COPENHAGEN
INTERNATIONAL
SOFTWARE DEVELOPMENT
CONFERENCE 2015

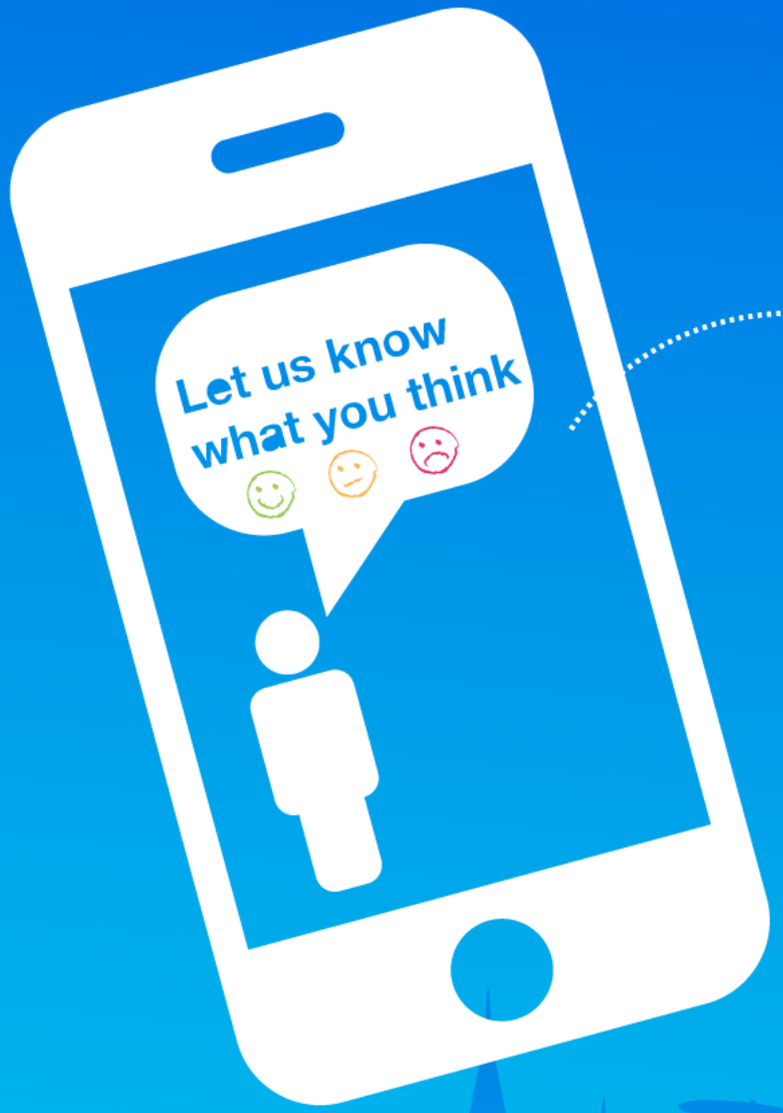
goto; conference

DevOps:

Entire solutions as deployable units with Azure Resource Manager

Henrik Westergaard Hansen
Evangelism Manager
@henrikwh

goto; conference



Click ‘engage’
to rate sessions
and ask questions

Join the conversation #gotocph

Today's Challenge

It's difficult to…

→Deploy or update a group of resources, repeatedly

→Reproduction

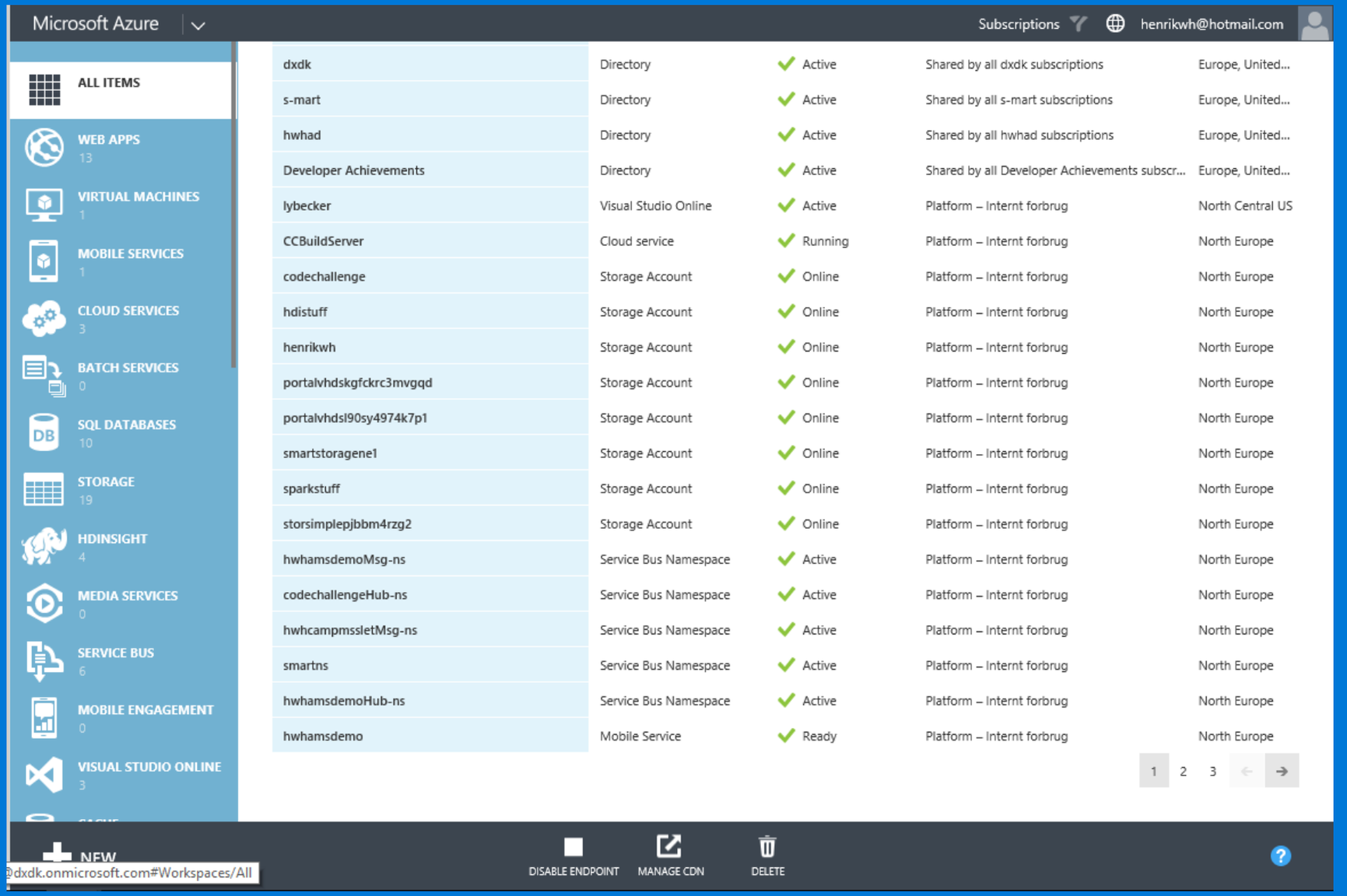
→Manage permissions on a group of resources

→Change management

Singletons

- → Deploy becomes more complex
- → Proper use of resources becomes more abstract
- → Isolation makes communication a challenge

Resource Centric Views

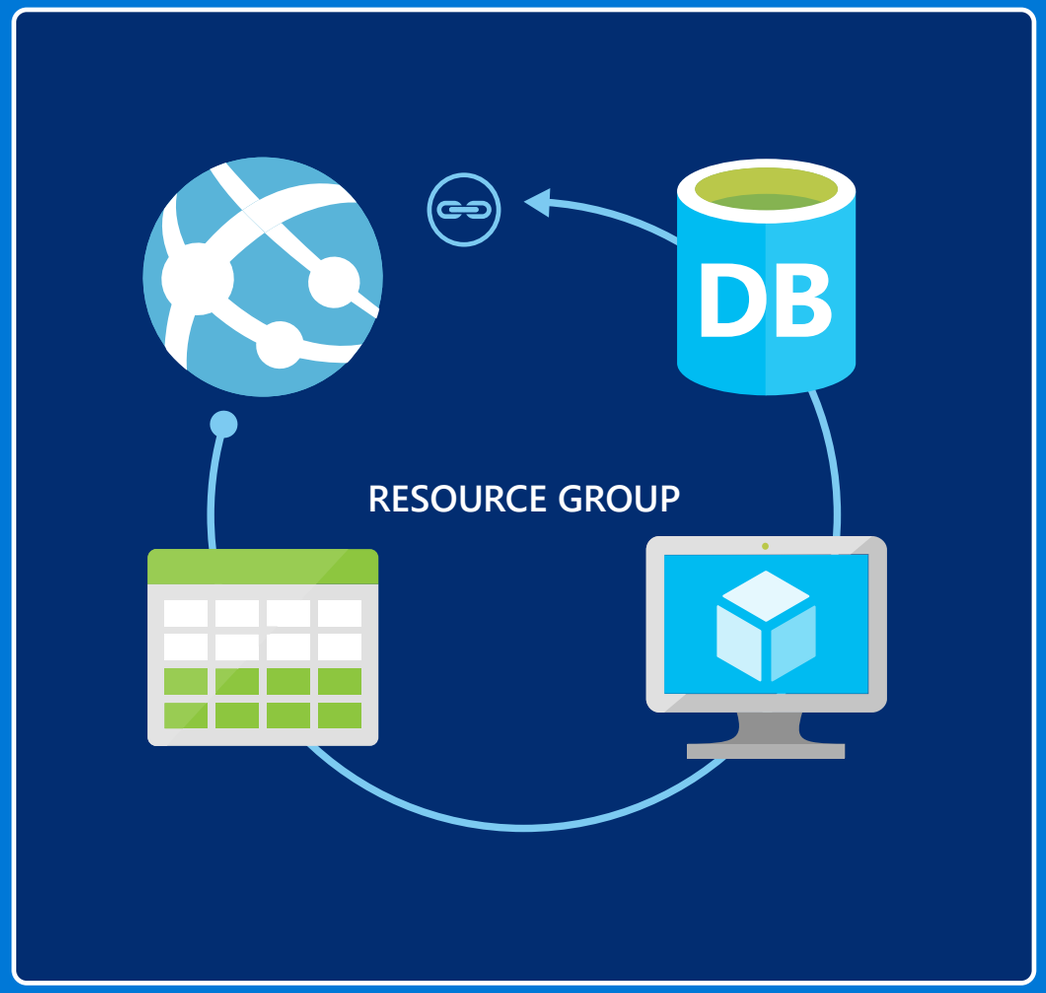


Introducing Resource Manager

→Application Lifecycle Container

→Declarative solution for Deployment and Configuration

→Consistent Management Layer



Resource Groups

- → Tightly coupled containers of multiple resources of similar or different types
- → Every resource *must* exist in one and only one resource group
- → Resource groups can span regions

Coupling for Resources

Resource Group is a unit of management

→ **Lifecycle**: deployment, update, delete, status

→ **Grouping**: metering, billing, quota: applied & rolled up to group

Resource Group Lifecycle

Question:

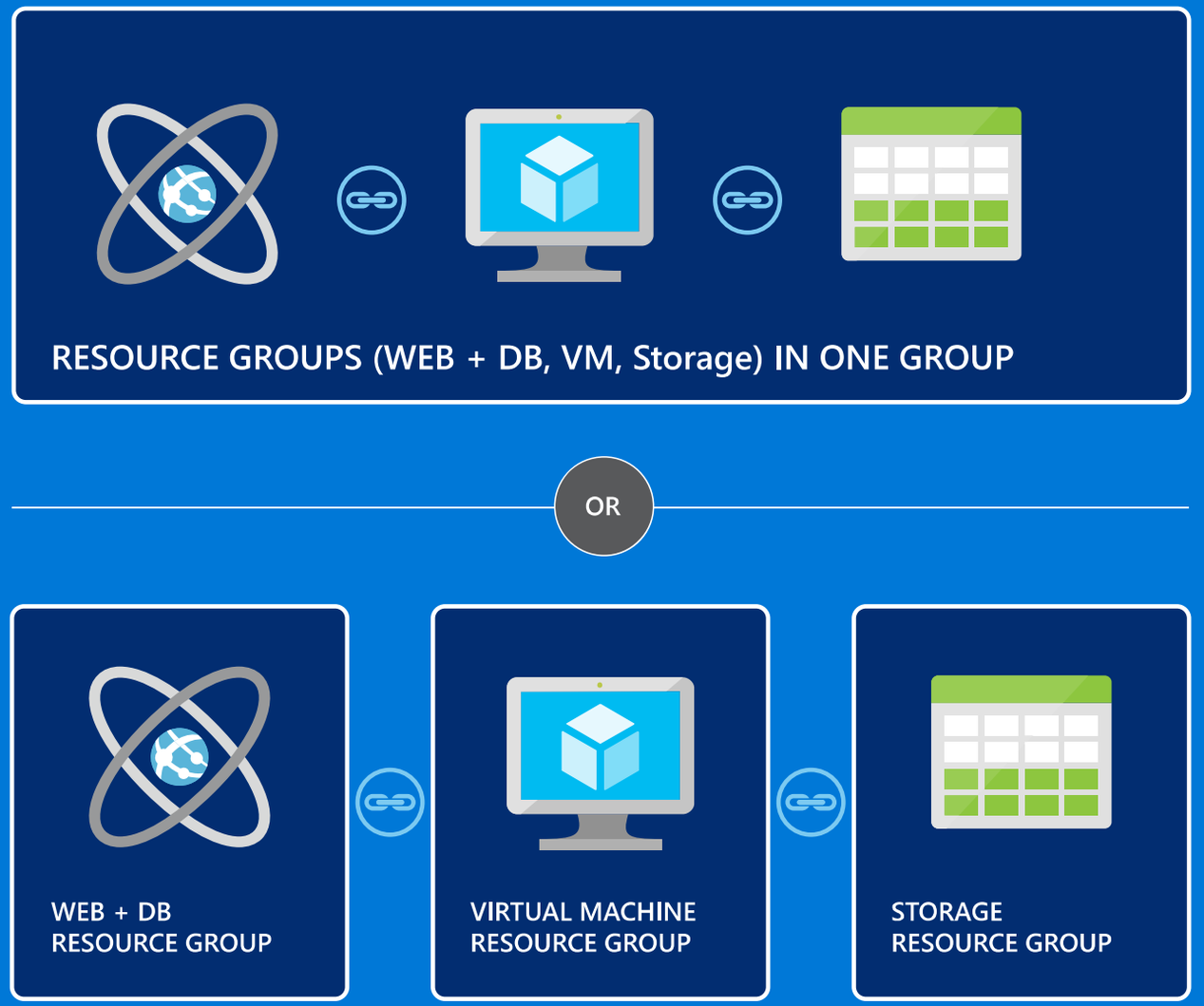
Should these resources be in the same group or a different one?

Hint:

Do they have common lifecycle and management?

Answer:

Up to you.



What is Azure Resource Manager?

Consistent Management Layer

Tools

Microsoft Azure

+

Command Line

+

Visual Studio

AZURE RESOURCE MANAGER API

RESOURCE MANAGER

Cloud + On-Premises

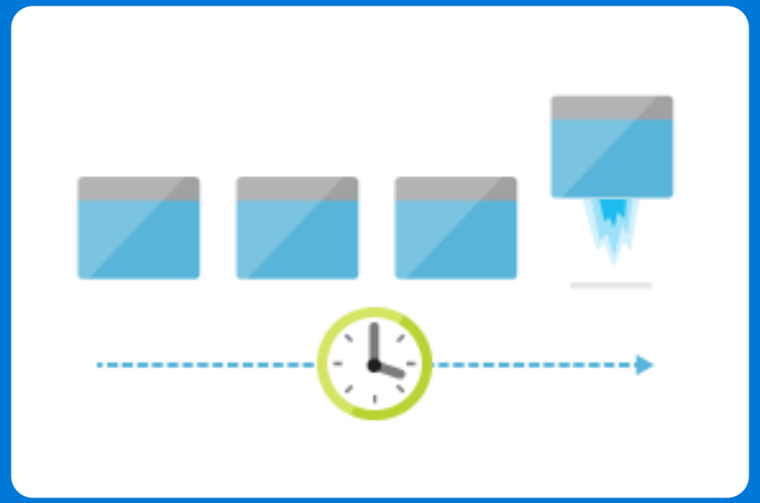
ADFS

AAD

RESOURCE PROVIDER CONTRACT

Provider Rest Points

Areas of Focus



Deploy

Organize

Control

Deploying with Azure Resource Manager

- template-driven
- declarative
- idempotent
- multi-service
- multi-region
- extensible

imperative or declarative

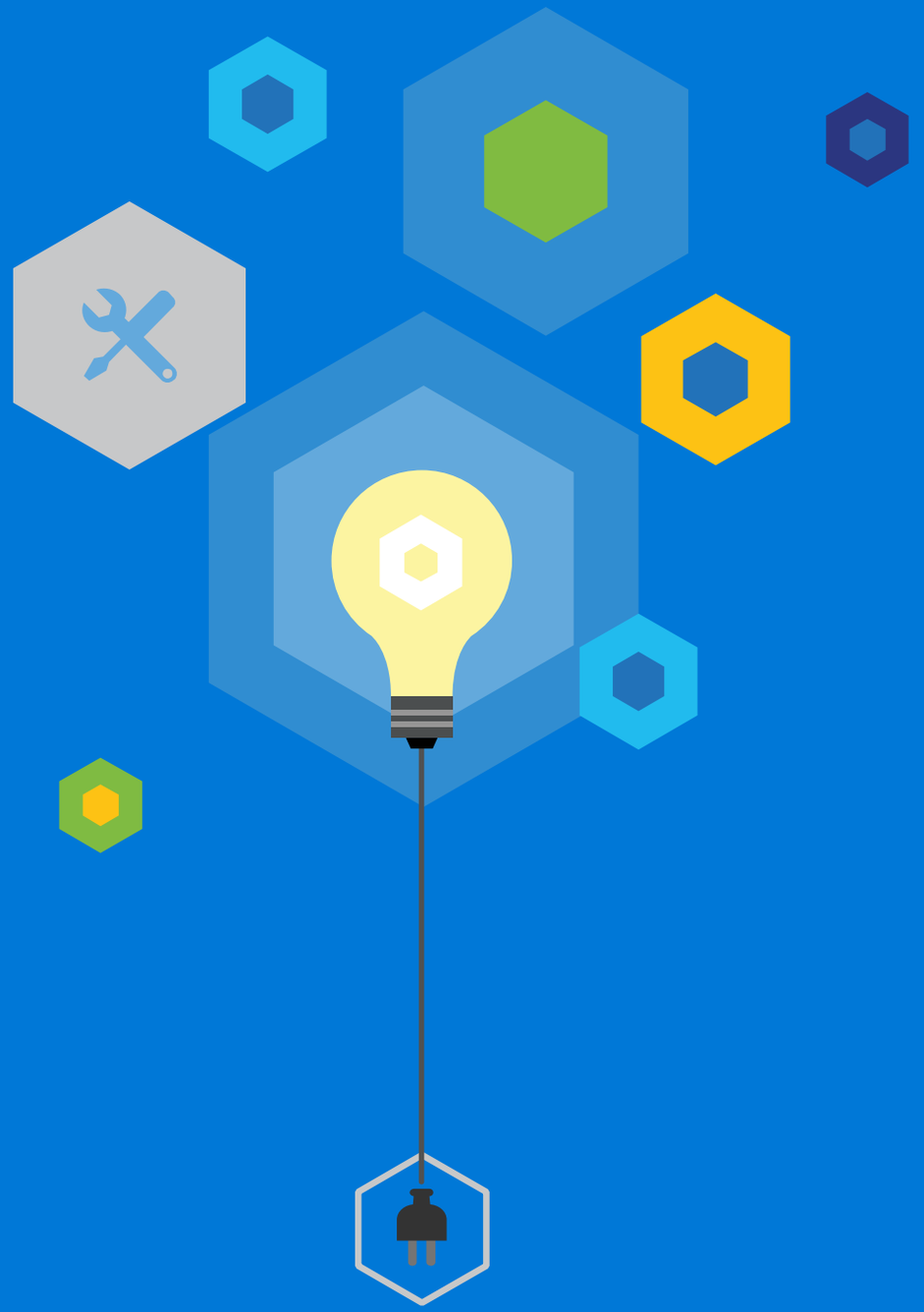
```
New-AzureVM –VM $myVM
New-AzureStorageAccount –StorageAccountName $acct
Set-AzureVNetConfig –ConfigurationPath -Path
```

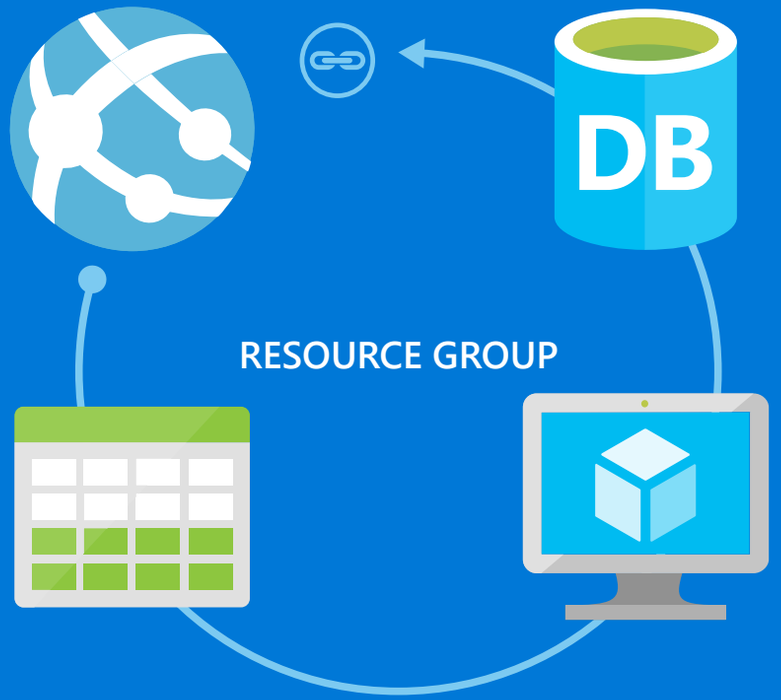
```
{
 "$schema": "https://../deploymentTemplate.json#",
 "contentVersion": "1.0.0.0",
 "parameters": {},
 "variables": {},
 "resources": [],
 "outputs": {}
}
```

You decide

Demo

Template format and concepts



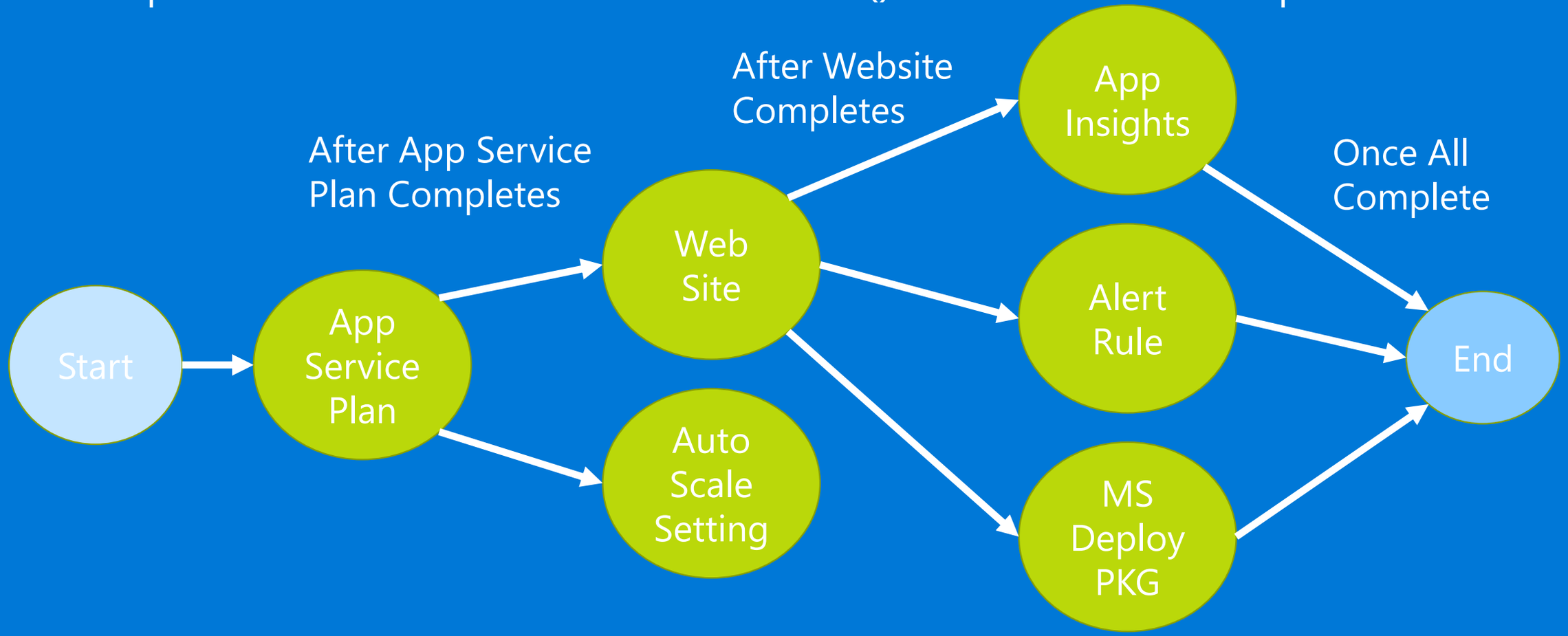


Deployment

- → tracks template execution
- → created within a resource group
- → allows nested deployments

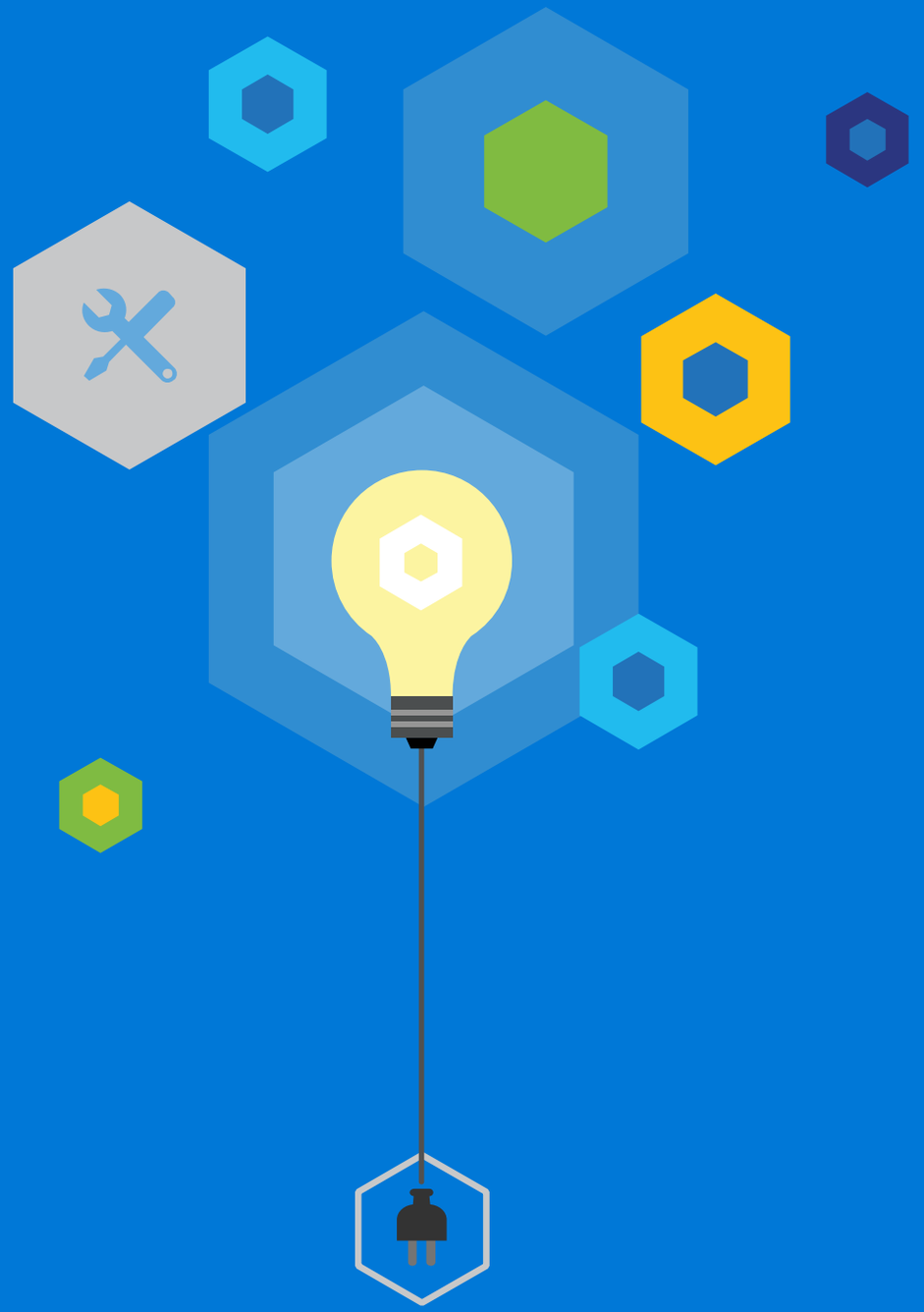
Template Execution

- Execution engine builds a state machine
- dependsOn and reference() establish dependencies



Demo

Dependencies



@ a glance - template language expressions*

```
base64encode(‘stringtoencode’)
concat(‘string’,’to’,’encode’)
copyIndex(offset)
listKeys(storageAccountResourceId, apiVersion)
padLeft(stringToPad,targetLength,paddingCharacter)
parameters(‘parameterName’)
providers(namespace, resourceType)
reference(resourceId,apiVersion)
resourceGroup()
resourceId(‘namespace/resourceType', ‘resourceName’)
subscription()
variables(‘variables’)
```

*Looking for examples? See these in action @ https://github.com/rjmax/ArmExamples

Organizing with Azure Resource Manager

- resource groups
- linked resources
- tags

Resource Tags

- Name-value pairs assigned to resources or resource groups
- Subscription-wide taxonomy
- Each resource can have up to 15 tags

Tagging Tips

- Tag by environment, e.g. dev/test/prod
- Tag by role, e.g. web/cache/db
- Tag by department, e.g. finance/retail/legal
- Tag by responsible party, e.g. Bob

Demo: Tags and Portal

Control with Azure Resource Manager

- role based access control
- audit logs
- resource locks

Role Based Access Control

- Allows secure access with granular permissions
- Assignable to users, groups, or service principals
- Built-in roles make it easy to get started

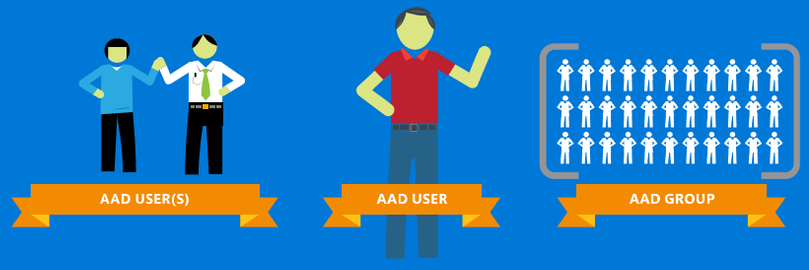
Role Based Access Control

SUBSCRIPTION

RESOURCE GROUPS

RESOURCES

ACCESS INHERITANCE



CONTRIBUTORS OWNER READERS

CONTRIBUTORS OWNER READERS

CONTRIBUTORS OWNER READERS

Demo: Role Based Access Control

Audit Logs

- journals all write/delete/actions
- central location
- common format

Demo: Audit Logs via PowerShell

Resource Locks

- Accidents happen. Resource locks help prevent them :)

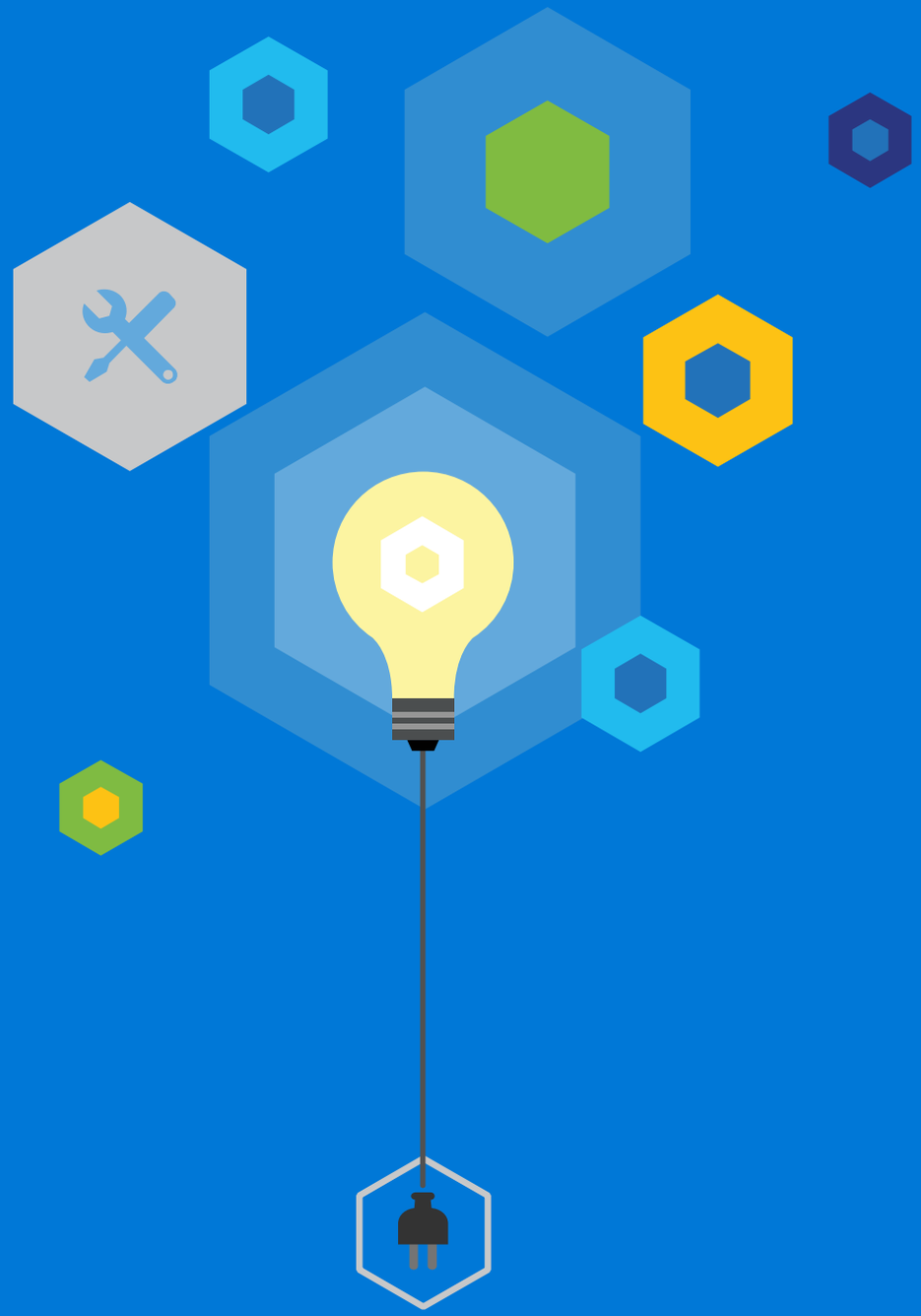
- Resource locks allow administrators to create policies which prevent accidental deletion.

Key Concepts

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

Demo (if time permits)

Rolling out git integration



Deploy a template today!

- Many IaaS examples available @ https://github.com/Azure/azure-quickstart-templates
- More language examples available @ https://github.com/rjmax/ArmExamples
- Documentation available @ http://azure.microsoft.com/en-us/documentation/articles/resource-group-overview/

goto; conference

Please

Remember to rate this session

Thank you!

Join the conversation #gotocph

Q & A