**High-Level Design Document**

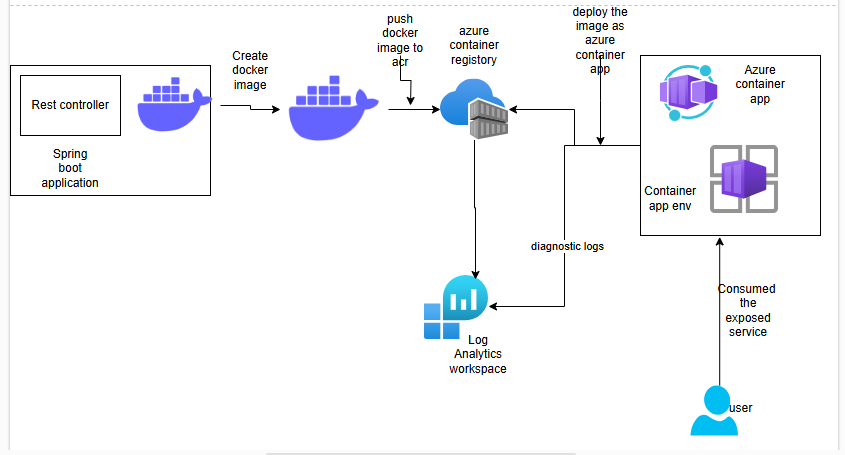
**1. Introduction**

This document outlines the architecture and deployment process for a Spring Boot application on Azure infrastructure. It includes descriptions of each component used in the deployment and the workflow of building, configuring, and deploying the application.

**2. Architecture Overview**

The architecture leverages Azure services for containerized application hosting and automates deployment with Azure DevOps pipelines. Key components include Docker, Azure Container Registry (ACR), Azure Container Apps, and Infrastructure as Code (IaC) tools like Bicep and Terraform.

**3. Architecture Diagram**

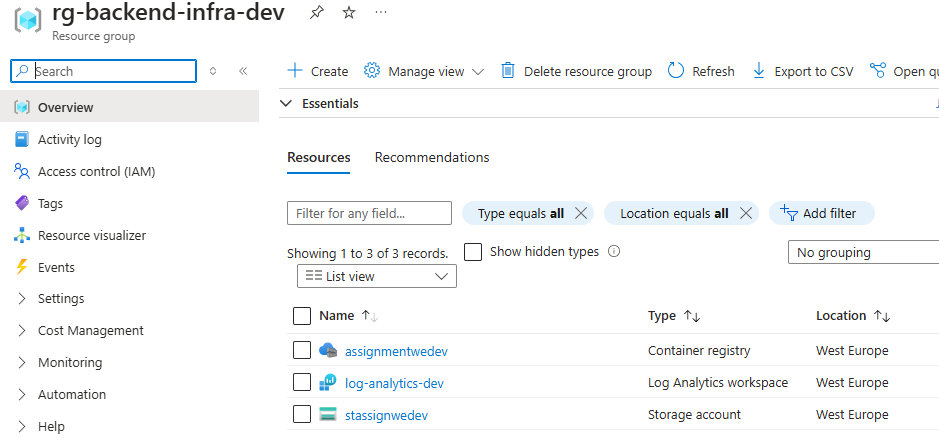
**

**4. Component Descriptions**

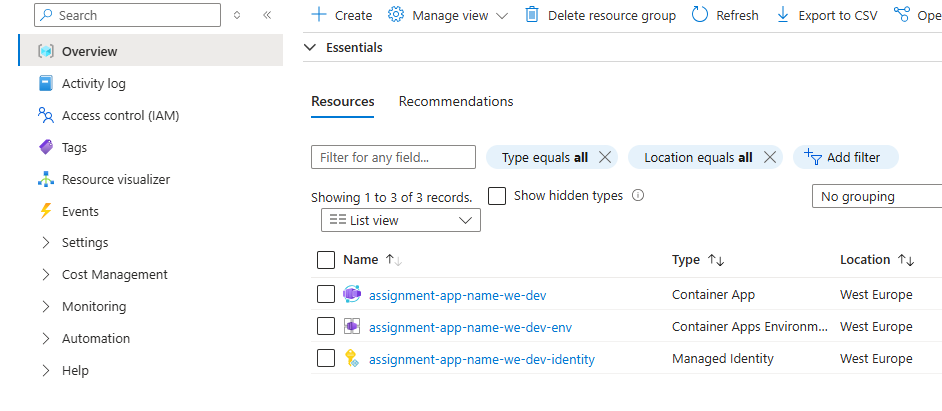
1. **Spring Boot Application**:
   * A Java-based application framework used to build and deploy the core application.
   * Packaged as a Docker image for containerized deployment.
2. **Docker**:
   * Docker is used to containerize the Spring Boot application, encapsulating its dependencies and environment configurations.
   * The Docker image is created from a Dockerfile, enabling reproducible builds across different environments.
3. **Azure Container Registry (ACR)**:
   * ACR stores the Docker images that will be deployed to Azure.
   * The Docker image is tagged and pushed to ACR from which it is deployed to Azure Container Apps.
4. **Azure Container Apps**:
   * A serverless container hosting service on Azure, used to deploy and scale the Dockerized application.
   * Provides an accessible URL for end-users to access the application post-deployment.
5. **Azure Bicep**:
   * A declarative IaC (Infrastructure as Code) language used to define and deploy infrastructure resources.
   * assignment-backend-infra-pipeline.yaml uses Bicep to create the storage account for Terraform state files, ACR, and Log Analytics.
6. **Log Analytics**:
   * A monitoring service on Azure used for collecting and analyzing application logs and metrics.
   * Integrated with Azure resources to provide insights into application health and performance.
7. **Azure DevOps Pipelines**:
   * **Assignment Backend Infrastructure Pipeline** (assignment-backend-infra-pipeline.yaml):
     + Automates the creation of prerequisite resources such as Storage Account, ACR, and Log Analytics.
     + Configures the environment for application deployment using Terraform state management.
   * **Assignment Infrastructure Pipeline** (assignment-infra-pipeline.yaml):
     + Deploys Azure Container Apps across Dev, Test, and Prod environments, applying IaC principles.
   * **Deployment Pipeline** (deploy-app.yaml):
     + Builds, publishes, and deploys the Docker image to the specified ACR and deploys it to Azure Container Apps.
8. **Service Connections**:
   * Secure connections in Azure DevOps, configured for Dev, Test, and Prod environments.
   * Utilize Azure App Registration credentials to authenticate and authorize the pipelines.
9. **App Registration and Service Principal**:
   * App Registration is configured in Azure Active Directory to allow the pipelines to interact securely with Azure resources.
   * The Service Principal provides role-based access control, securing the connection to ACR and other Azure services.

**5. Deployment Workflow**

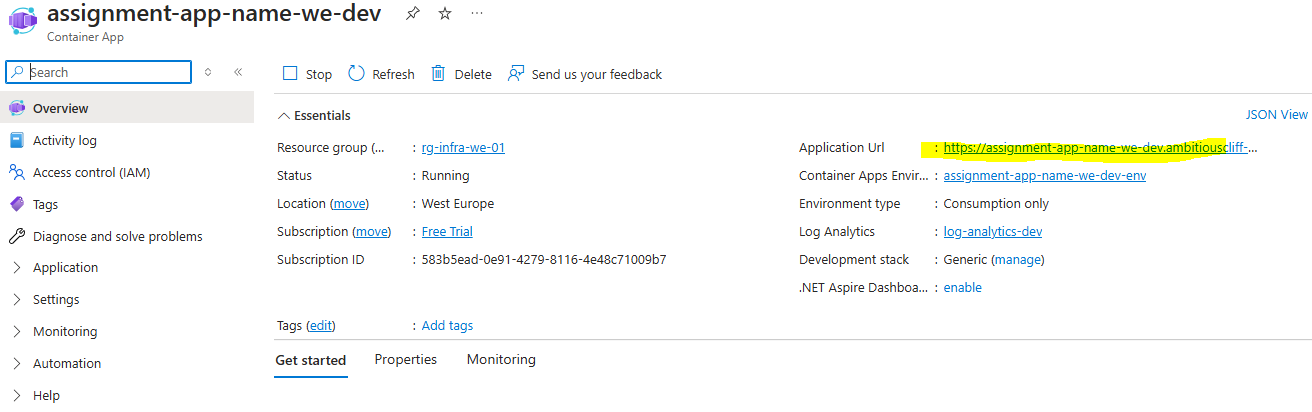
1. **Preparation of Docker Image**:
   * Create a Dockerfile from the Spring Boot application source code.
   * Build and push the Docker image to the ACR.
2. **Infrastructure Setup**:
   * Run assignment-backend-infra-pipeline.yaml to deploy initial resources:
     + **Storage Account**: Holds the Terraform state file.
     + **Azure Container Registry**: Stores Docker images.
     + **Log Analytics**: Used for logging and monitoring.



1. **Application Infrastructure Deployment**:
   * Execute assignment-infra-pipeline.yaml, deploying Azure Container Apps across Dev, Test, and Prod environments using Terraform for IaC.



1. **Application Deployment**:
   * Run deploy-app.yaml:
     + Builds and publishes the Docker image to ACR.
     + Deploys the Docker image to Azure Container Apps.
2. **Application Access**:
   * After deployment, access the application via the URL provided by Azure Container Apps. The login page will be displayed.



**6. Configuration Files**

* **Dockerfile**: Defines the environment for the Spring Boot application.
* **Environment YAMLs** (dev.yaml, test.yaml, prod.yaml): Each environment-specific file stores the connection and configuration details.
* **Terraform & Bicep Templates**: Provision and configure Azure resources.

**7. Security and Access Control**

* **Azure App Registration**: Grants secure access to resources within the Azure environment.
* **Service Principal Permissions**: Restricts access to necessary Azure resources, securing the application.
* **Environment Isolation**: Separate Dev, Test, and Prod environments to prevent unauthorized access and ensure secure deployment across all stages.

**8. Conclusion**

This high-level design document provides a structured approach to deploying a Spring Boot application using containerization and Azure DevOps pipelines. The described architecture ensures secure, scalable, and maintainable deployments on Azure.