DomainForge – Automated Subdomain Generator and Deployment Assistant

Submitted in the partial fulfillment for the award of

the degree of

BACHELOR OF ENGINEERING

IN

Cloud Computing , InfoSec

Submitted by:

Tanvi Agarwal (22BCC70170)
Vedant Pawar (22BCC70168)
Priyanshu (22IIS70022)
Upilli Dileep (22BCC70009)

Subject: Capstone Project-1(22CSR399)

Under the Supervision of:
Dr. Ankit Garg

Department of AIT-CSE

Outline

- Introduction to Project
- Problem Formulation
- Objectives of the work
- Methodology used
- System Architecture
- Results and Outputs
- Conclusion
- Future Scope
- References

Introduction

- DomainForge – Automated Subdomain Generator and Deployment Assistant is an automation framework designed for small to mid-sized development teams.
- It integrates subdomain generation, DNS configuration, and application deployment into a unified system.
- This project reduces deployment time and configuration errors, offering a web-based solution linked with GitHub repositories, domain registrar APIs, and SSL/TLS automation.
- Unlike heavy enterprise platforms, DomainForge provides a lightweight yet powerful deployment environment suitable for resource-limited setups.

Problem Formulation

- Small development teams face multiple challenges while deploying applications due to manual DNS and subdomain configuration processes.
- Existing enterprise solutions are resource-intensive and require specialized DevOps expertise.
- Lightweight scripts fail to provide scalability, integration, and automation, leading to inefficiencies and errors.
- Therefore, there is a need for a hybrid system that combines the simplicity of lightweight tools with the advanced automation capabilities of enterprise-grade platforms.

Objectives of the Work

- Automate domain provisioning and DNS configuration using registrar APIs such as Cloudflare to eliminate manual setup and ensure instant, error-free subdomain creation during deployments.
- Simplify end-to-end application deployment by integrating GitHub repositories with automated build and deployment workflows, enabling one-click execution from code push to live hosting.
- Provide flexible deployment support for both containerized (Docker-based) and traditional environments, allowing developers to deploy different types of applications seamlessly.
- Ensure a secure, modular, and resource-efficient architecture that operates efficiently even on low-resource setups, making it suitable for startups, freelancers, and educational environments.

Methodology used

The methodology adopted for DomainForge involves multiple structured phases:

- Requirement Analysis: Identified challenges faced by small developers and startups in deployment.
- System Design: Creating a modular architecture with separate layers for Web Interface, Backend Service, and Deployment Engine.
- Implementation: Used React for frontend, Node.js for backend, and Bash scripts with Docker for automation.
- Integration: Incorporated APIs from Cloudflare for DNS automation and GitHub for repository management.
- Testing: Conducted unit, integration, and user acceptance testing to ensure reliability and usability.

System Architecture

DomainForge's architecture includes four key components:

- Web Interface (React): Provides a user-friendly interface for deployment and subdomain management.
- Backend Service (Node.js + Express): Orchestrates automation workflows and API communications.
- Deployment Engine (Bash + Docker): Executes automation scripts for repository cloning, configuration, and hosting.
- External API Integration (Cloudflare, GitHub): Enables automated DNS setup and repository-based deployment.

  This modular design ensures scalability, flexibility, and maintainability.

Result and Output

DomainForge demonstrated significant performance improvements compared to manual deployment methods.

- Deployment time reduced from 10–15 minutes to 2–4 minutes (75–80% improvement).
- Configuration errors reduced by approximately 75%.
- DNS provisioning achieved 100% accuracy through automated API calls.
- Usability testing showed positive feedback for the web interface, particularly among non-technical users.
- System performed efficiently even on low-cost VPS setups with minimal overhead.

Conclusion

- It achieves significant time savings (up to 75–80%), reduces configuration errors, and minimizes human intervention, thus optimizing productivity in deployment cycles.
- The web-based interface ensures accessibility for users with limited DevOps experience, offering a user-friendly dashboard for complete deployment control.
- By supporting both containerized (Docker-based) and non-containerized environments, it enhances flexibility and compatibility with diverse project setups.
- DomainForge’s modular architecture ensures ease of maintenance, scalability, and adaptability to evolving technologies or API integrations.

Future Scope

- Integration of native CI/CD support to enable automated testing, build validation, and incremental rollouts for continuous delivery pipelines.
- Multi-cloud deployment compatibility across AWS, Azure, and Google Cloud Platform (GCP) to enhance flexibility and scalability in hosting environments.
- Real-time monitoring and alerting systems for deployment health checks, error tracking, and performance metrics visualization.
- AI-driven optimization for predictive scaling, auto-healing deployments, and intelligent resource utilization.
- Advanced error-handling, rollback, and self-recovery mechanisms to ensure uninterrupted deployment cycles.

References

- Hounsel et al., “Comparing the Effects of DNS, DoT, and DoH on Web Performance”, ACM IMC, 2020.
- Rodriguez & Buyya, “Container-based Cluster Orchestration Systems”, Journal, 2018.
- Saleh et al., “A Systematic Literature Review on CI/CD Pipeline Security”, arXiv, 2025.
- Barnes et al., “Automatic Certificate Management Environment (ACME)”, IETF RFC 8555, 2019.
- Forsgren, Humble & Kim, “Accelerate: The Science of Lean Software and DevOps”, IT Revolution, 2018.