République Tunisienne

Ministère de l'Enseignement Supérieur

et de la Recherche Scientifique

Université de Tunis El Manar

Institut Supérieur d'Informatique d'El Manar

Rapport de Projet de Fin d'Études

Présenté en vue de l'obtention du

Diplôme National d'Ingénieur en Sciences Appliquées et Technologiques

Spécialité : Computer Science and Engineering

Par

Ben Hadj Nasr Mohamed

Orchestrating Critical Application Deployment with Minimal Downtime

Encadrant professionnel : **Yazid Missaoui** Ingénieur R&D

Encadrant académique : **Monsieur/Madame Prénom NOM** Maître Assistant(e)

Réalisé au sein de Adactim

Année Universitaire 2023 - 2024

République Tunisienne

Ministère de l'Enseignement Supérieur

et de la Recherche Scientifique

Université de Tunis El Manar

Institut Supérieur d'Informatique d'El Manar

Rapport de Projet de Fin d'Études

Présenté en vue de l'obtention du

Diplôme National d'Ingénieur en Sciences Appliquées et Technologiques

Spécialité : Computer Science and Engineering

Par

Ben Hadj Nasr Mohamed

Orchestrating Critical Application Deployment with Minimal Downtime

Encadrant professionnel : **Yazid Missaoui** Ingénieur R&D

Encadrant académique : **Monsieur/Madame Prénom NOM** Maître Assistant(e)

Réalisé au sein de Adactim

Année Universitaire 2023 - 2024

I authorize the student to submit his internship report.

Encadrant professionnel, **Yazid Missaoui**

Signature et cachet

I authorize the student to submit his internship report.

Encadrant académique, **Monsieur/Madame Prénom NOM**

Signature

Dédicaces

Je dédie ce travail à :

Monsieur **Yazid Missaoui**, Monsieur **Monsieur/Madame Prénom NOM** pour m'avoir encadré et fait de leurs mieux afin de m'aider.

etc.

Ben Hadj Nasr Mohamed

Remerciements

Je remercie

Je suis reconnaissant

J'exprime ma gratitude

Table des matières

Table des figures

Liste des tableaux

Liste des abréviations

- ➢ **GISI** = Génie Informatique des Systèmes Industriels
- ➢ **GLSI** = **G**énie **L**ogiciel et **S**ystèmes d'**I**nformation
- ➢ **GTR** = Génie des Télécommunications et Réseaux

Introduction générale

Exemple d'utilisation de la bibliographie utilisée [1]. Le style utilisé est IEEE [2].

Une introduction d'une à 3 pages où vous poserez clairement le problème auquel vous allez tenter d'apporter une solution. L'introduction se rédige à la fin de votre travail de rédaction. Avant de rédiger l'introduction, structurez TOUT le PFE. L'introduction peut se faire en même temps que la conclusion.

L'introduction sert trois objectifs :

- elle introduit le sujet. Ceci signifie qu'il faut présenter succinctement le contexte général du travail accompli, par exemple l'environnement professionnel et l'entreprise pour un rapport de stage, puis définir le sujet en termes précis et concis ;
- elle énonce ensuite succinctement les objectifs du travail personnel, et les moyens mis en œuvre pour tenter de les atteindre ;
- elle s'achève sur une présentation claire du plan adopté pour la suite du corps du rapport. L'annonce du plan se fait au futur et doit être rédigée en entier.

L'introduction générale doit développer les points suivants :

- la présentation du contexte du projet (domaine exemple : télécommunication, sécurité, automate etc.) ;
- la présentation brève de l'entreprise d'accueil et de son domaine ;
- la description des objectifs du PFE/ Mémoire : justifier le sujet et poser le problème à résoudre ; indiquer la manière dont il sera traité en terme d'outils et de méthodes ; donner les raisons qui président à ce choix ; exposer les intérêts du sujet et sa problématique ;
- l'annonce du plan du rapport sans trop détailler. Il est recommandé, à partir de l'introduction générale, de recourir au « nous» de modestie.

Chapitre 1

GENERAL FRAMEWORK OF THE PROJECT

Plan

Introduction

Ensuring the smooth launch and ongoing availability of crucial applications is vital for businesses in today's digital landscape. This project outlines a strategic approach to achieve this goal by streamlining deployment processes and minimizing downtime.

We will implement detailed plans for launching each application, including clear timelines, defined responsibilities, and contingency measures to address potential risks and disruptions. This will ultimately enhance the stability and performance of our key applications, contributing to overall business success.

1.1 The Host Organization

ADACTIM is a Managed Services Operator specializing in the Cloud, application integration and outsourcing, ERP and BI, operating internationally via a presence in Europe, the Maghreb and Africa.

this figure 1.1 present the logo of the company.

Figure 1.1 : Logo Entreprise Adactim

Adactim Mission

ADACTIM enables the company to benefit from technological transformations in the areas of IT infrastructure and integrated business systems allowing it to focus its energy on its core business.

Our mission is to facilitate to businesses the access to technological innovations, to simplify their daily use, allowing the company to be more efficient and competitive.

Our company focus its resources on its development and its customers. To that end the company should be well-equipped in business software and IT infrastructure and outsources appropriate operations processes.

1.2 Existing study

Description of the Current Environment :

Our core business application currently runs on a manually provisioned Azure cloud infrastructure managed through the web portal.

Analysis of Existing Deployment Processes :

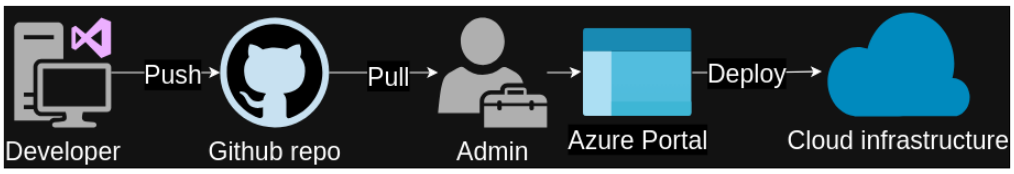

Figure 1.2 : existing strategy of the application deployment

The deployment of applications in our current setup involves several steps, including code preparation, testing, deployment scheduling, and monitoring. Each deployment requires coordination between multiple teams, including developers, quality assurance, and system administrators. Strengths and Weaknesses of the Current Approach :

Strengths :

- ➢ Our current deployment process follows a structured workflow, ensuring thorough testing before releasing applications into production.
- ➢ Effective communication and collaborative problem-solving are facilitated by teamwork during deployments.

Weaknesses :

- ➢ Extended outages caused by inefficient deployment practices negatively impact business continuity and revenue.
- ➢ Limited automation in certain areas leads to manual errors and delays during deployments.

1.3 Problematic

While our current deployment process prioritizes rigorous testing and cross-team collaboration, it faces significant challenges impacting both efficiency and reliability. One critical issue lies in the protracted nature of deployment timelines. This stems from the inherent complexity of coordinating and executing numerous manual testing processes. Consequently, not only are the releases of crucial applications delayed, but the potential for human error during manual interventions is also amplified

In essence our current application deployment process faces a critical challenge : balancing the strengths of its structured workflow and collaborative approach with the need for faster, more automated deployments.

1.4 Needs and requirements

Functional Needs :

- ➢ **Patch Deployment Automation :** Implement a system for automated deployment of patches to applications and systems.
- ➢ **Version Rollback Capability :** Provide the ability to revert to a previous version of an application during deployment in case of failure or unexpected issues.

Non-functional Needs :

- ➢ **Availability :** Ensure high availability of applications and services, minimizing downtime during deployments.
- ➢ **Security :** Implement Web Application Firewall (WAF) and Container Application Firewall (CAF) to safeguard applications and containers from cyber threats.
- ➢ **Performance :** Optimize deployment processes to maintain optimal performance levels of applications and systems.
- ➢ **Cost optimization :** Minimize the costs associated with deployment processes, including recourses and infrastructure.

1.5 Proposed Deployment Optimization Solution

In order to address the identified challenges and meet the outlined needs and requirements, we propose a solution that leverages modern technologies and methodologies.

- ➢ **Provisioning and Configuration Management :** Automate the provisioning and configuration of cloud infrastructure and application environments to ensure consistency and reliability using Infrastructure as Code (IaC) and Configuration Management tools.
- ➢ **Automating the build and testing process :** To minimize manual intervention and expedite development cycles, let's leverage automation across the build and testing pipeline. This not only reduces human error but also frees up valuable time for developers to focus on core tasks.

- ➢ **implementing a Deployment strategy :** Implement a deployment strategy that leverages automation to ensure seamless and efficient application deployments, minimizing downtime and errors. By applicating these pattern principles :
  - — **central secrets store :**storing and monitoring the secrets that our application needs. We will use Azure Key Vault to implement this pattern.
  - — **Rightsize resources for each environment :** ensuring that the resources allocated to each environment are appropriate for the expected load. We can do this by implementing workspaces in terraform.
  - — **Delete non-production environments :** ensuring that non-production environments are deleted when they are no longer needed.
  - —

1.6 Objectives

through out this project we will adopt the scrum methodologie to achieve the following objectives :

Tableau 1.1 : the key technical objectives for the project

the Sprints	Objectives	Delivrables
Sprint1(2 weeks)	Probision the cloud infrastructure as IaC	Terraform scripts
Sprint2(2 weeks)	Implement the Continuous Integration pipeline	a devops workflow
Sprint3(2 weeks)	Set up a deployment strategy	seamless deployment
Sprint4(1 week)	Optimization and cost reduction	cost reduction report
Sprint5(1 week)	Write the necessary documentation	Documentation

Conclusion

This chapter presented the challenges in our current deployment processes and proposed an optimized strategy. This strategy leverages automation, recovery mechanisms, security enhancements, and performance monitoring. It aims to improve efficiency, reliability, and agility in our deployments. The next chapter will go in fu

Chapitre 2

Detailed Existing study

Plan

Introduction

In this chapter we will explain various concepts that are crucial for the subsequent sections of this report. We will also a thorough detailed study of the current environment and the existing architecture and the deployment process.

2.1 Cloud Computing

Definition : Cloud computing is the on-demand availability of computer system resources, especially data storage and computing power, without direct active management by the user. A simple way to describe the Cloud it's multiple data centers available to many users over the Internet.
Cloud computing services are offered by various providers, with Amazon Web Services, Microsoft Azure, and Google Cloud Platform being some of the major players in the field.

Comparative study between cloud providers

So it makes sense to compare the three major cloud providers to see which one is the best for our project.

strengths of the three major cloud providers [2]

- **Amazon Web Services (AWS) :** AWS has had almost a 7-year head-start and vastly more offerings at present than other competitors. With that head start, the available talent pool is larger, meaning that more people know AWS.
- **Microsoft Azure :** Azure provides a pretty compelling transition path to the cloud, also Microsoft has a large number of enterprise customers, and many of them are already using Microsoft products. This makes it easier for them to use Azure.
- **Google Cloud Platform (GCP) :** Google just happens to originated one of the most popular container orchestration systems, that being Kubernetes. and with that it has been able to leverage that reputation to attract customers to its cloud platform.

Weaknesses of the three major cloud providers [2]

- **Amazon Web Services (AWS) :** with it's vast growth Amazon (the company that owns AWS) has become a direct rival to many retailers, and that has led to some customers looking for alternatives.

- ➢ **Microsoft Azure :** For the past decade or so, open-source software has found great acceptance, both on-prem and in the cloud, largely due to organizations seeking alternatives to commercial software vendors like Microsoft.
- ➢ **Google Cloud Platform (GCP) :** Google has a reputation for killing off products that don't meet its expectations, and that has led to some customers being wary of using Google Cloud Platform.

Verdict

since this project is about transitioning to the cloud, and the fact that the company Adactim has a golden partnership with Microsoft, we will be using Microsoft Azure as our cloud provider.

2.2 Cloud Computing Services

Definition : Cloud computing services are a broad set of services that are delivered over the internet. These services are divided into three main categories : Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS).

IaaS

Definition : IaaS provides virtualized computing resources that requires the developer to manage the infrastructure, including the network, servers, and operating systems. and with this comes great flexibility and scalability.

Offered Services : Azure offers a wide range of IaaS services, including virtual machines, storage, and networking. And with this we can simulate the on-premises infrastructure in the cloud with minimal changes.

PaaS

Definition : PaaS provides a platform allowing developers to build, run, and manage applications without the complexity of building and maintaining the infrastructure. This allows developers to focus on the application itself.

Offered Services : Some of the PaaS services offered by Azure include Azure App Service, Azure Functions. and these services have built-in deployment strategys that can be selected.

SaaS

Definition : SaaS is a software distribution model in which applications are hosted by a third-party provider so developers don't have to install, maintain, or update the software.

Offered Services : Azure offers a wide range of SaaS services, including Office 365, Dynamics 365, and many more. But unfortunately, these services are not relevant to us since we cannot use them to host our application.

Verdict

After this brief overview of the cloud computing services, we can conclude that the IaaS and PaaS services are the most relevant to us, so I will implement the deployment plan using both of them and choose the most suitable one.

2.3 Infrastructure Study

The company is currently using the baseline web application architecture provided by Microsoft[3]. This figure 2.1 presents the global architecture of the application.

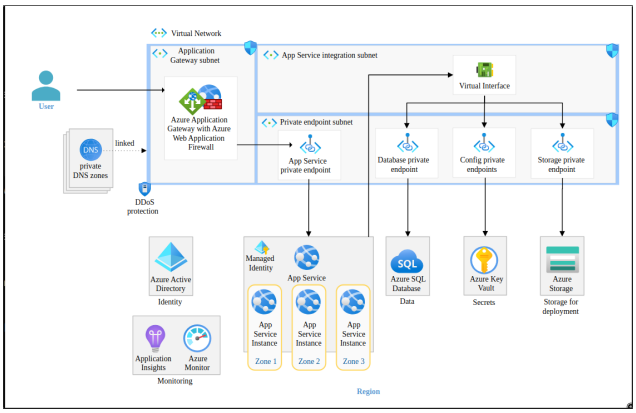

Figure 2.1 : gloabal architecture

Description : The architecture exposes a public endpoint via Azure Application Gateway with Web Application Firewall. The App Service application uses virtual network integration to securely communicate to Azure PaaS services such as Azure Key Vault and Azure SQL Database.

Componants of the architecture

➢ **Virtual Network :** This is the fundamental building block for your private network in Azure. It provides isolation and protection for your resources.

- **App Service :** This service is used to host the web application. It provides a fully managed platform for building, deploying, and scaling web apps.
- **Azure SQL Database :** This service is used to store the application data. It provides a fully managed relational database with built-in high availability and security.
- **Azure Key Vault :** This service is used to store and manage application secrets. It provides a secure and centralized storage for application secrets.
- **Azure Application Gateway :** This service is used to protect the web application from common web vulnerabilities. It provides a web application firewall and other security features.
- **Azure Monitor :** This service is used to monitor the health of the web application. It provides logging and application telemetry to monitor the health of the application.
- **Azure DevOps :** This service is used to automate the deployment of the web application. It provides a set of tools for building, testing, and deploying applications.
- **Virtual Interface :** This service is used to connect the web application to the virtual network. It provides a secure and private connection to the web application.
- **Application Insights :** This service is used to monitor the performance of the web application. It provides real-time monitoring and analytics for the web application.
- **Private DNS Service :** This service is used to resolve the DNS names of the Azure PaaS services. It provides a secure and private DNS resolution for the web application.
- **Private endpoint :** This service is used to connect the web application to the Azure PaaS services. It provides a secure and private connection to the Azure PaaS services.

Network flows

Inbound flow :

- The user issues a request to the Application Gateway public IP.
- The WAF rules are evaluated.
- The request is routed to an App Service instance through the private endpoint.

App Service to Azure PaaS services flow :

- App Service makes a request to the DNS name of the required Azure service. The request could be to Azure Key Vault to get a secret, Azure SQL Database.

➢ The request is routed to the service through the private endpoint.

Deploying to the app service :

The deployment process is initiated from Azure portal by the admin from his machine.

Architecture characteristics

➢ For security reasons, the network in this architecture has separate subnets for the Application Gateway, App Service integration components, and private endpoints. Each subnet has a network security group that limits both inbound and outbound traffic for those subnets to just what is required.

➢ The App Service baseline configures authentication and authorization for user identities (users) and workload identities (Azure resources) and implements the principle of least privilege.

➢ Azure Monitor collect and analyze metrics and logs from your application code, infrastructure (runtime), and the platform (Azure resources).

2.4 Description of the tools used

Terraform

Figure 2.2 : Terraform

Definition : Imagine building your software on a foundation pre-designed with specific instructions, rather than individually placing each brick. This is the essence of Terraform, an open-source IaC tool. It allows you to define the infrastructure your application needs using a simple language, similar to writing instructions. This simplifies managing resources across different environments (cloud-based or on-premises) with consistent configurations, ensuring everything is built according to your specifications.

Alternatives :

➢ **Azure Resource Manager (ARM) templates :** these templates are native to Azure, offering familiarity and direct management within the platform. However, they require more technical knowledge and lack the flexibility and readebility of IaC tools like Terraform.

➢ **Bicep** Think of Bicep as a specialized architect fluent in Azure, Microsoft's cloud platform. It speaks Azure's language directly, making it easier to design and manage resources within that specific environment. However,since its expertise is limited to Azure it does not have the community support offered by an open-source project like terraform.

By understanding these factors, we can make the informed decision that the IaC tool that best suits our requirement is Terraform.

Azure DevOps

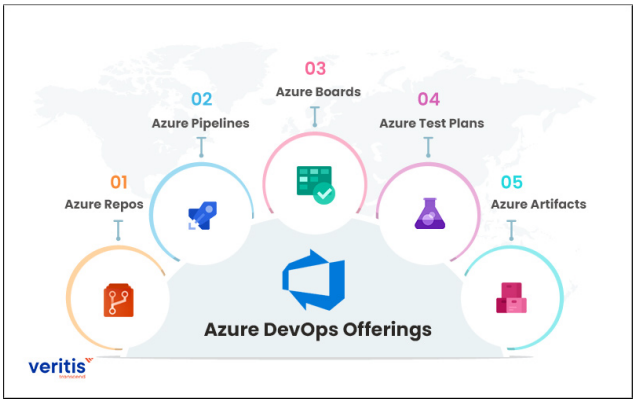

FIGURE 2.3 : Azure DevOps

Definition : This platform acts as a comprehensive toolkit for software teams, offering various features to manage the entire development lifecycle efficiently.

Features :

➢ **Azure Repos :** This feature keeps your code organized and secure, just like a well-structured library holding all your project versions. It can use either Git or Team Foundation Version Control (TFVC) to manage your code.

➢ **Azure Pipelines :** This service automates tasks like compiling code, running tests, and deploying new versions, saving time and minimizing errors.

➢ **Azure Boards :** Planning and tracking progress becomes transparent with this feature. It provides Kanban boards visually displaying tasks, backlogs listing upcoming work, and sprint planning tools.

➢ **Azure Artifacts :** Sharing reusable components becomes effortless with this feature. Think of it as a shared storage space for code modules, containerized applications, and other resources your team can easily access and reuse across projects.

Conclusion

This chapter has comprehensively analyzed the current state of our cloud environment, infrastructure, and deployment process. We compared various cloud service models, ultimately selecting Microsoft Azure due to its alignment with our company's existing partnership and strategic goals. Within Azure, we identified Infrastructure as a Service (IaaS) and Platform as a Service (PaaS) as the most suitable offerings for our application's needs.

Our in-depth examination of the existing infrastructure revealed a secure and well-structured architecture leveraging Azure's robust security features. Moving forward, we will delve deeper into Terraform and Azure DevOps, exploring their functionalities for enhanced infrastructure management and automated deployment processes. This comprehensive analysis establishes a solid foundation for transitioning to a secure, efficient, and automated cloud-based deployment framework.

Chapitre 3

The provisioning of the infrastructure

Plan

Introduction

The first sprint of this project focused on provisioning the necessary cloud resources and setting the connections between them. In this chapter, we will present how we customized the architecture to our needs and the challenges we faced during the process, and we will also showcase the features implemented in the terraform modules.

3.1 Activities Completed

- **Azure account setup :** We created an Azure account using the student package offer.
- **Azure CLI installation :** We installed the Azure CLI to manage our Azure resources.
- **created the necessary modules :** For a better organization of our code, we separated the resources into modules : the web app module, the database module and the storage account module.
- **established the network configurations :** We provisioned the virtual network and the DNS zones for each module and added the necessary links.
- **set up the different workspaces :** We created different workspaces for each environment (dev, QA, prod) and set the default values for each environment.
- **tested the architecture :** We tested the architecture by deploying the provisioned resources and deploying a simple web application.
- **wrote the documentation :** We wrote the documentation explaining how to modify the different variables and how to use the different workspaces(I auto-generated the documentation for the modules to ensure consistency using terraform-docs).

3.2 The modified architecture :

the slight modification made to the original architecture consisted of the removal of the application gateway due to it's high price so the user will just have to access the web app directly which is acceptable since we only have one web app.
the new simplified architecture is shown in figure 3.1.

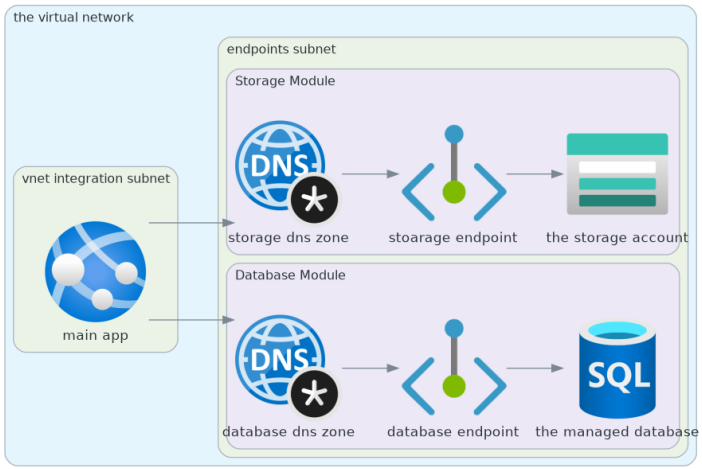

Figure 3.1 : The new architecture

3.3 Delivrables :

the deliverables for this sprint are the terraform configuration that adapts to three different workspaces :

- **dev :** the development environment :
  - the resources are the most basic cost to optimize the cost.
  - It also has a VM with a public IP inside the virtual network to access the database and the storage account.
  - the web app and the VM can only be accessed from the CIDN given in the variables.
- **QA :** the quality assurance environment :
  - the resources are a bit more expensive to better simulate the production environment.
  - It also has a VM with a public IP inside the virtual network to access the database and the storage account.
  - it is open to the public.
- **prod :** the production environment.

— the resources are the most expensive to ensure the best performance.

— it is open to the public.

the file structure :

this figure 3.2 shows the file structure of the terraform configuration.

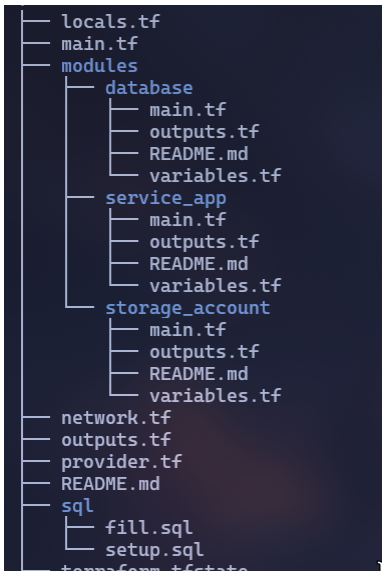

Figure 3.2 : The file structure

To promote code reusability, maintainability, and clarity, we adopted a modular approach for our Terraform configuration. This involved structuring the code into distinct modules, each encapsulating a specific aspect of the infrastructure (e.g., web app, database, storage account). This approach allows us to manage and update each module independently, and to reuse them across different environments.

Furthermore, to ensure consistency and streamline configuration changes, we grouped frequently used variables like SKU names, VNet prefixes, and subnet prefixes into a dedicated local.tf file.

This combined approach of modular structure and centralized variable management promotes efficient and well-organized Terraform configuration, fostering long-term project maintainability and scalability.

3.4 Challenges :

the problem :

The main challenge we faced during Sprint 1 was configuring the DNS zones to enable communication between the web application and the database. Despite successfully provisioning the necessary cloud resources, the web application encountered issues resolving the hostname of the database. This meant the application couldn't locate the database to retrieve or store data, hindering core functionality. Troubleshooting involved verifying several aspects :

- **DNS record configuration :** We double-checked the DNS record types (likely A record) and their values (database hostname and IP address) within the configured DNS zone. Any typos or incorrect mappings could have caused resolution issues.
- **Network security group (NSG) rules :** We had to verify the NSG rules to ensure the web application could communicate with the database.
- **verify connections :** We had to verify that the database was accessible from the web application. and that the web could access the DNS server.

By systematically examining these potential causes, we were able to identify that the web app did not have access to the DNS server provided by Azure. This experience highlights the importance of careful configuration and understanding of how DNS plays a crucial role in enabling communication between different components within a cloud infrastructure.

the solution :

To address the DNS resolution issue between the web application and the database, we capitalized on a core Azure concept : **the Wire Server**. This managed DNS server, automatically created within each virtual network, plays a critical role in resolving internal DNS queries using private DNS zones. Although offered as a free service, the Wire Server isn't automatically configured as the default DNS server for the virtual network.

Our solution involved leveraging Terraform to explicitly set the Wire Server's static IP address (168.63.129.16) as the default DNS server within our virtual network configuration. By making this configuration change, we ensured that the web application could effectively utilize the Wire Server to resolve the database hostname and establish the necessary communication channel. This approach eliminated the initial DNS resolution obstacle and facilitated seamless communication between the

application and the database.

```
resource "azurerm_virtual_network" "the_network" {
  name                = module.naming.virtual_network.name
  location            = local.location
  resource_group_name = azurerm_resource_group.the_group.nam
  address_space       = [local.vnet_prefixe]
  dns_servers         = ["168.63.129.16"]

  tags = {
    environment = "${terraform.workspace}"
  }
}
```

Figure 3.3 : The DNS configuration

3.5 Conclusion

This chapter detailed the successful completion of Sprint 1, focusing on provisioning the cloud infrastructure for our project. We presented a modified architecture that removed the application gateway due to cost considerations. The new architecture leverages Terraform modules for efficient code organization and utilizes workspaces to manage different environments (dev, QA, prod) with appropriate resource configurations.

We encountered a challenge during this sprint related to DNS configuration, where the web application struggled to resolve the database hostname. This issue was resolved by leveraging the Azure Wire Server as the default DNS server within the virtual network configuration.

By successfully provisioning the infrastructure and addressing the DNS resolution challenge, Sprint 1 laid the foundation for future sprints to focus on building and deploying the web application and its functionalities.

Conclusion générale

Rappel du contexte et de la problématique.
Brève récapitulation du travail réalisé et de la soultion proposée.

La taille de la conclusion doit être réduite, une page de texte tout au plus. Il est important de souligner que la conclusion ne comporte pas de nouveaux résultats ni de nouvelles interprétations.

Le plus souvent, la conclusion comporte :

- un résumé très rapide du corps du texte ;
- un rappel des objectifs du projet ;
- un bilan professionnel qui indique clairement les objectifs annoncés dans l'introduction et en particulier ceux qui n'ont pu être atteints. Il présente et synthétise les conclusions partielles ;
- un bilan personnel qui décrit les principales leçons que vous tirez de cette expérience sur le plan humain ;
- les limites et les perspectives du travail effectué.

Bibliographie

[1] M. Aleksic. « Containerization vs. Virtualization : understand the differences. » [16-02-2024], ubuntu. (18 January 2023), adresse : https://ubuntu.com/blog/containerization-vs-virtualization.

[2] S. Pletcher. « AWS vs Azure vs GCP : The big 3 cloud providers compared. » [16-02-2024], pluralsight. (8June 2023), adresse : https://www.pluralsight.com/resources/blog/cloud/aws-vs-azure-vs-gcp-the-big-3-cloud-providers-compared.

[3] . « Baseline highly available zone-redundant web application. » [20-02-2024], Microsoft. (-), adresse : https://learn.microsoft.com/en-us/azure/architecture/web-apps/app-service/architectures/baseline-zone-redundant.

Annexes

Annexe 1. Exemple d'annexe

Les chapitres doivent présenter l'essentiel du travail. Certaines informations-trop détaillées ou constituant un complément d'information pour toute personne qui désire mieux comprendre ou refaire une expérience décrite dans le document- peuvent être mises au niveau des annexes. Les annexes, **placées après la bibliographie**, doivent donc être numérotées avec des titres (Annexe1, Annexe2, etc.).

Le tableau annexe 1.1 présente un exemple d'un tableau dans l'annexe.

Tableau annexe 1.1 : Exemple tableau dans l'annexe

0	0
1	1
2	2
3	3
4	4

Annexe 2. Entreprise

La figure annexe 2.1 présente le logo entreprise.

Figure annexe 2.1 : Logo d'entreprise

Abstract

Put the English abstract here, put the English abstract here, put the English abstract here, put the English abstract here, put the English abstract here, put the English abstract here... Please don't exceed ten lines, Please don't exceed ten lines, Please don't exceed ten lines, Please don't exceed ten lines. Put the English abstract here, please don't exceed ten lines. Put the English abstract here, please don't exceed ten lines. Put the English abstract here, please don't exceed ten lines. Put the English abstract here, please don't exceed ten lines. Put the English abstract here, please don't exceed ten lines. Put the English abstract here, please don't exceed ten lines. Put the English abstract here, please don't exceed ten lines. Put the English abstract here, please don't exceed ten lines.

Keywords : Please don't use more than five keywords