

DATA SATURDAY #40 - DATA SATURDAY MÜNCHEN 2024

Infrastructure as
Code,
why should I care?

Marisol Steinau



Thank you!

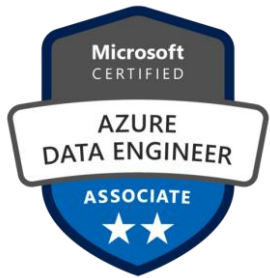


Many thanks to our sponsors, without whom such an event would not be possible.

Who am I?

Marisol Steinau

- Autodidact Data Enthusiast
- > 8 years experience using Microsoft Data Platform
- Senior Data Engineer @Be-terna
- LinkedIn: [linkedin.com/in/marisol-steinau-bb1253253](https://www.linkedin.com/in/marisol-steinau-bb1253253)

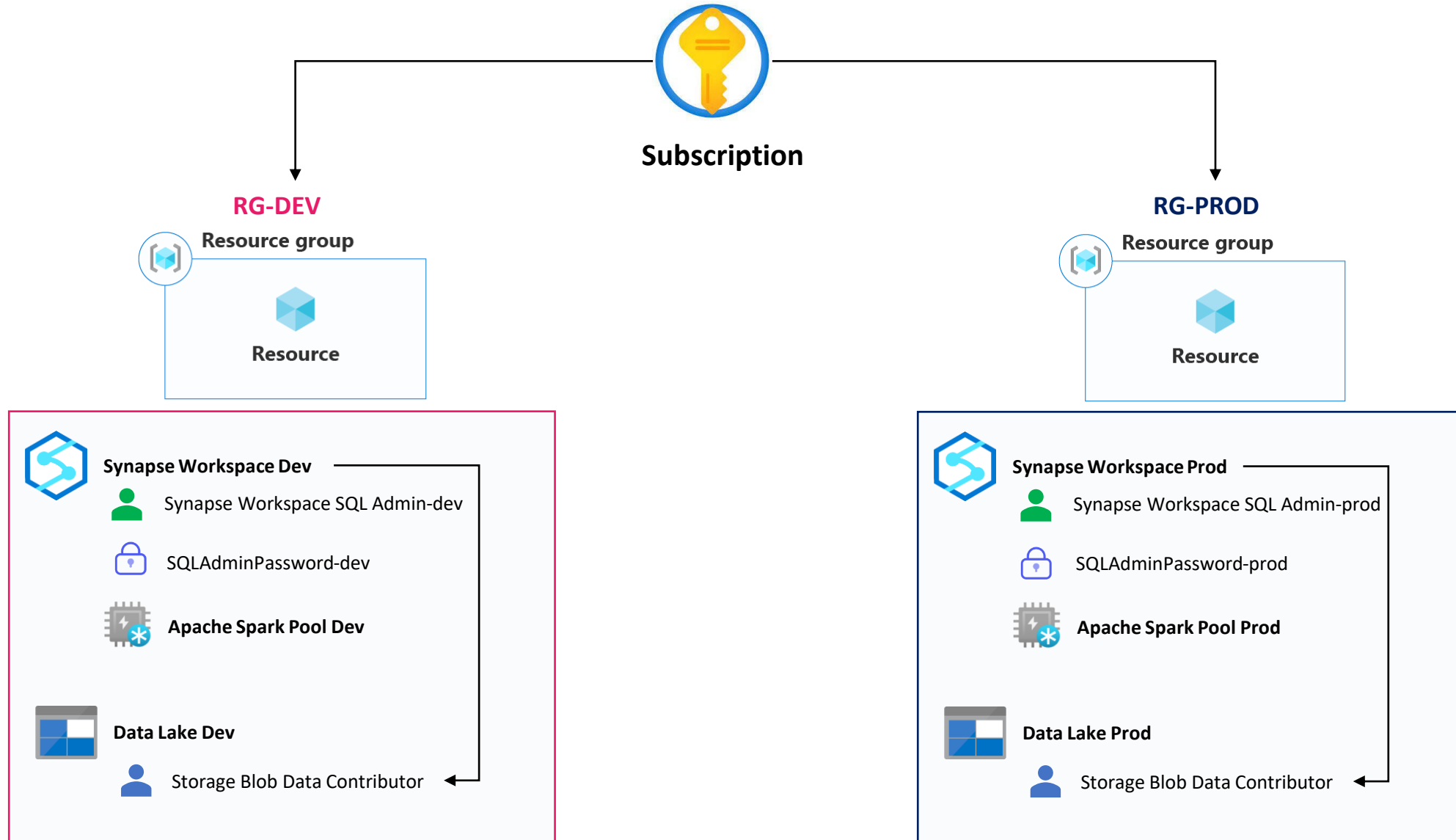


Agenda

- Why I care about IaC
- Advantages of IaC
- What is Bicep
- Demo



Why I care about IaC



Why I care about IaC



The Joys of Infrastructure as Code

Before: Complex, manual, undocumented steps

Vorgehensweise beim Anlegen von Synapse Umgebungen

Wednesday, November 22, 2023 11:04 AM

Ganz wichtig!!!!

Diese Rollen werden benötigt:

Auf Subscription ebene also RBAC : Contributor und User Access Administrator

[Assign Microsoft Entra roles in PIM | Microsoft Learn](#)

[Assign Azure resource roles in Privileged Identity Management | Microsoft Learn](#)

Auf Entra ID ebene -> User Administrator

[User Administrator - Microsoft Entra admin center](#)

[Assign Microsoft Entra roles in PIM | Microsoft Learn](#)

Es werden immer 2 Umgebungen angelegt: DEV und PROD

Jede Umgebung enthält:

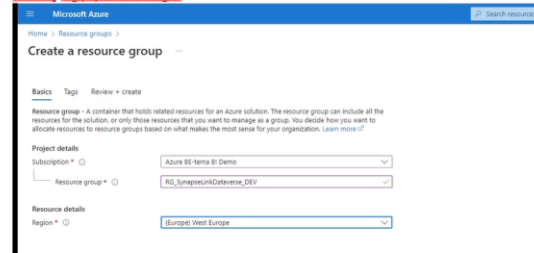
- Resource Group:
 - Datalake
 - Synapse Workspace

Voraussetzungen für Implementierung:

- RBAC Rolle "Contributor" auf die Subscription -> kann erstellt werden von einem Azure Global admin über PIM

1. Erstellung Resource Group

Beim Name am Ende _DEV angeben!
Benennung: RG_SynapseLinkDataverse_DEV



2. Storage Account anlegen

IaC: Template (Codefile)

```
40 var appNameFormatted = toLower(appName)
41 var locationFormatted = length(location) > 7 ? substring(location,0,7) : location
42 var kvnameCandidate = 'kv-${appNameFormatted}-boot-${locationFormatted}'
43 var bootstrapkvname = length(kvnameCandidate) > 24 ? substring(kvnameCandidate,0,24) : kvnameCandidate
44
45 resource resourceGroup_Shared 'Microsoft.Resources/resourceGroups@2023-07-01' existing = {
46   name: 'rg-${appNameFormatted}-shared-${locationFormatted}'
47 }
48
49
50 resource keyVault_bootstrap 'Microsoft.KeyVault/vaults@2023-07-01' existing = {
51   scope: resourceGroup_Shared
52   name: bootstrapkvname
53 }
54
55
56 resource deploymentResourceGroup 'Microsoft.Resources/resourceGroups@2023-07-01' = [ for env in environments : {
57   name: 'rg-${appName}-${env}'
58   location: location
59 }]
60
61
62
63 // module name --> identifier of module
64 // name : name of module
65 module StorageAccount 'modules/storage-account.bicep' = [ for (env,index) in environments : {
66   scope: deploymentResourceGroup[index]
67   name: 'mod-${appName}-${env}'
68   params: {
69     appName: appName
```

Advantages of IaC



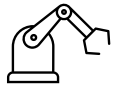
Repeatable

As often as you want, even better with parameters



Maintainable

Adapt a template to changing circumstances



Automation

The cloud does the hard work for you



Composable

Take 3 existing templates and make a new one



Cost reduction

Time is better spent elsewhere



Speed

Get more done faster



Less error-prone / Reduced risk

No human errors during deployment



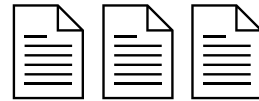
Documentation

Little need for extra documentation



Testable & verifiable

Check for errors before deploying new infrastructure



Versionable

If something doesn't work, you can go back to a working version



Accountable / Traceable

You can trace changes like on any other source code file



Scalable

Have fun doing 1000 deployments manually



Consistent

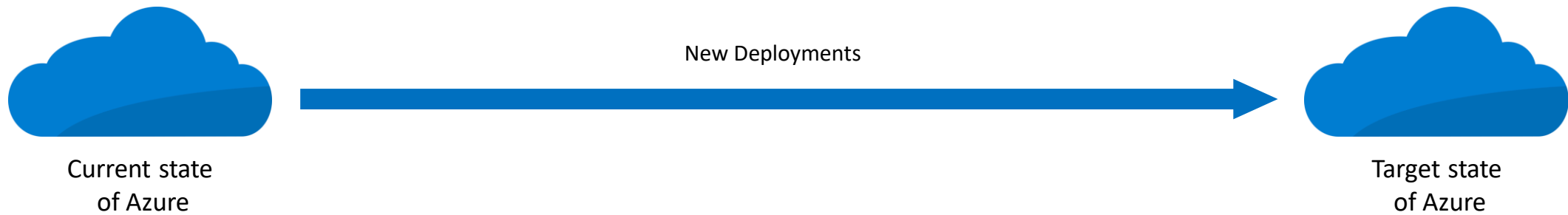
Same code, same results



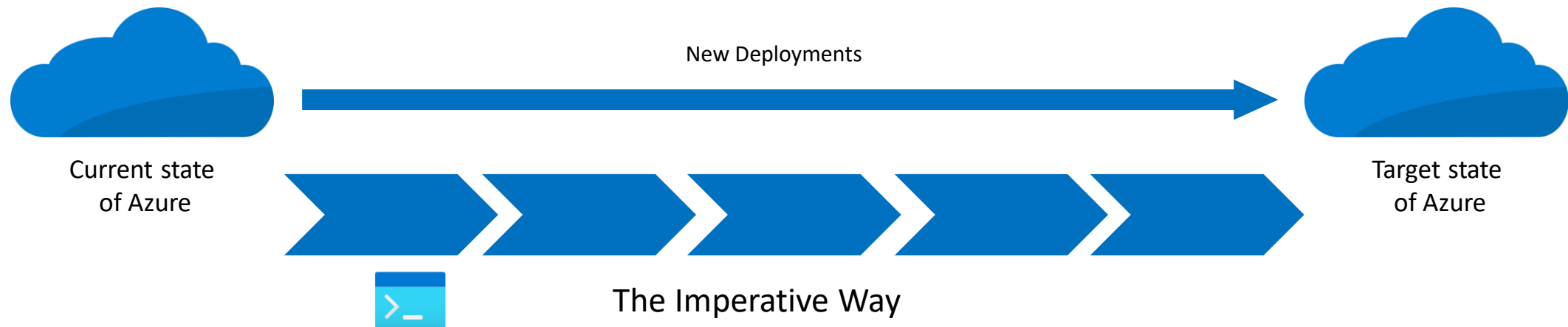
Better security

Enforced security standards & compliance, in code

Infrastructure as Code



IaC in Azure: How not to do it



Detail each step going from the current state to the target state. Baking the cake yourself with a recipe.

IaC in Azure: H



Current state
of Azure



Detail each step go

Ingredients

- ☐ 1 cup white sugar
- ☐ ½ cup butter
- ☐ 2 eggs
- ☐ 2 teaspoons vanilla extract
- ☐ 1 ½ cups all-purpose flour
- ☐ 1 ¾ teaspoons baking powder
- ☐ ½ cup milk

ADD ALL INGREDIENTS TO SHOPPING LIST

Directions

✓ Step 1

Preheat oven to 350 degrees F (175 degrees C). Grease and flour a 9x9 inch pan or line a muffin pan with paper liners.

✓ Step 2

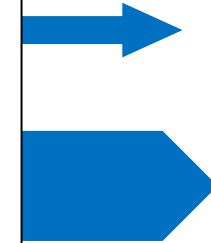
In a medium bowl, cream together the sugar and butter. Beat in the eggs, one at a time, then stir in the vanilla. Combine flour and baking powder, add to the creamed mixture and mix well. Finally stir in the milk until batter is smooth. Pour or spoon batter into the prepared pan.

✓ Step 3

Bake for 30 to 40 minutes in the preheated oven. For cupcakes, bake 20 to 25 minutes. Cake is done when it springs back to the touch.

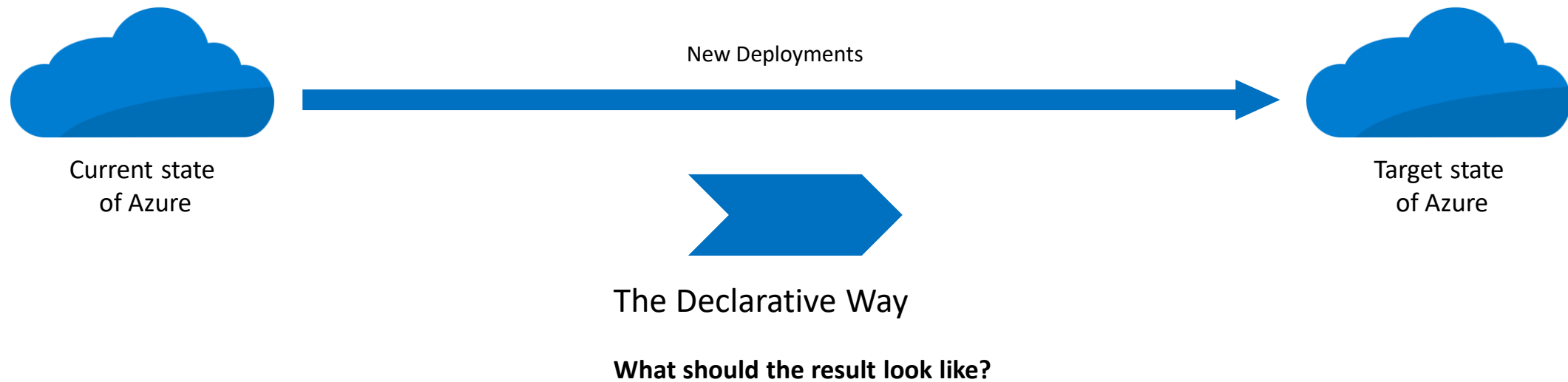


Target state
of Azure



he cake yourself with a recipe.

IaC in Azure: How to do it



IaC in Azure

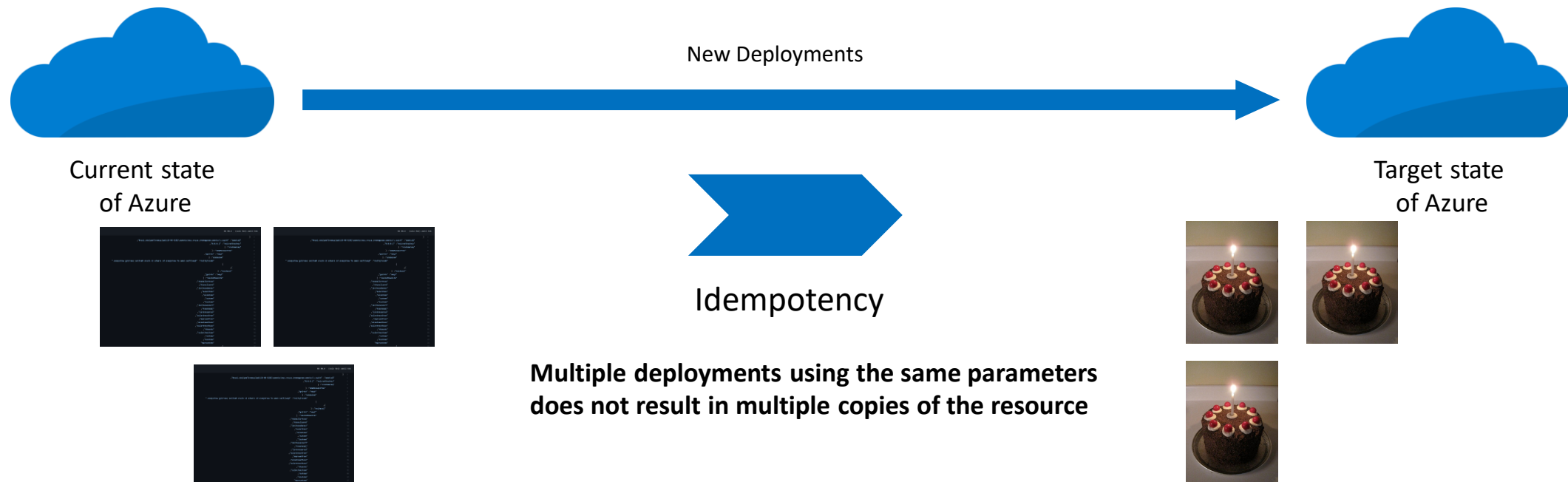


Current state
of Azure

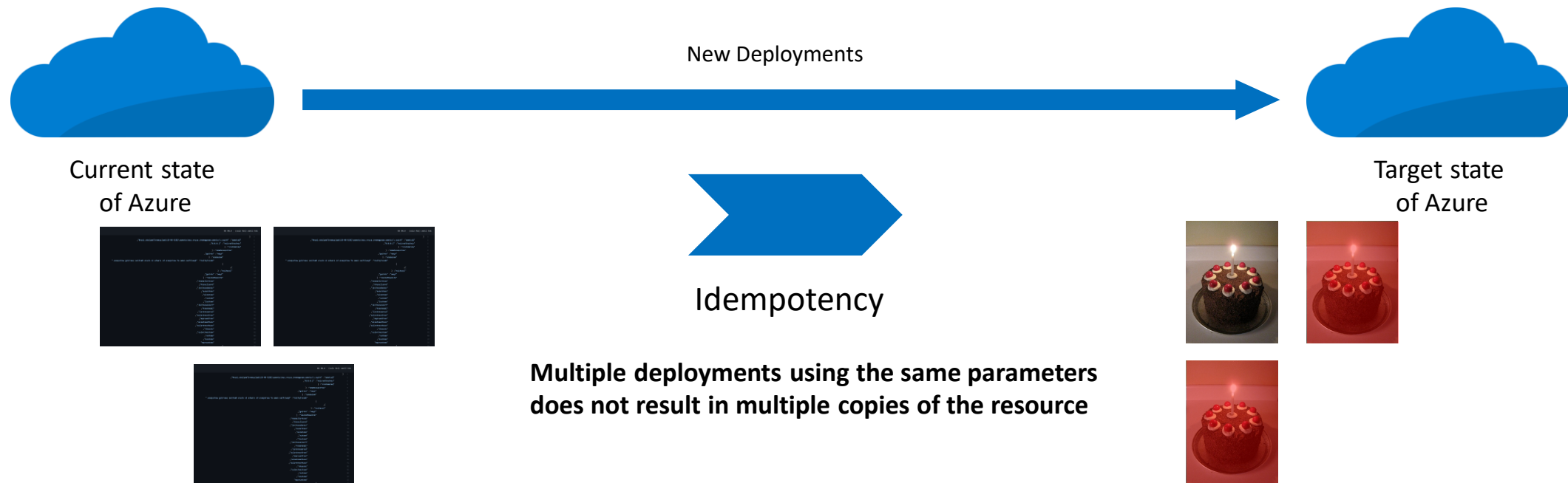


Target state
of Azure

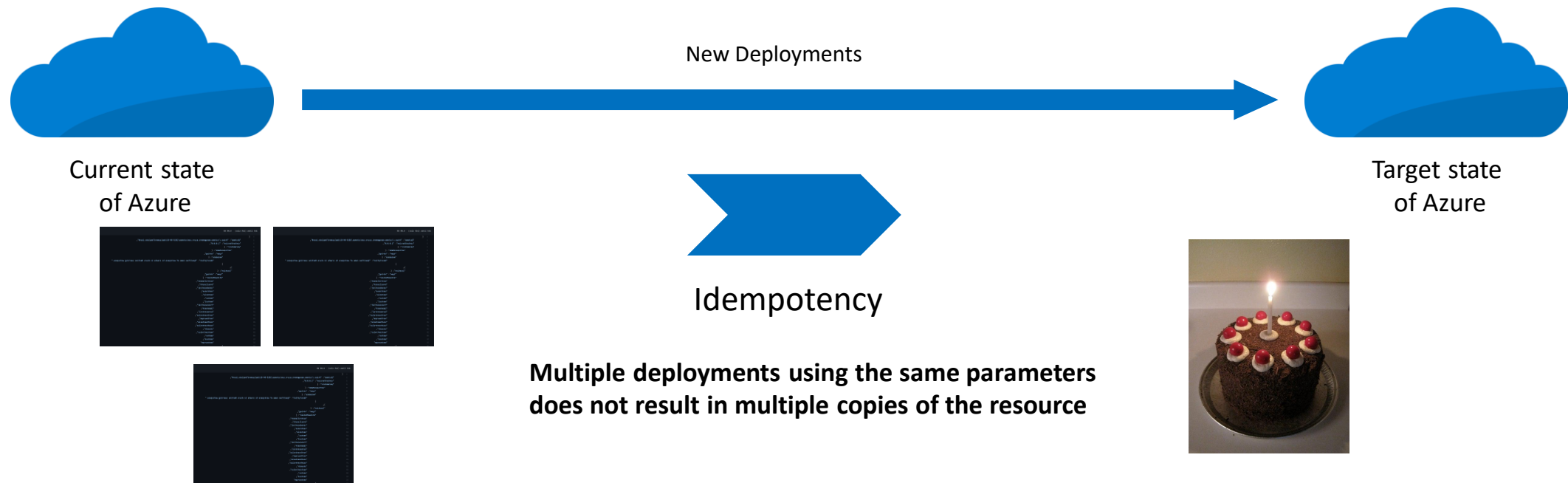
IaC in Azure: Idempotency



IaC in Azure: Idempotency



IaC in Azure: Idempotency



Azure Bicep

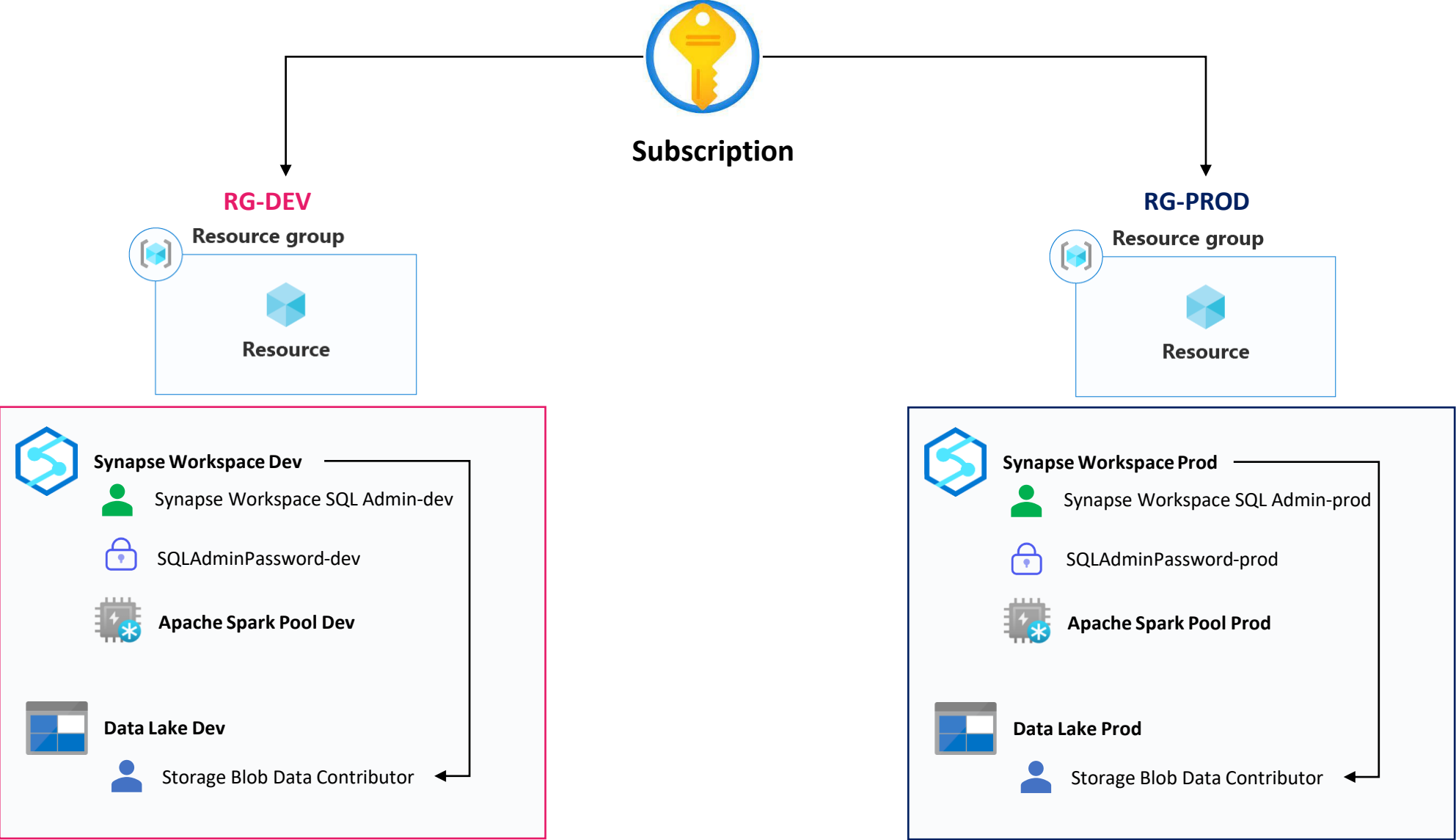


- Declarative
- Azure-native language for doing IaC
- Concise and easy to write

- Very little boilerplate
- Compile-time validation
- Integration with Azure ecosystem

Live Demo

Target



Misc

- Bicep is installed with AzCLI 2.20.0+ or PowerShell 5.6.0+
- Bicep Tutorial
- Bicep Playground
- VS Code Addon
- Full Template Reference in ARM & Bicep

Thank you for your attention. Any questions?

Feedback please!



<https://feedback.dsmuc.de/>

BACK-UP

Tools

- VS Code
- Azure CLI
- Bicep CLI
- Bicep Extension for VS Code
- Azure Subscription

Where to start with Bicep

- Microsoft Learn
 - [Learn modules for Bicep - Azure Resource Manager | Microsoft Learn](#)
 - [Azure resource reference - Bicep, ARM template & Terraform AzAPI reference | Microsoft Learn](#)
- Udemy:
 - Azure Bicep: Mastering Infrastructure as Code with Bicep
 - Master Azure Bicep: Explore Advanced Features and Techniques