

COMP7142 | COMP7142001

Popular Network Technology

Documentation Report

**Qualification**

# Document Information

|  |  |
| --- | --- |
| Developer Information | |
| **Developers** | **Client** |
| Stanley Nathanael Wijaya  NW25-1 | BINUS @Alam Sutera, Jl. Jalur Sutera Bar. No.Kav. 21, RT.001/RW.004, Panunggangan, Kec. Pinang, Kota Tangerang, Banten 15143 |
| **Development Period** | |
| 17/09/2025 | |

# Provision Infrastructure with Terraform

|  |  |
| --- | --- |
| Create Resource Group, Virtual Network, and AKS Cluster | |
| Tools/Method Used | Terraform, Azure CLI, Visual Studio Code |
| Step-by-Step | **Step by step to provision infrastructure with terraform:**   1. Login to azure: az login 2. Create terraform files: main.tf and other files 3. Initialize terraform by running: terraform init 4. Plan and apply: terraform plan then terraform apply |
| Result |  |
| Description | The Terraform script successfully provisioned a new resource group named linkasa-rg, a virtual network linkasa-vnet with a dedicated subnet aks-subnet, and an AKS cluster named linkasa-aks-cluster. The cluster is configured for high availability across three zones in the Southeast Asia region as required. |

*Note: You may add more table if needed. For every action, you need to separate the report table.*

# Containerize the Application

|  |  |
| --- | --- |
| Create Docker File and Build Application Image | |
| Tools/Method Used | Docker File, Docker Desktop, Visual Studio Code |
| Step-by-Step | **Step by step to containerize the application:**   1. Create Dockerfile: In the root of the application source code directory 2. Docker build: Build the docker image 3. Docker run: To test the image locally |
| Result | ./Backend/Dockerfile    ./Frontend/Dockerfile |
| Description | The application was successfully containerized. A Dockerfile was created to define the build steps, and from it, a portable and isolated Docker image named linkasa-app:v1 was built. This image contains all necessary code and dependencies to run the LinKasa application. |

*Note: You may add more table if needed. For every information, you need to separate the report table.*

# Create an Azure Container Registry (ACR)

|  |  |
| --- | --- |
| Create ACR and Push Container Image | |
| File Location | ./push\_images.ps1 |
| Step-by-Step | **Step by step to create an azure container registry (ACR):**  To create ACR and Push the container image I prefer to use PowerShell script (push\_images.ps1) you can also manually run using terminal or maybe you prefer to use shell script file, but in this project I made the push\_images.ps1 and this script will automate the creation of the ACR, building both Docker images, tagging them, and pushing them to the new registry.  So just run the push\_images.ps1 file to create the ACR |
| Result |  |
| Description | An Azure Container Registry named qualifnw251 for the Linkasa Frontend and Linkasa Backend was successfully provisioned. The containerized application image was pushed to this registry, making it available for secure deployment to the AKS cluster. |

*Note: You may add more table if needed. For every action, you need to separate the report table.*

# Deploying the Application on AKS

|  |  |
| --- | --- |
| Create and Apply Kubernetes Manifests | |
| File Location | ./kubernetes/deployment.yaml, ./kubernetes/postgres.yaml, ./kubernetes/service.yaml |
| Step-by-Step | **Step by step to deploying the application on AKS:**   1. Connect to AKS: Configure the kubectl 2. Integrate AKS with ACR: Grant aks cluster permission to pull images from the ACR. 3. Create Kubernetes Deployment Manifest: Creating the deployment.yaml file 4. Create Kubernetes Service Manifest: Creating the service.yaml file 5. Apply the Manifests: Deploy the application by applying the manifest 6. Verify the Deployment: Check status of the pods, kubectl get pods |
| Result | The application already can be opened: |
| Description | The LinKasa application has been successfully deployed to the Azure Kubernetes Service cluster. The deployment.yaml manifest created the running instances (pods) of the application, and the service.yaml manifest exposed it to the internet via a public IP. |

*Note: You may add more table if needed. For every action, you need to separate the report table.*