OperaTerra E-commerce Platform Infrastructure Deployment

Introduction

OperaTerra is launching a new e-commerce platform that demands a robust, scalable infrastructure on Microsoft Azure. As part of a small DevOps team, I developed this infrastructure with a focus on meeting project requirements and implementing best practices to ensure a seamless deployment process. Throughout development, I encountered several key decisions and challenges that required thoughtful consideration. One of the main choices was selecting the optimal project folder structure, which this report primarily discusses, focusing on the reasoning behind selecting the second structure alternative.

The chosen approach prioritizes scalability and clear organization. To enhance scalability, I implemented a modular design by placing all reusable components within a modules directory. This design enables straightforward integration of new modules without disrupting the existing setup. For managing different environments efficiently without redundant code, I utilized ".tfvars" files for environment-specific configurations, stored within the deployments directory. This arrangement allows for scalable, organized handling of multiple environments.

Maintaining clear configuration for long-term maintainability was also essential. I achieved this by using a single main.tf file within the deployments directory, ensuring a shared structure across environments and allowing infrastructure updates to be made in one location, simplifying maintenance. Additionally, I included a global directory for shared components across environments, which allows for easy updates to specific components based on their usage as global, environment-specific, or modular resources. This separation of concerns enables reusability and isolates changes to specific components like networking, databases, and storage.

The standardized project structure also facilitated CI/CD pipeline implementation. This consistency streamlined the pipeline, as it required referencing different variables by environment to control components. This approach reduced the amount of code and minimized directories compared to the first structure alternative, simplifying the architecture.

Challenges Faced and Solutions

1. Backend Configuration with Dynamic State Files

Challenge: Initially, I attempted to dynamically set the Terraform state file name in the backend configuration using terraform environment . However, Terraform does not allow variables in the backend block.

Solution: I created separate backend configuration files for each environment (e.g., backend.dev.conf, backend.staging.conf, backend.prod.conf). Each backend file specifies a unique state file name (e.g., dev.tfstate, staging.tfstate, prod.tfstate), which I have load at initialization with the -backend-config option.

2. Managing Sensitive Data Securely

Challenge: Handling sensitive information, such as SQL Server credentials, required careful management to ensure security.

Solution: I tried to implement an Azure Key Vault module to securely store and manage sensitive credentials. The Key Vault allows restricted access based on role assignments and access policies, enabling secure storage and retrieval of credentials by other resources as needed.

Conclusion

Ultimately, choosing the second folder structure alternative provided a balanced, efficient framework that facilitated scalability, maintainability, and effective CI/CD implementation. Despite technical challenges, the infrastructure was successfully deployed and met the project’s needs. Following discussions with OperaTerra, any remaining issues could be addressed before final delivery, ensuring a reliable and robust e-commerce platform.