Welcome to AZUG!

AZUG
Azure User Group Belgium

codit
proximus NXT

Today’s event host

codit

proximus NXT

Agenda

17:30 – 18:30

Welcome, Food & Misc

18:30 – 19:15

Azure Deployment Stacks (Annelotte)

19:15 – 19:30

Break

19:30 – 20:15

Real-World Azure Arc in Action (Tom)

Platinum partners

ae

AXXES_

cegeka

codit
proximus NXT

delaware

inetum

Microsoft

spikes
Powered by axi

ZURE

Silver partners

Arxus

Integration.team

U2U

Next Sessions

www.azug.be

Subscribe to our newsletter

- 02/04/2026: Azug @ Zure
  - Speakers: Mike Martin & Pieter Vandenheede
- 10 + 11/12: CloudBrew 2026 edition

Have fun this evening

And learn something new...

AZUG
Azure User Group Belgium

Azure Deployment stacks

AZUG 12/02/2026

Annelotte Mons

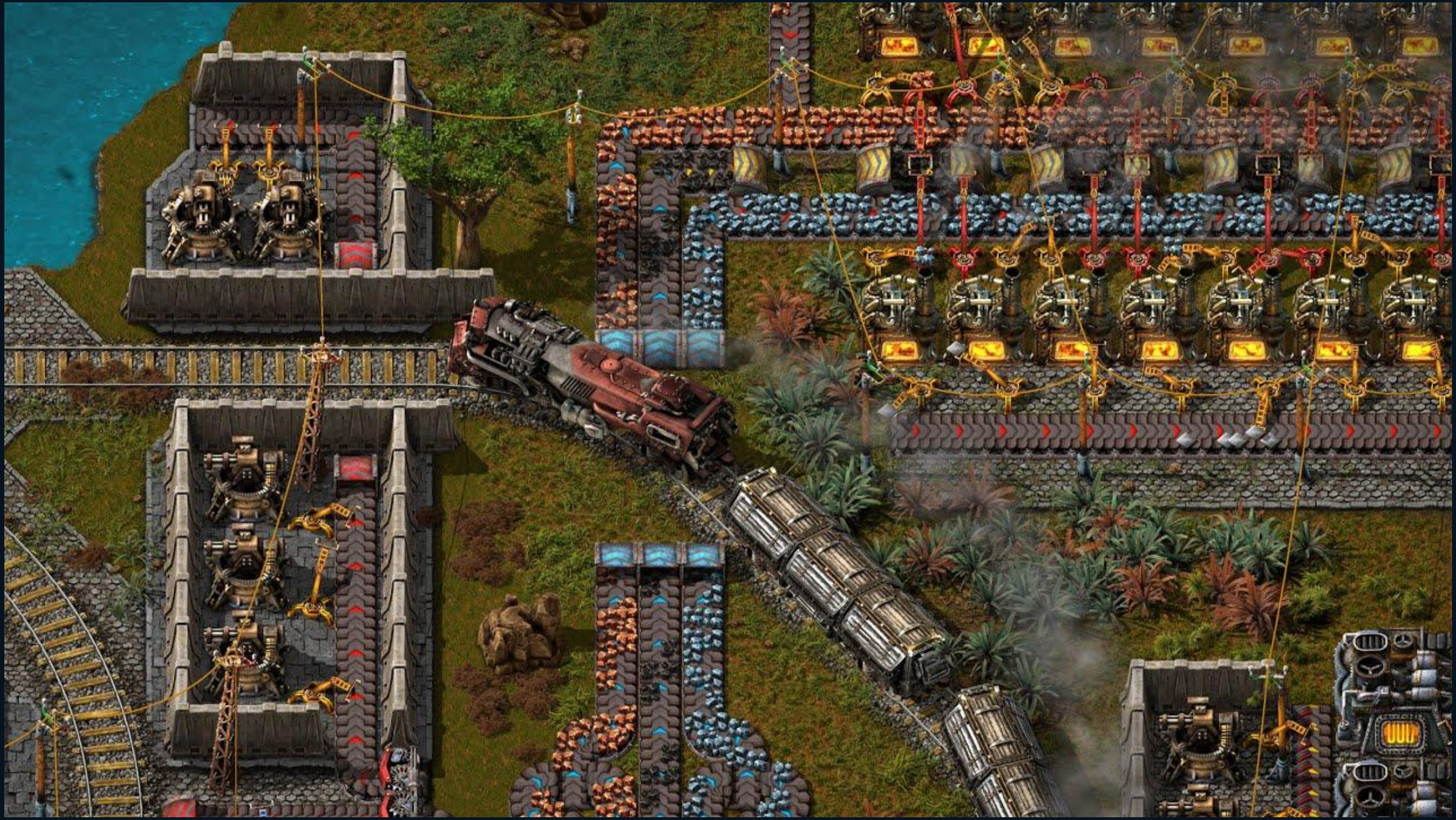
Sr. (Integration) Consultant @ Codit

annelotte-mons

Annelotte-Mons

AZUG 12/02/2026

Introduction



factorio

Resource Management

Resource Management

Automation

Idempotency

Modularity

…

Traceability
Observability

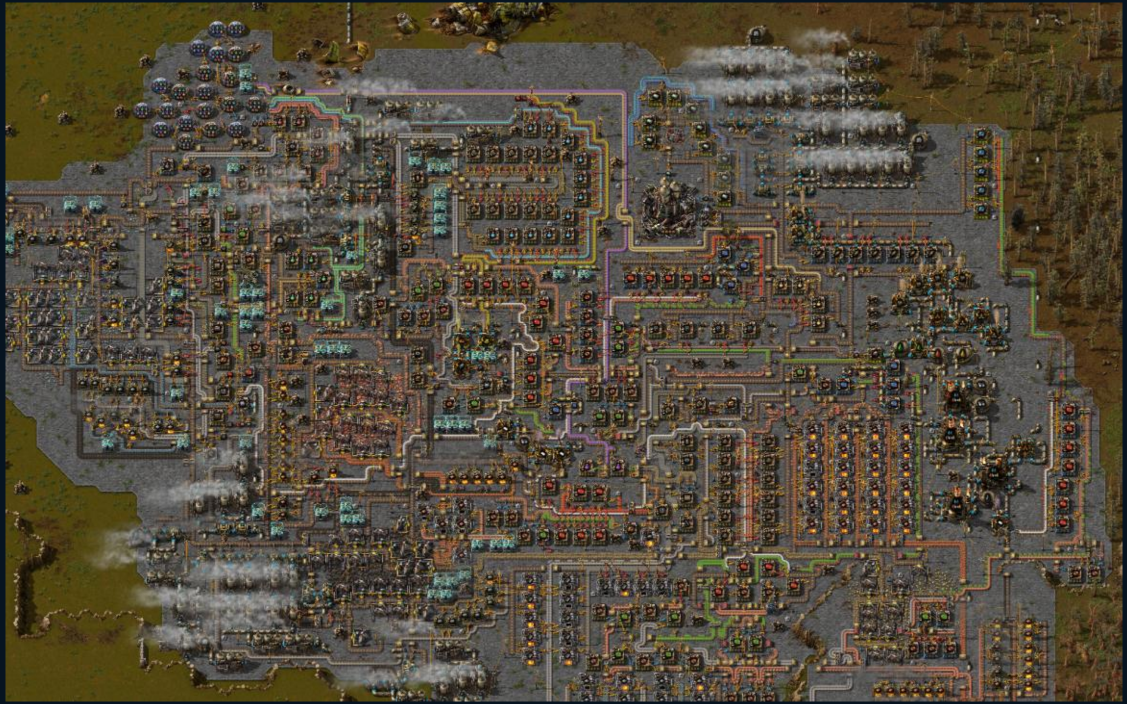
Governance

Scalability

Resource Management - Azure

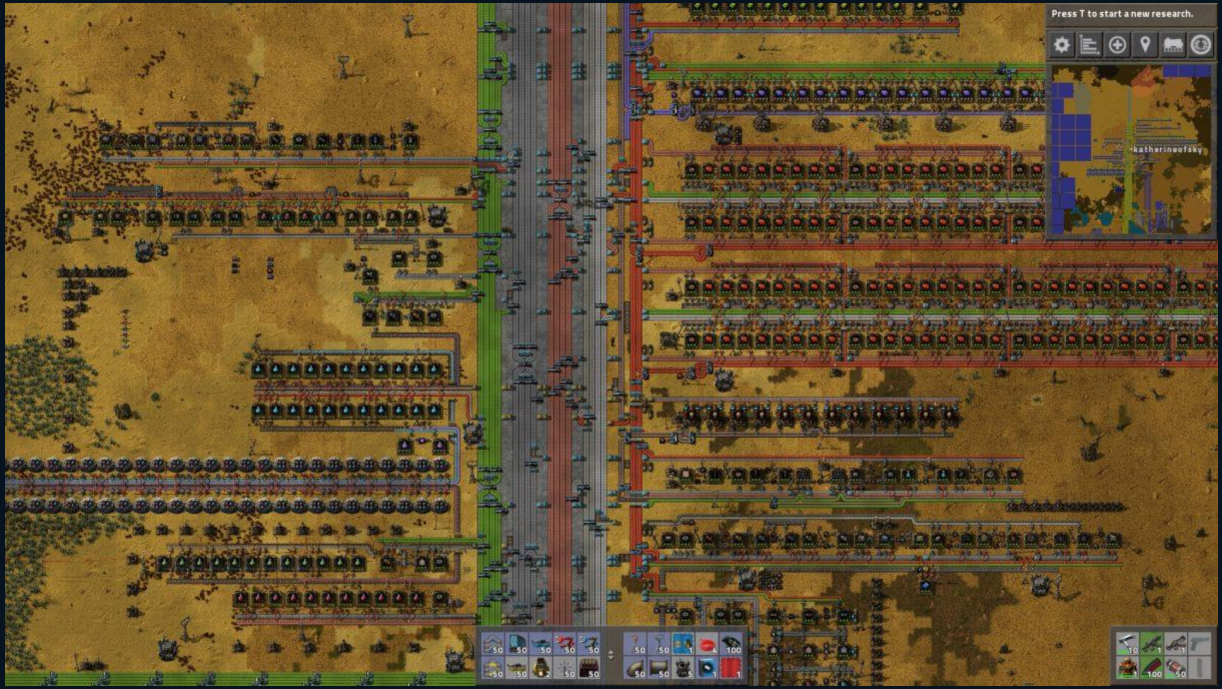
- IaC (Bicep, ARM, Terraform, ...)
- Git
- Tests
  - Unit/Integration/Load/...
- DevOps
  - Tickets
  - Pipelines
  - Gates/Approvals/Checkpoints/...

Resource Management



“Spaghetti factory”

Vs.

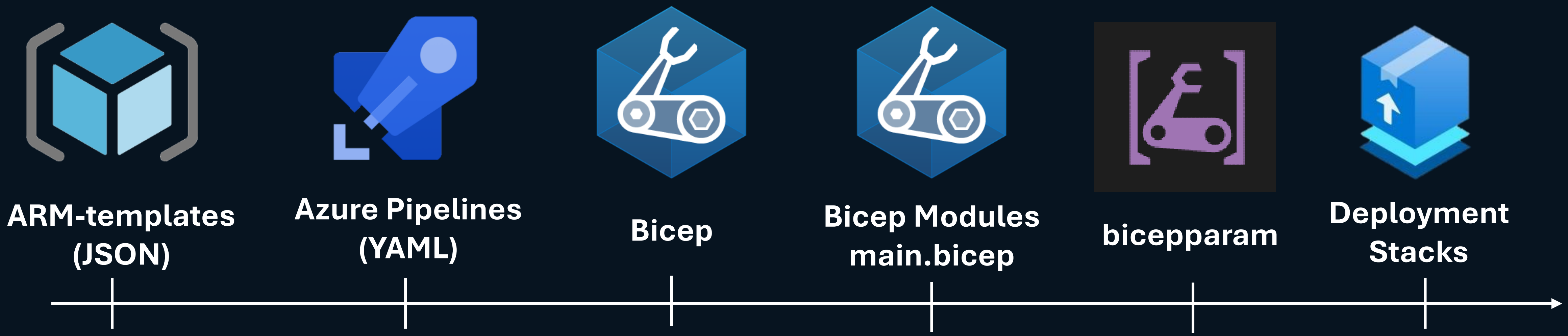


Formatted/modular factory

Evolution of Resource Management

codit proximus NXT

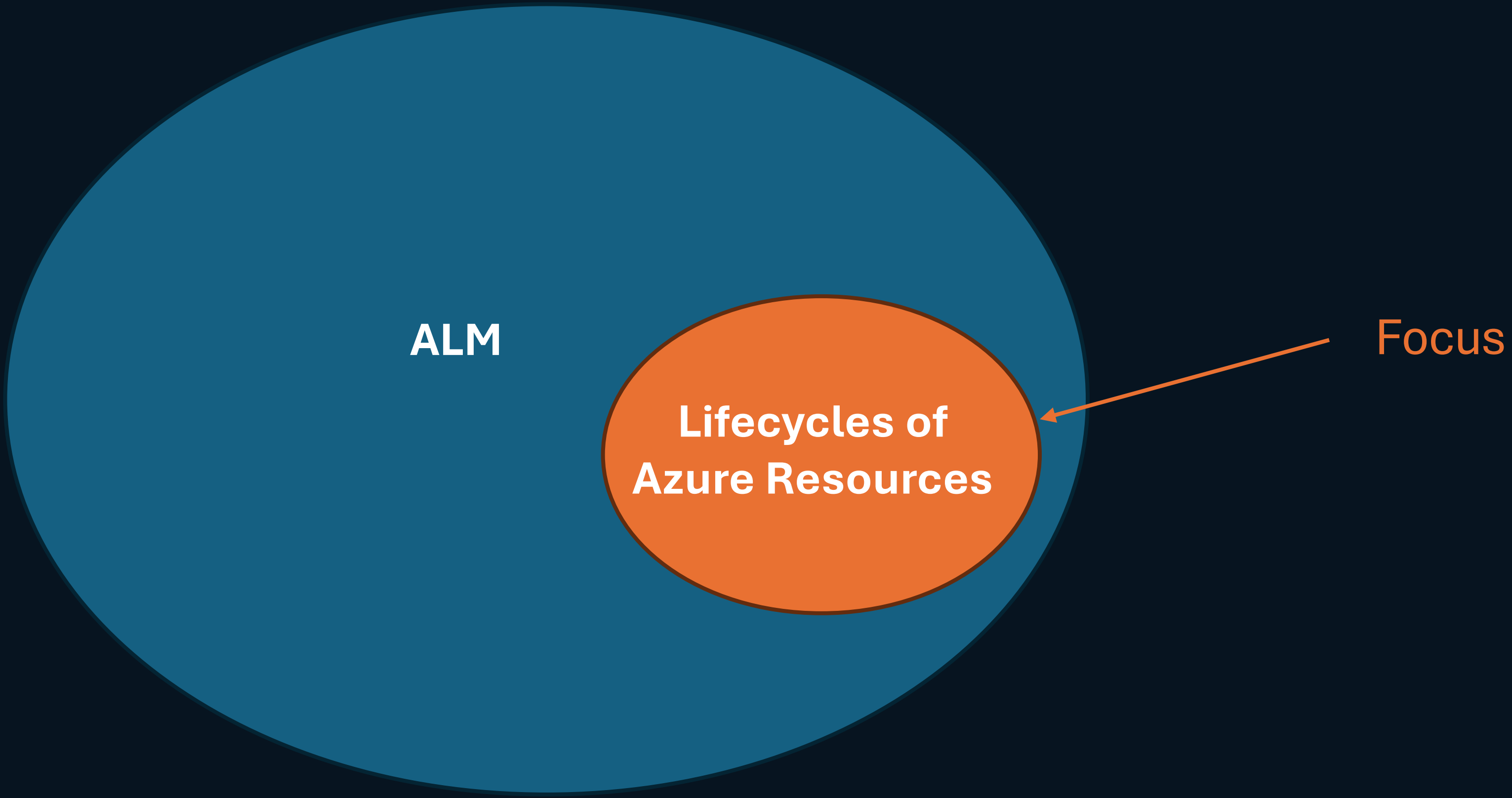
Some notable [r]evolutions in Azure resource management over the years



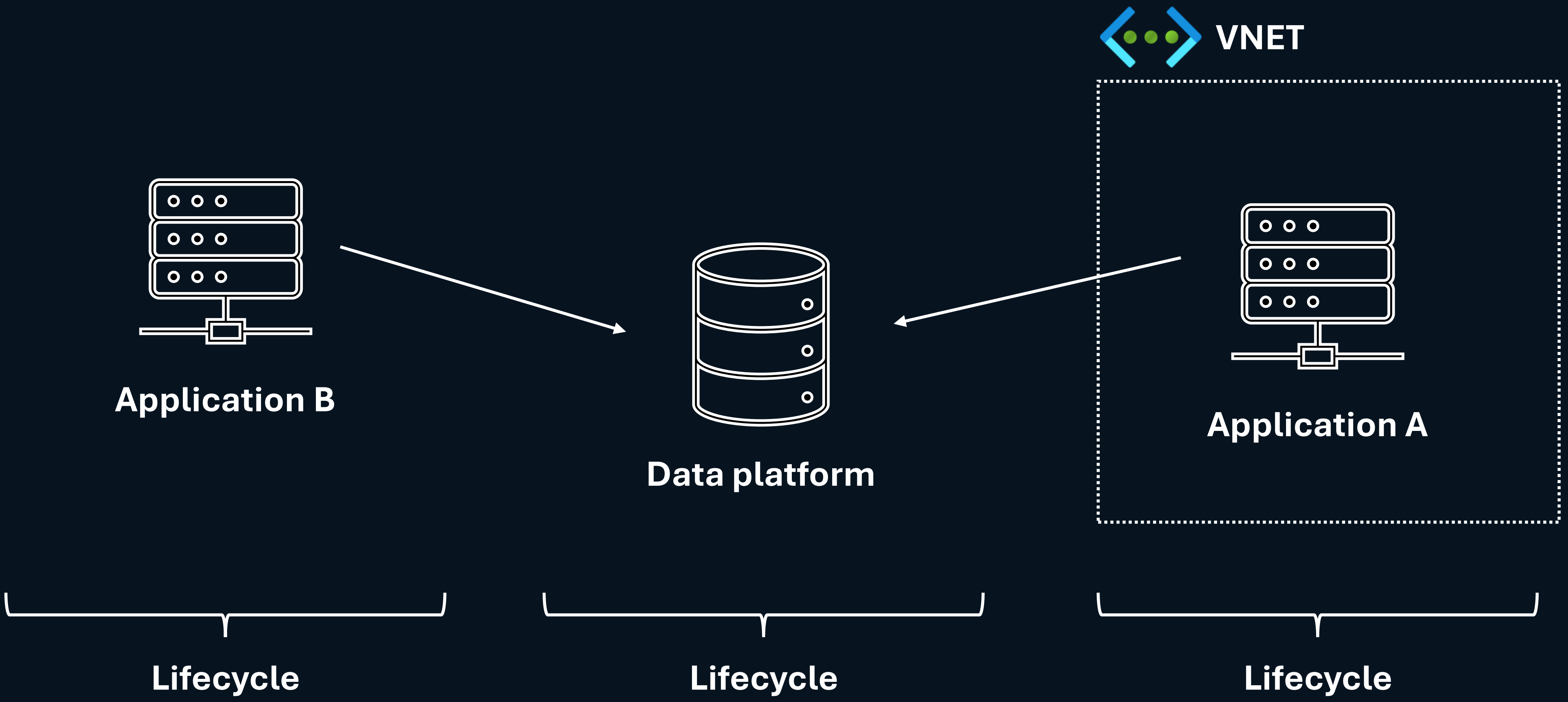
Lifecycles

Lifecycles

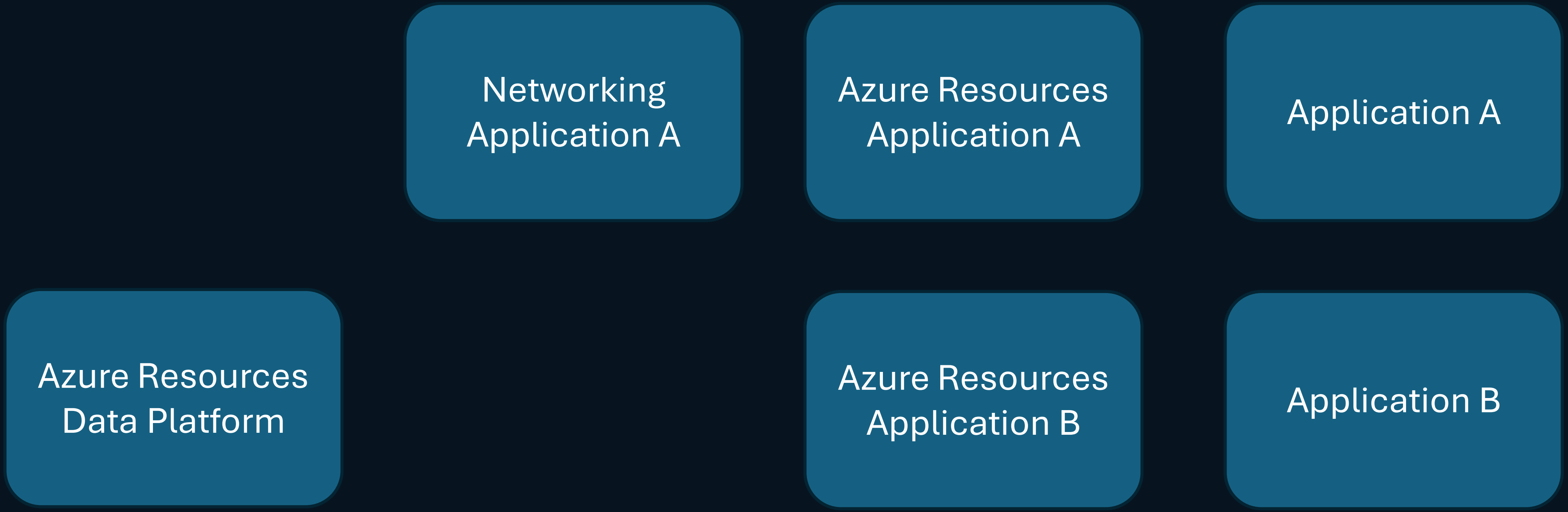
Application lifecycle management (ALM) is the combination of people, tools, and processes that manage the life cycle of an application from conception to end of life.



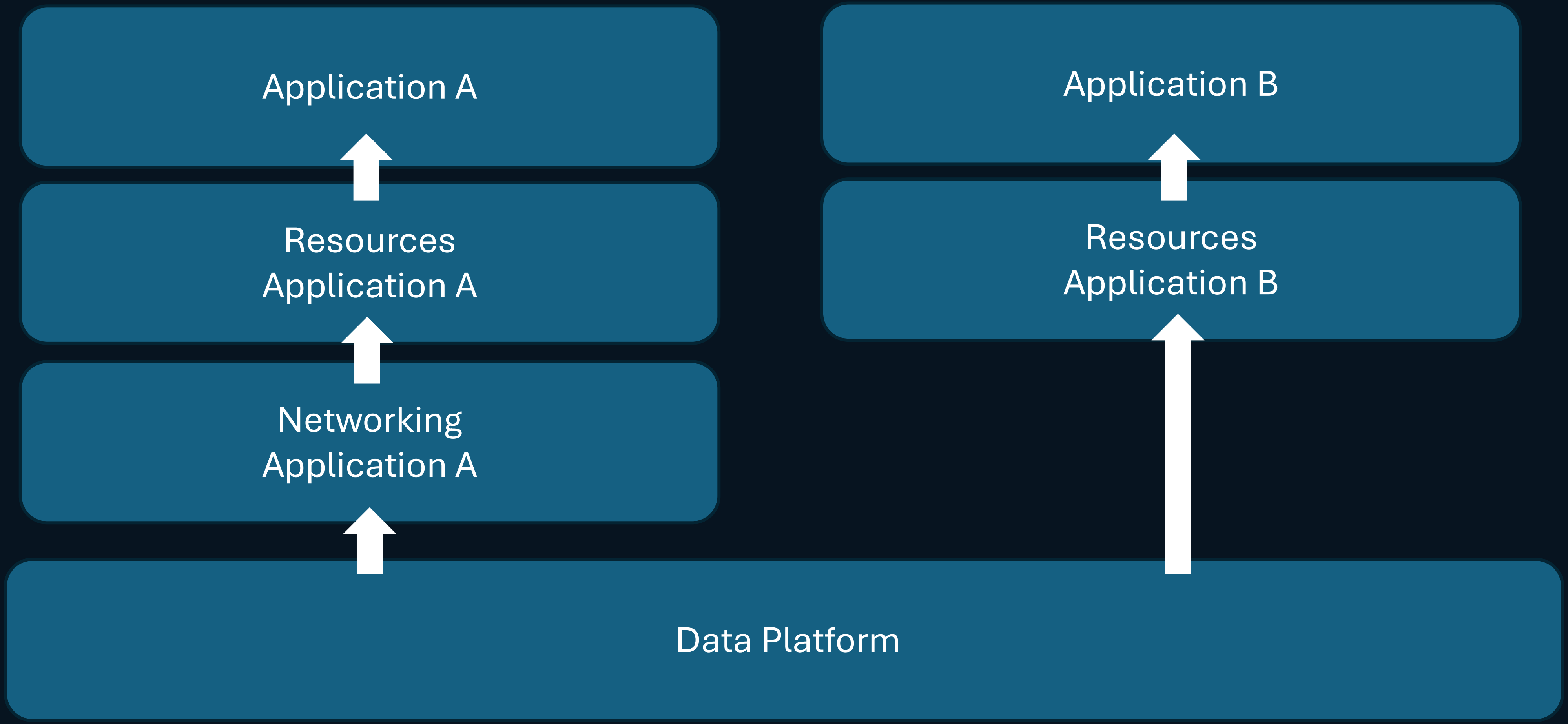
Lifecycles - Example



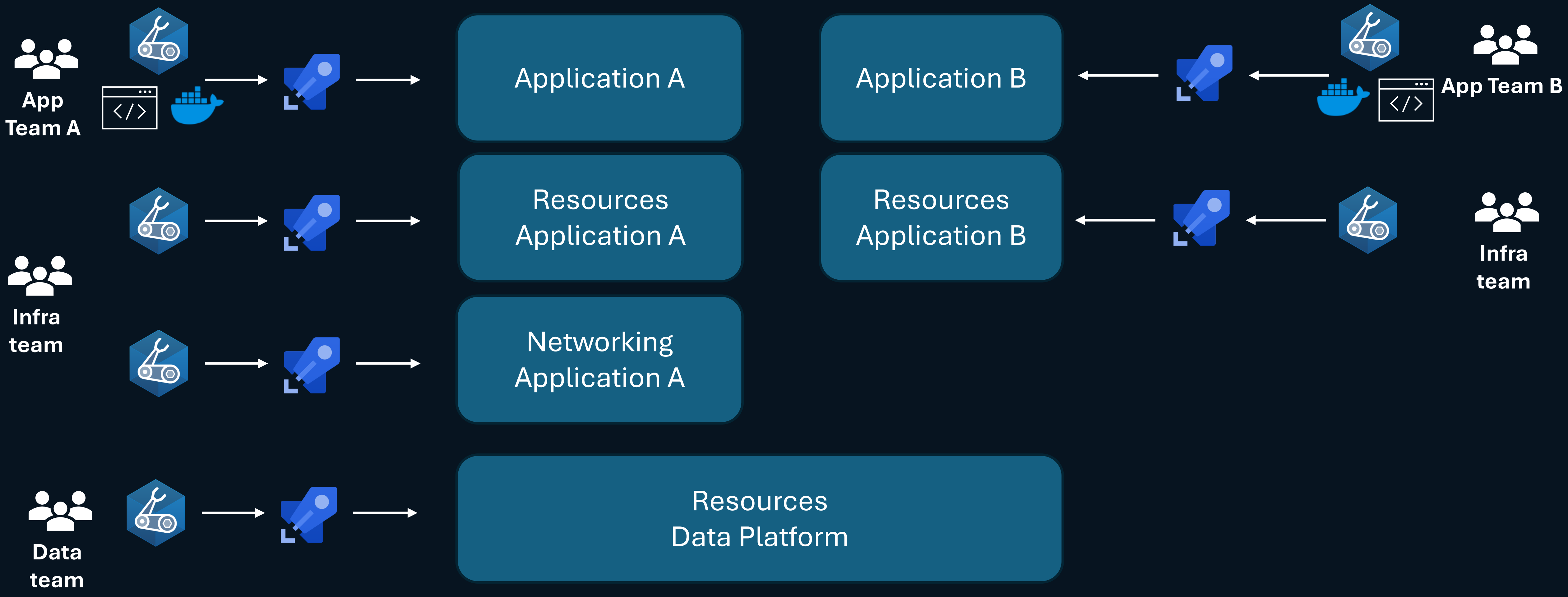
Lifecycles – Digging deeper


Networking Application A
Azure Resources Application A
Application A
Azure Resources Data Platform
Azure Resources Application B
Application B

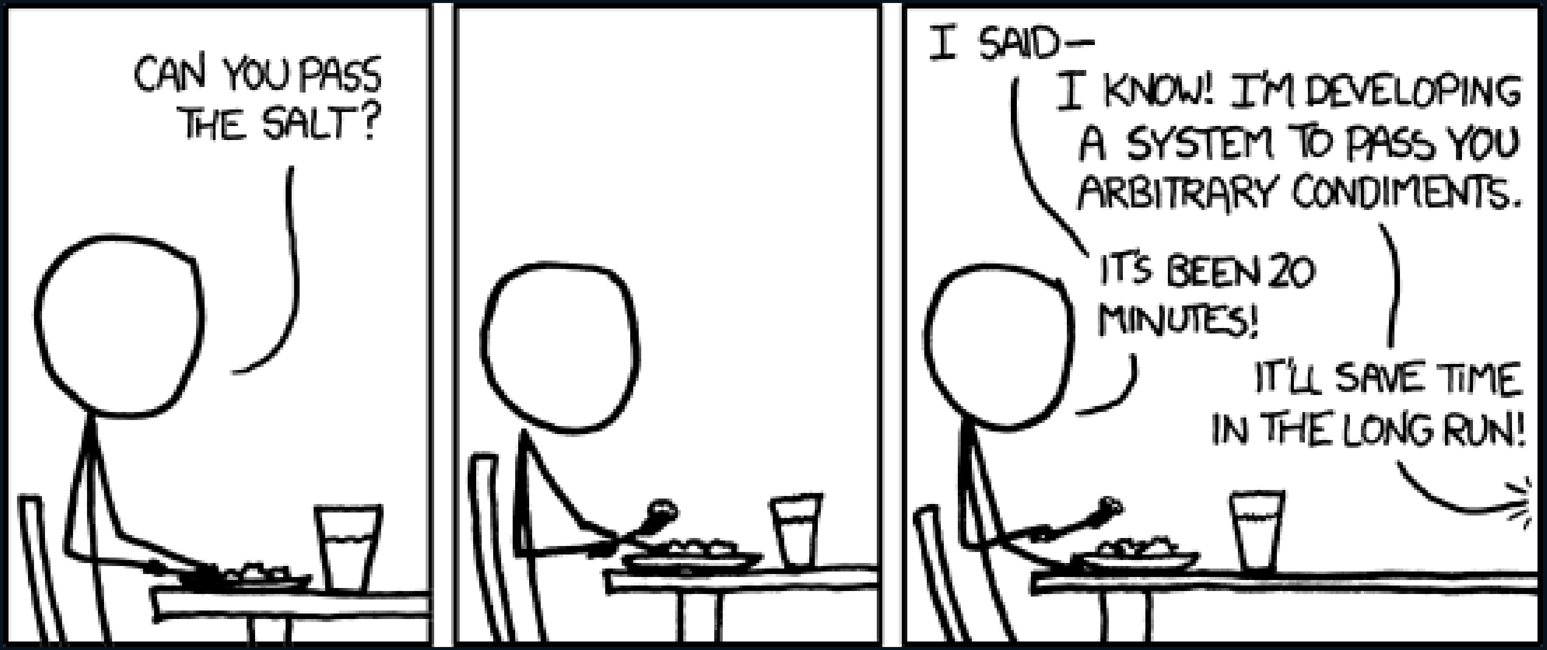
Lifecycles - Dependencies



Lifecycles – Scope of deployment



So, what about those Deployment stacks?


CAN YOU PASS THE SALT?
I SAID—
I KNOW! I'M DEVELOPING A SYSTEM TO PASS YOU ARBITRARY CONDIMENTS.
IT'S BEEN 20 MINUTES!
IT'LL SAVE TIME IN THE LONG RUN!

We're getting there, I promise!

Azure Deployment stacks

Deployment Stacks

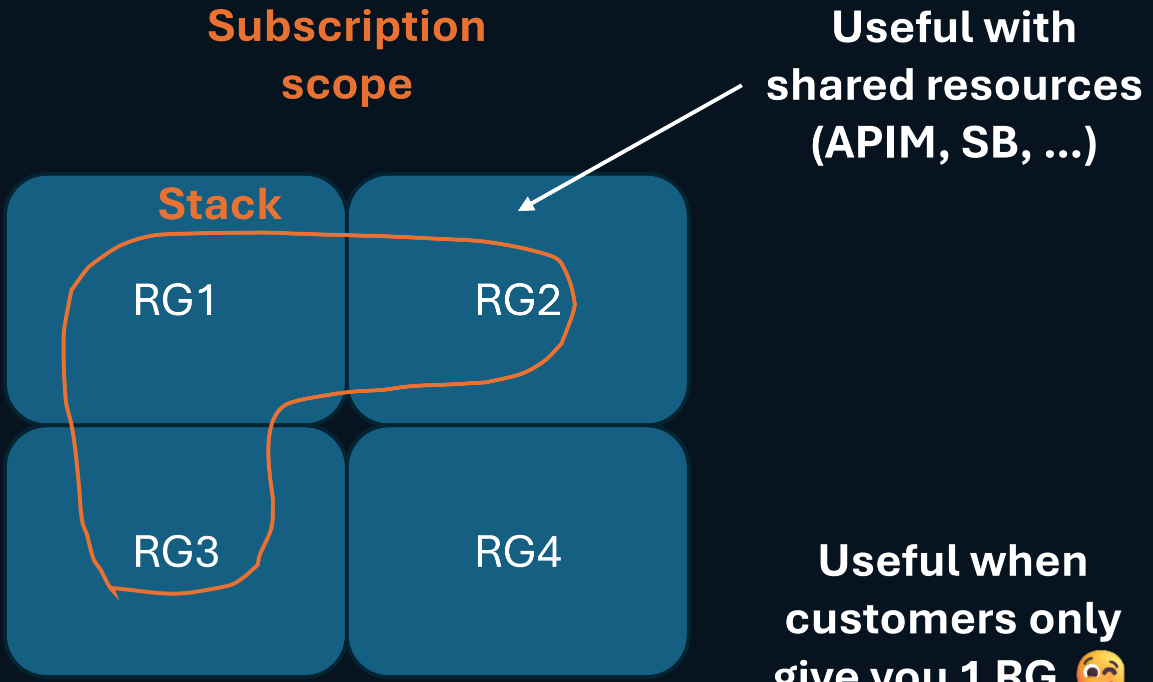
- Native Azure Resource Type
  - Manage a set of Azure Resources
  - Improved Resource Lifecycle management
    - Supports Stateful behavior (~ Terraform)
    - Actions on “unmanaged resources” (**action on unmanage**)
  - Multiple scopes (Mgmt Group, Subscription, ResourceGroup)

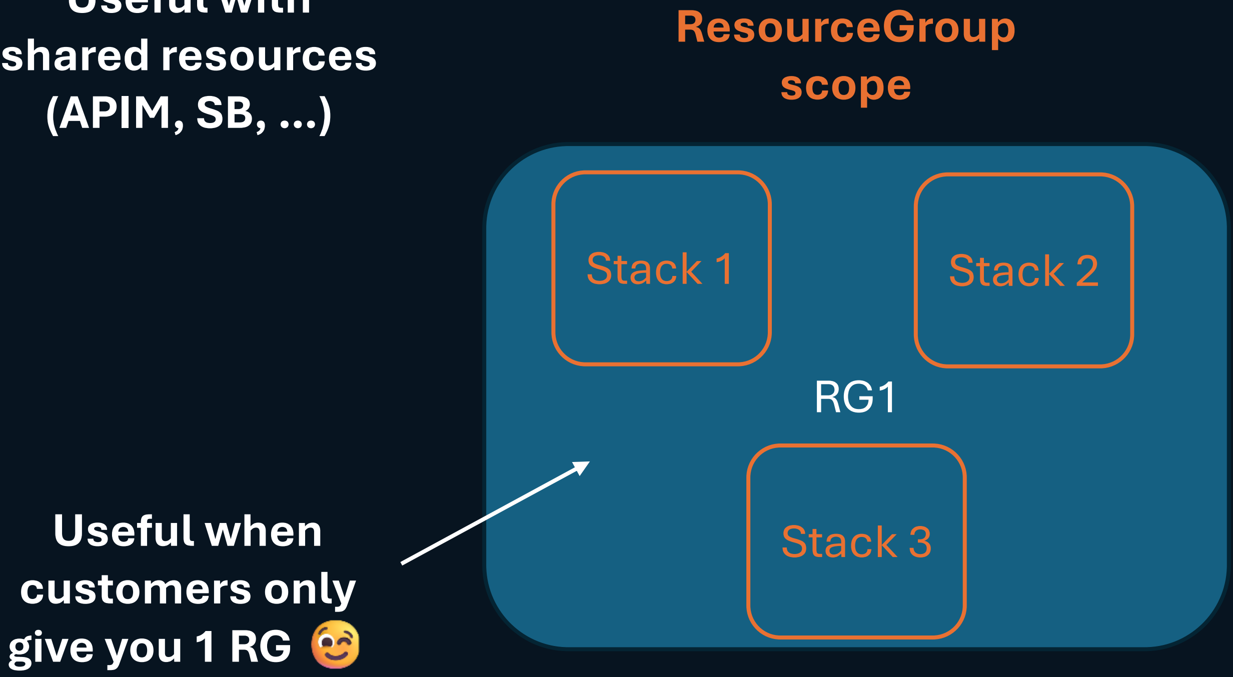
- Extra Governance control
  - Control plane
  - Stack = Scope for RBAC roles
  - Delete / Change prevention (**deny settings**)
    - Drift control

… But don't we have resourceGroups for that?

Deployment stacks are an extra management layer on top

- **Features for resource management (lifecycle + governance)**
- **Tied to a source template ('main.bicep' -- no ad-hoc resources)**
- **Difference in scopes**





Deploying deployment stacks

codit proximus NXT

- Prerequisites:
  - main.bicep with correct targetScope (mgmt, subs, rg)
  - Sufficient permissions (e.g. Azure Deployment Stack Contributor)

Az CLI (Azure PowerShell also supported)

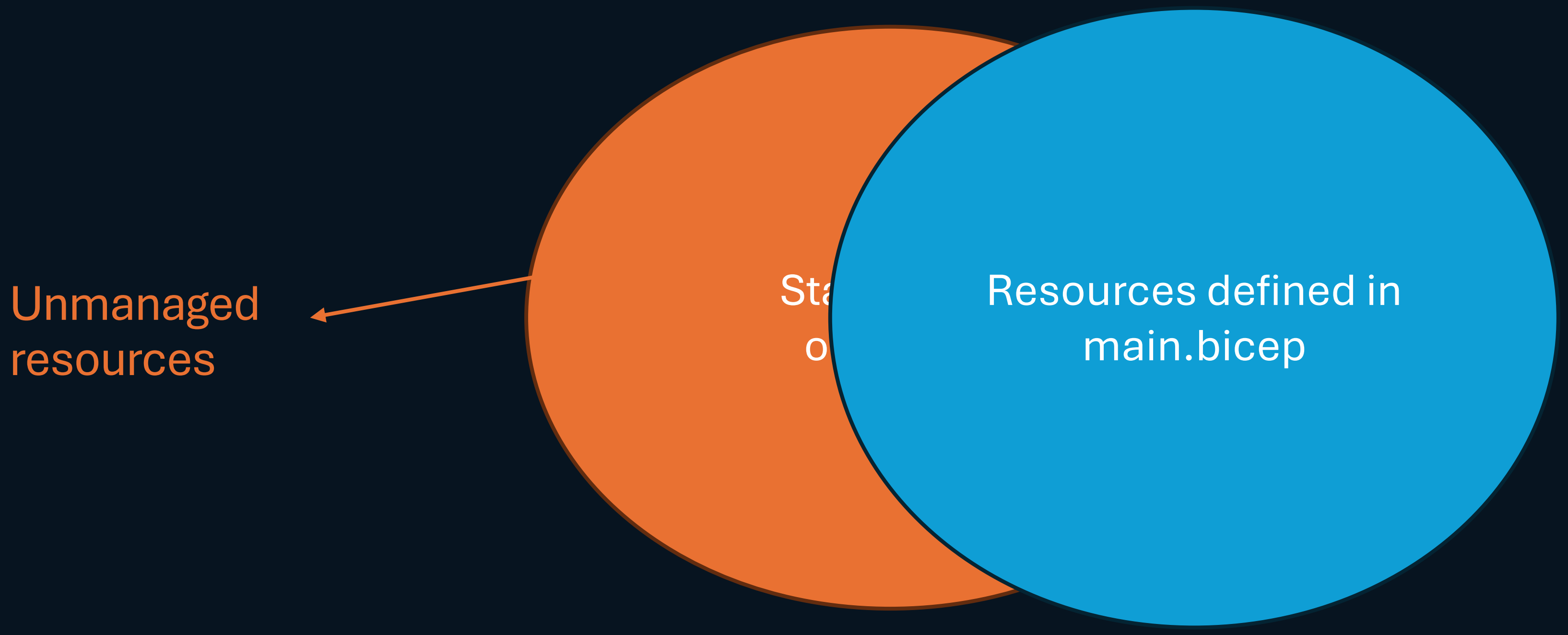
```
az stack group create
  --name <stackname>
  --resource-group <rg-r
  --template-file <path>
  --parameters <path>
  --action-on-unmanage <
  --deny-settings-mode <
```

```
az stack sub create
  --name <stackname>
  --location <location>
  --template-file <path>
  --parameters <path>
  --action-on-unmanage <Del
  --deny-settings-mode <non
```

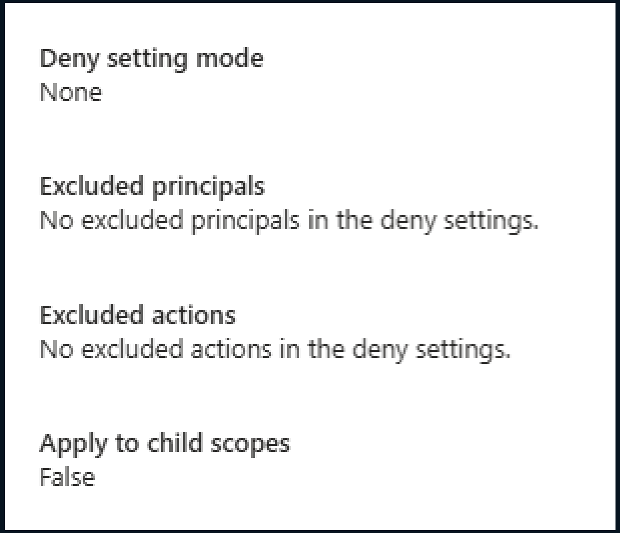
```
az stack mg create
  --name <stackname>
  --location <location>
  --template-file <path>
  --parameters <path>
  --deployment-subscription <subscriptionId>
  --action-on-unmanage <DeleteAll/DeleteResouces/DetachAll>
  --deny-settings-mode <none/denyDelete/denyWriteAndDelete>
```

Action on unmanage

- Stateful behavior
- Action for ‘Unmanaged resources’
  - DetachAll / DeleteAll / DeleteResources



Deny Settings



Deny setting mode
None

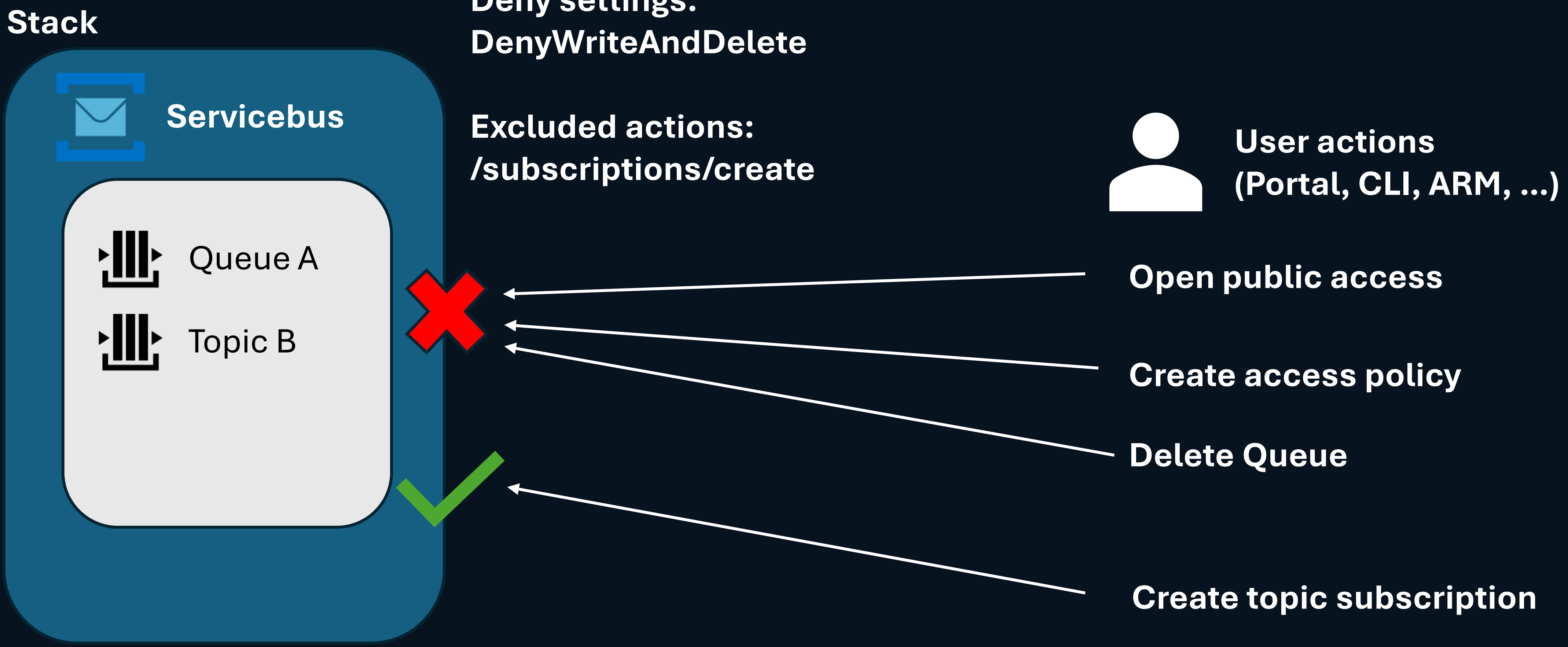
Excluded principals
No excluded principals in the deny settings.

Excluded actions
No excluded actions in the deny settings.

Apply to child scopes
False

- Requires elevated permissions to manage
  - e.g. Deployment Stack owner
- Deny Settings mode
  - None / DenyDelete / DenyWriteAndDelete
  - Drift control
- Applies to control plane only (not data plane)
- Excluded principals/actions
- Apply to child scopes
  - E.g. ServiceBus – ServiceBus/Queue

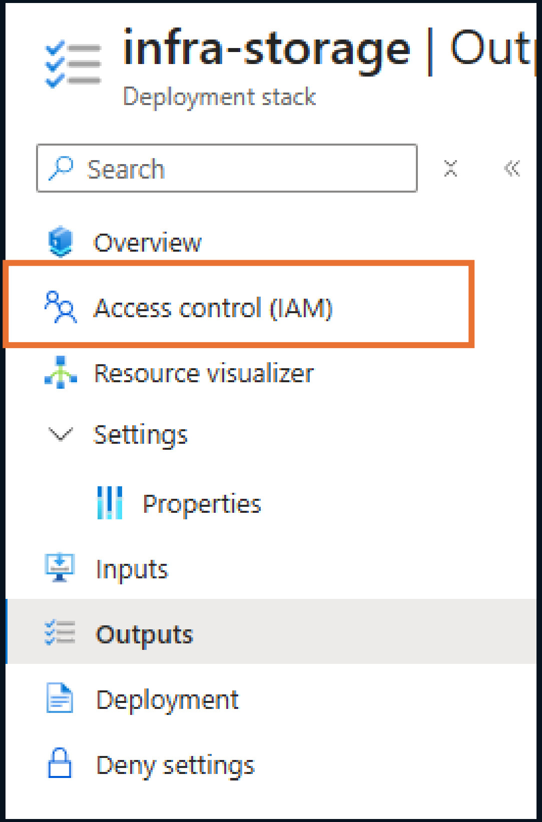
Deny Settings - Example



RBAC

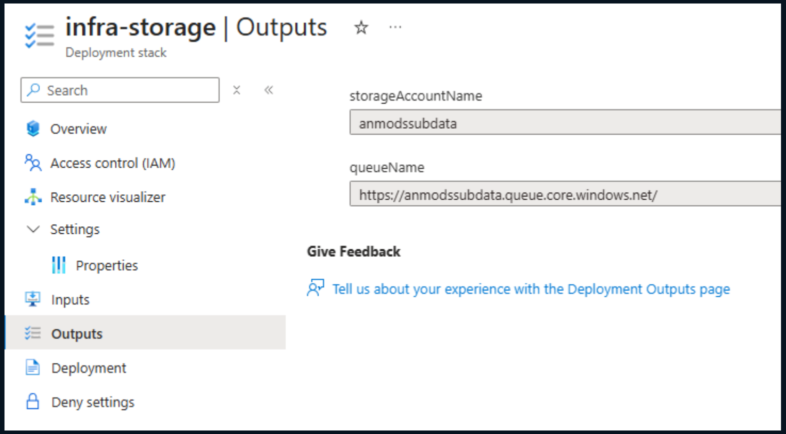
- Possibility to assign roles
  - Applies to all resources within the stack

  E.g. Blob Data contributor for a team.
  Can see data of blobs that belong to their stack.
  When other blob containers (not managed by stack) exist on shared storageAccounts, it will not grant access.



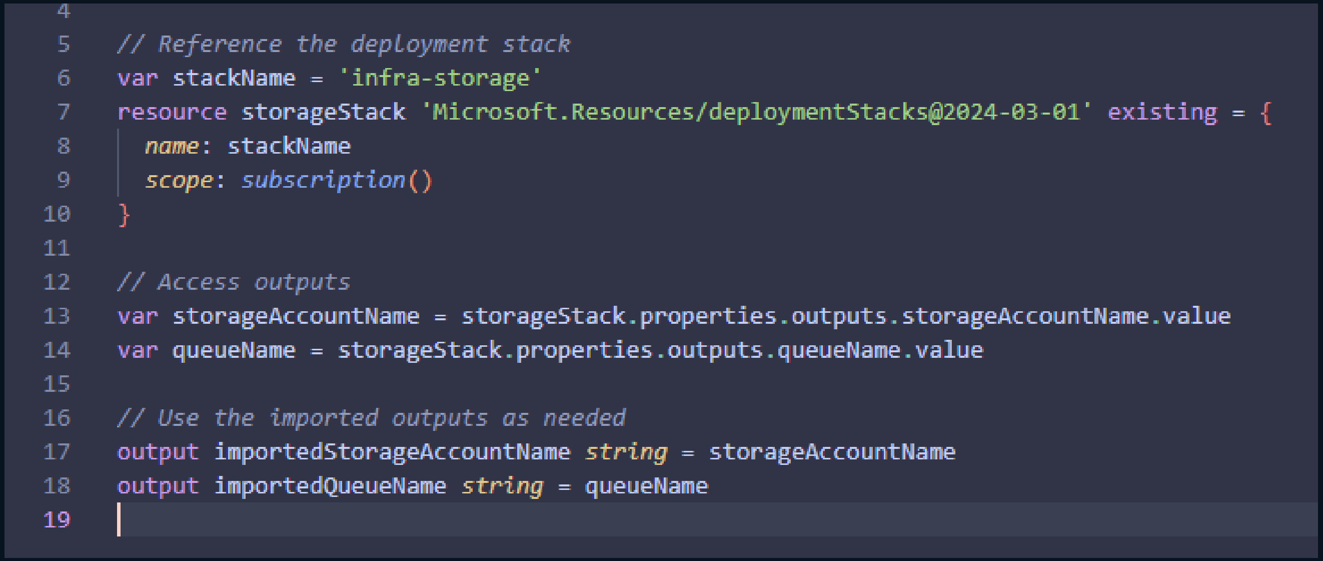
Stack outputs

- Similar to outputs of regular deployments
- Easily queryable
- Reflect latest state of whats deployed
- Reduces *implicit* dependencies
- DRY concept



storageAccountName: anmodssubdata

queueName: https://anmodssubdata.queue.core.windows.net/



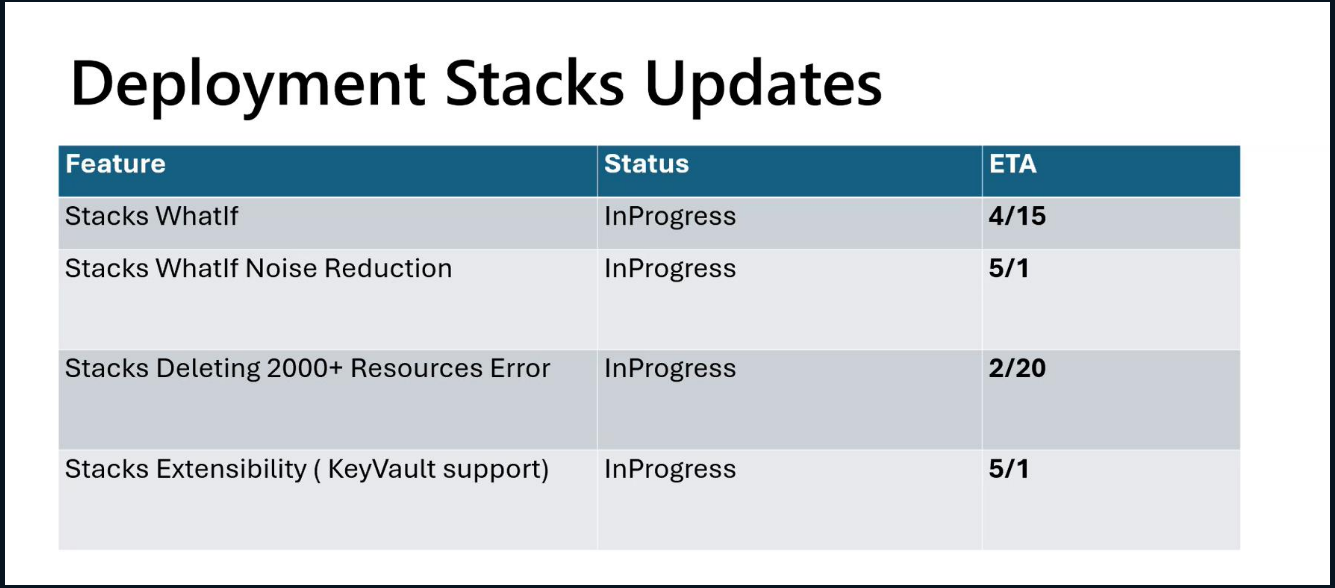
```
// Reference the deployment stack
var stackName = 'infra-storage'
resource storageStack 'Microsoft.Resources/deploymentStacks@2024-03-01' existing = {
  name: stackName
  scope: subscription()
}

// Access outputs
var storageAccountName = storageStack.properties.outputs.storageAccountName.value
var queueName = storageStack.properties.outputs.queueName.value

// Use the imported outputs as needed
output importedStorageAccountName string = storageAccountName
output importedQueueName string = queueName
```

Bicep – Deployment Stack ‘what-if’

- Coming soon™
- Overview of what actions will be taken prior to deploy
  - Detached/Deleted/…
- → Approval gate



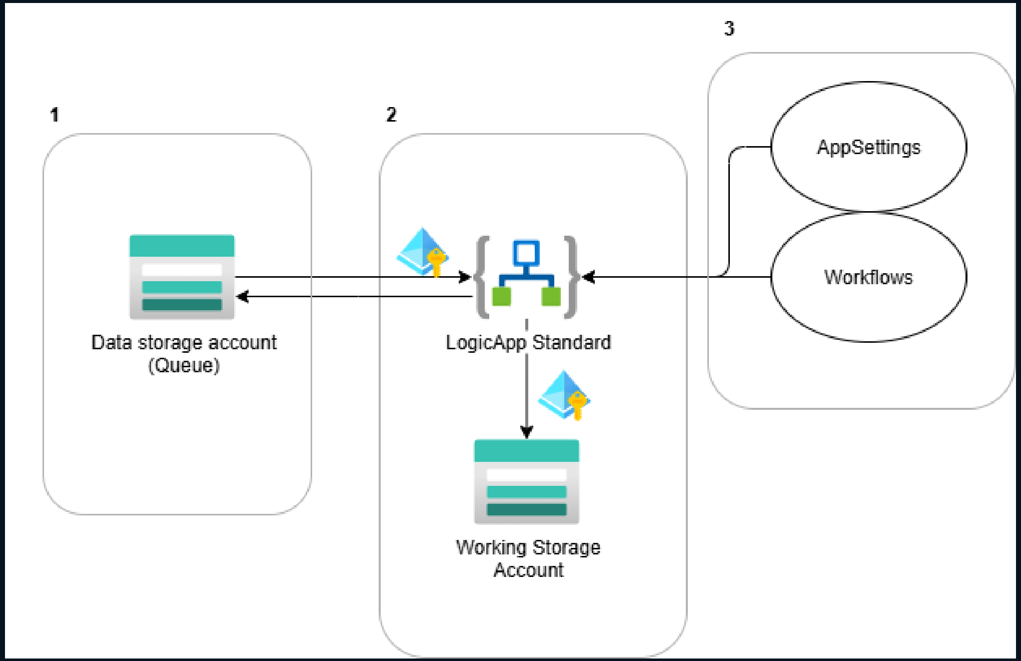
Deployment Stacks Updates

Feature	Status	ETA
Stacks WhatIf	InProgress	**4/15**
Stacks WhatIf Noise Reduction	InProgress	**5/1**
Stacks Deleting 2000+ Resources Error	InProgress	**2/20**
Stacks Extensibility (KeyVault support)	InProgress	**5/1**

Demo

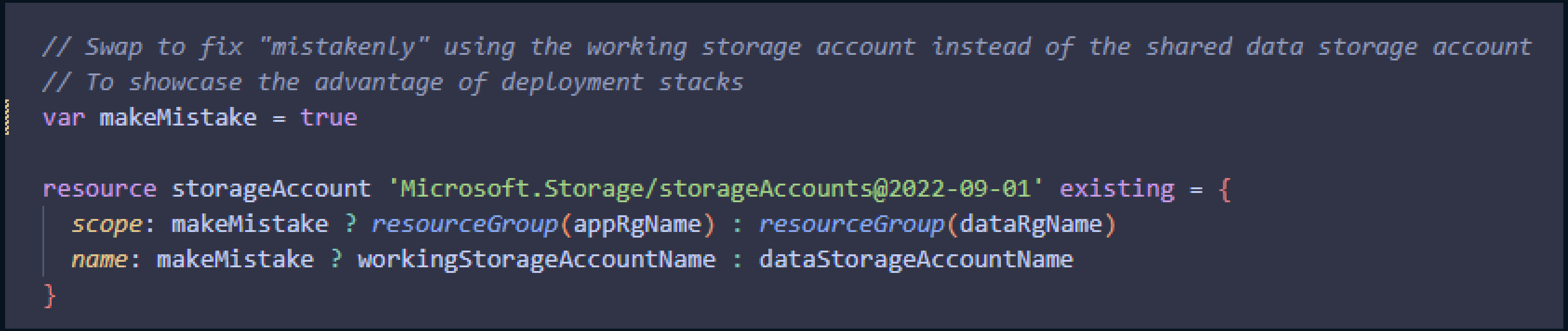
Demo

- Github: Annelotte-Mons/demo-deploymentStacks



Demo highlight

“Accidentally” assign RBAC on wrong storage account



```
// Swap to fix "mistakenly" using the working storage account instead of the shared data storage account
// To showcase the advantage of deployment stacks
var makeMistake = true

resource storageAccount 'Microsoft.Storage/storageAccounts@2022-09-01' existing = {
  scope: makeMistake ? resourceGroup(appRgName) : resourceGroup(dataRgName)
  name: makeMistake ? workingStorageAccountName : dataStorageAccountName
}
```

Correct our mistake, redeploy stack using “DeleteResources” mode

Previous (wrong RBAC) should be removed

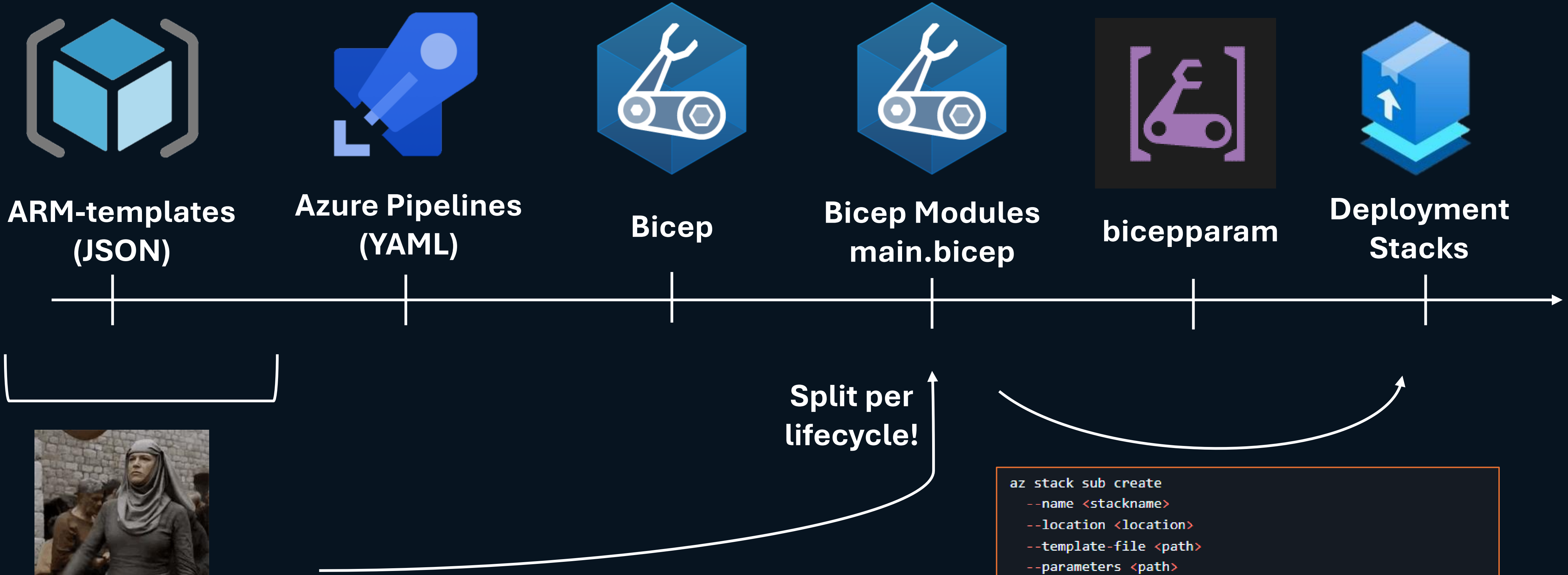
No more “Orphaned RBACs”!

Migration guide

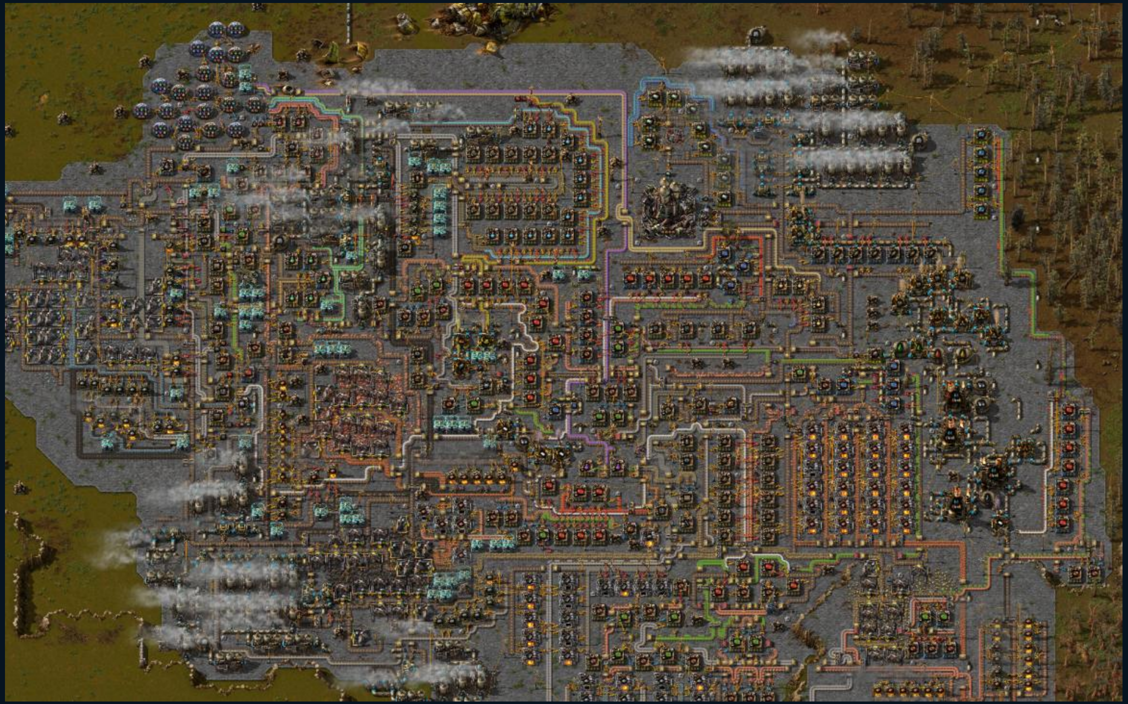
Road to implementing deployment stacks

Migration guide

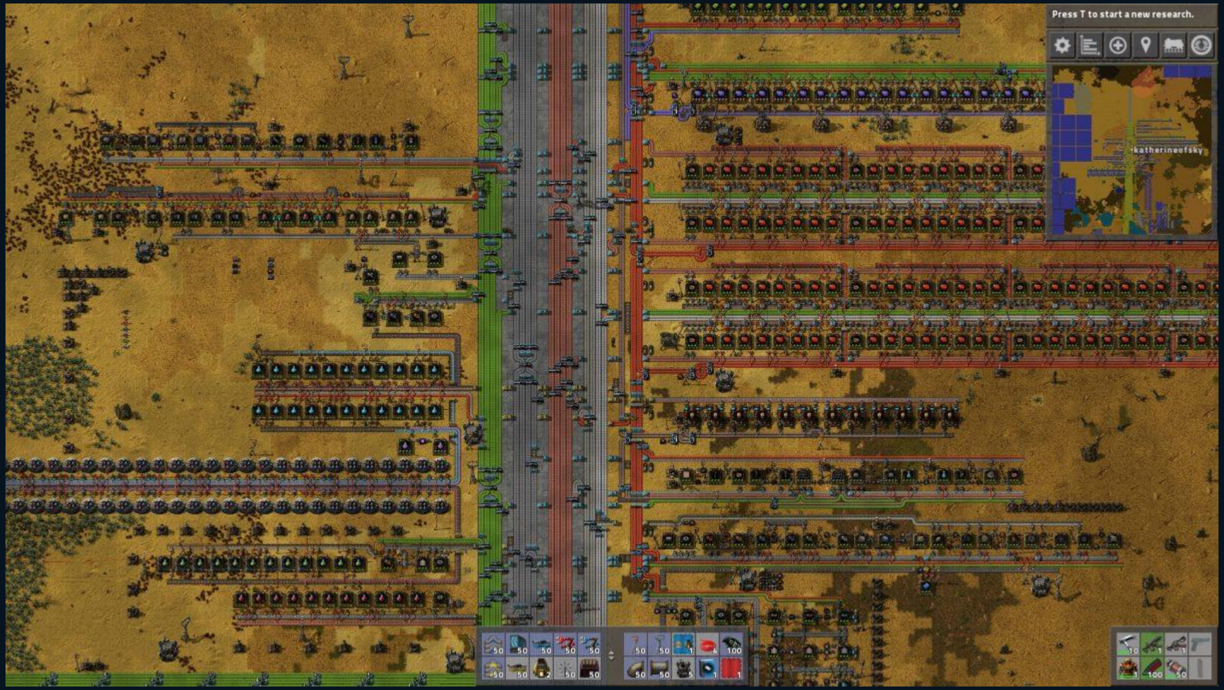
Where is your project situated?



```
az stack sub create
  --name <stackname>
  --location <location>
  --template-file <path>
  --parameters <path>
  --action-on-unmanage <DeleteAll/DeleteResouces/DetachAll>
  --deny-settings-mode <none/denyDelete/denyWriteAndDelete>
```



Vs.



Let's save the spaghetti for dinner

codit
proximus NXT

Q&A

Thank you for your time!