Azure Infrastructure

Overview & New Environment Setup

Date	Author	Version
14/01/2021	Sebastian Balan	1.0

Overview

The purpose of this document is to describe the DemoAir Environment setup in Microsoft Azure Cloud.

In the DemoAir Azure Infrastructure, the environment is defined in a **Azure Kubernetes Service (AKS)** instance.

Azure Cloud components

● **Application Gateway**: The main entry point in the DemoAir Azure infrastructure. It's a web traffic load balancer and connects the frontend (Public IP address) to the backend in kubernetes clusters. The primary role for this Application Gateway is to be a web application firewall, for this we are using OWASP 3 ModSecurity Core Rule Set.

● **Vnet**: A Virtual Network, enables Azure resources to securely communicate with each other.

● **Azure Kubernetes Service** (AKS): A managed Kubernetes cluster provided by Azure.

● **Key Vault**: Azure resource responsible for storing all the sensitive data for each applications/infrastructure

● **Container Registry (ACR)**: Azure resource that stores all the Docker images for DemoAir services deployed in AKS, Docker images are created via CI/CD

The AKS cluster has an internal **application-ingress** load balancer deployed, this load balancer is used to route traffic to the microservices deployed in the AKS instance.
Internally the communication with the kubernetes cluster is done via **Virtual Network**, using a private IP address space.

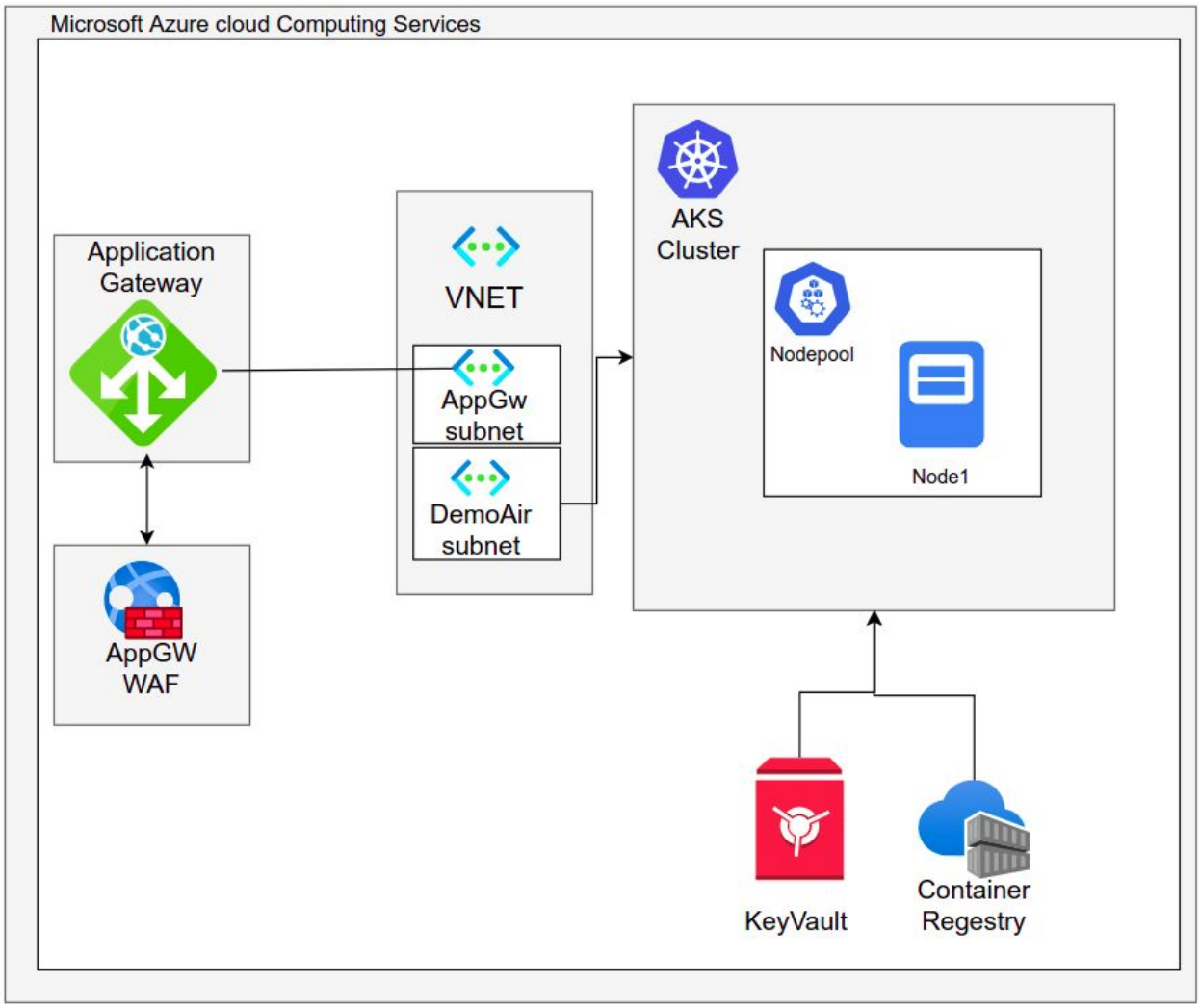
AKS Cluster

AKS Cluster has the following services (containers) deployed:

1. Webfrontend. The DemoAir product
2. **Application Ingress**. Routes traffic from Application Gateway to AKS
3. **KeyVault Integration**. It is using pod Identity, provides a secure way to use sensitive data
4. **Prometheus.** Metrics and monitoring of the Infrastructure.
5. **Grafana.** Alongside Prometheus it is used for analytics & monitoring & alerting

The following diagram describes the relationship between the deployed services.

DemoAir - Main Components Overview



Environments
DemoAir: http://demoairpublic.centralus.cloudapp.azure.com/

Estimated costs (month)

ServiceName	Estimated additional CostUSD for production
application gateway/fixed cost	329.59
virtual machines	150.89
log analytics	51.9
container registry	20.66
security center/workspaces	14.88
OTHER (IPs, disks, Storage account)	40
Total	605

Source code

Infrastructure (Terraform) repo can be found here:
https://dev.azure.com/projectdemo192/_git/Demo-Infra-Apss

Application repo can be found here:
https://dev.azure.com/projectdemo192/_git/Applications

Infrastructure as a code

Using either Azure CLI or the Azure Portal deploying process is very slow, involves running a lot of steps manually and is error-prone - we can skip some steps or if we identify at the end that some of the steps were not installed correctly we need to start over the setup.

In order to reduce these risks and increase the speed of setting up the infrastructure, our recommendation is to use a solution based on Infrastructure as a Code.

What is IaaC (Infrastructure as a Code)?

Infrastructure as Code (IaC) is a combination of standards, practices, tools, and processes to provision, configure, and manage computer infrastructure using code and other machine-readable files.

Advantages of IaaC (Infrastructure as a Code)

1. First, the documentation side, which is asynchronous to the task itself tends to get outdated and ignored. If the job requires you to document a change in a wiki page (or similar to that), you might easily forget to track that change. This results in differences between effective and assumed implementation, which can result in service failures, security or compliance issues.
2. Second, Automating regular tasks makes executing them easier, safer and more resilient. Automatization requires a structured definition of the required output, and by using IaaC the output is defined in some form of code. Automation makes it easier to compare a desired state with an effective state, and usually computers tend to be more error-prone than humans.
3. And third, disaster recovery. A backup is useless, if one never tried to restore it. The same applies to infrastructure. Knowing that a brand-new and up-to-date infrastructure is just a click away, one can have a disaster recovery in place, at no additional cost.

Terraform

Azure provides integration with popular open source and third-party tools and services—across the entire DevOps workflow. Spend less time integrating and more time delivering higher-quality software, faster.

In addition to using Azure Resource Manager for infrastructure as code, you can provision and manage Azure infrastructure directly from your favorite third-party tools, such as Ansible, Chef, Puppet, and Terraform.

Infrastructure as a code source control

There are multiple options available for source control that can be used with Azure. The recommendation is to use in this case the Azure DevOps Repos - we have unlimited private GIT repositories through our MSDN subscription - and access to this is made through Azure DevOps RBAC

Azure DevOps vs GitHub

1. Pricing is pretty straightforward: the Azure DevOps family of products is free for your first 5 users with unlimited private repositories in Azure Repos. For private repositories, GitHub is $9 / user / month.

2. As far as features, there's a large amount of parity: both support a rich code review experience with pull requests, branch policies, Git-LFS, webhooks, integration with Azure Pipelines, etc.

Application Scaling

Azure supports auto-scaling, for AKS (managed Kubernetes cluster in Azure) auto-scaling is done for PODs and for NODEs.

For DemoAir we need auto-scaling for PODs via HPA mechanism (https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale/)

Auto-scaling for NODEs is not implemented yet and it will be added at a later time. More information on how to auto scale a managed cluster in Azure (AKS): (https://docs.microsoft.com/en-us/azure/aks/cluster-autoscaler)

The 'HPA' and 'Cluster Autoscaler' will work together :
- 'HPA' will increase and decrease the number of PODs for each configured application based on used resources
- 'Cluster Autoscaler' will increase and decrease the number of NODEs to accommodate 'HPA'

scaling

Application Security

Azure Web Application Firewall (WAF)

on Azure Application Gateway provides centralized protection of your web applications from common exploits and vulnerabilities. Web applications are increasingly targeted by malicious attacks that exploit commonly known vulnerabilities. SQL injection and cross-site scripting are among the most common attacks.

WAF on Application Gateway is based on Core Rule Set (CRS) 3.1, 3.0, or 2.2.9 from the Open Web Application Security Project (OWASP). The WAF automatically updates to include protection against new vulnerabilities, with no additional configuration needed.

Azure Key Vault with Secrets Store CSI Driver

Using the pod identity project enables authentication against supporting Azure services. For your own services or applications without managed identities for Azure resources, you can still authenticate using credentials or keys. A digital vault can be used to store these secret contents.

When applications need a credential, they communicate with the digital vault, retrieve the latest secret contents, and then connect to the required service. Azure Key Vault can be this digital vault.