**Akhil**

**A hexagon with white text and a check mark

Description automatically generated**

**A blue hexagon with white text

Description automatically generated**

**PH: +19105576339**

**Email:** [**Hr@seafysoft.com**](mailto:Hr@seafysoft.com)

**GoLang Developer**

**Professional Summary**

* Golang developer with 9 years of experience in building high-performance backend services and scalable microservices. Skilled in optimizing application performance, developing robust APIs, and managing cloud deployments.
* Developed robust, high-performance backend services using **Golang** and implemented a microservices architecture to improve overall system efficiency, scalability, modularity, and maintainability.
* Experience in Golang concepts such as **slices**, **maps**, **structs**, **interfaces**, **goroutines**, and **channels**, with extensive knowledge of microservices, **Go kit**, **ORM GORM**, **dependency injection**, and **Gorilla Mux**.
* Experienced in using **gRPC** to implement high-performance, scalable communication between microservices and distributed systems. Leveraged **gRPC’s** support for bidirectional streaming and multi-language environments to enhance real-time data exchange and integration in complex architectures.
* Utilized **Golang's powerful concurrency model** to build efficient, scalable applications capable of handling high loads and performing parallel processing.
* Proficient in the **Echo framework**, with extensive experience in implementing and managing diverse types of APIs.
* Developed multiple services that interact with each other using **Protobuf** as the data exchange format.
* Expert in **Golang** with a strong understanding of **goroutines**, **event handling**, **error handling** and communication through **channels**.
* Successfully led the migration of complex applications from **Java to Golang**, achieving seamless integration with existing systems, while significantly improving performance and enhancing code maintainability.
* Designed and developed **RESTful APIs** with Golang, ensuring seamless integration and communication between services.
* Utilized Golang's powerful **concurrency** **model** to handle high-load processing tasks efficiently.
* Experienced in using **Golang** to interact with databases such as **MongoDB** and **Redis**, performing **CRUD** operations to create, update, retrieve, and delete data. Additionally, utilized **Redis channels** for efficient communication between services.
* Proficient in using **Redis** for key-value storage and data processing.
* Proficient in using **Golang drivers** to perform **CRUD** operations and write complex queries with **PostgreSQL** and **SQL Server**.
* Hands-on experience with web services including **REST**, **SOAP**, **OAuth2**, **API development**, and working with SQL and NoSQL databases.
* Experience in working with IDEs such as **GoLand**, **Visual Studio Code**, **Sublime Text** and **LiteIDE** for developing, deploying, and debugging Go applications.
* Experienced in defining automated test strategies using unit testing and mocking frameworks including Go's **testing** package and **testify**. Developed automated scripts for smoke and regression testing using Go testing tools.
* Developed backend systems in **Golang** for applications like Limit Monitoring and Feedback systems.
* Hands-on experience in developing interactive web pages using **HTML**, **CSS**, **JavaScript**, **TypeScript**, **AngularJS**, **ReactJS**, and **AJAX** for creating user interfaces and enabling client-server communication.
* Configured and created application log files using loggers like **logrus** and **zap** to trace required application messages. Used monitoring tools such as **Splunk**, **Kibana** for log aggregation and monitoring.
* Spearheaded the migration of legacy systems to AWS cloud infrastructure, leveraging services such as **EC2**, **S3**, **Lambda**, and **RDS** to improve reliability and reduce operational costs.
* Experience with Cloud Computing environments like Amazon Web Services (AWS) and Amazon cloud technologies such as Elastic Compute Cloud (**EC2**), **Simple Storage Service** (**S3**), **Auto Scaling**, **IAM**, **CloudWatch**, **AWS CLI**, **Route53**, **CloudFormation**, Elastic Load Balancers **(ELB)**, **RDS**, **SNS**, **EBS**, **EKS, DynamoDB**, **ECR** and **CI/CD** tools.
* Experience with Test Driven Development writing UI Unit tests before the functional code in **Jasmine**, **mocha**, and **Jest**.
* Designed and deployed **Docker** images for core components, ensuring consistent and portable application deployment.
* Implemented and managed **Kubernetes clusters** for orchestration and scaling of containerized applications, automating application deployments with **Ansible** for streamlined operations.
* Successfully deployed applications on Cloud Services, including Microsoft Azure and Amazon Web Services (AWS), utilizing advanced CI/CD practices through **Azure DevOps**, **Azure Pipeline**, and **AWS Code Pipeline**.
* Proficient in using Golang to work with messaging systems such as **RabbitMQ**, and **Apache Kafka** to communicate between client and server by writing producers and consumers. Skilled in creating and using **Kafka streams** and **Kafka connectors**.
* Gained practical experience with **GIT** within the Microsoft Azure Platform Service environment, leveraging version control for efficient collaboration.
* Hands-on experience in building monitoring solution using **Prometheus**, **Grafana**, **push gateway**, **alert manager** and deployment using **ansible**.
* Experienced in working on Software Development Life Cycle (**SDLC**) with a blend of proficiency and leadership in **AGILE-SCRUM** and **Test-Driven Development** (TDD).
* Experience in working with Agile and Waterfall methodologies and worked on various Service Oriented, **Client/Server**, **GUI**, and **Web based applications**.
* Coordinate with cross-functional teams to execute short and long-term product delivery strategies, with a successful track record of implementing best business practices.

**Technical Skills:**

|  |  |
| --- | --- |
| **Operating Systems** | Unix, Linux, Windows, Mac |
| **Programming Languages** | Golang, Python, C, C++ |
| **Query Languages** | SQL, PL/SQL |
| **Golang** | Goroutine, Channels, Echo framework, Gorilla Mux, Gin, gRPC, Microservices, ORM GORM, CLI, Go-kit, Go-hystrix, Go-CD, GoLand, GoDoc |
| **Databases** | MongoDB, Redis, PostgreSQL, SQL Server |
| **Web Technologies** | HTML, CSS, Java script, Typescript, ReactJS, Angular JS, Redux, AJAX, JSON |
| **Web Services and APIs** | RESTful APIs, SOAP, HTTP/HTTPS, OAuth2, OIDC, GRPC, Protobuf |
| **Web Servers** | Nginx, FastAPI |
| **IDE** | Visual Studio Code, Sublime Text, LiteIDE, Vim, IntelliJ |
| **Software Development** | Agile-SCRUM, Test-Driven Development (TDD), Service Oriented Architecture (SOA), Concurrency Model |
| **Cloud Technologies** | AWS (EC2, S3, Lambda, RDS, IAM, Cloud Watch, AWS CLI, EBS, ELB, Route53, CloudFormation, SQS, SNS, DynamoDB, ECR, VPC, CloudTrail, EMR, Elastic Beanstalk), Amazon Neptune, Microsoft Azure, Terraform. |
| **CI/CD and Version Control** | Jenkins, Git, CI/CD Pipeline, Azure DevOps, AWS Code Pipeline |
| **Containerization, Orchestration, and Automation** | Docker, Kubernetes, Ansible |
| **Monitoring and Logging** | Prometheus, Grafana, Alert manager, Push Gateway, Splunk, Kibana, Logrus, Zap |
| **Messaging and Streaming** | Rabbit MQ, Apache Kafka (Kafka streams & Kafka connectors) |

**Professional Experience**

**Client: Uptycs Inc, Waltham, Massachusetts. November 2022 to Present**

**Golang Developer**

**Project-1 Description:** In this project, I played a key role in developing and optimizing microservices using Golang, with a strong focus on enhancing performance, scalability, and security. A major aspect of the project involved transitioning critical services from Python to Golang, which led to significant improvements in both throughput and memory efficiency. I also implemented advanced concurrency patterns and ensured robust security protocols for gRPC and RESTful services. The deployment process was streamlined through the integration of Docker and Kubernetes, and I architected a scalable infrastructure using Amazon Web Services (AWS) Additionally, I developed dynamic web applications by integrating React.js with the Golang backend and utilized Kafka for real-time data processing.

**Responsibilities:**

* Worked in Agile environments, collaborating closely with cross-functional teams to align with customer expectations, meet deadlines, and deliver high-quality products, which improved project management and development processes.
* Developed and maintained gRPC and RESTful microservices in Golang, using MUX router for efficient web service implementation and HTTP services for smooth service interaction.
* Leveraged advanced concurrency techniques such as Wait Groups, goroutines, and channels to significantly boost application performance. Applied Singleton and Interface design patterns for enhanced code reusability and maintainability.
* Secured gRPC and RESTful web services by implementing SAML and JWT tokens, ensuring secure and authenticated communication across services.
* Designed and executed comprehensive test cases for Unit Testing using the testing and testify packages, ensuring code reliability, performance, and accuracy, leading to a stable codebase.
* Enhanced program efficiency and performance by implementing goroutines in API development, particularly for applications requiring high concurrency and parallel processing.
* Collaborated with clients to gather and analyze requirements, resulting in detailed Requirement Specification Documents that ensured accurate project scopes and successful delivery.
* Engineered microservices in Golang to process large datasets from databases like PostgreSQL and MySQL, optimizing data retrieval and processing to enhance overall system performance.
* Led the redesign of data pipelines within existing microservices, resulting in improved efficiency and reduced latency. Successfully migrated critical microservices from Python to Golang, significantly improving throughput and memory usage.
* Developed applications capable of handling concurrent data collection and processing for Wi-Fi devices using Goroutines and channels, ensuring efficient and timely data management.
* Created and managed Docker images using Dockerfile, optimized container snapshots, and utilized Docker containers to maintain consistent environments across development, testing, and production stages.
* Built high-performance APIs using Protocol Buffers and gRPC, utilizing protobuf for efficient data serialization and communication between services.
* Implemented various Golang libraries such as go-fmt, go-vet, and go-dep, and addressed race conditions by using the race flag, ensuring thread-safe and reliable code execution.
* Developed a container platform for Kubernetes using Go, facilitating efficient orchestration and management of containerized applications.
* Implemented routing in Golang using Gorilla Mux to handle incoming requests and direct them to appropriate handlers, ensuring efficient request processing.
* Used OpenAPI specifications with Go-Swagger to generate server-side code, streamlining development and ensuring compliance with API specifications.
* Integrated React.js front-end components with Golang backend services using REST APIs, creating dynamic and interactive web applications.
* Developed multiple RESTful services that interact seamlessly using JSON for data exchange, enabling efficient communication and data transfer between services.
* Created RPC-based services for processing large datasets from PostgreSQL using Golang, ensuring high performance and reliability in data-intensive applications.
* Integrated Apache Kafka for real-time data streaming, handling high-throughput, low-latency data processing, which enhanced the responsiveness of critical application services.
* Architected and managed infrastructure on Amazon Web Services (AWS) utilizing various AWS services to ensure scalable and efficient cloud-based solutions.
* Executed manual test steps based on test cases and reported bugs in JIRA, contributing to overall quality assurance and continuous application improvement.

**Environment**: Golang, gRPC, RESTful, MUX router, HTTP services, Wait Groups, goroutines, channels, Singleton, Interface design patterns, SAML, JWT tokens, Unit Testing, testing, testify, PostgreSQL, MySQL, Docker, Dockerfile, Protocol Buffers, protobuf, go-fmt, go-vet, go-dep, race flag, Kubernetes, Gorilla Mux, OpenAPI, Go-Swagger, React.js, JSON, RPC, Apache Kafka, AWS, JIRA, Agile

**Project-2 Description:** In this project, I played a crucial role in transitioning a large-scale application from a Java-based architecture to Golang, significantly enhancing the performance and scalability of the system. I engineered microservices and RESTful APIs using Golang, integrating them with other system components to create a robust backend. Additionally, I developed user interfaces using React.js, JavaScript, and other front-end technologies to deliver a responsive and engaging user experience. The project involved extensive use of AWS for deployment, as well as Docker and Jenkins for CI/CD, ensuring seamless deployment and operational efficiency. I also led the integration of Kafka for efficient data streaming, enhancing the application's real-time processing capabilities.

**Responsibilities:**

* Collaborated with clients to capture and clarify requirements, ensuring precise solution demonstrations and a clear understanding of project goals.
* Acquired expertise in Golang through active project involvement, transitioning from Java to Golang to leverage its powerful features for improved coding efficiency.
* Developed scalable and high-performance backend microservices using Golang, ensuring seamless integration with system components and enhancing application functionality.
* Built and managed RESTful APIs using a Node.js server, handling backend requests and optimizing Ajax calls for efficient data retrieval and interaction.
* Designed and implemented responsive user interfaces with HTML, CSS, JavaScript, jQuery, and React.js, delivering interactive experiences that boost user engagement.
* Applied advanced SQL programming skills to create and optimize queries and stored procedures, ensuring robust data management and efficient database performance.
* Created RESTful web APIs with Spring Boot, utilizing Spring Data for MongoDB integration to enhance data handling and application performance.
* Designed and implemented secure and reliable SOAP/HTTP web services using SOA technologies, ensuring effective web service communication.
* Developed web modules and middleware components using AWS and Spring Integration, facilitating seamless application deployment and integration within cloud environments.
* Engineered software modules and user interface components deployed on AWS, ensuring the scalability and reliability of applications.
* Utilized Docker and Jenkins to automate CI/CD pipelines, streamlining deployment processes to AWS and enhancing operational efficiency.
* Developed and consumed XML web services with a strong understanding of SOAP, ensuring effective and interoperable communication.
* Created and maintained SQL Server stored procedures, user-defined functions, and database objects, optimizing performance and ensuring data integrity.
* Developed scalable backend systems by creating RESTful/WCF web services and APIs, supporting complex application requirements.
* Conducted thorough unit and integration testing to ensure applications met specifications and maintained high quality before deployment.
* Integrated AWS with Kafka for real-time data streaming, enhancing the application's efficiency and synchronization capabilities.
* Led the migration of projects from Java to Golang, optimizing codebases for better performance and scalability.

**Environment:** Golang (Goroutines, Microservices, Channels, Gokit, GORM), CLI tools, Go-hystrix, godoc, Ginkgo, Gorilla Mux, gRPC. Front-end development with HTML5, CSS3, Redux, Bootstrap. REST APIs using Golang, Spring Boot, JavaScript (ES6), TypeScript. AWS (Connect, DynamoDB), Apache Kafka, Docker, Elasticsearch, JSON, MySQL. Build and version control with Maven, Git, SVN. SSO/SAML, Jira, Jenkins, Eclipse. Operated in a Linux environment, following Agile and Scrum methodologies.

**Project-3 Description:** Developed a GoLang application to automate the synchronization of images from AWS Elastic Container Registry (ECR) to a local container registry. This application periodically checks for updates in AWS ECR and pulls new images into the local registry, ensuring consistency across cloud-based and on-premises environments. The project reduces operational overhead, enhances reliability, and improves efficiency in the CI/CD pipeline by automating image deployment, ensuring all environments have access to the latest container images.

**Responsibilities:**

* Involved in the design, implementation, and deployment of the Full Software Development Life Cycle (SDLC) using Agile methodology and Test-Driven Development (TDD) for application development, utilizing JIRA as a project management tool.
* Designed and developed a Golang application to automate the synchronization of AWS ECR images with a local Docker registry, ensuring consistency across cloud and on-premises environments.
* Implemented periodic checks using AWS SDK for Go to identify and sync new or updated images, leveraging concurrency with goroutines and WaitGroups for efficient operations.
* Developed error handling, logging with Zap, and scheduling mechanisms to ensure reliable and consistent synchronization.
* Automated the deployment and configuration of the application using Docker and integrated it into the CI/CD pipeline with the DevOps team.
* Implemented Slack notifications to alert the team of new image updates, enhancing visibility and responsiveness.
* Conducted testing, optimized performance, and provided documentation for the synchronization process.

**Environment**: Golang, AWS ECR, Local registry, AWS SDK for Go, image retrieval, error handling, network issues, synchronization failures, Go routines, logging, Docker, DevOps, CI/CD pipeline, Testing, scalability, efficiency, authentication, authorization, Slack, Jira, Software Development Life Cycle, Agile Methodology, Test-Driven Development, LiteIDE, Zap.

**Client: Uptycs, India August 2019 to August 2022**

**Golang Developer**

**Project-1 Description:** Developed a crucial self-healing tool to enhance the management and reliability of production systems. The tool significantly reduces manual intervention by automating system monitoring and issue resolution. It features a GoLang-based agent that manages data transmission, command execution, and secure communication between servers and the core system using Redis channels. The core functionality, also in Golang, processes API requests, automates data collection, and stores information in MongoDB and Redis. An API with middleware ensures secure user request handling and permission validation through Role-Based Access Control (RBAC). Docker images for the Core, API, and Agent guarantee consistent and portable deployments, while Ansible automates deployment, incorporating features like compaction, data archival, and Redis cluster formation to maintain a reliable and scalable system setup across all servers.

**Responsibilities:**

* Worked in an Agile Environment and participated in requirement gathering, analysis, and design.
* Developed an Agent for a self-healing tool using Golang to monitor and manage production systems.
* Utilized the Echo framework for creating and managing API endpoints for tasks like container status checks, command execution, and PostgreSQL and Redis operations.
* Included endpoints for managing Cron jobs, monitoring Akka compaction and archival status, and handling Kafka retention periods.
* Implemented secure communication through TLS configuration and leveraged Viper for dynamic configuration management.
* Enhanced DevOps teams' capabilities with comprehensive monitoring and automation, improving the reliability and efficiency of production systems.
* Core component of the self-healing tool is designed to manage the self-healing process for production systems by parsing the inventory file of a stack.
* Developed core functionality in Golang to process API requests, automate data collection from agents, and store data in MongoDB and Redis.
* Configuration management is handled by Viper, allowing dynamic updates and easy access to parameters.
* The core continuously monitors production systems and triggers automated actions to address issues, manage cluster data, handle cron jobs. This ensures system stability, timely task execution, and effective troubleshooting.
* Periodic tasks are scheduled to ensure integrity checks, data synchronization, and automated action execution using the scheduler package.
* Asynchronous operations are managed with goroutines, enabling efficient handling of user-defined actions, action executions, debug commands, cron operations, and other processes.
* The core ensures secure communication and monitoring of its health status, with a graceful shutdown mechanism to properly close Redis connections.
* Created API in Golang with Middleware for authentication and authorization, handling user requests securely and validating permissions using RBAC.
* Implemented custom middleware for logging, CORES configuration, authentication, and request recording, enhancing the API server's security, monitoring, and operational capabilities.
* Utilized Viper for configuration management, including dynamic reloading of configuration files, to ensure that the API server adapts to changes in real-time.
* Established connections to Redis and MongoDB, incorporating initialization and connection handling to support efficient data management and retrieval.
* Implemented RBAC with role and permission management, including user authentication, authorization, and session handling, to ensure secure access to API endpoints.
* Designed and implemented various API routes for managing user roles, performing actions on containers and nodes, handling Kubernetes operations, and integrating with Prometheus for metrics and alerts.
* Incorporated Prometheus handlers to fetch and manage metrics and alerts, ensuring effective monitoring and alerting for system performance and issues.
* Added endpoints for user management, debug operations, and automated actions, providing tools for troubleshooting and managing system health.
* Implemented Slack integration to notify users of significant events, including automated actions, cluster formations, data archival, ORC compaction, and Redis cluster formation, ensuring real-time updates and effective communication within the system.
* Implemented additional validation checks for critical actions by integrating Jira ticket validation to ensure proper authorization and requiring a valid Jira ticket for action execution. Incorporated RBAC to verify whether the user has admin privileges before performing important operations.
* Deployed a Nginx server to forward routes, effectively managing and directing incoming traffic to appropriate backend services. This setup helped optimize load balancing, improve response times, and ensure efficient routing of requests within the infrastructure.
* Containerized all components including CORE, AGENT, API, Redis, MongoDB, UI, and Nginx using Docker, ensuring consistent and portable deployment across various environments. This approach streamlined the development and deployment processes, enhanced scalability, and facilitated easier management and orchestration of services.
* Implemented a CI/CD pipeline to build Docker images for all components, including CORE, AGENT, API, Redis, MongoDB, UI, and Nginx. Utilized Jenkins to automate the process of pushing the built images to AWS ECR, ensuring streamlined and efficient deployment of containerized services.
* Developed Ansible playbooks for deploying the agent on all production monitoring systems, ensuring consistent and automated setup.
* Additionally, wrote deployment scripts for CORE, Redis, MongoDB, Nginx, and the UI components, streamlining the deployment process and ensuring reliable and scalable infrastructure management across all services.
* Established secure communications between CORE and AGENT using HTTPS, including generating and deploying SSL/TLS certificates to ensure encrypted data transmission and enhance security.
* Collaborate effectively as a team player and offer technical support to the devops team to ensure high-quality deliverables and adherence to project timelines.
* Implemented compaction, data archival, and Redis cluster formation using goroutines to handle concurrent tasks efficiently. Utilized mutex for synchronization and Go Kit for building robust microservices to ensure smooth and reliable operations.
* Enabled direct query execution in PostgreSQL and SQL Server from the tool, facilitating real-time data retrieval and aiding developers and DevOps teams in debugging and troubleshooting.
* Utilized Visual Studio Code and GoLand for our development for each component.
* Used staging environment before taking into the production system and used Agile methodology, Test-Driven Development (TDD) for application development, utilizing JIRA as a project management tool.
* Utilized AWS services like EC2, AWS Code Pipeline, ELB, EBS, ASG, S3 for this self-healing tool.

**Environment**: Agile Environment, Golang, Echo Framework, API Endpoints, PostgreSQL, Redis Operations, Cron Jobs, Akka, Kafka, TLS Configuration, Viper, MongoDB, Scheduler Package, Goroutines, Mutex, RBAC, Middleware, Logging, Kubernetes, Prometheus, Metrics, Alerts, Slack Integration, Jira Ticket Validation, Nginx, Docker, CI/CD Pipeline, Jenkins, AWS ECR, Ansible, HTTPS, SSL/TLS Certificates, Visual Studio Code, GoLand, Staging Environment, Test-Driven Development (TDD), JIRA, AWS EC2, S3, Lambda, Auto Scaling, ECS, EBS, Elastic Load Balancers, PostgreSQL, SQL Server

**Project-2 Description:** Developed a high-performance Golang utility to simulate thousands of virtual endpoints using HTTP and gRPC for comprehensive big data pipeline testing. This utility was designed to handle a range of requests including TLS, enroll, distributed read, and write operations. By leveraging Go routines, the system effectively managed concurrent processing, with each routine responsible for tasks such as asset enrollment, ensuring system stability, and transmitting data from the database. The utility sent extensive dummy data through the big data pipeline to evaluate performance metrics like latency and throughput, providing critical insights for optimization and scalability.

**Responsibilities:**

* Developed a high-performance Golang utility to simulate thousands of virtual endpoints using HTTP and gRPC, ensuring comprehensive testing of the big data pipeline.
* Designed the utility to handle a range of requests, including TLS, enroll, distributed read, and write operations, utilizing Go routines for efficient concurrent processing.
* Utilized gRPC for efficient communication between simulated endpoints, including defining and implementing service contracts using Protocol Buffers (Protobuf).
* Managed gRPC client and server configurations to handle diverse request types and ensure reliable data exchange.
* Tested all components of the big data pipeline, including Kafka, Spark, Presto, HDFS, Nginx, and TLS, to ensure functionality and performance.
* Specifically checked performance metrics such as latency and throughput for each component during every sprint to validate and optimize system performance.
* Validated performance by sending thousands of lines of data from each virtual asset and checking whether the data from all assets reached the intended destinations accurately.
* Conducted tests for various API calls, including enroll, distributed read, and distributed write operations, to ensure accurate performance metrics.
* Implemented logging mechanisms to capture detailed insights into the performance and behavior of simulated endpoints, aiding in troubleshooting and optimization.
* Deployed the utility using Docker containers, ensuring consistent and portable deployment across different environments.
* Integrated with AWS services, leveraging AWS infrastructure for scalable deployment and management.
* Created and managed all necessary cloud resources using Terraform, ensuring consistent and repeatable infrastructure deployments.
* Deployed the application on AWS, utilizing services such as EC2 for compute resources, S3 for storage, and possibly other AWS resources to support the utility's operations.
* Additionally, designed and implemented a hierarchical data model using Amazon Neptune, utilizing its graph database capabilities to efficiently store, query, and manage complex relationships within the dataset, resulting in improved data retrieval times and enhanced scalability.
* Evaluated performance metrics such as latency and throughput by sending extensive dummy data through the big data pipeline, providing critical insights for optimization and scalability.
* Used the Golang utility to test the big data pipeline performance for every sprint by sending thousands of lines of data from each virtual asset, validating data integrity and accuracy by checking if all data reached its intended destination using specific parameters.
* Collaborated closely with the big data team to test and deploy updates into production every two weeks, ensuring timely and efficient integration of performance improvements and bug fixes.

**Environment**: Golang, HTTP, gRPC, Protocol Buffers (Protobuf), gRPC client/server configurations, TLS, Kafka, Spark, Presto, HDFS, Nginx, Docker, AWS EC2, AWS S3, latency, throughput, performance metrics, logging, data validation, integration, deployment, AWS services, big data pipeline, concurrent processing, virtual endpoints, Amazon Neptune

**Client: Wings info PVT Ltd, India** **August 2015 to July 2019**

**Golang/Java Developer**

**Project-1 Description:**  
Transformed Wings Info Pvt Ltd's digital operations by developing cutting-edge Java solutions. Focused on building robust backend systems for seamless data management and processing, while also designing user-friendly interfaces to improve client engagement. Implemented advanced security measures to safeguard sensitive data and ensure compliance with industry standards. Leveraged cloud technologies to enhance scalability and resilience, optimizing the performance of the company’s global operations.

**Responsibilities:**

* Worked in an Agile environment, participating in requirement gathering, analysis, and design.
* Developed user interfaces using HTML, CSS, and JavaScript.
* Implemented the service layer using Spring's Inversion of Control (IOC) features for efficient bean management.
* Developed core Java concepts, including Collections, Multithreading, Data Structures, Synchronization, and Exception Handling.
* Applied Dependency Injection (DI) and IOC concepts for application development within the Spring framework.
* Designed and developed microservices business components using Spring Boot and built cloud-based microservices.
* Developed microservices and RESTful web services using JAX-WS.
* Utilized Apache Kafka for reliable, asynchronous messaging between multiple business applications.
* Created JUnit test cases for unit testing and used Log4J for runtime exception logging.
* Developed shell scripts for monitoring production applications and managing custom Java applications.
* Collaborated with business teams to model database schemas, and created tables, SQL queries, PL/SQL stored procedures, and database triggers.
* Used Terraform modules to create reusable infrastructure configurations, ensuring consistency, scalability, and maintainability.
* Employed Hibernate to create a persistence layer for database interaction, updating information in the Oracle database.

**Environment:** HTML, CSS, Bootstrap, XML, Java, JDBC, Servlets, JSP, SOAP, JavaScript, Hibernate, Git, Java Mail API, AJAX, ANT, Log4J, Windows, Linux, NetBeans, Apache Tomcat, MVC, Terraform.

**Project-2 Description:** Developed a comprehensive web UI project, integrating backend control plane functionalities using Golang and a dynamic frontend with React JS, Angular JS, TypeScript, JavaScript, HTML, and CSS. Actively collaborated with the middleware team to test and validate their changes, employing the tool to simulate the enrollment of 100K to 200K virtual assets. Implemented robust data management in PostgreSQL, where each asset was assigned a unique identifier and associated dummy data. Established real-time asset monitoring by integrating Prometheus for data collection and Grafana for visualizing asset trends, ensuring effective tracking of asset counts and performance metrics. This tool streamlined debugging and performance analysis for developers, bigdata and middleware teams.

**Responsibilities:**

* Involved in the design, implementation, and deployment of the Full Software Development Life Cycle (SDLC) using Agile methodology and Test-Driven Development (TDD) for application development, utilizing JIRA as a project management tool.
* Developed the backend control plane functionality using Golang for a Web UI project.
* Designed and implemented the frontend using React JS (version 16.8), Angular (version 8), TypeScript, JavaScript, HTML, and CSS.
* Experienced in working with Redux Architecture using complex Object-Oriented concepts in improving the performance of the websites. Experience in using React JS components, Forms, Events, Keys, Router, Redux.
* **Developed a versatile UI for the Golang utility using React and Angular, supporting multiple stacks and teams, and allowing users to select different simulators and configure options such as total number of assets, stack types, number of customers, assets per customer, number of simulators, and Node.js settings.**
* **Provided configuration options in the UI for customizing simulation parameters, enabling users to tailor their testing environments for diverse scenarios.**
* **Included a feature for reviewing previous sprint reports in the UI, allowing users to access historical data and analyze past performance metrics.**
* **Implemented a comparison tool using React and Angular to display differences between current and previous sprint reports, facilitating detailed analysis and performance tracking over time.**
* Developed REST APIs using Golang and for the operation on asset.
* Using the REST API, we implemented the operations like update, deleting, adding a new asset.
* Established connection between Web API and Angular by using HTTP Call and designed GET, POST, PUT, and DELETE for Web API.
* Developing RESTful APIs using JSON, performing code management using GIT, and resolving code conflicts and involved in Test Driven Development (TDD), paired programming, and refactoring code.
* Implemented Infrastructure as Code (IAC) practices with Terraform to streamline environment setup and management.
* Worked on MongoDB with AWS to store the reports data for every sprint.
* Collaborated with the middleware team to test and validate changes during each sprint.
* Simulated the enrollment of 100K to 200K virtual assets, ensuring proper asset management and enrollment verification.
* Assets data added to PostgreSQL whenever asset is enrolled with some dummy data and one unique identifier as asset id generated UUID.
* Monitored asset count trends by integrating data with Prometheus and visualizing metrics with Grafana.
* Contributed to optimizing the tool’s performance for handling and analyzing large volumes of assets and data.
* Generated the report in the final after like 5 to 10 hours about the asset trend and mention about asset drop at the times which was happened.
* Collaborated with middleware and big data teams to test the functionality of the Golang utility in every sprint, producing detailed reports on improvements or performance issues compared to previous reports.

**Environment**: Golang, React JS (version 16.8), Angular (version 8), TypeScript, JavaScript, HTML, CSS, REST APIs, HTTP, MongoDB, AWS, GIT, JSON, PostgreSQL, UUID, Prometheus, Grafana, middleware, asset management, code management, performance optimization, Agile, SDLC, Test-Driven Development (TDD), JIRA, Redux.

**Certifications:**

1. AWS Solutions Architect Associate (Expires on Jan 2027).
2. AWS Cloud Practitioner (Expires on March 2027).