Let's build a modern API for your frontend

Go to www.menti.com with code 5327 5345

Bart Lannoeye – Glenn Versweyveld

August 18, 2021

ae.be

ae architects for business & ict

Who's speaking?

Bart Lannoeye

- Technical Architect & Owner @ SanITy BV
- Principal Consultant @ AE
- Twitter: @bartlannoeye

Glenn Versweyveld

- Technical Business Analyst @ Reynaers Aluminium
- Twitter: @depechie

Who are you?

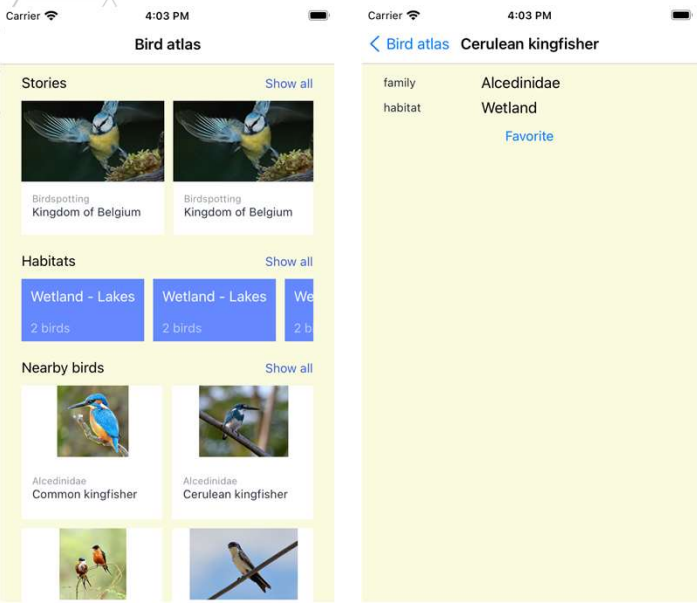
No unicorn solution

Inspirational tips, tricks & good practices for building Enterprise-quality APIs
Shortcuts taken for demo clarity ☺

ae architects for business & ict

Our use case: BirdAtlas

Simple bird information app



architects for business & ict

High level agenda

1. Initial API
2. DevOps
3. Moving towards an enterprise API
4. Security

ae architects for business & ict

What is an API?

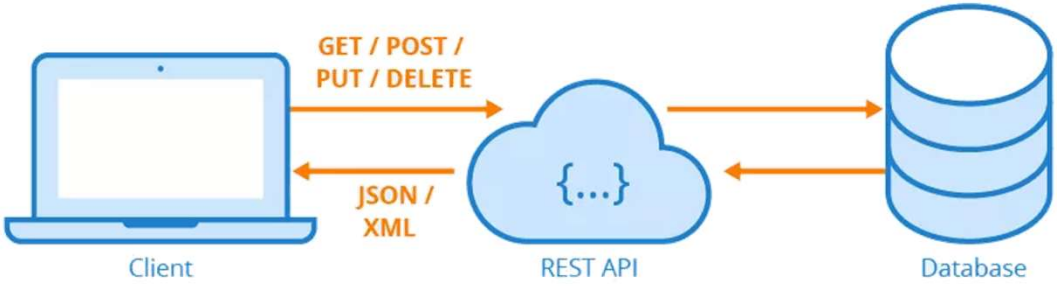
Application Programming Interface

- Software intermediary that allows two applications to talk to each other (= **contract**)

- Modern API
  - Adheres to standards (e.g. REST)
  - Easily accessible, developer friendly
  - Documented
  - Versioned

  ⇒ Designed for consumption

  - Own SDLC

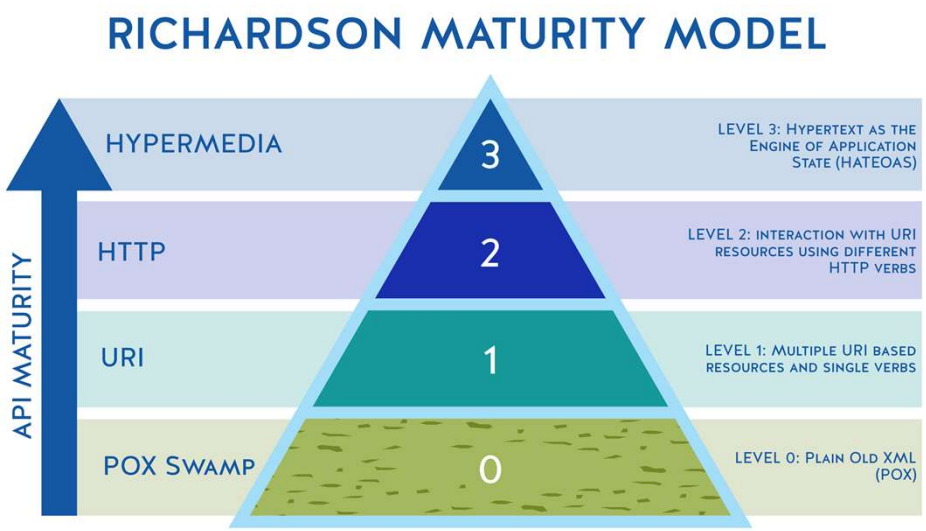


ae architects for business & ict

Standards: REST

Representational state transfer

- Architectural constraints:
  - Client-server architecture
  - Statelessness
  - Cacheability
  - Layered system
  - Code on demand (optional)
  - Uniform interface
- Following constraints = RESTful API

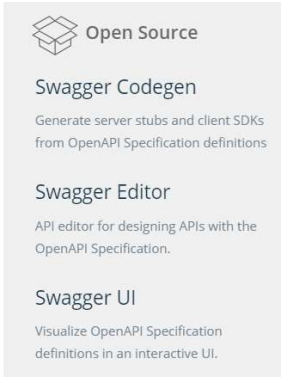


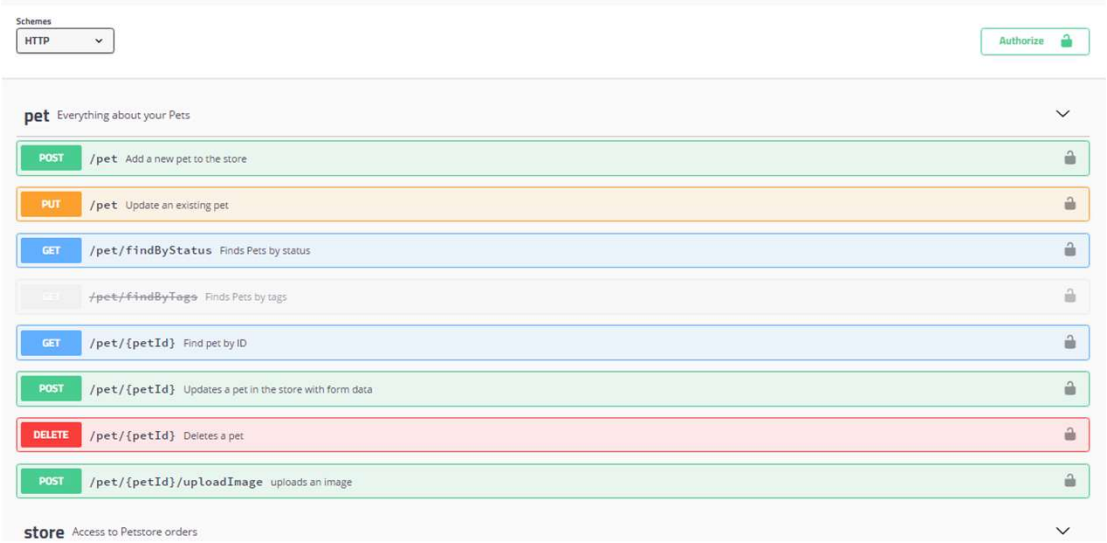
source: devopedia.org

Documentation: OpenAPI

a.k.a. Swagger

- Community-driven specification
- Current version: 3.1.0
  - https://github.com/OAI/OpenAPI-Specification/blob/main/versions/3.1.0.md
- Wide range of tooling
- .NET (
- Visual





1a

Building an API: getting started

Use the tooling to your advantage.

ae architects for business & ict

Tooling used

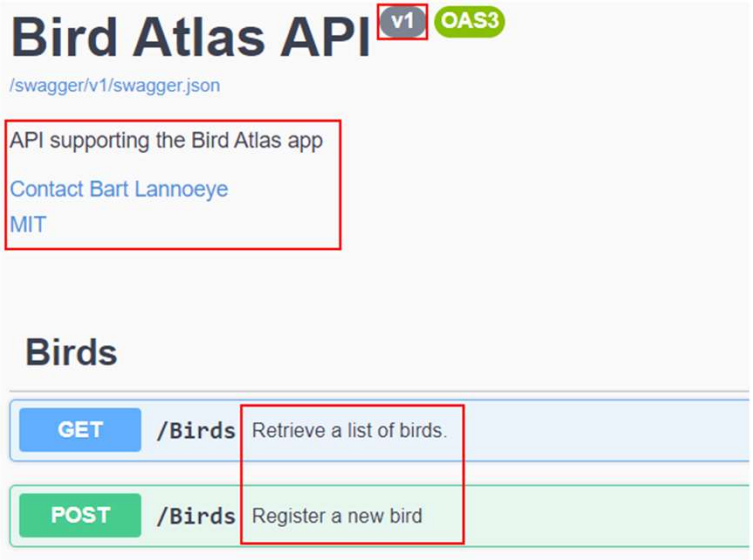
- IDE: Visual Studio 2019 (Community Edition or higher) or Visual Studio Code
  - .NET 5 / C#
- Microsoft Azure

- Note that most principles are available on other platforms as well (coding & hosting)

Demo: Creating an API

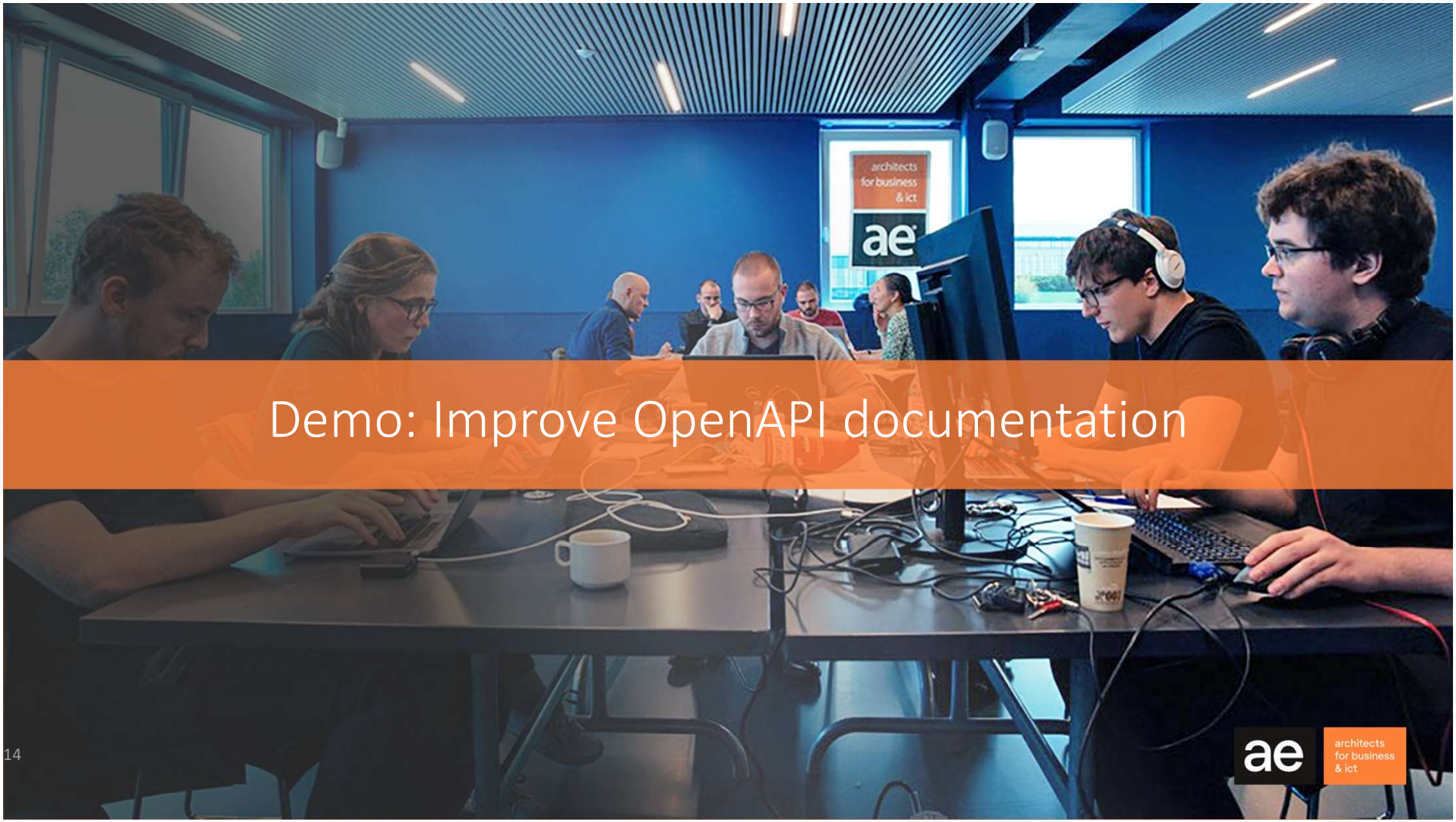
Improving your OpenAPI documentation

- Your OpenAPI specification is often the only available documentation
- Make sure it's easy to find
  - Default URL or even API root
  - *Developer portal*
- Add documentation on methods and resource models
  - Including required fields, formatting, ...
- Add API information (contact info, license, ...)
- *Add versioning*



ae architects for business & ict

Demo: Improve OpenAPI documentation



1b

Hosting your API

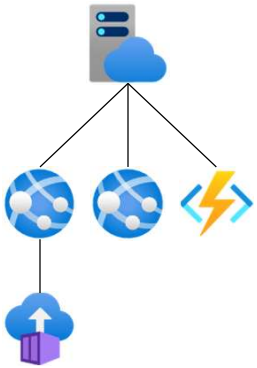
Never right-click deploy, except during demos

ae architects for business & ict

Hosting options: Azure

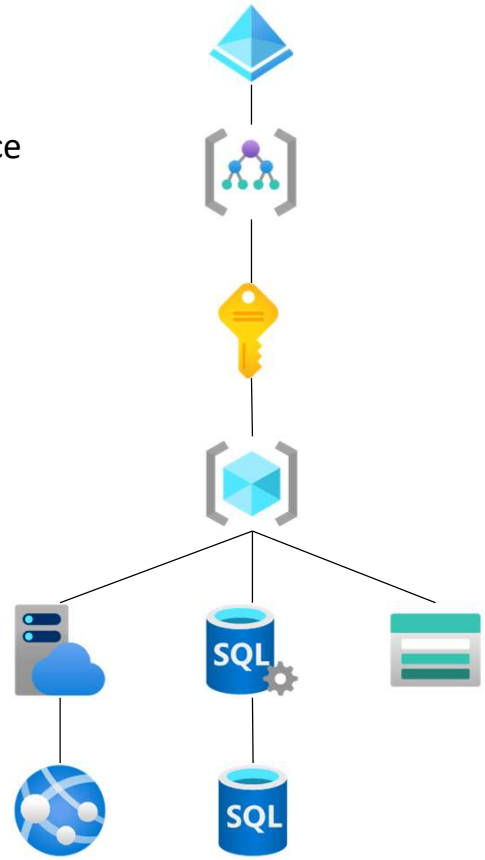
- Host on-prem (IIS, Windows Service, Linux)
- Host in the cloud (Azure)
  - VM with IIS
  - App Service (Plan)
  - Container

- See App Service Plan as a ‘managed cloud IIS’
- App Services run in a Plan
  - Managed solution
  - Security
  - Load balancing
  - Autoscaling
  - Able to host containers



Other Azure resources for our solution

- Hierarchy used for Azure Governance
- Actual resources



architects for business & ict

Demo: hosting in Azure

ae architects for business & ict

1c Consuming your API

ae architects for business & ict

Consuming API

- OpenAPI tells you the resource URI and resource format (= content)
  - Define your models, or generate code
- Write code to do HTTP requests, quite often using a service called HttpClient
  - Or use generated client code
  - Or use a library wrapping HttpClient, e.g. Refit
- Code generation
  - Pro: a lot less work
  - Con: less flexibility or work correcting the generation (manually/automated)

Demo: Calling the API from Postman

Consuming API

.NET MAUI : Multi-platform App UI

- New client side framework in preview
  - Will replace Xamarin Forms
  - Targets iOS, MacOS, Android, Windows from a single code base
- Installation: https://docs.microsoft.com/en-us/dotnet/maui/get-started/installation
  - .NET 6 preview 7
  - XCode 13 beta
  - Maui check tool (https://github.com/Redth/dotnet-maui-check)
  - VS 2022 for Windows preview or VS Code on Mac (with .NET cli to build and run)
    - https://egvijayanand.in/2021/04/11/net-maui-debug-with-comet-in-vs-code/
    - dotnet build -t:Run -f net6.0-ios /p:_DeviceName=:v2:udid=B8557B92-0841-4AE2-B841-E5CFBC85E220
- Links
  - https://github.com/CommunityToolkit/Maui
  - https://github.com/jsuarezruiz/xamarin-forms-to-net-maui

Demo: Consume with mobile .NET MAUI app

1d

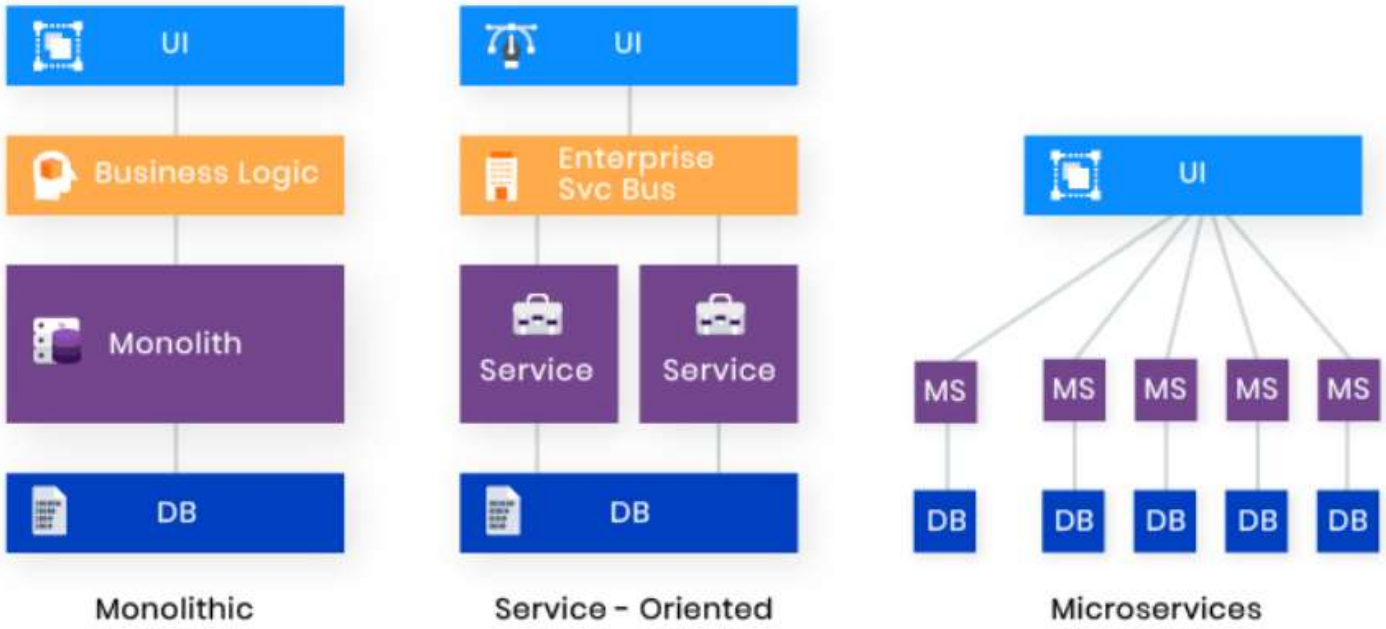
Alternative architectures

Having an impact on your APIs

ae architects for business & ict

Application architecture

Monolith vs SOA vs Microservices

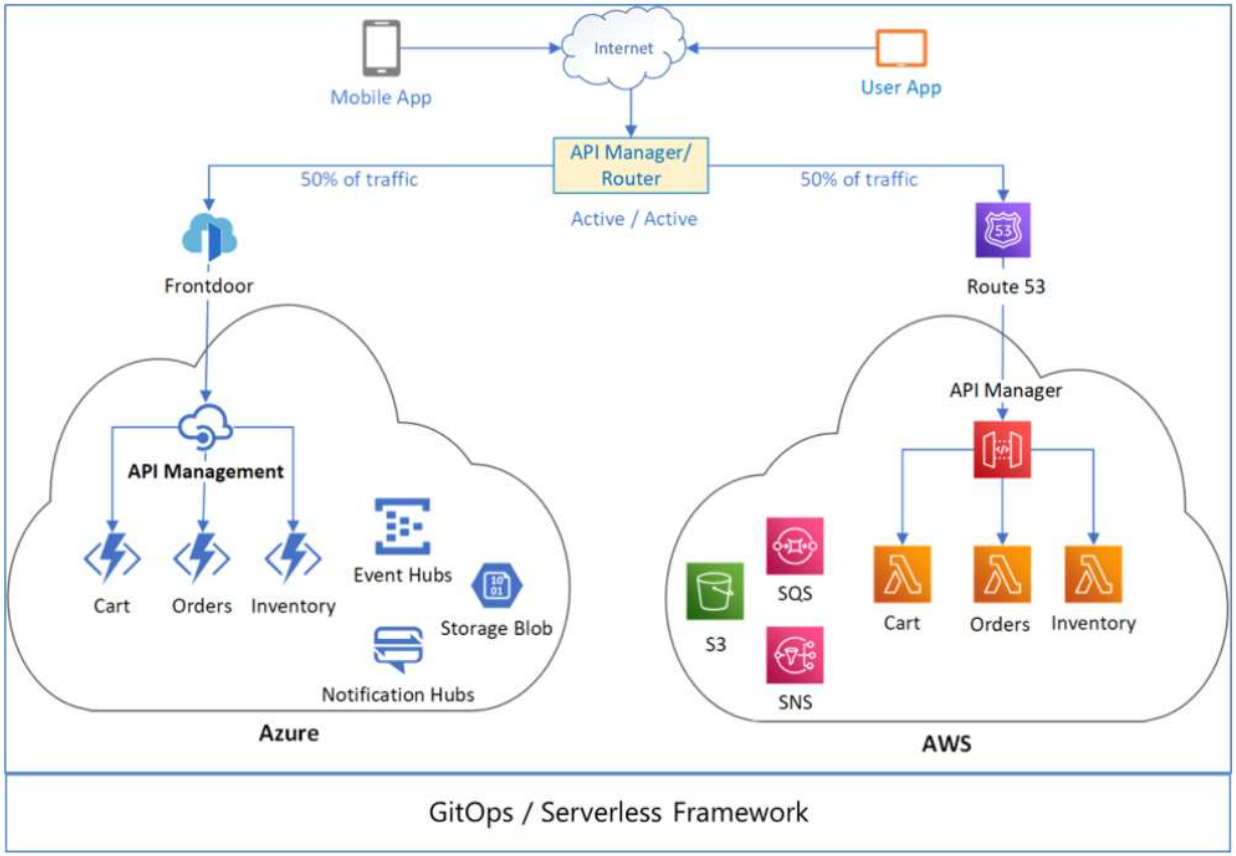


source: rubygarage.org

ae architects for business & ict

Application architecture

... vs Serverless



source: Microsoft docs

Application architecture: infrastructure

Possible options (on Azure), not rules

Architecture	Hosting	Data	Extra
Monolith			
SOA			
Microservices			
Serverless			

Application architecture: implementation

- Multi-layer: UI – API surface (controllers) – Business services (managers) – Repositories – Data
- CQRS (Command and Query Responsibility Segregation)
- Domain driven
- Event driven
- Plenty of variations, combinations & other options

- Protocol:
  - REST over HTTP is not made for internal service communication (but it works)
  - Look at options like gRPC (https://github.com/grpc/grpc-dotnet)

Architecture key takeaways

- All previous designs (architecture, infrastructure, implementation) are valid options
- Choose based on your needs (functional, non-functional) and your team's skillset
- Combine where useful!
  - Monolith core with edge services
  - Multi-layer CRUD with task based CQRS for other parts in the app
  - Synchronous core with asynchronous components

- Tips:
  - Dare to go outside comfort zone if necessary
  - Fail fast as part of your learning process

2 Automating deployments

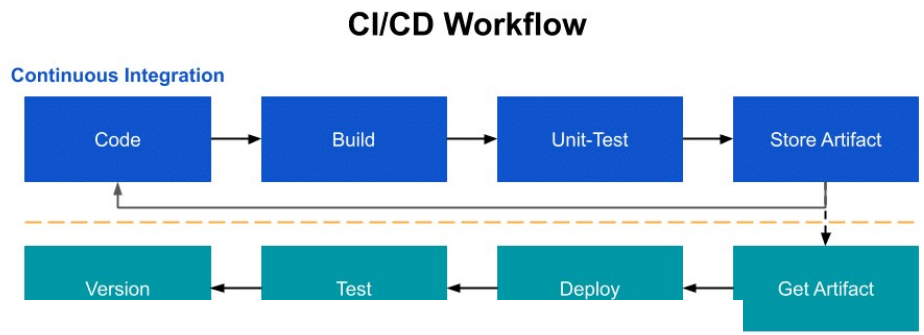
Like we said before: never right-click deploy!

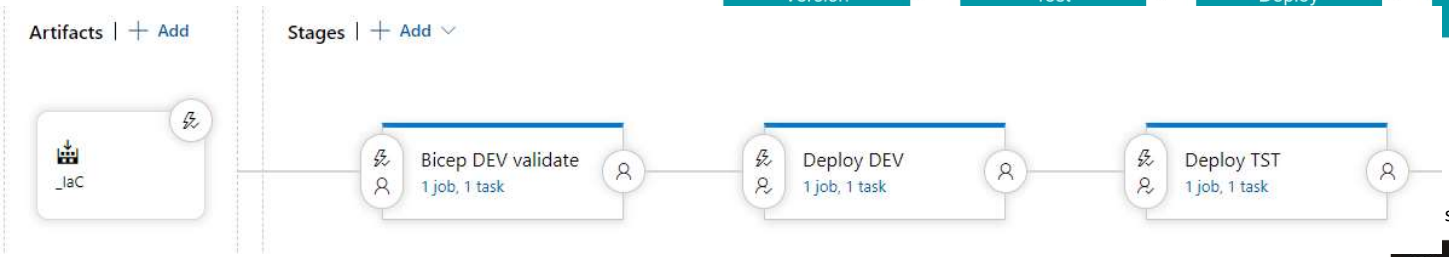
ae architects for business & ict

Benefits of automated deployments

- Able to deploy more often with smaller code changes
  - Test reliability
  - Fault isolation

  => Reduced cost
  - Faster time to market
- Codebase consistency between environments
  - (Re)deploy a single artifact
  - Use settings/environment files for variables





source: opsani.com

Automate your infrastructure: IaC

Manage infrastructure through machine-readable definition files

- Tackle bugs/failures caused by environment drift with configuration consistency
  - Reduce risk of manual intervention
  - Versioned
- Enable spinning up another cloud environment with a ‘single click’
- Are documentation of your infrastructure
- Bring security
  - Deploy with service principal
  - Easily implement security standards (reusable components, linter, ...)
- Approaches:
  - Declarative: ‘what’ should my target configuration be
  - Imperative: ‘how’ do I change my infrastructure to meet my desired state

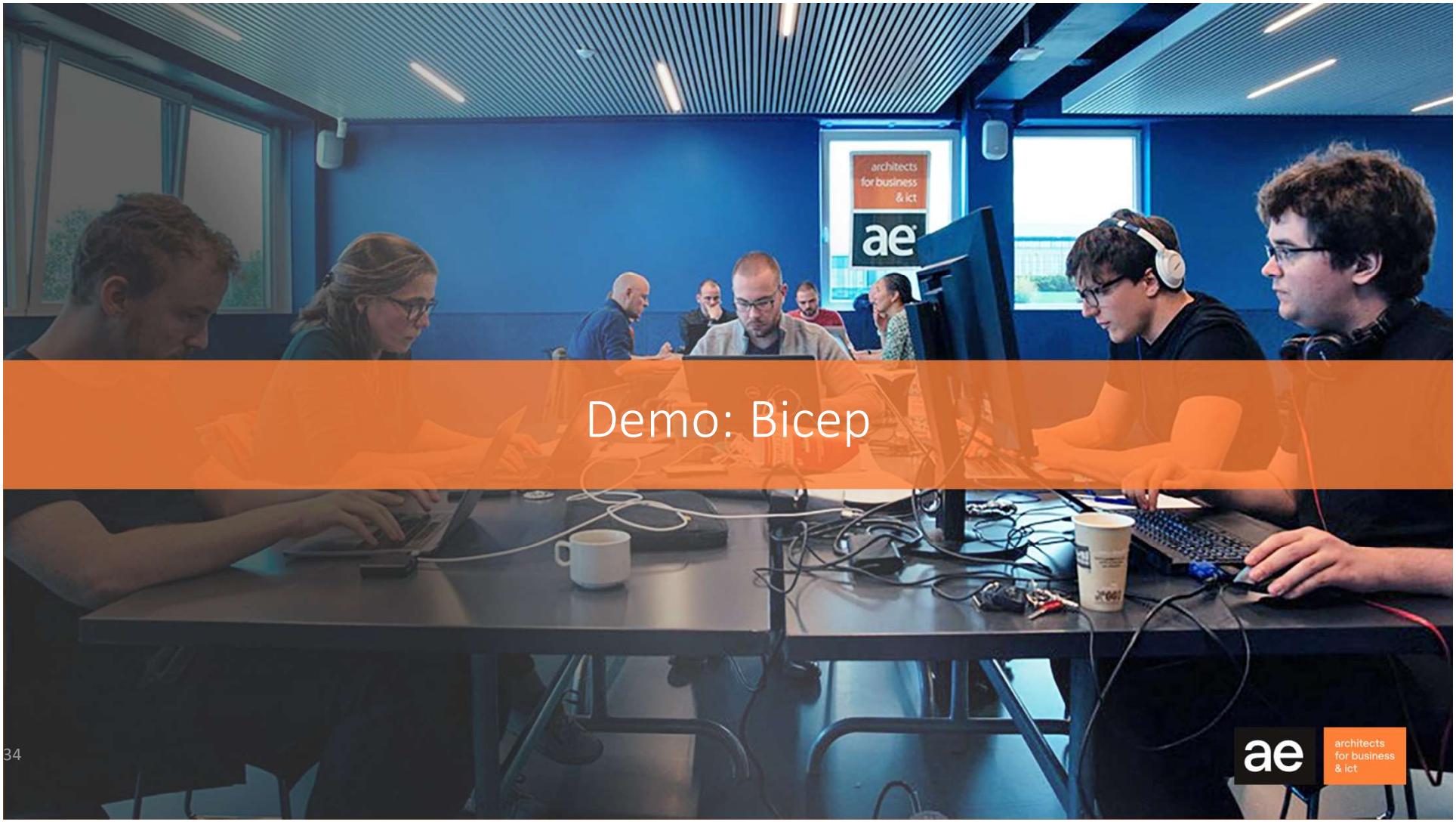
Terraform CHEF ANSIBLE Pulumi

ae architects for business & ict

Bicep

Next generation ARM templates

- Domain specific language for Azure resources
- Tooling:
  - Bicep CLI
  - Bicep VS Code Extension
- Deploy:
  - Azure CLI
  - Azure PowerShell
  - CI / CD
- Why? Azure is your 'state file'
- Links:
  - https://github.com/Azure/bicep
  - https://aka.ms/learnbicep
  - https://bicepdemo.z22.web.core.windows.net/ (Bicep Playground)



Demo: Bicep

ae architects for business & ict

Bicep file structure

Organize your code

```
param environmentType
param resourceNameSuffix
param storageAccountSkuName
param sqlDatabaseSkuName

var storageAccountName
var sqlDatabaseName

resource storageAccount
resource sqlDatabase

output storageAccountBlobEndpoint
```

```
param environmentType
param resourceNameSuffix

param storageAccountSkuName
var storageAccountName
resource storageAccount
output storageAccountBlobEndpoint

param sqlDatabaseSkuName
var sqlDatabaseName
resource sqlDatabase
```

Demo: Bicep modules

ae architects for business & ict

3a

Moving towards an enterprise-quality API

You're not the only API in your company.

ae architects for business & ict

API versioning

- Crucial part of API design
  - Iterate without breaking client applications
- Possible through URI, query parameters, headers or content negotiation

  Example: https://api.mydomain.com/**v1**/birds/{id}

- Code:
  - Microsoft versioning NuGet package
  - ApiVersion attribute
  - Decide how to deal with versioned code files (folders, namespaces, duplicate code, partial classes, …)

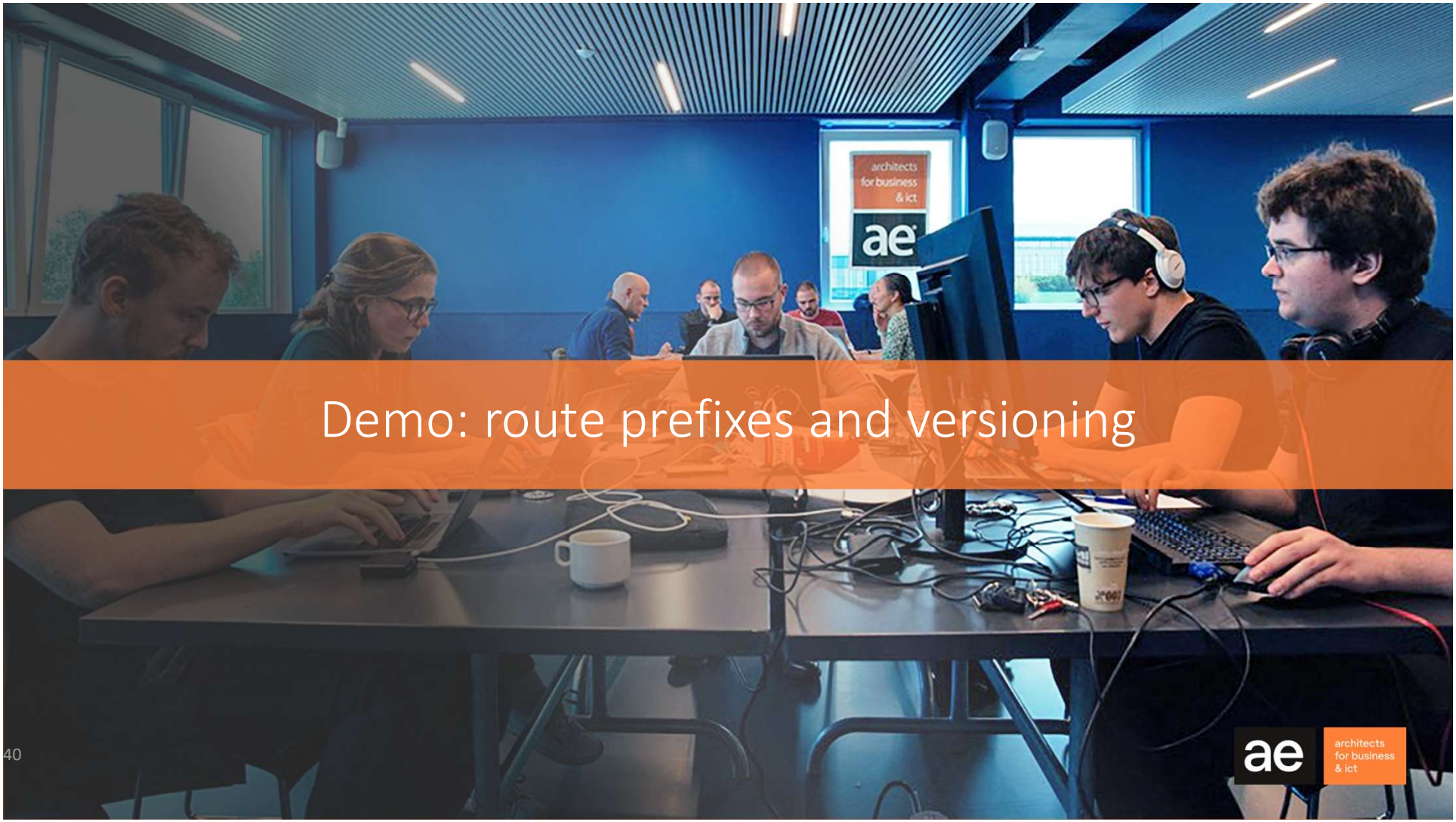
Flexible URI Routing: route prefixes

- Quite often you have multiple APIs to host, preferably on the same domain
- Maybe you want to split your APIs in business domains
- But keep URI segment behind version number clean
  - Not: https://api.mydomain.com/v1/atlas/animals/birds/{id}

- Example: https://api.mydomain.com/**atlas/**v1/birds/{id}

- Note:
  - Version number can be part of the route prefix, reusing route infrastructure
  - Solutions like API Management can replace this 'fix'

Demo: route prefixes and versioning



Integrated validation

- Basic property validation can easily be done with data annotations
- But what with more complex validation logic?
- Solution preferably integrates with ASP.NET Core request lifecycle so you don’t need to validate on each method

- Possible solution: FluentValidation NuGet package

Demo: integrating FluentValidation

Exception handling & Logging

Reactive insights

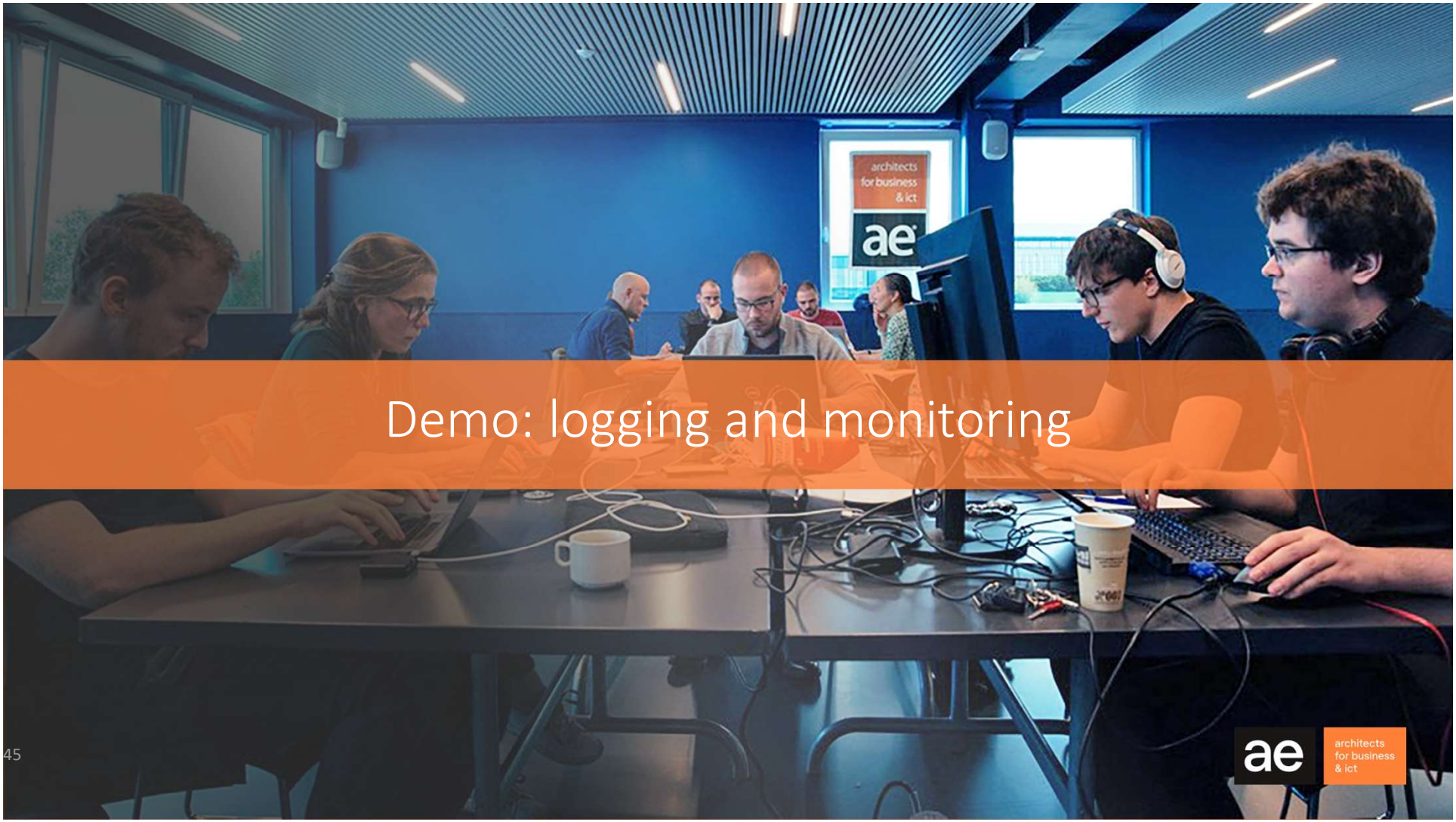
- Deploying to the cloud brings challenges
  - Often harder to debug
  - PaaS/FaaS doesn't allow RDP to check what goes wrong
  - Paas/Faas often can't log to local disk
- Need a way to troubleshoot, without exposing too much info to the client

Health checks & Monitoring

Proactive insights

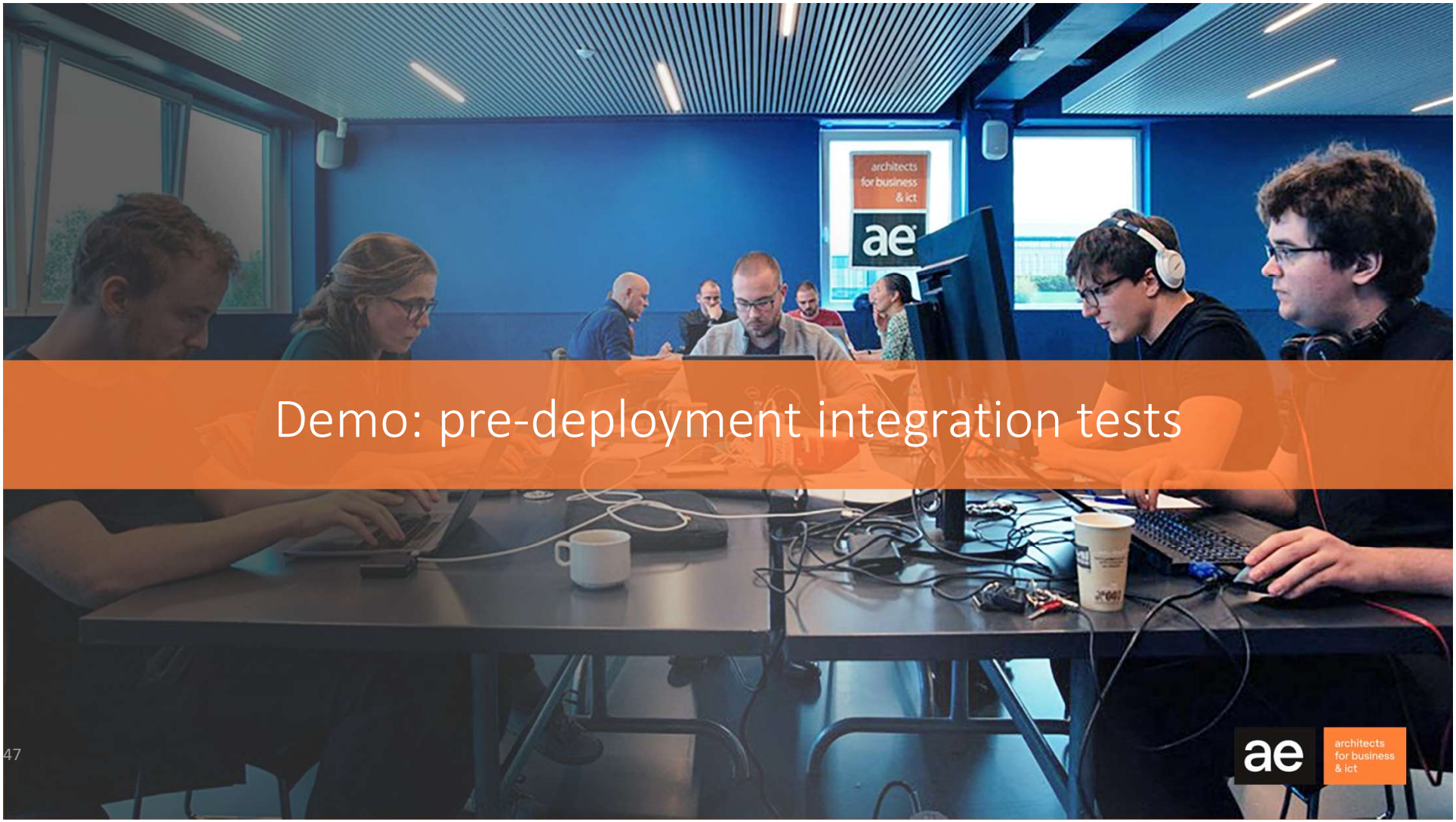
- Be notified of service failures before the user calls you
  - Both your own API and its dependencies
- Turn metrics and checks into alerts

Demo: logging and monitoring



Testing your code

- .NET (Core) supports IoC through DI
  - Ideal for unit testing with mocking
- Sometimes you want integration testing as well
  - Preferably **before** deployment
- ASP.NET 5 provides *WebApplicationFactory* for integration testing
  - Modify service registration: e.g. in-memory database



Demo: pre-deployment integration tests

ae architects for business & ict

3b

Moving towards an enterprise API consumer

APIs can break, your user shouldn’t feel the pain.

ae architects for business & ict

Circuit breaker / Retry pattern

- https://github.com/App-vNext/Polly
- A resilience and transient-fault-handling library
- Works with configured policies
  - Retry
  - Timeout
  - Fallback
  - Circuit breaker: https://martinfowler.com/bliki/CircuitBreaker.html

    = wrap your API call with a circuit; once an error threshold has reached, return an exception before even going out to the backend

Demo: Circuit Breaker / Retry pattern

4 Adding security

Security is important! Hire an expert (or become one).

Keeping your credentials safe

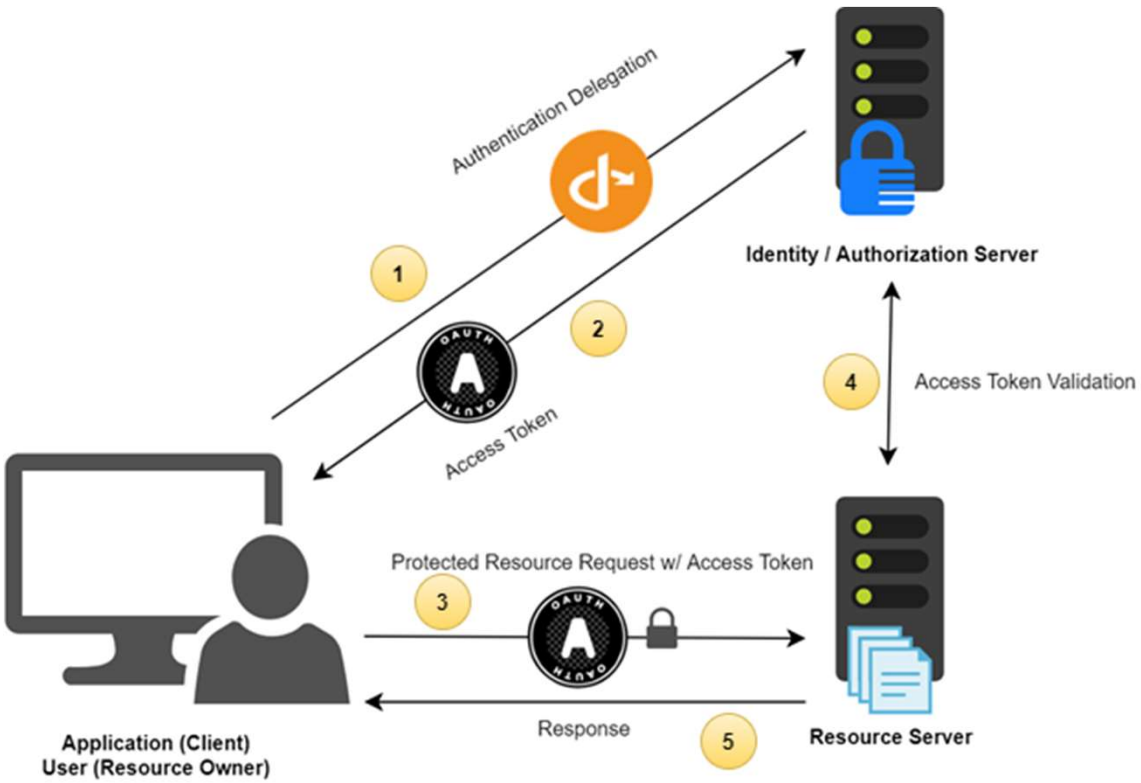
- On-prem solutions often have credentials in config files for connection strings, service logins, …
  - If lucky encrypted, more often clear text
- On-prem / private repos often have credentials in the repository
- Bots crawl public repositories for credentials
  - Data leaks
  - Steal compute power

- Integrate Key Vault by using Managed Identity
  - During deployment (IaC / Release pipelines)
  - During runtime

Application Security

- Protect your API endpoints
- Bonus: Assign roles to your users

- Protocol:
  - OpenID Connect (AuthC)
  - OAuth2.0 (AuthZ)
- Technology:
  - Azure Active Directory
  - AAD B2C
  - Auth0
  - KeyCloak
  - …



source: c-sharpcorner.com

Application Security: Using AAD

- Protect API: https://github.com/AzureAD/microsoft-identity-web
- Consume a protected API: MSAL
  - https://docs.microsoft.com/en-us/azure/active-directory/develop/msal-overview

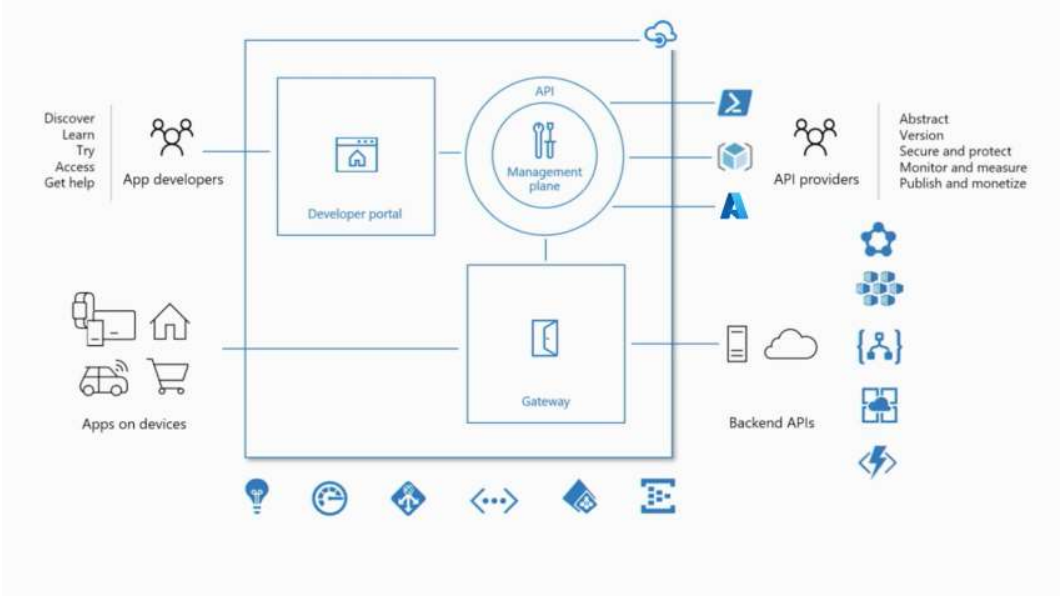
- Problem: protecting API = nothing gets in, not even Swagger UI

Demo: using Azure Active Directory

API Management

Centralizing control of your APIs

- API gateway: single point of entry
  - AuthC/AuthZ
  - Rate limiting / quotas
- API lifecycle management
  - Design, publish, deploy # versions
  - Define transformations
- Developer portal
- Analytics / Monitoring
- Bonus: API monetization



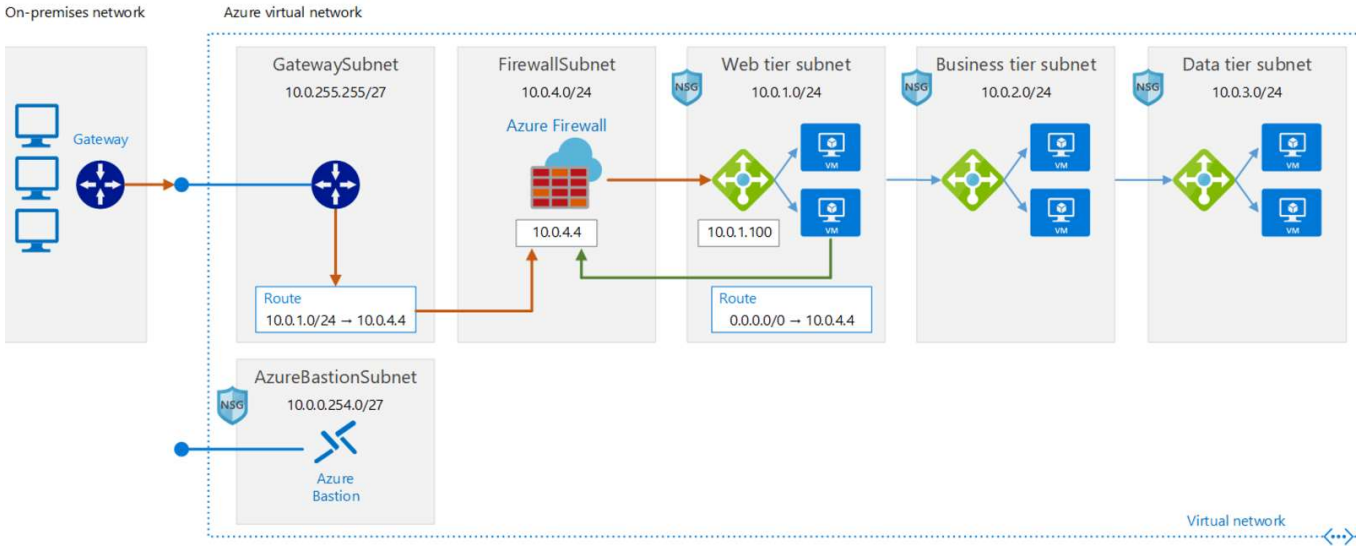
source: microsoft.com

Demo: API Management

Cloud Infrastructure Security

Secure Azure Development = Networking

- Azure Virtual Network (VNET) is similar to a traditional network that you'd operate in your own data center
  - Network Security Groups, Firewalls and subnet routing
  - Connect on-prem with Site-2-Site VPN, ExpressRoute
- Expose your secure endpoints through FrontDoor, ApplicationGateway, API Management

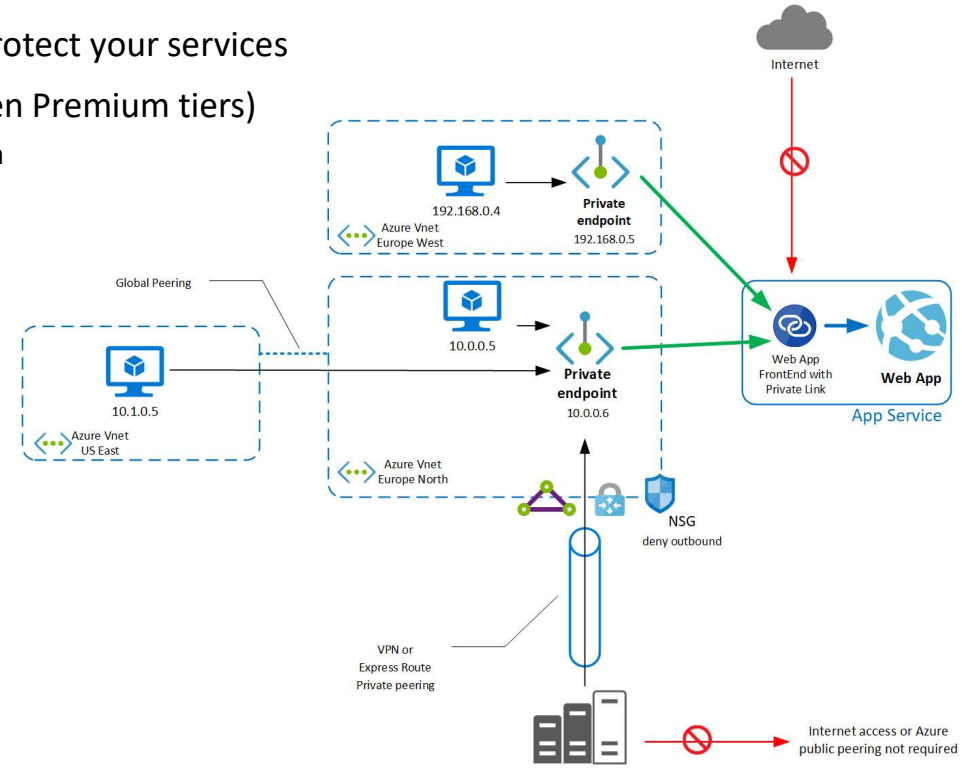


source: azureguru.org

architects for business & ict

VNET integrations

- No use in setting up a Virtual Network if you don't protect your services
- VNET integration (=security) comes with a price (often Premium tiers)
  - Cheapest hosting: Private Endpoints & VNET integration
  - More expensive: App Service Environment (ASE)
  - Don't forget other services, e.g. Azure Service Bus



source: azureguru.org

Demo: VNETs

ae architects for business & ict

5 Wrap-up

ae architects for business & ict

There are more topics to keep in mind

Full day workshop

- End-to-end tracing & debugging (correlation ids)
- Messaging & Event-driven architectures (ASB / EH / EG / FaaS)
  - Data consistency & transaction management
- Data (SQL/MongoDB)
- DevOps
  - Secret management
  - Deploy into VNET
- ...

- More time for hands-on

.NET tooling

Don’t reinvent the wheel (if you don’t want to/have time for/money for)

- API documentation (OpenAPI): Swashbuckle / NSwag
- API consumption: Refit / RestSharp / ServiceStack / Flurl / ...
- CQRS: Mediatr / EventFlow / CQRS.NET / ...
- Azure Service Bus: Azure Messaging SDK / MassTransit / NServiceBus / ...
  - Other Azure Services: https://github.com/Azure/azure-sdk
- Microsoft Azure Storage Explorer: https://azure.microsoft.com/en-us/features/storage-explorer/
- Using Azure Active Directory / AAD B2C: MSAL

Presentation materials

- Slides on Code PaLOUsa GitHub repo
- Code on https://github.com/SanITy-BV

- Twitter: @bartlannoeye

- Twitter: @depechie

Related sessions

- Thursday 1.30 PM: Building better security for your API platform using Azure API Management
- Thursday 1.30 PM: But It was Logged! Practical Logging & Monitoring with .NET Core
- Friday 8.30 AM: What is new in .NET 6 and the future of .NET
- Friday 11 AM: Practical Unit Testing Patterns With .NET Core

Thank you!

Feedback on to www.menti.com with code 5327 5345

Bart Lannoeye – Glenn Versweyveld

August 18, 2021

ae.be

ae architects for business & ict