



presented by

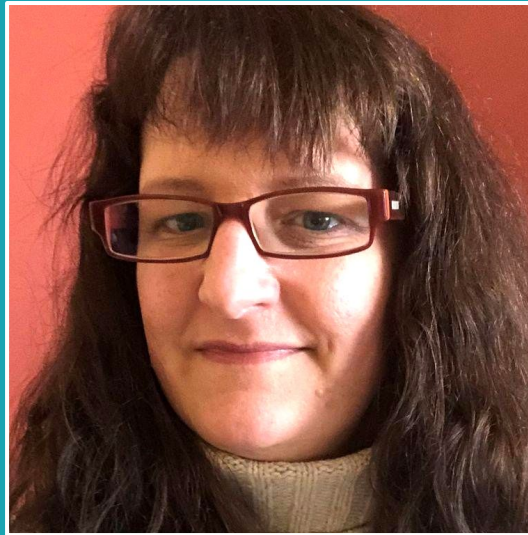


*WIR SEHEN  
UNS ZUM...*



**Nora Schöner**

 @wolkencode



**Sandra Gerberding**

 @stgerberding

## Hands-On Workshop

17. März • 9-17 Uhr

**Mit Terraform in drei  
Schritten zur lauffähigen  
Spring-Boot-Webanwendung  
in der AWS Cloud**

*MIT*



presented by



# 01

# Terraform Basics

# Inhalte

---



Intro-Story: Warum "as Code" ?

Was ist Infrastructure as Code?

Klicki-Bunti muss nicht sein

Was ist Terraform?

Terraform Code: Komponenten & Aufbau

Die Terraform CLI & States

# "Bring deine Applikation in die Cloud", haben sie gesagt...

Nur ein Klick in der Konsole

Wir brauchen bis  
Ende der Woche noch  
eine Test-Umgebung.

Es kommt  
eine neues  
Feature rein...

Ja, lass es mich nur  
eben in der Konsole  
zusammenklicken.

Foto © CodeCamp:N



# "Bring deine Applikation in die Cloud", haben sie gesagt...

Nur ein Klick in der Konsole



Foto © CodeCamp:N

# Klicki-Bunti muss nicht sein!

Schwierigkeiten



Illustration © undraw.io

Manuelles Eingreifen

Fehlende Transparenz

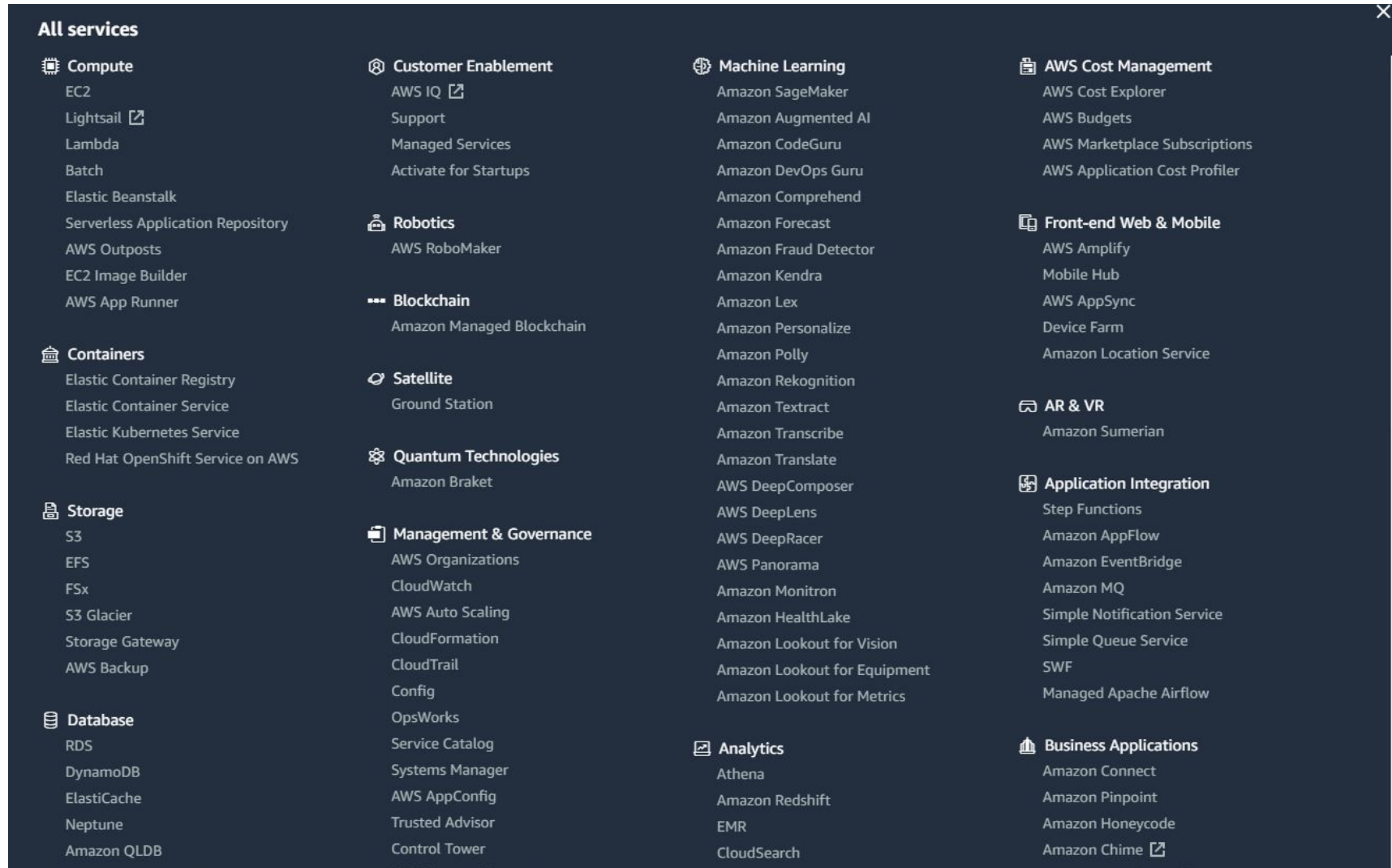
Nicht automatisierbar

Hoher Wartungs- und Kostenaufwand



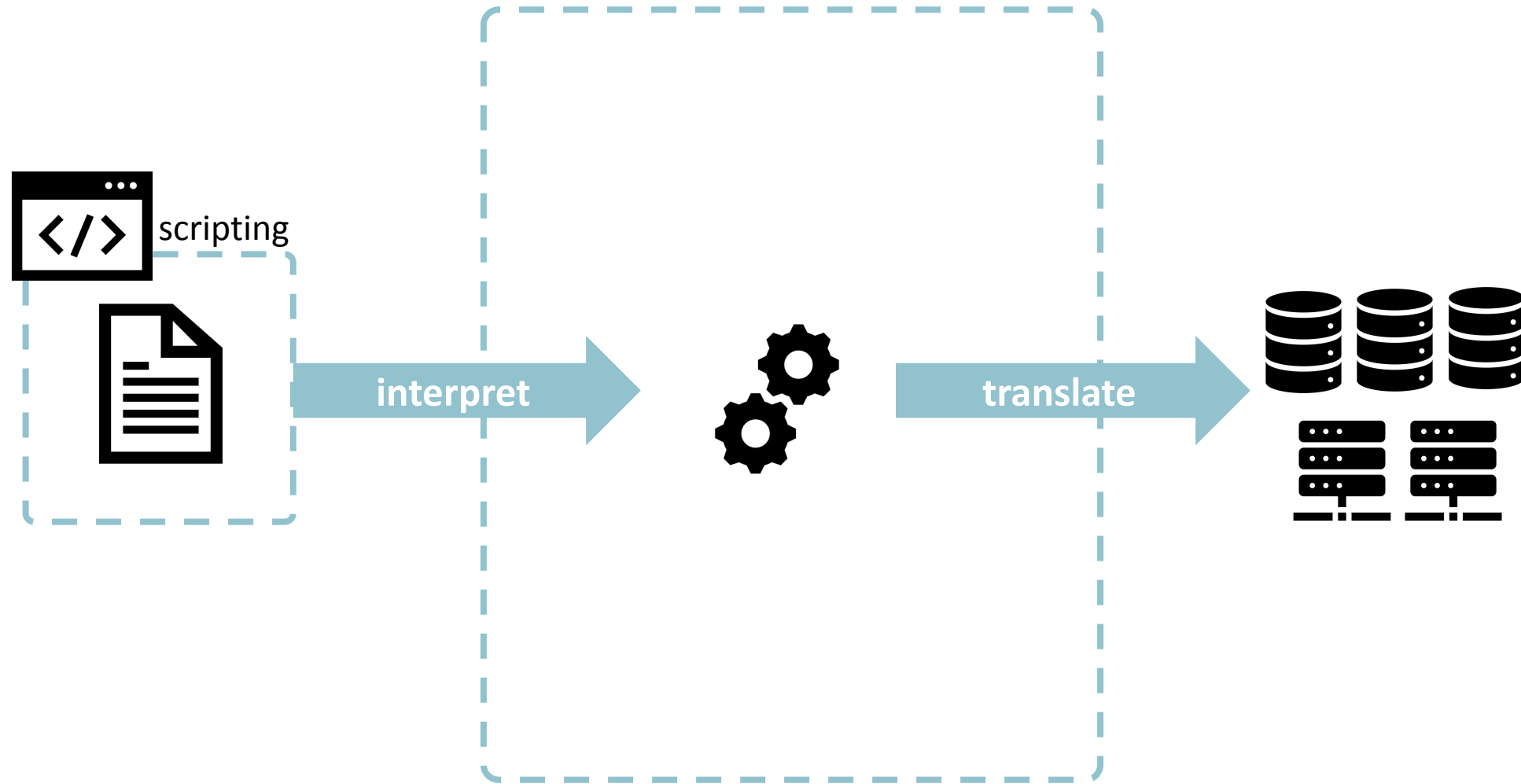
# Klicki-Bunti muss nicht sein!

## AWS Service Landschaft



# Infrastructure as Code, dein Freund und Helfer

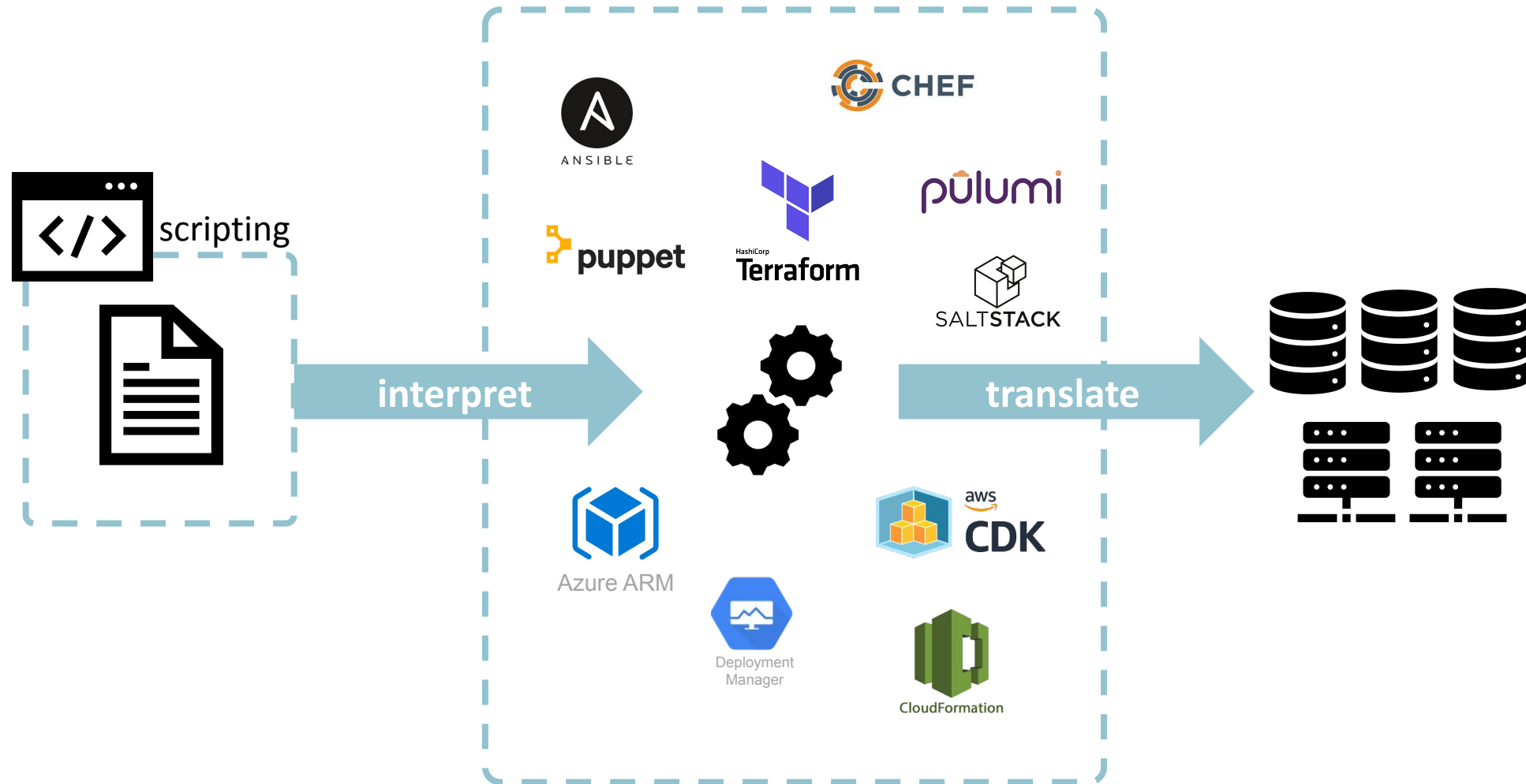
Was ist Terraform?





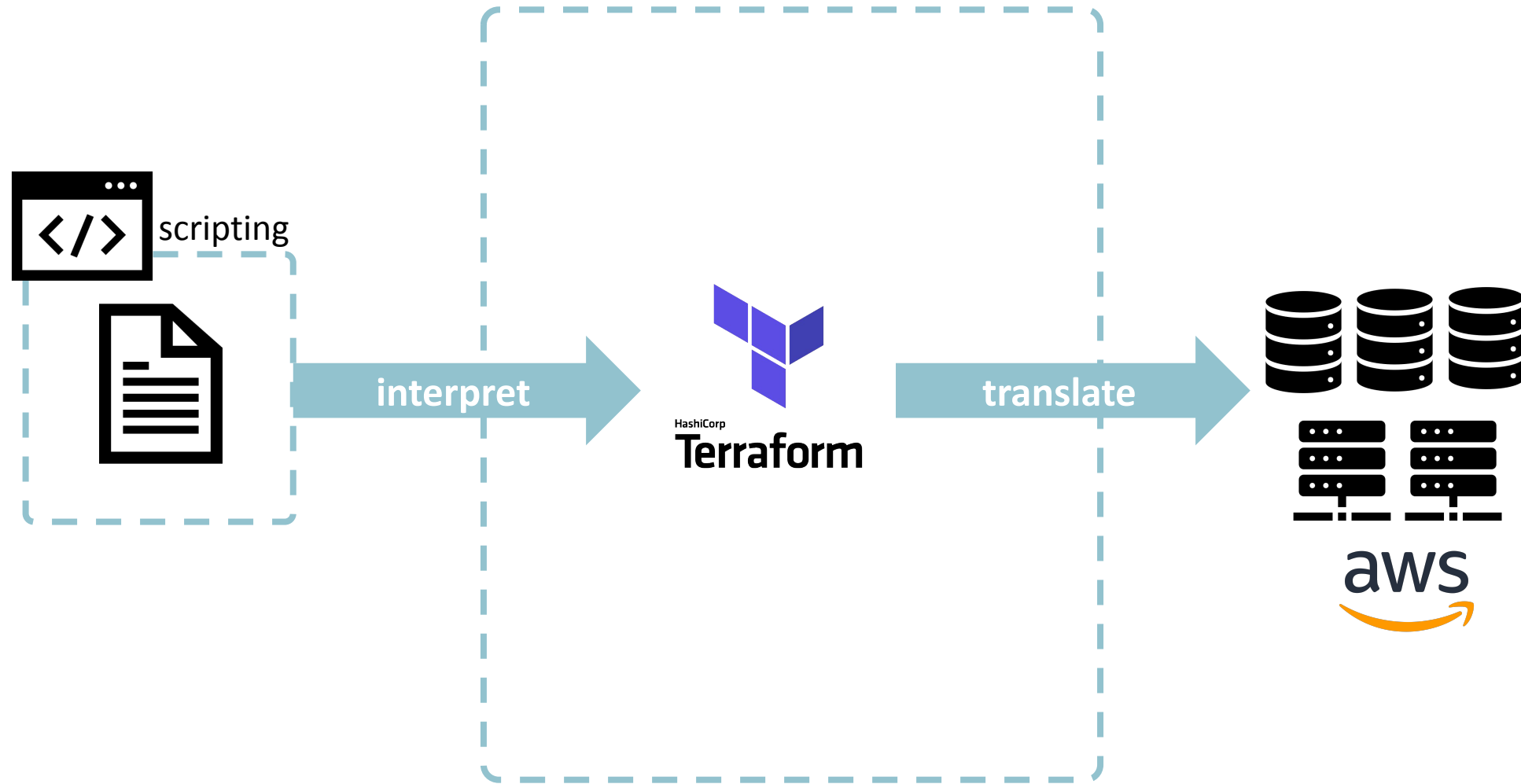
# Infrastructure as Code, dein Freund und Helfer

Was ist Terraform?



# Infrastructure as Code, dein Freund und Helfer

Was ist Terraform?



# Infrastructure as Code, dein Freund und Helfer

Welche Herausforderungen werden gelöst?



Illustration © undraw.io



Code once, use multiple times



Transparenz



Automatisierbar



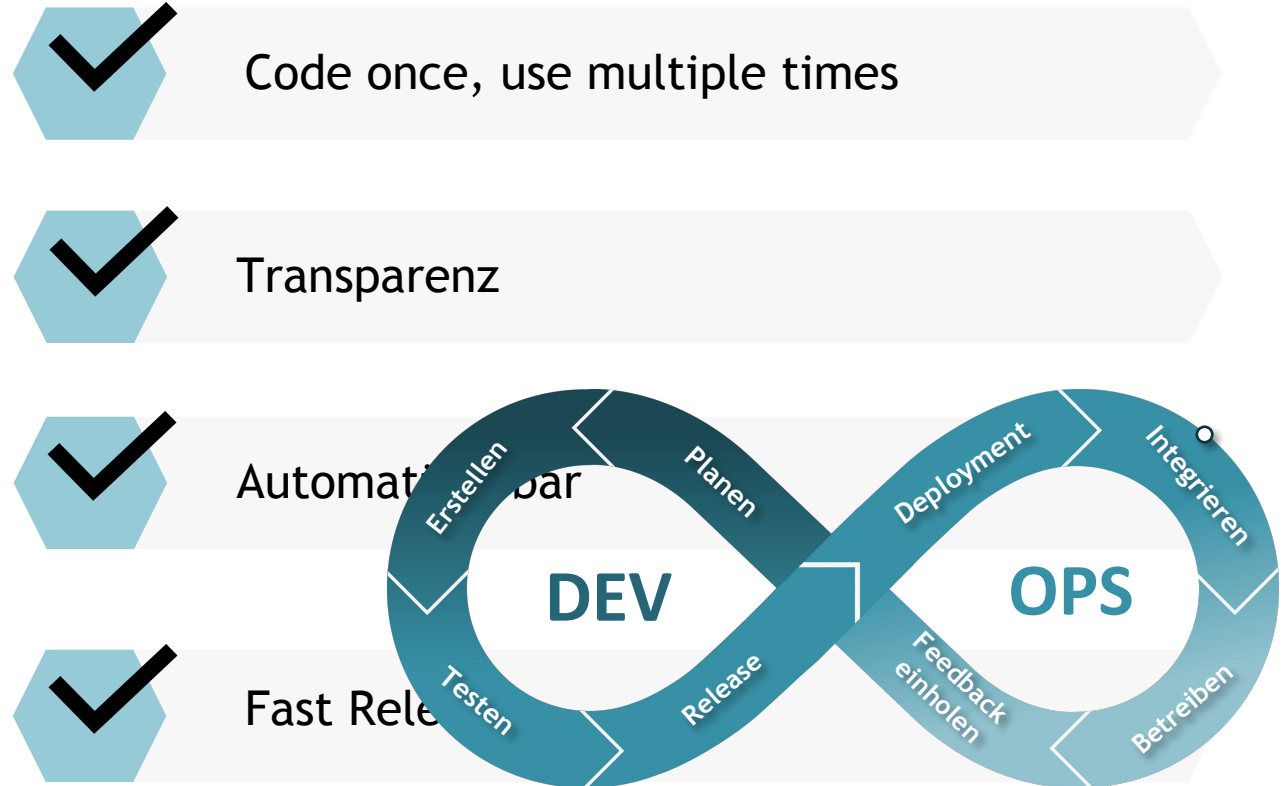
Fast Release

# Infrastructure as Code, dein Freund und Helfer

Welche Herausforderungen werden gelöst?



Illustration © undraw.io

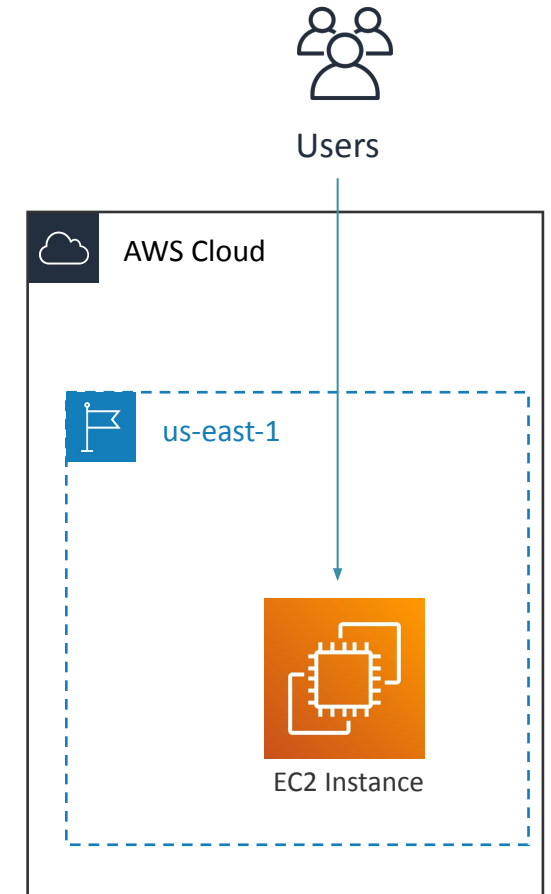


# Terraform, dein Freund und Helfer

Ein Code-Beispiel

main.tf

```
1  # set provider
2  provider "aws" {
3    region = "us-east-1"
4    profile = "profile-name"
5  }
6
7  # server/ EC2 Instance
8  resource "aws_instance" "server" {
9    ami = data.aws_ami.amazon_linux_2_arm64.image_id
10    instance_type = "t4g.micro"
11    associate_public_ip_address = true
12  }
```



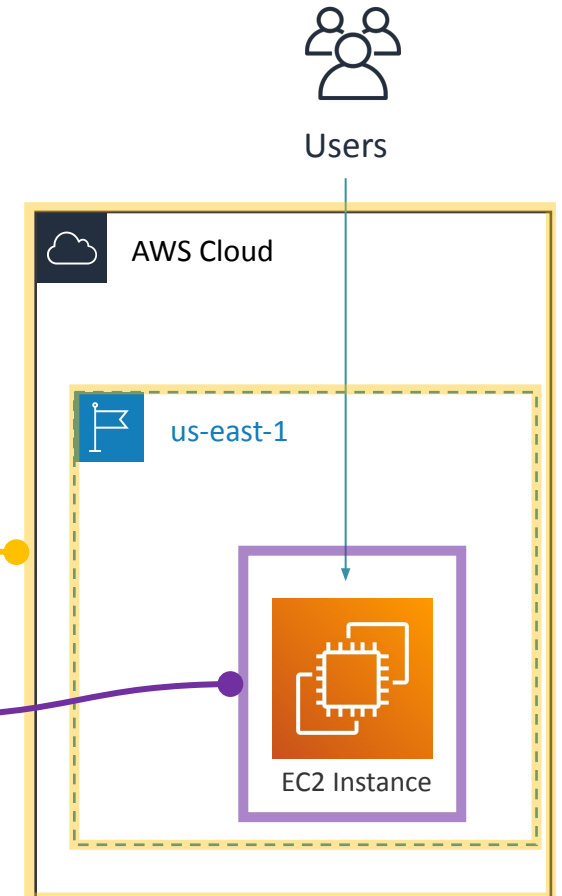
<https://wolkencode.de>

# Terraform, dein Freund und Helfer

Ein Code-Beispiel

main.tf

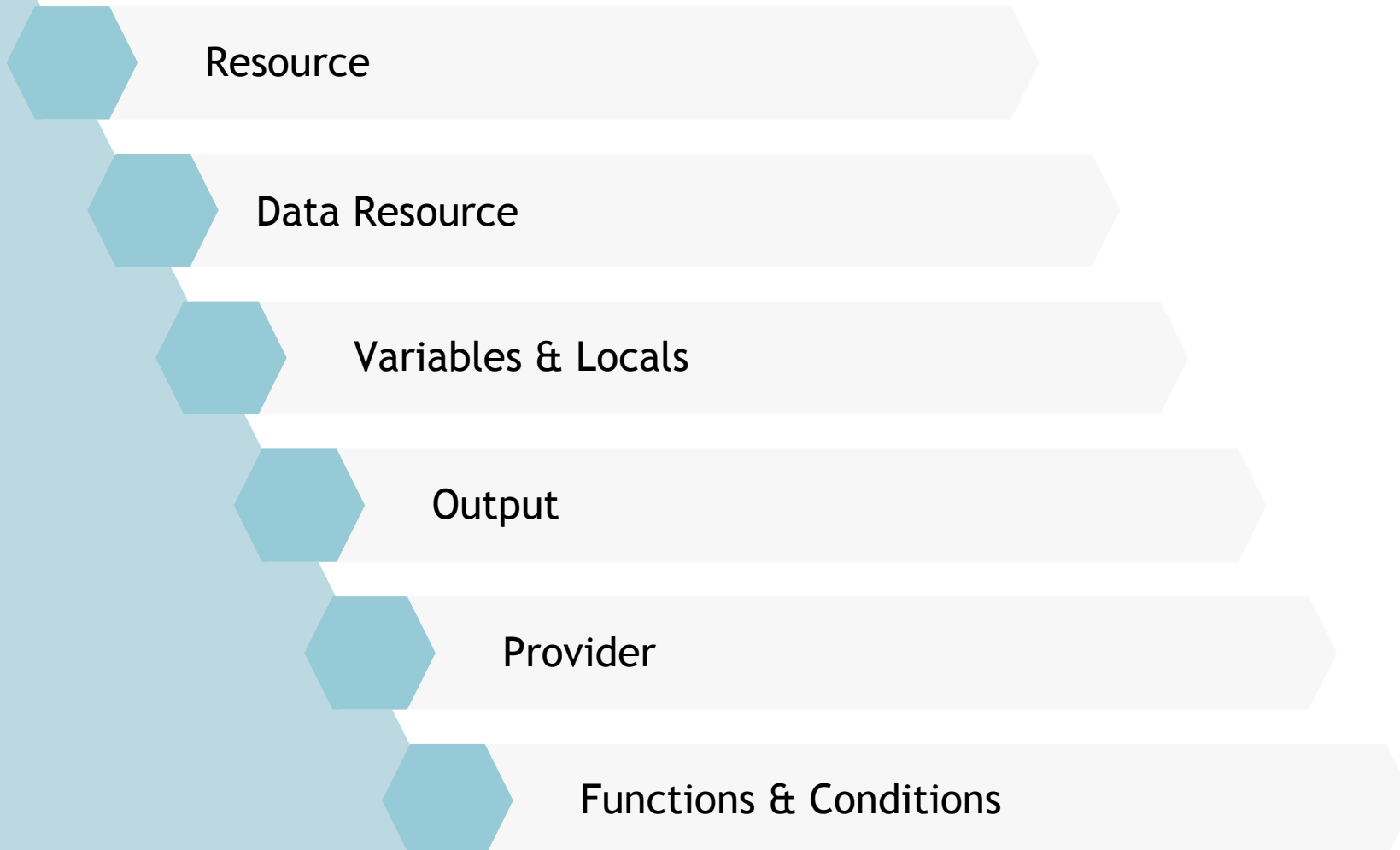
```
1  # set provider
2  provider "aws" {
3    region = "us-east-1"
4    profile = "profile-name"
5  }
6
7  # server/ EC2 Instance
8  resource "aws_instance" "server" {
9    ami = data.aws_ami.amazon_linux_2_arm64.image_id
10    instance_type = "t4g.micro"
11    associate_public_ip_address = true
12  }
```





# Terraform Komponenten

---



Resource

Data Resource

Variables & Locals

Output

Provider

Functions & Conditions

# Terraform Komponenten

## Resources

```
resource "aws_instance" "server" {  
  ami = data.aws_ami.amazon_linux_2_arm64.image_id  
  instance_type = "t4g.micro"  
  associate_public_ip_address = true  
}
```

block type "resource"

provider's resource id

unique resource name

resource's attributes

identifier = value

# Terraform Komponenten

Referenz auf eine Resource und ihre Attribute

provider's resource id . unique resource name . resources' attribute identifier



aws\_instance . server . public\_ip

# Terraform Komponenten

Referenz auf eine Resource und ihre Attribute

provider's resource id . unique resource name . resources' attribute identifier



aws\_instance . server . public\_ip

Achtung!  
Abfragbare Attribute können von  
verwendeten Attributen abhängen

# Terraform Komponenten

## Data Resources

```
data "aws_ami" "amazon_linux_2_arm64" {  
  most_recent = true  
  owners      = ["amazon"]  
  
  filter {  
    name     = "name"  
    values   = ["amzn2-ami-hvm*"]  
  }  
  
  filter {  
    name     = "architecture"  
    values   = ["arm64"]  
  }  
}
```

block type "data"

provider's resource id

unique data resource name

data resource's attributes

identifier = value

# Terraform Komponenten

## Data Resources

```
data "aws_ami" "amazon_linux_2_arm64" {  
  most_recent = true  
  owners      = ["amazon"]  
  
  filter {  
    name     = "name"  
    values   = ["amzn2-ami-hvm*"]  
  }  
  
  filter {  
    name     = "architecture"  
    values   = ["arm64"]  
  }  
}
```

block type "data"

provider's resource id

unique data resource name

data resources' attributes

identifier = value

Attribute, die die Ressource  
eindeutig identifizieren.



# Terraform Komponenten

Referenz auf eine Data Resource

block type "data" . provider's data resource id . unique data resource name . attribute identifier



data . aws\_ami . amazon\_linux\_2\_arm64 . image\_id

# Terraform Komponenten

Referenz auf eine Data Resource

block type "data" . provider's data resource id . unique data resource name . attribute identifier



data . aws\_ami . amazon\_linux\_2\_arm64 . image\_id

Wird von Terraform  
automatisch ausgelesen.

# Terraform Komponenten

## Variablen

```
variable "stage" {  
  type = string  
  description = "the name of the environment aka stage the resource"  
}
```

block type "variable"

unique variable name

variable's attributes

- type
- description
- default value
- validation
- sensitive

# Terraform Komponenten

## Variablen

```
variable "stage" {  
  type = string  
  description = "the name of the environment aka stage the resource"}
```



*variables.tfvars*

```
region = "us-east-2"  
project = "herbstcampus-workshop"  
stage = "dev"
```

block type "variable"

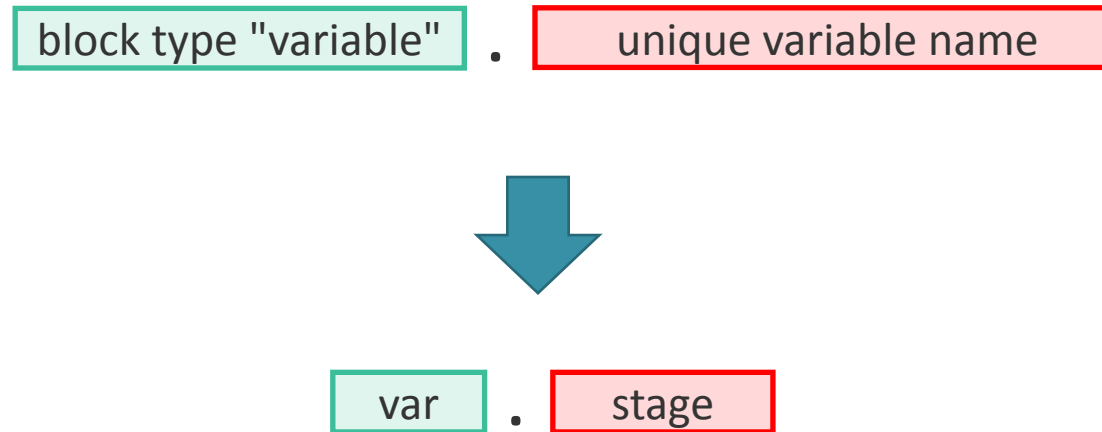
unique variable name

variable's attributes

- type
- description
- default value
- validation
- sensitive

# Terraform Komponenten

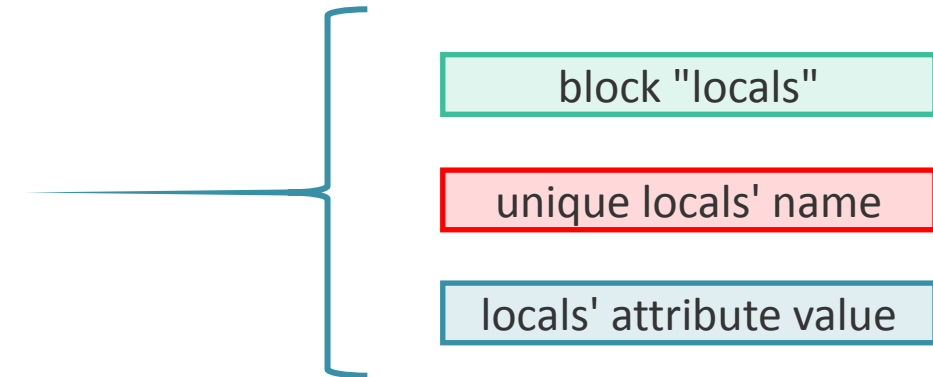
Referenz auf eine Variable



# Terraform Komponenten

## Locals - lokale Konstanten

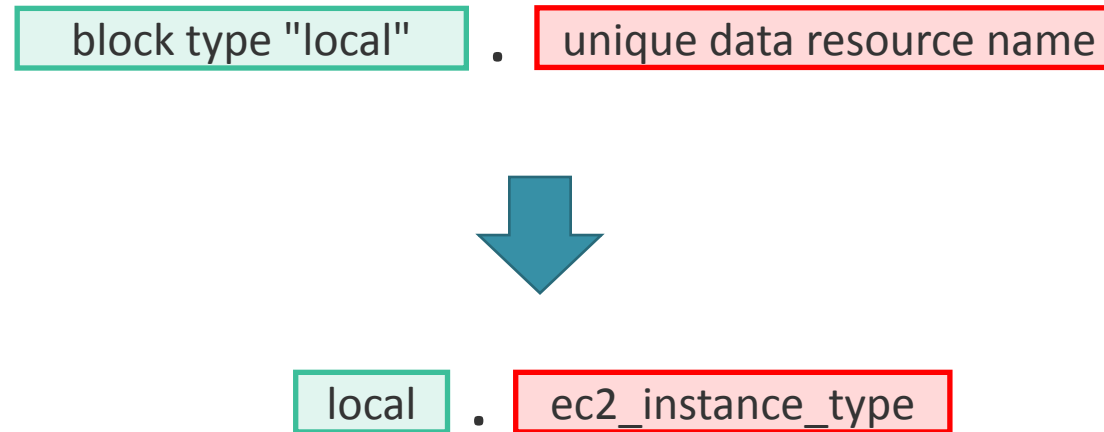
```
locals {  
  ec2_instance_type = "t4g.micro"  
}
```





# Terraform Komponenten

Referenz auf eine lokale Konstante



# Terraform Komponenten

## Output

```
output "server public ip" {  
  value = aws_instance.server.public_ip  
}
```

block "output"

unique output's name

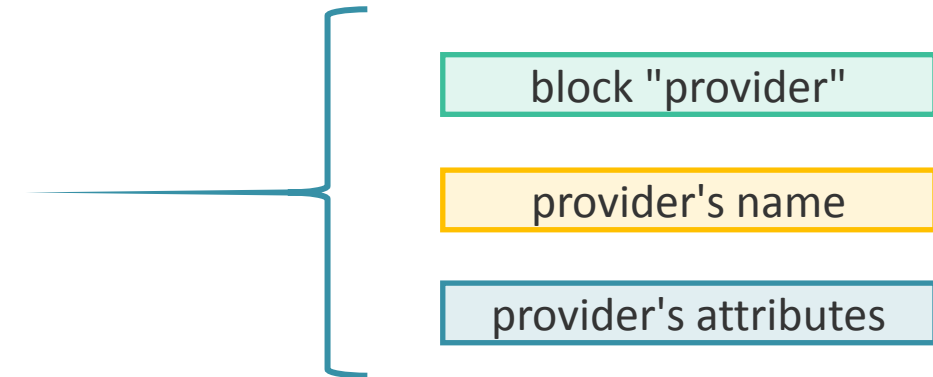
output's attributes

- value
- description
- sensitive
- depends\_on

# Terraform Komponenten

## Provider

```
provider "aws" {  
  region = "us-east-1"  
  profile = "profile-name"  
}
```



# Terraform Komponenten

## Built-in Funktionen

z.B.

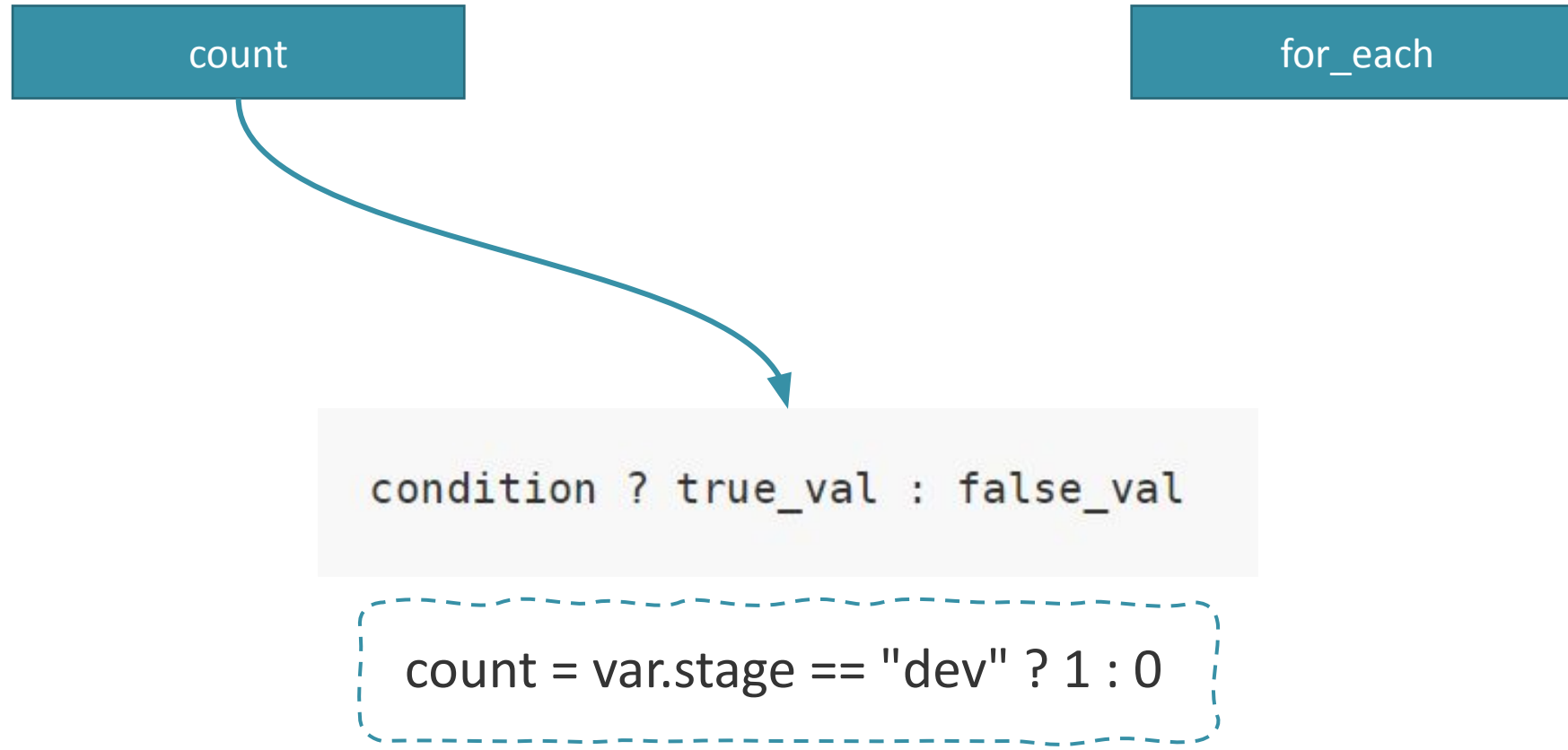
```
> upper("hello")  
HELLO
```

```
> abs(23)  
23  
> abs(0)  
0  
> abs(-12.4)  
12.4
```

```
> timestamp()  
2018-05-13T07:44:12Z
```

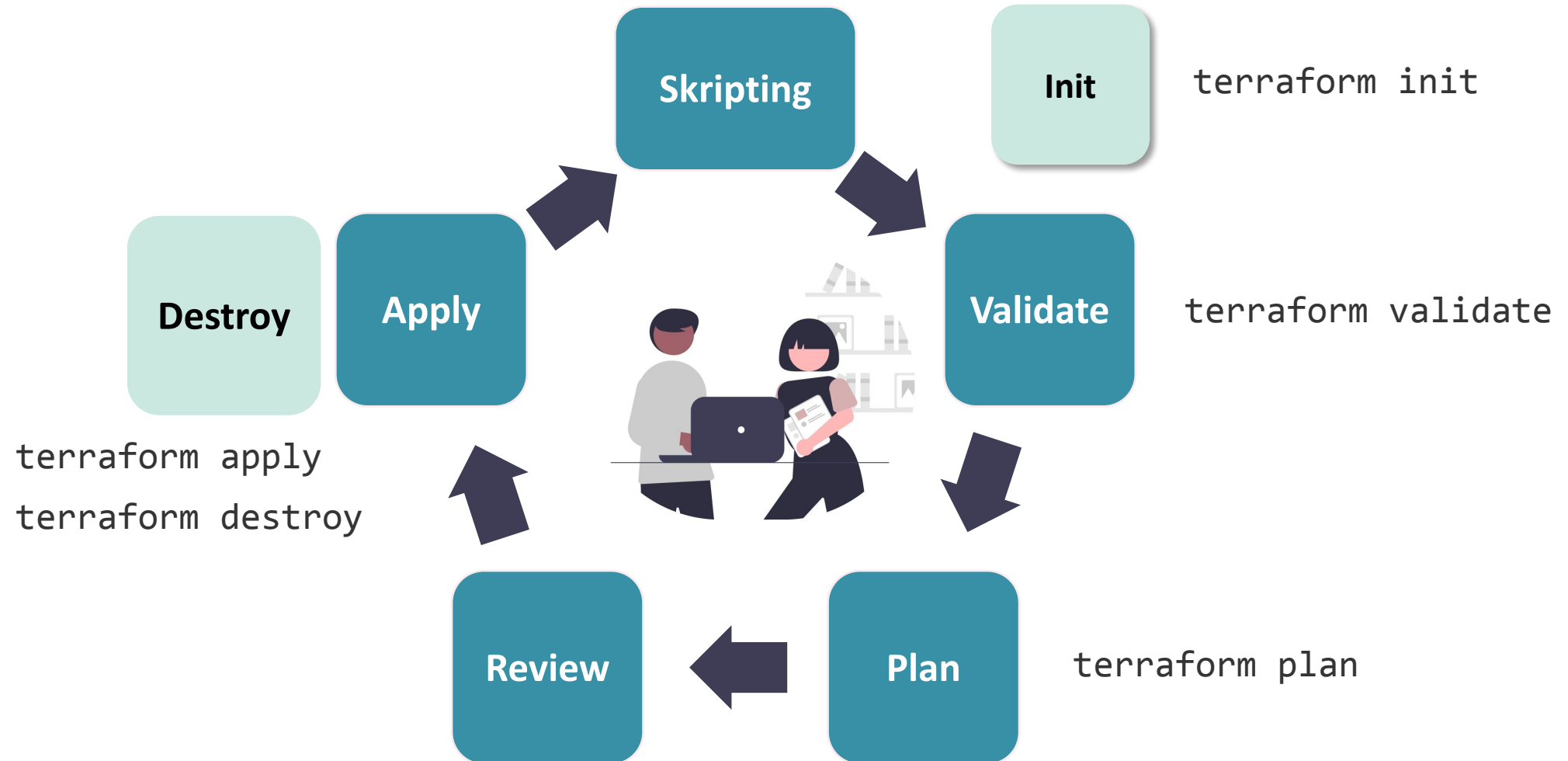
# Terraform Komponenten

## Conditions



# Terraform CLI

Mit Vollgas durch die Pipeline





# Aufgabe "Easy Up and Running S3 Website"

1. Baue gemeinsam im Live Coding eine Website, die über AWS S3 läuft.





presented by



# 02

# Terraform State Management

# Inhalte

---



Local vs Remote State



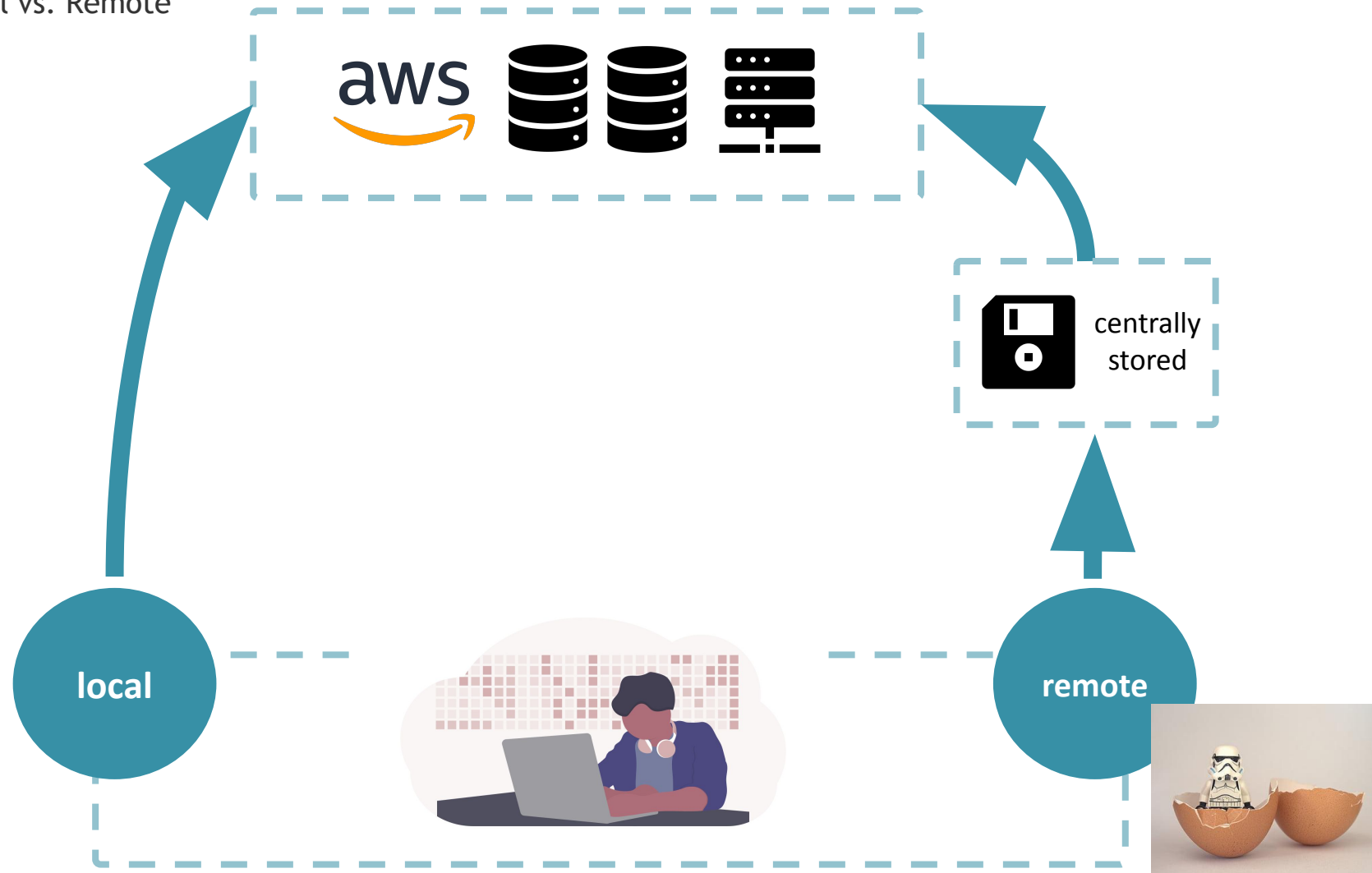
Aufbau und Manipulation



Ressourcen Import

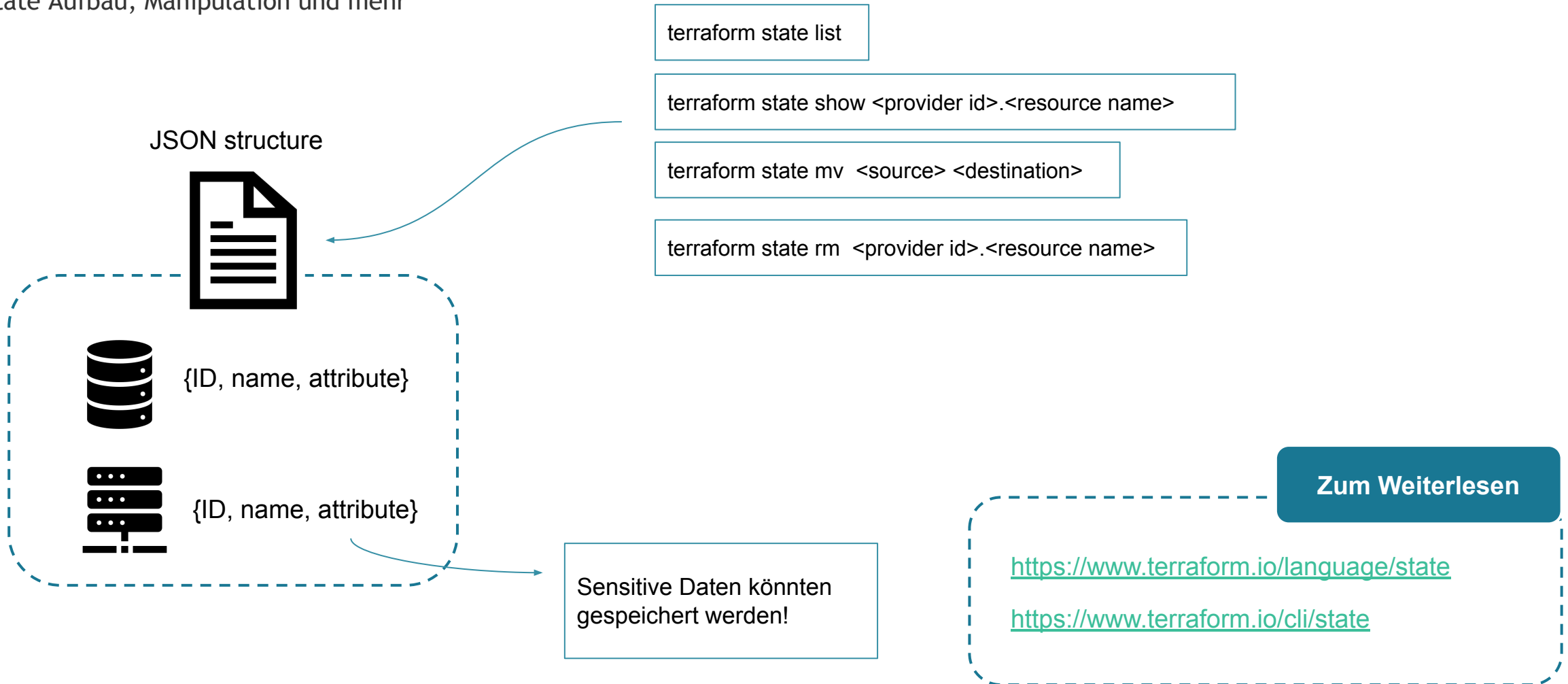
# Wie weiß Terraform denn, was ich habe?

State Management | Local vs. Remote



# Wie funktioniert es?

State Aufbau, Manipulation und mehr



# Oh weh, alles ist noch manuell!

Ressourcen importieren

## Import

---

S3 bucket can be imported using the `bucket`, e.g.

```
$ terraform import aws_s3_bucket.bucket bucket-name
```

[https://registry.terraform.io/providers/hashicorp/aws/latest/docs/resources/s3\\_bucket#import](https://registry.terraform.io/providers/hashicorp/aws/latest/docs/resources/s3_bucket#import)



```
terraform import aws_s3_bucket.website my-test-website
```

# Oh weh, alles ist noch manuell!

Skripte erzeugen

Zum Weiterlesen

<https://github.com/GoogleCloudPlatform/terraformer>

<https://github.com/dtan4/terraforming>

# Aufgabe "Baue dein State Management"

1. Baue gemeinsam im Live Coding das Terraform Backend mit S3 für unser State Management.