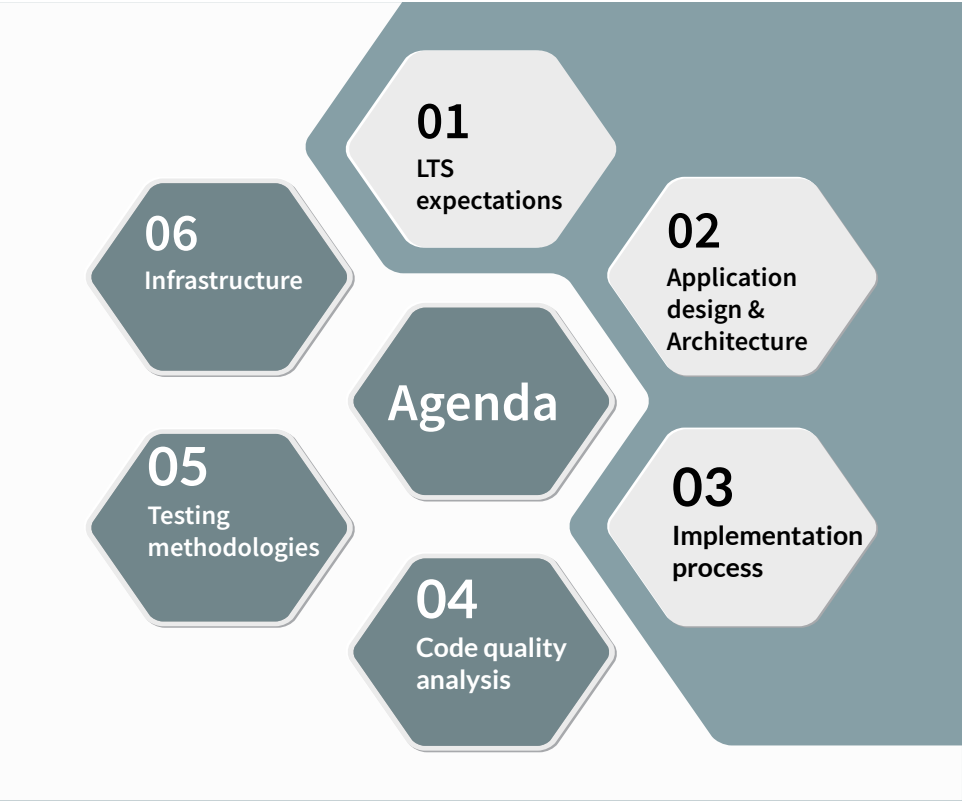
Large Scale Applications using Ruby on Rails

by Parth Barot

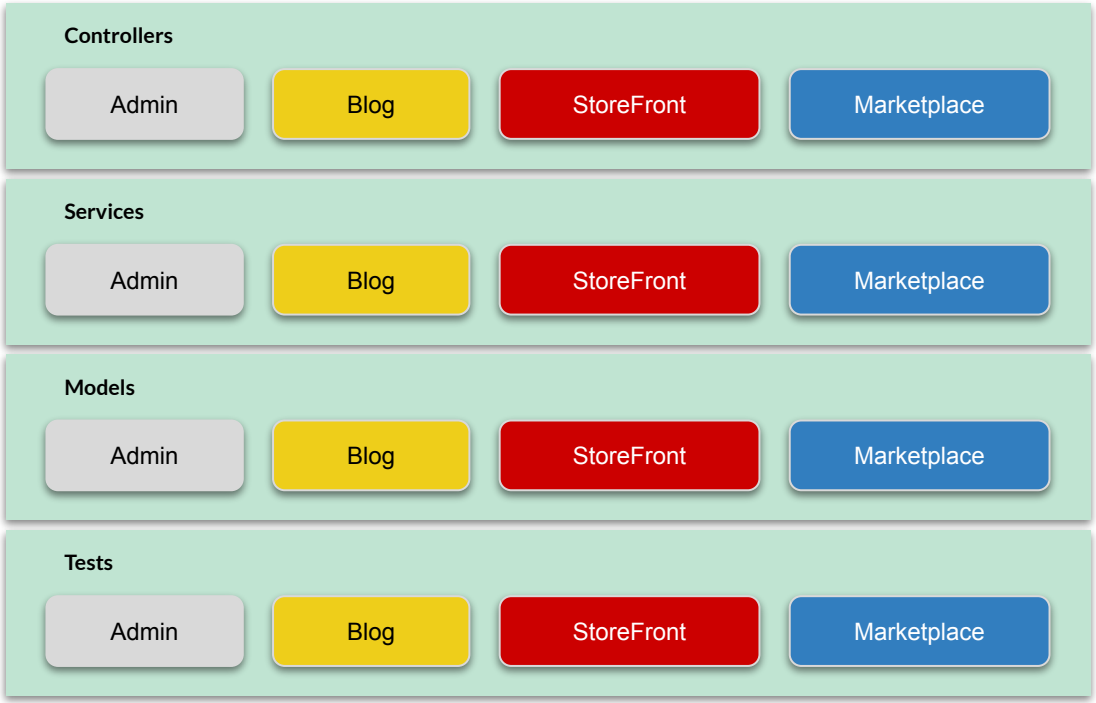
Agenda


01
LTS expectations
02
Application design & Architecture
03
Implementation process
04
Code quality analysis
05
Testing methodologies
06
Infrastructure
Agenda

Long Term Support - Expectations

- Simple and better architectural code and infrastructure design - easy to understand, change, maintain and support, monolith initially.
- Detailed documentation.
- Testability across application, with automation.
- Scalability

Application Design



Application Design

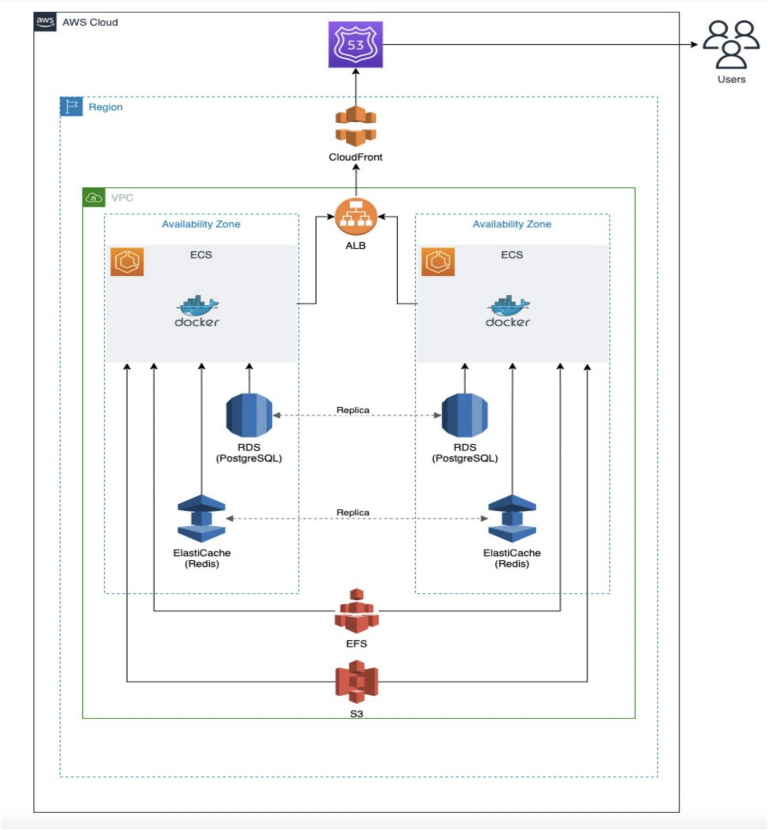
Initially, we have to keep it simple!

- Rails conventional structure with MVC / MVCS - Separate our code into components such that each layer is easily testable with unit tests!
- Rspecs/tests, having 90-95% test coverage
- Documentation in the code, following style guide and use README whenever needed
- *Scalability, later - with multiple nodes OR divided into microservices OR using serverless?*

Advanced Ideas

- Define modules for each component layer to group your components
- Usage of Rails engine to implement each feature as a plugin (like, blog feature as a separate engine)

Infrastructure Architecture - V1



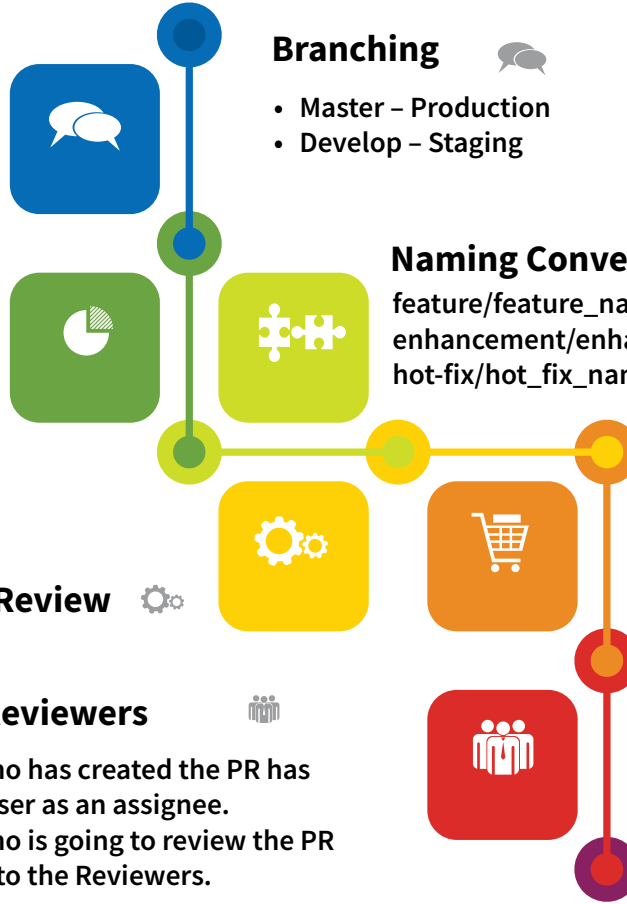
Infrastructure Architecture - V2 (future)

- Microservices based decomposition of monolithic applications
    - Extract out services based on functional responsibilities (Like, Blog, Store front and Marketplace can be extracted as separate apps)
    - User service discovery (Consul, etcd, Eureka OR kubernetes for more stuff).
    - Implement auto-scaling for each microservice as required.
    - Service could be another EC2 OR small serverless component (Like, small task of convert file formats into PDF OR image can be achieved by AWS Lambda and load will be reduced from the core app)

Implementation Process

01	Analysis	• Understanding, assumptions, Grey areas • Application workflow, Identify Approach “HOW” part • High level estimates
02	Design	• App & DB design, system architecture, Security • Coding standards, components, patterns • Build base framework with folder structure
03	Define Process	• Git, code review, merging, deployments • Agile project mgmt. With sprints • Documentation & deployment tools (Jira, CI/CD etc)
04	Product Implementation	• Sprint - Planning, Grooming, Run, Demo & Retro • Self functioning team, motivated, initiatives, concise communication, reduce tech. Debt • Asynchronous working style
05	Pre-Release	• Release docs, Automated deployment • Monitoring setup, User demos, guides • UAT & signoff
06	Live Release	• Live support, Monitoring & analysis, performance tuning & scaling • User feedback, prepare roadmap for next release

GIT Process



Branching

- Master – Production
- Develop – Staging

Branch Creation Criteria

- Creates a branch from Master when creating a new branch for a feature.
- Creates a branch from develop when creating a new branch from the developing branch.

Naming Convention

feature/feature_name, feature/sub_feature_name.feature_name, enhancement/enhancement_name, fix/fix_name, hot-fix/hot_fix_name

PR Labels

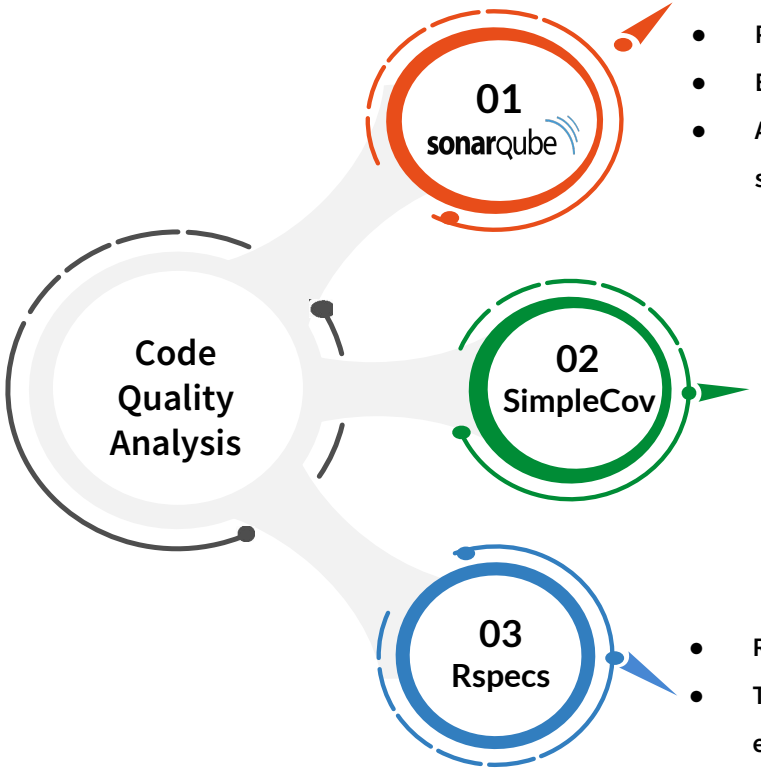
- There will be labels that you've to assign to the PR.
- If the feature is working, the PR label will be - WIP.
- If PR is ready to merge then the label will be - Ready To Merge
- If PR is reviewed and has unresolved comments or conflicts then the label will be - Do Not Merge

PR Creation & Review

Assignees & Reviewers

- The person who has created the PR has to select the user as an assignee.
- The person who is going to review the PR will be added to the Reviewers.

Conflict Resolving

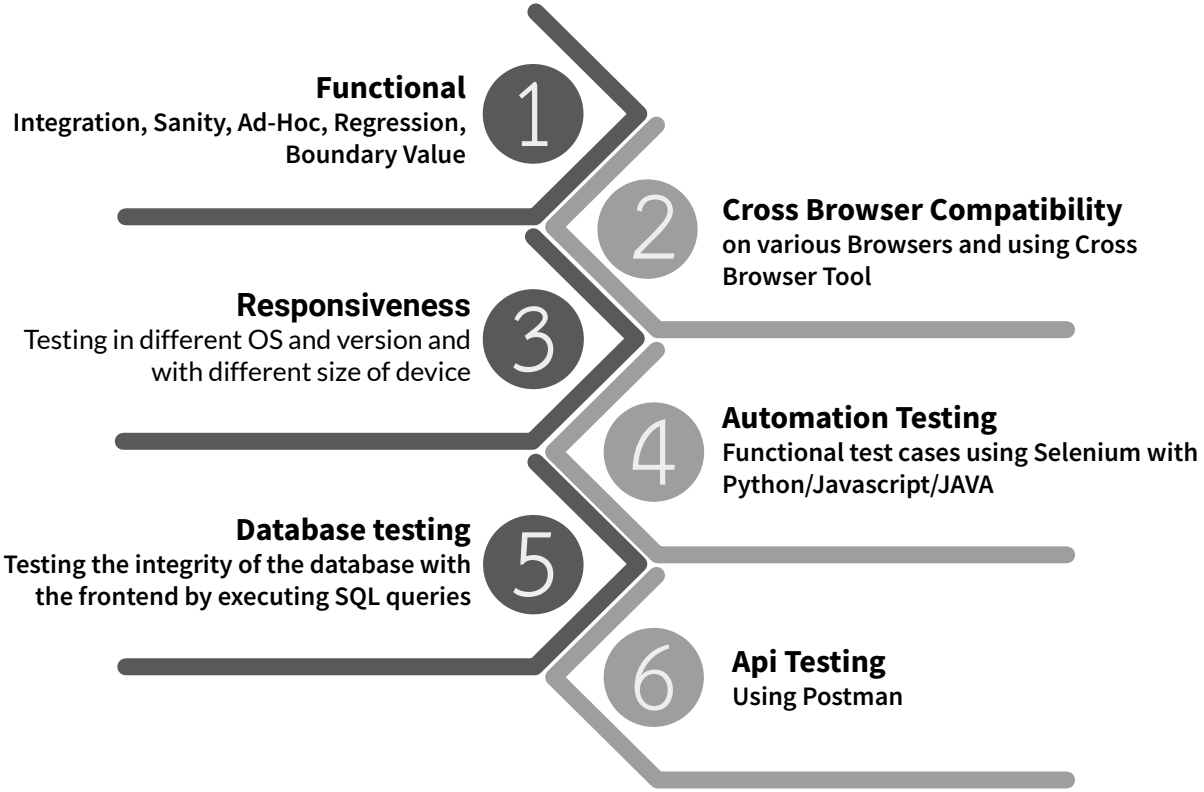


- Collects and analyzes source code.
- Provide reports for the code quality of the project.
- Enables measuring quality continuously over time.
- Analyze where the code is messing up and determine whether it has styling issues, code defeats, code duplication, complex code.

- SimpleCov is a code coverage analysis tool for Ruby.
- Provides clean API to filter, group, merge , etc. and display those results, giving us a complete code coverage suite.
- Cache and merge results when generating reports, which includes coverage across test suites and provides a better picture of blank spots.
- Coverage of rspecs via integrating SimpleCov gem of Rails.

- RSpec is a Ruby programmers' Behaviour-Driven Development tool.
- The executable example tests whether a portion of code exhibits the expected behaviour in a controlled context.
- Test for the API endpoints exposed for web frontend, ensures that they are returning the correct responses.

Testing Methodologies



Infrastructure

- Cloud services - DigitalOcean, AWS, Google cloud, MS Azure
- Cloud management using CI/CD - Kubernetes, Docker, Jenkins/CircleCI/TravisCI
- Automated code reviews - SonarQube/CodeClimate/Codacy
- Monitoring tools - Sentry, Newrelic
- Git repository - Github, BitBucket
- Project management & ticketing tools - Jira+Confluence, Asana, Basecamp, github issue and proj. mgmt.
- Collaboration - Slack, Hipchat, Flock, MS Teams, Skype business, Discord, Ryver
- Meetings - Zoom, Google Meet, MS Teams, Skype, Whatsapp
- Documentation - Confluence, Github Wiki, Google Docs

Thank You

Contact Us

contact@tntra.io

www.tntra.io