MASTER PROFESSIONAL ENGINEERING PORTFOLIO

A Complete Record of Engineering Experience, Technical Capability, and Professional Practice

DHEERAJ SINGH RAO

DevOps Engineer | Backend & Cloud Infrastructure | RHCSA Certified

+91-9001913550 | raodheeraj604@gmail.com | LinkedIn | GitHub | Bitbucket | Medium

Table of Contents

Table of Contents 2
Executive Summary 4
Chapter 1 — About Me 5
Professional Introduction 5
Engineering Philosophy 5
Career Vision 5
Professional Values 6
Why Linux, Automation, Backend, Cloud, and Production Systems 6
Chapter 2 — Career Journey 6
Grras Solutions: Building the Foundation 6
Emizen Tech: Learning to Own Production 7
Synoption: Discipline and Release Engineering 7
Freelancing and International Consulting: Combining Everything 7
What This Progression Taught Me 8
Chapter 3 — Technical Skills 8
Linux and Systems Administration 8
Docker, Docker Compose, and Kubernetes 8
CI/CD and Version Control 8
Cloud Platforms (Overview) 9
Web Servers, Reverse Proxies, and Networking 9
Databases and Release Management 9
Monitoring, Observability, and Incident Response 10
Backend Development: Python, FastAPI, and REST APIs 10
Scripting and Automation: Bash, PowerShell, and YAML 10
Collaboration and Productivity Tools 10
Chapter 4 — Cloud Experience 11
Amazon Web Services (AWS) 11
Microsoft Azure 11
Google Cloud Platform (GCP) 11
Oracle Cloud Infrastructure (OCI) 11
DigitalOcean and Nexcess 11
What Multi-Cloud Experience Has Taught Me 11
Chapter 5 — Work Experience 12
Grras Solutions Pvt. Ltd. — DevOps Engineer Trainee (October 2022 – July 2023) 12
Emizen Tech Pvt. Ltd. — DevOps Engineer (November 2023 – December 2024) 12
Synoption Pvt. Ltd. — DevOps Engineer (May 2025 – September 2025) 13

Freelance Consulting — DevOps and Backend Engineer (October 2025 – Present) 14
Chapter 6 — Sentronic GmbH — International Client Engagement 14
Project One: The Report Management System — Business Problem and Architecture 14
Report Management System — Live View, Trend View, and History View 14
Report Management System — Export Formats and Business Logic 15
Report Management System — Production Issues and Architecture Improvements 15
Report Management System — Lessons Learned and Future Scope 15
Project Two: The Docker-Based Automation and Reverse Proxy Platform 15
Docker Automation Platform — SSL, Access Control, and Environment Validation 16
What the Sentronic GmbH Engagement Represents 16
Chapter 7 — Major Projects — Case Studies 16
Real-Time Chat Application — CI/CD Deployment on Oracle Cloud 16
Android Application CI/CD Auto-Deployment 17
Auto-Deployment of the Shopware E-Commerce Platform 17
CI/CD Pipelines for Laravel, Magento2, and Node.js Applications 18
Deploying a Web Application with Nginx and Reverse Proxy 18
Chapter 8 — AI Tools and AI-Assisted Engineering 19
How I Use AI Tools 19
Chapter 9 — Certifications 19
Red Hat Certified System Administrator (RHCSA) 19
Chapter 10 — What I Can Do 20
Capabilities 20
Chapter 11 — Why Work With Me 20
Reliability and Ownership 21
Automation as a Mindset, Not a Checkbox 21
Evidence-Based Problem Solving 21
Documentation as Part of the Deliverable 21
Continuous Learning Without Chasing Trends 21
Communication Across Distance and Difference 21
Backend and Infrastructure, Together 21
Chapter 12 — Contact 21
Get in Touch 21

(If the table of contents above appears empty, right-click it and choose "Update Field," then select "Update entire table.")

Executive Summary

This document is the complete professional record of my career as a DevOps Engineer with backend engineering capability. It exists as a single reference for recruiters, hiring managers, CTOs, founders, and consulting clients who want more context than a resume can hold, and more structure than a portfolio website usually provides.

Over the past two years, I've worked across four organizations — Grras Solutions, Emizen Tech, Synoption, and, most recently, independent freelance consulting for international clients including Sentronic GmbH in Germany — building CI/CD pipelines, administering Linux infrastructure, deploying and monitoring production systems, and writing backend services in Python and FastAPI. I hold the Red Hat Certified System Administrator (RHCSA) certification and have worked hands-on across AWS, Azure, GCP, and Oracle Cloud Infrastructure.

This portfolio is organized into twelve chapters. It opens with who I am and how I think about engineering, walks through my career progression company by company, explains every significant technology I've used and how I've actually used it, dedicates a full chapter to my work with Sentronic GmbH, documents five independent projects as case studies, and closes with a direct explanation of the value I bring to a team or client. Nothing in this document is invented — every project, technology, and responsibility described here is drawn directly from my resume, prior cover letter, and professional introduction letter, cross-referenced and consolidated into one accurate account.

If you read only one section, I'd point you to Chapter 6, on my work with Sentronic GmbH — it's the clearest example of how I combine backend development, infrastructure automation, and production ownership in a single engagement.

Chapter 1 — About Me

Professional Introduction

I'm Dheeraj Singh Rao, a DevOps Engineer based in Jaipur, India, with just over two years of professional experience and a growing focus on backend engineering. My work has taken me from a structured trainee program through two full-time DevOps roles to, most recently, independent consulting for international clients. Across all of it, the common thread has been the same: I'm the person who makes sure code gets from a developer's machine into production, and stays running once it's there.

I'm RHCSA-certified, which reflects a Linux foundation I built early and have relied on ever since. I've built and maintained CI/CD pipelines using Jenkins, GitHub Actions, and Bitbucket Pipelines; deployed and managed applications on AWS, Azure, GCP, and Oracle Cloud Infrastructure; worked extensively with Docker and Docker Compose; and, over the last year, added backend development in Python and FastAPI to that skill set. That combination — infrastructure plus backend — is deliberate. I've found that understanding both sides of a deployment makes me faster and more useful than specializing in only one.

This document exists to give a complete, honest account of that experience: what I've built, how I think about engineering problems, and what I'd bring to a team or client that hires me.

Engineering Philosophy

I got into DevOps because I like systems that behave predictably under pressure, and I like the discipline required to make sure they do. That discipline is really the whole job: a CI/CD pipeline is only useful if people trust it enough not to double-check it manually, and a production system is only reliable if someone has thought carefully about what happens when a dependency fails, a disk fills up, or a deployment goes out at the wrong time.

Linux is where that discipline is most visible to me. I was drawn to it early because it rewards understanding over memorization — once you know how a service actually starts, where it logs, and how it fails, you can reason about almost any system built on top of it, even one you've never seen before. That kind of first-principles troubleshooting has been useful in nearly every role I've had, often more than familiarity with any specific tool.

Automation matters to me for a related reason. I don't automate because it's fashionable; I automate because manual steps are where mistakes and inconsistency creep in, especially under time pressure or when the same task needs to be repeated by different people. A well-designed pipeline removes that risk entirely for the cases it covers.

Production is where all of this gets tested for real. A deployment that works in staging but fails in production isn't a success, and I've learned to treat “it works on my machine” as the start of a question, not the end of one. Backend development complements this rather than competing with it — understanding how an API is actually built makes me a better DevOps engineer, and understanding how it gets deployed and monitored makes me a better backend engineer. I try to make the smallest reliable change that removes friction, and I revisit earlier decisions when they stop making sense rather than defending them out of habit.

Career Vision

My near-term goal is straightforward: to keep working at the intersection of backend development and infrastructure, on systems that are used in production rather than in a demo. I'm particularly drawn to roles and engagements where I can own a problem end-to-end — from writing the service, to defining how it deploys, to being the person who gets called if it breaks — because that ownership is where I learn fastest and do my best work.

Longer term, I want to keep deepening the backend side of my skill set without giving up the infrastructure fluency that got me here. Most engineers specialize in one or the other; I think the most useful people on a team are often the ones who can move between both, translating between “how should this be built” and “how will this actually run.” I'd rather be that kind of engineer than an expert in a single narrow tool, and every role I've taken since Grras Solutions has been a step in that direction.

Professional Values

A few things guide how I work, regardless of who I'm working for. Reliability comes first: I'd rather ship something slightly later and have it hold up in production than ship on time and spend the following week firefighting. Ownership follows from that — if something I built breaks, I want to be the one explaining why, not the one waiting to be told.

I default to documenting decisions and procedures in writing, because conversations get forgotten and documentation doesn't. That habit has mattered more with international clients, where time zones mean a question asked at the end of my day might not get answered until the next one — clear documentation closes that gap.

I also try to stay honest about uncertainty. If I don't know why something is failing, I say so and start investigating rather than guessing out loud. And I hold onto the belief that automation should serve people, not replace their judgment — the goal of removing a manual step is to free up attention for harder problems, not to remove the need for a person to understand the system.

Why Linux, Automation, Backend, Cloud, and Production Systems

Linux interests me because it's honest — the behavior of a system is knowable if you're willing to read logs, check configurations, and test hypotheses, rather than guess. That transparency is why I built my early career around it and pursued RHCSA certification rather than treating Linux as a black box other tools sit on top of.

Automation matters because engineering time is finite and repetitive manual work is where errors concentrate. Every pipeline I've built started from watching someone (often me) do the same deployment steps by hand more than once and deciding that the third time should be scripted.

Cloud infrastructure interests me because it turns infrastructure into something you can reason about and reproduce, rather than a physical machine that only one person understands. Working across AWS, Azure, GCP, and Oracle Cloud Infrastructure has shown me that the underlying engineering problems — networking, access control, storage, scaling — are similar everywhere, even when the console looks different.

Production systems matter to me more than any other part of the job, because production is where theory meets reality. A well-architected system that hasn't been tested against real traffic, real failures, and real users is still a hypothesis. I like being the person responsible for turning that hypothesis into something dependable.

Backend engineering is the newest addition to that list, and it's become important to me because it closes a loop. Building the Report Management System backend for Sentronic GmbH in Python and FastAPI showed me that I understand deployment and infrastructure differently when I've also written the service being deployed. I don't see backend development and DevOps as separate disciplines anymore; I see them as two views of the same problem.

Chapter 2 — Career Journey

My career so far has been a fairly linear progression: a structured training program, two full-time DevOps roles of increasing scope, and now independent consulting where I combine everything I've learned with backend development. Each stage added something the previous one didn't have, and none of it happened in isolation — the habits I built at Grras Solutions are still the ones I rely on today. This chapter walks through that progression in order; Chapter 5 provides the full technical detail for each role.

Grras Solutions: Building the Foundation

My first real exposure to DevOps came as a trainee at Grras Solutions in Jaipur, from October 2022 to July 2023. This was a structured learning environment rather than a production role, and that structure mattered — it gave me guided, hands-on repetition with AWS core services like EC2, VPC, and ELB, Docker, and Kubernetes fundamentals, at a pace where I could actually absorb why things worked, not just how to run the commands.

I configured Apache and Nginx web servers with reverse proxy setups, built CI/CD pipelines integrating Jenkins with GitHub, and got early exposure to monitoring through Prometheus and Grafana. I also supported deployment and backup

operations for PostgreSQL, MariaDB, and MySQL, which was my first real introduction to the idea that a database is not something you deploy once and forget — it needs an ongoing plan for backup, recovery, and safe schema changes.

Grras Solutions gave me the foundation everything else was built on. Without that guided repetition early on, I don't think I would have been ready to take on production ownership as quickly as I did in my next role.

Emizen Tech: Learning to Own Production

At Emizen Tech, from November 2023 to December 2024, the training wheels came off. I was responsible for CI/CD pipelines and production deployments across a genuinely wide range of client stacks — Laravel, Magento, Magento2, Shopware, and Node.js applications, plus Android app releases through Fastlane and Google Play Console. That breadth was demanding in a specific way: I couldn't specialize in one stack and coast, because the next client project might use a completely different one.

This role is where I built real fluency with Bitbucket, GitHub, and Jenkins as CI/CD tools, containerized applications with Docker and Docker Compose, and worked with Kubernetes and Vercel for deployment. I managed LEMP and LAMP server stacks, configured domains, SSL certificates, and firewall settings, and worked across AWS, Azure, and GCP depending on the client. I also set up monitoring with Zabbix, UptimeRobot, and New Relic, and took on backup and recovery responsibilities for application and database environments.

The biggest lesson from Emizen Tech was operational: when you support multiple client stacks simultaneously, documentation and consistent process matter more than expertise in any single technology, because you're constantly context-switching. I got comfortable reading unfamiliar codebases quickly and figuring out how a deployment was supposed to work before I could safely change it — a skill that's proven useful in every engagement since.

Synoption: Discipline and Release Engineering

At Synoption, working remotely from May to September 2025, my focus narrowed and deepened. Rather than supporting a wide range of client stacks, I worked within a single, more process-driven environment centered on Jenkins administration, database release management, and Jira-based workflows.

I managed Jenkins pipelines across multiple environments, automated repetitive operational tasks, and handled production deployments with a direct focus on reliability and minimal downtime. I managed MySQL and PostgreSQL database releases using Liquibase, which meant treating schema changes with the same rigor as application code — version-controlled, reviewed, and reversible. I enhanced Jenkins pipelines with automated email alerts for build and deployment failures, and generated daily operational reports so that issues surfaced early rather than being discovered after the fact.

Synoption also deepened my collaboration habits. I worked extensively with Jira for issue tracking, triggered deployment pipelines directly from tickets, and maintained Confluence documentation for release procedures. This was the role where I learned that release engineering is as much about communication and process discipline as it is about the pipeline itself — a well-run release process makes the technical work almost boring, in a good way.

Freelancing and International Consulting: Combining Everything

Since October 2025, I've worked independently as a freelance DevOps and backend engineer, taking on infrastructure consulting and backend development work for international clients. This transition was less a career pivot than a natural next step: freelancing let me combine CI/CD automation, cloud infrastructure, and the backend development skills I'd been building, in engagements where I own the outcome end-to-end rather than one part of it.

Working independently for international clients has meant adapting to asynchronous communication, time zone differences, and the need to document decisions clearly enough that a client on another continent can follow my reasoning without a live conversation. It has also meant being more deliberate about scoping work and setting expectations, since there's no larger team absorbing ambiguity on my behalf.

The clearest example of what this stage of my career looks like is my ongoing engagement with Sentronic GmbH, a German industrial technology company, which is significant enough that it has its own chapter later in this document.

What This Progression Taught Me

Looking back, the progression from Grras Solutions to Emizen Tech to Synoption to freelance consulting followed a clear pattern: learn fundamentals under guidance, take on broad hands-on ownership, develop process discipline in a more structured environment, then apply all three independently. Each stage exposed a gap the previous one hadn't — Grras Solutions gave me technical grounding but not production stakes; Emizen Tech gave me production stakes but limited process; Synoption gave me process discipline within a narrower scope. Freelancing is where those pieces came together, and it's also where I've grown the most as a backend developer, because client work like the Sentronic GmbH engagement required it.

Chapter 3 — Technical Skills

I've used a wide range of technologies across my roles, and rather than list them, this chapter explains what each one is, where and how I've actually used it, and what I've learned from that use. I've grouped related technologies together for readability.

Linux and Systems Administration

Linux is the base layer under nearly everything I do, and it's also where my RHCSA certification is grounded. In practice, this means user and process management, storage configuration, service management, networking, and file permissions — the fundamentals that everything else in this chapter depends on. I've administered Linux servers at every company I've worked for, from the trainee environments at Grras Solutions through client-facing production servers at Emizen Tech and Synoption.

The most important lesson Linux has taught me is diagnostic patience: rather than restarting a service and hoping, I check logs and application output, verify configuration files, and confirm resource usage (disk, memory, CPU) before making a change. That habit has saved me from masking real problems with quick fixes more than once. I've also used Linux as the base for LEMP and LAMP stack deployments, configuring Apache and Nginx, managing firewall rules, and handling SSL certificates directly on the server rather than relying entirely on managed services.

Docker, Docker Compose, and Kubernetes

Docker has been part of my toolkit since Grras Solutions and has only grown in importance since. I use it to package applications and their dependencies into reproducible containers, which removes an entire category of “it works on my machine” problems. At Emizen Tech, I containerized client applications across different stacks; in my freelance work, I designed a Docker-based automation platform (detailed in Chapter 6) that provisions containers dynamically alongside a pre-configured Nginx reverse proxy.

Docker Compose is how I manage multi-container setups — defining services, networks, and volumes declaratively so an environment can be recreated identically on a different machine or server. I've used it for both client deployments and personal projects, including a real-time chat application deployed on Oracle Cloud Infrastructure.

My Kubernetes experience is at the level of core concepts and applied environment setup — deployments, services, and basic orchestration concepts — built through hands-on exposure at Grras Solutions and infrastructure work at Emizen Tech, rather than deep production-scale cluster administration. I'm upfront about that distinction: I understand how Kubernetes fits into a deployment architecture and can work within an existing cluster, but I wouldn't yet call myself an expert in operating one from scratch at scale.

CI/CD and Version Control

CI/CD is arguably the core of what I do. I've built and administered pipelines in Jenkins, GitHub Actions, Bitbucket Pipelines, and CircleCI, and the specific tool matters less to me than the outcome: a pipeline that a team trusts enough not to double-check manually.

Jenkins is the tool I have the deepest administrative experience with, from Grras Solutions through Synoption, where I managed pipelines across multiple environments, added email notifications for build and deployment failures, and

generated automated daily operational reports. GitHub and Bitbucket are where I manage version control and, increasingly, where I define pipelines directly — GitHub Actions for personal projects like my Oracle Cloud chat application, and Bitbucket Pipelines for client work including Shopware and Laravel deployments at Emizen Tech.

Fastlane is more specialized: I've used it specifically for Android release automation, generating and uploading builds to the Google Play Console as part of a pipeline triggered by GitHub pushes and authenticated through Google Cloud, which allowed deployment without depending on a specific machine or server.

Across all of these tools, the pattern I look for is the same: builds that run automatically on push, checks that run before deployment, and a rollback path that doesn't require me to be present when something goes wrong. I've built pipelines with SSH-based deployment and rollback support for Laravel, Magento2, and Node.js applications specifically to guarantee that last part.

Cloud Platforms (Overview)

I've worked hands-on across AWS, Azure, GCP, Oracle Cloud Infrastructure, and DigitalOcean, along with Nexcess as a hosting platform for specific web application deployments. My approach to any cloud platform is the same regardless of vendor: understand the compute, networking, and storage primitives well enough to provision and secure them deliberately, rather than clicking through a console without understanding what each setting does.

AWS is where I have the broadest experience, including EC2, VPC, ELB, Route 53, and S3, primarily from Grras Solutions and Emizen Tech. Azure and GCP experience came from managing multi-cloud client infrastructure at Emizen Tech. Oracle Cloud Infrastructure is where I deployed and currently maintain a personal project, a containerized real-time chat application. Chapter 4 goes into more depth on each platform, including specific services and lessons learned.

Web Servers, Reverse Proxies, and Networking

Apache and Nginx are the two web servers I've configured most, often in the same environment serving different purposes. Nginx, in particular, has become one of my most-used tools — I've configured it as a reverse proxy for routing traffic to backend services (Node.js, Python, PHP), for SSL termination, and for handling both standard HTTP traffic and WebSocket connections, as in my real-time chat application project.

Reverse proxy architecture is a pattern I return to often because it solves a specific, recurring problem: routing multiple services or applications through a single entry point, with centralized SSL and access control, rather than exposing each service directly. My Docker-based automation platform for Sentronic GmbH's infrastructure builds directly on this pattern, using a pre-configured Nginx reverse proxy alongside dynamically provisioned containers.

SSL certificate management — through Let's Encrypt or manually configured certificates — and DNS configuration are things I handle as a normal part of deploying any client-facing application, not as a separate specialized task. I've set these up for personal projects, client Laravel and Magento deployments, and the Sentronic GmbH automation platform alike.

Databases and Release Management

I've worked with MySQL, PostgreSQL, MariaDB, and MongoDB, and the way I think about databases has shifted over time from “the thing an application connects to” toward “a system that needs its own deployment discipline.” That shift happened most concretely at Synoption, where I managed MySQL and PostgreSQL releases using Liquibase — a tool for tracking, versioning, and applying schema changes in a controlled, reversible way, rather than running ad hoc SQL scripts against production.

MongoDB experience comes from my work on the Sentronic GmbH Report Management System, where it serves as the operational database for historical industrial measurement data. Part of that engagement involved investigating performance issues and proposing architecture improvements for querying large volumes of historical data, while keeping MongoDB as the system of record — a good example of working with an existing database choice rather than defaulting to “replace it” as the first answer.

Across all of these, backup and recovery has been a consistent responsibility, from PostgreSQL, MariaDB, and MySQL backups at Grras Solutions and Emizen Tech through to production database support in later roles.

Monitoring, Observability, and Incident Response

I've configured and worked with Prometheus and Grafana for metrics collection and visualization, New Relic and Zabbix for application and infrastructure monitoring, and UptimeRobot for straightforward uptime checking — starting with hands-on exposure at Grras Solutions and expanding into production monitoring setups at Emizen Tech.

My approach to monitoring is shaped by a simple principle: alerts should tell you about a problem before a user does. At Synoption, I enhanced Jenkins pipelines with automated email notifications for build and deployment failures and generated daily operational reports, specifically so that issues surfaced on a predictable schedule rather than being discovered reactively.

Troubleshooting production issues — in pipelines, deployments, containers, and applications — has been a constant thread across every role, and it's the skill I'd say generalizes best across all the specific tools listed above. Whether the underlying system is a Jenkins pipeline, a Docker container, or a FastAPI service, the diagnostic process is largely the same: check logs, isolate the change that introduced the problem, and verify the fix under realistic conditions before considering it resolved.

Backend Development: Python, FastAPI, and REST APIs

My backend development experience is more recent than my infrastructure experience, and it's grown specifically through freelance client work, most significantly the Report Management System I built for Sentronic GmbH. I use Python and FastAPI to build backend services and REST APIs, and that project involved implementing distinct functional views (Live View, Trend View, and History View) for historical data, along with report generation logic supporting PDF, Excel, CSV, and raw data export formats.

What I find valuable about having both backend and DevOps experience is that I design APIs with their eventual deployment and monitoring in mind, rather than treating that as someone else's problem to solve later. I also approach backend debugging the same way I approach infrastructure troubleshooting — starting from logs and reproducible steps rather than guesswork. This is the area of my skill set I'm most actively growing, and I expect it to keep expanding as I take on more backend-focused freelance engagements.

Scripting and Automation: Bash, PowerShell, and YAML

Bash and PowerShell are the scripting languages I use most for automation, often in combination — Bash for Linux-side automation and PowerShell where a client environment or my own tooling runs on Windows, as in the Docker automation platform I built for Sentronic GmbH, which uses both to handle dynamic container creation, environment generation, and configuration updates.

YAML shows up constantly as the configuration format for CI/CD pipelines (Bitbucket Pipelines, GitHub Actions) and container orchestration (Docker Compose), and I treat it as seriously as any other code — reviewed, version-controlled, and tested before it reaches production. I think about infrastructure-as-code concepts the same way: configuration and deployment logic should live in version control and be reproducible, not exist only as manual steps someone remembers to run.

Collaboration and Productivity Tools

Jira and Confluence have been central to how I work at Synoption, where I used Jira for issue tracking and to trigger deployment pipelines directly from tickets, and Confluence to maintain documentation for deployment procedures and operational knowledge. I treat documentation as part of the deliverable, not an afterthought — a deployment procedure that only exists in my head isn't finished.

VS Code and Docker Hub are part of my daily development environment, and I use Git as the version control system underlying all of the CI/CD tools described earlier in this chapter. None of these tools are complicated on their own, but using them consistently — clear commit messages, up-to-date documentation, tickets that reflect actual work — is what makes a team's process trustworthy over time.

Chapter 4 — Cloud Experience

I've worked hands-on across five cloud and hosting platforms. This chapter goes deeper into each one specifically — what I used it for, which services, and what I learned — separately from the general technical skills covered in Chapter 3.

Amazon Web Services (AWS)

AWS is the cloud platform I have the broadest experience with, starting at Grras Solutions and continuing through Emizen Tech. I've worked with EC2 for compute, VPC for network isolation and security group configuration, ELB for load balancing traffic across instances, Route 53 for DNS management, and S3 for object storage.

My AWS work has generally centered on deploying and maintaining web applications for clients — provisioning instances, configuring security groups and networking correctly before opening anything to the internet, and setting up load balancing and DNS so that a domain reliably reaches the right backend. The lesson that's carried forward from this experience is to treat network configuration as a security decision, not a formality — most of the production issues I've seen traced back to overly permissive access rather than application bugs.

Microsoft Azure

My Azure experience comes from managing client infrastructure at Emizen Tech, where I worked across AWS, Azure, and GCP depending on which platform a given client was already using. Rather than treating each cloud as an entirely separate skill set, I've found that the underlying concepts — virtual machines, networking, storage, and access control — transfer across platforms once you understand them properly on one. Azure work at Emizen Tech involved supporting production deployments and infrastructure for client applications, alongside monitoring and troubleshooting in the same environments.

Google Cloud Platform (GCP)

GCP experience also comes primarily from Emizen Tech's multi-cloud client environments, and separately from a very specific use case: authenticating Android deployment pipelines through Google Cloud so that builds could be generated and uploaded to the Google Play Console without depending on a particular local machine. That project is a good example of using a cloud platform for a narrow, well-defined purpose — server-independent authentication — rather than full infrastructure hosting, and it's shaped how I think about matching a cloud service to the actual problem rather than defaulting to the biggest tool available.

Oracle Cloud Infrastructure (OCI)

Oracle Cloud Infrastructure is where I independently deployed and continue to maintain a personal project: a containerized real-time chat application, running on Docker and Docker Compose with Nginx as a reverse proxy handling both HTTP and WebSocket traffic. I chose OCI for this project specifically to get hands-on experience with a cloud platform outside the “big three,” and the experience confirmed what I'd already suspected — the core skills of provisioning a virtual machine, securing it, and deploying containers to it are portable across providers. I built a GitHub Actions CI/CD pipeline that deploys to the OCI virtual machine over SSH automatically on every push, which is the same deployment pattern I've used with SSH-based pipelines elsewhere, just applied to a different cloud provider.

DigitalOcean and Nexcess

DigitalOcean and Nexcess round out the hosting platforms I've worked with, primarily in the context of client web application deployments at Emizen Tech, where straightforward hosting was a better fit than a full enterprise cloud platform for a given client's scale and budget. These platforms reinforced a lesson that applies across every cloud experience in this chapter: choosing infrastructure is a cost-and-complexity decision as much as a technical one, and the right answer depends on what a client or project actually needs, not on which platform is most sophisticated.

What Multi-Cloud Experience Has Taught Me

Working across five different platforms has taught me to separate the concepts that matter from the vendor-specific interface around them. Compute, networking, storage, DNS, and access control work similarly everywhere, even when the

console, naming, and pricing model differ. That's made me comfortable picking up a new cloud platform quickly when a client or project requires it, rather than being limited to whichever provider I learned first. It's also made me a more careful evaluator of cloud costs and security defaults, since I've seen firsthand how the same mistake — an overly open access rule, an unmonitored resource — can occur on any platform if you're not deliberate about configuration.

Chapter 5 — Work Experience

This chapter documents each of my employers in detail — responsibilities, the specific technologies I used, challenges, and what I took away from each role. Chapter 2 told the story of how these roles connect; this chapter provides the full record.

Grras Solutions Pvt. Ltd. — DevOps Engineer Trainee (October 2022 – July 2023)

Grras Solutions, based in Jaipur, was a structured training environment rather than a production role, and I want to be clear about that distinction upfront — the value of this role was guided, hands-on repetition rather than independent ownership of live systems.

During this period, I gained hands-on experience with core AWS services, specifically EC2 for compute, VPC for networking, and ELB for load balancing. I worked with Docker for containerization and was introduced to Kubernetes fundamentals, focused on environment setup and deployment concepts rather than production cluster administration. I configured Apache and Nginx web servers with reverse proxy setups, which was my introduction to routing traffic through a controlled entry point rather than exposing an application server directly.

On the CI/CD side, I built pipelines integrating Jenkins with GitHub, automating builds triggered by code changes. I also got early, practical exposure to monitoring tools, specifically Prometheus and Grafana for metrics and visualization, and UptimeRobot for uptime checking. On the database side, I supported deployment and backup operations for PostgreSQL, MariaDB, and MySQL.

The biggest challenge at this stage wasn't technical difficulty so much as breadth — being introduced to a wide range of tools and concepts in a relatively short period and needing to actually retain and connect them, rather than treat each as an isolated exercise. What helped was consistently asking why something worked a certain way rather than just following instructions, a habit that's stayed with me since.

Grras Solutions taught me the fundamentals that everything else in my career has built on: Linux administration, core AWS services, containerization basics, and the habit of pairing infrastructure work with monitoring from the start rather than adding it later. It also gave me my first real exposure to the idea that infrastructure and application concerns are connected — a web server configuration decision affects how an application behaves, and a monitoring gap affects how quickly a problem gets noticed.

Emizen Tech Pvt. Ltd. — DevOps Engineer (November 2023 – December 2024)

At Emizen Tech, based in Jaipur, I moved from structured training into full production ownership, supporting multiple client projects simultaneously across a genuinely wide range of technology stacks.

I built and managed CI/CD pipelines using Bitbucket, GitHub, Jenkins, and Docker, tailored to whichever stack a given client used. This included dedicated CI/CD pipelines for Laravel, Magento2, and Node.js applications, and Android application release automation using Fastlane integrated with the Google Play Console. I also deployed PHP applications using Deployer for structured, repeatable releases.

Over the course of this role, I migrated and deployed applications built on Laravel, Magento, Magento2, Shopware, and Node.js to cloud servers. This meant regularly working with codebases and frameworks I hadn't necessarily used before, and getting comfortable, quickly, with reading unfamiliar code well enough to deploy and support it safely.

I administered Linux servers and cloud environments across AWS, Azure, and GCP, managed LEMP and LAMP server stacks, and containerized applications using Docker and Docker Compose. I also worked with Kubernetes for infrastructure management and deployed applications using Vercel where appropriate. Networking and security

responsibilities included configuring Nginx reverse proxies, Apache web servers, SSL certificates, and DNS records for production environments.

I configured Zabbix, UptimeRobot, and New Relic for monitoring and alerting across client environments, supported production deployments, application migrations, and release activities across staging and production, and performed backup and recovery operations for application and database environments. I also documented troubleshooting steps, error fixes, and CI/CD pipeline setups, and collaborated directly with development teams to resolve deployment and infrastructure issues as they came up.

The central challenge of this role was context-switching: supporting Laravel, Magento, Shopware, and Node.js applications for different clients meant I couldn't rely on deep specialization in any single stack. Instead, I had to get good at quickly understanding how a given application was structured and deployed, then working within that structure safely, rather than imposing a single preferred approach on every project.

By the end of this role, I had built genuine breadth across web application stacks, cloud platforms, and deployment tooling. The biggest lesson was operational: when supporting many different systems at once, consistent documentation and process discipline matter more than deep expertise in any one technology, because they're what let you move between systems quickly without making mistakes. This role also gave me my first sustained experience owning production reliability directly, rather than working under close supervision.

Synoption Pvt. Ltd. — DevOps Engineer (May 2025 – September 2025)

At Synoption, working remotely, my role narrowed in scope compared to Emizen Tech but deepened significantly in process discipline, centered on Jenkins administration, database release management, and structured collaboration through Jira and Confluence.

I managed and maintained Jenkins pipelines across multiple environments, upgraded existing pipelines, and improved deployment workflows over time rather than treating the pipeline as a fixed, one-time setup. I managed build and production deployments directly, automated repetitive operational tasks that had previously been done manually, and worked specifically to improve deployment reliability and reduce manual effort.

I managed Liquibase schema deployments for MySQL and PostgreSQL databases, coordinating application and database releases so that schema changes and application changes stayed in sync. This was a more rigorous approach to database changes than I'd used in prior roles — treating schema migrations as version-controlled, reviewed artifacts rather than one-off scripts.

I enhanced Jenkins pipelines with email notifications for build and deployment failures, configured alert mechanisms for application and deployment issues, and generated automated daily reports covering operational and deployment activity, so that problems and trends were visible on a predictable schedule.

I worked extensively with Jira for issue tracking and release workflows, including triggering deployment pipelines directly from Jira tickets, which tied release activity directly to tracked work rather than leaving it as an informal side process. I created and maintained Confluence documentation for deployment procedures and operational knowledge sharing, and collaborated closely with development and QA teams throughout release cycles.

I managed containerized applications and servers, supported production and staging environments, and handled deployment failures and production issues as they arose, including troubleshooting for application and container-related problems.

The main adjustment at Synoption was working within a more process-driven environment than I'd experienced before — every deployment tied to a ticket, every schema change reviewed and versioned, every operational report generated on schedule. Initially, this felt like more overhead than the faster-moving environment at Emizen Tech; over time, I came to see it as the reason releases were calm rather than stressful. This role taught me that good release engineering is less about any single tool and more about consistent process that removes ambiguity about what's being deployed, when, and why.

Freelance Consulting — DevOps and Backend Engineer (October 2025 – Present)

Since October 2025, I've worked independently, providing DevOps, backend development, and infrastructure consulting services to international clients under my own freelance practice. This work draws on everything from the roles described above, applied without the structure of a larger team around me.

My freelance engagements have centered on CI/CD automation, backend application development, cloud infrastructure, Docker-based deployments, production troubleshooting, technical architecture, and, in one significant engagement, enterprise reporting solutions. Unlike my earlier roles, where responsibilities were largely defined for me, freelance work has required me to scope engagements myself, communicate technical trade-offs clearly to non-technical stakeholders, and manage client expectations directly.

The most significant client relationship in this phase of my career is with Sentronic GmbH, a German industrial technology company, covered in full in Chapter 6. Working with an international client has meant adjusting to asynchronous, time-zone-shifted communication, being deliberate about documentation so that decisions are clear without a live conversation, and collaborating directly with stakeholders on requirement analysis, feature validation, and production troubleshooting rather than routing everything through a project manager.

Freelancing is where my backend development skills have grown the most, largely because client engagements have required building things, not just deploying them. It's also sharpened my ability to work independently: without a team to escalate to, I've had to become more comfortable diagnosing unfamiliar problems on my own, communicating honestly when something will take longer than expected, and taking full responsibility for the reliability of what I deliver. I see this as a natural continuation of the ownership I started building at Emizen Tech and the process discipline I developed at Synoption, now applied without a safety net.

Chapter 6 — Sentronic GmbH — International Client Engagement

Sentronic GmbH is a German industrial technology company, and my engagement with them, which began in October 2025 and continues today, is the centerpiece of my freelance consulting work. It's also the clearest single example of how I combine backend development, infrastructure automation, and direct client collaboration in one engagement, which is why it receives its own chapter rather than a brief mention.

The engagement has consisted of two distinct projects: an enterprise Report Management System, built primarily in Python and FastAPI, and a Docker-based automation platform supporting the underlying infrastructure. Both are described in full below. Throughout both, I've worked directly with Sentronic's stakeholders — not through an intermediary — on requirement analysis, feature validation, production troubleshooting, and technical discussions, which has meant taking real ownership of both the technical decisions and the communication around them.

Project One: The Report Management System — Business Problem and Architecture

Sentronic GmbH needed an enterprise Report Management System capable of generating structured technical reports from historical industrial measurement data — the kind of data produced continuously by industrial monitoring equipment, which needs to be turned into readable, exportable reports for technical and business stakeholders. The objective, as scoped, was to design and develop a system providing scalable reporting workflows, export capabilities, and production-ready backend services, not a one-off script or internal tool.

I designed and implemented the backend using Python and FastAPI, exposing REST APIs that the frontend and reporting workflows consume. MongoDB serves as the operational database, storing the historical measurement data the system reports on. I built the backend with the expectation that it would need to scale to handle large volumes of historical data over time, which shaped decisions about how data is queried and aggregated rather than treating every report as a fresh, unoptimized query against the full dataset.

Report Management System — Live View, Trend View, and History View

A core part of the system is three distinct ways of visualizing measurement data. Live View surfaces current or near-real-time measurement data. Trend View shows how measurements change over a defined period, which is what most

technical stakeholders actually care about when diagnosing equipment behavior. History View provides access to historical measurement records for reporting and auditing purposes. Designing and implementing these as distinct views, rather than one generic data table, meant thinking carefully about what each type of user actually needs to see and in what format.

Report Management System — Export Formats and Business Logic

I built report generation workflows supporting multiple output formats: PDF for formal technical reports, Excel (XLSX) for stakeholders who want to work with the data directly, CSV for lightweight data exchange, and raw data export for cases where a downstream system needs the unprocessed measurements. Supporting four export formats from the same underlying data meant building the report generation logic in a way that separates what data goes into a report from how it's formatted, rather than writing separate one-off exporters for each format.

Beyond the core reporting infrastructure, I developed specific business logic for multilingual report generation, since Sentronic's stakeholders and the equipment operators using these reports are not all working in the same language, and for multi-tank reporting workflows, which handle reporting across multiple industrial tanks or measurement points rather than a single source. Both of these requirements came directly from how Sentronic's equipment and customer base actually operate, and building them correctly required close collaboration with stakeholders who understood that operational context better than I did going in.

Report Management System — Production Issues and Architecture Improvements

Once parts of the system were in production, I investigated reporting issues as they arose, performing debugging and technical diagnostics to identify root causes rather than symptoms. I also prepared a technical proposal for optimizing the historical reporting architecture, specifically addressing large-scale historical data query performance, while keeping MongoDB as the operational database rather than proposing a database migration as the first solution. That distinction mattered: the goal was to make the existing system perform well at scale, not to solve a performance problem by introducing a new set of risks with a database migration.

Throughout this project, I collaborated directly with international stakeholders to analyze requirements, validate proposed solutions, and support delivery — communicating across time zones and, given the multilingual reporting requirement, being conscious that not every stakeholder was communicating in their first language either. This meant being extra deliberate about clear, unambiguous written communication and documentation.

Report Management System — Lessons Learned and Future Scope

This project reinforced something I'd suspected but hadn't tested at this scale: that designing a backend system well from the start — separating data logic from formatting, designing for multiple output formats early — pays off enormously once real requirements, like multilingual and multi-tank reporting, arrive later. It also taught me that performance problems in a historical-data system are usually architecture and query problems, not database problems, and that proposing a full migration should be a last resort, not a first instinct.

I produced technical documentation and solution recommendations for future project phases and reporting optimization, which means the next phase of this engagement is already scoped in terms of direction, even if the specific timeline depends on Sentronic's priorities.

Project Two: The Docker-Based Automation and Reverse Proxy Platform

Alongside the Report Management System, Sentronic needed a way to simplify service onboarding and automate routing and SSL management for the infrastructure supporting their applications — essentially, a way to bring up new services without manually reconfiguring a reverse proxy and certificates every time.

I designed and implemented an automated Docker container orchestration system built around a pre-configured Nginx reverse proxy. The core idea is that adding a new service shouldn't require manually editing Nginx configuration files and requesting a new SSL certificate by hand every time — the system handles both automatically as part of provisioning a new container.

I built scripts for dynamic container creation, environment generation, and configuration updates, using both PowerShell and Bash, since the target environment needed to support Windows-based tooling as well as standard Linux automation. This dual-scripting approach was a deliberate choice rather than a limitation — it meant the automation worked regardless of which environment it was triggered from.

Docker Automation Platform — SSL, Access Control, and Environment Validation

I implemented SSL certificate automation as part of the container provisioning flow, so that a new service becomes securely accessible without a separate manual certificate request step, along with access control mechanisms to manage which services and routes are exposed. The reverse proxy handles multi-service routing centrally, which keeps SSL and access policy in one place rather than duplicated across services.

I verified the entire setup on Windows Docker Desktop with WSL (Windows Subsystem for Linux), specifically to confirm compatibility and reliability in that environment rather than assuming a Linux-only deployment target would be sufficient. This mattered because it meant the automation had to behave consistently across two meaningfully different underlying environments.

The result is a platform that improved scalability and reduced manual configuration effort for containerized applications — new services can be onboarded through the automation rather than through a repeated manual process, and SSL and routing configuration stay consistent because they're generated the same way every time rather than hand-edited per service.

This project reinforced a pattern I now actively look for: recurring manual configuration work is usually a sign that the underlying task can be templated and automated, even when it initially seems too specific to standardize. It also gave me direct experience building automation that has to work reliably across different host environments (Linux and Windows/WSL), which required testing assumptions I might otherwise have taken for granted, like path handling and script portability.

What the Sentronic GmbH Engagement Represents

Taken together, these two projects are the best available example of how I actually work: backend development, infrastructure automation, and direct international client collaboration, applied to real production systems rather than a demo or a portfolio piece. The Report Management System shows my backend and data-architecture thinking; the Docker automation platform shows my infrastructure automation approach; and both required the same underlying skill of communicating clearly and reliably with stakeholders I've never met in person. If you want a single reference point for what I'm capable of, this engagement is it.

Chapter 7 — Major Projects — Case Studies

Beyond client and employer work, I've built and documented several independent projects, each written up publicly as an article on Medium or hosted in a public repository. This chapter presents each as a short case study: the problem, the architecture, my role, and what I learned.

Real-Time Chat Application — CI/CD Deployment on Oracle Cloud

Problem: I wanted hands-on experience deploying and operating a real-time application — one requiring persistent WebSocket connections, not just standard HTTP request-response — on a cloud platform outside the AWS, Azure, and GCP set I'd used professionally, to test how portable my skills actually were.

Architecture and stack: The application runs in Docker containers, orchestrated with Docker Compose, on a virtual machine hosted on Oracle Cloud Infrastructure. Nginx sits in front of the application as a reverse proxy, routing both standard HTTP traffic and WebSocket connections to the backend.

Implementation: I configured Docker networking so containers communicate correctly with each other and with the reverse proxy, and built a GitHub Actions CI/CD pipeline that connects to the Oracle VM over SSH and automatically rebuilds and restarts the relevant containers on every push to the repository.

Challenges: The main challenge was correctly proxying WebSocket connections through Nginx, which requires different configuration than standard HTTP proxying, specifically around connection upgrade headers — getting this wrong results in connections that silently fail to upgrade rather than producing an obvious error.

My role: I designed, built, and deployed the entire project independently, including the CI/CD pipeline and documentation.

Outcome: A working, automatically deploying real-time chat application, with the full deployment architecture, setup instructions, and CI/CD workflow documented in the project repository.

Lessons learned: This project confirmed that cloud deployment skills transfer well across providers, and it deepened my specific understanding of reverse-proxying non-standard traffic like WebSockets, which has since been useful in other reverse-proxy work, including the Sentronic GmbH automation platform.

Repository: github.com/dheerajsr/devops

Android Application CI/CD Auto-Deployment

Problem: Manual Android app releases — building, signing, and uploading to the Play Store — are repetitive and error-prone, and I wanted a pipeline where a release required nothing more than pushing code.

Architecture and stack: GitHub as the source repository and trigger, Fastlane to handle build generation and Play Store upload, and Google Cloud for authentication, which was the key design decision in this project.

Implementation: I configured the pipeline so that a build and deployment triggers automatically on every push to GitHub, with Fastlane generating and uploading the resulting build to the Play Store. Authentication is handled through Google Cloud rather than a locally stored credential, which was the specific design choice that made the deployment server-independent.

Challenges: Getting Fastlane's authentication working reliably without depending on a specific developer's local machine or credentials took the most iteration — the goal was a pipeline that would work identically regardless of who triggered it or from where.

My role: I designed and built the entire pipeline independently.

Outcome: A pipeline that automates APK and AAB generation, release management, and versioning, meaningfully improving deployment speed compared to manual releases.

Lessons learned: Server-independent authentication is worth the upfront complexity almost every time, because it removes a single point of failure — one person's machine or credentials — from a process that a team may eventually depend on.

Article: medium.com/@raodheeraj604/android-application-project-deployment-ci-cd-850451cdfa5b

Auto-Deployment of the Shopware E-Commerce Platform

Problem: Shopware deployments for a client project were being done manually, which was slow and left room for inconsistency between what was tested and what was actually deployed.

Architecture and stack: Bitbucket Pipelines as the CI/CD trigger, and PHP Deployer to handle the actual deployment logic on the target server, connected over SSH.

Implementation: I configured the pipeline to trigger automatically on every commit or push to the Bitbucket repository, establish an SSH connection to the remote server, and execute a deploy script that pulls and deploys the latest changes.

Challenges: Ensuring the deployment was safe to run repeatedly and wouldn't leave the application in a broken intermediate state if interrupted required careful use of Deployer's built-in release and rollback structure, rather than a simpler script that just overwrote files in place.

My role: I designed and implemented the full pipeline and deployment configuration.

Outcome: Reliable, low-effort updates to the production Shopware application, triggered automatically and requiring no manual server access for routine deployments.

Lessons learned: Using an established deployment tool rather than writing custom deployment scripts from scratch meant inheriting sensible defaults around releases and rollbacks that would have taken real effort to replicate safely on my own.

Article: medium.com/@raodheeraj604/deployment-of-shopware-with-deployer-method-55dd5358df04

CI/CD Pipelines for Laravel, Magento2, and Node.js Applications

Problem: Supporting multiple application stacks for different projects meant I needed a repeatable CI/CD pattern that could be adapted across frameworks, rather than building an entirely custom pipeline for each one.

Architecture and stack: Bitbucket Pipelines as the common CI/CD platform across all three application types, with SSH-based deployment scripts handling the actual release to remote cloud servers.

Implementation: I built pipelines that build, test, and deploy code changes automatically, with SSH and deployment scripts specifically designed to support zero-downtime updates and rollback if a deployment failed, rather than a simple overwrite-and-hope approach.

Challenges: Standardizing a pipeline pattern across genuinely different frameworks — Laravel, Magento2, and Node.js — meant identifying which parts of the process were truly framework-specific, such as build steps and dependency installation, versus which parts could be shared, such as deployment mechanics and rollback logic.

My role: I designed and implemented the pipeline pattern and adapted it across all three application types.

Outcome: Automated testing and staging workflows that reduced manual deployments and improved release consistency across environments, enabling deployments directly from local development to remote cloud servers without manual server access.

Lessons learned: Investing time in a reusable deployment pattern, rather than a one-off pipeline per project, paid for itself the second and third time it needed to be applied to a new stack.

Deploying a Web Application with Nginx and Reverse Proxy

Problem: A web application needed to be exposed securely and reliably to the internet, with proper SSL termination and the ability to route to different backend types — Node.js, Python, or PHP — depending on the project.

Architecture and stack: Nginx as the web server and reverse proxy, sitting in front of the application backend, with SSL handled through Let's Encrypt or manually configured certificates depending on the environment.

Implementation: I configured Nginx to route traffic to the appropriate backend, manage domain routing, and terminate SSL at the proxy layer rather than within the application itself, then tuned the configuration for performance and load handling.

Challenges: Balancing security — proper SSL configuration, secure headers — with performance — connection handling, caching where appropriate — required testing configuration changes under realistic load rather than assuming default settings were sufficient.

My role: I designed, configured, and documented the full reverse proxy setup independently.

Outcome: A production deployment pattern with high availability and secure traffic flow between clients and the application server, documented publicly as a reference for similar setups.

Lessons learned: This project became the foundation for how I approach reverse proxy configuration in every subsequent project, including the Sentronic GmbH Docker automation platform, which builds directly on the same Nginx patterns established here.

Article: medium.com/@raodheeraj604/deploying-a-web-application-using-nginx-server-and-reverse-proxy-3c84f82c9f59

Chapter 8 — AI Tools and AI-Assisted Engineering

How I Use AI Tools

AI-assisted development has become a normal part of how I work, and I want to describe that honestly rather than either overstating or downplaying it.

I use ChatGPT and Claude primarily for two things: working through a problem when I'm stuck, and reviewing my own reasoning before I commit to an approach. When I'm debugging something unfamiliar, I'll often describe the symptom and what I've already ruled out, and use the response to check whether I'm missing an obvious angle, rather than treating the output as an answer to copy directly. The value is in faster iteration on hypotheses, not in skipping the diagnostic process itself.

Cursor is the environment I use for AI-assisted development directly inside my editor, where I'm working with code, infrastructure configuration, and documentation together. Having AI assistance available in context, rather than in a separate chat window, has made it faster to draft boilerplate — pipeline configuration, deployment scripts, documentation structure — and then focus my own attention on the parts that actually require judgment: architecture decisions, security-relevant configuration, and anything going into production.

For documentation specifically, I use AI tools to help produce a first draft of technical write-ups, procedures, and explanations, which I then edit for accuracy against what I actually built. This has made me more consistent about documenting things at all, since the barrier to starting is lower.

I also use AI tools during code review, as a second pass after my own review, to catch things like inconsistent error handling or edge cases I might have overlooked, particularly in backend code written under time pressure.

None of this replaces understanding the system I'm working on. I don't deploy configuration or ship code I can't explain, regardless of whether AI assistance was involved in drafting it, because I'm the one who has to support it in production afterward. I think of these tools the way I think about any other tool in this document: useful for removing friction and speeding up the parts of the job that don't require judgment, not a substitute for the judgment itself.

Chapter 9 — Certifications

Red Hat Certified System Administrator (RHCSA)

I earned my RHCSA certification in April 2023, following my time at Grras Solutions. RHCSA validates practical, hands-on Linux system administration skills rather than theoretical knowledge, which is part of why I pursued it — I wanted a credential that reflected the ability to actually configure and troubleshoot a system, not just describe how one works.

The certification covers user and group management, storage configuration — partitions, logical volumes, and file systems — networking configuration, service management, security settings, and general system troubleshooting and automation.

Every Linux server I've administered since earning this certification — across Grras Solutions, Emizen Tech, Synoption, and my current freelance work — draws directly on this foundation. When I configure a firewall rule before exposing a new service, set up storage for a database that needs room to grow, or troubleshoot a service that won't start, I'm applying the same fundamentals RHCSA validated, not treating them as a one-time exam requirement.

RHCSA also shaped how I approach systems I haven't seen before. Because the certification is built around hands-on tasks rather than multiple-choice theory, it trained me to verify system state directly — checking actual configuration files, logs, and running processes — rather than assuming how a system is configured based on documentation or defaults. That habit has been useful well beyond Linux administration specifically; it's the same diagnostic approach I bring to debugging CI/CD pipelines, containers, and backend services.

Chapter 10 — What I Can Do

Rather than repeat a technology list, this chapter is a direct answer to a practical question: what can I actually do for a team or client?

Capabilities

I can build backend services and REST APIs in Python and FastAPI, designed from the start with deployment and monitoring in mind rather than treated as a separate concern to hand off afterward.

I can design and implement CI/CD pipelines using Jenkins, GitHub Actions, Bitbucket Pipelines, or CircleCI, tailored to whatever a team already has in place rather than insisting on a specific toolchain.

I can deploy and manage applications across AWS, Azure, GCP, and Oracle Cloud Infrastructure, provisioning compute, networking, and storage deliberately rather than by default configuration.

I can migrate applications between environments and cloud providers, having done this across Laravel, Magento, Magento2, Shopware, and Node.js applications for multiple clients, and I know what tends to break during a migration before it happens.

I can containerize applications with Docker and Docker Compose, and design multi-service architectures behind an Nginx reverse proxy with centralized SSL and routing, as I've done both for personal projects and for Sentronic GmbH's production infrastructure.

I can automate manual, repetitive operational work using Bash, PowerShell, and pipeline tooling, and I actively look for the point where a manual task has been repeated enough times to justify automating it.

I can troubleshoot production issues across the full stack — Linux systems, CI/CD pipelines, containers, databases, and backend services — starting from logs and reproducible evidence rather than guesswork.

I can manage database schema changes safely using tools like Liquibase, treating migrations as version-controlled, reviewed changes rather than one-off scripts run directly against production.

I can optimize systems under real constraints, including proposing architecture improvements for large-scale historical data querying without requiring a full database migration, as I did for Sentronic GmbH.

I can set up monitoring and alerting using Prometheus, Grafana, New Relic, Zabbix, and UptimeRobot, with alerting configured so that problems are caught before users notice them.

I can document infrastructure, deployment procedures, and technical decisions clearly enough that someone else — a teammate or a client — can follow them without needing me to explain it live.

I can collaborate directly with international stakeholders and clients, across time zones and without an intermediary, on requirement analysis, technical validation, and production support.

I can work independently, scoping and delivering engagements end-to-end as I currently do in freelance consulting, or as part of a larger team, adapting to whichever structure a given role or engagement requires.

What ties all of this together is that I don't see these as separate skills to be deployed situationally. On any given engagement, I'm usually doing several of them at once — building a backend service, deciding how it should be deployed, setting up the monitoring that will tell me if it's working, and documenting all of it clearly enough for someone else to pick up. That's the kind of engineer I am, and it's the value I'd bring to any team or client that brings me on.

Chapter 11 — Why Work With Me

If you're evaluating whether to bring me onto a team or engage me for a project, here's the honest case, focused on value rather than credentials I've already listed elsewhere in this document.

Reliability and Ownership

I treat production reliability as the actual measure of success, not a secondary concern after whether something works in a demo. Across every role documented in this portfolio, from database backups at Grras Solutions to production release management at Synoption to live industrial reporting systems at Sentronic GmbH, the systems I've built and maintained have stayed dependable because I plan for failure modes upfront rather than reacting to them after the fact.

I don't hand off responsibility at the point of deployment. If something I built breaks, I want to be the one who understands why and fixes it, not the one who escalates it and waits. This matters more with freelance and consulting work specifically, where there often isn't a larger team to absorb that responsibility if I don't.

Automation as a Mindset, Not a Checkbox

I don't automate to look busy or to check a box on a resume. I automate specific, recurring manual work because that's where errors and inconsistency accumulate, and I've done this consistently — Jenkins pipelines at three different companies, Android release automation, Docker container provisioning — always in response to an actual repeated task, not a hypothetical one.

Evidence-Based Problem Solving

Whether I'm debugging a failed deployment, a misbehaving container, or an unfamiliar backend service, my process starts the same way: logs, reproducible steps, and isolating what actually changed, rather than guessing based on what usually causes similar problems. This has served me well specifically because so much of my work has involved systems and codebases I didn't originally build.

Documentation as Part of the Deliverable

I don't consider a deployment procedure, a piece of infrastructure, or an API finished until it's documented well enough for someone else to use or maintain without me in the room. This habit, built through Confluence documentation at Synoption and technical write-ups from my personal projects, means the work I hand off doesn't create a dependency on me personally.

Continuous Learning Without Chasing Trends

I've built real depth in Linux, CI/CD, and cloud infrastructure over two years, and more recently added backend development because it made me better at the rest of it. I evaluate new tools by whether they solve a real problem I have, not by how much attention they're getting.

Communication Across Distance and Difference

Working with Sentronic GmbH and other international clients has required me to communicate clearly in writing, across time zones, and with people whose first language may not be English. That experience has made me more careful, not less, about being explicit and unambiguous, which matters whether you're a hiring manager evaluating clarity of thought or a client who needs confidence that I understand your requirements correctly.

Backend and Infrastructure, Together

Most engineers specialize in one side of this or the other. I bring both, which means I design systems with deployment and monitoring in mind from the start, and I deploy and operate systems with an understanding of how they're actually built. For a team or client that needs one person who can move across that boundary, that combination is the specific value I offer.

Chapter 12 — Contact

Get in Touch

Thank you for reading this far. This document represents the most complete account of my professional engineering career that I currently maintain, and I update it as my experience grows.

If you'd like to discuss a role, a freelance engagement, or a consulting project, I'm easy to reach through any of the channels below. I'm open to remote, hybrid, and on-site work, as well as contract and full-time positions, and I actively take on freelance and consulting engagements for clients anywhere in the world.

Dheeraj Singh Rao

Phone: +91-9001913550

Email: raodheeraj604@gmail.com

LinkedIn: linkedin.com/in/dheeraj-singh-rao-1918b2157

GitHub: github.com/dheerajsr

Bitbucket: bitbucket.org/automation-cicd/workspace/projects

Medium: medium.com/@raodheeraj604