All (some) about Azure!

Networking, Security & Services

So far

- Deployed containers to Azure
- Used Terraform for our infrastructure
- Built CI/CD pipelines

What's Missing?

A deeper understanding the **underlying Azure foundation**.

We don't always have the option of using the nice, shiny solution. Sometimes we have to pick up existing work, or we're forced to build something in another layer of detail.

- How does networking look in the cloud?
- How do we secure these things on the internet? Should they even be on the internet?
- What other services should we consider?

Networking at a Glance

Virtual Networks (VNets)

Think of it as your own **private network in the cloud**

- Completely isolated from other customers and the internet
- You control the IP address space
- Resources communicate securely within the VNet

Subnets

Shortened word for "Subset of a Virtual Network"

Divide your available IP address space into smaller chunks to get more granularity

A bit like having lockers at the gym!

IP address ranges:

```
VNet: 10.0.0.0/16
├── Web subnet: 10.0.1.0/24
├── App subnet: 10.0.2.0/24
└── Data subnet: 10.0.3.0/24
```

Each subnet gets its own address range from the VNet

This X.X.X.X/Y notation is known as a CIDR range, remember this for asking Copilot questions

Network Security Groups (NSGs)

To continue the Gym analogy, this is the padlock on each locker

Control what traffic is allowed in and out

NSGs contain rules that:

- Allow or deny traffic
- Based on source, destination, port, protocol
- Applied to subnets or individual resources

Common NSG rules:

Rule	Description
Allow HTTPS from internet	Port 443 from anywhere
Allow SSH from office	Port 22 from specific IP
Allow database from app subnet	Port 5432 from 10.0.2.0/24
Deny all other inbound	Default deny rule

Security by default:

> *NSGs follow a "deny all unless explicitly allowed" approach*

This means you only open the ports you actually need. Azure gives us a default deny all, but only **after** it's opened up internally.

Best practice here is that your first rule should be to block **everything**. Then add what you need iteratively.

Public vs Private IP Addresses

Two types of addresses in Azure

Public IP Addresses:

- Accessible from the internet
- Used for resources that need external access
- Cost money in Azure
- Examples: Load balancers, VMs with SSH access

Private IP Addresses:

- Only accessible within your VNet, meaningless to the outside world
- Free to use
- More secure (not exposed to internet)
- Examples: Databases, internal APIs, backend services

Best practice:

> *Keep as much as possible on private IPs, only expose what absolutely needs to be public*

Generally, we want to avoid anything being available to the internet as much as we can. It's on a needs-must basis

DNS Zones

Manage domain names in Azure

We covered DNS basics in Part 2 - now let's look at Azure's DNS service

Azure DNS Zones provide:

- Host your domain in Azure
- Manage DNS records (A, CNAME, etc.)
- High availability and fast resolution
- Integration with Azure resources

Two types of DNS zones:

Public DNS Zones:

- Resolve names from the internet
- Example: `academy.kainos.com` → Public IP -> some resource intended for public use

Private DNS Zones:

- Resolve names within VNets only
- Example: `database.internal.kainos.com` → Private IP -> internal only resource

A reminder here that we love DNS because it is so much more flexible than a single IP address!

Security

Identity and Access Management (IAM)

Who can do what in Azure?

Control access to your Azure resources to different people, workloads, and automation

Terminology

- **Authentication:** ?
- **Authorization:** ?
- **Role-Based Access Control (RBAC):** ?

Terminology

- **Authentication:** Prove who you are
- **Authorization:** What you're allowed access to do
- **Role-Based Access Control (RBAC):** Assign permissions via roles

Common Azure roles:

Role	Access Level
Owner	Full control, can grant access
Contributor	Manage resources, no access control
Reader	View resources only
Custom	Specific permissions you define

Best practice:

> *Grant the **absolute minimum permissions** needed to do the job*

This is called the **principle of least privilege** and is critical in production systems. We do not need to Own a resource to see that it exists, for instance.

Service Principals

Identities for applications and services

When your application needs to access Azure resources

Use cases:

- Terraform accessing Azure
- CI/CD pipelines deploying resources
- Applications reading from Key Vault
- Automated scripts managing infrastructure

Why go to all the effort?

- Service principals don't expire when you leave
- Can be scoped to specific resources
- Better audit trail
- No MFA interruptions for automation

Managed Identities

Even better than service principals

Azure-managed identities with no credentials to manage

System-assigned:

- Tied to a specific resource (VM, Container App, etc.)
- Created and deleted with the resource
- One identity per resource

User-assigned:

- Standalone identity
- Can be shared across multiple resources
- Independent of resource lifecycle

Why managed identities are great:

- No passwords or secrets to manage
- Azure handles credential rotation
- More secure than storing credentials
- Simpler to use

Azure Firewall

Centralized network security

A managed firewall service for your Azure resources

Key features:

- Filter outbound traffic
- Threat intelligence
- Application and network rules
- High availability built-in

When to use Azure Firewall:

- Need centralized security for multiple VNets
- Require advanced threat protection
- Complex routing and filtering needs
- Compliance requirements

When NSGs are enough:

- Simple allow/deny rules
- Resource or subnet-level filtering
- Starting out with Azure