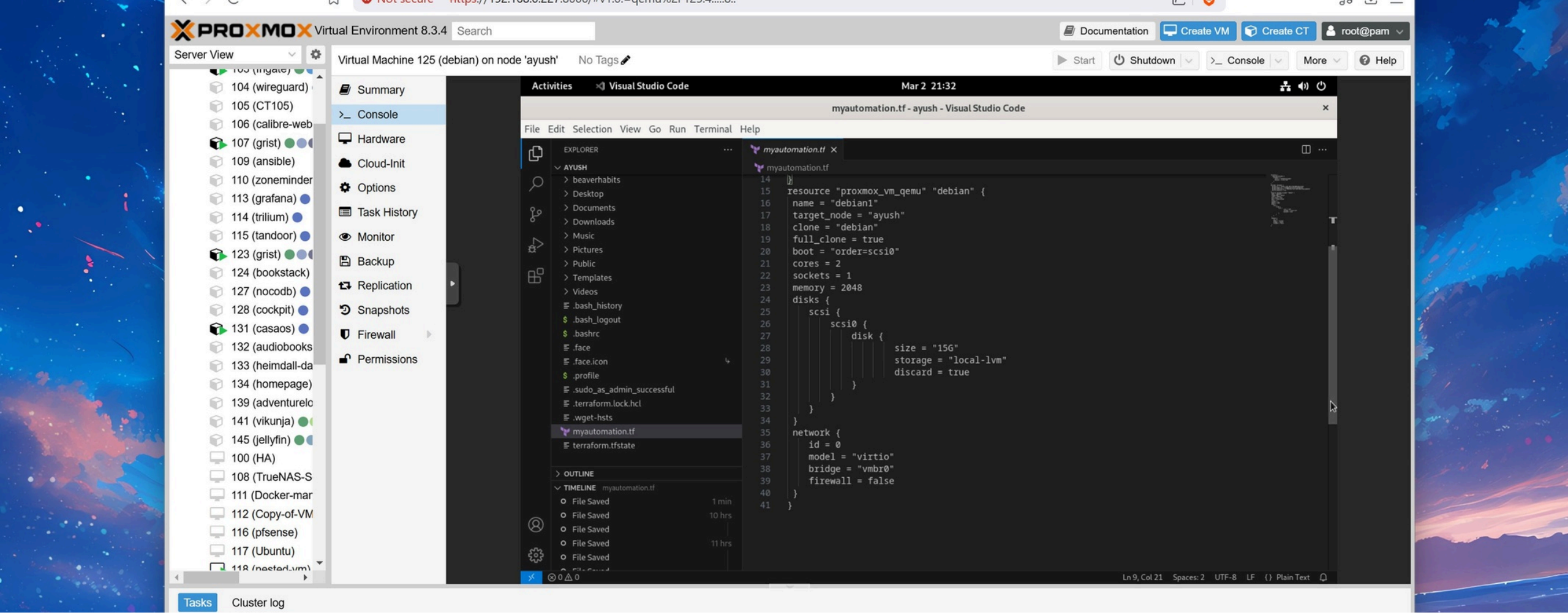


Automating Virtual Machine Provisioning with Natural Language and Proxmox

Ashwath Ramanathan Rajamanthiram

Master's in Digital Transformation & Innovation at the University of Ottawa.

Ashwath Ramanathan

Master's Student – Digital Transformation & Innovation

University of Ottawa

I have hands-on experience working with virtualized infrastructure from my time at GE Healthcare, where I managed and maintained virtual machine environments. After starting my master's program, I realized how tedious it can be to set up virtual machines manually for academic or personal projects. That's what motivated me to build an automated solution using Proxmox and natural language prompts—making VM provisioning faster, simpler, and more accessible.

Project Motivation

- **Manual VM Provisioning was Time-consuming and Error-prone**

  Creating virtual machines on Proxmox often required navigating complex menu structures, configuring various settings, and manually inputting numerous parameters. This process was cumbersome and prone to human errors, which often led to inconsistent and suboptimal VM configurations.

- **Desire to Improve User Experience and Productivity**

  The motivation was to develop a solution that would simplify the VM provisioning process by allowing users to create VMs using natural language prompts, reducing the cognitive load and improving overall productivity.

- **Explore the Potential of Natural Language Interfaces for Systems Automation**

  The project aimed to investigate how natural language processing (NLP) techniques, such as language models, could be leveraged to enable users to provision VMs more intuitively and efficiently, without the need for deep technical knowledge.

- **Enhance Accessibility and Inclusivity**

  By providing a natural language-based interface, the solution aimed to make VM provisioning more accessible to a wider range of users, including those with limited technical expertise or disabilities, thereby promoting inclusivity in the use of virtualization technologies.

Solution Overview

The user initiates the process by providing a natural language request, such as "Create an Ubuntu 22.04 VM with 4GB RAM and Docker."

The user's prompt is passed to a large language model, such as GPT-3, which processes the natural language input and generates a structured JSON response containing the necessary VM configuration details.

The JSON response from the LLM is parsed and validated to ensure it contains the required information for provisioning the virtual machine.

The parsed JSON data is then used to interact with the Proxmox API, which handles the actual creation and configuration of the virtual machine on the Proxmox platform.

Technical Stack

Python

The primary programming language used for the project, providing the core functionality and integrations.

Flask

A lightweight web framework used to build the application's API and handle user interactions.

Proxmox API

The API provided by Proxmox, the open-source virtualization platform, used to interact with and manage virtual machines.

Mock Layer

A component that simulates the Proxmox API, allowing for testing and development without a live Proxmox environment.

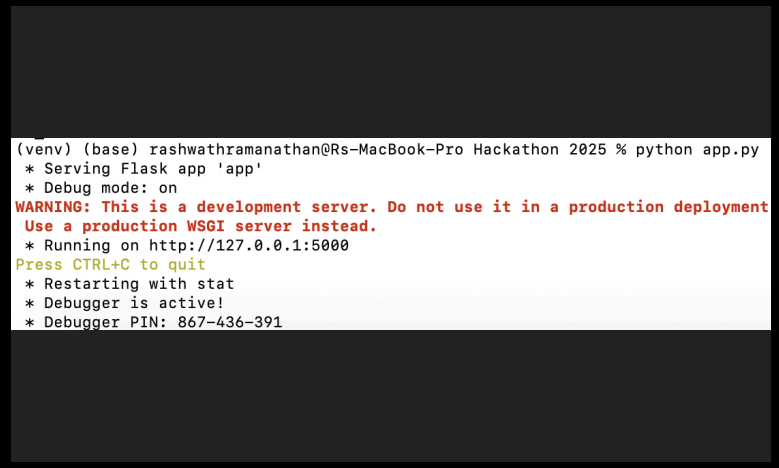
Regex-based JSON Parsing

Utilized regular expressions to parse the output from the language model and extract the necessary information to provision VMs.

HF API

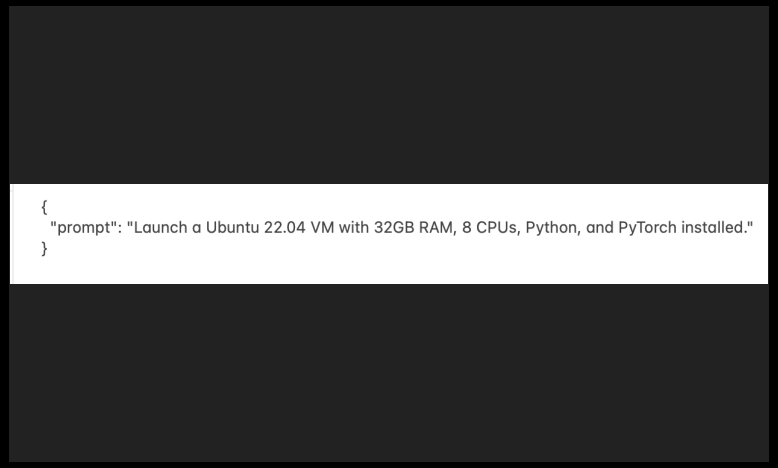
Leveraged the Hugging Face API to access and utilize pre-trained language models for natural language processing.

Demo Scenarios



Flask Server Running the Automation Backend

This image shows the Flask application actively running in debug mode on localhost. The backend listens for incoming prompts and triggers the virtual machine provisioning logic.



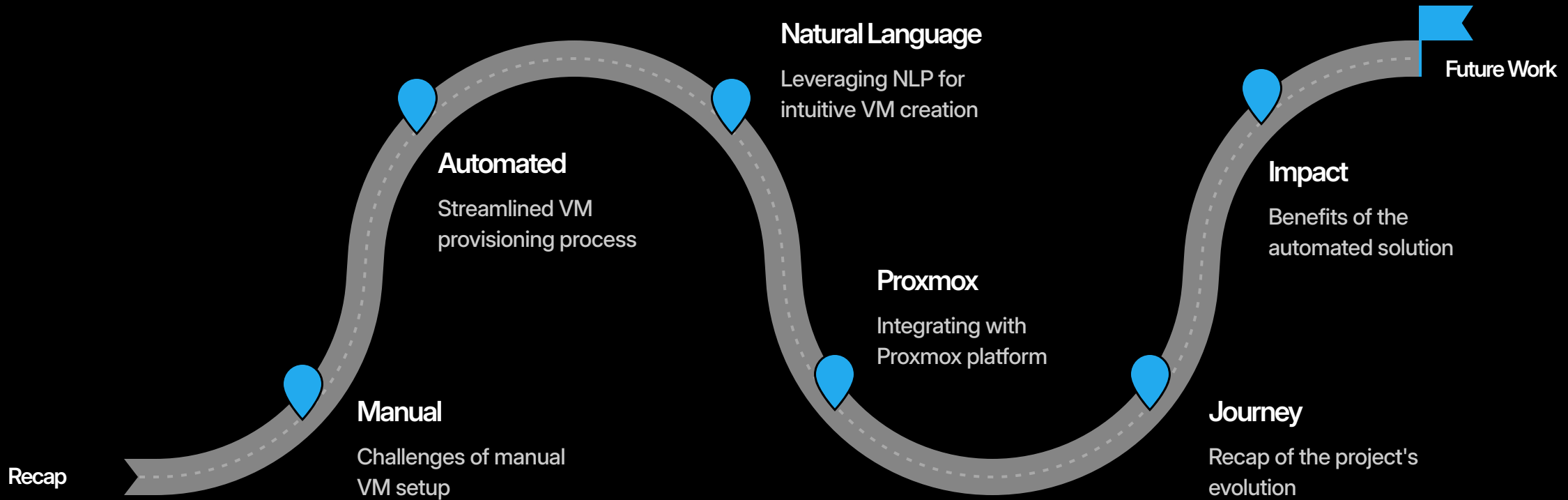
Natural Language VM Request

A simple user prompt — “Launch a Ubuntu 22.04 VM with 32GB RAM, 8 CPUs, Python, and PyTorch installed.” — is used to trigger the automatic generation of a virtual machine configuration via an LLM backend.

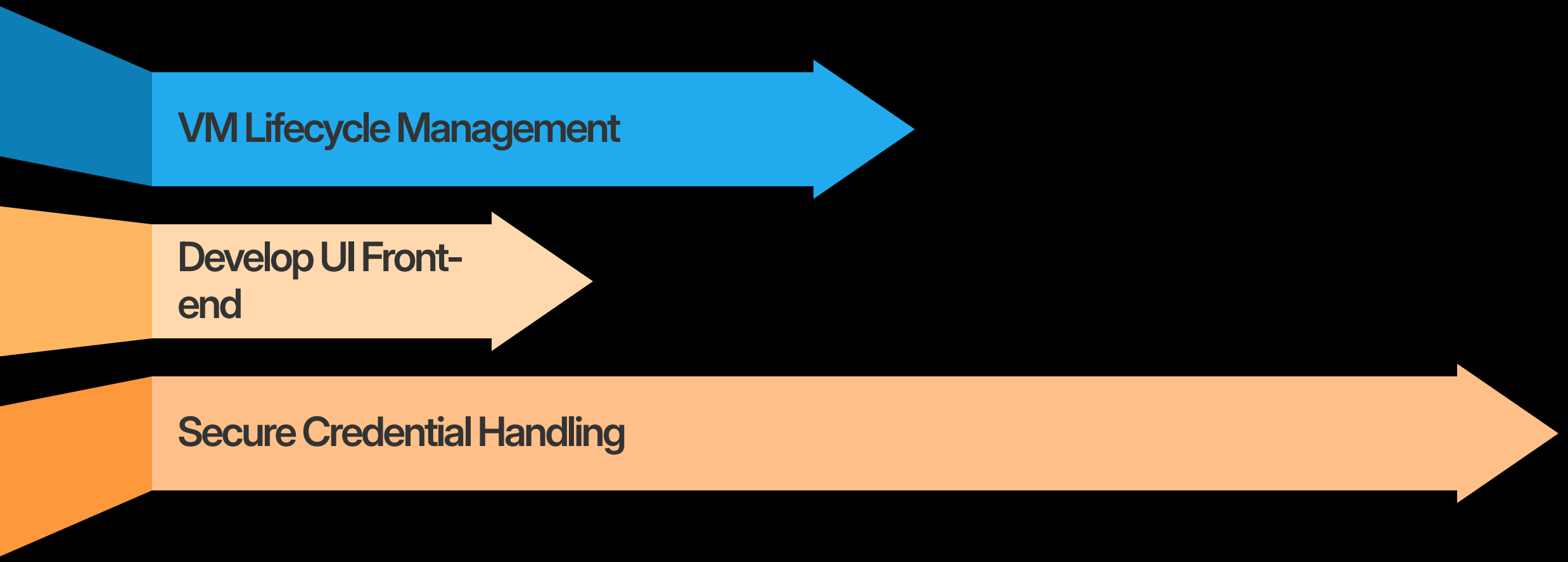
Generated VM Configuration JSON

This JSON was automatically generated in response to a natural language prompt requesting an AI development VM. It defines the VM specs (32GB RAM, 8 vCPUs), installs Python and PyTorch via a startup script, and triggers the provisioning through the API.

Conclusion


Recap
Manual
Challenges of manual VM setup
Automated
Streamlined VM provisioning process
Natural Language
Leveraging NLP for intuitive VM creation
Proxmox
Integrating with Proxmox platform
Journey
Recap of the project's evolution
Impact
Benefits of the automated solution
Future Work

Future Improvements



"Thank You"