Case Study

Moving from Monolith to Microservices Infrastructure

✔ About Our Client:

Our client is the USA's best-known **B2B Fintech services provider** and a hub that helps professionals to use/centralize/integrate with the various tools for better business growth. They approached us to boost their product development and streamline the existing ecosystem. We understood their project requirements carefully and provided an experienced product development team. Our team offered best-in-class solutions for the two software products they wanted to enhance.

✔ Challenges We Faced:

1. One of the project requirements was to deconstruct the legacy architecture of our client's community communication website, built in WordPress. Our dedicated development team migrated it into microservices and rebuilt it into a top-notch technology- Ruby on Rails to leverage technological advantages.

2. The client also wanted to implement multiple functionalities in the website like user authentication, billing, integrations, and many more. To work on these requests, we followed specific testing and deployment procedures for better growth and customer experience to take the system to the next level.

3. The software products on the client's website had a manual release cycle which occupied a bunch of team members doing the dynamic work. To make our client's primary focus on core business activities, we streamlined the process and kept the routine efficient and easily manageable for them.

✔ Solutions We Offered:

1. For better website performance, we optimized the interactions with the help of smart caching. We used Redis to migrate our client's monolith WordPress-based blog website into microservices and rebuilt the community platform with Ruby on rails.

2. To follow different procedures for testing and deployment, we used virtual containers provided by Docker for microservice interactions. Our team of developers used Heroku and PostgreSQL to create the unification.

3. To give quick access to any new team member, we ensured that the architecture built for them should be based on the subject matter and entities/logic they are familiar with. Entities like the project, integration, card, etc. To make it happen, our skilled and experienced developers used Trailblazer to implement the domain-driven design on RoR.

✔ Tech Stack We Used:

- **Technologies & Languages:** Angular, Ruby on rails, Vue.js, Rspec,
- **Frameworks:** Active Admin, Capybara, Sidekiq,
- **Library:** EventBus
- **Tools:** Selenium, MiniTest
- **Database:** PostgreSQL, Redis,
- **Deployment and hosting:** Docker, Heroku, AWS,
- **3rd party integrations:** Stripe, Slack, Jira, Zapier, Sentry**,** Algolia, Livechat, Hootsuite, Twitter, Sales navigator, Google Analytics, Shopify, Salesforce and Datadog

✔ Team Required

- 3 Full Stack developers (Ruby on rails, Vue, Angular)
- 2 DevOps engineers
- 1 Cloud engineer
- 1 Project manager
- 1 Business analyst
- 1 Quality analyst