Jacob Victor Berg

Full Stack / Senior Software Engineer / Site Reliability Engineer

(865)804-5952 / jacobvictorberg@gmail.com / Indianapolis, IN

Certifications: AWS Certified Developer, AWS Solutions Architect Associate

Education: IU Bloomington(2007-2012), Kenzie Academy(2018-2019)

Experience: DMI - Consulting at Eli Lilly (2018-Present)

Maestro - Study Orchestration (2024-Present):

Assisted lead development of data processing system utilizing Temporal Workflows, plug-in architecture, GraphQL, DynamoDB/NoSQL, Saga Pattern. Built out a React Flow-based rule management system to enable configurations for non-technical users, enhancing scalability and adaptability.

- Used Temporal Workflows to orchestrate clinical trial data, ensuring easy scalability and reliability via a distributed and microservice architecture. Temporal Workflows were utilized for automatic updates to system data via triggers and schedules, managed by a plugin architecture system.
- Deployed containerized applications using OpenShift, ensuring rapid scaling. Automated release process using GitHub Actions / Openshift Templates, Argo CD, maintaing parity between environments with environment variables.
- Utilized backing services such as S3 to: manage plugins for Plugin Architecture, store all actions between Temporal Workflows with large datasets, and for auditing.
- Enhanced disposability and robustness of Temporal worker by designing a remediation mechanism into system startup. Pulled all existing plugins from S3 to create or delete workflow schedules that were out of sync during system downtime.
- Conducted impact analysis for integrating React Flow based business rules management, ensuring alignment with non-technical user workflows.
- Created dynamic workflows for trial coordinators to adapt patient activities based on clinical protocol requirements using React Flow, Next.js, JDM models with Zen Engine.
- Streamlined local development of microservices using Docker, docker-compose, and Makefiles. Implemented localstack to spin up AWS services locally(DynamoDB, S3) through docker-compose.
- Designed and implemented a scalable audit log system using AWS Kinesis Streams and DynamoDB to handle high-throughput updates and real-time monitoring.
- Improved workflow execution speed by optimizing Temporal plugin queries and batching fetches.
- Used PynamoDB Models and Strawberry GraphQL types to maintain a strict adherance to schema.
- Designed GraphQL mutations and queries to using repository structure for multiple tables.
- Integrated Apollo Client to optimize queries and cache trial data, reducing redundant API calls. Also used Redis for backend caching and smart updates
- Manage DNS, certificate creation, and load balancing for UI application.
- Used **DynamoDB Global Secondary Indexes (GSI)** to sort and query entity updates by their last modified timestamps efficiently.

LillyDev SRE Responsibilities & IMS(Inventory Management System) (2021-2024):

Managed shared DevOps services and infrastructure-as-code using CloudFormation, scaling enterprise tools on ECS. Led the IMS project to optimize SaaS resource usage by 20%+ via event-driven microservices with SNS, SQS, and Lambda, improving automation and reducing enterprise footprint.

- Built serverless architectures using AWS Lambda, SNS, and SQS to streamline data flow and reduce latency. Utilized SQS retries for system error handling.
- Developed event-driven architecture using Lambda, SNS, and SQS to process and notify changes in SASS inventory states. Used SQS and SNS to trigger different Lambdas to update system data. Created "resource scanners" and "resource loaders" to distribute work across the system.
- Reduced Heroku and Contentful utilization by lowering enterprise footprint by approx 20% in, spaces, apps and resources. Using the state machine pattern, warnings and notifications were triggered by AWS EventBridge and sent to application teams. This allowed the product owner to confidently delegate the spin down unused resources.
- The refactor allowed for resources to scale during resource collection and loading. This reduced resource collection time by half, drastically improving development efficiency.
- Enhanced disposability and robusteness with a remediation Lambda that would scan for ownership status system was offline during resource collection.
- Managed ingress and egress for Lambdas, EC2, Postgres, and DynamoDB in AWS.
- Utilized roles and security groups for granular access to resources created for applications.
- Created offboarding system for Heroku, Contentful, Slack, as well as management tools for updating user/team configurations via Python based Lambdas.
- Created admin access only specific page to update critical data, using JWTs, OAuth, Azure groups. Admin could update important system data for Heroku Stack lifecycle updates. Admin would stage outdated/unclaimed Heroku applications for spin-down.
- Docker with docker-compose for local development of backend event-driven services. Implemented localstack to spin up AWS services locally(SNS, SQS).
- NPM workspaces to structure monorepo of multiple services and modules. Including Lambda Layers that were dynamically structured for deployment.
- Gained expert understanding of CI/CD practices GitHub Actions, Cloudformation, AWS CLI, Docker, Artifactory by Vue, React, ECS apps from CodePipleine for better CI/CD.
- Worked closely with product owner of multiple SASS products to develop and architect multiple programs and applications ensuring user management, resource management, authentication.
- Used Jest to ensure end to end test coverage for the Next.js application.
- Designed GraphQL API that used Dataloader for cached querying to SQL DB.
- Created pagination mechanism via GraphQL for paging through large DB queries.
- Simplified user access setup in Graphql API by creating user context class that checks the users token and if call came from an internal app.
- Designed DynamoDB table schemas to optimize for high-volume reads/writes.
- Manage DNS, certificate creation, and load balancing for multiple UI applications.
- Utilized health checks and rolling updates to ensure system stability.
- Coached and mentored interns about best practices.

eCTS (Enterprise Clinical Trial System) (2020-2021):

Modernized a legacy clinical trial system with AWS Lambdas, API Gateway, and serverless architectures, aligning with strict business and compliance goals to enhance scalability and performance.

- Redesigned legacy systems into microservices for improved modularity and maintainability. Utilized Lambdas and shared code to share data between microservices.
- Built a serverless architecture API that was agnostic to the original system and schema so that data could be constumed accross enterprise. Used transformer classes to translate inconsistant data in legacy database that was still required.

- Built React app with Redux Saga as state management system.
- Manages form updates in React with higher-order components.

Data Marketplace (2020):

Developed a responsive React frontend with CI/CD pipelines using AWS CloudFront and CodePipeline, enabling rapid prototype iterations and deployment for evolving business goals.

- Collaborated with UX teams to align evolving user interface designs with business objectives, ensuring rapid prototyping in React / Next.js.
- Project required agile, flexible UI development, and I worked directly with product owners and UX to rapidly iterate on prototypes that aligned with evolving business goals.
- Utilized Redux and React Context for state management.
- Created loading pattern components for more structured and reliable page updates.

Lilly Trial Guide (2019-2020):

Migrated a public-facing application to a Next.js/Express stack, integrating with Contentful and ElasticSearch to improve scalability and accessibility, while ensuring compliance with company standards.

- Heroku application deployment with environments including a pre-production environment for blue green deployment.
- Used Contentful resources to dynamically build React components based off of resource type and content, giving control for site content to business, including multilingual support.
- Maintained CMS model management system to synchronize Contentful environment.
- Created script to run and update Typescript types pulled from Contentful.
- Used Jest to ensure robust test coverage for the Next.js application.
- Used GraphQL for efficient data retrieval and integrated it with a responsive frontend.
- Integrated Apollo Client to optimize queries and cache trial data, reducing redundant API calls.
- Customized Material UI for component library with Styled Components.
- Expert understanding of React code management with Context and Hooks.