Prajwal Koirala

Certified Multi-Cloud Architect | AWS, GCP & Azure | Software Engineer Specializing in Scalable Systems and Infrastructure Optimization

New York City in New York State | 8456820433 | prajwalkoirala638@gmail.com

Profiles

Prajwal-Koirala

Prajwal-Koirala

Summary

I am a Certified Multi-Cloud Architect with extensive expertise spanning AWS, Google Cloud, and Microsoft Azure. For over a decade, I have been at the forefront of designing, deploying, and managing scalable cloud infrastructures that power mission-critical systems in industries including finance, healthcare, e-commerce, and technology.

My career in software development and cloud engineering is driven by a passion for innovation and a commitment to excellence, transforming legacy systems into agile, modern environments that leverage automation and continuous integration.

I have led diverse teams in implementing robust Infrastructure as Code (IaC) practices using tools such as Terraform and CloudFormation to ensure that cloud deployments are repeatable, secure, and cost-effective. My work emphasizes both technical precision and strategic foresight, adhering to stringent security and compliance standards like GDPR, HIPAA, and CCPA while integrating CI/CD pipelines and DevOps methodologies to reduce deployment times and optimize performance. This balanced approach enables organizations to achieve operational resilience and sustainable growth.

My credentials include AWS Certified Solutions Architect – Professional, Google Cloud Professional Cloud Architect, Microsoft Azure Solutions Architect Expert, and HashiCorp Terraform Associate, reflecting a deep and proven ability to execute complex multi-cloud strategies. I actively contribute to open-source initiatives and mentor emerging professionals, fostering collaborative environments that drive continuous improvement and innovation.

By aligning cutting-edge technical solutions with strategic business objectives, I ensure that every project delivers tangible, long-term value. Dedicated to staying ahead of emerging trends, I consistently explore new technologies to solve intricate challenges and pioneer transformative change.

Whether modernizing existing systems or architecting new cloud-native solutions, I work closely with stakeholders to deliver outcomes that are secure, scalable, and forward-thinking. I welcome opportunities to collaborate on initiatives that redefine cloud architecture and drive digital transformation. My unwavering commitment to innovation, security, and efficiency is the cornerstone of every project I lead, ensuring that organizations remain agile and competitive in a rapidly evolving digital landscape.

Experience

IpEngine.xyz — **Apr 2023 - Mar 2025**

Senior Software Developer

https://ipengine.xyz/

IP Intelligence Platform Development

Outcome: Increased data accuracy by 20%.

Key Actions: I played a key role in developing the core features of the IP intelligence platform. These features focused on enabling real-time analysis of IP addresses, enhancing our data collection, and ensuring that the data collected was more accurate. This allowed us to generate real-time risk scores for IPs based on various factors, such as geographical location, usage patterns, and historical data.

Value Added: Improving data accuracy is critical for smarter decision-making. With better IP intelligence, businesses could make more informed decisions regarding user verification, fraud prevention, and targeted services. The platform's enhanced data integrity also streamlined planning processes, especially for clients relying on IP data for analysis or security purposes.

Engineering: To handle the real-time nature of IP risk scoring and classification, I built scalable data pipelines using Python for data processing and Go for performance-critical tasks. These pipelines were designed to ingest, analyze, and classify IP data in real-time, while ensuring high availability and low latency.

Backend System Optimization

Outcome: Achieved 99.9% uptime.

Key Actions: I focused on optimizing both database and API performance. This included improving database indexing to speed up query processing and refining API efficiency to handle high volumes of requests. I also implemented redundancy and failover mechanisms to ensure that the system remained operational even in case of a failure, minimizing downtime.

Value Added: Achieving 99.9% uptime significantly enhanced the reliability of our system. Customers could trust that the platform would be available when they needed it, reducing interruptions in their workflows. The optimization also lowered long-term maintenance costs by reducing the need for manual intervention and ensuring a more self-sustaining infrastructure.

Engineering: To optimize backend services, I refactored the core services using Golang for its performance advantages. I also enhanced PostgreSQL performance by adding efficient indexing strategies. For caching, I introduced Redis to reduce the load on the database and improve API response times, particularly for frequently accessed data.

Multi-Cloud Deployment Strategy

Outcome: Delivered scalable, high-availability systems.

Key Actions: I architected a multi-cloud deployment strategy using AWS and GCP. This allowed us to leverage the strengths of both cloud platforms while ensuring high availability and scalability. We used Docker containers to package our applications, and Kubernetes for orchestration, enabling us to run applications consistently across both clouds without worrying about vendor lock-in.

Value Added: The multi-cloud architecture provided us with the flexibility to choose the best services and features from AWS and GCP, while also ensuring that the system could scale easily to handle growing demand. Additionally, the high-availability setup meant that services were always up, even if one cloud provider experienced issues.

Engineering: I implemented a CI/CD pipeline using Terraform to manage infrastructure as code, Helm to deploy applications on Kubernetes, and GitHub Actions for automating the build and deployment processes. This setup allowed for seamless deployment and scaling across both cloud platforms, reducing manual work and improving overall deployment efficiency.

Legacy System Migration to Cloud-Native

Outcome: Reduced deployment times by 30% and lowered overhead.

Key Actions: As part of the migration process, I led the effort to move from a monolithic architecture to a microservices-based, cloud-native model. The transition involved breaking down legacy applications into smaller, manageable services that could be deployed independently. We also moved the applications to cloud-native technologies to take advantage of scalability, cost savings, and easier maintenance.

Value Added: By re-architecting the systems, we lowered operational costs and improved the maintainability of the infrastructure. The new microservices approach allowed us to innovate more rapidly and deploy changes faster, significantly reducing deployment times.

Engineering: I was responsible for re-architecting the monolithic applications into containerized microservices using Kubernetes for orchestration. We also used gRPC to facilitate communication between microservices, ensuring low-latency and efficient data transfer between services.

Mentorship & Team Development

Outcome: Elevated code quality and team collaboration.

Key Actions: I mentored junior developers, helping them grow their skills and understanding of best practices in coding, testing, and deployment. I led code reviews, providing constructive feedback to ensure that the code met quality standards and maintained consistency across the team. I also worked to establish a culture of continuous improvement by encouraging the adoption of new tools, frameworks, and practices.

Value Added: This initiative fostered a culture of knowledge-sharing and continuous improvement. The increased collaboration led to better code quality and more efficient workflows. The team's ability to innovate and solve problems also improved significantly, resulting in faster delivery of features and better overall product quality.

Engineering: I helped establish engineering guidelines that covered key areas such as CI/CD pipelines, automated testing, and infrastructure as code. This ensured that the team had a solid foundation for building, testing, and deploying software reliably and consistently.

System Monitoring & Incident Response

Outcome: Reduced downtime by 25%.

Key Actions: To improve system reliability, I implemented proactive monitoring systems that could detect issues before they caused significant downtime. I integrated tools like Prometheus and Grafana for real-time metrics monitoring and visualization, along with AWS CloudWatch for broader infrastructure monitoring. Additionally, I developed automated incident response mechanisms that triggered alerts and initiated self-healing actions to address issues without manual intervention.

Value Added: By reducing downtime by 25%, system availability and performance were improved, leading to better user experience and higher customer satisfaction. Proactive monitoring also allowed us to quickly detect and fix issues, minimizing disruption.

Engineering: I developed custom scripts using Terraform and Ansible to automate the monitoring and incident response processes. These scripts helped to automate the setup of monitoring dashboards and alerting systems, while also handling basic self-healing actions like restarting services or scaling infrastructure based on load.

Collaboration with Product & Design

Outcome: Delivered user-centric features that improved customer satisfaction.

Key Actions: I worked closely with product managers and design teams to ensure that new features were aligned with customer needs and business goals. This collaboration helped create user-centric features that enhanced the overall experience, ensuring that each feature added value to the users and supported the company's objectives.

Value Added: Cross-functional collaboration led to more thoughtful, intuitive features that better met customer needs. By integrating feedback from multiple teams, we were able to deliver a product that was both functional and user-friendly, which in turn improved customer satisfaction and retention.

Engineering: I implemented feature flags and A/B testing to ensure that new features were tested in real-world conditions before being fully deployed. We also used API versioning to maintain backward compatibility while iterating on features, allowing for a smoother rollout of changes without disrupting existing users.

Performance Optimization

Outcome: Increased system throughput by 30%.

Key Actions: To boost system performance, I focused on eliminating backend bottlenecks and optimizing both database queries and API calls. I implemented caching strategies using Redis to reduce load times for frequently accessed data and optimized database queries to speed up data retrieval.

Value Added: This resulted in a more responsive system, reducing latency and allowing the platform to handle more users and higher traffic volumes. The performance improvements also laid the foundation for future scalability, ensuring that the system could grow alongside demand.

Engineering: I used PostgreSQL's EXPLAIN ANALYZE command to identify slow queries and then optimized them for better performance. I also integrated Redis caching to reduce load on the database and speed up response times for commonly queried data, enabling the system to handle a higher throughput efficiently.

Cloud Native Computing Foundation — **Nov 2020 - Mar 2023**

Lead Cloud Engineer — New York City

https://www.cncf.io/

Multi-Cloud Architecture Design

Outcome: Optimized cost efficiency and scalability for clients.

Key Actions: I designed multi-cloud architectures leveraging AWS, GCP, and Azure. The architecture used AWS Lambda, Azure Functions, and Google Kubernetes Engine (GKE) to create highly available, fault-tolerant solutions. The use of these technologies allowed for seamless workloads across different cloud platforms, reducing vendor lock-in and providing clients with the flexibility to choose the best services based on cost, performance, and geographic proximity.

Value Added: By designing multi-cloud systems, clients were able to take advantage of the unique strengths of each cloud provider, optimize their cloud service usage, and lower costs. This setup improved agility, as clients could shift workloads between clouds depending on the requirements or changes in pricing, thus enabling better overall flexibility.

Engineering: I built cloud-agnostic systems using infrastructure automation and orchestration to ensure smooth operation across all three cloud environments. This approach minimized downtime during deployments, ensured optimal cloud resource use, and allowed the architecture to scale dynamically with demand.

Legacy System Migration to Cloud

Outcome: Reduced infrastructure costs by 30%.

Key Actions: I led the migration of legacy systems to cloud-native solutions by re-architecting them into microservices, leveraging containerization, and adopting serverless architectures. The migration process involved breaking down monolithic applications into smaller, independent services that could scale autonomously and be deployed more easily. This allowed for more efficient resource use and faster feature delivery.

Value Added: The migration reduced overhead costs associated with maintaining legacy systems and improved the scalability and flexibility of the infrastructure. With a cloud-native setup, the systems could quickly adapt to changes in demand, provide faster feature releases, and lower maintenance costs.

Engineering: I re-architected the existing monolithic systems into microservices using Docker for containerization and Kubernetes for orchestrating those containers. For serverless compute, I used AWS Lambda to enable on-demand processing, allowing us to scale with minimal infrastructure overhead while ensuring seamless operations.

Automated Deployment Pipelines

Outcome: Reduced deployment times by 40% and sped up release cycles.

Key Actions: I automated the CI/CD pipelines using tools like Terraform and AWS CloudFormation, creating an infrastructure as code (IaC) setup that allowed for rapid, consistent releases. These pipelines were designed with integrated testing and security checks to ensure code quality and compliance before deployment.

Value Added: By automating deployment processes, the development cycle was accelerated, allowing for more frequent releases and faster delivery of features. This also reduced manual intervention, increasing developer productivity and lowering the risk of human error in deployments.

Engineering: I developed fully automated pipelines that handled everything from infrastructure provisioning with Terraform to environment setup with CloudFormation. This ensured that deployments were not only faster but also more reliable and repeatable, improving overall workflow efficiency.

Cloud Cost Optimization

Outcome: Saved clients 25% on annual cloud costs.

Key Actions: I worked closely with clients to optimize their cloud usage by analyzing resource consumption with AWS Cost Explorer and Azure Cost Management. We implemented right-sizing strategies for instances, adjusted storage costs, and optimized resource allocation to ensure that clients only paid for what they actually needed. Additionally, I enabled auto-scaling to ensure that resources were allocated dynamically based on usage, reducing unnecessary costs.

Value Added: The cost optimization efforts saved clients a significant portion of their annual cloud expenses without sacrificing performance. By aligning cloud resources more closely with demand, clients achieved better budget efficiency and avoided over-provisioning.

Engineering: I implemented auto-scaling and continuous resource monitoring using cloud-native tools to ensure that the infrastructure adapted to changing demands. This was critical in maintaining optimal cloud costs while preserving the required performance levels for clients.

Disaster Recovery and Availability

Outcome: Achieved 99.9% uptime and improved continuity.

Key Actions: I designed and implemented highly available, fault-tolerant systems using multi-region and multi-AZ architectures, ensuring that the infrastructure could recover quickly from any disruptions. Additionally, I integrated cloud-native backup solutions and automated failover mechanisms to guarantee that the system remained operational, even during outages.

Value Added: The architecture improved the overall reliability and availability of services, reducing downtime and minimizing data loss in case of failure. This provided clients with peace of mind, knowing that their systems would remain up and running even during unforeseen incidents.

Engineering: I designed systems that leveraged multi-region deployments and distributed cloud resources to provide continuous availability. Automated failover and backup strategies were implemented using native cloud tools, allowing the systems to recover swiftly and ensure that clients' services were always available.

Containerization and Kubernetes Adoption

Outcome: Modernized deployments and improved resource management.

Key Actions: I championed the adoption of containerization and Kubernetes for managing microservices. The use of Docker allowed for consistent packaging of applications across different environments, and Kubernetes provided orchestration for scaling, managing, and maintaining containers efficiently. This allowed for faster and more reliable deployments and centralized management of resources.

Value Added: By containerizing applications and managing them through Kubernetes, we improved system reliability and reduced the complexity of deployment workflows. Kubernetes also allowed us to scale applications seamlessly, adjusting resources dynamically to meet demand and optimizing the use of infrastructure.

Engineering: I worked with Helm for Kubernetes to streamline deployment management, allowing us to define, install, and upgrade applications efficiently. Kubernetes provided the orchestration needed to manage containerized applications, while Docker ensured consistency across development, testing, and production environments, enabling smooth scaling and faster time to market for new features.

Nvidia — **Jul 2017 - Sep 2020**

Cloud Operations Specialist — New York City

https://www.nvidia.com

Cloud Operations for SaaS Applications

Outcome: 99.9% uptime.

Key Actions: Managed SaaS applications hosted on AWS and Azure using auto-scaling and load balancing strategies to handle traffic spikes. Auto-scaling dynamically adjusted resources based on real-time demand, ensuring that the system could handle peak loads without performance degradation. Implemented load balancing to distribute incoming traffic evenly across servers, maintaining high availability during periods of heavy traffic.

Value Added: These measures ensured continuous high availability, minimized disruptions, and optimized the user experience. By maintaining stable performance even during peak traffic, customer satisfaction and retention were significantly improved.

Engineering: Utilized AWS Auto Scaling, Azure Scale Sets, Elastic Load Balancing (ELB), and Azure Load Balancer to manage scaling and distribute traffic.

Disaster Recovery Solutions

Outcome: RTOs under 10 minutes.

Key Actions: Designed and implemented disaster recovery solutions that included cross-region replication, automated failover, and real-time backup strategies to reduce recovery time objectives (RTO). In the event of a failure, traffic was automatically rerouted to a secondary region, and data was restored from real-time backups.

Value Added: The ability to recover in under 10 minutes minimized downtime, ensuring continuity of services for clients. This improved system resilience and enabled seamless operation during emergencies.

Engineering: Used AWS Route 53 for DNS failover, AWS S3 for backup storage, Azure Site Recovery for failover, and Azure Backup for real-time data protection.

Automated Cloud Resource Provisioning

Outcome: Reduced manual workload by 40%.

Key Actions: Automated the provisioning of cloud resources using AWS CLI, PowerShell, and Google Cloud SDK. I created reusable scripts to automate the setup of resources like EC2 instances, VPCs, and storage, reducing human error and the time spent on manual configuration.

Value Added: This automation increased operational efficiency by reducing manual tasks, minimizing configuration errors, and allowing teams to focus on higher-value work.

Engineering: Created scripts to provision AWS EC2 instances, S3 buckets, and Google Cloud Compute Engine instances using the AWS CLI, PowerShell, and Google Cloud SDK.

Custom Monitoring Dashboards

Outcome: Real-time system insights.

Key Actions: Developed custom monitoring dashboards using CloudWatch, Azure Monitor, and Google Cloud Operations Suite to provide real-time insights into the health of applications and infrastructure. These dashboards monitored key metrics such as CPU usage, memory utilization, and network throughput.

Value Added: The custom dashboards allowed teams to detect performance degradation or system issues proactively, which helped with faster resolution and improved decision-making.

Engineering: Integrated CloudWatch for AWS, Azure Monitor for Azure, and Google Cloud Operations Suite to pull real-time performance metrics into custom dashboards.

Cloud Infrastructure Cost Optimization

Outcome: Reduced cloud spending by 20%.

Key Actions: Optimized cloud infrastructure costs by analyzing usage patterns, rightsizing instances, and implementing reserved instances. This included using auto-scaling policies to ensure that only the required resources were in use at any given time.

Value Added: The optimization efforts reduced unnecessary resource allocation, lowering cloud costs by 20%, while ensuring that the systems maintained the necessary performance and scalability.

Engineering: Leveraged AWS Cost Explorer, Azure Cost Management, and auto-scaling capabilities in AWS EC2 and Google Cloud Compute Engine to adjust instance sizes based on traffic and usage.

Automated Scaling Policies

Outcome: Efficient handling of fluctuating workloads.

Key Actions: Developed automated scaling policies for AWS EC2, Azure VMs, and Google Cloud Compute Engine instances. These policies dynamically scaled resources up or down based on traffic demand, ensuring cost-efficiency while maintaining optimal performance.

Value Added: The automated scaling policies helped reduce costs during low-traffic periods, while ensuring that performance remained optimal during traffic surges, leading to more efficient resource utilization.

Engineering: Implemented auto-scaling using AWS Auto Scaling, Azure VM Scale Sets, and Google Cloud Instance Groups.

Security Best Practices Implementation

Outcome: Enhanced security and reduced vulnerabilities.

Key Actions: Implemented multi-factor authentication (MFA), role-based access control (RBAC), and encryption for data in transit and at rest. These security measures ensured that sensitive data was protected and access to cloud resources was securely managed.

Value Added: By improving security practices, the risk of breaches was minimized, customer trust was increased, and compliance with regulations like GDPR and CCPA was maintained.

Engineering: Configured AWS IAM roles and policies, Azure RBAC, and used AWS KMS and Azure Key Vault for data encryption, and enforced MFA for critical resources.

CI/CD Pipeline Management

Outcome: Reduced deployment time by 30%.

Key Actions: Managed and streamlined CI/CD pipelines using Jenkins, Azure DevOps, and GitLab CI, automating testing, integration, and deployment processes. The automation included setting up validation checks to ensure consistent and secure deployments across multiple environments.

Value Added: Reduced deployment times by automating manual steps, resulting in faster release cycles and higher-quality code delivered to production with fewer errors.

Engineering: Integrated Jenkins for automation, Azure DevOps for build and release pipelines, and GitLab CI for source code management and CI pipeline automation.

Cloud-Native Monitoring Solutions

Outcome: Improved response times.

Key Actions: Deployed cloud-native monitoring solutions that integrated proactive alerting mechanisms. These alerts notified teams of potential disruptions, enabling a quick response to resolve issues before they escalated.

Value Added: Real-time alerting significantly reduced response times and minimized downtime by resolving issues before they affected end-users.

Engineering: Used AWS CloudWatch, Azure Monitor, and Google Cloud Operations Suite to implement monitoring and alerting systems for proactive issue resolution.

Complexorganizations — **Apr 2012 - Jun 2017**

Software Developer — New York City

https://www.complexorganizations.com

Infrastructure-as-Code Solutions Design

Outcome: Reduced manual intervention by 60% and lowered infrastructure costs by 30%.

Key Actions: Designed and implemented Infrastructure-as-Code (IaC) solutions using Terraform for automated deployment of resources across AWS and Azure. Automated infrastructure provisioning for VPCs, EC2 instances, S3 buckets, and RDS databases on AWS, and Azure Virtual Networks, Virtual Machines, and Blob Storage on Azure. Led cloud migration efforts, standardizing configurations across environments with Terraform modules and enforced security best practices by configuring IAM roles, security groups, and encryption policies.

Value Added: Eliminated repetitive manual tasks, reduced errors, and provided a scalable, cost-efficient infrastructure with enhanced reliability, allowing for faster provisioning and more consistent environments.

Engineering: Created reusable Terraform modules to provision and manage cloud resources consistently across multiple environments. Automated security measures using AWS IAM, Azure RBAC, and encrypted storage solutions like AWS KMS and Azure Key Vault for secure key management.

Team Leadership and Project Delivery

Outcome: Delivered 10+ successful projects on schedule, improving productivity and product quality.

Key Actions: Led a team of 10 developers, collaborating with product owners to define technical requirements and project scopes for deploying cloud infrastructure. Coordinated efforts on containerization, CI/CD pipeline setup, and automation projects. Mentored junior engineers on best practices in cloud architecture, IaC, and containerization technologies like Docker and Kubernetes.

Value Added: Enhanced team collaboration, reduced technical debt, and ensured timely delivery of high-quality products. Fostered a culture of continuous improvement and skill development within the engineering team.

Engineering: Managed project timelines, assisted with architectural design decisions, and ensured high-quality deliverables. Conducted code reviews and implemented best practices for scalable, maintainable cloud infrastructure.

CI/CD Pipeline Implementation

Outcome: Increased deployment frequency by 50% and improved system stability.

Key Actions: Implemented CI/CD pipelines using Jenkins, GitLab CI, and Azure DevOps for automating the build, test, and deployment processes. Integrated unit tests, integration tests, and security checks into pipelines. Deployed applications to AWS Elastic Beanstalk, Azure App Service, and Kubernetes clusters on AWS EKS and Azure AKS. Automated rollback mechanisms using Helm for Kubernetes and Blue/Green deployments for zero-downtime releases.

Value Added: Accelerated software delivery, improved code reliability, minimized manual intervention, and ensured security compliance through automated testing and vulnerability scans.

Engineering: Developed automation scripts for the CI/CD pipeline, integrated Docker containers, and utilized Helm for Kubernetes deployment management. Incorporated testing tools like SonarQube and security scanning tools like OWASP Dependency-Check.

Containerization and Orchestration Adoption

Outcome: Reduced provisioning time by 70% and improved uptime.

Key Actions: Containerized applications using Docker and deployed them on Kubernetes clusters managed by AWS EKS and Azure AKS. Utilized Helm charts for managing application deployment, enabling seamless scaling and versioning of containers. Implemented Horizontal Pod Autoscaling and Cluster Autoscaling in Kubernetes to automatically adjust the cluster size based on load. Integrated Prometheus and Grafana for real-time monitoring and alerting.

Value Added: Simplified application deployments, reduced provisioning complexity, and improved resource utilization, ensuring applications were highly available and resilient to failures.

Engineering: Implemented Docker containers for packaging applications and Kubernetes for orchestration. Configured Kubernetes HPA (Horizontal Pod Autoscaler) and cluster-level auto-scaling, ensuring that applications could scale dynamically based on demand.

Security and Compliance Integration

Outcome: Achieved SOC 2 and GDPR compliance, strengthening security posture.

Key Actions: Integrated Multi-Factor Authentication (MFA) using AWS MFA and Azure MFA, implemented Role-Based Access Control (RBAC) for access management using AWS IAM and Azure RBAC, and encrypted sensitive data in transit and at rest using TLS for data in transit and AWS KMS and Azure Key Vault for data encryption. Conducted regular security audits using AWS Inspector and Azure Security Center, ensuring compliance with SOC 2 and GDPR regulations.

Value Added: Ensured secure access, reduced vulnerabilities, improved risk mitigation, and built customer trust through robust data protection and compliance efforts.

Engineering: Implemented MFA and RBAC policies, managed encryption keys with AWS KMS and Azure Key Vault, and conducted vulnerability assessments using AWS Inspector and Azure Security Center to identify and resolve security risks.

Education

State University of New York College at Potsdam — Project-Based Learning

Cloud Computing

https://www.potsdam.edu/

- Computer Science Club: Participated in workshops, tech talks, and group projects.

- ACM (Association for Computing Machinery): Joined coding competitions and hackathons.

- Cybersecurity Club: Explored ethical hacking and participated in security challenges.

- Tutoring/Mentoring: Helped peers with programming and coursework.

- Hackathons: Competed in local and regional events to solve real-world problems.

- Volunteer Tech Support: Provided IT assistance for campus events or non-profits.

From August 2015 to May 2019, I earned a Bachelor of Science in Computer Science at SUNY Potsdam. I studied topics like software development, algorithms, databases, and AI, gaining hands-on experience through projects such as web applications and coding competitions. The program helped me build strong technical skills and problem-solving abilities, preparing me for challenges in the tech industry.

Projects

wireguard-manager

WireGuard-Manager is a tool designed to simplify the deployment and management of WireGuard VPNs. It emphasizes user-friendliness and security, aiming to streamline VPN configuration for both personal and professional use.

https://github.com/complexorganizations/wireguard-manager

WireGuard-Manager offers a user-friendly interface and tools to manage VPN configurations efficiently, making it an ideal choice for individuals and organizations looking to enhance their VPN experience.

network-database

Network-Database is a comprehensive repository of IP addresses, meticulously categorized to support various network management and security needs. It serves as a pivotal resource, offering a dynamic and regularly updated collection of millions of IP addresses, each associated with specific types of network activity or threats.

https://github.com/complexorganizations/network-database

Network-Database provides an extensive collection of over 10 million IP addresses, organized into categories that represent different network activities or threats. This organization aids in identifying and mitigating potential security risks, making it an invaluable tool for network administrators and security professionals.

proxy-registry

Proxy-Registry is a robust service that provides up-to-date listings of open and public proxies. It is designed to cater to a broad range of users, from individuals seeking anonymous browsing to businesses requiring reliable proxy solutions. The service automates the collection of proxies, validates them for speed, uptime, and anonymity, and refreshes the proxy list every 24 hours to ensure access to the latest and most reliable proxies.

https://github.com/complexorganizations/proxy-registry

Proxy-Registry offers a comprehensive and efficient solution for obtaining and managing open and public proxies. Its automated proxy scraping, validation processes, and regular updates make it a valuable resource for users seeking reliable proxy services.

content-blocker

Content Blocker is a tool designed to enhance your online privacy and security by blocking ads, trackers, malware, and unwanted content at the DNS level. By filtering out intrusive elements, it allows you to browse the web with greater peace of mind.

https://github.com/complexorganizations/content-blocker

Content Blocker empowers users to reclaim their online experience by effectively blocking various types of unwanted content, thereby improving privacy and security during web browsing.

disposable-services-registry

The Disposable Services Registry is a comprehensive and continually updated list of disposable domains used by companies worldwide to combat fraudulent activities. These domains are often employed for short-term purposes, such as email verification or temporary access, and are commonly associated with spam and fraud. By maintaining an extensive database of known disposable domains, the registry assists organizations in verifying user data, preventing spam, and enhancing online security.

https://github.com/complexorganizations/disposable-services-registry

The Disposable Services Registry offers an extensive and regularly updated collection of disposable domains, enabling organizations to identify and block domains frequently used in fraudulent activities. This resource aids in data validation, fraud prevention, and overall enhancement of online security.

Skills

Kubernetes

Kubernetes (K8s) is an open-source platform for managing and deploying containerized applications. It automates tasks like scaling, deployment, and monitoring.

Kubernetes, Docker

AWS

Amazon Web Services, Inc. is a subsidiary of Amazon that provides on-demand cloud computing platforms and APIs to individuals, companies, and governments, on a metered, pay-as-you-go basis. Clients will often use this in combination with autoscaling.

AWS, Cloud

Python

Python is a high-level, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation. Python is dynamically type-checked and garbage-collected.

Python

Go

Go is a high-level general purpose programming language that is statically typed and compiled. It is known for the simplicity of its syntax and the efficiency of development that it enables by the inclusion of a large standard library supplying many needs for common projects.

Go, Golang

Certifications

Angular (Basic) Certificate — **Feb 2025**

HackerRank

https://www.hackerrank.com/certificates/0291ba980735

Angular is a TypeScript-based free and open-source single-page web application framework. It is developed by Google and by a community of individuals and corporations. Angular is a complete rewrite from the same team that built AngularJS.

C# (Basic) Certificate — **Feb 2025**

HackerRank

https://www.hackerrank.com/certificates/fa740ca26405

C# is a general-purpose high-level programming language supporting multiple paradigms. C# encompasses static typing, strong typing, lexically scoped, imperative, declarative, functional, generic, object-oriented, and component-oriented programming disciplines.

CSS (Basic) Certificate — **Feb 2025**

HackerRank

https://www.hackerrank.com/certificates/e80a5cd09c3a

Cascading Style Sheets is a style sheet language used for specifying the presentation and styling of a document written in a markup language such as HTML or XML. CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript.

Frontend Developer (React) Certificate — **Feb 2025**

HackerRank

https://www.hackerrank.com/certificates/80784aa4c1d2

React is a free and open-source front-end JavaScript library that aims to make building user interfaces based on components more "seamless". It is maintained by Meta and a community of individual developers and companies.

Go (Basic) Certificate — **Feb 2025**

HackerRank

https://www.hackerrank.com/certificates/a14728e1a83b

Go is a high-level general purpose programming language that is statically typed and compiled. It is known for the simplicity of its syntax and the efficiency of development that it enables by the inclusion of a large standard library supplying many needs for common projects.

Java (Basic) Certificate — **Feb 2025**

HackerRank

https://www.hackerrank.com/certificates/dfb08375ebd0

Java is a high-level, class-based, object-oriented programming language that is designed to have as few implementation dependencies as possible.

JavaScript (Basic) Certificate — **Feb 2025**

HackerRank

https://www.hackerrank.com/certificates/d2d64add8276

JavaScript (JS) is a dynamic, weakly typed programming language primarily used to add interactivity to websites. It is one of the core technologies of the World Wide Web, along with HTML and CSS. JavaScript enables dynamic updates, multimedia control, animation, and more on web pages. It is also used in non-browser environments like Node.js.

Node (Basic) Certificate — **Feb 2025**

HackerRank

https://www.hackerrank.com/certificates/b5a761746b2d

Node.js is a cross-platform, open-source JavaScript runtime environment that can run on Windows, Linux, Unix, macOS, and more. Node.js runs on the V8 JavaScript engine, and executes JavaScript code outside a web browser. Node.js lets developers use JavaScript to write command line tools and for server-side scripting.

Python (Basic) Certificate — **Feb 2025**

HackerRank

https://www.hackerrank.com/certificates/c4b6369bbdea

Python is a high-level, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation. Python is dynamically type-checked and garbage-collected.

R (Basic) Certificate — **Feb 2025**

HackerRank

https://www.hackerrank.com/certificates/8326e3bbfbbb

R is a programming language for statistical computing and data visualization. It has been adopted in the fields of data mining, bioinformatics and data analysis. The core R language is augmented by a large number of extension packages, containing reusable code, documentation, and sample data.

React (Basic) Certificate — **Feb 2025**

HackerRank

https://www.hackerrank.com/certificates/95c9bfbeef59

HackerRank is a technology company that focuses on competitive programming challenges for both consumers and businesses. Developers compete by writing programs according to provided specifications.

Rest API (Intermediate) Certificate — **Feb 2025**

HackerRank

https://www.hackerrank.com/certificates/810995d96b08

A REST API, or Representational State Transfer API, is a way for applications to communicate with each other over the internet. REST APIs are based on the REST architectural style, which is a set of guidelines for building web APIs.

SQL (Basic) Certificate — **Feb 2025**

HackerRank

https://www.hackerrank.com/certificates/99392019bdf2

Structured Query Language is a domain-specific language used to manage data, especially in a relational database management system. It is particularly useful in handling structured data, i.e., data incorporating relations among entities and variables.

SQL (Intermediate) Certificate — **Feb 2025**

HackerRank

https://www.hackerrank.com/certificates/50ca77b2f0aa

Structured Query Language is a domain-specific language used to manage data, especially in a relational database management system. It is particularly useful in handling structured data, i.e., data incorporating relations among entities and variables.

Software Engineer Certificate — **Feb 2025**

HackerRank

https://www.hackerrank.com/certificates/ce64184de52a

Software engineering is **a field within computer science and a branch of engineering focused on designing, developing, testing, and maintaining of software applications**. It involves applying engineering principles and computer programming expertise to develop software systems that meet user needs.

Software Engineer Intern Certificate — **Feb 2025**

HackerRank

https://www.hackerrank.com/certificates/99ac01955af1

Software engineering is **a field within computer science and a branch of engineering focused on designing, developing, testing, and maintaining of software applications**. It involves applying engineering principles and computer programming expertise to develop software systems that meet user needs.

SQL (Advanced) Certificate — **Feb 2025**

HackerRank

https://www.hackerrank.com/certificates/9b6630c5c25f

Structured Query Language is a domain-specific language used to manage data, especially in a relational database management system. It is particularly useful in handling structured data, i.e., data incorporating relations among entities and variables.

AWS Certified Developer – Associate — **Feb 2025**

Amazon Web Services Training and Certification

https://www.credly.com/badges/124f471e-06ae-4243-af1b-a23bb35a5eca

Earners of this certification have a comprehensive understanding of application life-cycle management. They demonstrated proficiency in writing applications with AWS service APIs, AWS CLI, and SDKs; using containers; and deploying with a CI/CD pipeline. Badge owners are able to develop, deploy, and debug cloud-based applications that follow AWS best practices.

AWS Certified Cloud Practitioner — **Feb 2025**

Amazon Web Services Training and Certification

https://www.credly.com/badges/584f1db1-544a-41bb-a3ab-447adaad06f3

Earners of this certification have a fundamental understanding of IT services and their uses in the AWS Cloud. They demonstrated cloud fluency and foundational AWS knowledge. Badge owners are able to identify essential AWS services necessary to set up AWS-focused projects.

HashiCorp Certified: Terraform Associate (002) — **Feb 2025**

HashiCorp

https://www.credly.com/badges/be45aae8-0c0e-4d29-acde-ed8ca302969b

Earners of the HashiCorp Certified: Terraform Associate certification know the basic concepts, skills, and use cases associated with open source HashiCorp Terraform. They understand and can utilize Terraform according to the certification objectives. Additionally, they understand why enterprises choose to extend Terraform Community with Terraform Enterprise to solve business critical objectives.

Languages

English