**1. DevOps Architecture**

**1.1. Introduction:**

The DevOps architecture for the Speech-to-Text Converter project is designed to ensure continuous integration, continuous deployment (CI/CD), and continuous monitoring of the application. This architecture leverages Azure DevOps services to automate the build, test, and deployment processes, ensuring that the application is always in a deployable state.

**1.2. Key Components:**

** Source Control (Azure Repos):**

o Description: A version control system to manage and track changes to the source code.

o Role: Ensures that all changes to the code are tracked and can be reverted if necessary. It also allows multiple developers to collaborate on the codebase.

** CI/CD Pipelines (Azure Pipelines):**

o Description: Automated pipelines for building, testing, and deploying the application.

o Role: Automates the process of compiling the code, running tests, and deploying the application to different environments. This ensures that changes are validated before they reach production.

** Artifact Management (Azure Artifacts):**

o Description: A package management solution for storing and managing build artifacts.

o Role: Stores the build outputs and dependencies required for deployment, ensuring that the same artifacts are used across different environments.

** Configuration Management (Azure Key Vault):**

o Description: A service for securely storing and accessing secrets, keys, and configuration settings.

o Role: Manages sensitive information such as API keys, connection strings, and passwords used by the application.

 Infrastructure as Code (Azure Resource Manager Templates):

o Description: Templates to define and manage the infrastructure required for the application.

o Role: Ensures that the infrastructure is consistent and can be easily replicated or modified.

 Monitoring and Logging (Azure Monitor and Azure Log Analytics):

o Description: Services for monitoring the application performance and logging application activities.

o Role: Provides insights into the application's health and performance, helping to identify and resolve issues proactively.

**1.3. Workflow:**

 Code Commit:

o Developers commit changes to the source code repository (Azure Repos).

 Continuous Integration (CI):

o Upon code commit, Azure Pipelines automatically triggers the build pipeline.

o The build pipeline compiles the code, runs unit tests, and creates build artifacts.

o The build artifacts are stored in Azure Artifacts.

 Continuous Deployment (CD):

o After a successful build, the deployment pipeline is triggered.

o The deployment pipeline deploys the application to different environments (development, staging, production) using Azure Resource Manager templates.

o Configuration settings are retrieved from Azure Key Vault.

 Monitoring and Logging:

o The deployed application is monitored using Azure Monitor.

o Logs are collected and analyzed using Azure Log Analytics to ensure the application is performing as expected.

**1.4. Detailed Steps:**

**1. Source Control (Azure Repos):**

o Initialize a Git repository in Azure Repos.

o Commit and push the source code to the repository.

o Set up branch policies to enforce code reviews and quality checks.

**2. CI Pipeline (Azure Pipelines):**

o Define a build pipeline in Azure Pipelines using a YAML file.

o Configure the pipeline to:

 Pull the latest code from Azure Repos.

 Restore dependencies and compile the code.

 Run unit tests.

 Create build artifacts and publish them to Azure Artifacts.

**3. CD Pipeline (Azure Pipelines):**

o Define a release pipeline in Azure Pipelines.

o Configure the pipeline to:

 Retrieve the build artifacts from Azure Artifacts.

 Deploy the application to the development environment.

 Run integration tests.

 Promote the deployment to staging and production environments based on approval gates.

**4. Configuration Management (Azure Key Vault):**

o Store sensitive configuration settings in Azure Key Vault.

o Grant the deployment pipeline access to Azure Key Vault.

o Retrieve configuration settings during deployment.

5. Infrastructure as Code (Azure Resource Manager Templates):

o Define the required infrastructure using ARM templates.

o Include resources such as Azure App Services, Azure SQL Database, Azure Blob Storage, etc.

o Deploy the infrastructure using the deployment pipeline.

6. Monitoring and Logging (Azure Monitor and Azure Log Analytics):

o Set up Azure Monitor to track application performance metrics.

o Configure Azure Log Analytics to collect and analyze application logs.

o Set up alerts to notify the team of any issues.

**1.5. DevOps Architecture**

**1.6. Conclusion:** The DevOps architecture ensures a robust, automated, and scalable process for building, testing, and deploying the Speech-to-Text Converter application. By leveraging Azure DevOps services, the architecture supports continuous delivery and integration, ensuring that the application remains in a deployable state and meets the quality standards required for production.