

Praktikum CPS

Versuch 1: „Azure“

1.	Microsoft Azure: Portal and DevOps	3
1.1.	Beschreibung von benötigten Ressourcen.	4
1.2.	Deployment durch GUI.	5
2.	IaC	9
2.1.	Installation von Visual Studio Code and Terraform.	10
3.	Deployment von Container und Data Factory durch Terraform.	15

1. Microsoft Azure: Portal and DevOps

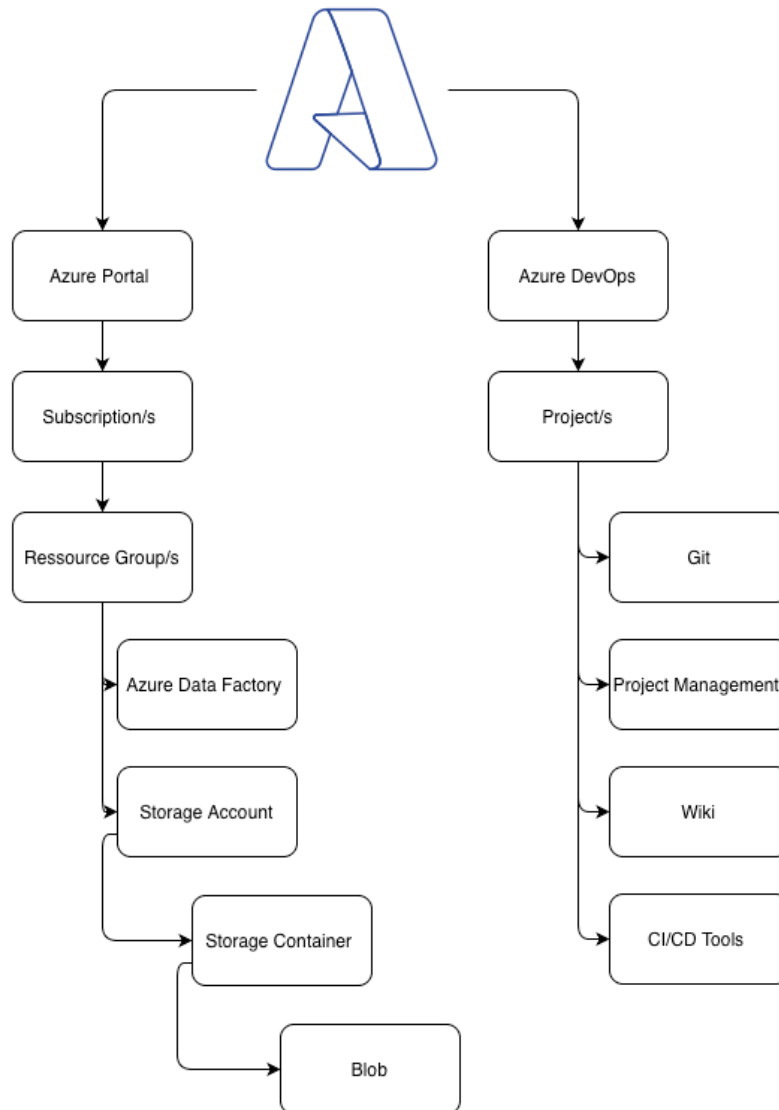


Abbildung 1 - Azure Cloud structure

Das **Azure-Portal** ist eine webbasierte, zentrale Computing-Plattform, wo alle Clouddiensten gesteuert werden können. Microsoft Azure bietet unter anderem folgende Diensten wie SQL Azure, Azure Data Factory, Databricks, Azure AI, die sich in erste Linie an Softwareentwickler richten.

Azure DevOps ist eine Software-as-a-Service (SaaS)-Plattform, die es ermöglicht, den gesamten Lebenszyklus eines Softwareentwicklungsprojekts gemäß den Prinzipien von DevOps umzusetzen. Dies wird aber im Rahmen dieser Veranstaltung nicht betrachtet.

1.1. Beschreibung von benötigten Ressourcen.

Azure Portal stellt mehr als 140 Services(weiter wird als Ressource bezeichnet) für Softwareentwickler zur Verfügung.

Im Rahmen unserer Praktikum-Veranstaltung werden wir uns mit Azure Blob Storage, Azure Data Factory und Virtual Maschine auseinandersetzen.

Ressource Group in Azure ist eine Sammlung von Ressourcen, die in logische Gruppen organisiert sind, um eine einfache oder sogar automatische Bereitstellung, Überwachung und Zugriffsteuerung sowie eine effektive Kostenverwaltung zu ermöglichen.

Blob Storage ist ein Cloud-Ressource von Azure, mit der sich große Mengen unstrukturierter Daten schnell und sicher speichern lassen. Hier ergibt sich bereits ein grundlegender Vorteil für die Entscheidung zugunsten einer cloudbasierten Infrastruktur anstelle einer On-Premises-Lösung: praktisch unbegrenzte Datenmengen

Azure Data Factory(ADF) ist ein Microsoft-Dienst, der es Entwicklern ermöglicht, mit Daten zu interagieren. Es geht grundsätzlich darum, die ETL(Extract, Transform, Load) Prozessen aufzubauen.

Virtual Maschine(VM) ist ein Computer, der vollständig auf der Cloud-Seite gehostet wird, indem der Entwickler ihre Codes ausführen können.

1.2. Deployment durch GUI.

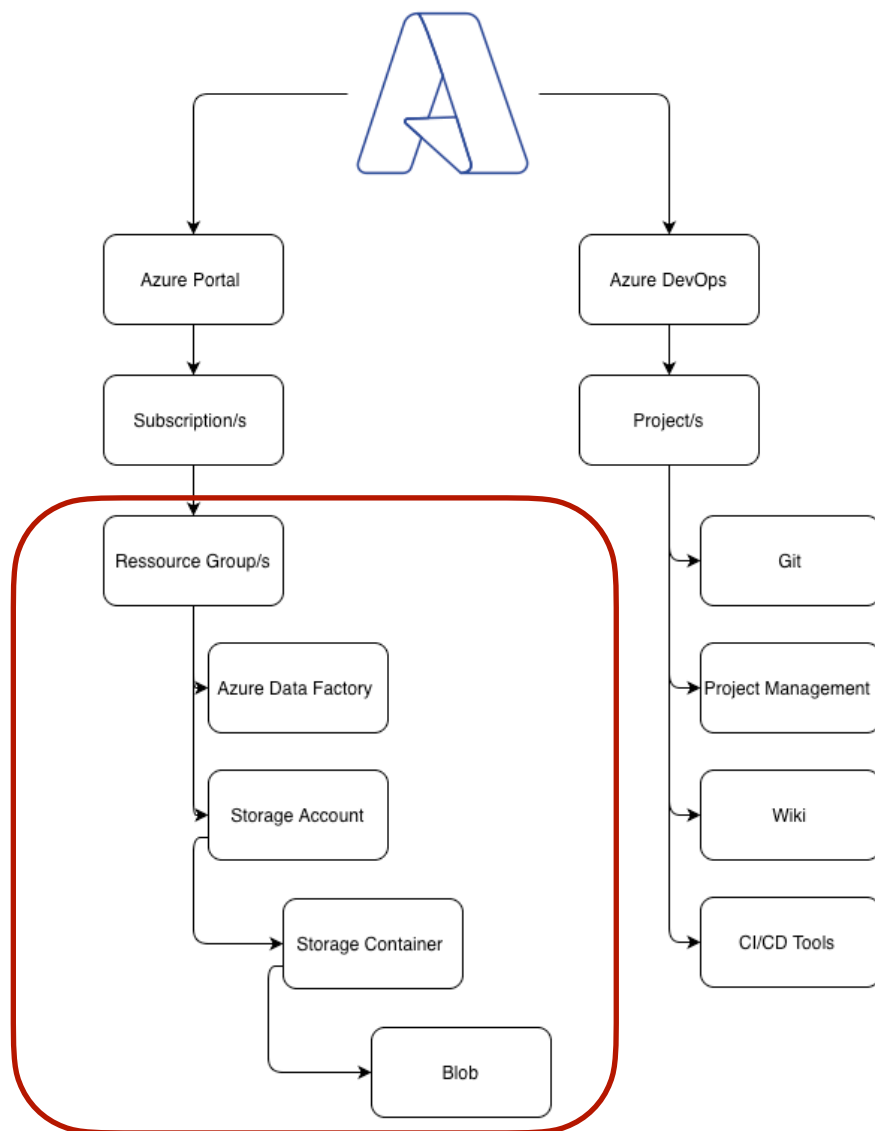


Abbildung 2 - Benötigte Azure Ressource

1. Erstellen Sie erst die **Ressource Group**.
2. Gehen Sie zu <https://portal.azure.com/>
3. Zugriff zu den Dienst kann über den folgenden Link erhalten werden. <https://www.hochschule-bochum.de/cit/hard-software/microsoft/microsoft-azure-dev-tools-for-teaching/>

4. Unter den Azure Services finden Sie „**Resource Groups**“, wählen Sie die aus und klicken Sie dann auf „**Create Resource Group**“.

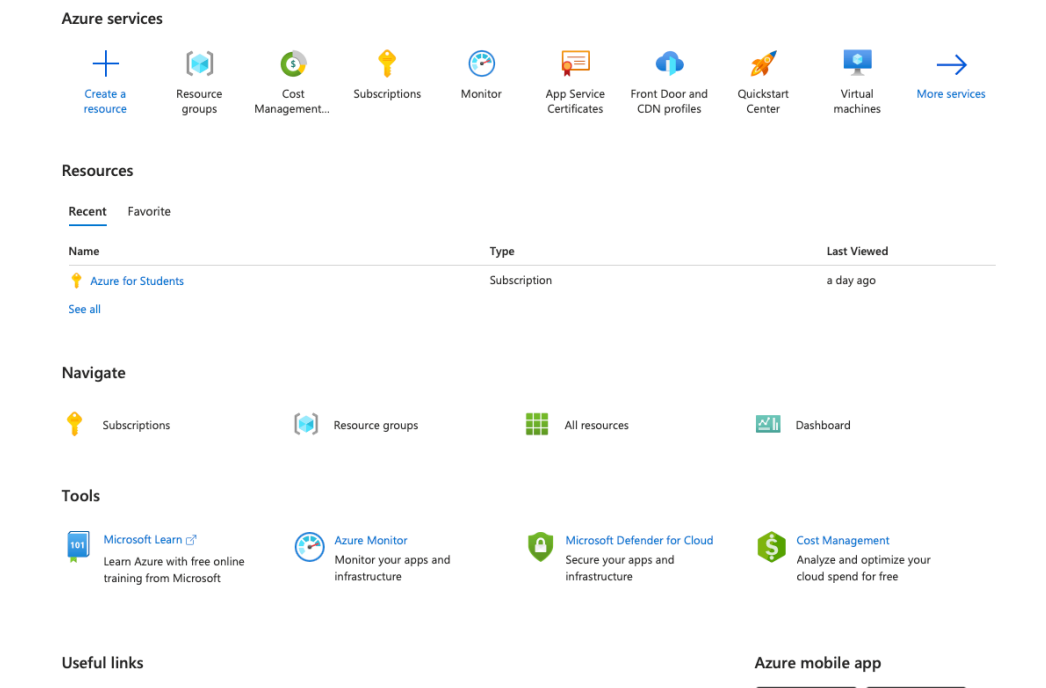


Abbildung 3 - Portal Homepage

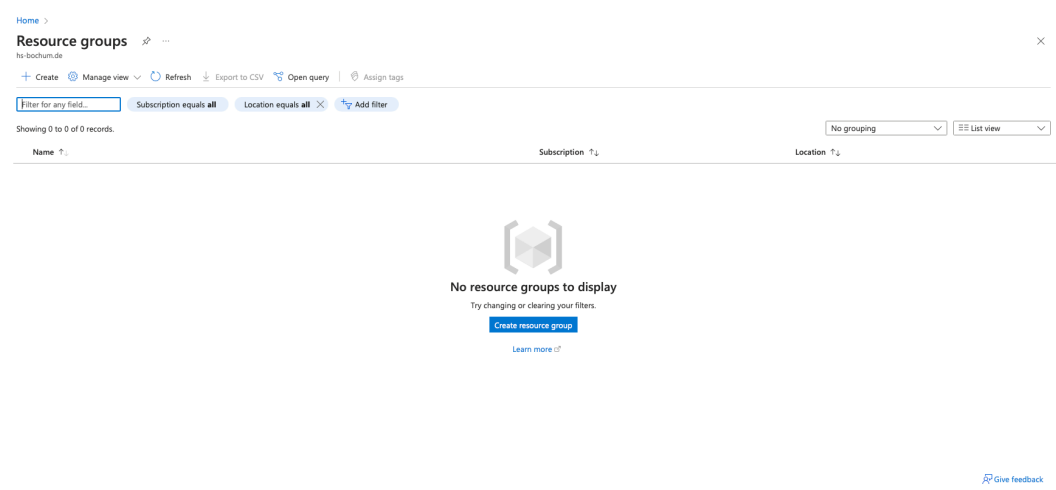


Abbildung 4 - Resource Group Erstellung

5. Wählen Sie die Subscription - ‚**Azure for Students**‘ aus, geben Sie einen Namen für die neue Ressource Group ein und wählen Sie schließlich die Region für die Erstellung - ‚**West Europa**‘ aus.
 - Resource group name: **rg_cps_praktikum** (Ihre Name).
 - (Ihre Name) - in Kleinbuchstaben und ohne Klammern, zum Beispiel **rg_cps_praktikum_mustermann**.
 - Die restliche Angaben können vorerst überspringen werden.
6. Gehen Sie auf die Homepage, um sicherzustellen, dass Ihre neu erstellte Resource Group vorhanden ist.

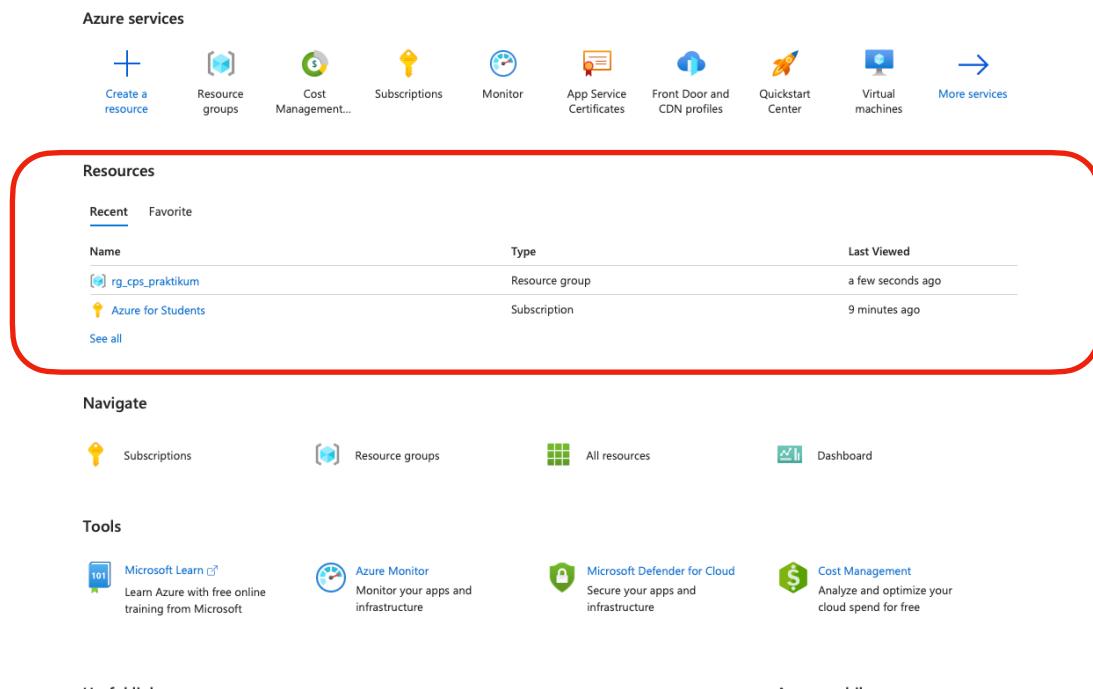


Abbildung 5 - Homepage mit Resource Group

7. Gehen Sie in die Resource Group und klicken Sie auf ‚**Create resource**‘
8. Suchen Sie nach ‚**Storage account**‘
 - Der Name sollte das Format: ‚**sacpspraktikum**(Ihre Name)‘ haben.
 - Sie können die restlichen Angaben vorerst überspringen
9. Nach erfolgreichem Account Deployment gehen Sie in der Account und klicken Sie auf ‚**+ Container**‘, um einen neuen Container zu erstellen.
 - Der Name sollte das Format: ‚**containercpspraktikum**(Ihre Name)‘ haben.
10. Gehen Sie in der Container und laden Sie **sensor.csv** - Datei von Ihrem Computer hoch.

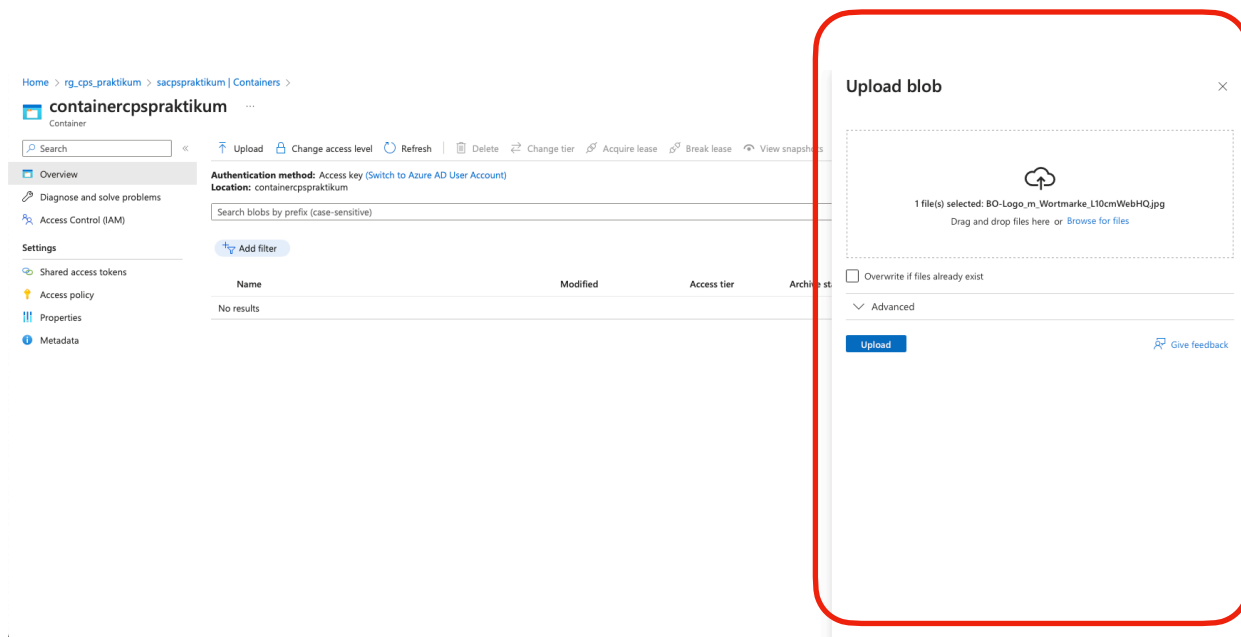


Abbildung 6 - File upload

2. IaC

Jetzt löschen Sie den Storage account und die Resource Group.

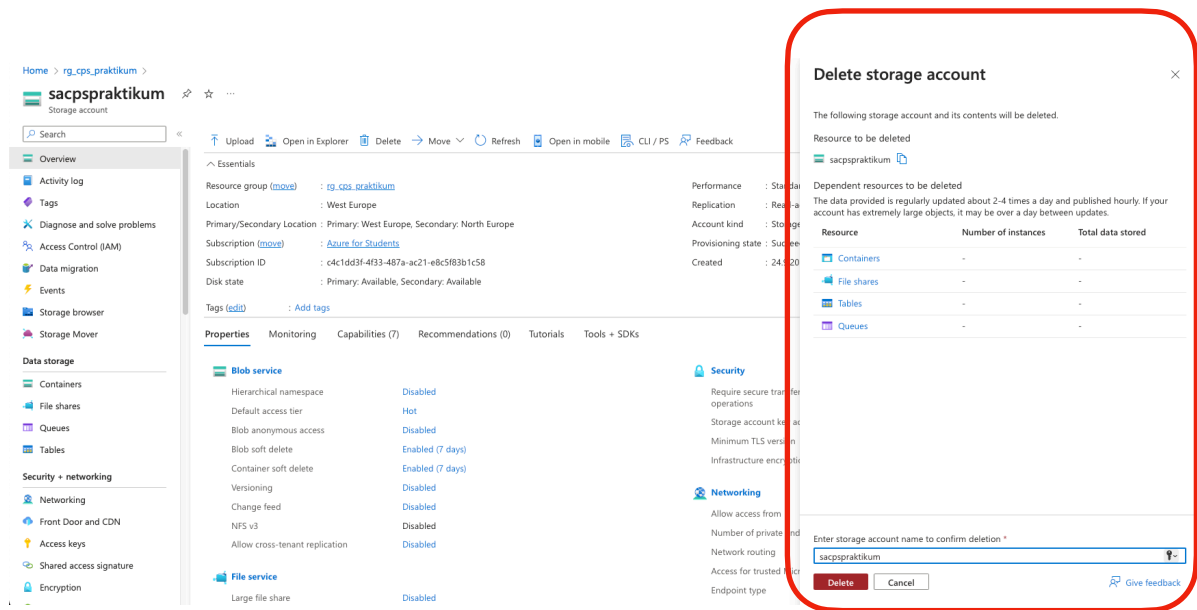


Abbildung 7 - Resource löschen

Infrastructure as Code (IaC) ist ein Konzept, bei dem die Infrastruktur eines Rechenzentrums oder einer Cloud-Plattform mithilfe von Code verwaltet wird. Anstatt manuell Ressourcen über eine grafische Benutzeroberfläche (GUI) zu erstellen und konfigurieren, werden Skripte verwendet, um die gewünschte Infrastruktur zu definieren.

Ein Beispiel hierfür ist **Terraform**.

Terraform ermöglicht es, **Infrastruktur als Code** zu behandeln, indem es eine deklarative Sprache verwendet, um Ressourcen wie virtuelle Maschinen, Datenbanken und Netzwerke zu beschreiben.

Dies bietet den Vorteil, dass Sie komplexe und skalierbare Infrastrukturen effizient und wiederholbar erstellen können. Mit Terraform können Sie leicht Hunderte oder Tausende von Ressourcen in einer einzigen Konfigurationsdatei verwalten, was in einer GUI nahezu unmöglich wäre. Dies verbessert die Konsistenz, Sicherheit und Verwaltbarkeit Ihrer Infrastruktur und ermöglicht eine effektive Skalierung.

2.1. Installation von Visual Studio Code and Terraform.

Für unsere Versuche sind folgende Software benötigt.

1. **Visual Studio Code(VSC):** <https://code.visualstudio.com/download>
2. **Azure CLI:** <https://learn.microsoft.com/en-us/cli/azure/install-azure-cli>
3. **Terraform:** <https://developer.hashicorp.com/terraform/tutorials/aws-get-started/install-cli>

Nach der Installation öffnen Sie den VSCode, wählen Sie einen Ordner als Workspace und führen Sie folgende command in Terminal(VSCode) aus:

az login

Danach werden Sie zur HS-Webseite weitergeleitet, um Ihre Identität zu überprüfen

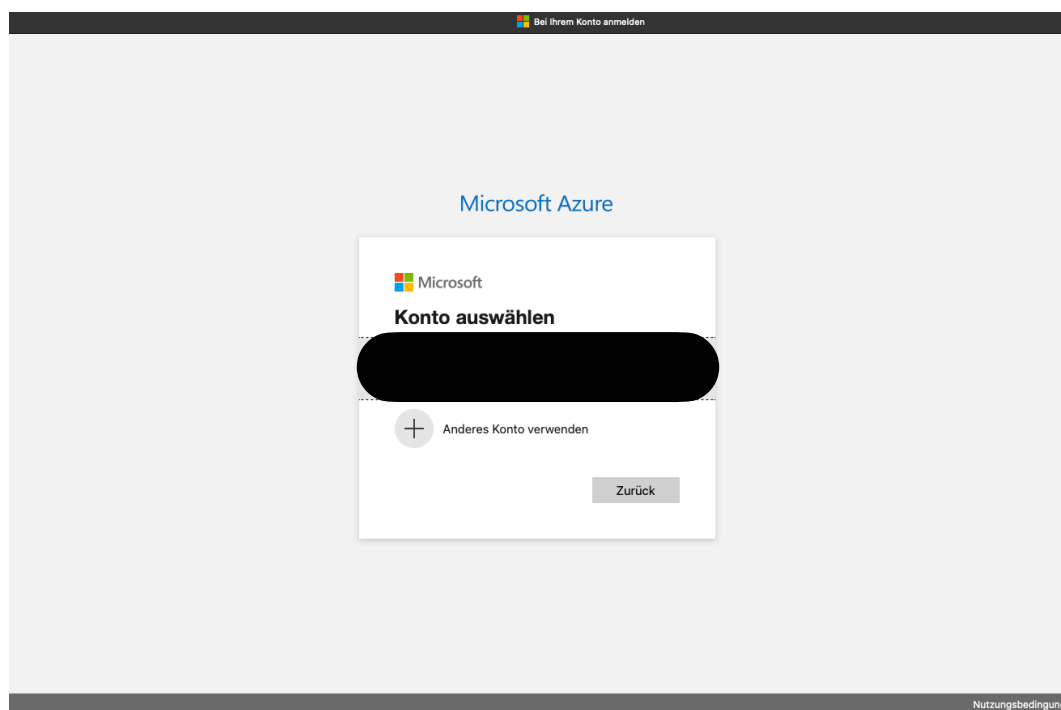
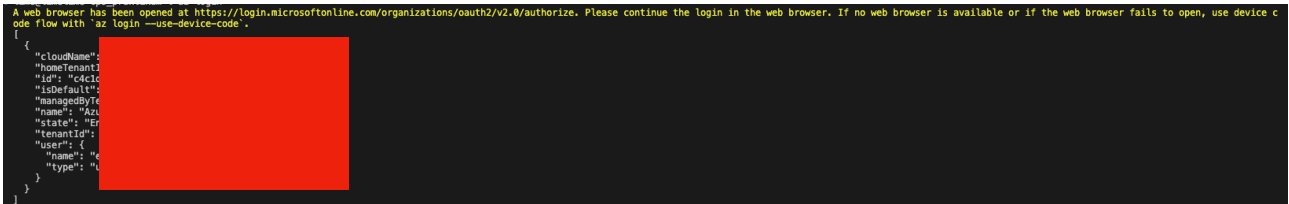


Abbildung 8 - Anmeldung mit Azure CLI

Schließlich sollten Sie die folgende Meldung im Terminal erhalten:

A terminal window showing the output of the 'az login' command. The output is a JSON object containing user information. A large red rectangle obscures the middle part of the JSON output. The visible parts of the JSON are: {"cloudName": "...", "homeTenantId": "...", "id": "c4c1c...", "isDefault": true, "managedBy": "...", "name": "Az...", "state": "Er...", "tenantId": "...", "user": {"name": "...", "type": "..."}.

```
A web browser has been opened at https://login.microsoftonline.com/organizations/oauth2/v2.0/authorize. Please continue the login in the web browser. If no web browser is available or if the web browser fails to open, use device code flow with 'az login --use-device-code'.
{
  "cloudName": "...",
  "homeTenantId": "...",
  "id": "c4c1c...",
  "isDefault": true,
  "managedBy": "...",
  "name": "Az...",
  "state": "Er...",
  "tenantId": "...",
  "user": {
    "name": "...",
    "type": "..."
  }
}
```

Abbildung 9 - Anmeldungsbestätigung mit Azure CLI

4. In Ihrem Arbeitsverzeichnis erstellen Sie eine Datei mit dem Namen **main.tf**. Das Suffix **.tf** steht für Terraform, ähnlich wie **.py** für Python steht.
5. Als Terraform eine deklarative Programmiersprache ist, werden Sie in der Datei **main.tf** die benötigten Ressourcen beschreiben. Vorlagen für die verfügbaren Ressourcen finden Sie in der offiziellen Dokumentation von Terraform.

Hier ist ein Beispiel für die Erstellung von Resource Group: https://registry.terraform.io/providers/hashicorp/azurerm/latest/docs/resources/resource_group

6. Bevor Sie mit der Erstellung von Ressourcen beginnen, müssen Sie noch in der Datei **main.tf** der Provider, die Terraform-Version und Ihre Subscription angegeben werden.

Sie können folgende Vorlage nutzen:

```

terraform {
  required_providers {
    azurerm = {
      source  = "hashicorp/azurerm"
      version = "~> 3.0.2"
    }
  }
  required_version = ">= 1.0.4"
}

provider "azurerm" {
  subscription_id = „*****-****-****-****-*****“
  features {
  }
  skip_provider_registration = true
}

resource "azurerm_resource_group" "resourcegroup" {
  name      = "rg_cps_praktikum"
  location  = "West Europe"
}

```

Die Angabe für **subscription_id** finden Sie im Portal unter Subscriptions.

7. Jetzt führen Sie im Terminal folgenden Befehl aus:

terraform init

Als Ergebnis sollte die Meldung erscheinen : “Terraform has been successfully initialized!”

8. Führen Sie anschließend in der Terminal folgenden Befehl aus:

terraform plan

An dieser Stelle ist Vorsicht geboten! Lesen Sie bitte genau, was wird hinzugefügt, geändert oder gelöscht.

Da Ihr Portal jetzt leer ist, das heißt keine Ressourcen sind in Portal vorhanden und in Script haben Sie nur ein einziges Ressource definiert, nämlich Resource Group.

Dementsprechend soll erwarten werden, dass nur eine Ressource hinzugefügt wird und nichts geändert bzw. gelöscht wird

Also: **1 to add, 0 to change, 0 to destroy**

```
lime@limetime cps_praktikum % terraform plan

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
+ create

Terraform will perform the following actions:

# azurem_resource_group.rg_cps will be created
+ resource "azurem_resource_group" "rg_cps" {
  + id           = (known after apply)
  + location     = "westeurope"
  + name         = "rg_cps_praktikum"
}

Plan: 1 to add, 0 to change, 0 to destroy.
```

Abbildung 9 - terraform plan

Sollte in der Meldung aber stehen: **0 to add, 1 to change, 1 to destroy**, bedeutet dies, dass einige Ressourcen geändert oder gelöscht werden. Beachten Sie, dass dieser Schritt irreversibel ist.

9. Jetzt führen Sie im Terminal den folgenden Befehl aus:

terraform apply

```
lime@limetime cps_praktikum % terraform apply

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
+ create

Terraform will perform the following actions:

# azurem_resource_group.rg_cps will be created
+ resource "azurem_resource_group" "rg_cps" {
  + id           = (known after apply)
  + location     = "westeurope"
  + name         = "rg_cps_praktikum"
}

Plan: 1 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?
Terraform will perform the actions described above.
Only 'yes' will be accepted to approve.

Enter a value: yes

azurem_resource_group.rg_cps: Creating...
azurem_resource_group.rg_cps: Creation complete after 0s [id=/subscriptions/c4c1dd3f-4f33-487a-ac21-e8c5f83b1c58/resourceGroups/rg_cps_praktikum]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
```

Abbildung 10 - terraform apply

Wenn alles korrekt ist, bestätigen Sie anschließend mit „yes“. Jawohl, Sie haben Ihre erste Ressource durch Terraform erstellt.

Offtop:

Während des gesamten Praktikums verwenden wir kein Source Control System wie Git. Das bedeutet, dass der sogenannte 'Single Point of Truth' auf Ihrem eigenen Computer liegt. Wenn Sie etwas in Terraform löschen oder ändern, wäre es unmöglich, alles wiederherzustellen. Daher ist es wichtig, genau darauf zu achten, was Sie ändern oder löschen.

Nach dem Ausführen des Befehls '**terraform init**' werden einige zusätzliche Dateien in Ihrem Arbeitsverzeichnis erstellt, darunter die Datei '**terraform.tfstate**'. Dies ist eine von Terraform automatisch generierte Datei.

Manuelle Änderungen in dieser Datei sind nicht erlaubt. Wenn hier irgendetwas nicht korrekt geändert wird, könnte dies zu Problemen führen und die weitere Verwendung beeinträchtigen.

3. Deployment von Container und Data Factory durch Terraform.

Jetzt erstellen Sie mithilfe von Terraform zwei **Storage Container** und eine **Data Factory**. Code-Vorlagen finden Sie in der offiziellen Dokumentation.

Am Ende sollten Sie die folgenden erstellten Ressourcen in Ihrem Portal sehen, wobei unter Account noch zwei Container befinden sollten.

Beachten Sie bitte die folgende Namenskonvention

- Storage Accountname: **sacps**
- Storage Containername: **sinkcontcps** und **sourcecontcps**
- Azure Data Factory: **adfcpsspraktikum**

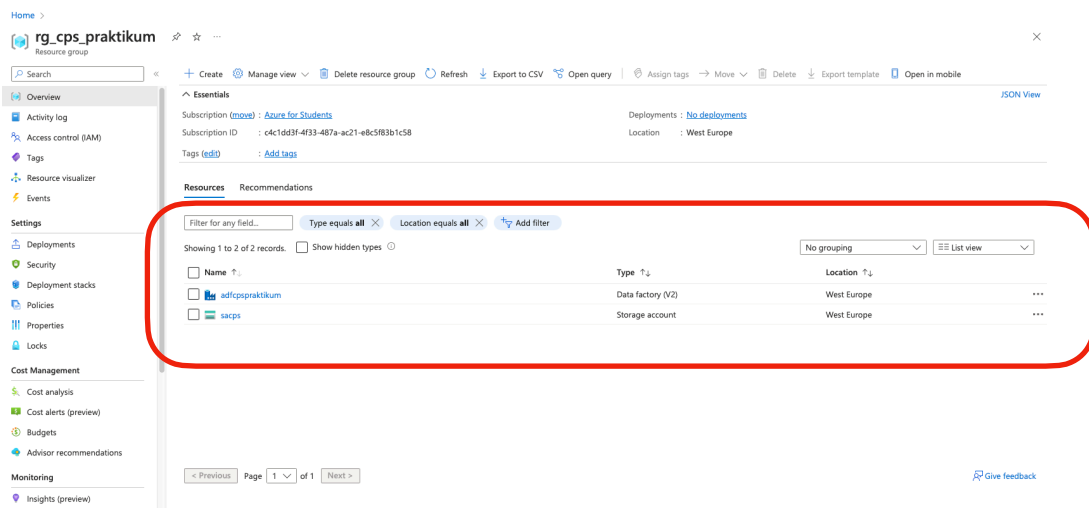


Abbildung 11 - Resource Group mit Ressourcen