Automatisierte Infrastrukturbereitstellung über ARM und Terraform

02.07.2019

Über white duck

- Spezialisiert auf Application Development, Cloud Infrastructure, DevOps und IoT
- Ganzheitliches Angebot als Cloud Solution Provider - von der Entwicklung bis zum Betrieb von Cloud-Lösungen
- Gegründet 2012 mit Sitz in Rosenheim, derzeit 18 Mitarbeiter
- Erfahrung aus mehr als 15 Jahren Softwareentwicklung
- Technologie-Fokus: Microsoft Azure Cloud, Azure DevOps, .NET C#, .NET CORE, REST, Angular, TypeScript
- Konzeption, Implementierung und Betrieb von SaaS-, Web-, Mobile- und IoT-Anwendungen

technologie

kompetenz

teamwork

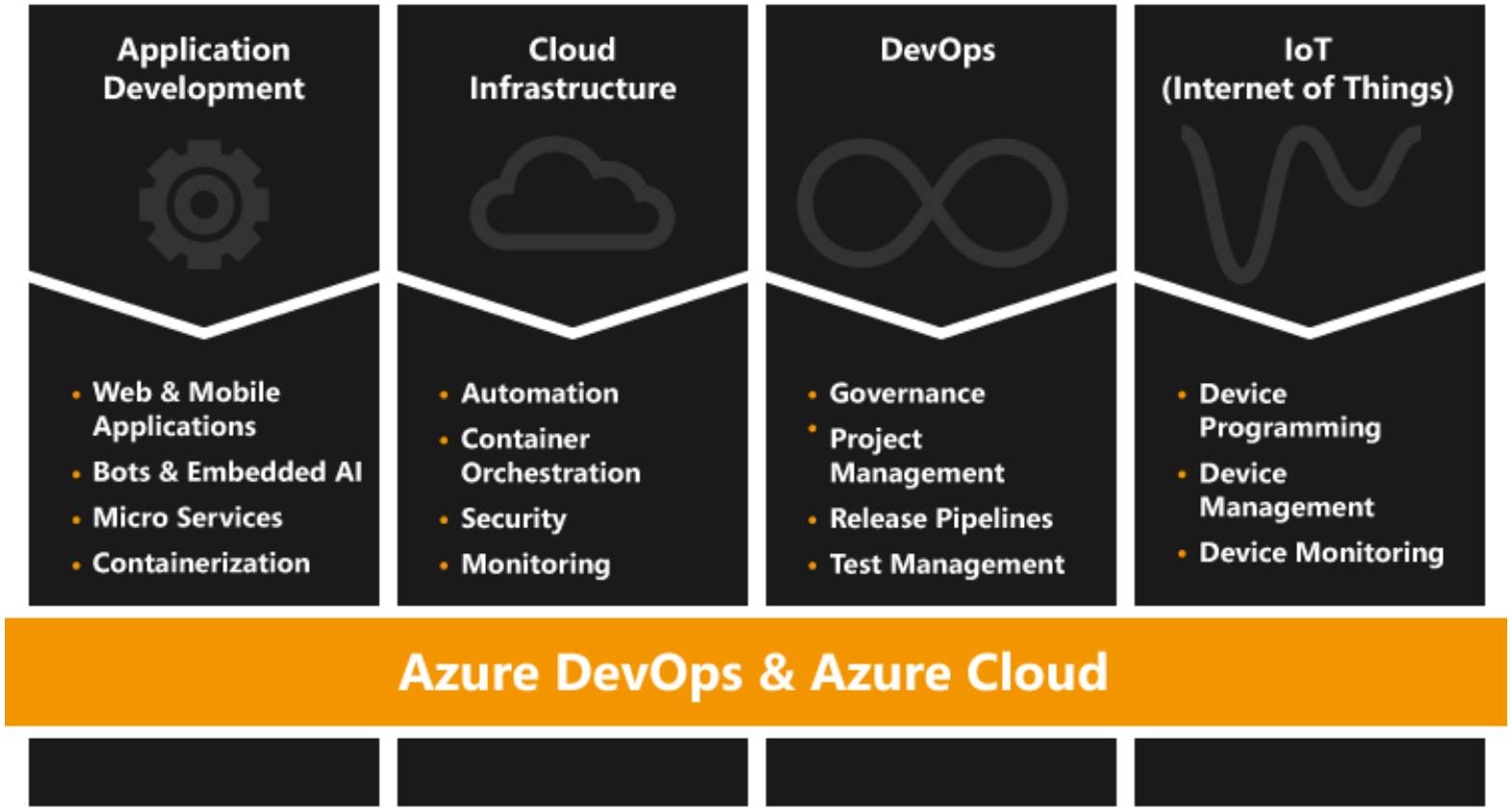
Entwicklung, Beratung und Coaching rund um die Microsoft Azure Cloud

Microsoft Partner
Microsoft
Gold DevOps
Gold Cloud Platform

CLOUD ECOSYSTEM
Mitglied

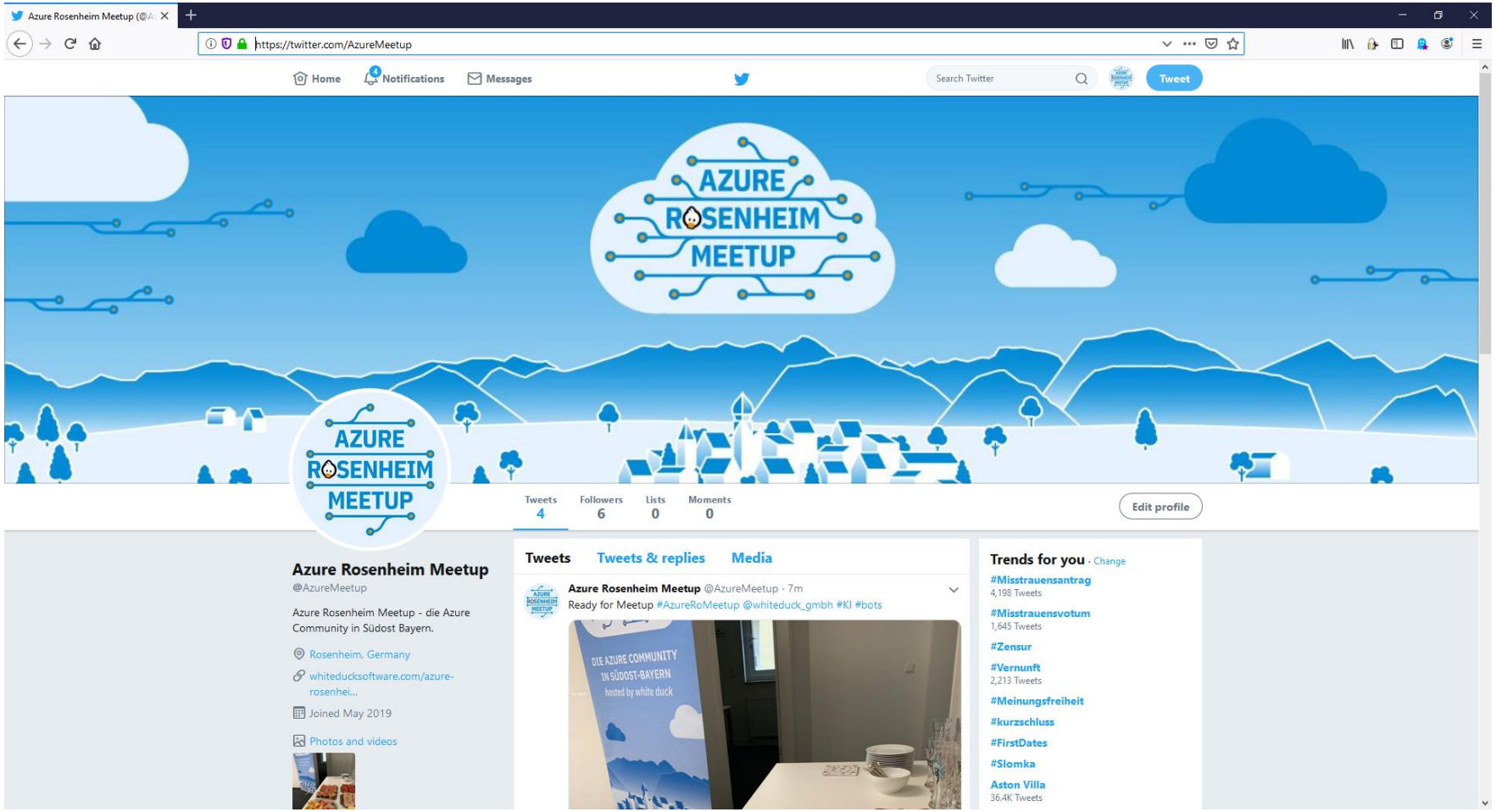
Cloud- und Software-Engineering

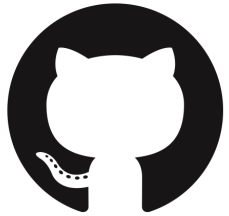
Konzeption, Entwicklung und Betrieb von individuellen Cloud Anwendungen.



Azure Rosenheim Meetup auf Twitter

@AzureMeetup





Ressourcen zum Meetup

white duck

Folien und Ressourcen zum Meetup werden auf GitHub bereitgestellt:

https://github.com/whiteducksoftware/azure-meetup-rosenheim

Agenda

ARM
Daniel Kerschagl, white duck
Übersicht ARM Deployment & Pipelines

Demo: Deployment mittels ARM Templates in Azure DevOps

Terraform
André Ratzenberger, white duck
Terraform Basics
Terraform vs ARM

Demo: Deployment mit Terraform in Azure DevOps

Ausblick: Automatisierung und Modularisierung mit Terraform in Azure DevOps
Martin Brandl, white duck

Azure Resource Manager – Definitionen

white duck

- Resource Group: Logisch gruppierte Entitäten mit gemeinsamen Lifecycle
- ARM – Template: Declarative JSON Datei die Ziel und State definiert
- Deployment: Umsetzung des ARM Deployments
- Parameters: Werte um das Deployment anzupassen

Azure Resource Manager – Definitionen

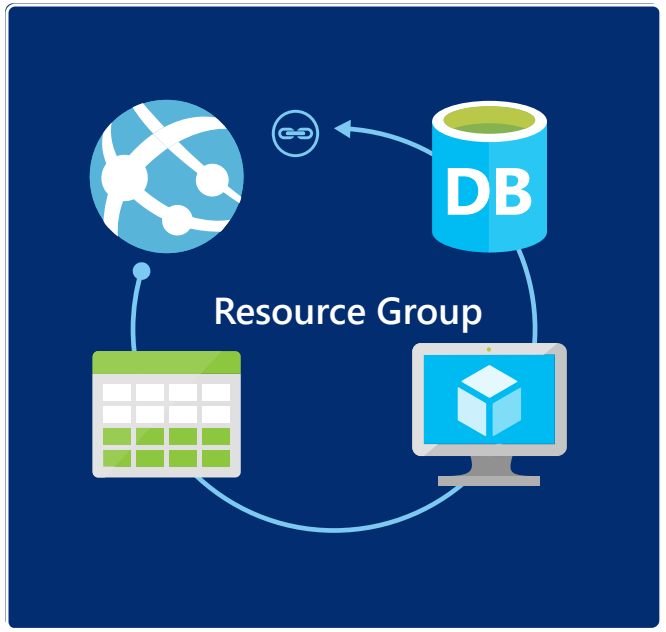
white duck

- Parameter File: JSON file for parameters and values
- API Version: Versioning
- Resource Type: Specifies the type of the resource
- Resource provider: Manages specific kinds of resources

Resource Gruppe als Management Unit

white duck

- In einer Resource Group ein Application Lifecycle – Deployment, update, delete and status
- Declarativ – “Config as Code”
- Gruppierung – Tags / Kosten
- Consistent Management Layer
- Access Control – Scope for RBAC permissions



Resource Providers

white duck

- Werden genutzt um spezifische Ressourcentypen zu deployen

- Zugewiesen über provider namespace (z.B Microsoft.Compute, Microsoft.Storage, Microsoft.Web)

- Ressourcen Typen
  - Ein Provider managet eine oder mehrere Ressourcen
  - Verschiedene regionale Verfügbarkeit und API Versionen

- Abfrage z.B über Portal oder PowerShell
  - Portal → Subscription → Resource Providers
  - Get-AzureRMLocation / "az account list-locations -o table"
  - Get-AzureRMResourceProvider / Get-AzResourceProvider -ListAvailable

Continius Delivery & Pipelines

white duck

- Build, test, configure, deploy
- Vom Commit → Build → Release
- Kurzer Releasezyklus
  - Geringeres Risiko
  - „Faster to market“
  - Higher Quality
- Any Language, any Platform, any cloud
- Extensible

white duck

Was ist ARM & Was sind ARM Templates?

- Azure ist über die Azure Resource Manager (ARM) API gemanaged
- Ressourcen sind Objekte in Azure (Databases, VM`s etc.)
- Deployments sind idempotent (Immer gleiches Ergebnis)
- Erstelle ein neues Objekt oder verändere ein bestehendes wie vorgegeben

white duck

Was ist ARM & Was sind ARM Templates?

- ARM templates definieren die Objekte (Typ, Name, Eigenschaften) mittels JSON
  - Content einer Ressourcengruppe
  - Complete/Incremental Mode

- REST API
  - Parses JSON → Parameter befallen → Führt ARM funktionalitäten aus → Ruft REST API für Ressourcen auf die erstellt warden sollen

- Deployment kombinierbar mit z.b PowerShell für weitere Funktionalität

Bestandteile eines ARM Template

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#",
  "contentVersion": "",
  "apiProfile": "",
  "parameters": {  },
  "variables": {  },
  "functions": [  ],
  "resources": [  ],
  "outputs": {  }
}
```

Element name	Required	Description
$schema	Yes	Location of the JSON schema file that describes the version of the template language. For resource group deployments, use: `https://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#` For subscription deployments, use: `https://schema.management.azure.com/schemas/2018-05-01/subscriptionDeploymentTemplate.json#`
contentVersion	Yes	Version of the template (such as 1.0.0.0). You can provide any value for this element. Use this value to document significant changes in your template. When deploying resources using the template, this value can be used to make sure that the right template is being used.
apiProfile	No	An API version that serves as a collection of API versions for resource types. Use this value to avoid having to specify API versions for each resource in the template. When you specify an API profile version and don't specify an API version for the resource type, Resource Manager uses the API version for that resource type that is defined in the profile. The API profile property is especially helpful when deploying a template to different environments, such as Azure Stack and global Azure. Use the API profile version to make sure your template automatically uses versions that are supported in both environments. For a list of the current API profile versions and the resources API versions defined in the profile, see API Profile. For more information, see Track versions using API profiles.
parameters	No	Values that are provided when deployment is executed to customize resource deployment.
variables	No	Values that are used as JSON fragments in the template to simplify template language expressions.
functions	No	User-defined functions that are available within the template.
resources	Yes	Resource types that are deployed or updated in a resource group or subscription.
outputs	No	Values that are returned after deployment.

Terraform im Überblick – Core Ideas

white duck

- Infrastructure as code
- Plattform agnostisch
- Hashicorp Configuration Language (“Human-friendly JSON”)
- Einfaches Model, welches Ressourcen als Entities mit Attributen beschreibt
- Deklarative Beschreibungssprache die Inferenzabhängikeiten unterstützt
- Parallelisierung im Deployment von nicht voneinander abhängigen Ressourcen

Terraform im Überblick – Begrifflichkeiten

white duck

- Plan
  - Überprüfen der Infrastruktur auf nötige Änderungen
  - Keine Änderungen der Infrastruktur

- Apply
  - Wendet das Plan-Ergebnis an
  - Änderungen der Infrastruktur -> Ergebnis in Statefile

- Statefile
  - Ist-Zustand der Infrastruktur
  - Wird im Azure-Kontext in einem Blob Storage abgelegt

Terraform im Überblick – Begrifflichkeiten

white duck

- Destroy
  - Löscht alle Resources
  - Resources können “geschützt” konfiguriert werden

- Modules
  - Modularisierung einzelner Resourcen oder Resourcenpakete
  - “Lego Baukasten’”

- *.tfvars-Files
  - Befüllen von definierten Variabeln
  - Abbilden der Parametrisierung verschiedener Stages

```
{
  "$schema": "
https://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "storageAccountType": {
      "type": "string",
      "defaultValue": "Standard_GRS"
    },
    "location": {
      "type": "string",
      "defaultValue": "West Europe"
    },
    "StorageAccountName":{
        "type": "string",
        "defaultValue":"storexample01"
    }
  },
  "resources": [
    {
      "type": "Microsoft.Storage/storageAccounts",
      "name": "[parameters('storageAccountName')]",
      "location": "[parameters('location')]",
      "apiVersion": "2018-07-01",
      "sku": {
        "name": "[parameters('storageAccountType')]"
      },
      "kind": "StorageV2",
      "properties": {}
    }
  ]
}
```

white duck

white duck

```
resource "azurerm_resource_group" "StorageRG01" {
  name     = "resourceGroupName"
  location = "west Europe"
}

resource "azurerm_storage_account" "testsa" {
  name                     = "storexample01"
  resource_group_name      = "${azurerm_resource_group.testrg.name}"
  location                 = "West Europe"
  account_tier             = "Standard"
  account_replication_type = "GRS"
}
```

Terraform vs ARM

white duck

- Pro
  - Leserliche Beschreibungssprache
  - Entitäten mit Attributen
  - Inferenzabhängigkeiten
  - “plan”
  - Kommentare

- Contra
  - Support für neue oder geänderte Azure Dienste
  - Statefile
  - Kein “Reverse Engineering”

Links

Terraform AzureRM Dokumentation
https://www.terraform.io/docs/providers/azurerm/

Azure DevOps Labs zu Terraform
https://www.azuredevopslabs.com/labs/vstsextend/terraform/

ARM template structure
https://docs.microsoft.com/en-nz/azure/azure-resource-manager/resource-group-authoring-templates

Azure Pipelines Dokumentation
https://docs.microsoft.com/en-us/azure/devops/pipelines/?view=azure-devops