**Hands On Aufgabe:**

Erstelle eine Virtual Machine mit Terraform in Azure.

Die VM soll eine Linux Maschine sein, die in West Europe zur Verfügung gestellt wird.

Die VM soll in einem Zustand sein, der es ermöglicht Monitoring Tooling auf der VM bereitzustellen.

**Hands On Task Fragen:**

1. Warum wurde die vorhanden VM Sitze gewählt?
2. Wurden nötige Ressourcen erstellt, damit die VM erreichbar ist?
3. Ist der Terraform Code parametrisiert, damit der Code in unterschiedlichen Environments wiederverwendet wurde?
4. Wie kann ich sicherstellen, dass die VM über Terraform verwaltet wird?
5. Wie muss der Code angepasst werden, damit er in einem Team Setup funktioniert?
6. Wie kann sichergestellt werden, dass Resourcen die voneinander Abhängig sind in der richtigen Reihenfolge erstellt werden?
7. Wie kann dieser Code automatisiert ausgeführt werden? Welche Terraform Commands können wann Sinn machen?
8. Welche Vor- und Nachteile hat die Verwendung von Terraform?

**DevOps Verständnis Fragen:**

1. Was bedeutet DevOps für dich?
2. Welches DevOps Tooling hast du bisher verwendet? Falls ja wie hast du damit gearbeitet?
3. Welche Tools hast du verwendet um Applikationen und Infrastruktur zu monitoren und loggen?
4. Was ist der Unterschied zwischen Monitoring, Alerting und Logging?
5. Welche Rolle spielt Kommunikation in einem Alltag eines DevOps Teams?

**Hands-On Task:**

Create a virtual machine in **Azure** using **Terraform**.  
The VM should be a **Linux machine** deployed in the **West Europe** region.  
The VM should be in a state that allows **monitoring tooling** to be set up.

**Hands-On Task Questions:**

1. Why was the selected VM size chosen?
2. Were the necessary resources created to ensure the VM is reachable?
3. Is the Terraform code parameterized so it can be reused across different environments?
4. How can I ensure that the VM is managed by Terraform?
5. How must the code be adapted to work in a team setup?
6. How can I ensure that resources with dependencies are created in the correct order?
7. How can this code be executed automatically? Which Terraform commands make sense and when?
8. What are the advantages and disadvantages of using Terraform?

**DevOps Understanding Questions:**

1. What does DevOps mean to you?
2. Which DevOps tools have you used so far? If so, how have you worked with them?
3. Which tools have you used for monitoring and logging applications and infrastructure?
4. What is the difference between monitoring, alerting, and logging?
5. What role does communication play in the daily life of a DevOps team?

## ****Hands-On Task Answers****

### ****1. Why was the selected VM size chosen?****

The selected VM size fits the purpose: small and cost-effective (e.g. Standard\_B1s or B2s) — enough for running lightweight monitoring tools like Prometheus, Node Exporter, or Telegraf.

### ****2. Were the necessary resources created to ensure the VM is reachable?****

Yes, typically this includes:

* A **Public IP address** (or a private IP with a Bastion host)
* A **Network Interface (NIC)**
* A **Network Security Group (NSG)** allowing traffic (e.g. port 22 for SSH)
* A **Virtual Network (VNet)** and **Subnet**

### ****3. Is the Terraform code parameterized so it can be reused across environments?****

Yes. By using variables (variables.tf) for:

* Region
* VM size
* Resource Group name
* Tags, etc.

…it becomes reusable for different environments like dev, test, prod.

### ****4. How can I ensure that the VM is managed by Terraform?****

* Terraform tracks all managed resources in the **terraform.tfstate** file.
* Running terraform plan shows changes to tracked infrastructure.
* If the VM is deleted or changed manually, Terraform will detect a **drift** and suggest reconciliation.

### ****5. How must the code be adapted to work in a team setup?****

* Use a **remote backend** (e.g. Azure Storage Account with blob container) for shared state.
* Enable **state locking**.
* Split code into **modules** (e.g., network, vm).
* Use **version control** (e.g., GitHub) and **CI/CD pipelines**.

### ****6. How can I ensure that dependent resources are created in the right order?****

Terraform handles this automatically through **resource references**.

* Example: a NIC references a subnet → the subnet is created first.
* Use depends\_on if explicit ordering is needed.

### ****7. How can this code be executed automatically? What Terraform commands make sense and when?****

Use CI/CD tools like GitHub Actions, Azure Pipelines, or GitLab CI. Key Terraform commands:

* terraform fmt – format code
* terraform validate – check for syntax errors
* terraform plan – preview changes
* terraform apply – apply changes
* terraform destroy – clean up

### ****8. What are the pros and cons of using Terraform?****

**Pros:**

* Infrastructure as Code (IaC)
* Version-controlled
* Modular and reusable
* Multi-cloud support

**Cons:**

* Steep learning curve
* State file management can be tricky
* Some features may lag behind official cloud provider capabilities

## 💬 ****DevOps Understanding Questions – Sample Answers****

### ****1. What does DevOps mean to you?****

DevOps is the **collaboration between development and operations**, focused on automating and accelerating software delivery while maintaining reliability and quality.

### ****2. What DevOps tools have you used so far, and how?****

Examples:

* **CI/CD**: GitHub Actions, GitLab CI – for automated deployments
* **IaC**: Terraform – to provision infrastructure
* **Containers**: Docker, Kubernetes – for deploying applications
* **Monitoring/Logging**: Prometheus, Grafana, ELK – to observe systems

### ****3. Which tools have you used for monitoring and logging?****

* **Monitoring**: Prometheus, Azure Monitor, Grafana
* **Logging**: ELK Stack (Elasticsearch, Logstash, Kibana), Azure Log Analytics
* **Alerting**: Alertmanager, Grafana Alerts, Azure Alerts

### ****4. What is the difference between monitoring, alerting, and logging?****

* **Monitoring**: Collecting metrics (CPU, memory, traffic) to understand system health.
* **Logging**: Recording events and messages for debugging and auditing.
* **Alerting**: Triggering notifications when thresholds are exceeded or failures occur.

### ****5. What role does communication play in a DevOps team’s daily work?****

Communication is key for:

* **Incident resolution**
* **Planning deployments**
* **Collaborating with developers**
* **Sharing knowledge** and reducing silos

Strong communication ensures **fast feedback**, **fewer misunderstandings**, and **more stable systems**.