Burger Builder Azure Deployment Report

Date: $(date '+%Y-%m-%d')

Executive Summary

The `devops-project2-ih` repository provisions and deploys a production-style three-tier application on Azure. Terraform modules build the network, data, and compute layers; Dockerized React/Vite and Spring Boot services run on Azure App Service behind an Application Gateway; GitHub Actions automates image delivery and infrastructure updates. Monitoring relies on Application Insights with alerting on gateway health, frontend load, and SQL DTU usage. Ansible- and SonarQube-related objectives remain outstanding.

Architecture Overview

\*\*Compute\*\*: Two Azure Linux Web Apps (frontend React/Vite, backend Spring Boot) defined in `TFmain/main.tf` via `Azure/azurerm\_webapp` module.

\*\*Networking\*\*: Virtual network with four subnets (frontend, backend, Application Gateway, SQL) and associated NSGs (defined in `TFmain/providers.tf`).

\*\*Ingestion\*\*: Application Gateway Standard\_v2 (`Azure/azurerm\_app\_gateway/application\_gateway.tf`) routes `/` to frontend and `/api/\*` to backend; backend probes hit `/api/ingredients`.

\*\*Data\*\*: Azure SQL server & database with private endpoint and private DNS zone (`Azure/azurerm\_sql/`).

\*\*Observability\*\*: Application Insights resource (`Azure/azurerm\_application\_insights/application\_insights.tf`) shared by both web apps; alert rules in `TFmain/alerts.tf`.

devops-project2-ih/

├─ Azure/ # Terraform modules (app gateway, webapp, sql, etc.)

├─ TFmain/ # Root Terraform stack (providers, main, alerts)

├─ backend/ # Spring Boot API + Dockerfile

├─ frontend/ # React/Vite app + Dockerfile

├─ .github/workflows/ # CI/CD pipelines (infra, backend, frontend)

└─ README.md, environment.env\*

Networking & Security

Subnets defined in `TFmain/providers.tf:48-83`:

- `frontend-subnet` (`10.0.2.0/24`) delegated to `Microsoft.Web/serverFarms`.

- `backend-subnet` (`10.0.3.0/24`) delegated to `Microsoft.Web/serverFarms`.

- `appgw\_subnet` (`10.0.1.0/24`) dedicated to the Application Gateway.

- `sql-subnet` (`10.0.4.0/24`) with `Microsoft.Sql` service endpoint.

NSGs per subnet with explicit App Gateway inbound allowances (`TFmain/providers.tf:85-142`).

Azure SQL public access avoided by using private endpoint + DNS link; App Service instances have no public IPs and use VNet integration.

Application Gateway is the sole public entry point exposing frontend and backend routes.

Infrastructure Automation (Terraform)

Remote state stored in Azure Storage (`TFmain/providers.tf:9-14`).

Module composition in `TFmain/main.tf` orchestrates resource group, VNet, NSGs, subnets, web apps, SQL, App Gateway, and Application Insights.

Outputs (e.g., hostnames, SQL FQDN, gateway public IP) available via module outputs.

Provisioning steps documented in `README.md` (install CLI tools, set `TF\_VAR\_sql\_admin\_password`, run `terraform init/plan/apply`).

Application Deployment & CI/CD

\*\*Docker\*\*:

- `frontend/Dockerfile` performs Node build stage then serves static assets via Nginx.

- `backend/Dockerfile` builds Maven jar and runs on Temurin 21 JRE.

\*\*GitHub Actions\*\*:

- `front.yml` builds/pushes frontend image, injecting API URL.

- `back.yml` builds/pushes backend image.

- `infra.yml` authenticates with Azure, runs Terraform init/apply with secrets `AZURE\_CREDENTIALS` and `SQL\_ADMIN\_PASSWORD`.

Separate pipelines preserve independent deployability of frontend and backend as required.

Monitoring & Alerting

Application Insights shared connection string provided to both apps (`TFmain/main.tf:23-55`).

Alert rules (`TFmain/alerts.tf`):

1. `appgw-backend-health-alert` – App Gateway unhealthy host count > 0 for 5 minutes.

2. `fe-requests-alert` – Frontend requests total > 10 within one minute.

3. `sql-dtu-alert` – Azure SQL DTU consumption percent > 80% averaged over 5 minutes.

Terraform now outputs App Gateway ID and SQL database ID to scope these alerts.

README instructs verifying alerts via Azure Monitor and suggests manual smoke tests.

Validation Guidance

Frontend URL: `https://fe-project2-aalhatlan.azurewebsites.net/`

Backend health: `https://be-project2-aalhatlan.azurewebsites.net/api/health`

Sample curl:

curl -s https://be-project2-aalhatlan.azurewebsites.net/api/ingredients | jq '.[0]'

README outlines browser and API tests to confirm end-to-end functionality, including Application Gateway routing and SQL-backed data access.

Gap Analysis Against Assignment Objectives

| Requirement | Status | Notes |

| --- | --- | --- |

| Design 3-tier Azure architecture with private networking | ✅ | VNet, subnets, delegated services established via Terraform modules. |

| Azure SQL with private endpoint & public network disabled | ✅ | Private endpoint + DNS configured; public network access implicitly disabled. |

| Deploy React/Vite frontend & Java backend separately | ✅ | Independent Dockerfiles and App Service deployments. |

| Application Gateway as sole public entry (`/`, `/api/\*`) | ✅ | Implemented in `Azure/azurerm\_app\_gateway/application\_gateway.tf`. |

| Monitoring (App Insights, alerts) | ✅ | AI resource + 3 alerts defined. |

| Terraform automation (remote state, modules) | ✅ | Remote backend & modular structure. |

| GitHub Actions for deploy & infra | ✅ | `front.yml`, `back.yml`, `infra.yml`. |

| SonarQube + Ansible automation | ❌ | No Ansible content or SonarQube provisioning present. |

| Additional required alerts (CPU) | ⚪ | Only gateway health, frontend requests, SQL DTU implemented; CPU alert pending. |

| Architecture diagram & screenshots | ❌ | Not included in repo. |

| Functional validation evidence | ⚪ | README explains steps; screenshots/results not committed. |

Recommended Next Steps

1. Implement Ansible automation (e.g., `config/ansible/`) to deploy and configure SonarQube VM.

2. Add CPU utilization alert for web apps or future VMSS to meet assignment requirement.

3. Generate architecture diagram and monitoring dashboards, commit under `docs/`.

4. Capture validation artifacts (screenshots of running app, Azure resource view, alert history).

5. Explicitly set `public\_network\_access\_enabled = false` for Azure SQL in Terraform to make the posture unambiguous.

---

Prepared by: Codex Assistant