# INFRASTRUCTURE AS CODE

# Eelco Koster

Lead Azure Architect

@eelco_zelf

http://eelco.azurewebsites.net

## Over Eelco

Meer dan 15 jaar IT ervaring in software ontwikkeling.
5 jaar ervaring in ontwikkelen en ontwerpen voor het Microsoft Azure Platform.

## Mijn werk

- Presentaties
- Training
- Azure Architect (PaaS)
  - Migraties
  - Nieuwe ontwikkeling
  - DevOps

# Doel

Waarom zou je IaC willen?

Mogelijkheden laten zien

Tools en hoe je zelf aan de slag kunt

# Why Automation?

Repetitive tasks are boring…

Azure is highly automated

- Service healing
- DevOps

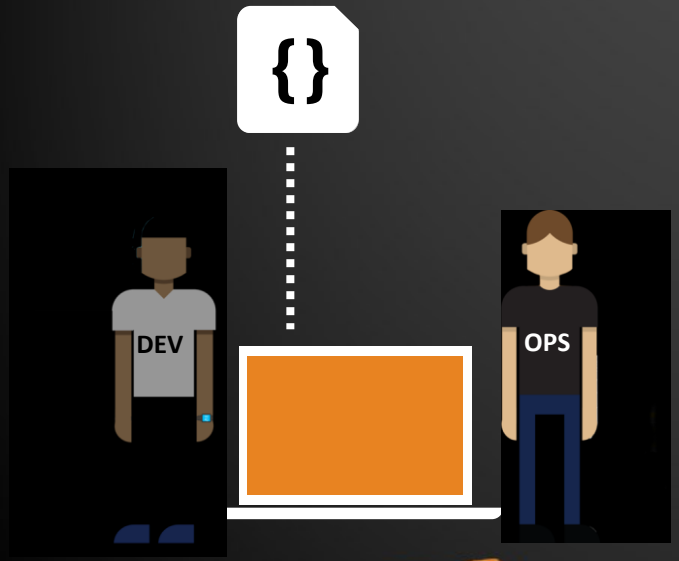
Time to provision full environments

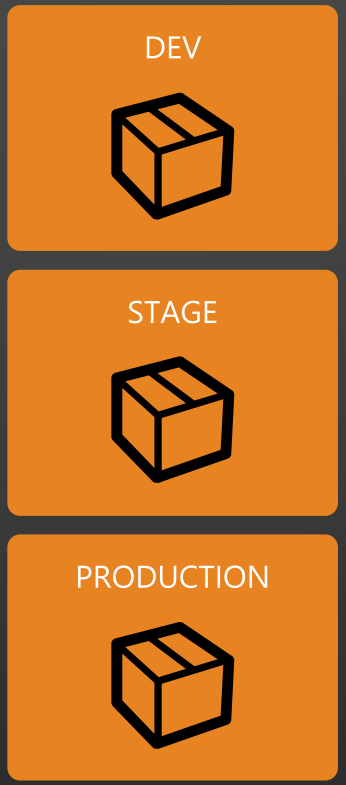
- Compute, storage, database, etc

Deployment to multiple environments/geographies

- Change only configuration / parameters

# Infrastructure as Code?



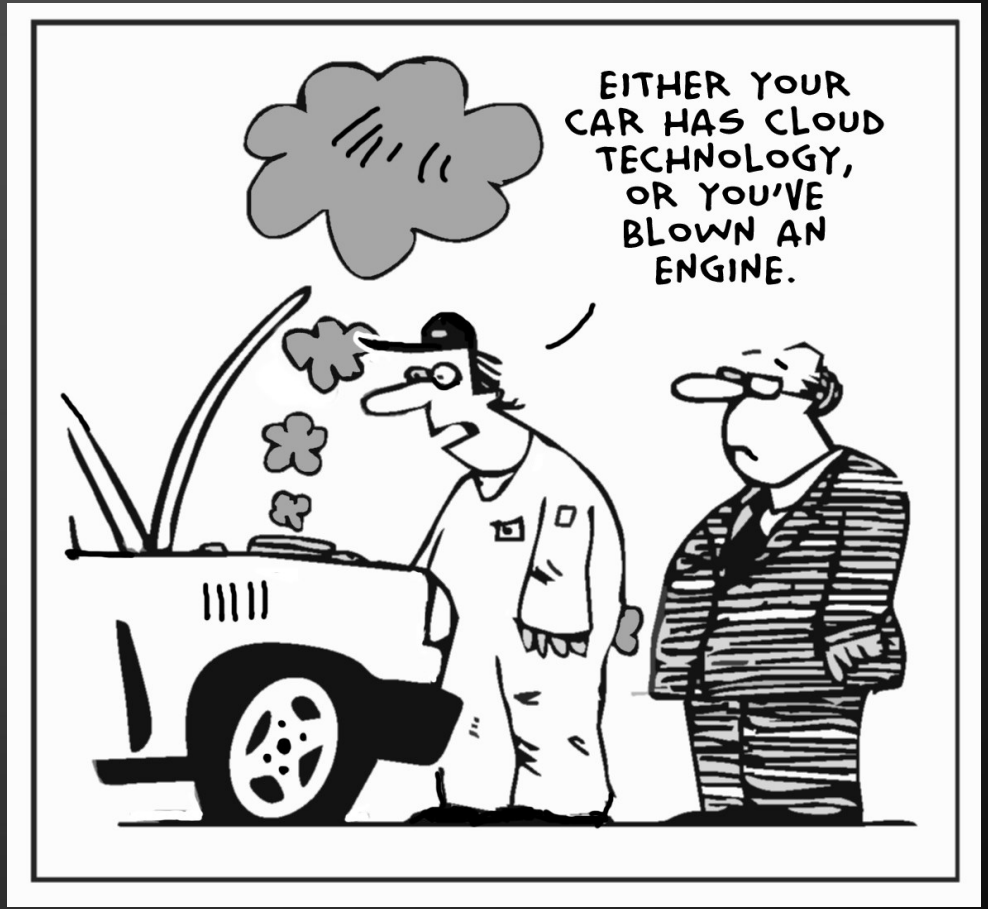




Better collaboration between development and IT operations

## Value

- Optimized Resources
- Accelerate Delivery
- Less Configuration Errors
- Source Control

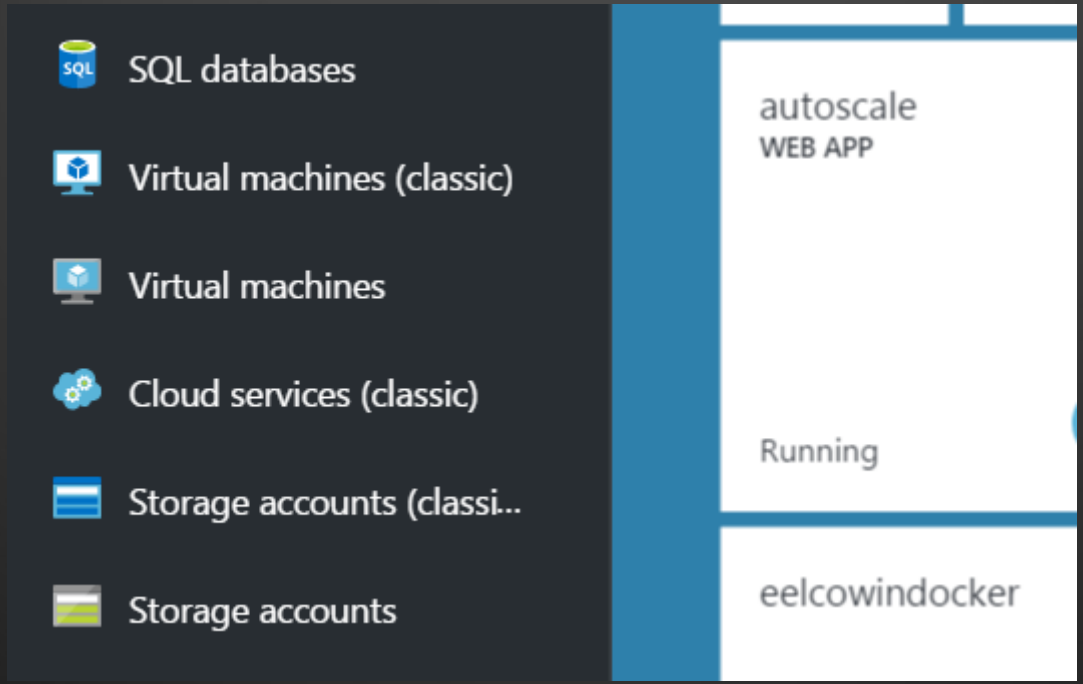
# Under the hood



# Azure Resource Manager

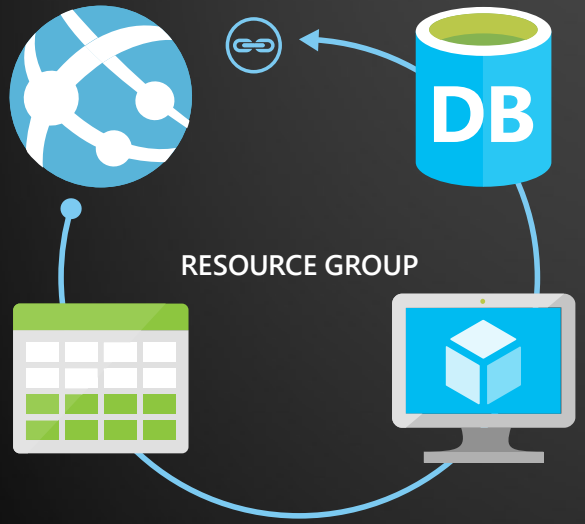
- Introduced at //Build 2014
- New deployment model
- Deep integration with New Portal
- Make use of Resource Groups

# Classic vs ARM



# Azure Resource Group

## Grouping your environment

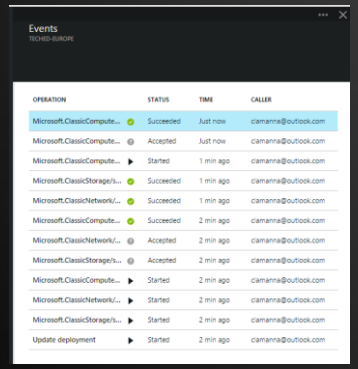



- → container for multiple resources
- → resources exist in one* resource group
- → resource groups can span regions
- → resource groups can span services

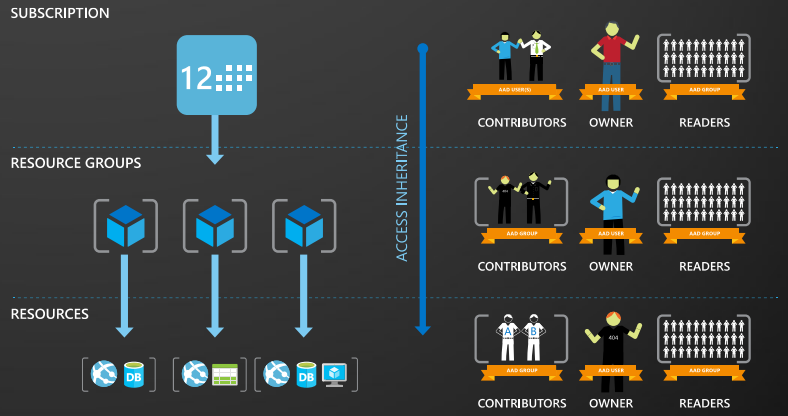
*and only one

# Resource Group

- Lifecycle: deployment, update, delete, status
- Grouping: metering, billing, quota, UX (portal, PowerShell, CLI)
- Access Control: scope for RBAC permissions
- Identity: resources can link to each other







# Resource Manager Templates

*Automate your deployments*

+

# PowerShell

Automate your work

Interactively at a command prompt
Automatically through scripts

Install Microsoft Azure PowerShell by running Microsoft Web Platform Installer

# PowerShell

Automate your work

- Login

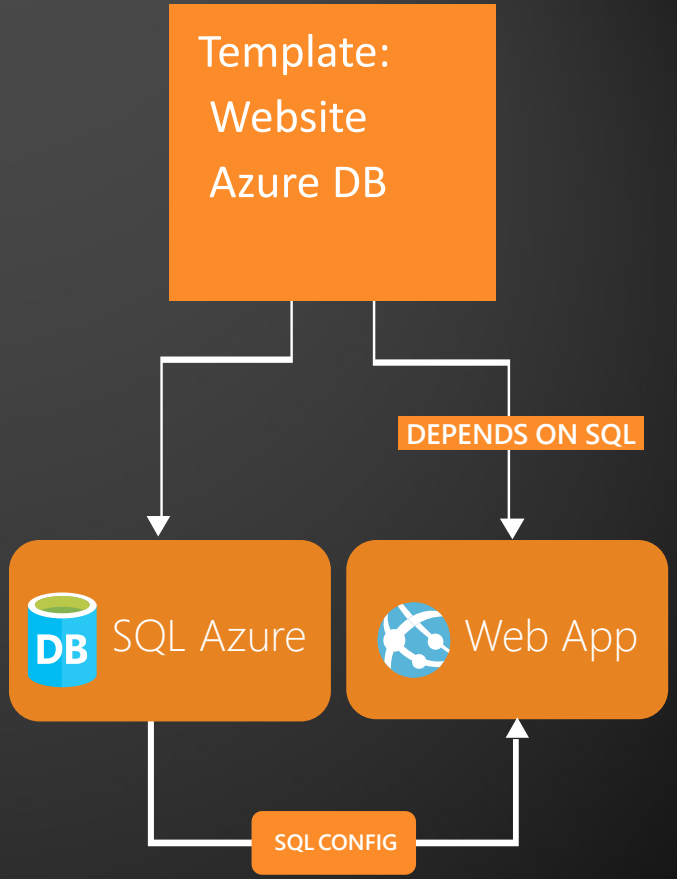
  ```
  Login-AzureRmAccount
  ```

- Different type of commands ASM / ARM

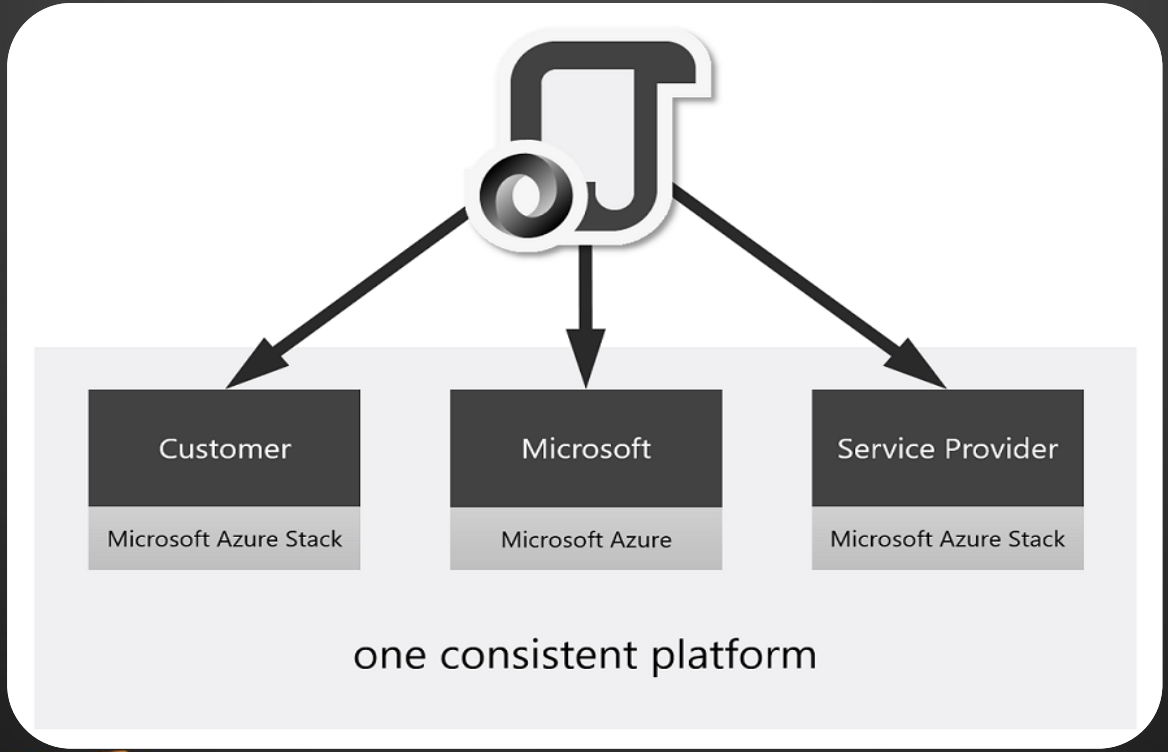
  ```
  Get-AzureVM
  Get-AzureRmVM
  ```

# Resource Templates

- JSON format
- Declarative, model based specification of resources and their configuration, code, and extensions
- Idempotent
- Consistent deployment
- Source file, checked-in
- Parameterized input/output



# Create once, deploy everywhere



# Anatomy of Resource Templates

Json file, containing 4 main segments

1. Parameters
2. Variables
3. Resources
4. Outputs

# Resource Templates

```
"$schema": "http://schema.management.azure.com/schemas/json",
"parameters": {
    "name": {
        "type": "string"
    },
    "location": {
        "type": "string",
        "defaultValue": "West Europe"
    },
    "environmentName": {
        "type": "string",
        "allowedValues": [
          "test",
          "prod"
    ]}
```

# Resource Templates

```
"variables": {
   "environmentSettings": {
       "test": { "storageType": "Standard_LRS" },
       "prod": { "storageType": "Standard_GRS" }
   },
   "currentEnvSettings":
    "[variables('environmentSettings')[parameters('environmentName')]]"

 }
```

# Resource Templates

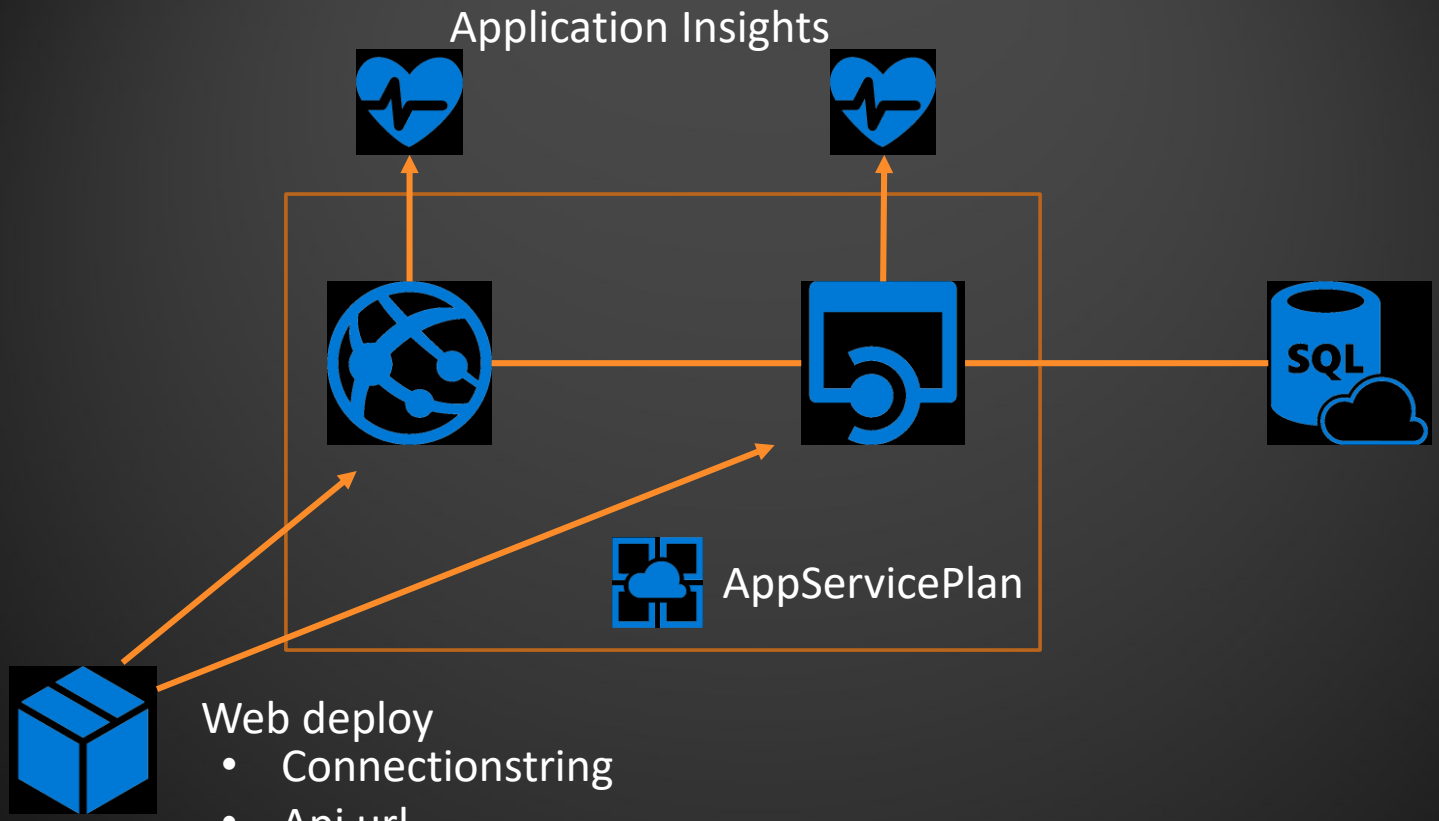
```
"resources": [
  {
     "apiVersion": "2016-01-01",
     "type": "Microsoft.Storage/StorageAccounts",
     "name": "[parameters('name')]",
     "location": "[parameters('location')]",
     "dependsOn" : [ ],
     "tags" : {"deployedBy" : "Eelco Koster" },
     "properties": {
         "accountType": "[variables('currentEnvSettings').storageType]"
     }
  }]
```

# Resource Templates

```
"outputs": {
      "storageKey" : {
      "type" : "string",
      "value": "[listKeys(
         resourceId('Microsoft.Storage/storageAccounts',
         parameters('name')), '2016-01-01').primaryKey]"
    }
 }
```

# Demo

## Azure Resource Manager


Application Insights
SQL
AppServicePlan
Web deploy
• Connectionstring
• Api url
• Application insights key

# Getting Started

- Automation Options in Create
- Export Template from previous deployments
- Export Template from running Resource Group

# Demo

## Portal features

# Resource Policy

- Cloud enables agility
- How can you maintain control?
- Resource Policies allow us to govern acceptable resource configurations

# Demo

## Resource Policy

# Resource Locks

Accidents happen. Resource locks help prevent them ☺

- Resource lock
  - Policy which enforces a "lock level" at a particular scope
- Lock level
  - Type of enforcement; current values CanNotDelete and ReadOnly
- Scope
  - The realm to which the lock level is applied. Expressed as a URI; can be set at the resource group, or resource scope

```
New-AzureRmResourceLock -LockLevel CanNotDelete -LockName
"production lock" -ResourceGroupName Production
```

# Tools

- Visual Studio
- http://resources.azure.com
- http://armviz.io/