TARTU ÜLIKOOL

FACULTY OF SOCIAL SCIENCES

NARVA COLLEGE

INFORMATION TECHNOLOGY SYSTEMS DEVELOPMENT

Yevhenii Tomberg

DEPLOYMENT SOFTWARE FOR APPLICATION DEPLOYMENT ON LINUX BASED SERVERS

Diploma thesis

Supervisor: Andre Sääsk, M. Sc.

Narva 2025

Olen koostanud töö iseseisvalt. Kõik töö koostamisel kasutatud teiste autorite tööd, põhimõttelised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on viidatud.

……/töö autori allkiri/

Non-exclusive license to reproduce the thesis: I, Yevhenii Tomberg (date of birth: 01.04.2002),

1. herewith grant the University of Tartu a free permit (non-exclusive license) to reproduce, for the purpose of preservation, including for addition to the DSpace digital archives until the expiry of the term of validity of the copyright “developing an application for participation in local events”, supervised by A. Sääsk,

2. I am aware that the author retains the right referred to in point 1.

3. This is to certify that granting the non-exclusive license does not infringe the intellectual property rights or rights arising from the Personal Data Protection Act.

Narva, 23.04.2024

Contents

Terms and abbreviations

Electron – a framework for building desktop applications using JavaScript, HTML, and CSS. By embedding Chromium and Node.js into its binary, Electron allows you to maintain one JavaScript codebase and create cross-platform apps that work on Windows, macOS, and **Linux** – no native development experience required.

HTML – the standard markup language for creating Web pages.

CSS – the language we use to style a Web page.

JavaScript – the Programming Language for the Web. JavaScript can update and change both HTML and CSS. JavaScript can calculate, manipulate and validate data.

Chromium – an open-source browser project that aims to build a safer, faster, and more stable way for all users to experience the web. This site contains design documents, architecture overviews, testing information, and more to help you learn to build and work with the Chromium source code.

Node.js – a powerful open-source JavaScript runtime environment that is built on Chrome's V8 JavaScript engine. It allows developers to run JavaScript code on the server-side, rather than just in web browsers. This means that developers can use JavaScript to write server-side scripts for web applications, enabling them to create dynamic and scalable web applications.

API – Application Programming Interface. It's essentially a set of rules and protocols that allow different software applications to communicate with each other. APIs define the methods and data formats that developers can use to interact with a service, library, or system.

URL – Uniform Resource Locator. It is a reference or address used to locate a resource on the internet. A URL specifies the protocol used to access the resource, the domain or IP address where the resource is located, and the specific path to the resource on that server.

JSON – JavaScript Object Notation. It is a lightweight data interchange format that is easy for humans to read and write, and easy for machines to parse and generate. JSON is often used to transmit data between a server and a web application, as well as for storing configuration data and exchanging data between different systems.

Version Control System (VCS) – a software tool or system that helps developers manage changes to source code over time. It provides a way to track modifications to files, compare

changes between different versions, revert to previous versions if needed, and collaborate with others on the same codebase. Version control systems are essential for software development teams to maintain code integrity, track progress, and facilitate collaboration.

Git – a distributed version control system (DVCS) widely used for tracking changes in source code during software development. Developed by Linus Torvalds in 2005 for managing the Linux kernel development, Git has since become one of the most popular version control systems in the world.

API endpoint – a specific URL within an API that represents a resource or a service. It defines the location where clients can access or interact with a particular function, data, or operation provided by the API.

HTTP – Hypertext Transfer Protocol. It is an application-layer protocol used for transmitting hypermedia documents, such as HTML files, over the internet. HTTP is the foundation of data communication for the World Wide Web.

JWT – JSON Web Token. It is an open standard (RFC 7519) that defines a compact and self-contained way for securely transmitting information between parties as a JSON object. JWTs are commonly used for authentication and authorization in web applications and APIs.

NPM – Node Package Manager. It is the default package manager for the Node.js runtime environment and the largest software registry in the world. npm is used by developers to discover, share, and reuse code packages and libraries for building Node.js applications.

Axios – a popular JavaScript library used for making HTTP requests from web browsers and Node.js environments. It provides a simple and elegant API for interacting with web servers and consuming APIs.

JSX – JavaScript XML. It's an extension to the JavaScript language syntax that allows developers to write HTML-like code within JavaScript. JSX is most commonly associated with React, a popular JavaScript library for building user interfaces.

Redux – a predictable state container for JavaScript applications, primarily used with libraries like React or Angular for managing application state. It is inspired by Flux, a pattern for managing state in JavaScript applications, but it introduces a few key concepts and patterns to simplify state management and make it more predictable.

SSH – Secure Shell. It is a cryptographic network protocol used for secure communication between two networked devices, typically a client and a server. SSH is commonly used for remote access to servers and for secure file transfer between computers.

Python – a high-level, interpreted programming language known for its simplicity and readability.

Nginx – a popular open-source web server, reverse proxy server, and load balancer known for its high performance, scalability, and reliability. Originally created by Igor Sysoev in 2004, Nginx has since gained widespread adoption and is used by millions of websites and web applications worldwide.

GitHub – a web-based platform and hosting service for version control using Git. It provides a wide range of features for software development teams and individual developers, including code hosting, version control, collaboration tools, project management, and continuous integration/continuous deployment (CI/CD) pipelines.

Parser – a software component or program that analyzes a piece of input data (such as text or code) according to a set of rules defined by a formal grammar or syntax. The parser breaks down the input data into its component parts, validates their structure and relationships, and generates a data structure (such as a parse tree or abstract syntax tree) that represents the parsed content in a structured and meaningful way.

Docker – an open-source platform for developing, shipping, and running applications in containers. Containers are lightweight, portable, and self-sufficient units that encapsulate all the necessary dependencies and components required to run an application, including the code, runtime, libraries, and system tools. Docker provides tools and workflows for creating, managing, and deploying containers efficiently across different environments.

Server – a computer or a software program that provides services or resources to other computers, known as clients, over a network. Servers are essential components of client-server architecture, where clients request and consume services or data provided by servers.

Java – a widely used, high-level programming language developed by Sun Microsystems (now owned by Oracle Corporation) in the mid-1990s. It is designed to be platform-independent, meaning that Java programs can run on any device or operating system that has a Java Virtual Machine (JVM) installed. Java is known for its simplicity, reliability, security, and scalability, making it a popular choice for building a wide range of applications, from mobile apps to enterprise systems.

Spring Framework – an open-source, comprehensive framework for building enterprise Java applications. It provides a wide range of features and capabilities for developing

scalable, maintainable, and high-performance applications across various domains, including web development, microservices architecture, and enterprise integration.

Introduction

In today's fast-paced world of software development, deploying applications to remote Linux servers can often be a complex and time-consuming process. Enter YTD Deployer – a cutting-edge solution designed to streamline and simplify the deployment of your applications with ease.

Built on Electron, a powerful framework for building cross-platform desktop applications using web technologies such as HTML, CSS, and JavaScript, YTD Deployer offers seamless and intuitive user experience for managing deployments from your local machine to remote Linux servers.

With YTD Deployer, developers can say goodbye to manual and error-prone deployment processes. Whether you're deploying web applications, microservices, or any other type of software, YTD Deployer provides a unified interface that automates common deployment tasks and handles the complexities of server configuration.

Problem

One common problem encountered during application deployment on remote Linux servers is the lack of consistency and reliability across different environments. This inconsistency can lead to deployment failures, runtime errors, and unexpected behavior, ultimately impacting the availability and performance of the deployed applications.

Here are some specific challenges associated with application deployment on remote Linux servers:

Environment Configuration: Remote Linux servers often have different configurations, dependencies, and software versions, which can vary from one server to another. This inconsistency can lead to compatibility issues and runtime errors when deploying applications that rely on specific dependencies or system configurations.

Dependency Management: Managing dependencies and libraries required by the application can be challenging, especially when deploying to remote Linux servers with limited internet connectivity or restricted access to package repositories. Ensuring that all required dependencies are installed and properly configured on each server can be time-consuming and error-prone.

Network Connectivity: Deploying applications to remote Linux servers over the network can be susceptible to network interruptions, latency, and bandwidth limitations. Slow or unreliable network connections can cause deployment failures or timeouts, especially when transferring large application files or dependencies.

Security Concerns: Security is a critical consideration when deploying applications to remote Linux servers. Ensuring secure authentication, encryption, and access control mechanisms are in place to protect sensitive data and prevent unauthorized access to the server infrastructure.

Deployment Automation: Manual deployment processes are prone to human error and can be time-consuming, especially when deploying applications to multiple servers or environments. Implementing automated deployment workflows and scripts can help streamline the deployment process and ensure consistency across different environments.

Rollback and Recovery: In the event of deployment failures or errors, it's essential to have mechanisms in place for rollback and recovery. Rolling back to a previous version of the application or reverting configuration changes can help mitigate downtime and minimize the impact on users.

Monitoring and Logging: Monitoring the deployed applications and server infrastructure is crucial for identifying issues, diagnosing problems, and ensuring optimal performance. Implementing comprehensive monitoring and logging solutions can provide insights into application health, resource utilization, and performance metrics.

Solution

YTD Deployer – Simplifying Consistent and Reliable Application Deployment on Remote Linux Servers

YTD Deployer is a robust and comprehensive solution designed to address the challenges associated with application deployment on remote Linux servers. By leveraging its powerful features and capabilities, YTD Deployer streamlines the deployment process, ensures consistency across different environments, and enhances the reliability and performance of deployed applications.

Key Features and Solutions provided by YTD Deployer:

1. Environment Configuration Management:

YTD Deployer provides a unified interface for configuring and managing server environments, allowing users to define consistent configurations across different servers.

It supports configuration management tools such as Ansible, Chef, and Puppet, enabling automated provisioning and configuration of server environments to ensure consistency and reliability.

2. Dependency Management:

YTD Deployer automates dependency management by packaging application dependencies into self-contained deployment artifacts.

It supports containerization technologies such as Docker, enabling the creation of lightweight and portable containers that encapsulate the application and its dependencies, ensuring consistent runtime environments across different servers.

3. Network Connectivity Optimization:

YTD Deployer optimizes network connectivity for deployment by implementing intelligent transfer protocols and bandwidth management techniques.

It supports resumable transfers and parallelized downloads/uploads, allowing for efficient and reliable data transfer even over slow or unreliable network connections.

4. Security Enhancements:

YTD Deployer enhances security by implementing robust authentication, encryption, and access control mechanisms.

It supports secure authentication methods such as SSH keys and integrates with identity management solutions such as LDAP and Active Directory for centralized user authentication and authorization.

5. Deployment Automation:

YTD Deployer automates the deployment process with customizable deployment workflows and scripts.

It supports continuous integration/continuous deployment (CI/CD) pipelines, enabling seamless integration with version control systems (e.g., Git) and automated deployment to multiple environments.

6. Rollback and Recovery:

YTD Deployer provides built-in support for rollback and recovery mechanisms to mitigate deployment failures and errors.

It maintains version history and deployment logs, allowing users to easily rollback to previous versions or perform automated recovery actions in the event of deployment issues.

7. Monitoring and Logging:

YTD Deployer offers integrated monitoring and logging features for tracking deployment progress and monitoring application health.

It integrates with monitoring tools such as Prometheus, Grafana, and ELK stack for real-time monitoring, alerting, and log analysis.

Outline

Section 1 provides descriptions of comparable solutions.

Section 2 describes the application development process.

Section 3 presents the development results and ideas for the future.

1. Market Research

1.1. ManageEngine

1.1.1. Description

ManageEngine is a suite of IT management software solutions developed by Zoho Corporation, a leading provider of cloud-based software products. ManageEngine offers a wide range of tools and applications designed to help IT administrators and organizations manage and optimize their IT infrastructure, systems, networks, and services more effectively.

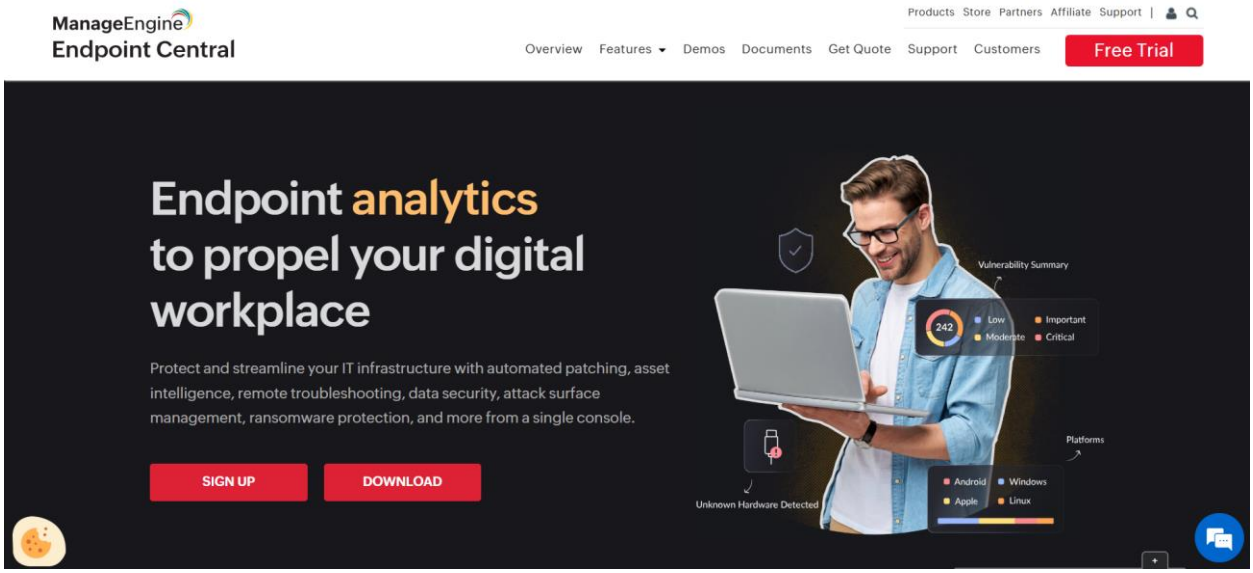

Figure 1. ManageEngine landing page. (https://www.manageengine.com/)

1.1.2. Strengths and weaknesses

Strengths:

1. Comprehensive Suite
2. Modular Approach
3. User-Friendly Interface
4. Automation and Integration
5. Scalability and Flexibility

Weaknesses:

1. Complexity
2. Customization Limitations
3. Integration Challenges

4. Support and Documentation

1.2. Chef

1.2.1. Description

Chef Software, Inc., often referred to simply as Chef, is a company that provides automation software for infrastructure management. Chef's flagship product is Chef Infra, formerly known as Chef, which is a configuration management tool used for deploying and managing infrastructure as code. Chef Infra allows users to define the desired state of their infrastructure using code, typically written in Ruby-based domain-specific language (DSL). With Chef Infra, users can automate the provisioning, configuration, and management of servers, virtual machines, containers, and cloud resources across heterogeneous environments.

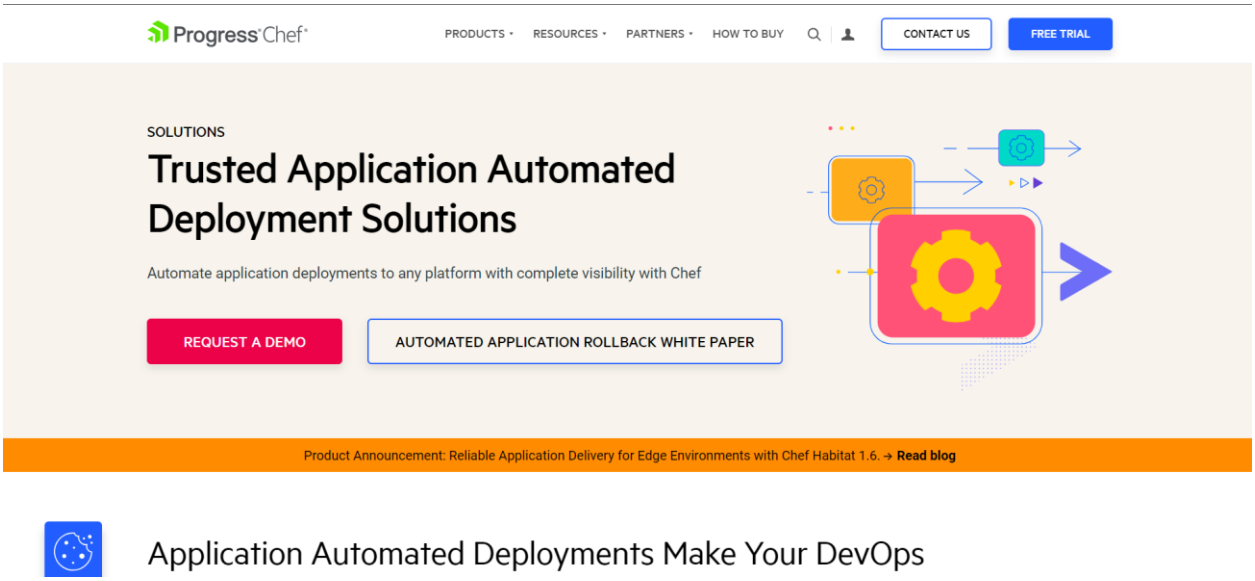

Figure 2. Chef landing page. (https://www.chef.io/)

Strengths:

1. Powerful Configuration Management
2. Scalability and Flexibility
3. Infrastructure as Code (IaC)
4. Community and Ecosystem
5. Integration and Extensibility

Weaknesses:

1. Learning Curve
2. Complexity

3. Resource Intensive
4. Competition
5. License Costs

2. Application Development

2.1. Backend Services

2.1.1. Application Configuration

Java version: 17

Spring Framework version: 3.2.1

MySQL version: 8

Jasypt version: 3.0.5

Jsonwebtoken version: 0.11.5

Application is up and running on the following host (entry endpoint):

ytd-deployer.tomberg.tech.

2.1.2. Public Endpoints

POST /api/public/login – login endpoint.

This endpoint is waiting for a request body from user with following credentials:

1. Username
2. Password

If credentials are correct and user is not locked/disabled the program will generate JWT, and it will send it back to the user. Otherwise, the user will be notified that credentials are incorrect.

Credentials are alive for 24 hours.

POST /api/public/authenticated – endpoint to figure out if user's session is alive.

This endpoint is used by the program itself to check if user can proceed with any action.

POST /api/public/register – registration endpoint.

This endpoint is used to register new users in the system and check input data.

This endpoint is waiting for a request body from user with following credentials:

1. Username
2. Password
3. Email

2.1.3. CRUD Endpoints

The application has three data types what user can manage by CRUD API:

1. dockerImage
2. server
3. user

POST /api/crud/:datatype – create mapping

PUT /api/crud/:datatype/:id – update mapping

DELETE /api/crud/:datatype/:id – delete mapping

2.1.4. Authentication

The application is using JWT authentication. Below is the configuration of the JWT generation:

Secret key is encrypted using “HmacSHA256” algorithm. JWT signed using HS512.

Expiration time: 24 hours.

2.2. Frontend Services

2.2.1. Preface

In fact, the application does not have a so-called front end. YTD Deployer is a desktop application that runs on Electron. Since Electron uses NodeJS, it can display HTML and use all sorts of web development features. It is NodeJS that allows the application to interact with the operating system and its utilities.

2.2.2. React and Other Libraries

The application is using the following libraries and dependencies:

Electron – a framework for building cross-platform desktop applications using web technologies.

electron-forge – a command-line tool and framework for streamlining the development, packaging, and distribution of Electron applications.

node-scp – a Node.js module that provides a simple interface for performing secure file transfers using the Secure Copy Protocol (SCP)

ssh2 – a library for Node.js that enables developers to implement Secure Shell (SSH) client and server functionality in their applications.

Axios – a popular Promise-based HTTP client for JavaScript that can be used in both browser and Node.js environments.

Bootstrap – a popular open-source front-end framework primarily used for designing and developing responsive and mobile-first websites and web applications.

React – a popular JavaScript library for building user interfaces, particularly for single-page applications (SPAs) and interactive web applications.

react-dom – a package in the React ecosystem that provides DOM-specific methods for working with React.

react-icons – a popular library for adding icons to React applications.

react-redux – an official library for integrating React applications with Redux, a predictable state container for JavaScript applications.

react-modal – a popular library for creating modal dialogs in React applications.

react-router-dom – a popular routing library for React applications that enables developers to implement client-side routing and navigation.

react-scripts – a set of scripts and configurations provided by the Create React App toolchain. It simplifies the process of setting up and managing React applications by abstracting away the complexities of build configurations, bundling, and development server setup.

react-spinners – a library for creating loading spinners and animated loading indicators in React applications.

react-toastify – a library for displaying notifications and toast messages in React applications.

react-tooltip – a library for creating tooltips in React applications.

UUID – a widely-used library for generating universally unique identifiers (UUIDs) in JavaScript.

2.3. Development Process

2.3.1. Project Setup

Set up the development environment by installing necessary tools and dependencies for ElectronJS, React, and Spring Boot.

Create a new project directory and initialize Git for version control.

2.3.2. Backend Development with Spring Boot

Create a new Spring Boot project using Spring Initializr or your preferred method.

Define the necessary data models, controllers, services, and repositories for managing deployment configurations, server connections, and deployment processes.

Implement RESTful APIs to handle requests from the frontend for deploying applications, managing servers, and retrieving deployment status.

Configure Spring Security for authentication and authorization if needed.

Test the backend APIs using tools like Postman or Swagger.

2.3.3. Frontend Development with React

Set up a React project using Create React App or your preferred method.

Design the user interface for the YTD Deployer application, including pages for managing deployment configurations, servers, and deployment processes.

Implement React components for displaying deployment status, managing configurations, and interacting with the backend APIs.

Integrate React Router for client-side routing to navigate between different pages of the application.

Implement form validation and error handling for user input.

Test the frontend components and interactions using tools like Jest and React Testing Library.

2.3.4. Integration with ElectronJS

Install ElectronJS as a dependency in the React project.

Configure Electron main process and renderer process to communicate with each other and with the backend APIs.

Set up Electron window management, including creating a main window for the React application and handling window events.

Implement Electron menu and tray functionality for accessing application features and options.

Package the Electron application for distribution using tools like Electron Builder or Electron Forge.

2.3.5. Integration Testing

Perform integration testing to ensure that the frontend React components and Electron features work seamlessly together.

Test the end-to-end flow of the application, including user interactions, API requests, and server responses.

Address any issues or bugs identified during testing and make necessary adjustments to the codebase.

2.3.6. Deployment

Deploy the YTD Deployer application to a staging environment for final testing and validation.

Once the application is stable and ready for production, deploy it to a production environment for end users to access.

Monitor the application's performance, usage, and user feedback, and make continuous improvements based on insights gathered from monitoring and feedback.

2.3.7. Maintenance and Updates

Regularly maintain and update the YTD Deployer application to address security vulnerabilities, compatibility issues, and feature requests.

Follow best practices for version control, documentation, and collaboration to ensure smooth development and maintenance of the application over time.

2.4. Functional Requirements

Functional requirements for the YTD Deployer application:

1. User Authentication

The application provides user authentication functionality, allowing users to securely log in with their credentials.

Users can register for a new account or reset their password if they forget it.

2. Dashboard

Upon logging in, users should be presented with a dashboard that provides a list of connected servers.

3. Deployment Configuration Management

Users can create, update, view, and delete deployment configurations for different applications.

Each deployment configuration includes details such as the application name, version, deployment script, server targets, and environment variables.

4. Deployment Process

Users can initiate deployment processes for selected applications to specified servers.

The deployment process should execute the deployment script on the target servers, copy application files, set environment variables, and handle any dependencies or configurations required for the application to run.

5. Deployment Status Tracking

Users can track the status of ongoing deployment processes in real-time.

The application should provide feedback on the progress of each deployment, including status updates, error messages, and logs generated during the deployment process.

6. Notifications

The application provides notifications to users about important events, such as successful deployments, failed deployments, server errors, or pending actions requiring user attention.

Notifications are delivered via the user interface.

2.5. Non-Functional Requirements

Non-Functional requirements for the YTD Deployer application:

1. Performance

The application is responsive and provides quick response times for user interactions, such as navigating between pages, initiating deployment processes, and viewing deployment status.

Deployment processes are executed efficiently, minimizing downtime and ensuring timely delivery of applications to target servers.

2. Scalability

The application is scalable to accommodate a growing number of users, deployment configurations, and server connections.

It can handle concurrent deployment processes and user interactions without degradation in performance or responsiveness.

3. Security

The application adheres to industry best practices for security, including data encryption, secure communication protocols, and protection against common security threats such as cross-site scripting (XSS) and SQL injection.

User authentication and authorization mechanisms is implemented securely to prevent unauthorized access to sensitive functionality or data.

2.6. User Interface

2.6.1. UI examples

In this chapter, I'm going to show different views of YTD Deployer application.

Figure 3. YTD Deployer welcome page.

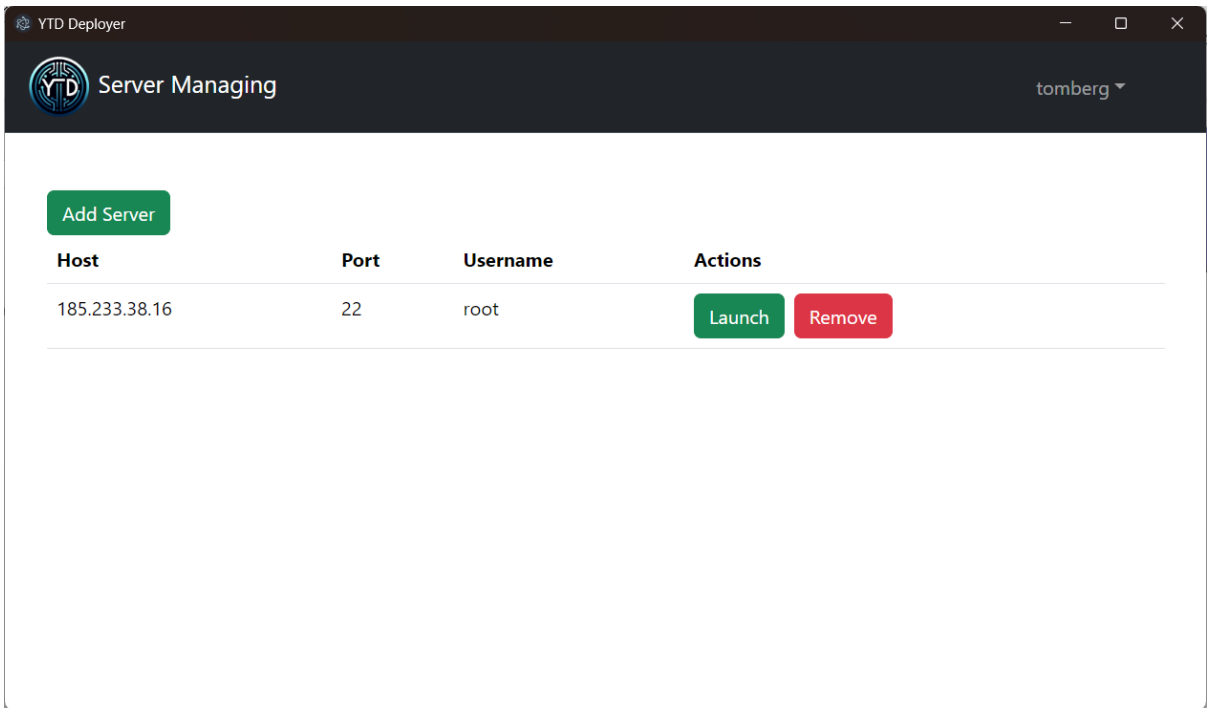

Figure 4. Profile view.

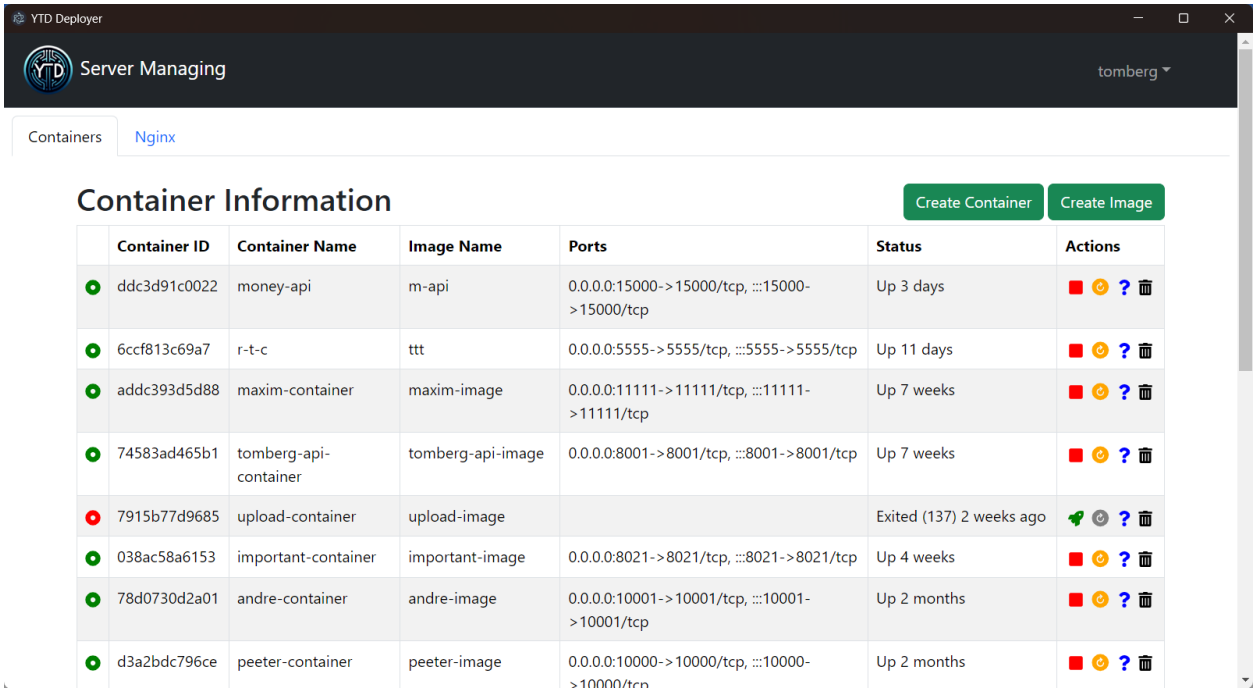

Figure 5. Launched server view.

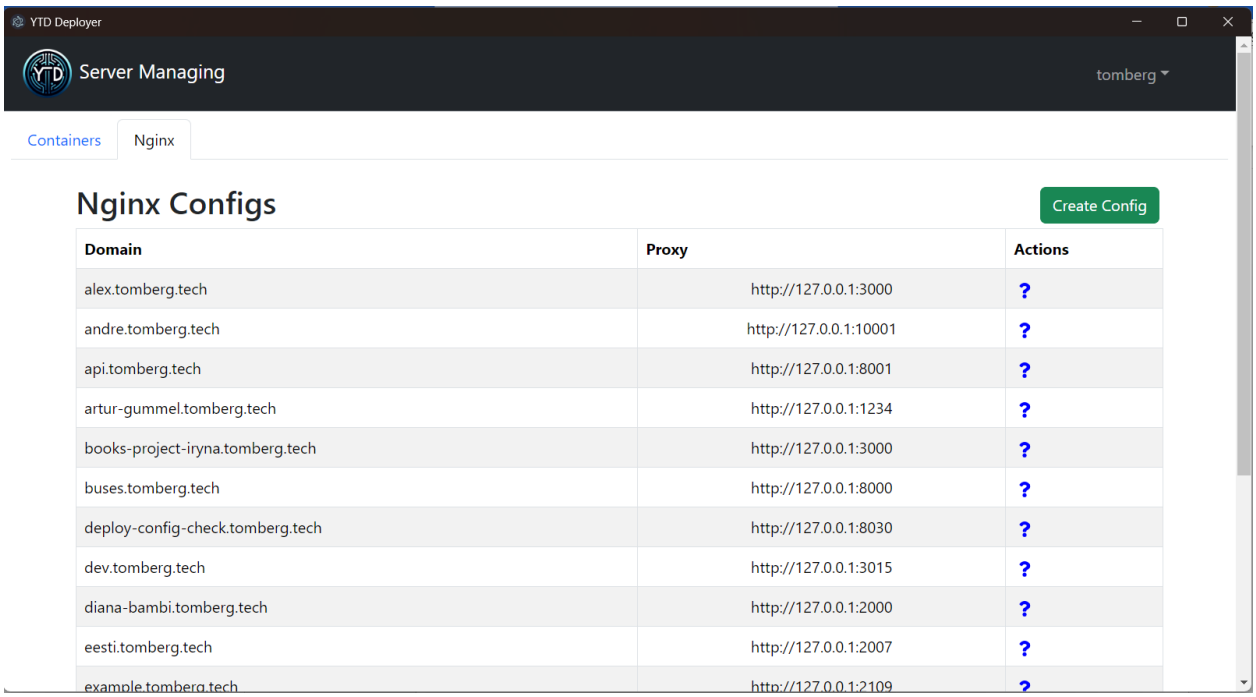

Figure 6. Nginx configuration manager view.

Conclusion

In conclusion, the YTD Deployer application presents a comprehensive and efficient solution for deploying applications on Linux-based servers. By leveraging the power of ElectronJS and React for the frontend, coupled with Spring Boot for the backend, YTD Deployer offers a seamless deployment experience with a user-friendly interface and robust functionality. With YTD Deployer, users can easily manage deployment configurations, monitor server status, and track deployment processes, all within a single intuitive dashboard.

The application's emphasis on performance, reliability, security, and usability ensures that users can deploy applications with confidence, knowing that their deployment processes are efficient, secure, and error-free. Furthermore, YTD Deployer's adherence to industry standards for scalability, accessibility, compatibility, and maintainability ensures that it can meet the evolving needs of users and stakeholders, providing a dependable deployment solution for a wide range of applications and environments.

In essence, YTD Deployer streamlines the deployment process, enhances productivity, and empowers users to efficiently manage their application deployments on Linux servers. Whether for small-scale projects or enterprise-level deployments, YTD Deployer stands as a reliable and indispensable tool for modern software development workflows.

Kokkuvõte

Kokkuvõtteks võib öelda, et rakendus YTD Deployer pakub kõikehõlmavat ja tõhusat lahendust rakenduste juurutamiseks Linuxi-põhistes serverites. Kasutades ElectronJS-i ja Reacti võimsust kasutajaliidese jaoks koos Spring Bootiga taustaprogrammi jaoks, pakub YTD Deployer kasutajasõbraliku liidese ja tugeva funktsionaalsusega sujuvat juurutuskogemust. YTD Deployeriga saavad kasutajad hõlpsalt hallata juurutamise konfiguratsioone, jälgida serveri olekut ja juurutusprotsesse – seda kõike ühel intuitiivsel armatuurlaual.

Rakenduse rõhuasetus jõudlusele, töökindlusele, turvalisusele ja kasutatavusele tagab, et kasutajad saavad rakendusi enesekindlalt juurutada, teades, et nende juurutusprotsessid on tõhusad, turvalised ja veatud. Lisaks tagab YTD Deployeri järgimine mastaapsuse, juurdepääsetavuse, ühilduvuse ja hooldatavuse valdkonna standarditele, et see suudab vastata kasutajate ja sidusrühmade muutuvatele vajadustele, pakkudes usaldusväärset juurutuslahendust paljude rakenduste ja keskkondade jaoks.

Sisuliselt lihtsustab YTD Deployer juurutusprotsessi, suurendab tootlikkust ja annab kasutajatele võimaluse oma rakenduste juurutusi Linuxi serverites tõhusalt hallata. Olenemata sellest, kas tegemist on väikesemahuliste projektidega või ettevõtte tasemel juurutamisega, on YTD Deployer usaldusväärne ja asendamatu tööriist tänapäevaste tarkvaraarenduse töövoogude jaoks.

References

1. "Electron." Electron. (2024). Available at https://www.electronjs.org/.
2. "HTML Tutorial." W3Schools. (2024). Available at https://www.w3schools.com/html/.
3. "CSS Tutorial." W3Schools. (2024). Available at https://www.w3schools.com/css/.
4. "What is JavaScript?" JavaScript.com. (2024). Available at https://www.javascript.com/.
5. "Chromium." Chromium Projects. (2024). Available at https://www.chromium.org/.
6. "What is Node.js?" Node.js. (2024). Available at https://nodejs.org/.
7. "API." MDN Web Docs. (2024). Available at https://developer.mozilla.org/.
8. "Uniform Resource Locator (URL)." MDN Web Docs. (2024). Available at https://developer.mozilla.org/.
9. "JSON." MDN Web Docs. (2024). Available at https://developer.mozilla.org/.
10. "Version Control Systems." Atlassian. (2024). Available at https://www.atlassian.com/.
11. "Git – Distributed Version Control System." Git. (2024). Available at https://git-scm.com/.
12. "HTTP – Hypertext Transfer Protocol." MDN Web Docs. (2024). Available at https://developer.mozilla.org/.
13. "JWT.io – JSON Web Tokens." JWT.io. (2024). Available at https://jwt.io/.
14. "npm – Node Package Manager." npm. (2024). Available at https://www.npmjs.com/.
15. "Axios – Promise based HTTP client for the browser and node.js." Axios. (2024). Available at https://axios-http.com/.
16. "Bootstrap." Bootstrap. (2024). Available at https://getbootstrap.com/.
17. "React – A JavaScript library for building user interfaces." React. (2024). Available at https://reactjs.org/.
18. "Redux – A Predictable State Container for JS Apps." Redux. (2024). Available at https://redux.js.org/.
19. "SSH – Secure Shell." SSH.com. (2024). Available at https://www.ssh.com/.
20. "Python – Official Website." Python.org. (2024). Available at https://www.python.org/.
21. "NGINX – High Performance Load Balancer, Web Server, & Reverse Proxy." NGINX. (2024). Available at https://www.nginx.com/.
22. "GitHub: Where the world builds software." GitHub. (2024). Available at https://github.com/.
23. "Parser." Techopedia. (2024). Available at https://www.techopedia.com/.

24. "Docker – Build, Ship, and Run Any App, anywhere." Docker. (2024). Available at https://www.docker.com/.
25. "Java Programming Language." Oracle. (2024). Available at https://www.java.com/.
26. "Spring Framework." Spring Framework. (2024). Available at https://spring.io/.
27. "ManageEngine – IT Management Software." ManageEngine. (2024). Available at https://www.manageengine.com/.
28. "Chef – Infrastructure Automation for DevOps." Chef. (2024). Available at https://www.chef.io/.
29. "Electron Forge." Electron Forge. (2024). Available at https://www.electronforge.io/.
30. "node-scp." npm. (2024). Available at https://www.npmjs.com/package/node-scp.
31. "ssh2." npm. (2024). Available at https://www.npmjs.com/package/ssh2.
32. "uuid." npm. (2024). Available at https://www.npmjs.com/package/uuid.
33. ManageEngine and other IT management solutions offered by the company. (2024). Available at https://www.zoho.com/.
34. Official documentation for Chef Infra, including guides, tutorials, and reference materials for infrastructure automation. (2024). Available at https://docs.chef.io/infra/.
35. GitHub project. (2024). Available at https://github.com/yevheniitomberg/container-manager.