Azure AD

Memi Lavi
www.memilavi.com

Azure AD

- Short for Azure Active Directory
- Central identity and access management cloud service
- Used to manage access to thousands of apps
  - Among them – the Azure Portal
- Secure, robust, intelligent

Azure AD

- Advanced features:
  - MFA
  - Conditional Access
  - Device management
  - Hybrid identity
  - Identity protection
  - Monitoring and reports
  - And lots more…

Azure AD

- We're interested mainly in:
  - Control access to Azure resources
    - By setting up users, groups, roles
    - Not entirely architecture- and dev- related, but still important
  - Use Azure AD to add authentication to our apps
  - Can be done also via Azure AD B2C

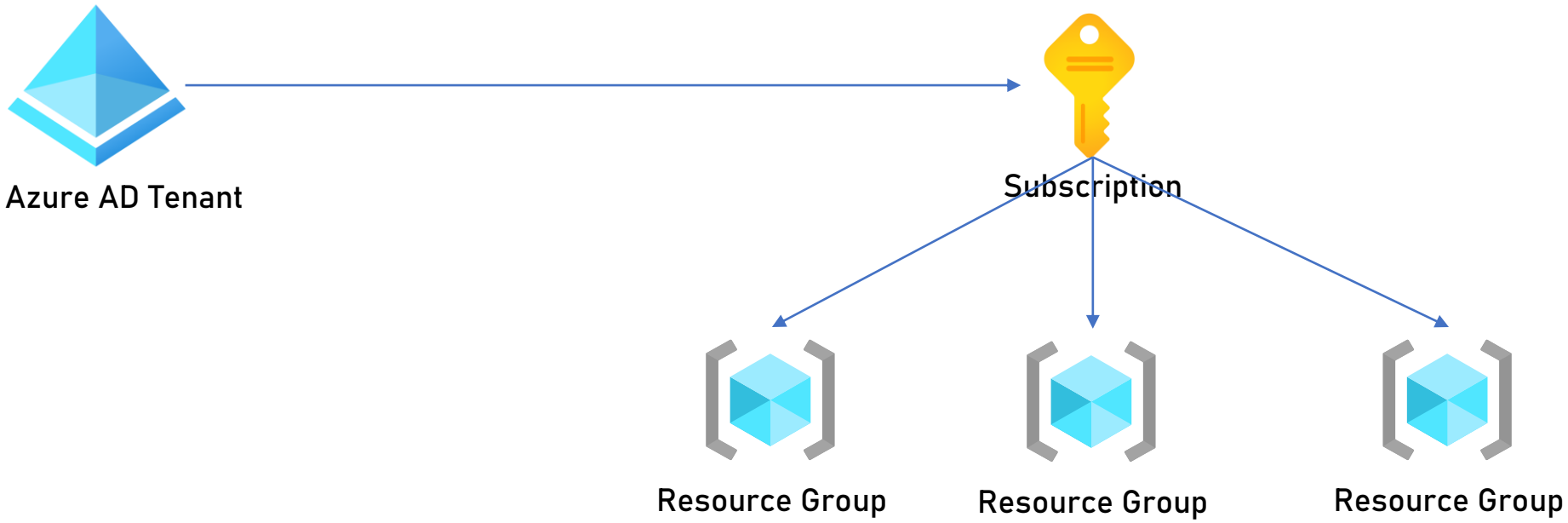
Azure AD Figures

- Integrates with more than 2,800 apps
- Manages more than 1.2 billion identities
- Processes over 8 billion authentications every day
- Secured using 3,500 security experts in Microsoft
- …which invests more than $1bn annually on cybersecurity
- The largest identity and access management service in the world

Tenant

- A specific instance of Azure AD containing accounts and groups
- Called also Directory
- Is NOT part of the subscription hierarchy
  - Exists beside the subscription
  - For new subscriptions, a new tenant is created automatically
- A tenant can be assigned to multiple subscriptions

Tenant



Users and Groups

- Two of the main three objects managed by Azure AD
  - The 3rd one is Roles (later…)
- Manages and stores the users that are part of the tenant
- Groups the users in Groups
  - Examples: IT Admins, Developers, etc.
  - Allows defining roles to groups instead of each user

Azure AD Licenses

- Azure AD Licenses have great effect on the functionality and price of Azure AD
- Important to understand the differences and recommend the right solution

Azure AD Licenses

	Free	Premium 1	Premium 2
Max Objects	500,000	Unlimited	Unlimited
Users & Groups	X	X	X
MFA	X (All or nothing)	X (With Conditional Access)	X (With Conditional Access)
Dynamic Groups		X	X
Conditional Access		X	X
Risk Detention			X
Risk based Conditional Access			X
Privileged Identity Management (PIM)			X
Price	Free	6$ user / month	9$ user / month

FROM SECURITY COURSE,

S6L2, 14:30-18:44

Azure AD Security Defaults

- Increases protection of the organization in the Free tier
- Adds preconfigured security settings:
  - Requiring all users to use MFA (block 99.9% of account compromises)
  - Blocks legacy authentication
  - And more…
- No additional cost (so…still free ☺)
- For more fine-grained management – use Conditional Access (P1)

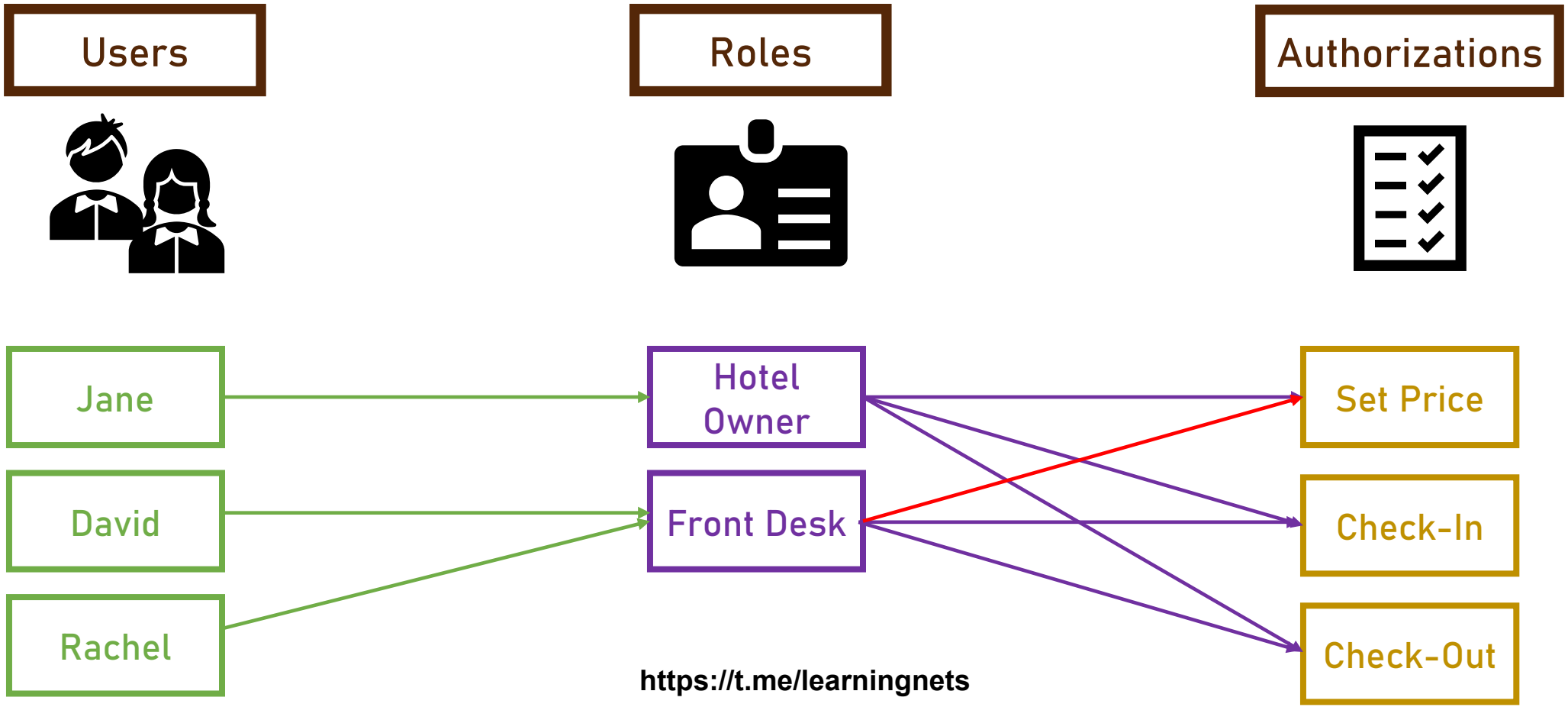
Role Based Access Control (RBAC)

- In the past, authorization was defined per user or user group
- Examples:
  - David is allowed to add items to the inventory
  - John is not allowed to read data of other doctors
- Very granular, extremely hard to maintain

RBAC to The Rescue!

Role Based Access Control (RBAC)

- With RBAC:



Role Based Access Control (RBAC)

- Where are roles managed?
  - In the Authentication Engine (ie. User Groups in Active Directory)
    - Passed to the component as part of the user's token
  - In the component's user store
    - Retrieved after receiving the user's token from the Authentication Engine

RBAC Implementation

Action Authorization

Usually using built-in support in the development platform

```
[Authorize(Roles = "Manager, Administrator")]
public class DocumentsController : Controller
{
    public ActionResult ViewDocument()
    {
        //Your code here
    }
    [Authorize(Roles = "Administrator")]
    public ActionResult DeleteAllDocuments()
    {
        //Your code here
    }
}
```

RBAC Implementation

Action Authorization

Usually using built-in support in the development platform

```
// Add this to the top of the file
const { roles } = require('../roles')

exports.grantAccess = function(action, resource) {
 return async (req, res, next) => {
  try {
   const permission = roles.can(req.user.role)[action](resource);
   if (!permission.granted) {
    return res.status(401).json({
     error: "You don't have enough permission to perform this action"
    });
   }
   next()
  } catch (error) {
   next(error)
  }
 }
}
```


Source: https://blog.soshace.com/implementing-role-based-access-control-in-a-node-js-application/

RBAC Implementation

Data Authorization

- Using the database's Row Level Security (RLS)
- Self development

```
Public MedData GetMedicalData(doctorId doc)  {

        using (var db = new MedContext())  {

                var medData=db.MedicalData
                    .Where(d=>d.ownerDoctor==doc)
                    .Order(d=>d.creationDate))
                    .ToList();

        }
}
```

RBAC in Azure

- In order to perform any operation, or access any data in Azure you have to have the appropriate role
- If you want to:
  - Create resource groups
  - Access data in SQL
  - See metrics of App Service
- … then you have to have the right role
- If you don't – you'll get an empty portal

RBAC in Azure

- In general, three types of roles:

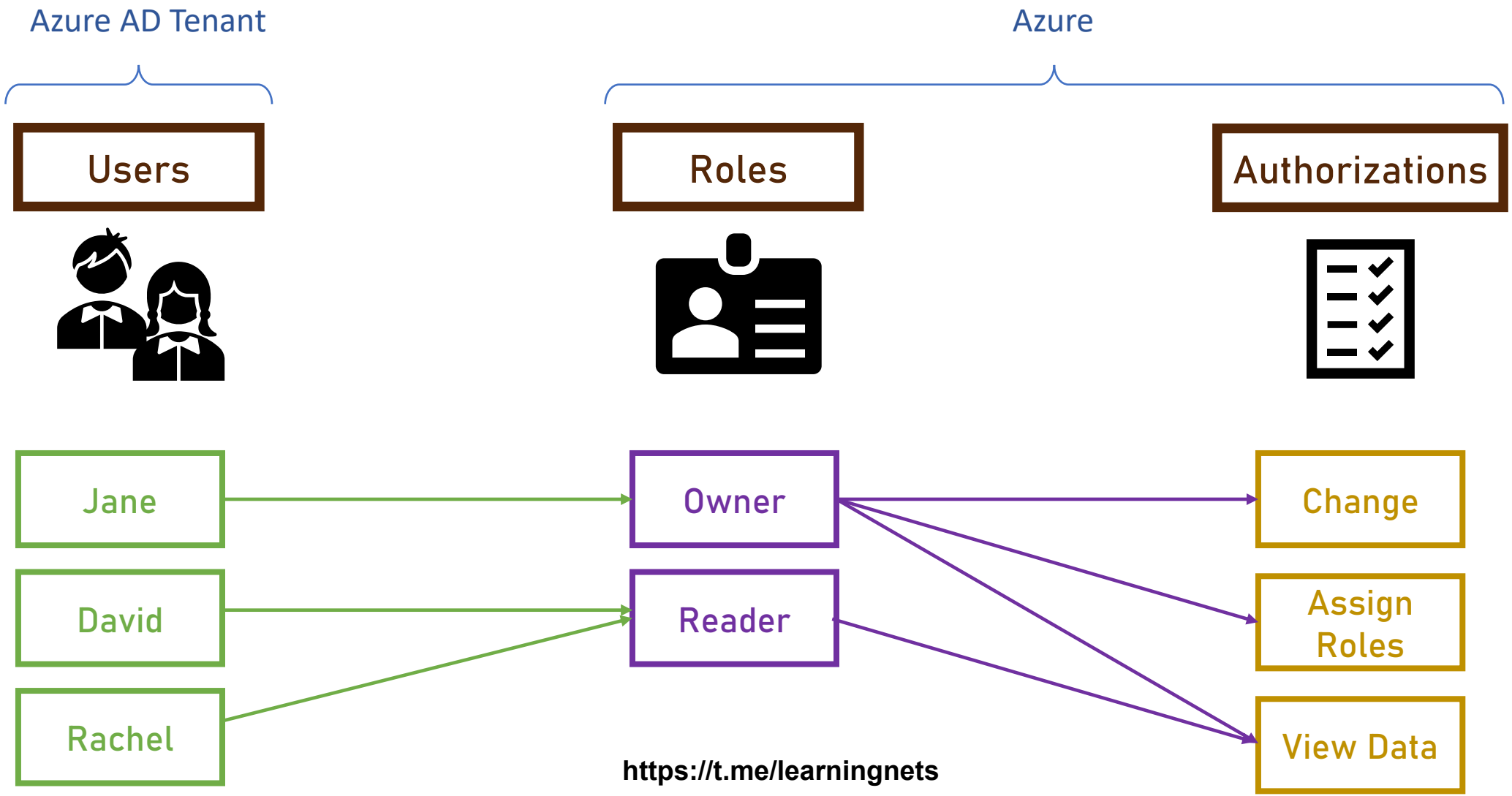
Owner	Can perform any action on the resource, including assigning roles to it
Contributor	Can perform any action on the resource, but cannot assign roles to it
Reader	Can only view data, but cannot change anything

RBAC in Azure

- Examples:

Virtual Machine Contributor	Can manage virtual machines
Cosmos DB Account Reader	Can read Azure Cosmos DB account data
Service Bus Data Owner	Allows full access to Service Bus resources

RBAC in Azure



RBAC in Azure

- It's always better to assign roles to groups and not individual users
- Easier maintenance

Managed Identities

- The ability to assign Azure AD identity to Azure resource
- The resource can connect to other Azure resources using this identity
- No need to handle credentials (usernames, passwords etc.)

Managed Identities

- Two types of Managed Identities:
  - System assigned – Managed by Azure, tied to the resource's lifecycle (when the resource is deleted – so is the identity)
  - User assigned – Managed by the user. Can be assigned to multiple resources, not tied to any lifecycle

Managed Identities

- Resources that can be assigned Managed Identity:
  - App Service
  - Virtual Machine
  - Event Grid
  - Function
  - And more…

Managed Identities

- Resources that can be authorized using Managed Identity:
  - SQL
  - Event Hubs
  - Service Bus
  - Storage
  - Key Vault
  - And more…

Using Azure AD on Our App

- Azure AD can be used as authentication engine on other apps
- Not just the Azure Portal
- It can be used on our own app!

Using Azure AD on Our App

- The process:
  - Register the app in Azure AD
  - Add code to use Azure AD as authentication engine
    - For App Services – can be configured via the Portal

Using Azure AD on Our App

- The authentication:
  - Uses OAuth and JWT

Authentication Protocols

Merge with Security course, Section 6,

Lecture 3 (maybe not until the end)

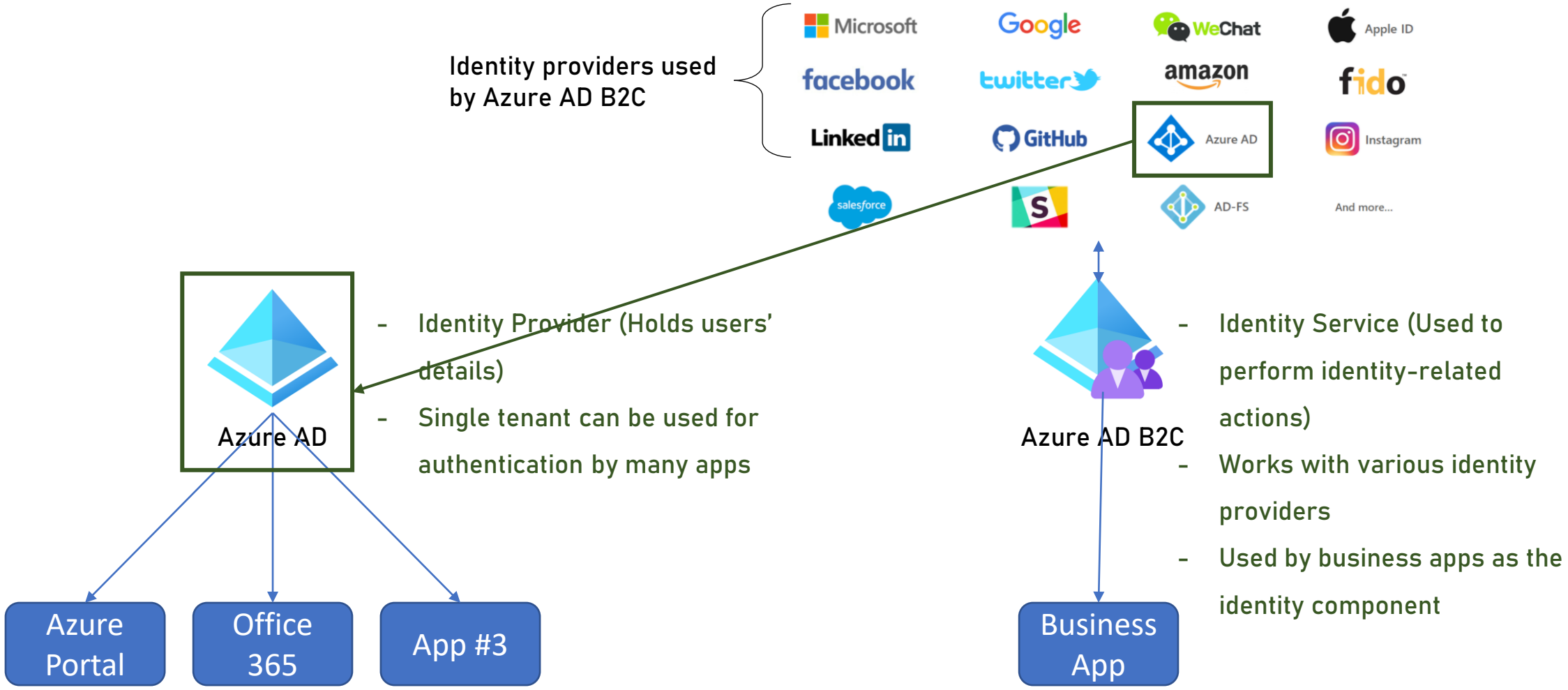
Azure AD B2C

- Identity-as-a-service for your application
- A Business-to-Customer (B2C) service
- Enables integrating identity services in your app
- Works with various identity providers
- Provides various user flows
- Enables customization

Azure AD B2C

- Identity services provided by Azure AD B2C:
  - Sign Up
  - Sign In
  - Log Out
  - Reset Password
  - And more…

Azure AD B2C vs Azure AD



Authentication Features

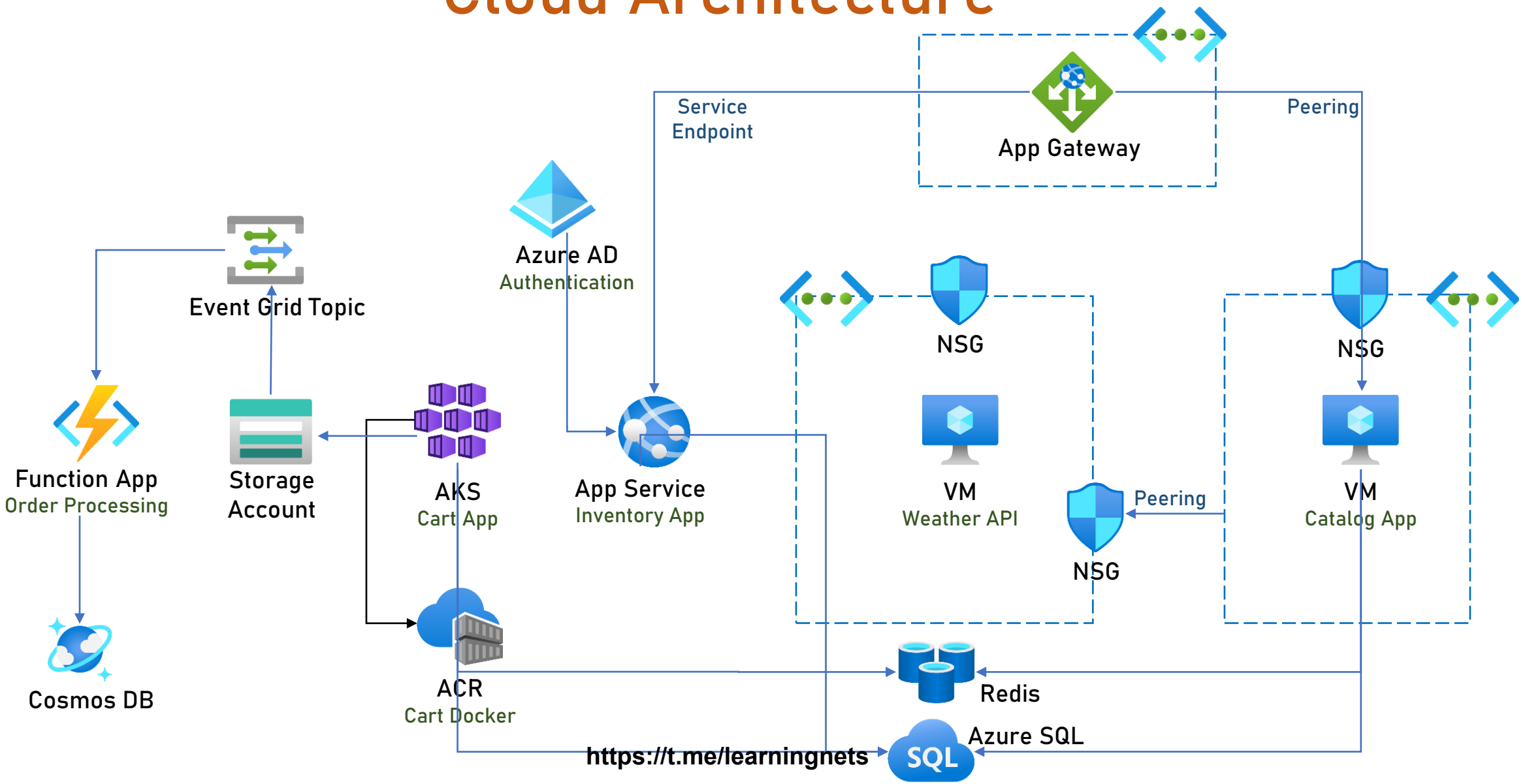
- MFA
- Conditional Access
- Audit Log
- Custom policies
- Custom pages
- And more…

Authentication Features

- Quite complex to set up
- A lot of moving parts
- We won't demonstrate it…

ReadIt!

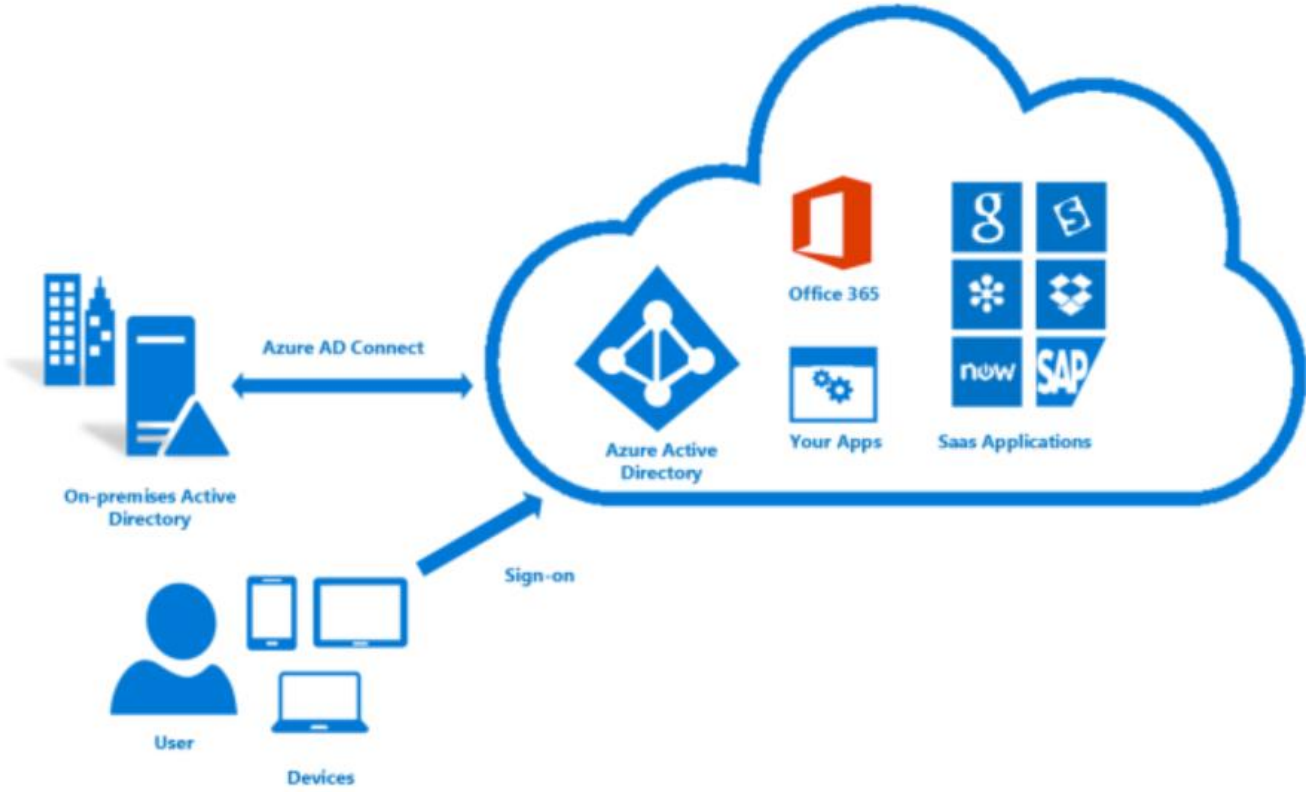
Cloud Architecture



Syncing Azure AD with On Prem

- Many organizations want to sync their on prem Active Directory with Azure AD
- Useful when the organization has apps on prem and in cloud and wants to have a single user base

AD Connect



Authentication with AD Connect

Password Hash Sync	→ The passwords are copied to Azure AD, Authentication happens in the cloud
Pass-Through	→ Passwords stay on-prem, Azure AD passes data to on-prem for validation