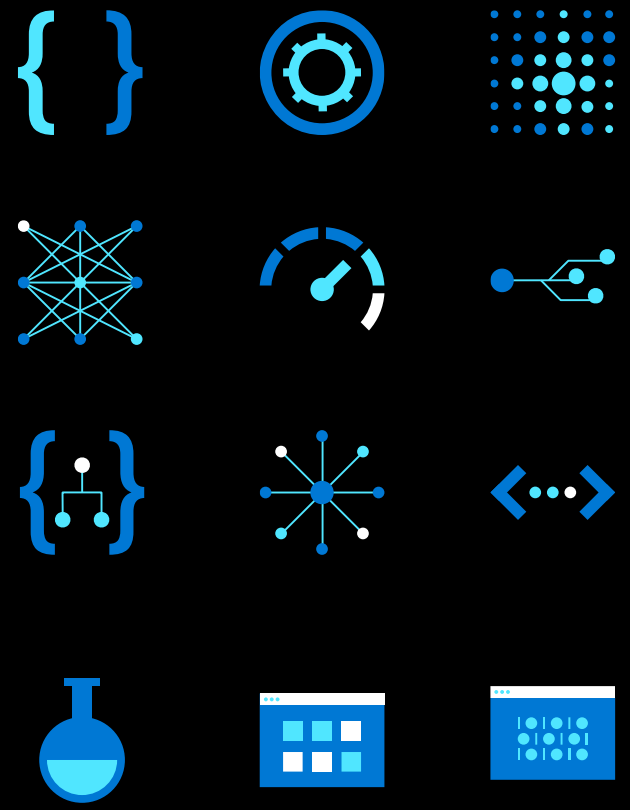
Microsoft Azure

# Azure Training Day

## Migrating web applications to Azure



Microsoft

Azure

# Azure VM lift and shift

## Part 2 of 5 in the Migrate web apps to Azure series

MSUSDev@Microsoft.com

Sep, 2019

# About us…


## Ryan Berry

- Principal Cloud Solution Architect

## Matthew Calder (macalde@microsoft.com)

- Sr. Product Marketing Manager – Azure, US Marketing and Operations


For questions or help with this series
MSUSDev@Microsoft.com

For the lab guides and sample code
https://github.com/MSUSDEV/Migrating-web-apps-to-Azure

Efficiently Deploying and Migrating Virtual Machines to Azure

# Lift & Shift Migration

# Infrastructure as Code (IAC)

# What is Infrastructure As Code (IAC)

http://en.Wikipedia.org/wiki/Infrastructure_as_Code

- The process of **managing and provisioning** computer data centers through machine-readable **definition files**.
- The deployment can use **either scripts or declarative definitions**, rather than manual processes.
- While the terminology points to Infrastructure, it **should not be confused with Infrastructure as a Service (IaaS)**, as IAC also allows you to deploy other (cloud) components like Platform as a Service (PaaS)

# Infrastructure As Code (IAC) - Values

## 1. FASTER EXECUTION

- Build once – Deploy many
- Minor changes required to deploy different models
- Deployment of compute resources goes faster than manual provisioning

## 2. REDUCE RISK

- Remove Errors and Mistakes
- Remove Security Violations
- Overwrite changes without touching existing deployment state

## 3. REDUCE COST

- Manual labor is expensive
- Easily scale out your environment
- Portable across different environments (dev/test/staging/different organizations)

## 4. INTEGRATION WITH DEVOPS

- Infrastructure as Code provides an integration with DevOps concepts and processes within an organization
- Allows for end-to-end application landscape provisioning, not just ‘infrastructure’

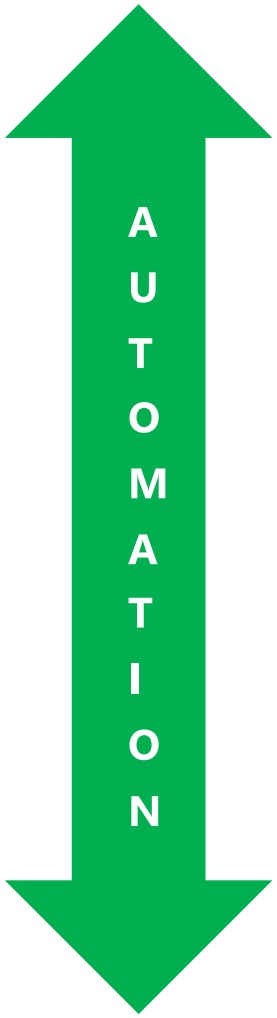
# Infrastructure As Code (IAC) - Methods

## DECLARATIVE (FUNCTIONAL) "WHAT"

- The final state of the system / environment is defined (declared), in such a way that it defines in "what" state it should be.
- When the process is run, it will configure the system/environment to have the declared state as the final result.

## IMPERATIVE (PROCEDURAL) "HOW"

- The automation code used to setup or configure the system / environment is written in such a way, it goes through each configuration step-by-step.
- Automation code built in this way, defines the process of "how" the system is to be configured and what steps need to be taken, in the exact order, to obtain the final result state.



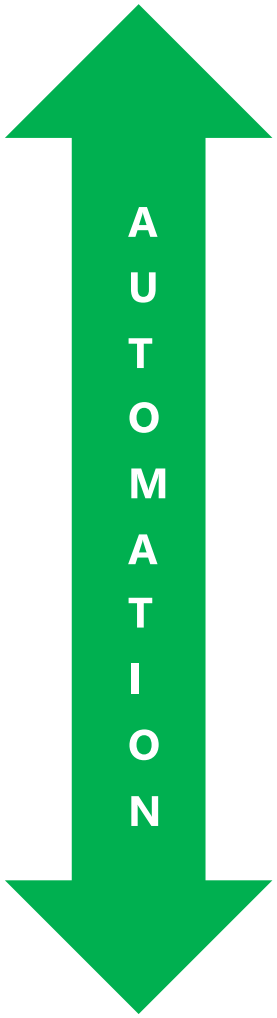
# Infrastructure As Code (IAC) – Configuration Options

## CONFIGURATION ORCHESTRATION

- Designed and used to automate the deployment of servers and other related (cloud) infrastructure
- Some tools have overlap with configuration management

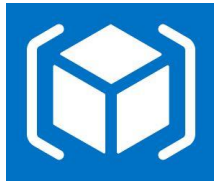
## CONFIGURATION MANAGEMENT

- Designed and used to automate the configuration of systems and software on top of the infrastructure which has already been provisioned (out of configuration orchestration)
- Some tools have overlap with configuration Orchestration



# Infrastructure as Code – Tools (Azure Popular)

## CONFIGURATION ORCHESTRATION





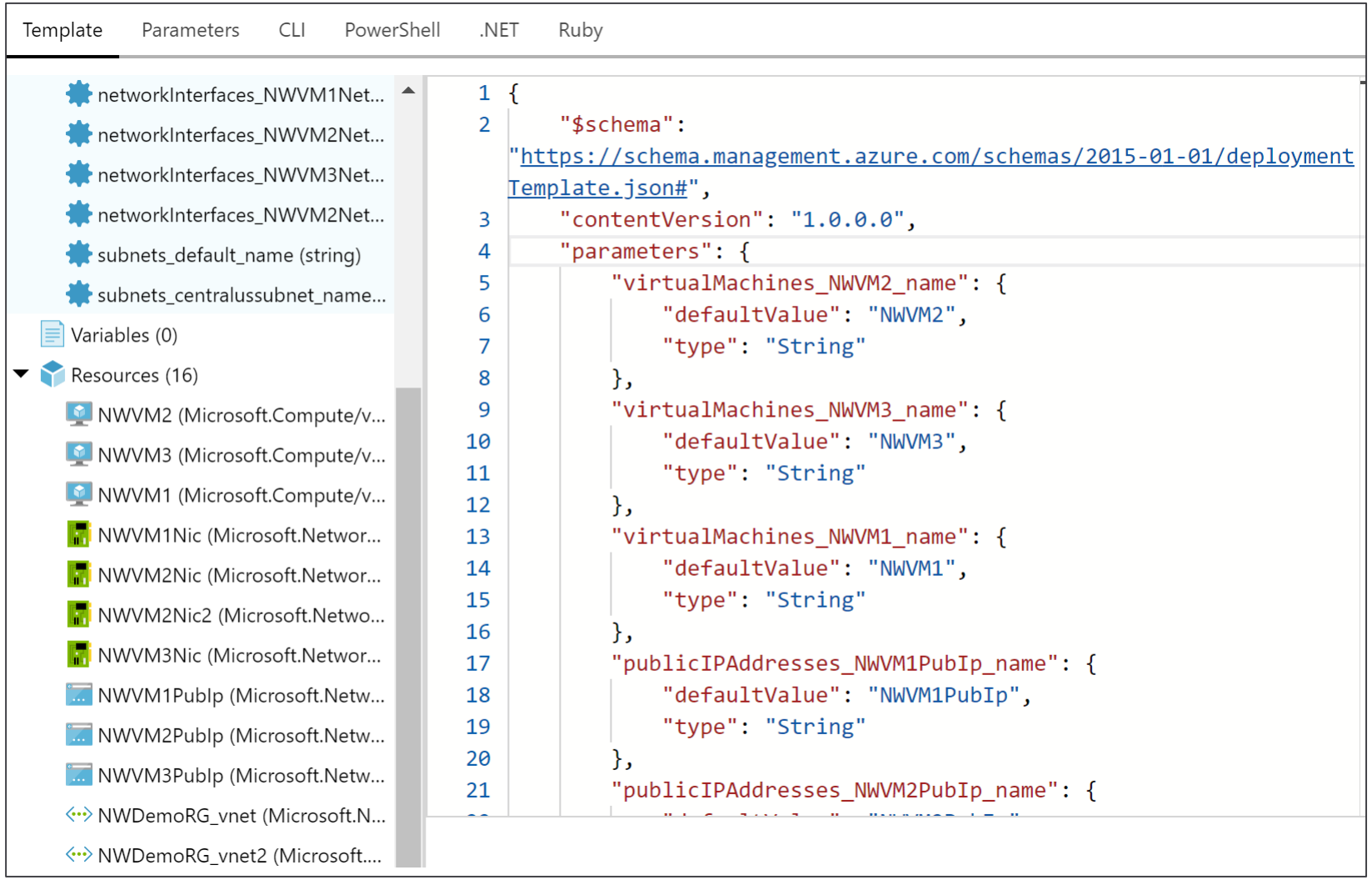
---

## CONFIGURATION MANAGEMENT



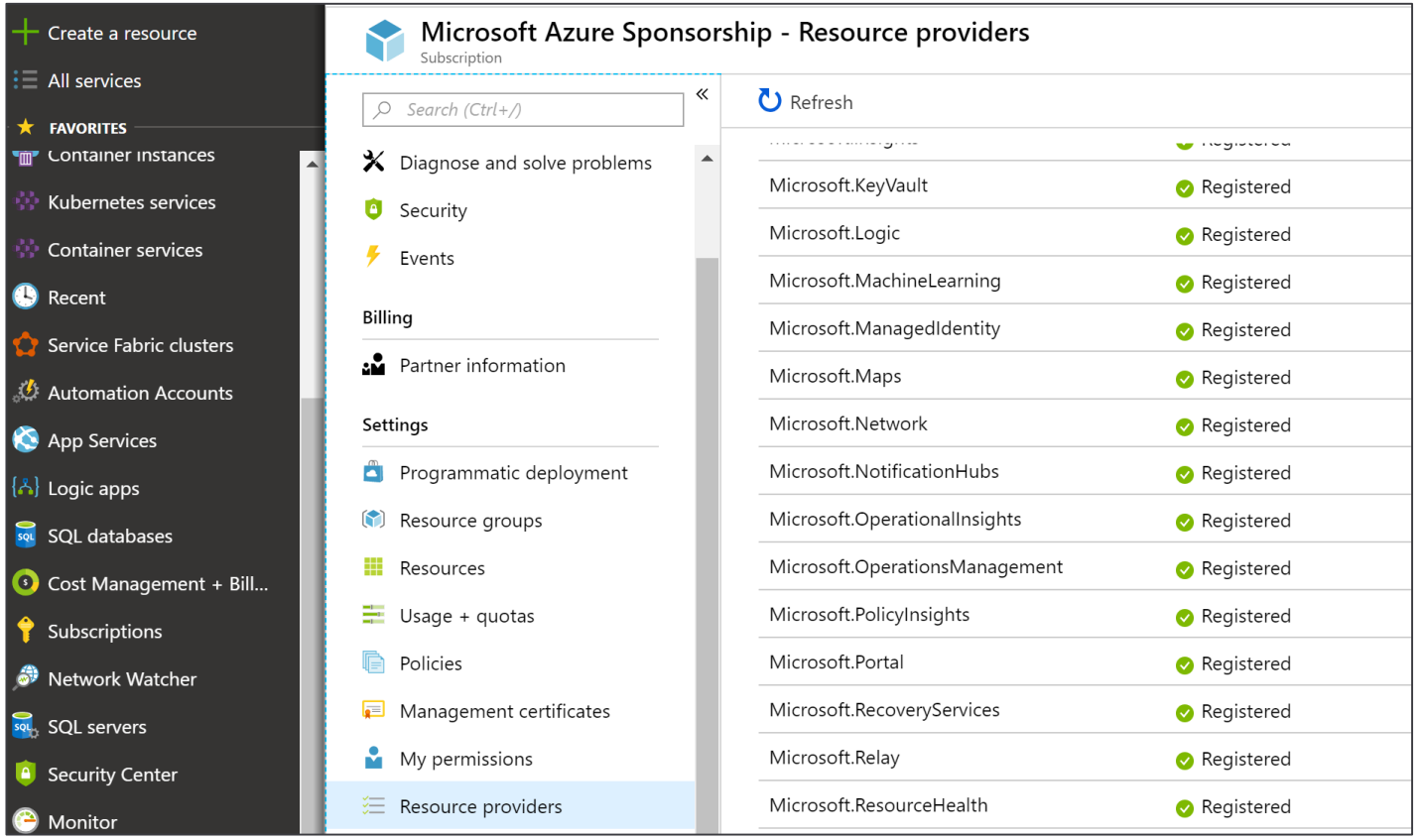
# Azure Resource Manager Templates

**ARM templates are based on JSON syntax**



# Azure Resource Providers

Resource providers registered for use with your subscription can be found in the portal (or via PowerShell, REST API, or CLI).



# Why use ARM Templates?

- Include the configuration of Azure resources in source control (“Infrastructure as Code”)
- Repeat the deployment process numerous times
- Automate deployments
- Employ continuous integration techniques
- Utilize DevOps principles and practices
- Repeatedly utilize testing infrastructure then de-provision it when finished

# Ways to create an ARM Template (*…What works for Peter…)*

1. From the automation script available from the Azure Portal (which is imperfect – more on that in a moment)

2. An ARM template in Visual Studio / Visual Studio Code

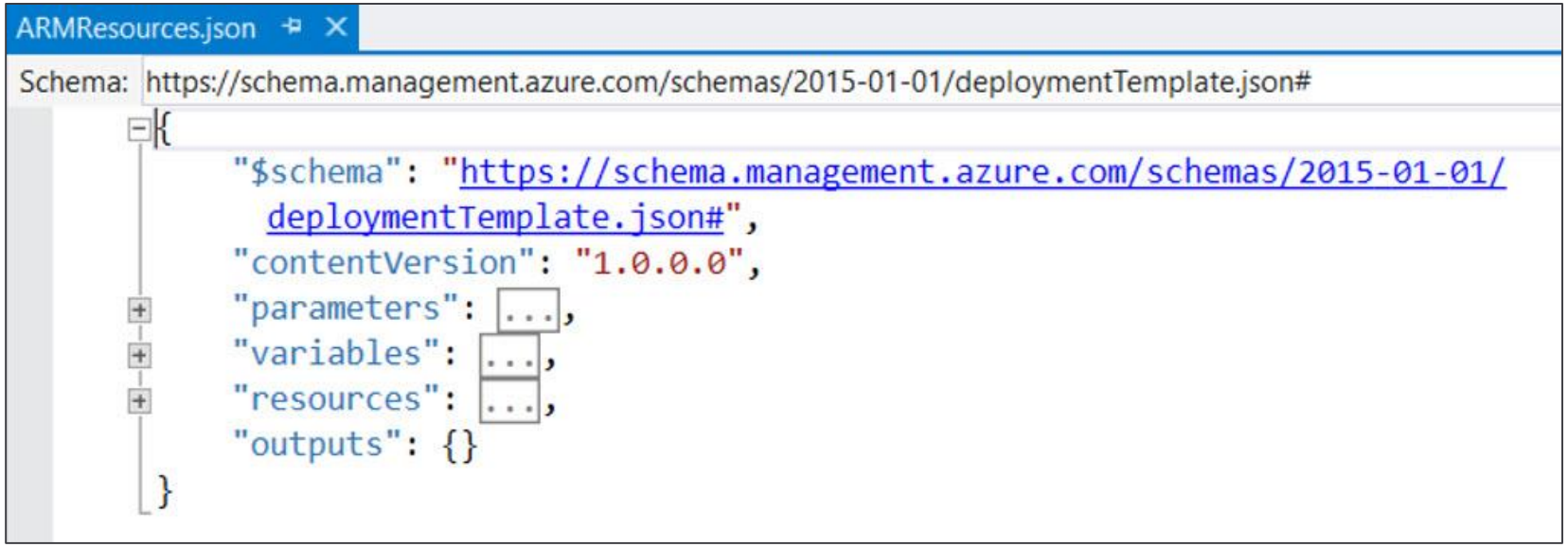
3. QuickStart Templates from GitHub (there's a lot to choose from – the templates that start with 101 are less complex)

4. Create from the ground up

5. Start with a combination of 1 and 2 or 3, customizing the way you like it

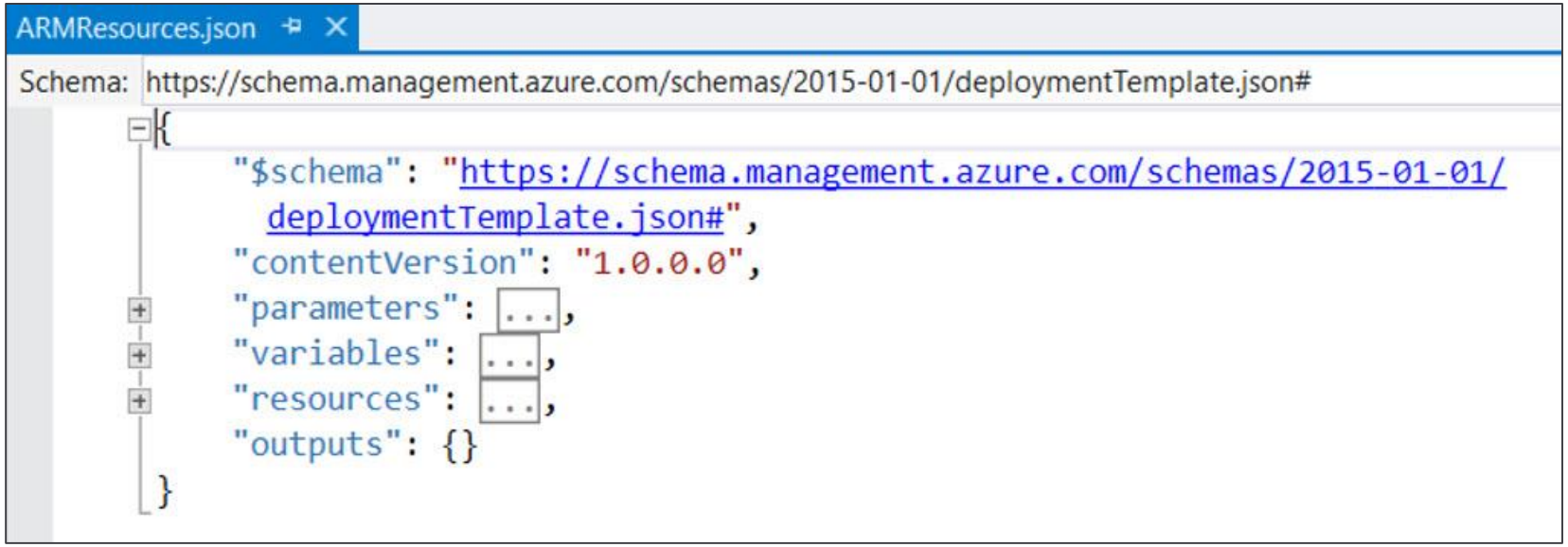
# Basic Template Structure: $schema

This refers to the JSON schema. Note that the schema specified is different in the parameters file vs. the main ARM deployment file.



# Basic Template Structure: Contentversion

You can increment this version if you'd like to manage the changes made over time. The default is "1.0.0.0."



# Basic Template Structure: Parameters

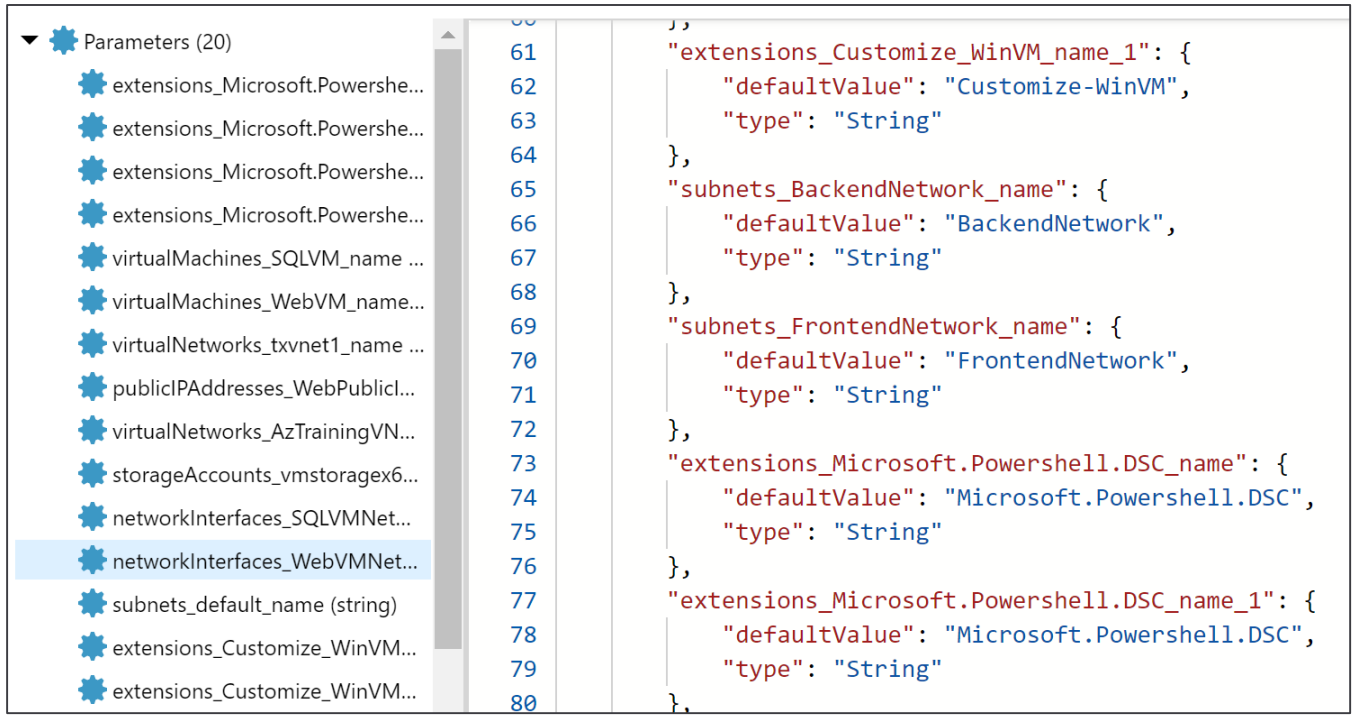
## This is where you store the definitions of all parameters used throughout the ARM template

### Type

Most common types are string or int;
Password = securestring;

### MetaData / Description

Helpful info to remember what it's used for, or any other notes you want to leave for yourself and your team.



# Basic Template Structure: Parameters

This is where you store the definitions of all parameters used throughout the ARM template

## Allowed Values

Provides a list to choose from
(e.g. restrict list of VM Sizes)

## Default Value

Specifies the default value, but leaving choices (e.g. deploy DS2_v2 as VM size, but one can choose another from the list)

```
90          "type": "Microsoft.Compute/virtualMachines",
91          "name": "[parameters('virtualMachines_SQLVM_name')]",
92          "apiVersion": "2017-12-01",
93          "location": "eastus",
94          "tags": {
95              "displayName": "SQLVM"
96          },
97          "scale": null,
98          "properties": {
99              "hardwareProfile": {
100                 "vmSize": "Standard_DS1_v2"
101             },
102             "storageProfile": {
103                 "imageReference": {
104                     "publisher": "MicrosoftSQLServer",
105                     "offer": "SQL2014SP2-WS2012R2",
106                     "sku": "Standard",
107                     "version": "latest"
108                 },
109                 "osDisk": {
110                     "osType": "Windows",
```

# Basic Template Structure: Variables

The variables section contains references to settings, mostly picked up by the resources section later on, making definitions easier

# Basic Template Structure: Resources

The resources section defines each resource to be deployed, with references to parameters and variables as necessary. The elements which are defined vary based on the kind of resource which is being deployed.

```
192     {
193       "name": "[parameters('WebVMName')]",
194       "type": "Microsoft.Compute/virtualMachines",
195       "location": "[resourceGroup().location]",
196       "apiVersion": "2015-06-15",
197       "dependsOn": [
198         "[resourceId('Microsoft.Storage/storageAccounts', variables('vmstorageName'))
199         "[resourceId('Microsoft.Network/networkInterfaces', variables('WebVMNicName')
200       ],
201       "tags": {
202         "displayName": "WebVM"
203       },
204       "properties": {
205         "hardwareProfile": {
206           "vmSize": "[variables('WebVMVmSize')]"
207         },
208         "osProfile": {
209           "computerName": "[parameters('WebVMName')]",
210           "adminUsername": "[parameters('WebVMAdminUsername')]",
211           "adminPassword": "[parameters('WebVMAdminPassword')]"
212         },
213         "storageProfile": {
214           "imageReference": {
215             "publisher": "[variables('WebVMImagePublisher')]",
216             "offer": "[variables('WebVMImageOffer')]",
217             "sku": "[parameters('WebVMWindowsOSVersion')]",
218             "version": "latest"
219           },
```

# Basic Template Structure: Resources (typical)

Location: You might be tempted to create a parameter for Location. However, a better practice is to inherit the location from the resource group

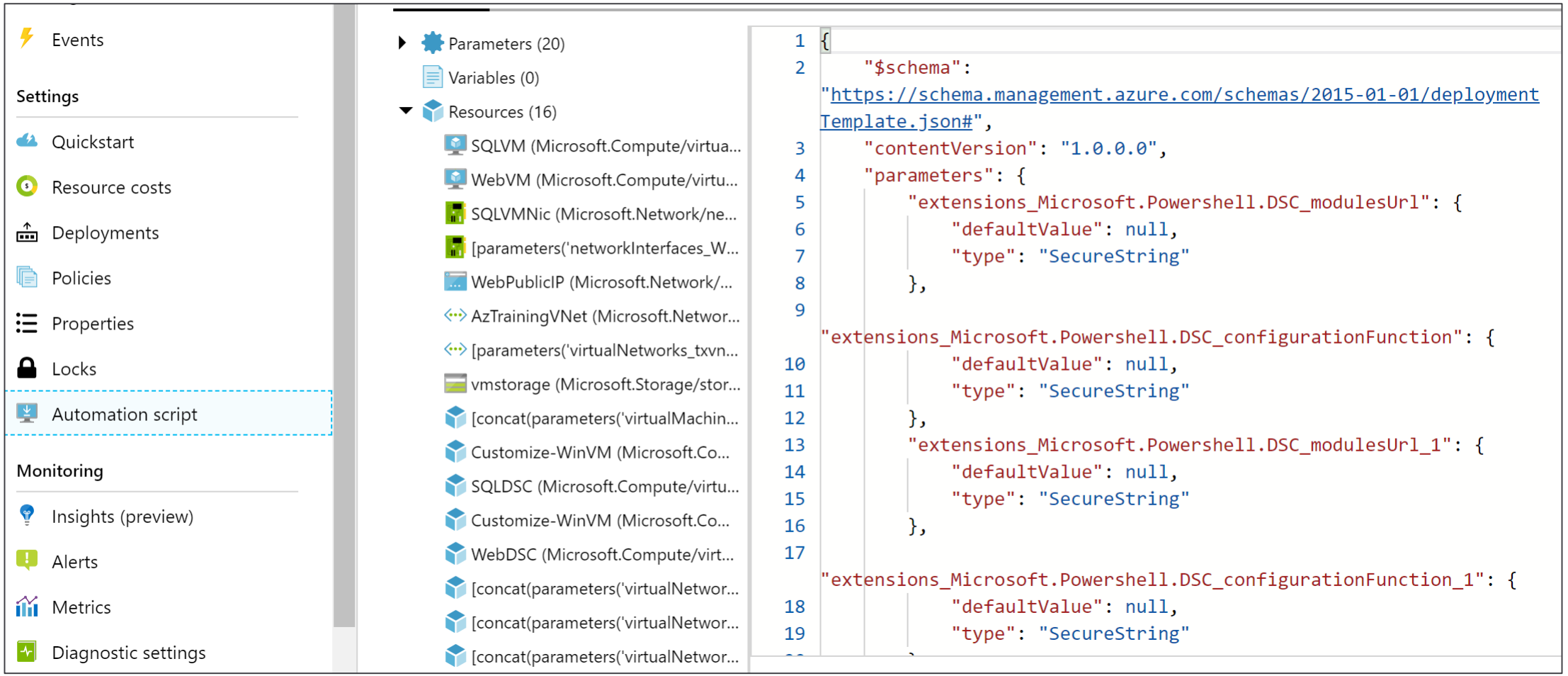
Type: The Type element for a resource is a combination of Resource Provider (discussed before) plus the Resource Type (ex: Microsoft.sql/servers)

Comments: Helpful info to clarify what the resource is, or what it's being used for.

DependsOn: This helps Azure understand dependencies, so it can deploy resources in parallel, or sequentially, as appropriate

API Version: A version is specified for each resource which is associated with a version of the REST API. The version impacts which elements can be specified for the resource, so the versions are updated on occasion

# Azure Automation Script – Resource Group Export



# Demo

Azure Automation Script

# Questions Landing Spot

"…If you want good answers, ask better questions…"

# Azure Automation Script – Resource Group Export - Challenges

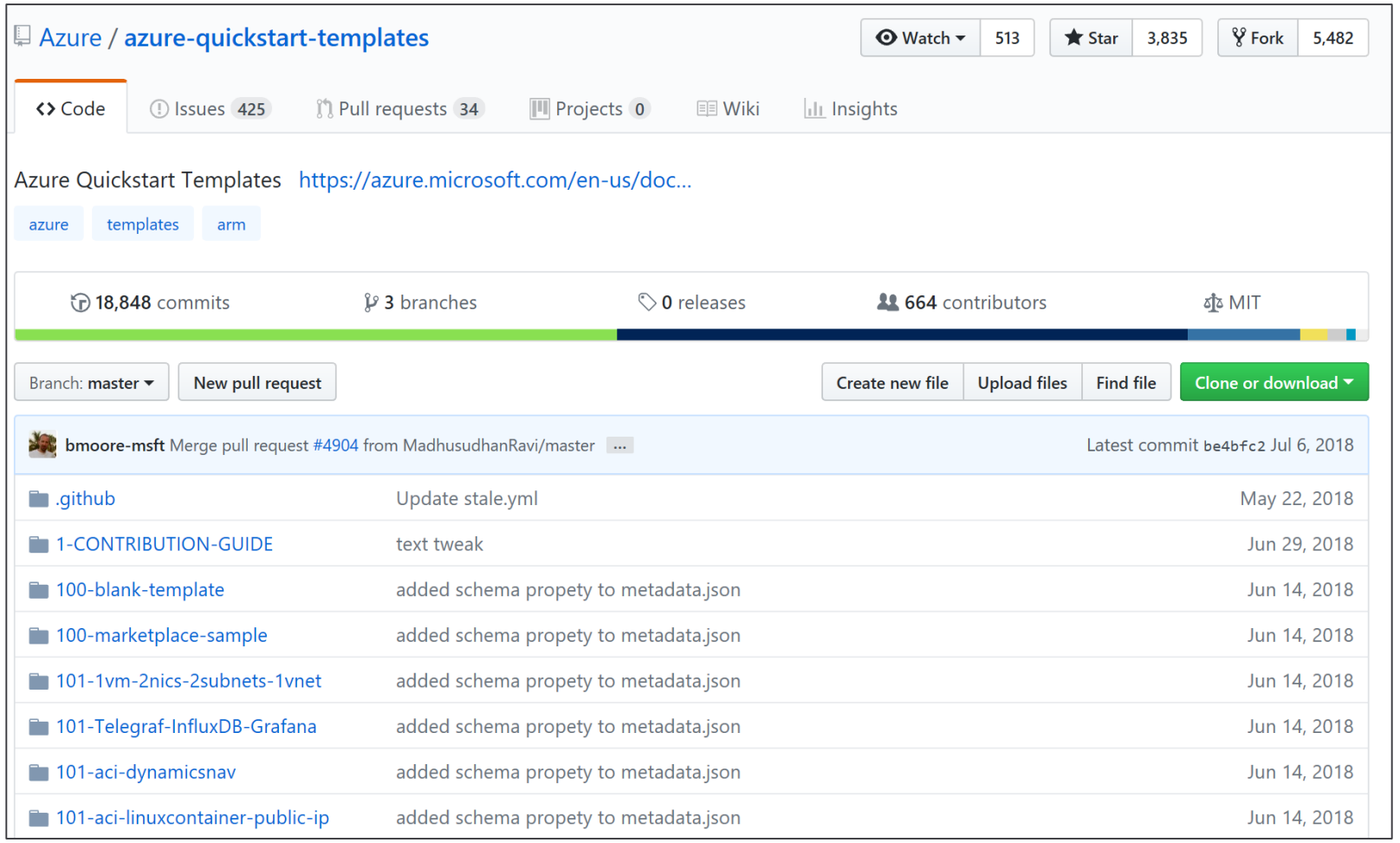
- The value is part of the parameter name. Ex: if you have a web app named "007FFFWebAppDemo" it will generate a parameter called "sites_ 007FFFWebAppDemo _name."
- Not all resources can be scripted out in this manner yet. (You'll see a message at the top of the template pane when this occurs)
- Default values are overused
- The admin password (when required) is not parameterized
- There are a few inaccurate values that come out, which result in deployment failures when reusing the template as-is

# ARM Templates flow to get kickstarted

- Use **Visual Studio 2017/2019** as your « authoring » environment
- Split out **parameter** file in a separate parameters.azuredeploy.json
- **Fine-tune** parameters according your needs
- Matching **naming conventions**
- Add **metadata description** for each parameter
- Add **parameters** for anything that could vary between environments (dev, test, customer A, customer B,…)
- **NEVER** store passwords as cleartext (rather securedstring with prompt or integrate with Azure Key Vault)
- Add **variables** where they make sense
- **Fine-tune** the Resources section
- Use **Resource Group** inherited location
- Use **Tags** (in relation to Azure Policies)

# ARM Templates – GitHub Quickstart Templates

http://www.github.com/Azure/Quickstart-Templates

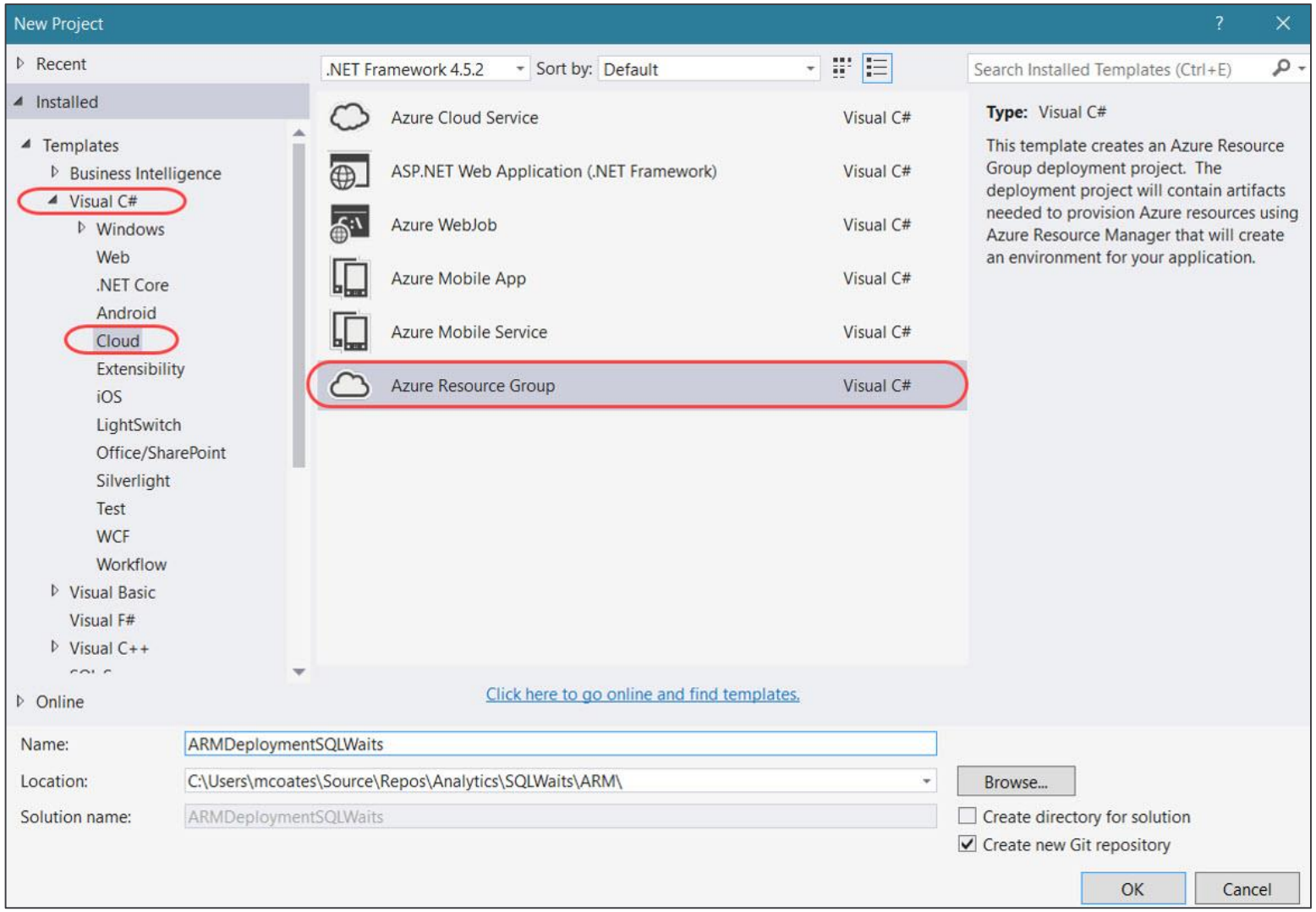


# Demo

Azure QuickStart Templates

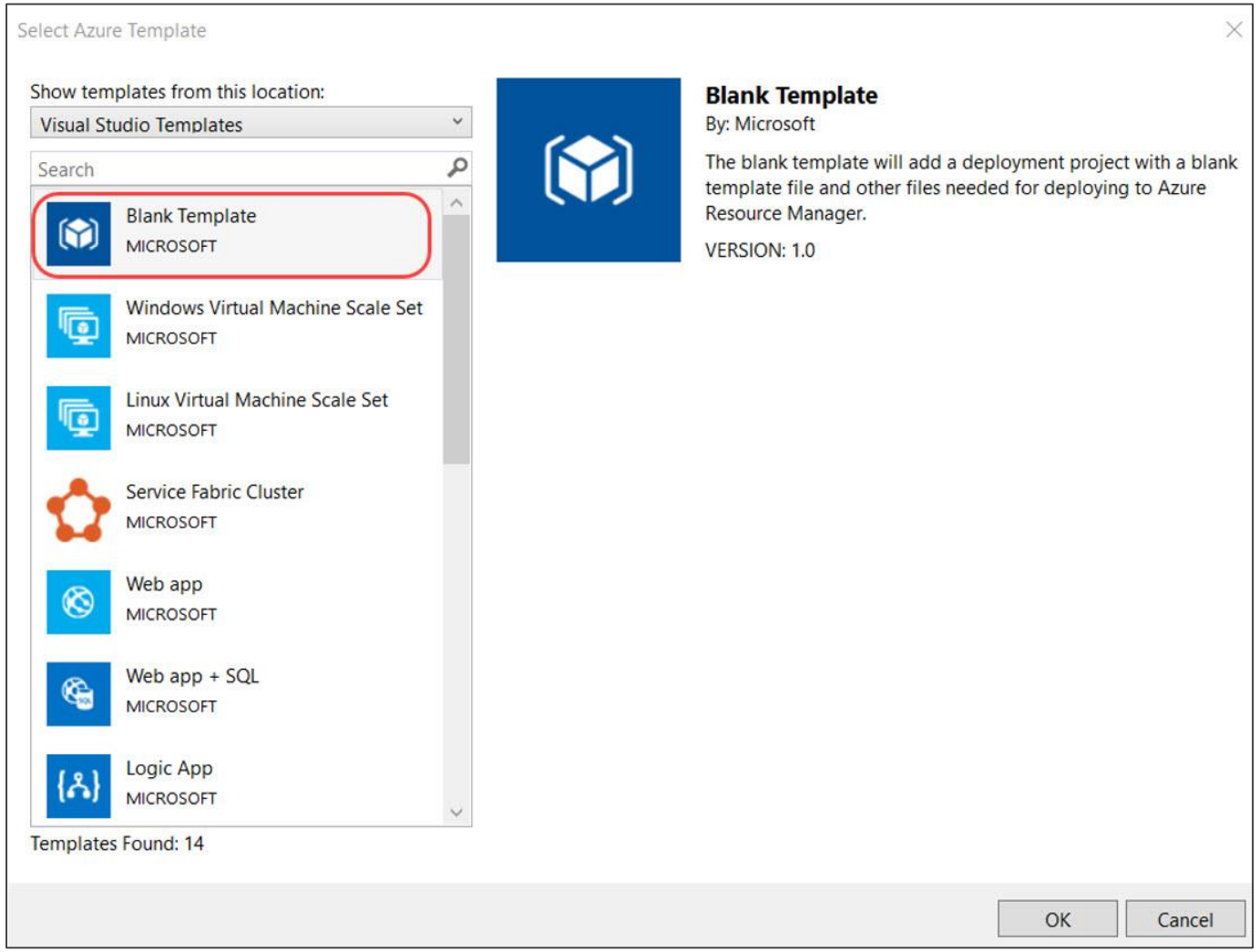
# Create an ARM Project in VS2017 / VS2019

If you don't see this option, it means the Azure SDK has not been installed yet



# Create an ARM Project in VS2017

**Start from a Blank Template or use any of the GitHub QuickStart Templates**

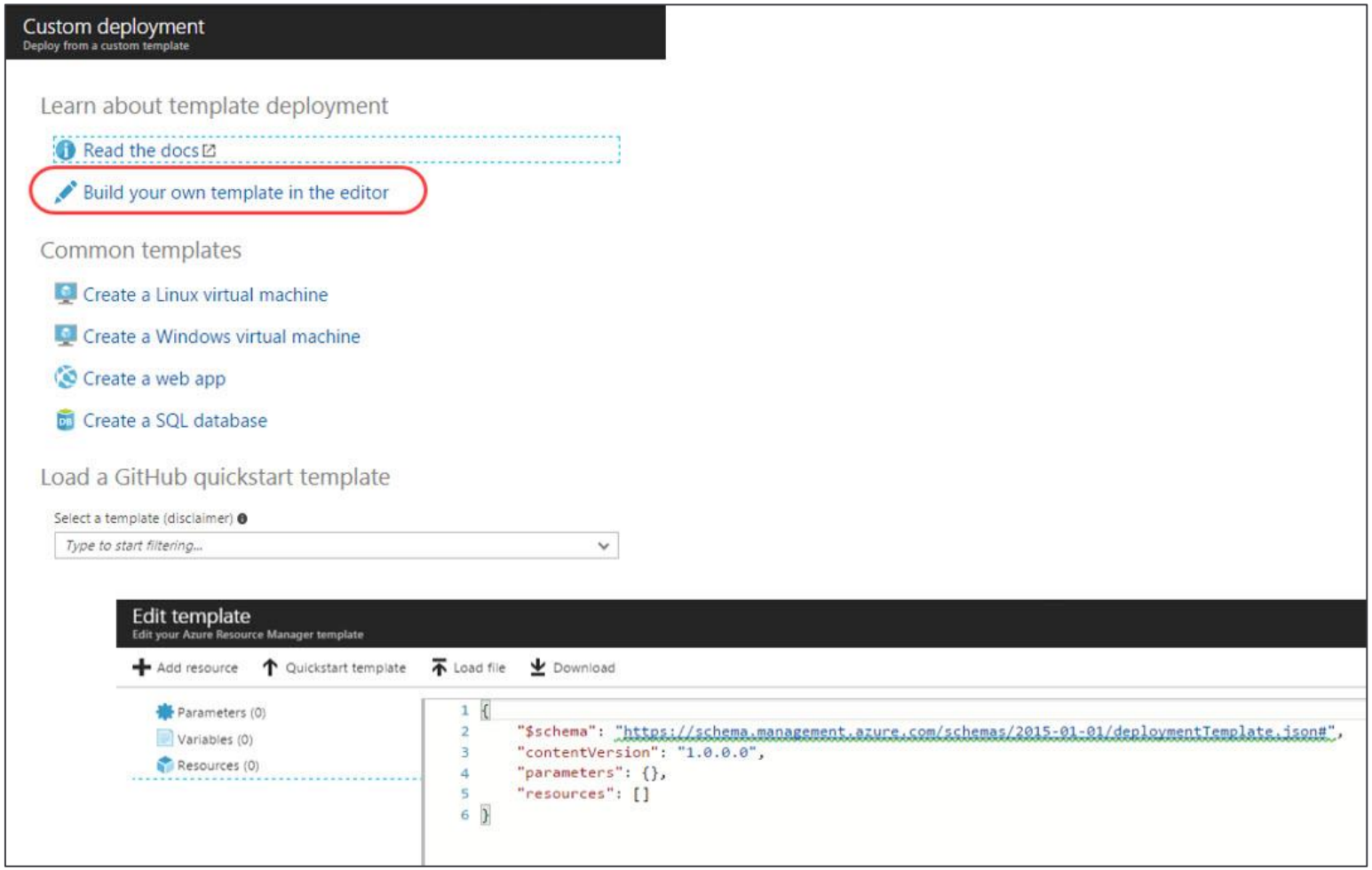


# Demo

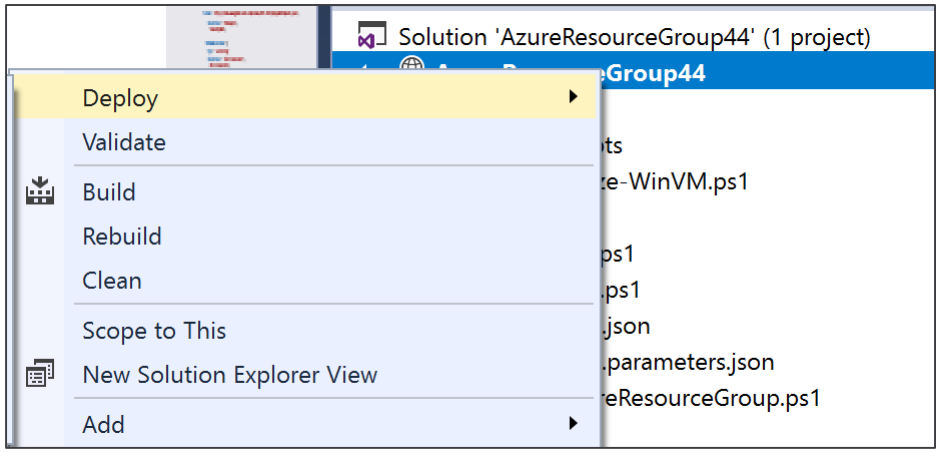
Authoring ARM Templates using Visual Studio 2019

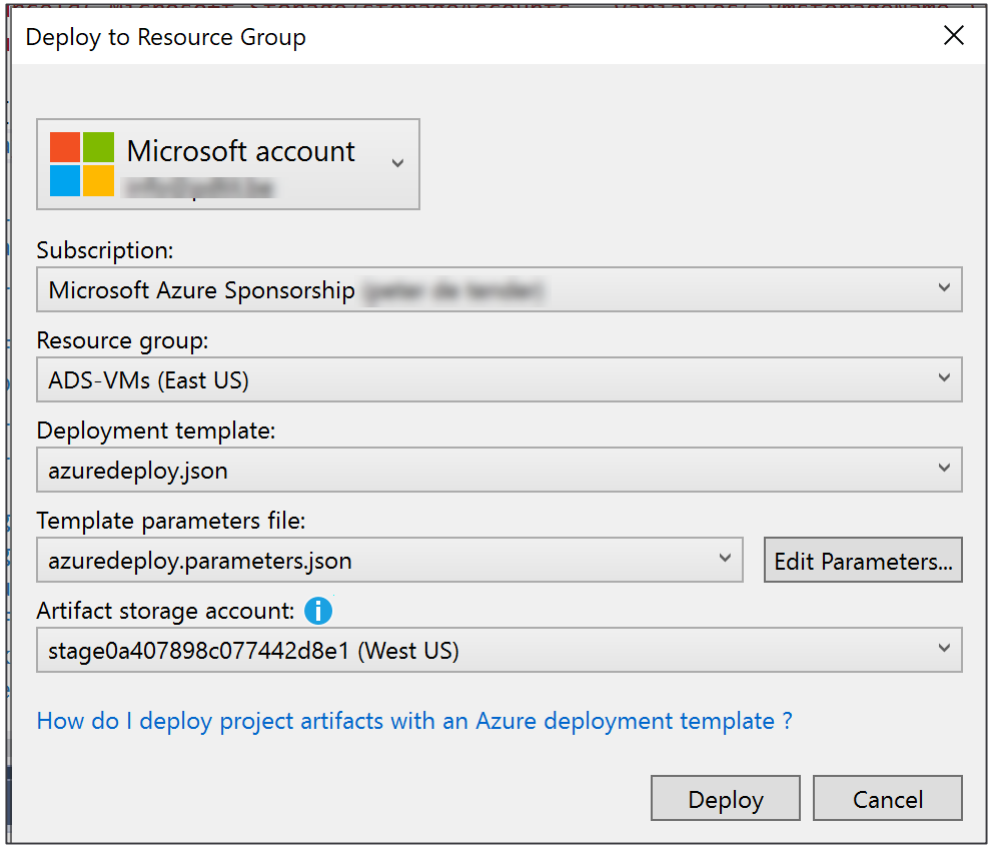
# Running a template deployment from the Azure Portal

Add Resource
/ Template Deployment



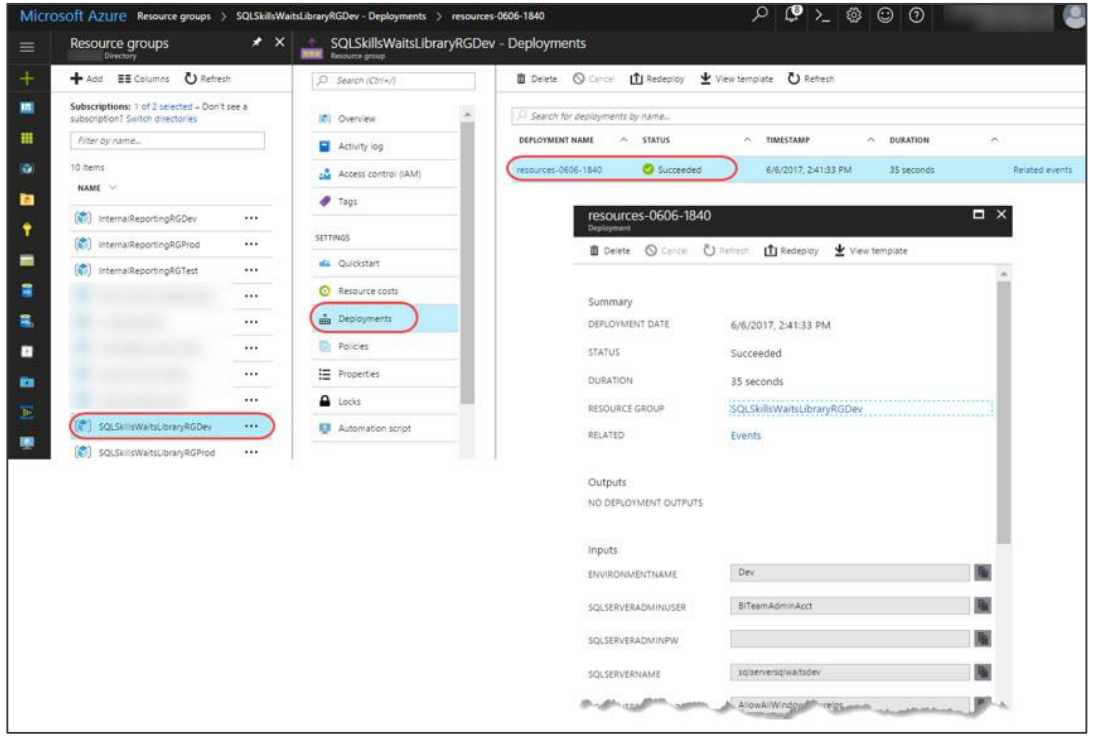
# Running a template deployment from VS2017/VS2019



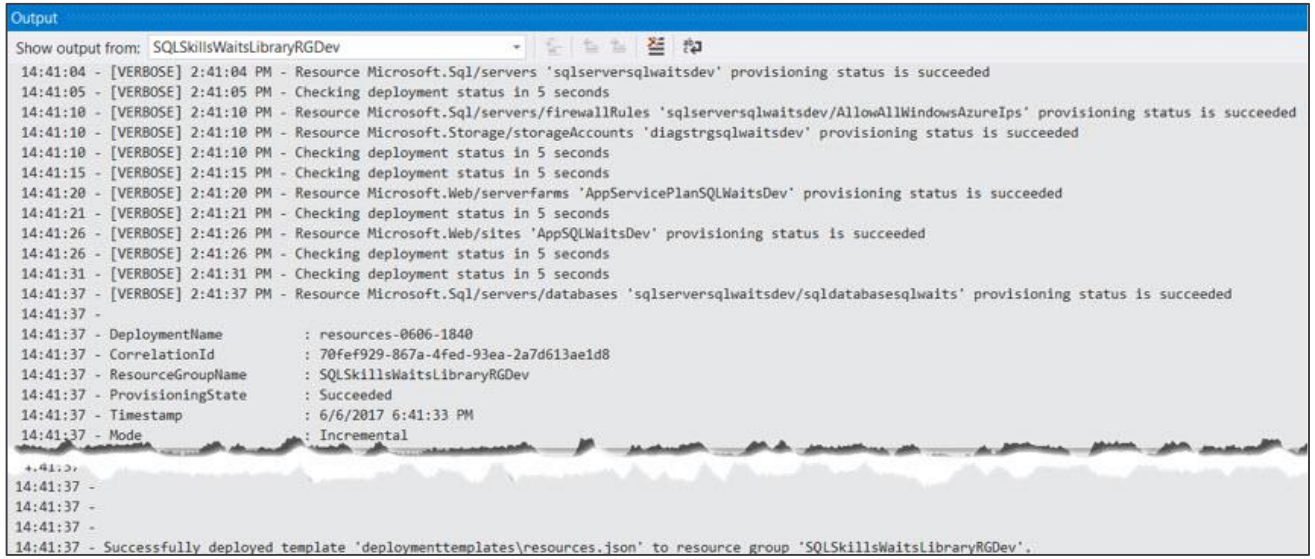



# Monitoring ARM template deployment status

## Azure Portal




## Visual Studio Output

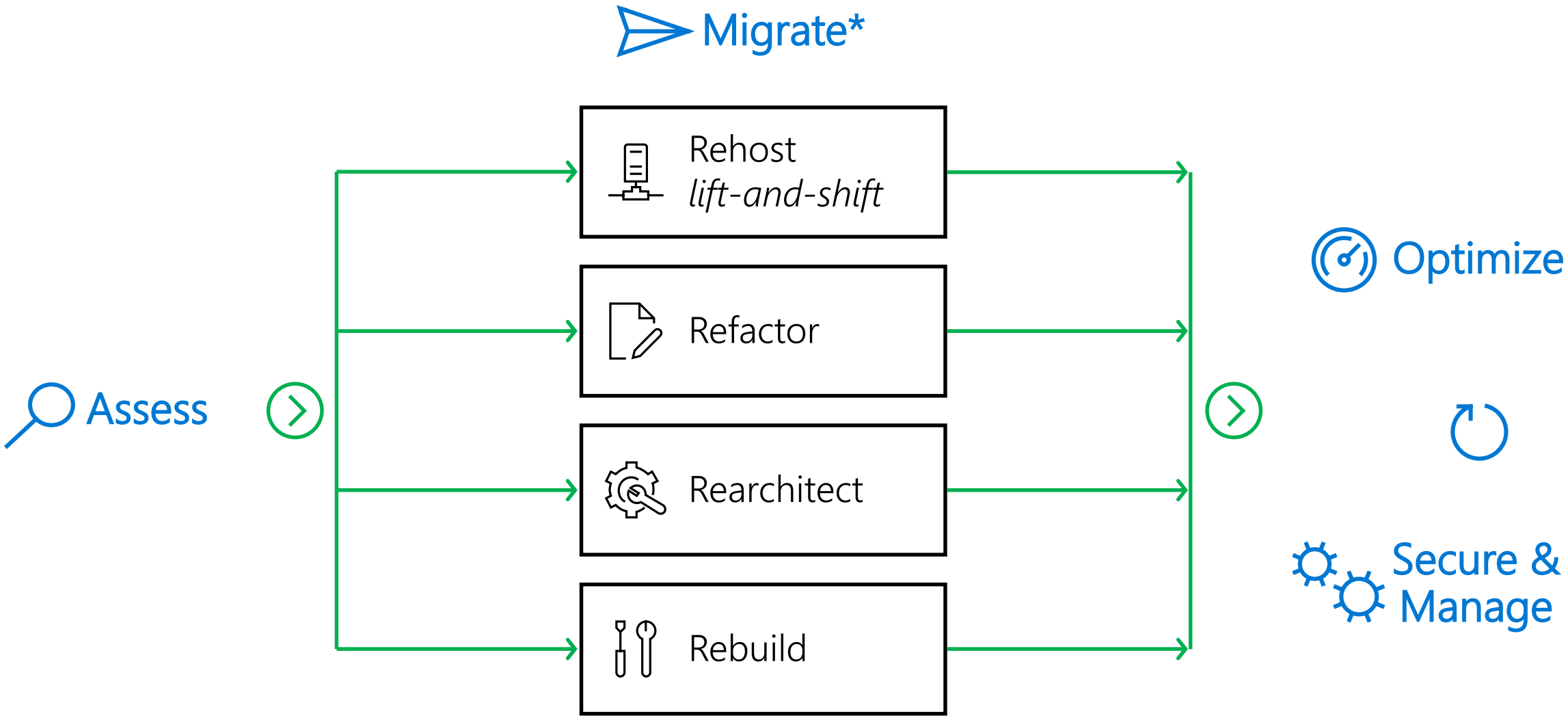


# Demo

Monitoring ARM Template Deployment

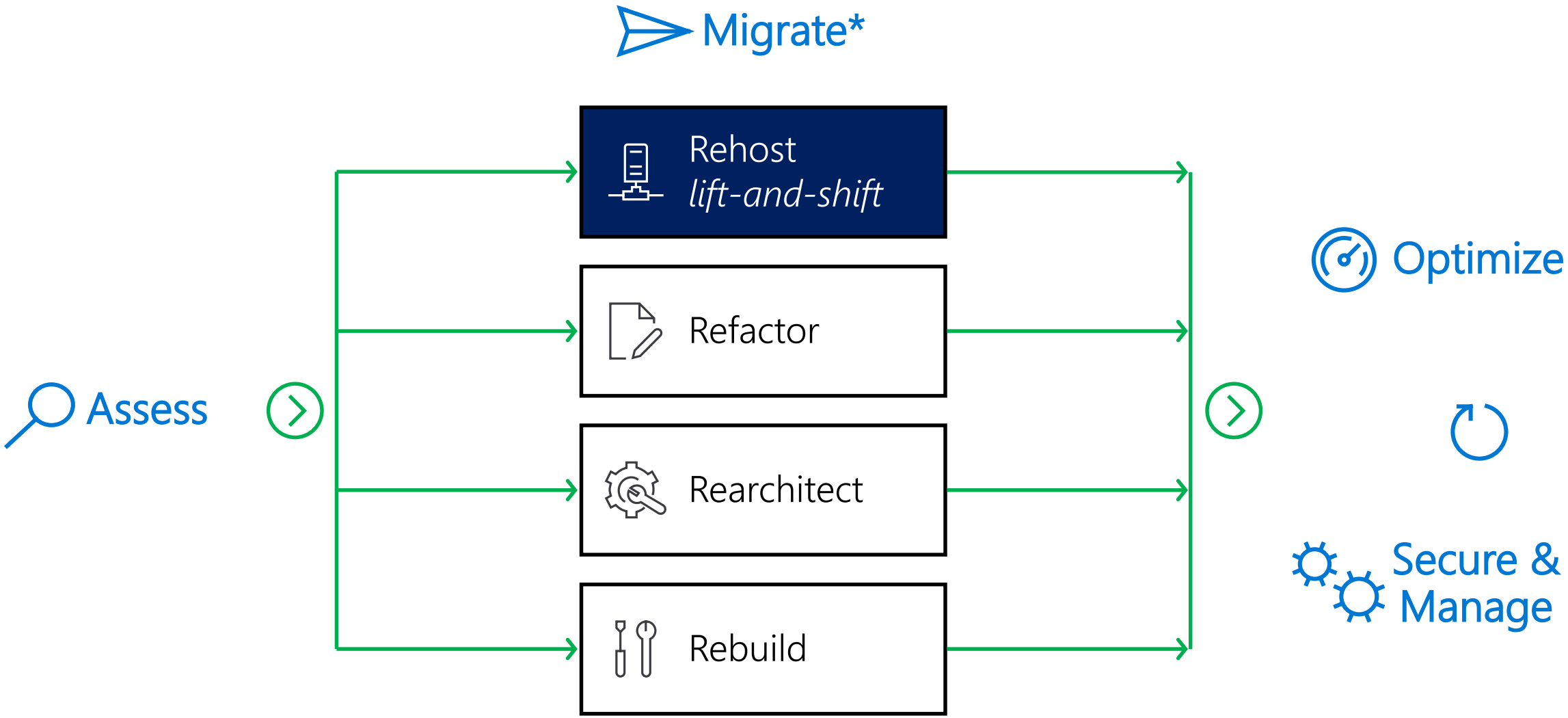
# VM Lift & Shift Migration using Azure Migrate

# Azure Migration Experience: Unified and Extensible




*These migration strategies are adopted from Gartner research. Gartner also calls out a 5th strategy called "Replace" with SaaS

# Azure Migration Experience: Unified and Extensible




*These migration strategies are adopted from Gartner research. Gartner also calls out a 5th strategy called "Replace" with SaaS

# Workloads Migrations Matrix

| Migration Type | Rehost (lift and shift | Refactor | Revise | Rebuild | Replace |
|---|---|---|---|---|---|
| Complexity | Moderate to Complex | Moderate to Complex | Moderate to Complex | Complex | Moderate |
| Example Workloads | RDS, Web Applications, Database | RDS, Web Applications | Legacy Applications, Databases | Legacy Applications | Legacy Applications |
| Tooling Capabilities | Extensive (First and 3rd party) | Several (First and 3rd party) | Minimal (First and 3rd party) | Extensive (First and 3rd party) | NA |
| Benefits | Proven, Quick and Container Support | Semi-Optimized and Container Support | Optimized, Container Support and PaaS enabled | Fully Optimized, Containerized and PaaS enabled | State of the Art, SaaS, HA, BCDR enabled. |
| DevOps / Automation | Moderate (Automation) | Moderate (Automation, Continuous Delivery) | Moderate (Automation, CI/CD) | Maximum (Automation, CI/CD) | NA |

# Migration strategies: Rehost application (i.e., lift & shift)

- **Rehost (lift and shift)**
- Refactor
- Rearchitect
- Rebuild

## What is it?

Redeploy an existing application to a cloud platform without modifying its code. The application is migrated "as is", which provides baseline cloud benefits without the risk or costs of making code changes.

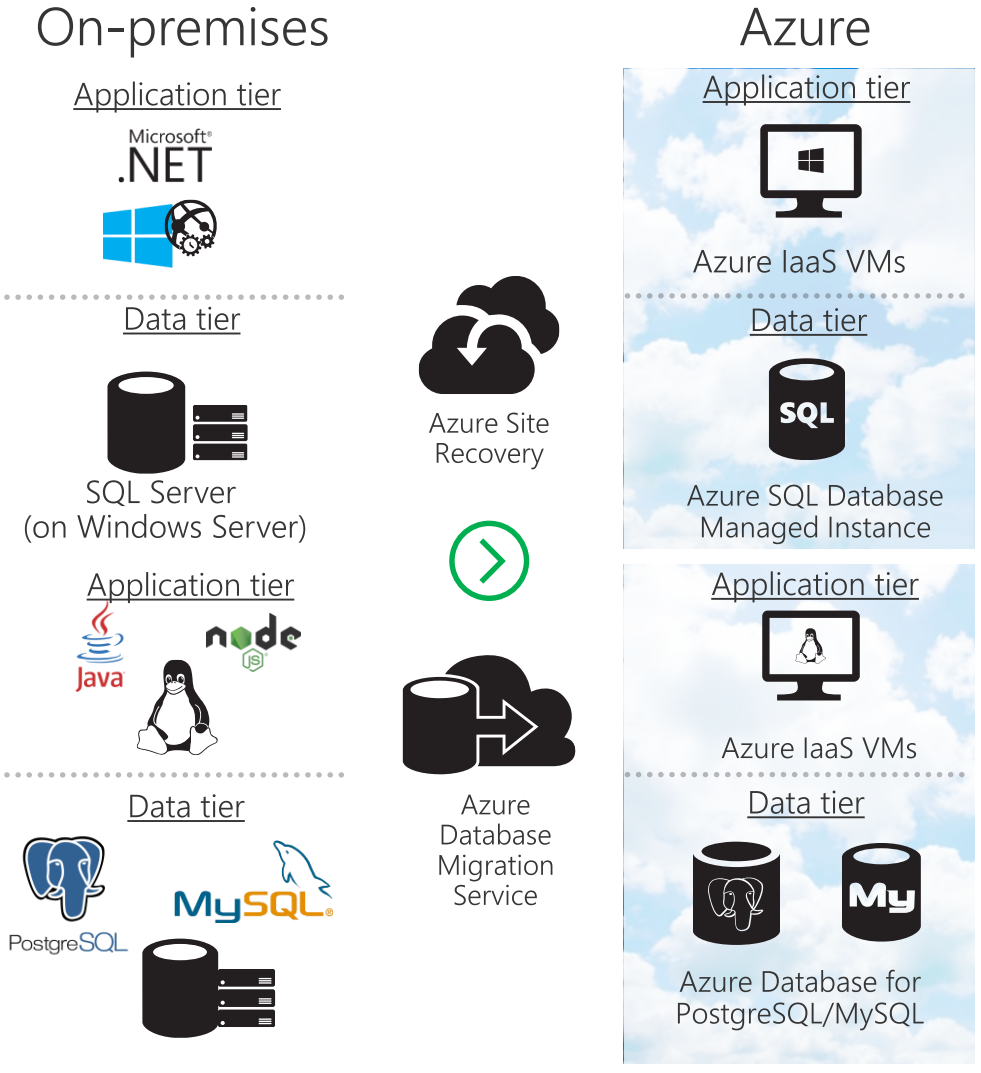
## When to use

- Need to quickly move applications from on-premises to the cloud (e.g., datacenter contract expiry)
- When the application is needed, but evolving its capabilities isn't a business priority
- For applications which are architected to leverage Azure IaaS scalability
- Specific application or database requirements which can only be met by Azure VMs

## Example

Move a line of business application to Azure VMs

## Sample illustration



# Choice of tools for every stage and every requirement

## Assess

Azure Migrate

Data Migration Assistant (DMA)

SQL Server Migration Assistant (SSMA)

## Migrate

Azure Site Recovery (ASR)

Azure Database Migration Service

Azure Data Box

## Optimize

Azure Cost Management

## Secure & Manage

Azure Security & Management (e.g., Blueprint, Security Center, Backup, Monitor)

## Partners

MOVERE
Cloudamize
CloudAtlas®
BitTitan
appOrbit
CHEF
STRATOZONE®
turbonomic
CloudEndure®
Zerto
Corent
intigua
stiim
ATTUNITY
DATOMETRY
MORPHIS
Ispirer
Informatica
QuerySurge

# Choosing between 1st and 3rd party assessment tools

## Use Azure Migrate when customer needs…

- A Microsoft Azure branded tool which is free
- Assessment of VMware environments[1]
- Assessments of brand new Azure platform functionality

[1] Support for Hyper-V and physical servers is coming soon

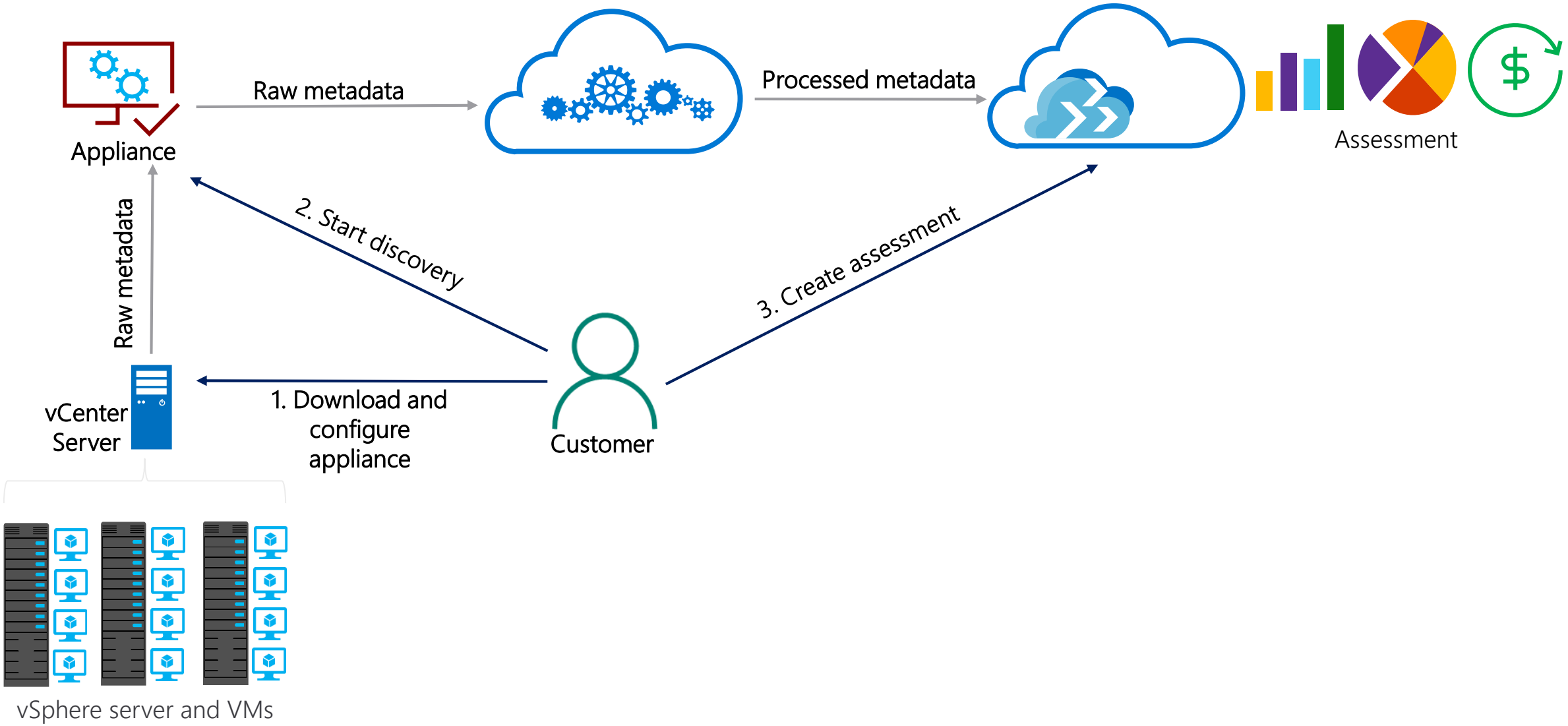
## Use ISV tools when customer needs…

Cloudamize
TSO Logic
turbonomic

| | |
|---|---|
| Assessment of heterogeneous environments (physical, VMware and Hyper-V) | ✓ |
| Agentless dependency visualization | ✓ |
| Cloud economics cost modelling (including storage/networking costs) | ✓ |

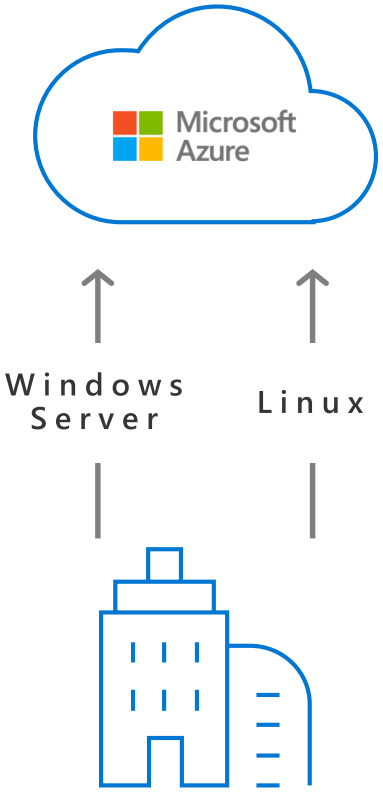
*Tools listed are recommendations, but not an exhaustive list. See azure.com/migration/partners for full list.*

# Azure Migrate – How it works



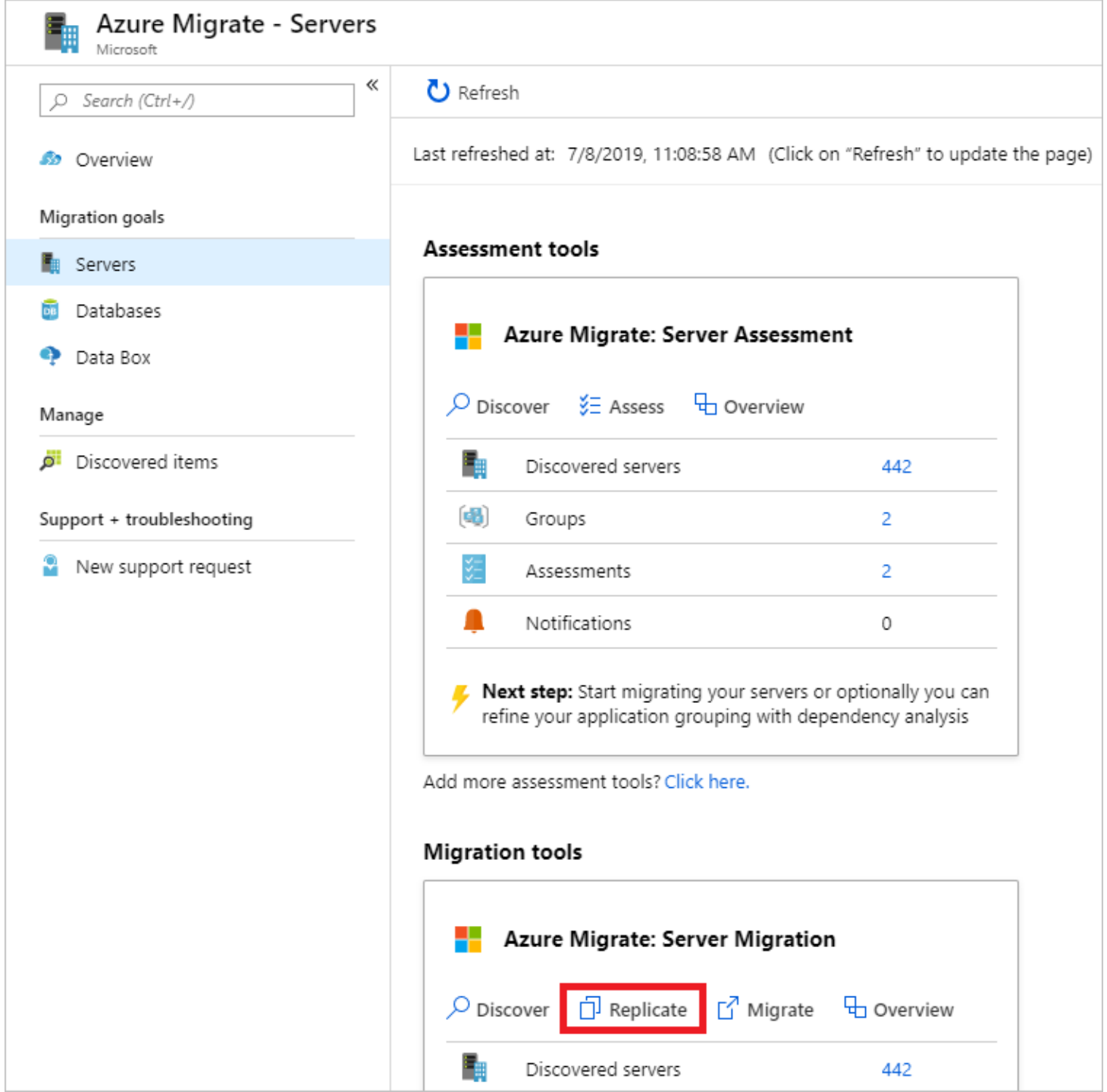
# Azure Migrate v2

- **Unified migration platform**: Use a single portal to start, run, and track your migration journey to Azure.
- **Range of tools**: Azure Migrate provides native tools, and integrates with other Azure services, as well as with ISV tools. Select the right assessment and migration tools, based on your organizational requirements.
- **Azure Migrate Server Assessment**: Use the Server Assessment tool to assess on-premises VMware VMs and Hyper-V VMs, for migration to Azure.
- **Azure Migrate Server Migration**: Use the Server Migration tool to migrate on-premises VMware VMs, Hyper-V VMs, cloud VMs, and physical servers to Azure.
- **Azure Migrate Database Assessment**: Assess on-premises databases for migration to Azure.
- **Azure Migrate Database Migration**: Migrate on-premises databases to Azure.



# Replicate VMs

Azure **Migrate project** > **Servers**, Azure Migrate: Server Migration, click **Replicate**

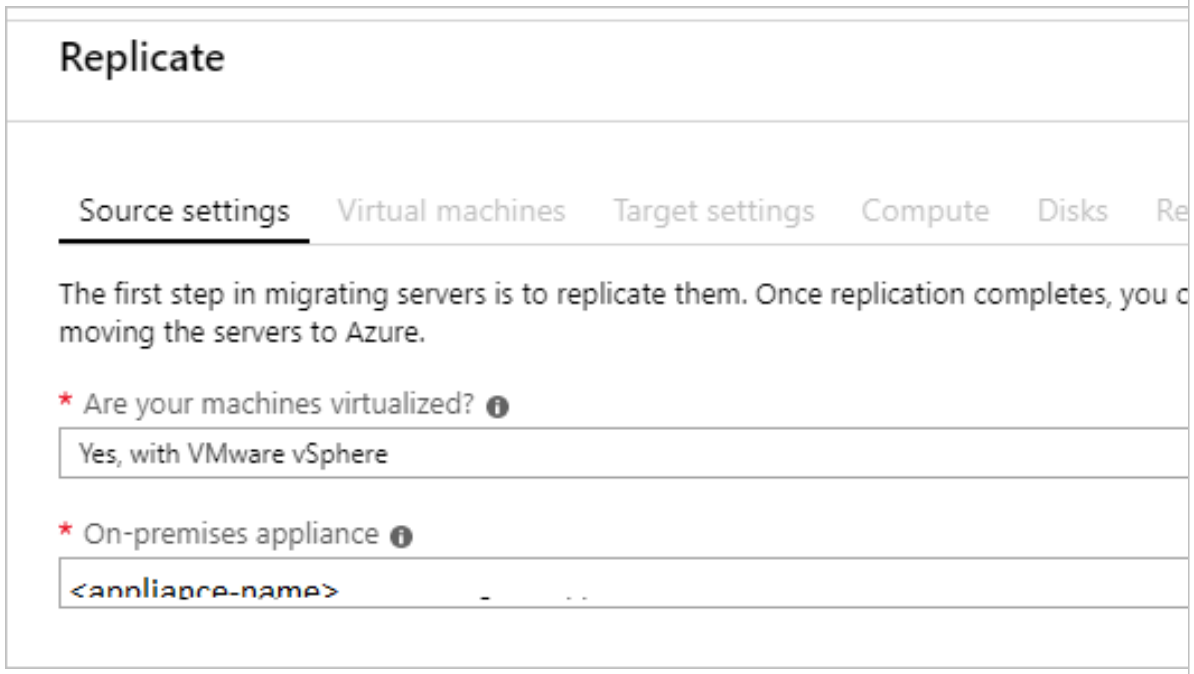


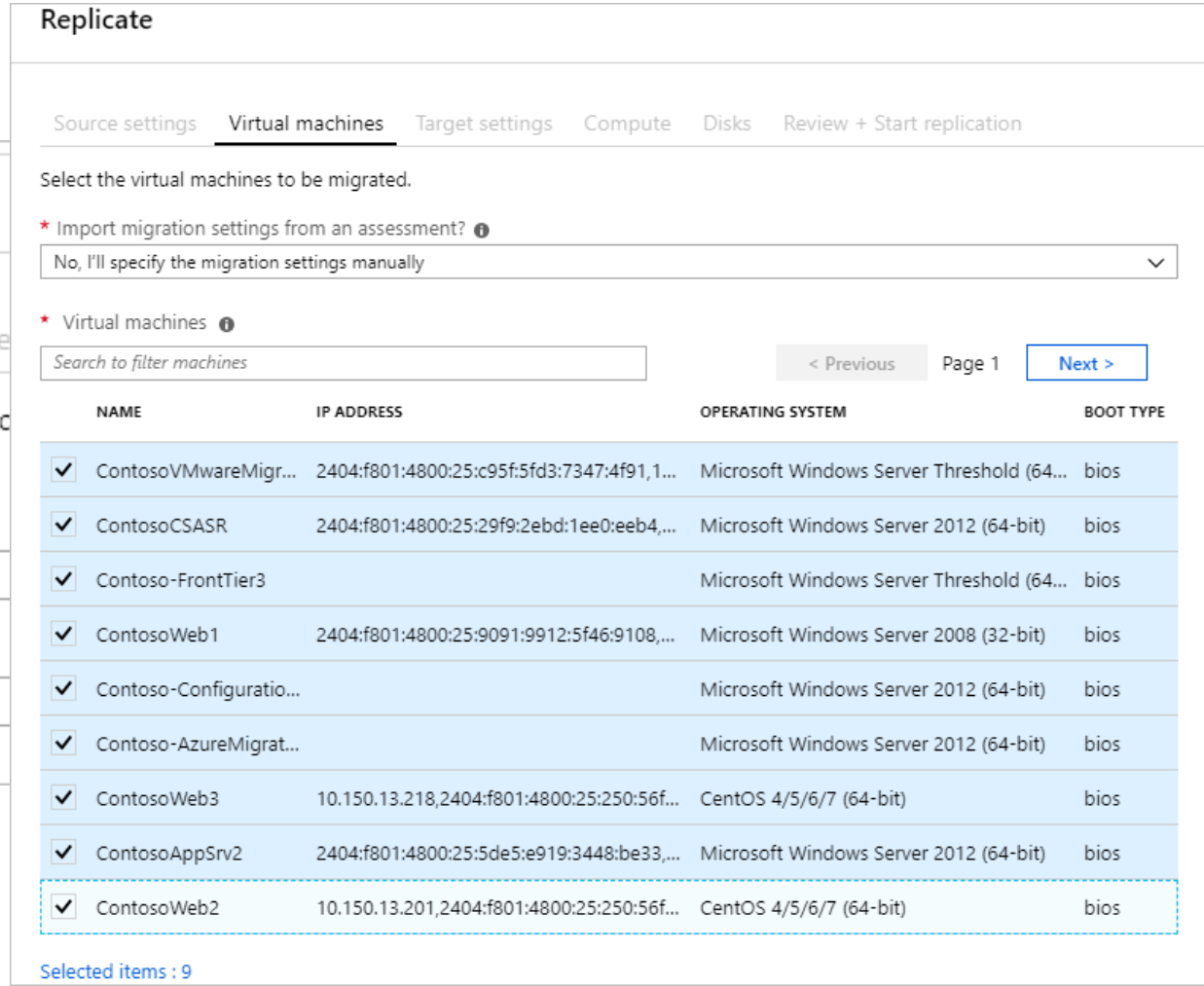
# Demo

VM Lift & Shift migration using Azure Migrate

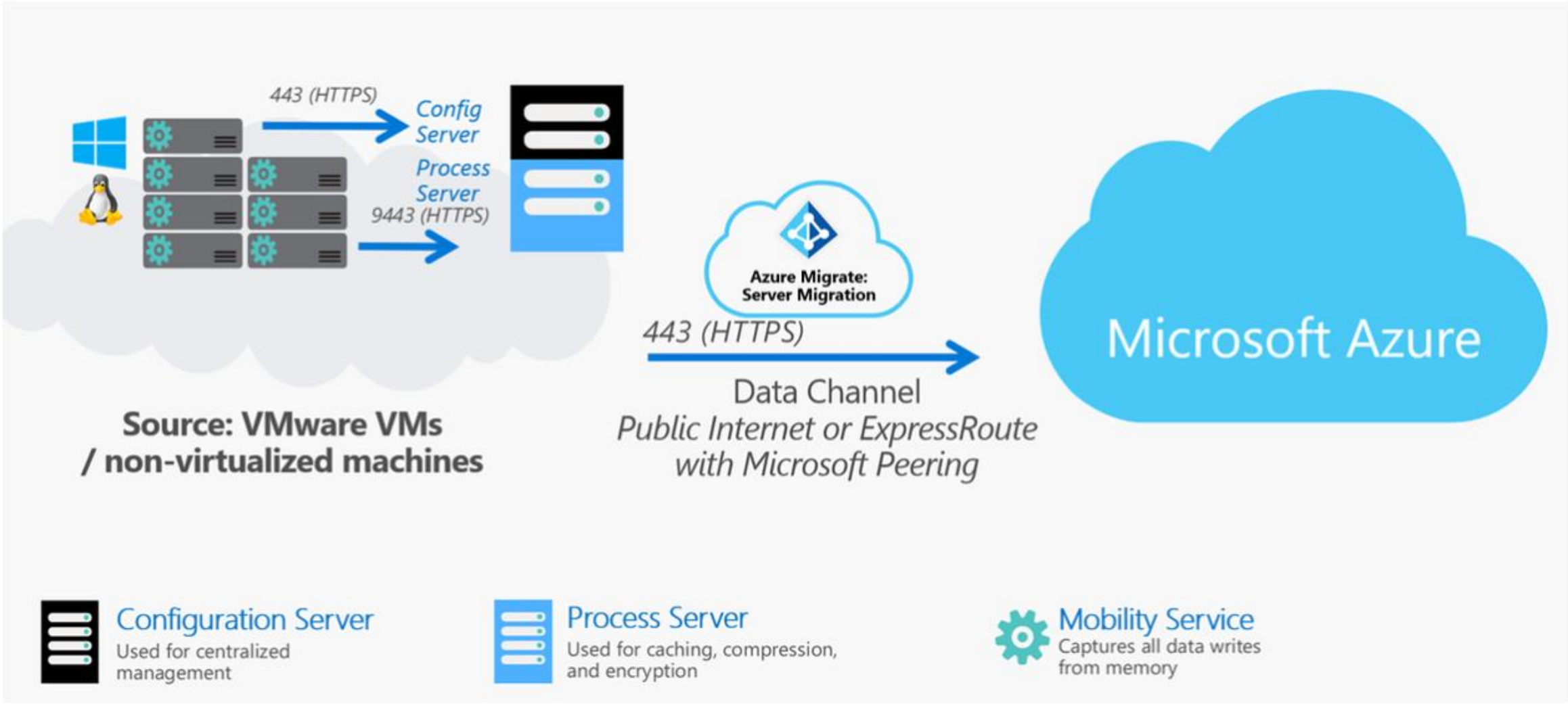
# Select the source

**Replicate**, > **Source settings** > **Are your machines virtualized?**, select **Yes, with VMware vSphere**.

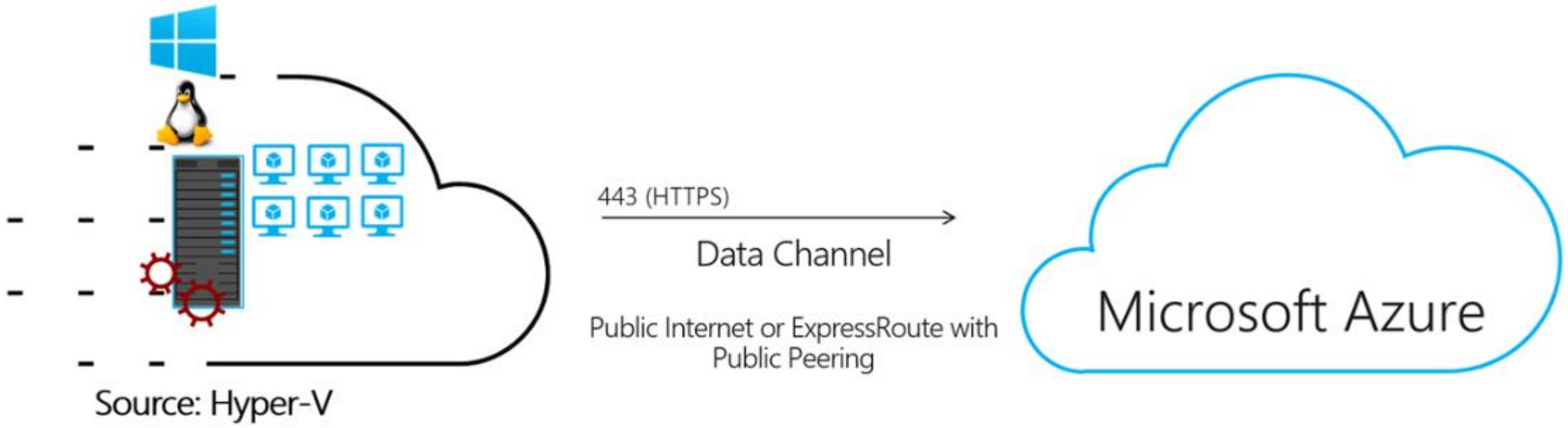


# Agent based Server Migration Architecture (VMware, AWS, Physical)



# Agent based Server Migration Architecture (Hyper-V)



# Azure Site Recovery for Azure IaaS VMs Lift & Shift Migration

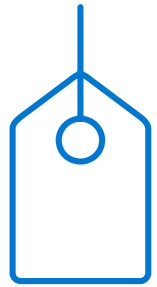
Ensure compliance

1 **No-impact DR testing**

2 **Meet your RPO and RTO SLAs**

3 **Centralized monitoring and alerting**

# Azure Site Recovery for Azure IaaS VMs Lift & Shift Migration



Simple

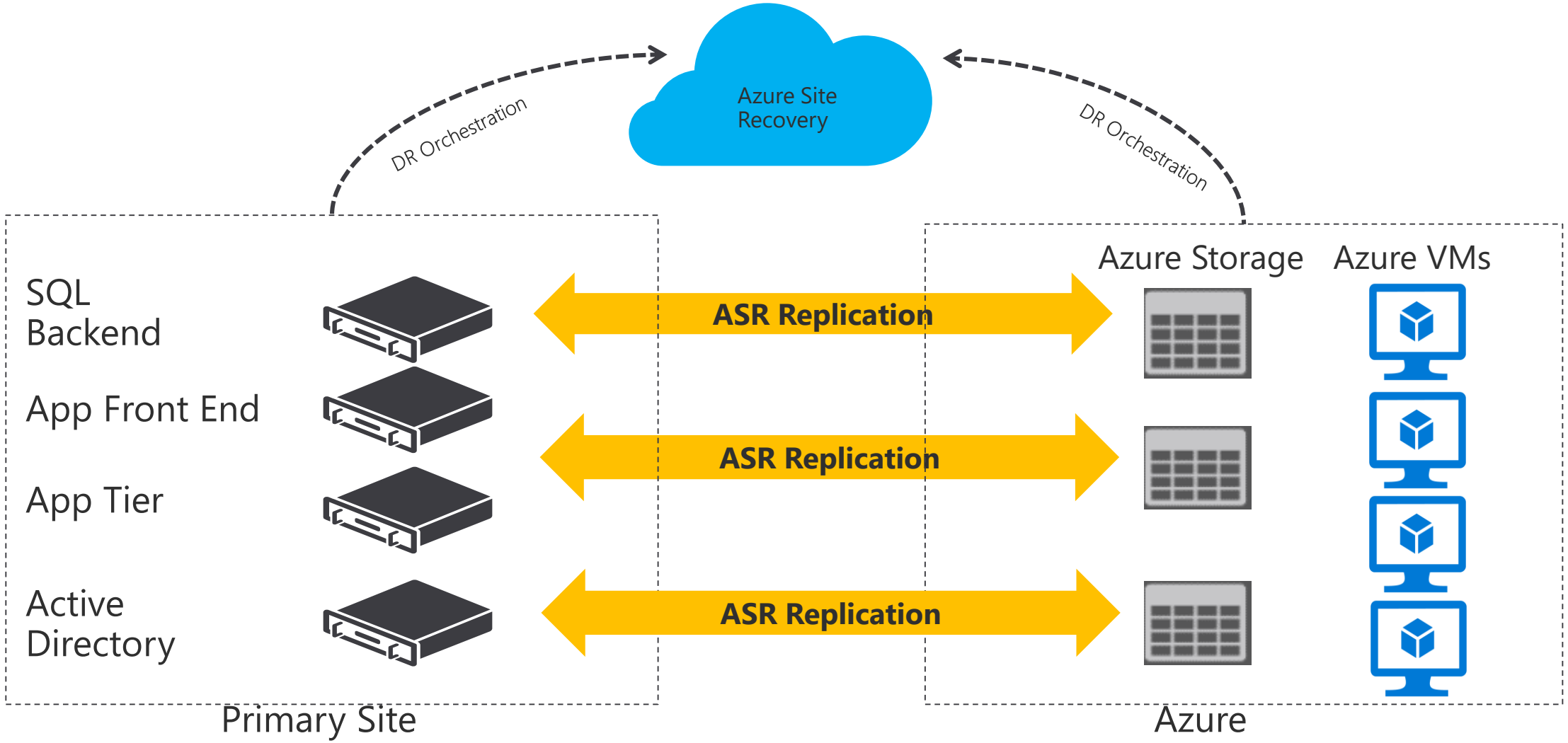
1 **One-click replication**

2 **One-click VM Migration**

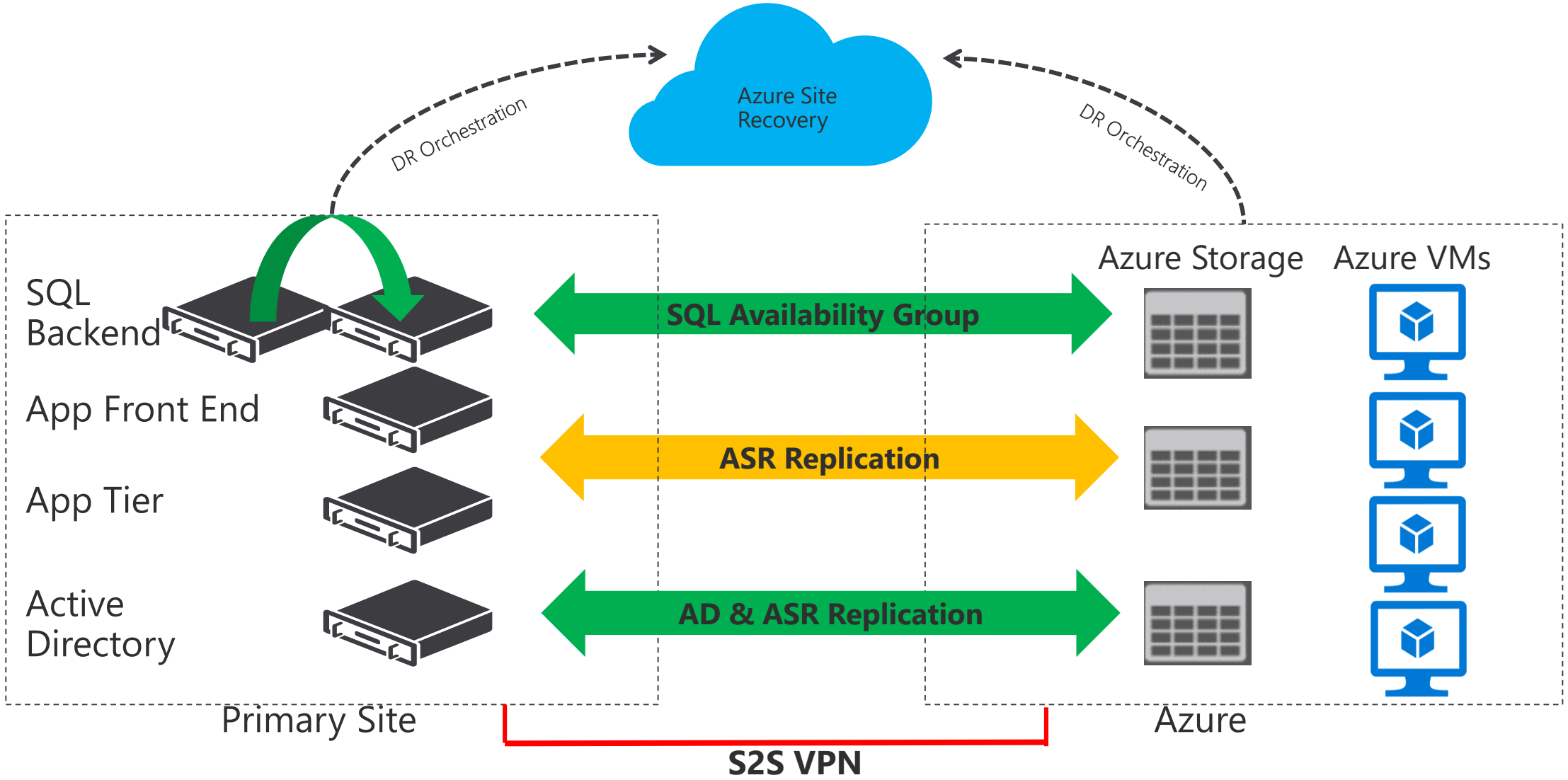
# Azure Site Recovery for Azure IaaS VMs Lift & Shift Migration


- **Zero** application **data loss** during migration
- **Near-zero** application **downtime** during migration
- **Broad coverage** for hypervisors, applications, operating systems, and Azure features
- **No-impact** application **testing** in Azure

# Azure Site Recovery for Azure IaaS VMs Lift & Shift Migration



# Azure Site Recovery for Azure IaaS VMs Lift & Shift Migration



# Demo

Use Azure Site Recovery for Lift & Shift VM Migration

# Section Take-Aways

1. Infrastructure as Code (IAC) allows you to deploy Azure resources in an automated, repeatable way

2. Azure Migrate (v2) is recommended for agent-less migration – of VM Lift & Shift scenarios

3. Azure Site Recovery (ASR) performs the actual VM replication process, and can be used out of Azure Migrate, or as a stand-alone service for Lift & Shift migrations

# Questions Landing Spot

"…If you want good answers, ask better questions…"

# Next Module...

## (SQL) Database Migration

Microsoft

# Thank You