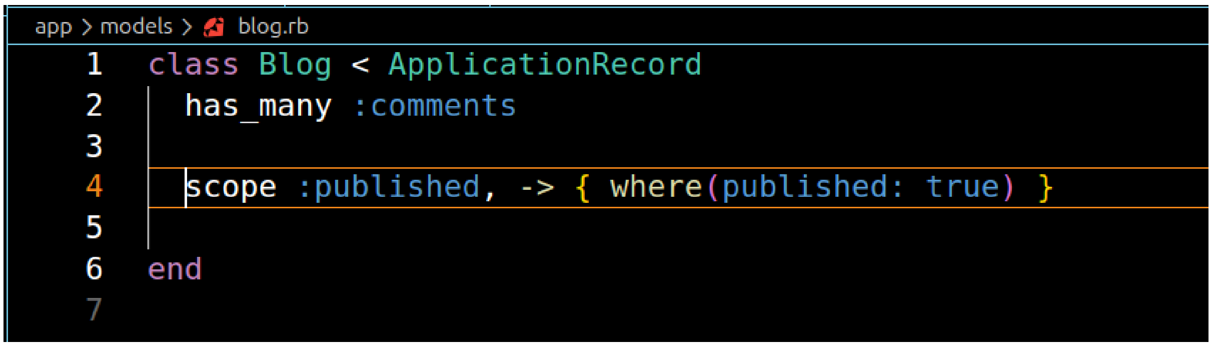
4. Explore migration helpers like add_index, rename_column



```ruby
class AddIndexToBlogsPublished < ActiveRecord::Migration[7
  def change
    add_index :blogs, :published
  end
end
```

```
sakshijain1011@Sakshi-Jain:~/Downloads/blog_app$ rails db:migrate
== 20260216063319 AddIndexToBlogsPublished: migrating =========================
-- add_index(:blogs, :published, {:name=>"index_blogs_on_published"})
   -> 0.0015s
== 20260216063319 AddIndexToBlogsPublished: migrated (0.0016s) ================

sakshijain1011@Sakshi-Jain:~/Downloads/blog_app$
```

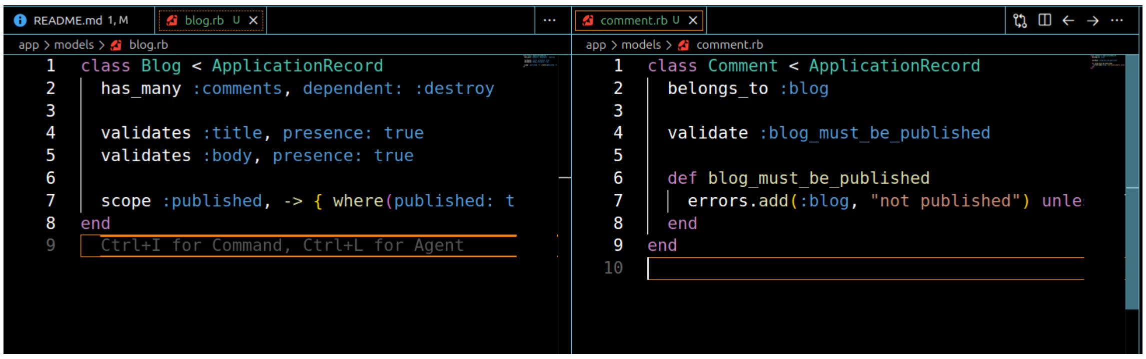
5. Add a published scope to blog model (5 mins)



app > models > blog.rb

```ruby
class Blog < ApplicationRecord
  has_many :comments

  scope :published, -> { where(published: true) }

end

```

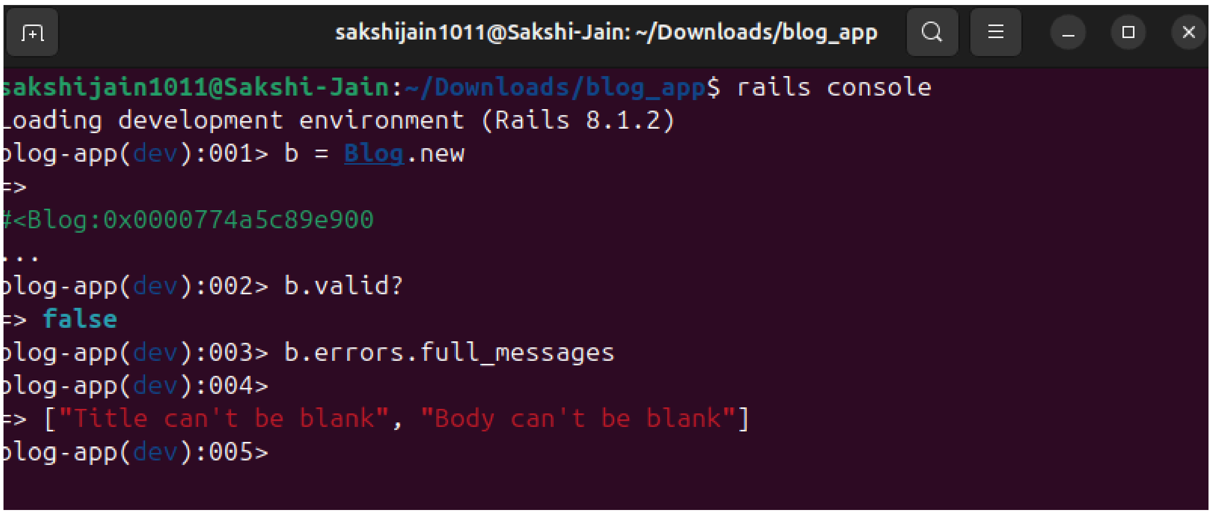
6. Add validations for blog, comment models. Comments can only be added for published blogs. (20 mins)



```ruby
class Blog < ApplicationRecord
  has_many :comments, dependent: :destroy

  validates :title, presence: true
  validates :body, presence: true

  scope :published, -> { where(published: t
end
  Ctrl+I for Command, Ctrl+L for Agent
```

```ruby
class Comment < ApplicationRecord
  belongs_to :blog

  validate :blog_must_be_published

  def blog_must_be_published
    errors.add(:blog, "not published") unle
  end
end
```

7. Verify validations using a valid method on the console. (20 mins)



```
sakshijain1011@Sakshi-Jain:~/Downloads/blog_app$ rails console
Loading development environment (Rails 8.1.2)
blog-app(dev):001> b = Blog.new
=>
#<Blog:0x0000774a5c89e900
...
blog-app(dev):002> b.valid?
=> false
blog-app(dev):003> b.errors.full_messages
blog-app(dev):004>
=> ["Title can't be blank", "Body can't be blank"]
blog-app(dev):005>
```

If validation fails:

- Record is NOT saved
- Errors are stored in errors object

8. Explore other validation helpers
   - **presence** – value must exist and cannot be blank
   - **length** – controls minimum or maximum size of a string
   - **uniqueness** – prevents duplicate values in a column
   - **numericality** – ensures the value is a number and allows numeric limits
   - **format** – checks value against a regex pattern
   - **inclusion** – value must be inside a given list
   - **exclusion** – value must not be inside a given list
   - **confirmation** – requires a matching confirmation field (commonly for passwords)
   - **acceptance** – validates that a checkbox or agreement was accepted
   - **comparison** – compares a value against another value (greater than, less than, etc.)
   - validations run automatically on save/create/valid? and stop the record from saving if rules fail. errors.full_messages shows why validation failed.

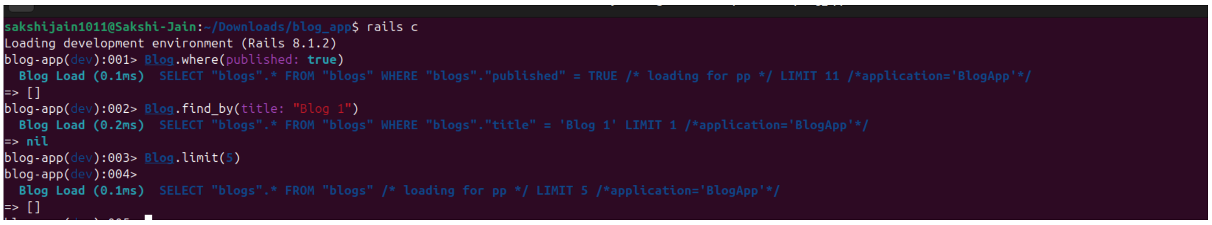
9. Explore other lifecycle callbacks

- **before_validation** – runs before validations execute, used to clean or normalize data

- **after_validation** – runs after validations complete
- **before_save** – runs before a record is saved (create or update)
- **after_save** – runs after a record is saved
- **before_create** – runs only before a new record is created
- **after_create** – runs only after a new record is created
- **before_update** – runs before an existing record is updated
- **after_update** – runs after an existing record is updated
- **before_destroy** – runs before a record is deleted
- **after_destroy** – runs after a record is deleted
- **after_commit** – runs after the database transaction is successfully committed

callbacks allow automatic logic around persistence events but should be used carefully to avoid hidden side effects and hard-to-trace behavior

10. Use AR method on rails console: where, find_by, limit



```
sakshijain1011@Sakshi-Jain:~/Downloads/blog_app$ rails c
Loading development environment (Rails 8.1.2)
blog-app(dev):001> Blog.where(published: true)
  Blog Load (0.1ms)  SELECT "blogs".* FROM "blogs" WHERE "blogs"."published" = TRUE /* loading for pp */ LIMIT 11 /*application='BlogApp'*/
=> []
blog-app(dev):002> Blog.find_by(title: "Blog 1")
  Blog Load (0.2ms)  SELECT "blogs".* FROM "blogs" WHERE "blogs"."title" = 'Blog 1' LIMIT 1 /*application='BlogApp'*/
=> nil
blog-app(dev):003> Blog.limit(5)
blog-app(dev):004>
  Blog Load (0.1ms)  SELECT "blogs".* FROM "blogs" /* loading for pp */ LIMIT 5 /*application='BlogApp'*/
=> []
```

11. Explore order, joins, includes, eager_load, Identify the difference between includes and eager_load
    joins – performs SQL INNER JOIN : Used for filtering across tables.
    Blog.joins(:comments)

- Returns blogs that have comments.

includes – eager loads associations to prevent N+1 queries : Rails may run multiple optimized queries.

Blog.includes(:comments)

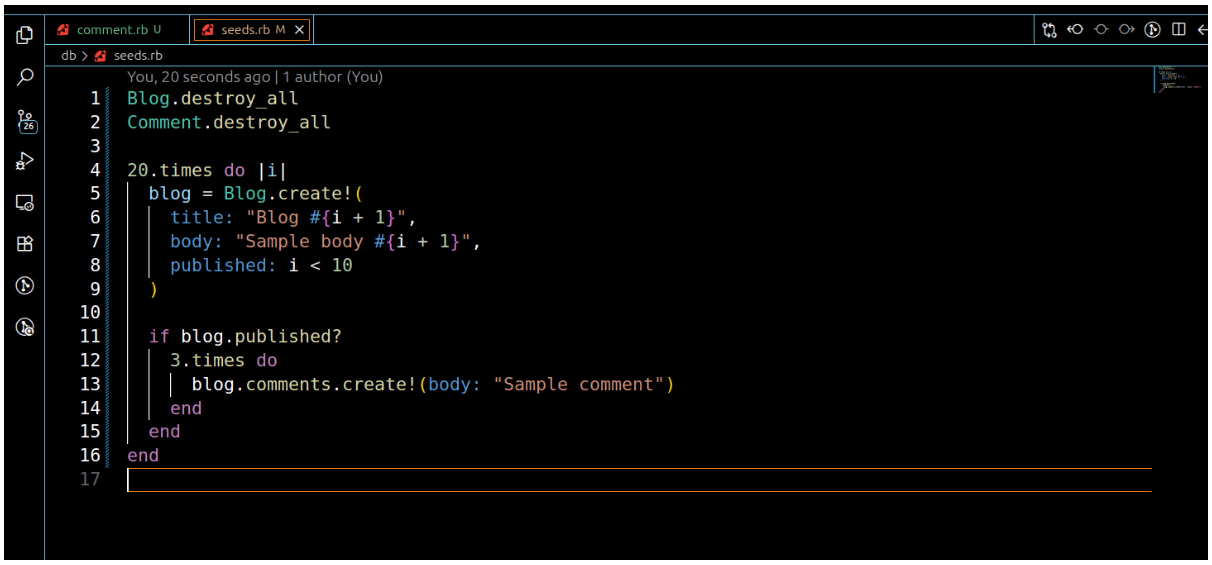
- Best default choice for performance when loading associations.

eager_load – forces a LEFT OUTER JOIN in one query

Blog.eager_load(:comments)

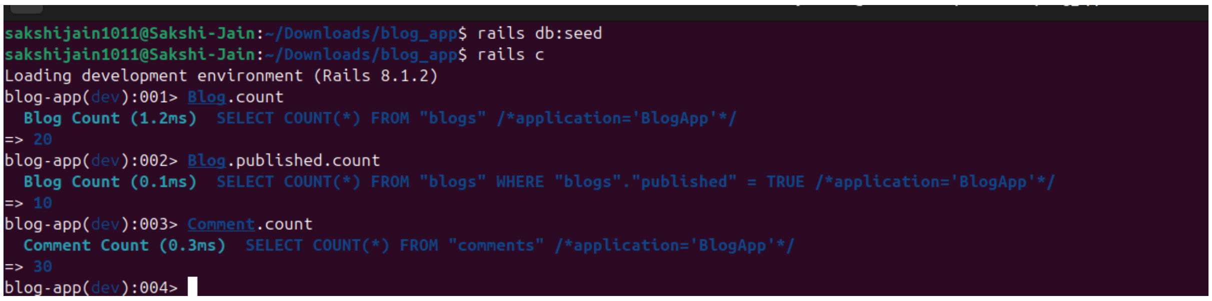
Difference:
includes → flexible, Rails decides best strategy
eager_load → always one big SQL join

12. Explore seeds, and try to create 20 blogs(10 published, 10 unpublished) with comments.



```ruby
Blog.destroy_all
Comment.destroy_all

20.times do |i|
  blog = Blog.create!(
    title: "Blog #{i + 1}",
    body: "Sample body #{i + 1}",
    published: i < 10
  )

  if blog.published?
    3.times do
      blog.comments.create!(body: "Sample comment")
    end
  end
end
```

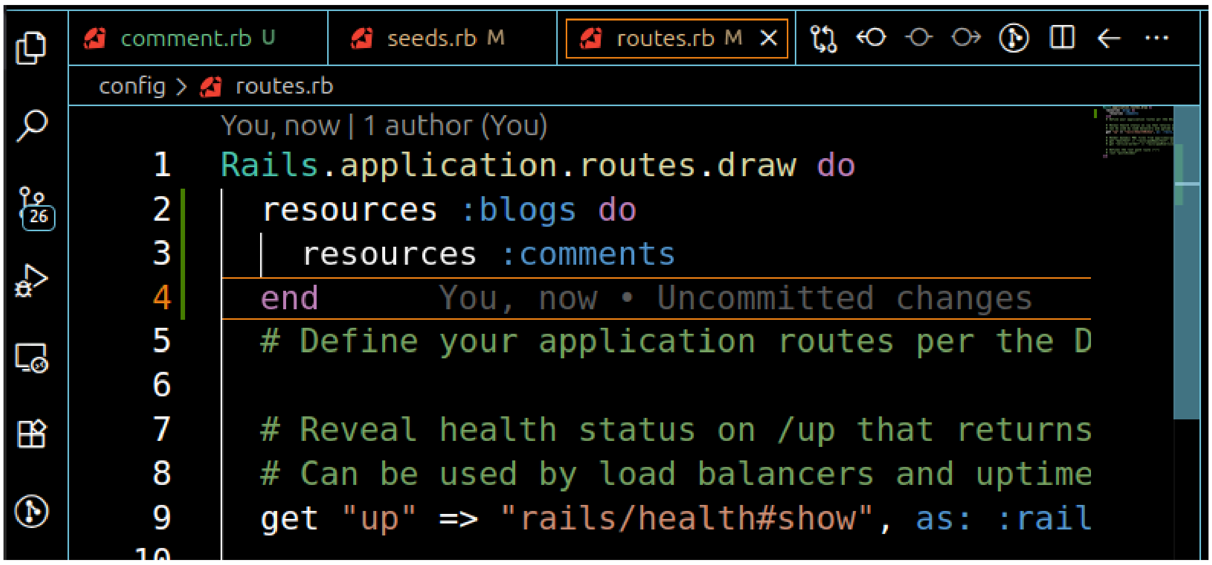
Seeded and verified:



```
sakshijain1011@Sakshi-Jain:~/Downloads/blog_app$ rails db:seed
sakshijain1011@Sakshi-Jain:~/Downloads/blog_app$ rails c
Loading development environment (Rails 8.1.2)
blog-app(dev):001> Blog.count
  Blog Count (1.2ms)  SELECT COUNT(*) FROM "blogs" /*application='BlogApp'*/
=> 20
blog-app(dev):002> Blog.published.count
  Blog Count (0.1ms)  SELECT COUNT(*) FROM "blogs" WHERE "blogs"."published" = TRUE /*application='BlogApp'*/
=> 10
blog-app(dev):003> Comment.count
  Comment Count (0.3ms)  SELECT COUNT(*) FROM "comments" /*application='BlogApp'*/
=> 30
blog-app(dev):004>
```

Controllers

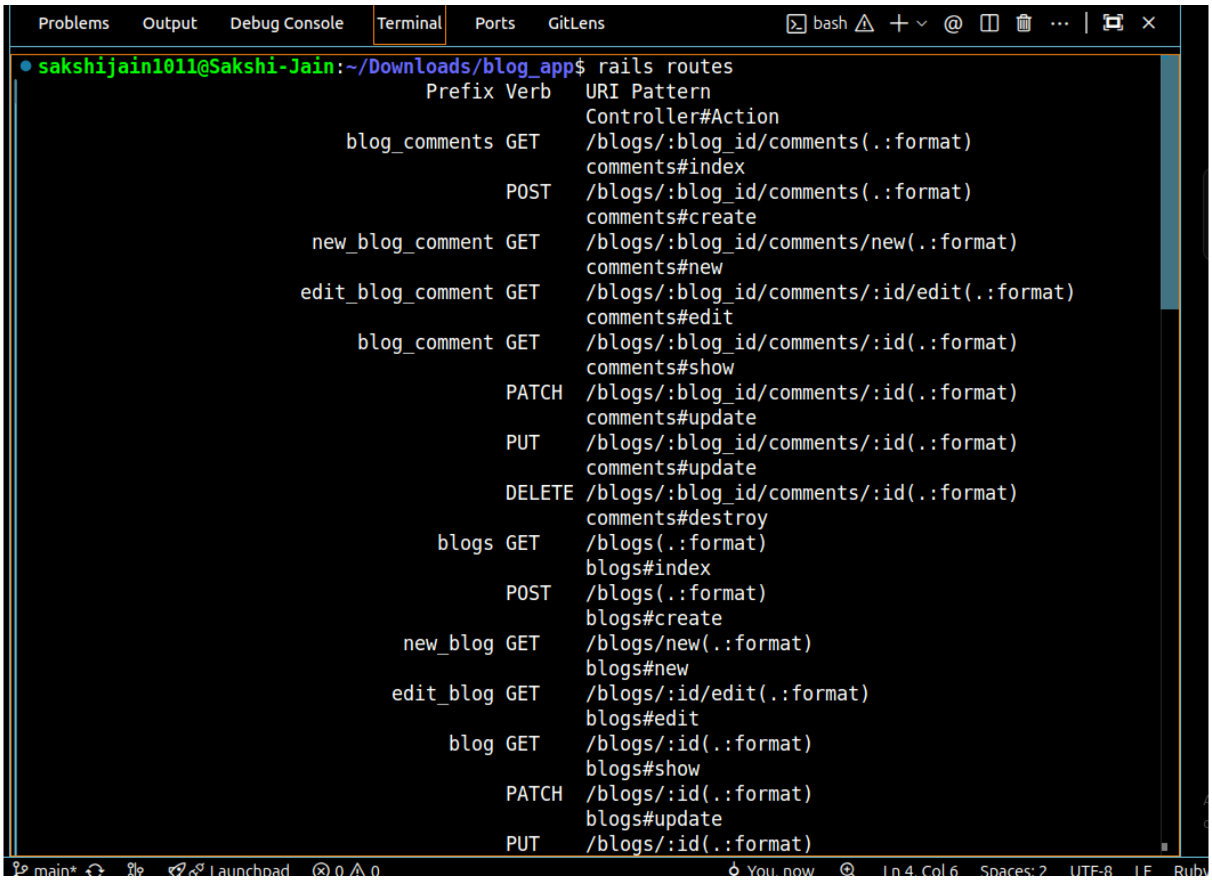
1. Add RESTful routes and actions for comment. (20 mins)



```ruby
Rails.application.routes.draw do
  resources :blogs do
    resources :comments
  end        You, now • Uncommitted changes
  # Define your application routes per the D

  # Reveal health status on /up that returns
  # Can be used by load balancers and uptime
  get "up" => "rails/health#show", as: :rail
```

This generates standard REST routes for comments tied to a specific blog (index, show, create, update, destroy).

2. Verify routes using rails routes

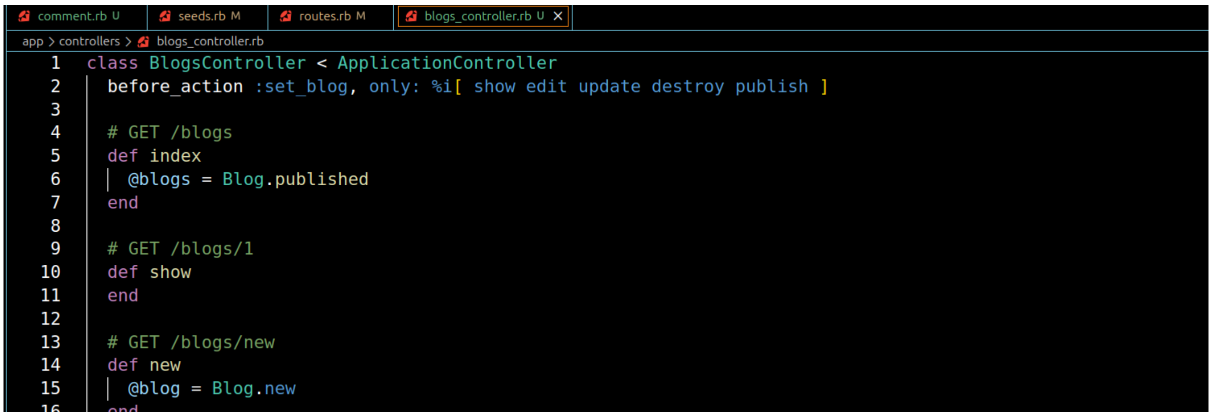


```
● sakshijain1011@Sakshi-Jain:~/Downloads/blog_app$ rails routes
                              Prefix Verb   URI Pattern
                                            Controller#Action
                       blog_comments GET    /blogs/:blog_id/comments(.:format)
                                            comments#index
                                     POST   /blogs/:blog_id/comments(.:format)
                                            comments#create
                    new_blog_comment GET    /blogs/:blog_id/comments/new(.:format)
                                            comments#new
                   edit_blog_comment GET    /blogs/:blog_id/comments/:id/edit(.:format)
                                            comments#edit
                        blog_comment GET    /blogs/:blog_id/comments/:id(.:format)
                                            comments#show
                                     PATCH  /blogs/:blog_id/comments/:id(.:format)
                                            comments#update
                                     PUT    /blogs/:blog_id/comments/:id(.:format)
                                            comments#update
                                     DELETE /blogs/:blog_id/comments/:id(.:format)
                                            comments#destroy
                               blogs GET    /blogs(.:format)
                                            blogs#index
                                     POST   /blogs(.:format)
                                            blogs#create
                            new_blog GET    /blogs/new(.:format)
                                            blogs#new
                           edit_blog GET    /blogs/:id/edit(.:format)
                                            blogs#edit
                                blog GET    /blogs/:id(.:format)
                                            blogs#show
                                     PATCH  /blogs/:id(.:format)
                                            blogs#update
                                     PUT    /blogs/:id(.:format)
```

3. Explore resources vs resource and namespace vs scope
   - **resources** – plural routes for many records, full CRUD (index, show, create, update, delete)
   - **resource** – singular route for one record, no index, used when only one instance exists

   - **namespace** – adds URL prefix + controller module
     example: /admin/blogs → Admin::BlogsController
   - **scope** – adds URL prefix only, controller stays same
     example: /admin/blogs → BlogsController

4. Implement a callback to initialize the resource before show action for all controllers. (10 mins)

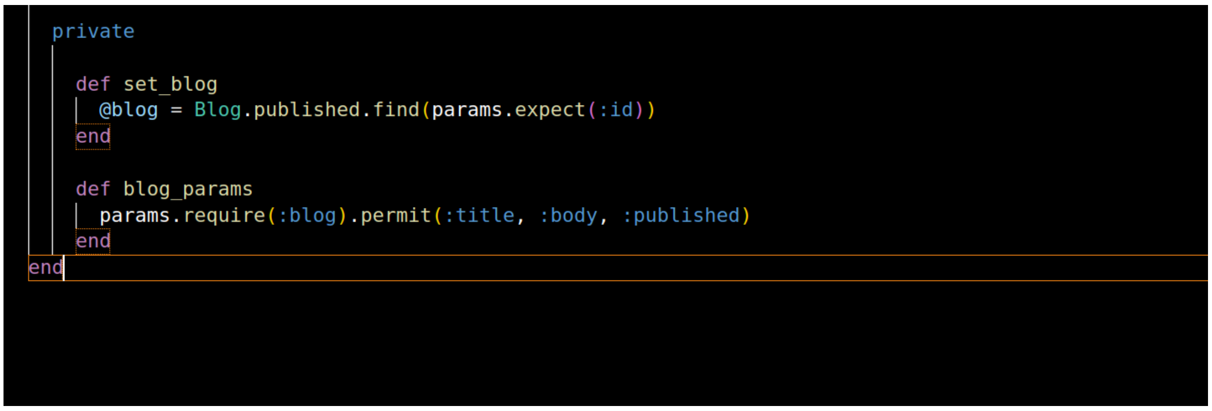


```ruby
class BlogsController < ApplicationController
  before_action :set_blog, only: %i[ show edit update destroy publish ]

  # GET /blogs
  def index
    @blogs = Blog.published
  end

  # GET /blogs/1
  def show
  end

  # GET /blogs/new
  def new
    @blog = Blog.new
```

A callback avoids repeating lookup logic. It loads a record automatically before an action runs.

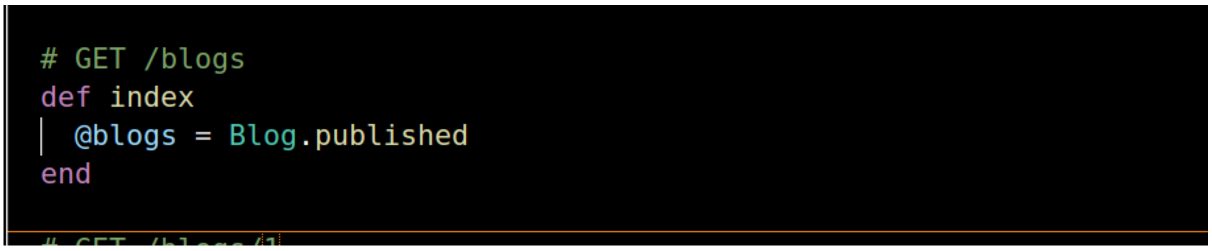
5. Implement strong parameters. (10 mins)

Strong params restrict allowed attributes.



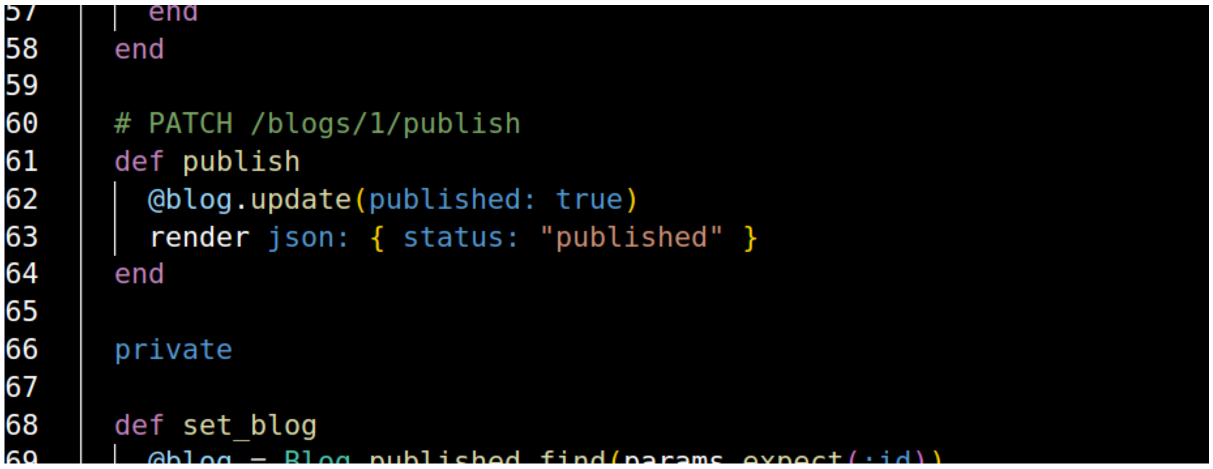
```ruby
  private

    def set_blog
      @blog = Blog.published.find(params.expect(:id))
    end

    def blog_params
      params.require(:blog).permit(:title, :body, :published)
    end
end
```

6. Add logic to only show published blogs.

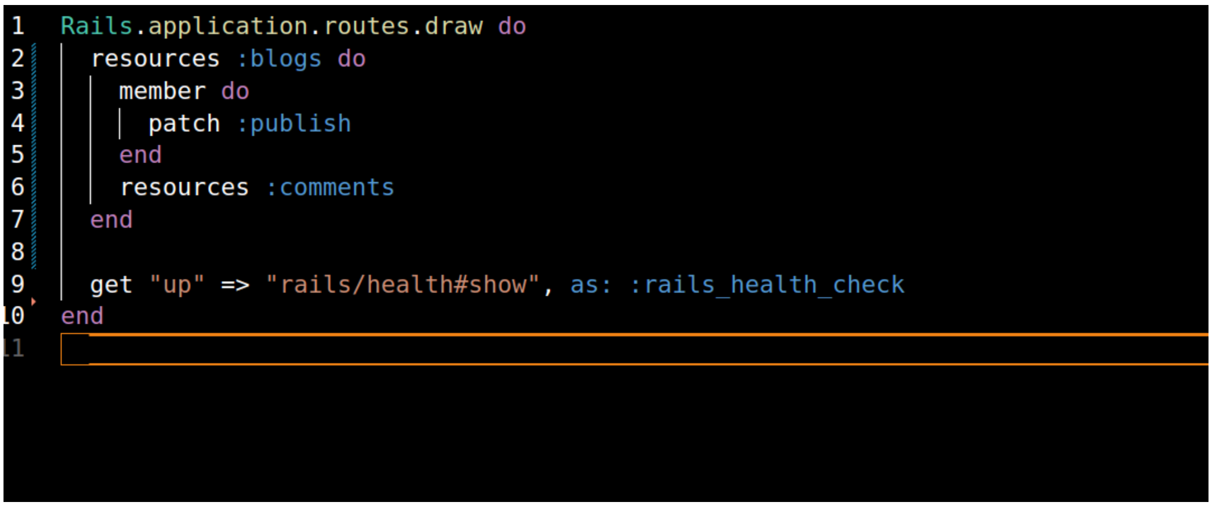


```ruby
# GET /blogs
def index
  @blogs = Blog.published
end
```

7. Add an API to publish a blog.



```ruby
    end
  end

  # PATCH /blogs/1/publish
  def publish
    @blog.update(published: true)
    render json: { status: "published" }
  end

  private

  def set_blog
    @blog = Blog.published.find(params.expect(:id))
```



```ruby
Rails.application.routes.draw do
  resources :blogs do
    member do
      patch :publish
    end
    resources :comments
  end

  get "up" => "rails/health#show", as: :rails_health_check
end

```

API ENDPOINT: PATCH /blogs/:id/publish