Things you should definitely do in a greenfield application

A lot of these are Rails specific, but can be adapted to other stacks.

- Implement multi-tenancy correctly from day one: https://www.flightcontrol.dev/blog/ultimate-guide-to-multi-tenant-saas-data-modeling
  - Steal some things from https://github.com/discourse/discourse
  - Use https://victoriametrics.com/ instead of Prometheus/Grafana stack
  - Puma (taken from Nate's Four Line Friday from 2025-06-20)
    - Fewer containers with more processes per container for web. Try this simulation.
    - Friends don't let friends autoscale Puma, Unicorn or Sidekiq based on CPU utilization. Or response time. Or requests per second.
    - The optimal Puma configuration for 80% of apps is probably 4 workers, 5 threads on a 4vCPU machine with ~8GB of memory.
  - Support parallel git worktrees
  - Postgres
    - QueryTags in query logs, including the code namespace and function (Job, Controller+Action, etc) as well as the trace id
    - Have automation in place to reindex bloated B-tree indexes
      - https://github.com/vitabaks/pg_auto_reindexer
    - Periodically cleanup table bloat with pg_squeeze
    - Set aggressively low timeouts
      - Set a low **statement_timeout** (e.g. 10s)
      - Set a low **idle_in_transaction_session_timeout** (e.g. 3s)
      - Set a low **lock_timeout** (e.g. 3s)
      - Set a low **transaction_timeout**
      - Set an **idle_session_timeout** (e.g. 30s), set a slightly lower timeout on the application connection pooler
    - Lock as little as possible, lock contention is a performance killer

    - Hold locks for the shortest possible time. This goes hand in hand with keeping transactions as short as possible, which you should also do
    - Set **default_toast_compression** to **lz4**
        - https://x.com/gwenshap/status/1993114834435817786
    - Follow pganalyze's performance checklist
    - Use https://pganalyze.com/ to track performance, load, and help discover bottlenecks
    - See other Postgres specific notes
- Tests
    - Use minitest + https://github.com/grosser/maxitest
    - Use fixtures, not factories
    - Postgres
        - Use tablespaces and put tables on tmpfs (RAM)
            - https://gajus.com/blog/setting-up-postgre-sql-for-running-integration-tests
        - Use unlogged tables
        - **fsync=off**
        - **full_page_writes=off**
        - **synchronous_commit=off**
        - **autovacuum=off**
        - **checkpoint_timeout=60m**
        - **wal_level=minimal**
        - **max_wal_senders=0**
    - Enforce a maximum test runtime
    - Fix or delete flaky tests
    - https://github.com/basecamp/gh-signoff and don't use a CI
- Preconnect cross-origin domains
- Compress assets + use a CDN
- Compress response payloads (HTML/JSON)
- Sidekiq jobs
    - Leverage https://github.com/sidekiq/sidekiq/wiki/Iteration
    - SLO based queue naming (**within_6_hours**, **within_0_seconds**, etc.)
        - **Idea:** expand **sidekiq_options** to accept **slo** and **weight** to pick/generate a queue name which is automatically used
            - **slo: 5.minutes, weight: 0.5**

        - Manual load shedding
            - e.g. Short-circuit all jobs of class X with argument[0] = Y
        - Implement Sidekiq batch invalidation by default for every job
        - Provide fairness between tenants
            - https://brooker.co.za/blog/2026/02/25/sfq.html
    - Avoid using UUIDs at all
        - Use bigint primary keys, for public identifiers use sqids with a secret
    - RMP in all environments
    - RUM in browser
    - Autoscale web and worker
    - Instrument request queue time
    - Automated alerts/SLOs in Terraform
    - Use Prosopite or enable Strict loading
        - https://github.com/charkost/prosopite
        - **config.active_record.strict_loading_mode = :n_plus_one_only**
            - Enable for every model: **config.active_record.strict_loading_by_default = true**
            - In production set **config.active_record.action_on_strict_loading_violation = :log** to avoid raising an exception, but set up an alert for these.
            - Disable it in console: **console { ActiveRecord::Base.strict_loading_by_default = false }**
            - Disable it in factory setup:

```
# spec/factories.rb

FactoryBot.define do
  after :build do |record|
    if record.is_a? ActiveRecord::Base
      record.strict_loading! false
    end
  end

  after :stub do |record|
```

```
      if record.is_a? ActiveRecord::Base
        record.strict_loading! false
      end
    end
  end
```

- jemalloc
  - Turbo/Datastar/HTMX
  - Mise Tasks
  - Apply limits to every feature
    - You'll regret not adding limits from the beginning
  - Code formatting
    - Standardrb
    - Enforce with git hooks https://hk.jdx.dev/
    - fasterer: ⚡ Don't make your Rubies go fast. Make them go fasterer™ ⚡
    - hone: Find performance optimization opportunities by combining static AST analysis with runtime profiling data
  - https://evilmartians.com/chronicles/gemfile-of-dreams-libraries-we-use-to-build-rails-apps
  - Enforce a zero-bug policy
  - All "Repository" operations must be batched
  - Setup a CSP policy from day one
  - Track performance regressions
    - **Metric A:** What I want to monitor, e.g., # of requests that took longer than 5 seconds, grouped by controller action
    - **Metric B:** Same, but timeshifted week ago
    - **Alert:** B/A > 2
  - Protect against CSRF attacks by leveraging **Sec-Fetch-Site** instead of old-school CSRF tokens
    - https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site_Request_Forgery_Prevention_Cheat_Sheet.html
  - Protect against SSRF attacks
    - Very relevant if you support user-defined webhooks
    - https://cheatsheetseries.owasp.org/cheatsheets/Server_Side_Request_Forgery_Prevention_Cheat_Sheet.html
  - Feature flags
    - Support a tree-like structure, since some feature flags are nested by nature. Enabling an child node should enable all parent nodes

        - This can be done simply with a **parent_feature_ids** array column, but you may also leverage the ltree extension
    - Notify when feature flags are fully rolled out or fully disabled, every week
        - As a reminder to delete dead code branches
- https://github.com/speedshop/gvl_metrics_middleware
- Setup span/slog fields for metrics/signals/monitoring, per request/job:
    - Tenant ID
    - User ID
    - Feature Flags
    - Error code
    - Number of DB queries
    - Number of DB tables queries
    - Number of DB services hit
    - Number of Elastic queries
    - Number of Elastic services hit
    - Number of cache reads
    - Number of cache writes
    - Number of cache services hit
    - Number of object allocations
    - Number of HTTP requests
    - Number of HTTP request retries
    - Number of HTTP requests uniqued by domain
    - Time spent in DB
    - Time spent reading from DB
    - Time spent writing to DB
    - Time spent in cache
    - Time spent reading from cache
    - Time spent writing to cache
    - Time spent in view layer
    - Time spent in HTTP requests
    - Time spent
    - CPU wall time spent