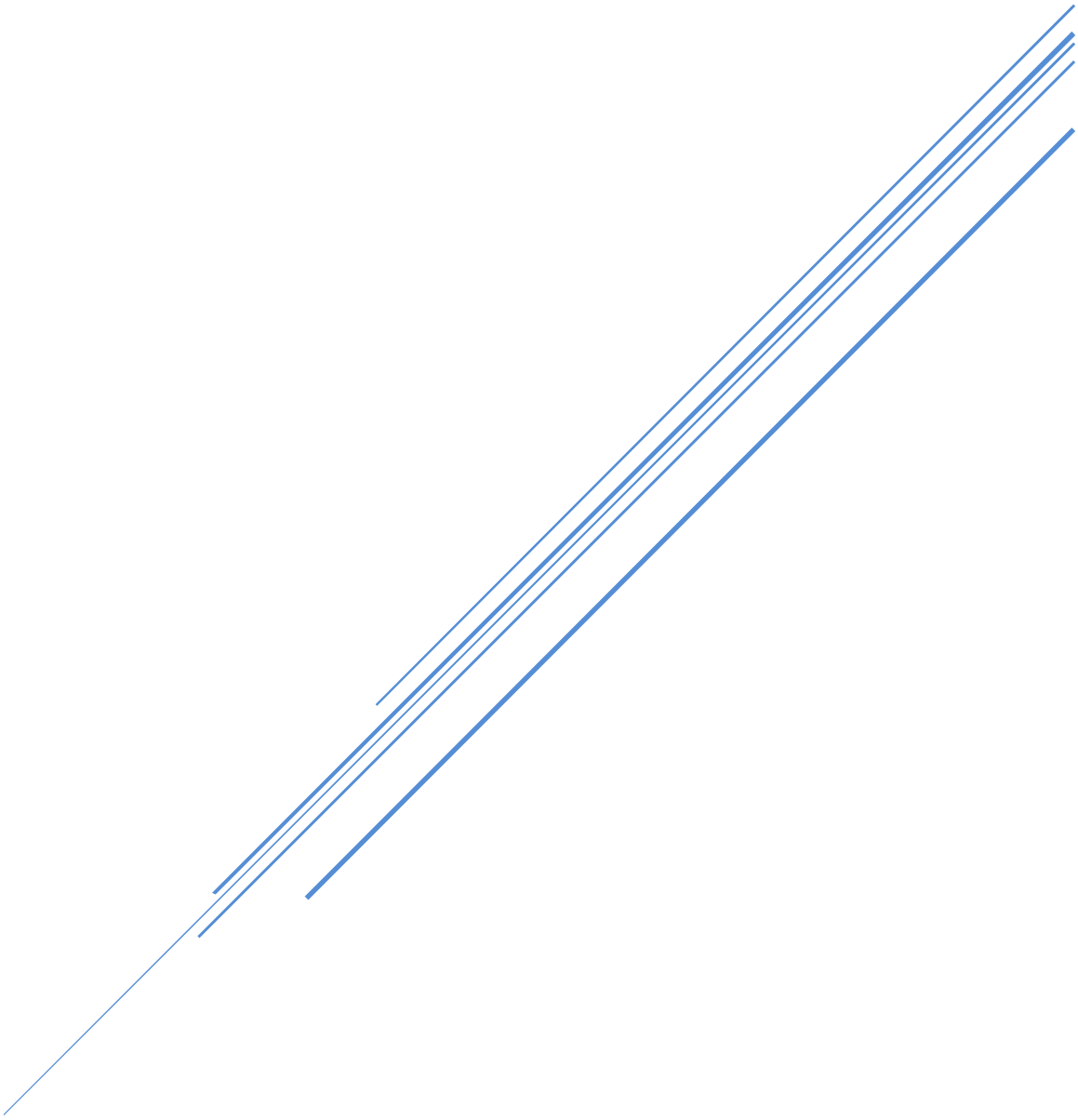
AZDEVOPS-AKS-PROJECT

The GitOps-Way



Bilal Chabbi

Inhalt

Introduction and Goals

- Underlying business goals: The software architects and development team should align their work with the business goals of the organization. In this case, the goals may include improving the agility and scalability of the application deployment process, reducing operational costs, and increasing the reliability of the deployed applications.
- Essential features: The software architects and development team should consider the essential features that the AKS deployment process should provide, such as automatic scaling of the AKS cluster, automatic deployment of new versions of the application, and integration with monitoring and logging systems.
- Essential functional requirements: The software architects and development team should identify and consider the essential functional requirements of the AKS deployment process, such as deployment of containerized applications using Kubernetes manifests, configuration of network policies, and integration with Azure DevOps pipelines.
- Quality goals for the architecture: The software architects and development team should define quality goals for the AKS deployment process, such as reliability, scalability, maintainability, and security. They should also identify the key quality attributes that the architecture should address, such as performance, availability, and fault tolerance.
- Relevant stakeholders and their expectations: The software architects and development team should identify the relevant stakeholders, such as developers, operations teams, security teams, and business stakeholders. They should also understand their expectations regarding the AKS deployment process, such as ease of use, scalability, security, and compliance.

Overall, by considering these requirements and driving forces, software architects and development teams can ensure that the AKS deployment process meets the needs of the organization and its stakeholders while achieving the desired quality goals.

Requirements Overview

Functional requirements:

- Automatic deployment of containerized applications on Azure Kubernetes Service (AKS)
- Automatic scaling of the AKS cluster based on application load.
- Integration with Azure DevOps pipelines for continuous integration and deployment (CI/CD)
- Configuration of network policies for secure application communication
- Integration with monitoring and logging systems for visibility and troubleshooting
- Ability to roll back to previous versions of the application in case of errors or issues.

- Support for different application architectures, such as microservices and monoliths

Driving forces:

- Need to improve the agility and scalability of the application deployment process.
- Need to reduce operational costs by automating deployment and scaling processes.
- Need to increase the reliability and availability of the deployed applications.
- Need to ensure compliance with security and regulatory requirements.
- Need to provide a flexible and extensible architecture that can adapt to changing business needs.

Extract/Abstract of requirements:

The "azdevops-aks-project" should provide a solution for deploying and managing containerized applications on Azure Kubernetes Service (AKS) using Azure DevOps pipelines. The solution should be able to automatically deploy and scale the AKS cluster based on application load and configure network policies for secure application communication. It should also integrate with monitoring and logging systems to provide visibility and troubleshooting capabilities. The solution should be flexible and extensible to support different application architectures and adapt to changing business needs. The driving forces for this solution include the need to improve agility, scalability, and reliability of the application deployment process while reducing operational costs and ensuring compliance with security and regulatory requirements.

Quality Goals

Contents

1. Reliability: The architecture should be designed to ensure that the application can operate continuously and with minimal downtime, even in the face of failures or unexpected events. This is important to stakeholders such as operations teams, who are responsible for ensuring the application is available and performing as expected.

2. Maintainability: The architecture should be designed to make it easy to maintain and modify the application over time. This includes factors such as modularization, clear separation of concerns, and adherence to best practices and standards. This is important to stakeholders such as developers, who are responsible for adding new features or fixing issues in the application.

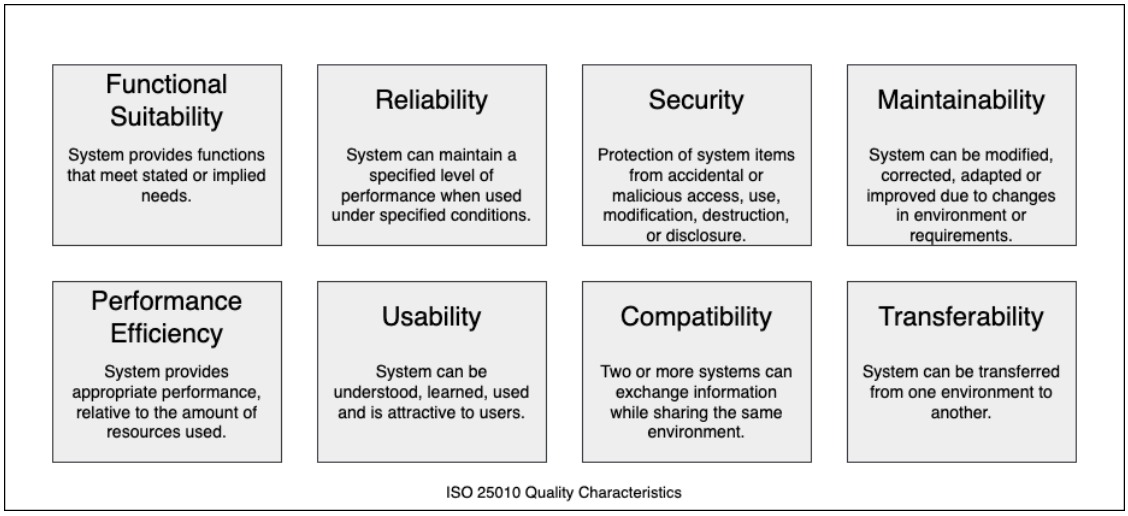
3. Security: The architecture should be designed to ensure the security and integrity of the application and its data. This includes measures such as authentication, authorization, encryption, and secure communication protocols. This is important to stakeholders such as security teams, who are responsible for ensuring the application meets security and compliance requirements.

Other potential quality goals that could be relevant for the "azdevops-aks-project" architecture, as per ISO 25010 standard, may include:

- Performance efficiency

- Compatibility
- Usability
- Portability
- Scalability
- Interoperability

However, the prioritization and importance of these goals would depend on the specific needs and requirements of the stakeholders and the organization.



Stakeholders

Contents

1. **Developers**: Developers are responsible for implementing the application code based on the architecture design. They need to understand the architecture to ensure that the code they write adheres to the design principles and best practices.
2. **Operations team:** The operations team is responsible for deploying, monitoring, and maintaining the application in production. They need to understand the architecture to ensure that the application can be deployed and operated efficiently and effectively.
3. **Security team:** The security team is responsible for ensuring that the application meets security and compliance requirements. They need to understand the architecture to ensure that security measures are implemented correctly, and that the application is secure and compliant.
4. **Project managers:** Project managers are responsible for planning and executing the project. They need to understand the architecture to ensure that the project plan aligns with the architecture design and that the project goals are met.

5. **Business stakeholders:** Business stakeholders are the individuals or organizations who have a vested interest in the application's success. They need to understand the architecture to ensure that the application aligns with the business goals and meets the needs of the users and customers.
6. **Quality assurance team:** The quality assurance team is responsible for ensuring the quality of the application. They need to understand the architecture to ensure that the application meets the quality goals and standards set for the project.
7. **End-users:** End-users are the individuals or organizations who will be using the application. They may not need to understand the architecture itself, but they need to be able to use the application efficiently and effectively based on the architecture design.
8. **Technical writers:** Technical writers are responsible for creating documentation for the architecture and the application. They need to understand the architecture to create accurate and useful documentation for the stakeholders.
9. **External auditors:** External auditors are responsible for evaluating the application's compliance with regulatory and industry standards. They need to understand the architecture to ensure that the application meets the relevant standards and requirements.

Overall, the stakeholders of the "azdevops-aks-project" include developers, operations teams, security teams, project managers, business stakeholders, quality assurance teams, end-users, technical writers, and external auditors. They may need to know about, be convinced of, work with, or make decisions about the architecture and its development.

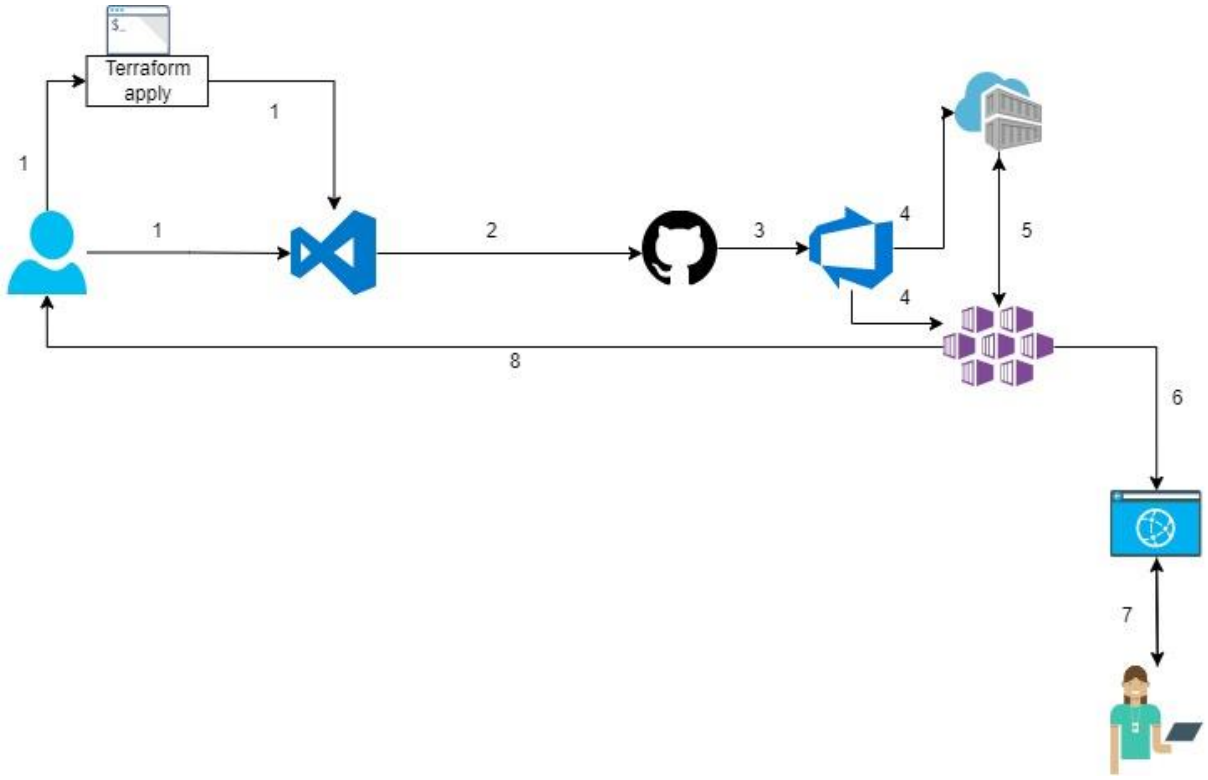
Prerequisites

1. Azure subscription: You will need an active Azure subscription to create the required resources for this project.
2. Azure KeyVault: The safest way to store all secrets regarding the project.
3. Azure DevOps organization: You will need an Azure DevOps organization to store the project code, track work items, and automate the CI/CD pipelines.
4. Azure Kubernetes Service (AKS) cluster: You will need an AKS cluster to deploy the application. (Will be deployed with Terraform)
5. Docker: You will need Docker installed on your local machine to build and push Docker images to the Azure Container Registry (ACR).
6. Azure DevOps Service connection: you need to create a Kubernetes-service-connection with your actual kubeconfig file, a Docker-registry-service connection with your ACR and subscription-service-connection.

7. Azure service Principal: Create or add Service Principal: to add do the following steps `az ad sp create-for-rbac --scopes /subscriptions/mySubscriptionID`
8. Helm: You will need Helm installed on your local machine to package, install, and manage Kubernetes applications.
9. kubectl: You will need kubectl installed on your local machine to manage Kubernetes clusters.
10. Azure CLI: You will need Azure CLI installed on your local machine to create and manage Azure resources.
11. Git: You will need Git installed on your local machine to clone and work with the project code.
12. Visual Studio Code: You will need Visual Studio Code (or any other code editor of your choice) installed on your local machine to work with the project code.

Architecture and Deployment

Contents



Firstly, we use Terraform to create our infrastructure. This involves creating an AKS cluster and an ACR registry. Once the infrastructure is set up, we trigger the pipeline to deploy our application.

The pipeline consists of several steps. Firstly, we make a safe Azure login through the pipeline to use the Azure CLI. This ensures that we have secure access to our Azure resources.

Once we have access, we retrieve the secrets we need from the Azure Key Vault. These secrets are set up in the pipeline environment so that we can use them in further steps of the pipeline, such as when we build the Docker image and push it to the ACR registry.

By retrieving the secrets securely from the Azure Key Vault, we ensure that our pipeline is secure and that our sensitive information is protected.

Secondly, we build the application and create a Docker image. This image is then pushed to the ACR registry.

After that, the pipeline deploys the application to the AKS cluster. We use Helm to install the HA-Proxy ingress on the pipeline agent, and then deploy it to the AKS cluster. The HA-Proxy ingress allows us to expose our application to external users via an external IP.

Finally, we use a simple test to check whether the application and the ingress are working correctly. The pipeline exposes the external IP for the agent, and we test it with a curl -v command to check if the application is running as expected.

In summary, we use Terraform to set up our infrastructure, and then use a pipeline to build and deploy our application to the AKS cluster. The pipeline ensures that everything is tested and working correctly before exposing it to external users.