Terraform

https://learn.microsoft.com/en-us/azure/devops/pipelines/process/phases?view=azure-devops&tabs=yaml

https://dev.azure.com/aldfrance/ALDFAPI/\_git/Aldfbp.TnC.Templates?path=/templates&\_a=contents&version=GBmaster

IdentityServer

SendGrid

https://aldfrance.visualstudio.com/ALDFAPI/\_wiki/wikis/ALDFAPI.wiki/1020/Introduction

**Terraform**

<https://blog.stephane-robert.info/docs/infra-as-code/provisionnement/terraform/introduction/>

<https://aldfrance.visualstudio.com/ALDFAPI/_wiki/wikis/ALDFAPI.wiki/1022/Terraform>

is an open-source infrastructure as code (IaC) tool developed by HashiCorp

-provider.tf : defines the used providers azurerm and azuread

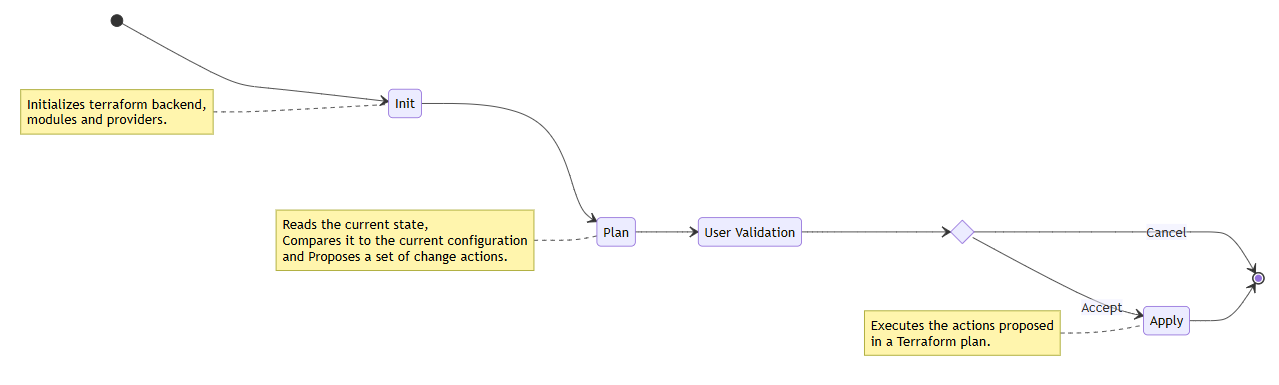
-variables.tf : defines the variables that will used in terraform script

-variables-int.tfvars, variables-stg.tfvars and variables-prd.tfvars : defines variables values for each environment

-main.tf : this is the main terraform script that will describe the infrastructure for provisioning

-modules : contains the definition of the modules that will be consumed in main.tf

The main terraform script is excecuted by a YAML pipeline (azure-pipeline-tf-resources.yml) following these steps :



**Pipelines**

<https://aldfrance.visualstudio.com/ALDFAPI/_wiki/wikis/ALDFAPI.wiki/1024/Azure-Pipelines>

variables.yml defines the default and shared variables that will be consumed by other YAML pipelines such as application name, pools names, Terraform,

.NET SDK, Entity Framework, Docker, AKS, MSSQL, APIM and NSwag variables.

These values will be overrided in variables-int.yml, variables-stg.yml and variables-prd.yml to adapt to each environment.

Terraform pipeline

azure-pipeline-tf-resources.yml executes the Terraform script over all environments to provision needed Azure resources on Azure subscription defined in

the environment according variables file.

AKS pipelines

These pipelines build, test and deploy code on Kubernetes. It may generate and execute SQL migration script if the API was generated with --ef arguments.

List of AKS pipelines:

azure-pipeline-int.yml

azure-pipeline-stg.yml

azure-pipeline-prd.yml

APIM pipelines

These pipelines publish the APIM package to the T&C APIM instance. The APIM package located in .apim-package solution folder contains an OpenAPI specification in swagger.json file. This specification defines API to be published.

The APIM pipelines use the aiInstrumentationKey defined in the variables YAML file to enable APIM monitoring through Azure Application Insights service.

List of APIM pipelines:

azure-pipeline-apim-dev.yml

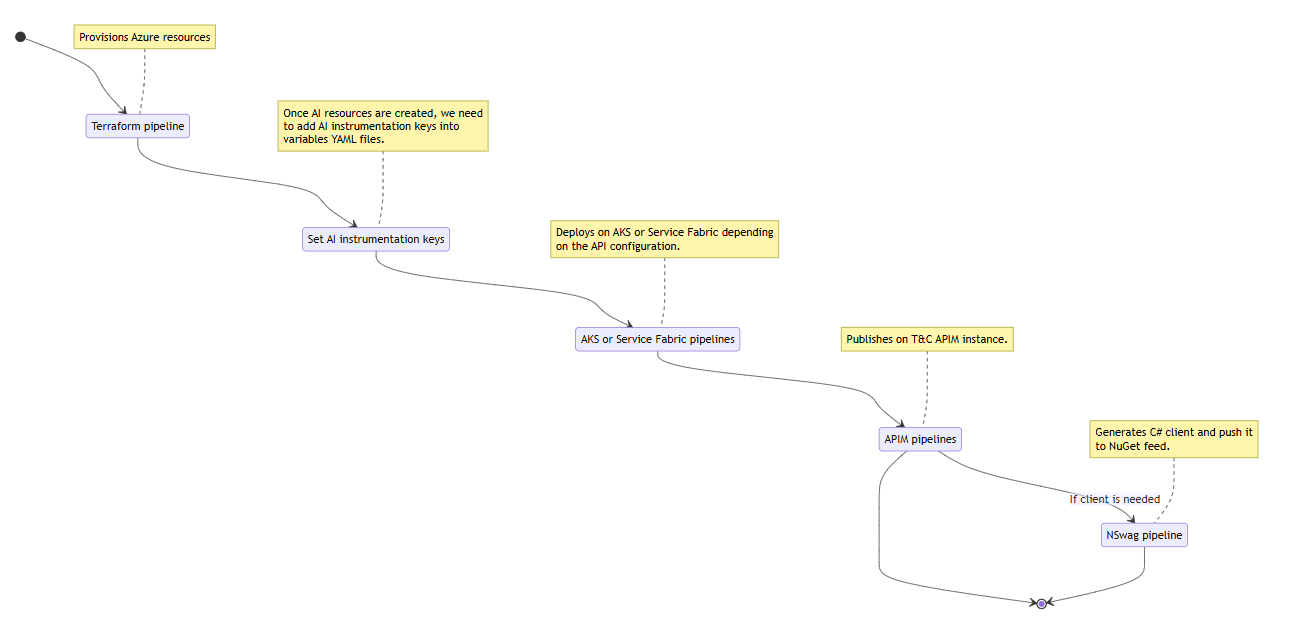
azure-pipeline-apim-master.yml

NSwag pipeline (https://github.com/RicoSuter/NSwag)

NSwag is a Swagger/OpenAPI 2.0 and 3.0 toolchain for .NET, .NET Core, Web API, ASP.NET Core, TypeScript (jQuery, AngularJS, Angular 2+, Aurelia, KnockoutJS and more) and other platforms, written in C#. The OpenAPI/Swagger specification uses JSON and JSON Schema to describe a RESTful web API.

The azure-pipeline-nswag-api-client.yml pipeline generates a C# clients from an OpenAPI specification based on swagger.json in .apim-package and publishes it on the NuGet feed aldfr.packages.

Pipelines execution workflow:



**APIM**

<https://aldfrance.visualstudio.com/ALDFAPI/_wiki/wikis/ALDFAPI.wiki/1026/APIM>

When generating a new API project using Aldfbp Api Template, the solution comes with the folder .apim-package. This folder contains the the API product, policies, Swagger file and the content file that defines the api, its variables and product.

Using the APIM pipelines, APIM package will be published on all environments based on the configuration specified in the pipeline and variables files and the OpenAPI specification of the API in the APIM package.

Azure API Management is a fully managed service that helps organizations publish, secure, monitor, and scale APIs across cloud and hybrid environments.

https://www.youtube.com/watch?v=oQYhiN9bj3Y&list=PLU1w\_BFZFd2oSHrHLdy0uhmqsYsUvgvby&index=8

**DOCKER**

Docker is an open-source tool that allows developers, sys-admins etc. to easily deploy their applications in a sandbox (called containers) to run on the host operating system i.e. Linux. T

Dockerfile

A Dockerfile is a simple text file that contains a list of commands that the Docker client calls while creating an image. It's a simple way to automate the image creation process.

Docker-compose

Docker Compose is a tool for defining and running multi-container Docker applications.

K8s

Kubernetes , also known as K8s, is an open-source system for automating deployment, scaling, and management of containerized applications.

A Kubernetes manifest file is your personal guide through a Kubernetes cluster: A configuration file written in a format called YAML or JSON, that describes the resources you want in your cluster. These resources can be a myriad of things: pods (that run your applications), services (that help your applications communicate), and deployments (that manage your applications).

Deployment

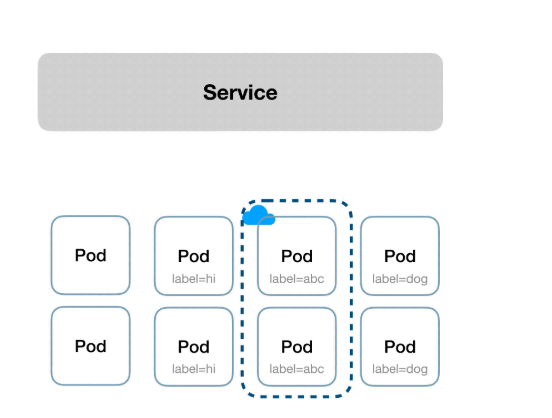
A Kubernetes Deployment YAML specifies the configuration for a Deployment object—this is a Kubernetes object that can create and update a set of identical pods.

The Deployment object not only creates the pods but also ensures the correct number of pods is always running in the cluster, handles scalability, and takes care of updates to the pods on an ongoing basis. All these activities can be configured through fields in the Deployment YAML.

Service

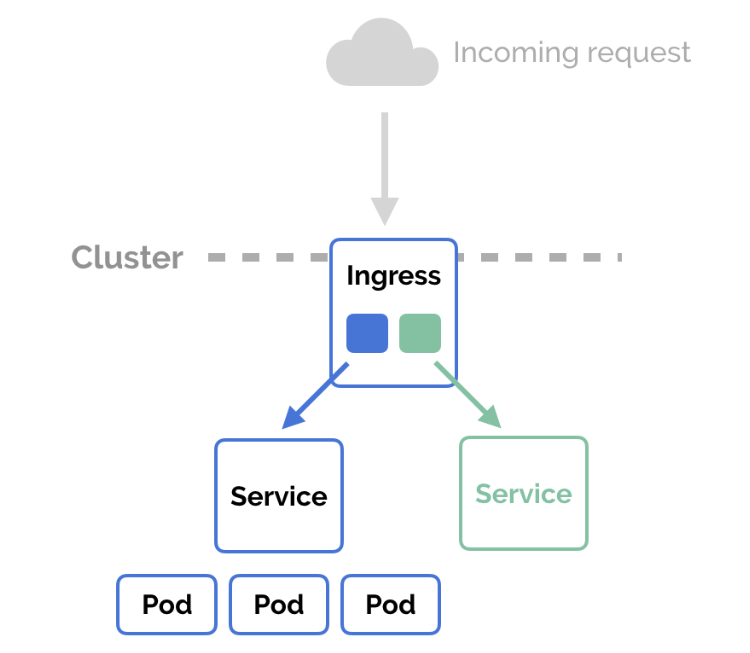
A Service enables network access to a set of Pods in Kubernetes.

Services select Pods based on their labels. When a network request is made to the service, it selects all Pods in the cluster matching the service's selector, chooses one of them, and forwards the network request to it.



**Ingress**

In Kubernetes, an Ingress is an object that allows access to your Kubernetes services from outside the Kubernetes cluster. You configure access by creating a collection of rules that define which inbound connections reach which services.



**SonarQube**

**AutoRest** is a powerful tool that facilitates the generation of client libraries for interacting with RESTful web services. It requires a specification that outlines the REST API using the OpenAPI format, enabling the streamlined production of client code

**NSwag** serves as a robust toolchain for working with Swagger/OpenAPI versions 2.0 and 3.0, designed for environments like .NET, .NET Core, ASP.NET Core, and TypeScript, and it is developed in C#. This tool empowers developers to create OpenAPI specifications directly from their existing API controllers and subsequently generate client code based on those specifications.

**Azure Service Bus**

[Azure Service Bus using Aldfbp.Boilerplate - Overview](https://aldfrance.visualstudio.com/ALDFAPI/_wiki/wikis/ALDFAPI.wiki/1187/Azure-Service-Bus-using-Aldfbp.Boilerplate)

Azure Service Bus is a fully-managed message broker with message queues and topics in a namespace. It is used to decouple applications and services from each other, providing the following benefits:

* Load-balancing work across competing workers
* Safely routing and transferring data and control across service and application boundaries
* Coordinating transactional work that requires a high-degree of reliability

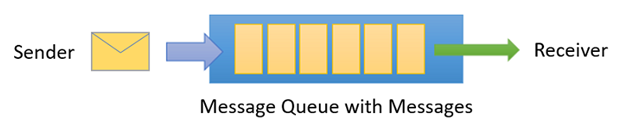
Data is transferred between different applications and services using messages. A message is a container decorated with metadata, and contains data. The data can be any kind of information, including structured data encoded with the common formats such as the following ones: JSON, XML, Apache Avro, Plain Text.

## Namespaces

A namespace is a container for all messaging components (queues and topics). Multiple queues and topics can be in a single namespace, and namespaces often serve as application containers.

## Queues

Messages are sent to and received from queues. Queues store messages until the receiving application is available to receive and process them.



Messages in queues are ordered and timestamped on arrival.

## Topics

Topics can have multiple, independent subscriptions, which attach to the topic and otherwise work exactly like queues from the receiver side. A subscriber to a topic can receive a copy of each message sent to that topic.

