OeXYZ.

INFRASTRUCTURE AS CODE · FIELD GUIDE

AWS IaC mit Python

CloudFormation verstehen. AWS CDK sicher einsetzen. Infrastruktur reproduzierbar bauen und betreiben.

FOKUS
AWS CDK v2 · Python

BASIS
CloudFormation · YAML

STAND
Version 1.2 · 15. August 2026

00 · START

Aufbau und Nutzung des Guides

Der Guide ist Lernpfad und Nachschlagewerk zugleich. Lies zuerst 02 bis 05, baue dann ein kleines Projekt und nutze die restlichen Seiten beim Arbeiten.

Übungsregel

Jede Änderung folgt derselben Schleife: **Code → Synthese → Test → Diff → Deployment → Verifikation**. Klicke nicht parallel in der Konsole an denselben Ressourcen.

Einsteiger

Seiten 03 bis 13, dann Übung 1 auf Seite 27.

Praxis

Seiten 14 bis 22 bei Security, Betrieb und Fehlern.

Referenz

Seiten 23 bis 26 und 28 neben dem Terminal offen halten.

01 · ENTSCHEIDEN

Werkzeugwahl nach Betriebsmodell

Für ein reines AWS-Projekt mit Python ist CDK v2 meist der beste Einstieg. CloudFormation bleibt dabei die Ausführungs- und Zustandsbasis.

WERKZEUG	STÄRKEN	GEEIGNET FÜR	TRADE-OFF
AWS CDK v2 EMPFOHLEN	Python, Abstraktionen, Wiederverwendung, Tests, native AWS-Integration	AWS-only, komplexere Systeme, Entwicklerteams	Synthese und Token-Modell müssen verstanden werden
CloudFormation	Deklarativ, AWS-nativ, kein externes State-Backend	Kleine Stacks, bestehende YAML/JSON-Templates, Plattformteams	Mehr Wiederholung, große Templates werden schwer lesbar
AWS SAM	Kurze Serverless-Syntax, lokale Test- und Deploy-Befehle	Lambda, API Gateway, Event-getriebene Anwendungen	Fokus auf Serverless; darunter liegt CloudFormation
Terraform	Multi-Cloud, breites Provider-Ökosystem	Hybrid- und Multi-Cloud, etablierte Terraform-Teams	Eigenes State-Management und HCL statt Python

Die Beziehung

Python-CDK-Code
deine Konstrukte und Regeln

↓ synth

CloudFormation-Template
deklarativer Zielzustand

↓ deploy

AWS-Ressourcen
realer Zustand im Account

Nimm CDK, wenn ...

- Python deine Arbeitssprache ist.
- du sichere Standards kapseln willst.
- du wiederverwendbare Constructs brauchst.
- du Infrastruktur mit Unit-Tests prüfen willst.

Nimm CloudFormation direkt, wenn ...

- ein vorhandener Template-Bestand existiert.
- der Stack klein und deklarativ bleibt.
- keine Programmlogik nötig ist.

Nicht mischen ohne Grenze

Eine Ressource hat genau einen IaC-Eigentümer. CDK, Terraform und manuelle Änderungen dürfen nicht gleichzeitig dieselbe Ressource verwalten.

Arbeitsdefinition: IaC beschreibt einen versionierten und überprüfbaren Zielzustand. Ein Erzeugungsskript allein erfüllt diese Anforderung nicht.

02 · GRUNDLAGEN

Vom Quellcode zum tatsächlichen AWS-Zustand

01
Author
Python oder YAML im Repository ändern.

02
Validate
Syntax, Tests, Security und Policies prüfen.

03
Plan
CDK Diff oder Change Set lesen.

04
Apply
CloudFormation führt Änderungen geordnet aus.

05
Operate
Status, Drift, Logs, Kosten und Backups prüfen.

Die fünf Zustände, die du vergleichst

Source	Code im Git-Commit
Synthese	erzeugtes CloudFormation-Template
Change Set	geplante Operationen
Stack State	CloudFormation-Verwaltungszustand
Actual State	reale Eigenschaften der Ressourcen

Abweichungen zwischen Stack-Definition und realem Zustand heißen **Drift**.

CloudFormation denkt in Ressourcen

- **Logical ID:** Identität im Template. Änderung kann Ersatz auslösen.
- **Physical ID:** tatsächlicher Name oder ARN in AWS.
- **Dependency:** bestimmt Erstellungs- und Löschreihenfolge.
- **Update behavior:** in-place, interruption oder replacement.
- **Rollback:** versucht bei Fehlern zum vorherigen Zustand zurückzukehren.

Begriffe, die früh sitzen müssen

Stack
Deployment- und Lebenszyklusgrenze.

Construct
CDK-Baustein aus einer oder mehreren Ressourcen.

Asset
Datei, Lambda-Code oder Container-Image für das Deployment.

Bootstrap
CDK-Hilfsressourcen pro Account und Region.

Token
Wert, der erst bei Synthese oder Deployment aufgelöst wird.

Context
Lookup- oder Konfigurationswert für deterministische Synthese.

Change Set
Vorschau der geplanten CloudFormation-Änderungen.

Drift
Manuelle oder externe Abweichung vom IaC-Zielzustand.

Abschlusskriterien

Code committed, Tests grün, Diff geprüft, Deployment erfolgreich, Anwendung verifiziert, Alarme ruhig, kein unerwarteter Drift.

03 · VORBEREITUNG

Lokale Werkzeuge, Profil und Zielumgebung

Voraussetzungen

- Python 3.11 oder eine im Projekt festgelegte neuere Version
- Node.js für die CDK CLI
- AWS CLI v2
- AWS CDK CLI v2
- Git und ein Editor mit Python- und YAML-Unterstützung
- SSO, OIDC oder kurzlebige Rollen statt statischer Schlüssel

Identität vor jeder Änderung prüfen

```
aws sts get-caller-identity
aws configure list
aws configure get region

# Bei mehreren Umgebungen explizit:
$env:AWS_PROFILE = "sandbox-admin"
$env:AWS_REGION  = "eu-central-1"
```

Nie raten: Account-ID, Rolle und Region vor einem Deployment sichtbar bestätigen.

PowerShell · neues Python-CDK-Projekt — lokal

```
# Arbeitsordner anlegen und CDK-App initialisieren
New-Item -ItemType Directory aws-iac-lab
Set-Location aws-iac-lab
cdk init app --language python

# Virtuelle Umgebung aktivieren
.\.venv\Scripts\Activate.ps1
python -m pip install --upgrade pip
python -m pip install -r requirements.txt

# Zielumgebung einmalig für CDK vorbereiten
cdk bootstrap aws://123456789012/eu-central-1

# Früh prüfen
cdk doctor
cdk synth
```

Versionen festlegen

`requirements.txt` oder Lockfile committen. CLI und Library kompatibel halten.

Kontext committen

`cdk.context.json` committen, wenn Lookups verwendet werden. So bleibt Synthese reproduzierbar.

Artefakte ignorieren

`.venv/`, `cdk.out/`, Caches und lokale Geheimnisse gehören nicht in Git.

Bootstrap ist Infrastruktur

Der Bootstrap-Stack enthält unter anderem Asset-Speicher und Rollen. Nutze vertrauenswürdige Accounts, passende Berechtigungsgrenzen und konsistente Qualifier.

04 · STRUKTUR

Projektstruktur für wartbare CDK-Stacks

Empfohlener Aufbau

```
aws-iac-lab/
├─ app.py
├─ cdk.json
├─ cdk.context.json
├─ requirements.txt
├─ requirements-dev.txt
├─ pytest.ini
├─ src/
│  ├─ network_stack.py
│  ├─ data_stack.py
│  ├─ app_stack.py
│  └─ constructs/
│     └─ secure_bucket.py
├─ tests/
│  ├─ test_data_stack.py
│  └─ test_app_stack.py
└─ README.md
```

`app.py`

Komposition, Umgebungen, globale Tags und Stack-Abhängigkeiten. Keine umfangreiche Ressourcenlogik.

`src/*_stack.py`

Stacks nach Lebenszyklus und Eigentümer schneiden: Netzwerk, Daten, Anwendung, Observability.

`src/constructs/`

Wiederverwendbare sichere Bausteine mit klarer API und Tests.

`tests/`

Template-Assertions und Regeln. Prüfe Ergebnisse, nicht Implementierungsdetails.

Minimaler Einstiegspunkt

app.py — Python

```python
import os
import aws_cdk as cdk
from src.data_stack import DataStack

app = cdk.App()
environment = cdk.Environment(
    account=os.getenv("CDK_DEFAULT_ACCOUNT"),
    region=os.getenv("CDK_DEFAULT_REGION", "eu-central-1"),
)

stack = DataStack(
    app,
    "LabData",
    env=environment,
    termination_protection=True,
    description="OeXYZ AWS IaC learning data stack",
)
cdk.Tags.of(stack).add("Project", "aws-iac-lab")
cdk.Tags.of(stack).add("ManagedBy", "cdk")

app.synth()
```

Deterministisch

Gleicher Commit plus gleicher Context erzeugt dasselbe Template.

Keine Laufzeitlogik

CDK-Code läuft vor dem Deployment. Er reagiert nicht auf spätere AWS-Ereignisse.

05 · PYTHON

Beispiel: verschlüsselter S3-Daten-Stack

src/data_stack.py — AWS CDK v2 · Python

```python
from aws_cdk import (
    CfnOutput, RemovalPolicy, Stack,
    aws_kms as kms,
    aws_s3 as s3,
)
from constructs import Construct


class DataStack(Stack):
    def __init__(self, scope: Construct, construct_id: str, **kwargs) -> None:
        super().__init__(scope, construct_id, **kwargs)

        key = kms.Key(
            self,
            "DataKey",
            alias="alias/aws-iac-lab-data",
            description="KMS key for the AWS IaC learning data stack",
            enable_key_rotation=True,
            removal_policy=RemovalPolicy.RETAIN,
        )

        bucket = s3.Bucket(
            self,
            "DataBucket",
            block_public_access=s3.BlockPublicAccess.BLOCK_ALL,
            encryption=s3.BucketEncryption.KMS,
            encryption_key=key,
            enforce_ssl=True,
            versioned=True,
            removal_policy=RemovalPolicy.RETAIN,
        )

        CfnOutput(
            self,
            "BucketArn",
            value=bucket.bucket_arn,
            description="ARN of the encrypted data bucket",
        )
```

Kein fester Bucket-Name

CloudFormation erzeugt einen eindeutigen physischen Namen. Das erhöht Portabilität und verhindert Kollisionen.

Daten bleiben erhalten

`RETAIN` schützt bei Stack-Löschung oder Ersatz. Löschung ist dann ein bewusster separater Vorgang.

Schlüsselrotation aktiv

KMS-Key und Bucket-Verschlüsselung sind explizit. Zugriff wird später per `grant_*` vergeben.

> **Entwicklung gegen Produktion**
>
> Für kurzlebige Lernressourcen darfst du gezielt `RemovalPolicy.DESTROY` verwenden. Bei S3 sind zusätzlich `auto_delete_objects=True` und die Kostenwirkung zu verstehen. In produktiven Daten-Stacks ist `RETAIN` die sichere Basis.

Prüfschritte für dieses Beispiel

```
python -m pytest
cdk synth --strict
cdk diff LabData
cdk deploy LabData --require-approval broadening
```

06 · ABSTRAKTION

Construct-Ebenen und Deployment-Tokens

L1

CloudFormation-nah

`CfnBucket`, `CfnRole`. Direkte Abbildung der Ressourcenspezifikation. Maximale Kontrolle, wenig Komfort.

L2

Intent-basiert

`Bucket`, `Function`, `Vpc`. Sichere Hilfen, Grants, sinnvolle APIs. Standardwahl.

L3

Pattern

Mehrere Ressourcen für einen Use Case. Schnell, aber mit größerem Meinungsanteil und Scope.

Tokens: Werte aus der Zukunft

```
bucket = s3.Bucket(self, "Data")

# Noch kein echter Name beim Python-Lauf:
name = bucket.bucket_name

CfnOutput(self, "Name", value=name)
```

Der Bucket-Name ist ein Token. CDK serialisiert eine Referenz in das Template. String-Ausgaben beim Synth können deshalb Platzhalter zeigen.

Escape Hatch

```
cfn_bucket = bucket.node.default_child
cfn_bucket.add_property_override(
    "Metadata.Owner", "platform"
)
```

Nur verwenden, wenn L2 eine neue Eigenschaft noch nicht anbietet. Den Override testen und später wieder entfernen.

Construct-API gestalten

Eingaben

- fachliche Optionen statt roher Property-Sammlungen
- sichere Defaults
- kleine, typisierte Props

Ausgaben

- Interfaces wie `IBucket`
- ARNs und Tokens nur wenn nötig
- keine Geheimnisse in Outputs

Grenzen

- eine klare Verantwortung
- keine versteckten kontoübergreifenden Nebenwirkungen
- Dokumentation der Kosten- und Löschwirkung

Construct-ID ist Identität

Umbenennen oder Verschieben kann die Logical ID ändern und eine Ressource ersetzen. Vor Refactorings immer `cdk diff` lesen.

Review-Hinweis: Eine Abstraktion darf Wiederholung kapseln. Kosten-, Lösch- und Berechtigungswirkung müssen sichtbar bleiben.

07 · ARCHITEKTUR

Stack-Grenzen und Kopplung

Network

VPC, Subnetze, Endpoints, zentrale Sicherheitsgruppen.

SELTEN GEÄNDERT

Data

KMS, S3, Datenbanken, Backups.

STATEFUL

App

Compute, API, Queues, Rollen.

HÄUFIG GEÄNDERT

Ops

Logs, Alarme, Dashboards, Topics.

BEOBACHTET

Starke, typisierte Referenz

app.py — gleicher Account und gleiche Region

```python
data = DataStack(app, "Data", env=env)

api = ApiStack(
    app,
    "Api",
    data_bucket=data.bucket,
    env=env,
)

# CDK erzeugt nötige Export/Import-Beziehung.
```

api_stack.py — Least Privilege

```python
def __init__(..., data_bucket: s3.IBucket, **kwargs):
    super().__init__(..., **kwargs)
    fn = lambda_.Function(...)
    data_bucket.grant_read(fn)
```

Kopplung bewusst wählen

Direkte Referenz	Typisiert und bequem, erzeugt aber Deployment-Kopplung.
SSM Parameter	Lose Kopplung über Namen/ARN. Gut für getrennte Deployments.
CloudFormation Export	Nativ, aber Export kann nicht entfernt werden, solange Imports existieren.
Lookup	Für vorhandene Ressourcen; Context committen und Änderungen bewusst refreshen.

Cross-Stack-Deadlock

Eine verwendete Export-Referenz lässt sich nicht direkt löschen. Erst den Consumer entkoppeln und deployen, danach den Export entfernen.

Abhängigkeitsregeln

Automatisch

Ressourcenattribute und Grants erzeugen meist die passende Abhängigkeit.

Explizit

`stack.add_dependency(other)` nur für echte Ordnungsanforderungen ohne Referenz.

Vermeiden

Zyklen. Gemeinsame Ressource in einen dritten Stack verschieben oder lose koppeln.

08 · TESTS

Template-Assertions mit pytest

tests/test_data_stack.py — pytest

```python
import aws_cdk as cdk
from aws_cdk.assertions import Match, Template
from src.data_stack import DataStack


def test_bucket_is_encrypted_and_versioned():
    app = cdk.App()
    stack = DataStack(app, "TestData")
    template = Template.from_stack(stack)

    template.has_resource_properties(
        "AWS::S3::Bucket",
        {
            "VersioningConfiguration": {
                "Status": "Enabled"
            },
            "PublicAccessBlockConfiguration":
                Match.object_like({
                    "BlockPublicAcls": True,
                    "BlockPublicPolicy": True,
                }),
        },
    )
    template.resource_count_is("AWS::S3::Bucket", 1)
```

Testpyramide für IaC

Unit	Template-Assertions, schnell, pro Commit
Policy	cfn-guard, cdk-nag, Organisationsregeln
Snapshot	grobe Änderungen erkennen, nicht blind aktualisieren
Integration	in Sandbox deployen und AWS-Verhalten prüfen
Smoke	nach Deployment Endpunkt oder Ressource verifizieren

cdk-nag einschalten

```python
from aws_cdk import Aspects
from cdk_nag import AwsSolutionsChecks

Aspects.of(app).add(AwsSolutionsChecks())
```

Abhängigkeit separat pinnen. Suppressions nur mit konkreter Begründung und kleinstem Scope.

Validierungs-Gates

01 **Format** — ruff / black
› 02 **Types** — mypy
› 03 **Tests** — pytest
› 04 **Synth** — strict
› 05 **Policy** — nag / guard
› 06 **Diff** — review

Teste gewünschte Eigenschaften

„Bucket ist verschlüsselt und nicht öffentlich“ ist stabiler als „Logical ID lautet exakt DataBucketE3889A50“.

09 · TEMPLATE

Anatomie eines CloudFormation-Templates

template.yaml — CloudFormation

```yaml
AWSTemplateFormatVersion: "2010-09-09"
Description: OeXYZ AWS IaC learning storage stack

Parameters:
  Environment:
    Type: String
    AllowedValues: [dev, stage, prod]
    Default: dev

Conditions:
  IsProduction: !Equals [!Ref Environment, prod]

Resources:
  DataBucket:
    Type: AWS::S3::Bucket
    DeletionPolicy: Retain
    UpdateReplacePolicy: Retain
    Properties:
      BucketEncryption:
        ServerSideEncryptionConfiguration:
          - ServerSideEncryptionByDefault:
              SSEAlgorithm: AES256
      PublicAccessBlockConfiguration:
        BlockPublicAcls: true
        BlockPublicPolicy: true
        IgnorePublicAcls: true
        RestrictPublicBuckets: true
      VersioningConfiguration:
        Status: Enabled
      Tags:
        - Key: Environment
          Value: !Ref Environment
        - Key: ManagedBy
          Value: cloudformation

Outputs:
  DataBucketArn:
    Description: ARN of the encrypted data bucket
    Value: !GetAtt DataBucket.Arn
    Export:
      Name: !Sub "${AWS::StackName}-DataBucketArn"
```

Parameters

Echte Deployment-Eingaben. Constraints statt freier Werte.

Mappings

Statische Zuordnung, etwa Region zu AMI oder Konfiguration.

Conditions

Ressourcen oder Properties abhängig erzeugen.

Resources

Einziger Pflichtabschnitt. Logical IDs bleiben stabil.

Outputs

Nicht sensitive Ergebnisse und optionale Exports.

Metadata

Werkzeug-Metadaten, niemals Geheimnisse.

Rules

Parameterkombinationen vor Erstellung validieren.

Transform

SAM, Include oder Makros vor Verarbeitung anwenden.

10 · SPRACHE

Intrinsics, Pseudo-Parameter und Conditions

KURZFORM	BEDEUTUNG	BEISPIEL
`!Ref`	Parameterwert oder primärer Wert einer Ressource	`!Ref Environment`
`!GetAtt`	Attribut einer Ressource	`!GetAtt DataBucket.Arn`
`!Sub`	String mit Variablen zusammensetzen	`!Sub "${AWS::StackName}-logs"`
`!Join`	Liste mit Trennzeichen verbinden	`!Join [":", [a, b]]`
`!FindInMap`	Wert aus Mapping lesen	`!FindInMap [Config, !Ref AWS::Region, Size]`
`!ImportValue`	Export aus anderem Stack importieren	`!ImportValue shared-vpc-id`
`!If`	Property-Wert abhängig setzen	`!If [IsProd, 3, 1]`
`!Select`	Element aus Liste wählen	`!Select [0, !GetAZs ""]`
`!GetAZs`	Availability Zones einer Region	`!GetAZs !Ref AWS::Region`
`!Base64`	Text Base64-kodieren	`!Base64 "#!/bin/bash"`

Pseudo-Parameter

```
AWS::AccountId
AWS::Region
AWS::Partition
AWS::StackId
AWS::StackName
AWS::URLSuffix
AWS::NoValue
AWS::NotificationARNs
```

`AWS::Partition` statt festem `aws` verwenden, wenn Templates partitionsfähig sein sollen.

Bedingte Property

```
Conditions:
  IsProd: !Equals [!Ref Environment, prod]

Resources:
  Function:
    Type: AWS::Lambda::Function
    Properties:
      ReservedConcurrentExecutions: !If
        - IsProd
        - 50
        - !Ref AWS::NoValue
```

`AWS::NoValue` entfernt die Property vollständig, statt einen leeren Wert zu setzen.

YAML-Kurzformen nicht beliebig verschachteln

Bei komplexen Intrinsics die Langform oder ein Mapping verwenden. Lesbarkeit ist wichtiger als die kürzeste Syntax.

11 · KONTROLLE

Deployment mit Change Set und Drift-Prüfung

01
Lint
`cfn-lint`

02
Validate
Syntax und Schema

03
Change Set
Ersatz und IAM lesen

04
Execute
kontrolliert anwenden

05
Verify
Status und Funktion

Direkter Dev-Workflow

```
cfn-lint template.yaml

aws cloudformation validate-template \
  --template-body file://template.yaml

aws cloudformation deploy \
  --template-file template.yaml \
  --stack-name lab-data \
  --parameter-overrides Environment=dev \
  --capabilities CAPABILITY_NAMED_IAM \
  --no-fail-on-empty-changeset
```

In PowerShell Zeilenfortsetzung mit Backtick oder den Befehl einzeilig schreiben. Die Darstellung nutzt Backslashes aus Platzgründen.

Explizites Change Set

```
aws cloudformation create-change-set \
  --stack-name lab-data \
  --change-set-name review-001 \
  --change-set-type UPDATE \
  --template-body file://template.yaml \
  --capabilities CAPABILITY_NAMED_IAM

aws cloudformation describe-change-set \
  --stack-name lab-data \
  --change-set-name review-001

aws cloudformation execute-change-set \
  --stack-name lab-data \
  --change-set-name review-001
```

Für einen neuen Stack `--change-set-type CREATE` verwenden.

Drift erkennen

```
aws cloudformation detect-stack-drift \
  --stack-name lab-data

aws cloudformation describe-stack-resource-drifts \
  --stack-name lab-data
```

Fehlerursache finden

```
aws cloudformation describe-events \
  --stack-name lab-data \
  --filters FailedEvents=true
```

Konkrete Fehler vor „Resource creation cancelled“ untersuchen.

12 · IAM UND SECRETS

IAM-Grants und Secret-Referenzen

CDK: Grants statt Handarbeit

```
secret = secretsmanager.Secret.from_secret_name_v2(
    self, "DbSecret", "prod/app/database"
)

fn = lambda_.Function(...)
bucket.grant_read(fn)
queue.grant_send_messages(fn)
secret.grant_read(fn)
```

Grants erzeugen zielgerichtete Identity- oder Resource-Policies. Das Secret wird beim Synth nicht gelesen.

CloudFormation: dynamische Referenz

```
MasterUserPassword:
  '{{resolve:secretsmanager:prod/app/database:
    SecretString:password}}'
```

Der Wert bleibt außerhalb des Templates. Keine Ausgabe, kein Logging, kein Klartextparameter.

Least-Privilege-Check

- Welche konkrete Action ist nötig?
- Auf welchen ARN kann sie begrenzt werden?
- Ist eine Condition möglich?
- Wer darf die Rolle annehmen?
- Wer darf `iam:PassRole` verwenden?
- Ist KMS-Key-Policy zusätzlich korrekt?

Verbotene Muster

- `Action: "*"` und `Resource: "*"` ohne belegte Notwendigkeit
- Access Keys im Repository oder in Parametern
- Secrets in Outputs, User Data, Logs oder Stack-Namen
- breite Trust Policies

Prüfbereiche pro Stack

FLÄCHE	SICHERER STANDARD	PRÜFUNG
Identität	SSO/OIDC, kurzlebige Sessions, getrennte Rollen	`sts get-caller-identity`, CloudTrail
Daten	Verschlüsselung, Retention, Backups, Versionierung	Template-Tests, Restore-Test
Netzwerk	private Pfade, minimale Ingress-Regeln, VPC Endpoints	Security-Group-Review, Reachability
Deployment	Service Role, Permissions Boundary, Approval Gates	Change Set, Access Analyzer
Erkennung	Logs, Alarme, CloudTrail, Config oder Drift-Prüfung	Alarmtest und periodische Kontrolle

Secret Safety

Keine Befehle verwenden, die Secret-Werte auslesen. Für lokale Prozesse dynamische Secrets-Manager-Referenzen mit `asm-exec` zur Laufzeit auflösen, damit Werte nicht in den Arbeitskontext gelangen.

13 · ÄNDERUNGEN

Replacement, Import und sichere Refactorings

ÄNDERUNG	MÖGLICHE WIRKUNG	SICHERES VORGEHEN
Construct-ID oder Pfad ändern	Neue Logical ID, Ressource wird neu erstellt	`cdk diff`, Refactoring getrennt von Property-Änderungen
Immutable Property ändern	Replacement, möglicherweise Datenverlust oder Downtime	Change Set lesen, Migration und Rollback planen
Stateful-Ressource entfernen	Löschung oder Retain abhängig von Policy	`RETAIN`, Backup, Eigentumsübergang dokumentieren
Cross-Stack-Export entfernen	Deployment blockiert, solange Import existiert	Consumer zuerst entkoppeln, getrennt deployen
Manuell in AWS ändern	Drift, spätere Updates können Änderung überschreiben	Notfalländerung in IaC zurückführen und Drift bereinigen

Schutzschichten

- **Termination Protection:** schützt vor Stack-Löschung.
- **Removal Policy:** Verhalten bei Entfernung aus dem Stack.
- **UpdateReplacePolicy:** Verhalten beim Ersatz.
- **Stack Policy:** schützt ausgewählte Ressourcen vor Updates.
- **Backup:** schützt Daten unabhängig vom Stack.

Import vorhandener Ressourcen

1. Ist-Zustand und Eigentümer dokumentieren.
2. Ressource mit übereinstimmenden Properties modellieren.
3. Keine weiteren Änderungen im selben Schritt.
4. `cdk diff` und Import-Mapping prüfen.
5. `cdk import` ausführen.
6. Danach getrennt normalisieren.

Drei-Deployment-Regel für riskante Änderungen

DEPLOY 1

Kompatibilität

Neue und alte Struktur gleichzeitig unterstützen. Backup und Migrationspfad bereitstellen.

DEPLOY 2

Umschalten

Consumer auf die neue Ressource oder Referenz umstellen. Funktion verifizieren.

DEPLOY 3

Aufräumen

Alte Referenz oder Ressource entfernen. Retained Daten separat behandeln.

> **`cdk destroy` ist kein Aufräumskript**
>
> Vorher Diff, Schutzrichtlinien, Abhängigkeiten, Datenhaltung und Kosten verbleibender Retained-Ressourcen prüfen.

14 · MULTI-ACCOUNT

Account- und Regionengrenzen

Management

Organizations und Governance. Keine Workloads.

Security

Zentrale Logs, Security- und Audit-Funktionen.

Non-Prod

Entwicklung, Tests, Experimente mit Budgets.

Prod

Strenge Rollen, Approvals, Backups und Alarme.

CDK pro Zielumgebung

```
targets = {
    "dev": cdk.Environment(
        account="111111111111",
        region="eu-central-1",
    ),
    "prod": cdk.Environment(
        account="222222222222",
        region="eu-central-1",
    ),
}

for stage, env in targets.items():
    AppStage(
        app,
        f"App-{stage}",
        env=env,
        stage_name=stage,
    )
```

Account-IDs sind Konfiguration, keine Geheimnisse. Rollen und Regionen dennoch zentral und überprüfbar halten.

Grenzen und Mechanismen

Account	Blast Radius, Abrechnung, Quotas und IAM-Grenze
Region	Latenz, Datenresidenz, Service-Verfügbarkeit
Stack	Deployment und Lebenszyklus
Stage	zusammengehörige Stacks einer Umgebung
StackSets	standardisierte Ausrollung über Accounts und Regionen
SCP	maximal zulässige Aktionen in Organizations

Deployment-Prinzipien

Gleicher Artefaktstand

Ein getesteter Commit wandert durch Umgebungen. Nicht pro Umgebung neu bauen.

Unterschiede als Daten

Kapazität, Domains und Features konfigurierbar machen. Architektur nicht per Copy-paste verzweigen.

Prod bewusst

Manuelle Freigabe, Wartungsfenster, Rollback-Plan und Observability vor Deployment.

Bootstrap pro Account und Region

Jede CDK-Zielumgebung benötigt die passenden Bootstrap-Ressourcen. Trust-Beziehungen und Permissions Boundaries klein halten.

15 · PIPELINE

CI/CD-Gates für Infrastructure as Code

01 **Install** pinned deps › 02 **Lint** Python + CFN › 03 **Test** assertions › 04 **Synth** artifact › 05 **Scan** policy › 06 **Approve** prod diff › 07 **Deploy** verify

Pipeline-Gates

Pull Request	Format, Lint, Typen, Unit-Tests, Synth, Policy, Diff-Artefakt
Merge	unveränderliches Artefakt erzeugen, Signatur und Herkunft festhalten
Dev	automatisch deployen, Integrationstest, Smoke-Test
Stage	gleicher Build, realistische Datenflüsse, Last- und Restore-Tests
Prod	Diff-Freigabe, Change Window, Deployment, Alarme, Verifikation

OIDC statt statischer Schlüssel

CI-System erhält über einen vertrauenswürdigen Identity Provider kurzlebige AWS-Zugangsdaten.

- Trust auf Repository, Branch und Workflow begrenzen.
- Separate Rollen für Synth, Diff und Deploy.
- `iam:PassRole` nur für definierte Rollen.
- Session-Name und Tags für Audit nutzen.
- Keine langfristigen Access Keys als CI-Secrets.

Artefakte

`cdk.out`, Template, Asset-Hashes, Testberichte, Policy-Ergebnisse und Diff nachvollziehbar speichern.

Parallelität

Pro Stack nur ein Writer. Pipeline-Concurrency oder Locking verhindert kollidierende Deployments.

Rollback

CloudFormation-Rollback ist kein vollständiger Anwendungsrollback. Datenmigrationen separat planen.

Self-mutation bewusst einsetzen

Eine Pipeline, die ihre eigene Infrastruktur ändert, braucht besonders enge Rechte, Staging, Versionskontrolle und einen dokumentierten Recovery-Pfad.

16 · WELL-ARCHITECTED

Betriebsanforderungen nach dem Deployment

Operational Excellence

Runbooks, kleine reversible Changes, automatische Verifikation, Drift-Checks.

Security

Least Privilege, Verschlüsselung, Audit, getrennte Accounts und Secret-Referenzen.

Reliability

Backups, Restore-Tests, Multi-AZ wo nötig, Quotas und Failure Modes.

Performance

Messbare Anforderungen, passende Services, Load-Tests und Skalierungsgrenzen.

Cost Optimization

Tags, Budgets, Right-Sizing, Retention, Dev-Abschaltung, Eigentümer.

Sustainability

Bedarfsgerechte Kapazität, Managed Services, effiziente Regionen und Architekturen.

Verbindliche Tags

```
Project       = aws-iac-lab
Environment   = dev | stage | prod
Owner         = platform-team
CostCenter    = learning
ManagedBy     = cdk
DataClass     = internal
Criticality   = low | medium | high
```

Tag-Policies zentral definieren. Tagging ist für Kosten und Eigentum wichtig, ersetzt aber keine IAM- oder Account-Grenze.

Operational Readiness

- Dashboard und Alarme existieren.
- Log-Retention und Datenklassifizierung sind festgelegt.
- Backup wurde erfolgreich wiederhergestellt.
- Service Quotas passen zur Last.
- Fehlerbudget und SLO sind bekannt.
- Runbook nennt Diagnose und Eskalation.
- Kostenalarm und Budget sind aktiv.
- Eigentümer und On-Call-Weg sind klar.
- Drift-Prüfung läuft periodisch.

Diff ist keine Kostenprognose

Ein kleiner Template-Unterschied kann große Kostenwirkung haben. Preise, Datenvolumen, Requests, Logs, NAT-Verkehr und Retention separat bewerten.

Betriebsfreigabe: Das System ist beobachtbar, wiederherstellbar, kostenmäßig eingeordnet und einem verantwortlichen Team zugeordnet.

17 · SERVERLESS

Wann AWS SAM sinnvoll ist

SAM ist eine CloudFormation-Erweiterung für Lambda, API Gateway, Events und verwandte Services. Der Transform erzeugt daraus reguläre CloudFormation-Ressourcen.

template.yaml — AWS SAM

```
Transform: AWS::Serverless-2016-10-31
Resources:
  ApiFunction:
    Type: AWS::Serverless::Function
    Properties:
      Runtime: python3.13
      Handler: app.handler
      CodeUri: src/
      Timeout: 10
      Events:
        Api:
          Type: Api
          Properties:
            Path: /health
            Method: get
```

SAM passt hier

- Lambda und Events bilden den Kern.
- Lokales Build und Emulation sind wichtig.
- Das Team bevorzugt deklaratives YAML.
- Nur wenige allgemeine Infrastrukturkomponenten sind nötig.

CDK passt hier

- Viele Services werden komponiert.
- Python-Abstraktionen und Unit-Tests sind zentral.
- Mehrere Stacks und Accounts gehören zusammen.

Lokaler Workflow

```
sam validate --lint
sam build
sam local start-api
sam deploy --guided
```

Transform lesen

Das erzeugte CloudFormation-Template prüfen, besonders IAM, APIs und implizite Ressourcen.

Runtime prüfen

Die Beispielruntime ist versionsabhängig. Vor Einsatz die aktuell unterstützten Lambda-Runtimes verifizieren.

Lokale Grenzen

Emulation ersetzt keine Integrationstests in einer echten AWS-Sandbox.

Dokumentationsgebundene Referenz

SAM ist eine Einordnung in diesem CDK-zentrierten Guide. Befehle und Runtime müssen gegen die installierte SAM-CLI und die aktuelle AWS-Dokumentation geprüft werden.

17 · MULTI-PROVIDER

Wann Terraform sinnvoll ist

Terraform ist sinnvoll, wenn Multi-Cloud, Hybrid-Infrastruktur oder ein vorhandener Plattformstandard wichtiger sind als die native Python-CDK-Integration.

Der Kernworkflow

```
terraform fmt -check
terraform init
terraform validate
terraform plan -out=tfplan
terraform apply tfplan
```

Der gespeicherte Plan wird geprüft und genau dieser Plan angewendet.

State ist Betriebsinfrastruktur

Backend	remote, verschlüsselt und verfügbar
Locking	nur ein Writer pro State
Zugriff	kleinster Kreis, auditierbar
Backup	Versionierung und Recovery getestet
Secrets	sensitive Markierung ist kein Verschlüsselungsersatz

Entscheidung in 30 Sekunden

FRAGE	ANTWORT	STARTPUNKT
Nur AWS und Python-Kompetenz?	Ja	**AWS CDK v2**
Fast nur Lambda, API Gateway und Events?	Ja	**AWS SAM**
Kleines bestehendes YAML-Template?	Ja	**CloudFormation**
Mehrere Clouds oder Provider?	Ja	**Terraform**
Schon ein standardisiertes Plattformwerkzeug?	Ja	Bestehenden Standard verbessern

Provider pinnen

Upgrades getrennt planen und Lockfile committen.

Module versionieren

Kleine APIs, sichere Defaults und dokumentierte Migrationen.

Eigentum trennen

Terraform und CloudFormation verwalten nie dieselbe Ressource gleichzeitig.

18 · TROUBLESHOOTING

Häufige Fehlerbilder

SYMPTOM	TYPISCHE URSACHE	NÄCHSTER SCHRITT
`NoCredentials` `ExpiredToken`	SSO-Session abgelaufen, falsches Profil oder Rolle	`aws sts get-caller-identity`, Profil und Region prüfen
`AccessDenied`	fehlende Action, Resource, Condition, Trust oder KMS-Key-Policy	genaue API-Action und ARN lesen; CloudTrail korrelieren
`AlreadyExists`	physischer Name kollidiert oder Ressource existiert außerhalb des Stacks	Namen generieren lassen oder kontrolliert importieren
`ROLLBACK_COMPLETE`	Create fehlgeschlagen und zurückgerollt	Failure Events prüfen; neuen Create bewusst vorbereiten
`UPDATE_ROLLBACK_FAILED`	Rollback konnte Ressource nicht zurücksetzen	betroffene Ressource reparieren, dann Rollback fortsetzen
Export cannot be deleted	anderer Stack importiert den Export	Consumer zuerst entkoppeln und deployen
Dependency cycle	gegenseitige Referenzen zwischen Ressourcen oder Stacks	gemeinsame Ressource extrahieren oder lose koppeln
Asset publish failed	Bootstrap, Docker, Pfad oder Berechtigung	`cdk doctor`, Bootstrap und Asset-Pfad prüfen
Unerwartetes Replacement	Logical ID oder immutable Property geändert	Deployment stoppen, Diff und Spezifikation lesen
Stack drifted	manuelle Änderung oder externer Controller	Eigentümer klären; IaC oder Ist-Zustand korrigieren
S3 bleibt nach Destroy	`RETAIN` oder nicht leerer/versionierter Bucket	Daten und Ressource bewusst separat behandeln

Template-Level

Falsche Property, Dependency, IAM-Definition oder unverträgliche Änderung im IaC-Code.

Environment-Level

Quota, fehlende Berechtigung, Namenskollision, Servicezustand oder vorhandene Ressource.

Application-Level

Deployment erfolgreich, aber Health Check, Datenmigration oder fachlicher Test schlägt fehl.

> **Keine Template-Änderung für ein reines Umgebungsproblem**
>
> Erst die Fehlerklasse bestimmen. Sonst kaschiert eine Codeänderung Quotas, Rollenfehler oder inkonsistenten Zustand.

18 · TROUBLESHOOTING

Diagnoseablauf für fehlgeschlagene Deployments

Diagnosefolge

1. Account, Rolle, Region und Stack bestätigen.
2. Ersten spezifischen Failure Event finden.
3. Alle parallelen spezifischen Fehler sammeln.
4. Template-, Environment- oder Application-Level klassifizieren.
5. Kleinste sichere Korrektur entwerfen.
6. Diff lesen, erneut deployen und verifizieren.

read-only Diagnose — AWS CLI v2

```
aws sts get-caller-identity

aws cloudformation describe-events \
  --stack-name LabData \
  --filters FailedEvents=true

aws cloudformation describe-stack-resources \
  --stack-name LabData

aws cloudtrail lookup-events \
  --lookup-attributes \
  AttributeKey=EventName,AttributeValue=CreateBucket
```

„Resource creation cancelled“ ist meist Folge, nicht Ursache

Suche nach einem Event mit konkreter Fehlermeldung. Bei mehreren echten Fehlern alle Berechtigungs- oder Quota-Lücken gemeinsam erfassen.

Evidence Pack für eine saubere Übergabe

Kontext

Zeitpunkt, Account, Region, Stack, Commit und Pipeline-Run.

Plan

CDK Diff oder Change Set, erwartete Ersetzungen und IAM-Änderungen.

Events

Alle spezifischen Failure Events, nicht nur die letzte Meldung.

Audit

Relevante CloudTrail-Events und aufgerufene API-Actions.

Impact

Betroffene Nutzer, Daten, Regionen und laufende Ressourcen.

Nächster Schritt

Kleinste reversible Korrektur plus Verifikation.

CLI-Voraussetzung

`describe-events` ist die aktuelle CloudFormation-API mit flexiblen Filtern. Ältere AWS-CLI-Versionen kennen den Befehl möglicherweise noch nicht.

19 · REFERENZ

AWS CDK v2: täglicher Workflow

```
# Projekt und Version
$ cdk init app --language python
$ cdk doctor
$ cdk --version

# Umgebung
$ cdk bootstrap aws://ACCOUNT/REGION
$ cdk bootstrap --show-template

# Erkunden
$ cdk list
$ cdk context
$ cdk metadata StackName
```

```
# Qualität und Plan
$ python -m pytest
$ cdk synth --strict
$ cdk diff StackName
$ cdk diff --security-only

# Deployment
$ cdk deploy StackName
$ cdk deploy --all
$ cdk deploy --require-approval broadening
$ cdk deploy --outputs-file outputs.json
```

Freigabefolge für Produktion

01	02	03	04	05	06
Test	**Synth**	**Diff**	**Approve**	**Deploy**	**Verify**
pytest	strict	replacement	review	CloudFormation	smoke test

Vor jedem Deploy

`aws sts get-caller-identity` ausführen und Account, Rolle sowie Region sichtbar bestätigen.

Validierter Stand

Diese Befehle wurden für Version 1.2 gegen AWS CDK CLI `2.1136.0` geprüft.

19 · REFERENZ

CDK-Kommandos für Import, Drift und Recovery

```
# Schnelle lokale Entwicklung
$ cdk watch StackName

# Bestehende Ressourcen
$ cdk import StackName
$ cdk import --resource-mapping mapping.json

# Drift
$ cdk drift StackName
$ cdk drift StackName --fail
```

```
# Diagnose und Recovery
$ cdk deploy StackName --verbose
$ cdk rollback StackName
$ cdk rollback StackName --orphan LogicalId

# Entfernen
$ cdk destroy StackName
```

OPTION	WANN SINNVOLL	GRENZE
`--profile NAME`	explizites AWS-Profil	Identität trotzdem mit STS prüfen
`--context k=v`	CDK-Kontext setzen	niemals für Geheimnisse verwenden
`--exclusively`	nur gewählten Stack deployen	Abhängigkeiten müssen bereits passen
`--concurrency N`	unabhängige Stacks parallel	Quotas und Shared Resources beachten
`--hotswap`	schnelle lokale Entwicklung	umgeht regulären CloudFormation-Weg; nie Produktion
`--no-rollback`	gezielte Diagnose	hinterlässt Teilzustand; kein Standard

Import enthält nur Import

Beim `cdk import` keine Updates oder Löschungen im selben Diff mischen. Die Ressource zuerst exakt modellieren und erst nach erfolgreichem Import normalisieren.

20 · REFERENZ

CloudFormation: Validierung und Zustand

```
# Lokale optionale Werkzeuge
$ cfn-lint template.yaml
$ cfn-guard validate -r rules -d template.yaml

# AWS-Syntaxprüfung
$ aws cloudformation validate-template
  --template-body file://template.yaml

# Direkter Dev-Deploy
$ aws cloudformation deploy
  --template-file template.yaml
  --stack-name NAME
  --capabilities CAPABILITY_NAMED_IAM
  --no-fail-on-empty-changeset
```

```
# Inventar und Status
$ aws cloudformation list-stacks
$ aws cloudformation describe-stacks
  --stack-name NAME
$ aws cloudformation describe-stack-resources
  --stack-name NAME

# Drift
$ aws cloudformation detect-stack-drift
  --stack-name NAME
$ aws cloudformation describe-stack-resource-drifts
  --stack-name NAME
```

CREATE

Stack existiert noch nicht. Fehler kann in `ROLLBACK_COMPLETE` enden.

UPDATE

Template und Parameter werden mit dem verwalteten Zustand verglichen.

DELETE

Policies und Abhängigkeiten bestimmen, welche Ressourcen verbleiben.

Validierter Stand

Die AWS-Befehle wurden für Version 1.2 gegen AWS CLI `2.36.19` geprüft. `cfn-lint` und `cfn-guard` bleiben optionale, separat zu installierende Werkzeuge.

20 · REFERENZ

CloudFormation: Change Sets und Recovery

explizites Change Set — AWS CLI v2

```
aws cloudformation create-change-set \
  --stack-name NAME \
  --change-set-name review-001 \
  --change-set-type UPDATE \
  --template-body file://template.yaml \
  --capabilities CAPABILITY_NAMED_IAM

aws cloudformation describe-change-set \
  --stack-name NAME \
  --change-set-name review-001

aws cloudformation execute-change-set \
  --stack-name NAME \
  --change-set-name review-001
```

Fehler und Recovery — read first

```
aws cloudformation describe-events \
  --stack-name NAME \
  --filters FailedEvents=true

aws cloudformation cancel-update-stack \
  --stack-name NAME

aws cloudformation continue-update-rollback \
  --stack-name NAME
```

Recovery erst nach Ursache

Vor `continue-update-rollback` die blockierende Ressource und ihren realen Zustand verstehen.

Capabilities

`CAPABILITY_IAM`	IAM-Ressourcen mit generierten Namen
`CAPABILITY_NAMED_IAM`	IAM-Ressourcen mit expliziten Namen
`CAPABILITY_AUTO_EXPAND`	Makros dürfen das Template vor Deployment erweitern

Review

Replacement, IAM, Netzwerk, Datenhaltung und Tags prüfen.

Execute

Nur das gelesene Change Set ausführen.

Verify

Stack-Status, Anwendung, Alarme und Drift kontrollieren.

21 · PRAXIS

Übungsplan für vier Wochen

WOCHE 1

Stack-Grundlagen

Bau: verschlüsselter, versionierter S3-Bucket mit KMS-Key, Tags und Outputs.

- CloudFormation-Template nach Synth lesen
- Template-Assertions schreiben
- Diff erklären können
- Deploy und Retention verifizieren

WOCHE 2

Event-getriebene Anwendung

Bau: S3 → SQS → Lambda mit DLQ, Alarm und minimalen Grants.

- Timeout und Visibility Timeout begründen
- Fehlerfall erzeugen
- DLQ und Alarm prüfen
- Logs ohne Secrets kontrollieren

WOCHE 3

Mehrere Stacks

Bau: Data-, App- und Ops-Stack in Dev und Prod-Konfiguration.

- Abhängigkeiten zeichnen
- Stateful und Stateless trennen
- Cross-Stack-Referenz entfernen üben
- Drift erkennen und korrigieren

WOCHE 4

Pipeline und Recovery

Bau: PR-Checks, Sandbox-Deploy, Approval und Smoke-Test.

- OIDC-Rolle begrenzen
- Policy-Gate absichtlich brechen
- Replacement im Diff erkennen
- Backup und Restore demonstrieren

Fragen für das technische Review

Problem
Welches konkrete Problem löst der Stack und für wen?

Grenze
Was kann diese Version ausdrücklich nicht?

Beweis
Welche Tests und Messwerte zeigen, dass sie funktioniert?

Risiko
Welche Änderung kann Datenverlust, Downtime oder hohe Kosten auslösen?

Recovery
Wie wird aus Backup oder vorherigem Stand wiederhergestellt?

Betrieb
Wer sieht Fehler und wer reagiert darauf?

Abschlussprojekt

Dokumentiere Architektur, IaC-Code, Tests, Threats, Kostenannahmen, Deployment, Rollback, Restore und bekannte Grenzen in einem öffentlichen oder privaten Repository.

22 · NACHSCHLAGEN

Kompaktglossar

ARN	globales AWS-Ressourcenkennzeichen
Asset	Datei, Codebundle oder Container-Image für ein Deployment
Blast Radius	maximale Auswirkung eines Fehlers
Bootstrap	CDK-Hilfsressourcen pro Account und Region
Change Set	CloudFormation-Vorschau geplanter Änderungen
Construct	CDK-Baustein aus einer oder mehreren Ressourcen
Context	CDK-Werte für Synthese und Lookups
Drift	Abweichung zwischen erwartetem und realem Zustand
Idempotenz	wiederholte Ausführung führt zum gleichen Zielzustand
Logical ID	Identität einer Ressource im Template
Physical ID	echter Name oder Identifier in AWS
Replacement	alte Ressource wird durch eine neue ersetzt
Rollback	Rückkehr zum letzten stabilen Stack-Zustand
Stateful	Ressource enthält erhaltenswerte Daten
Synthese	CDK erzeugt CloudFormation und Assets
Token	noch nicht aufgelöster Deployment-Wert

Arbeitsregeln

- Ein IaC-Eigentümer pro Ressource.
- Keine Produktion ohne gelesenen Diff.
- Backups zählen erst nach erfolgreichem Restore.
- Secrets referenzieren, nicht auslesen.
- Stateful-Ressourcen separat schützen.
- Manuelle Änderungen sind Drift, keine Dokumentation.

Release-Check

- [ ] Account, Rolle und Region bestätigt
- [ ] Tests und Policy-Gates erfolgreich
- [ ] Replacement und IAM im Diff geprüft
- [ ] Rollback und Restore verstanden
- [ ] Smoke-Test und Alarme verifiziert
- [ ] Drift und Kostenwirkung bewertet

> Systeme verstehen. Komplexität reduzieren. Dinge bauen, die bleiben.

22 · NACHSCHLAGEN

Primärquellen und Release-Validierung

AWS CDK v2 Developer Guide
docs.aws.amazon.com/cdk/v2/guide/home.html

AWS CDK CLI Reference
docs.aws.amazon.com/cdk/v2/guide/ref-cli-cmd.html

AWS CDK Python API Reference
docs.aws.amazon.com/cdk/api/v2/python/

AWS CloudFormation User Guide
docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/Welcome.html

CloudFormation Best Practices
docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/best-practices.html

CloudFormation CLI: describe-events
docs.aws.amazon.com/cli/latest/reference/cloudformation/describe-events.html

Intrinsic Function Reference
docs.aws.amazon.com/AWSCloudFormation/latest/TemplateReference/intrinsic-function-reference.html

Dynamic References
docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/dynamic-references.html

CloudFormation Guard
docs.aws.amazon.com/cfn-guard/latest/ug/what-is-guard.html

AWS SAM Developer Guide
docs.aws.amazon.com/serverless-application-model/latest/developerguide/what-is-sam.html

Choosing an IaC Tool
docs.aws.amazon.com/prescriptive-guidance/latest/choose-iac-tool/choose-tool.html

AWS Well-Architected Framework
docs.aws.amazon.com/wellarchitected/latest/framework/welcome.html

Release-Validierung

AWS CLI `2.36.19` und gepinnte AWS CDK CLI `2.1136.0`. Der generierte `validation-report.md` dokumentiert jeden geprüften Befehl und jede erwartete Option.

Portabler Build

Relative Font-Assets, Inter-Lizenz, Python-Build, Überlaufprüfung, PDF-Seitenprüfung und Release-ZIP sind Teil des Quellpakets.

Stand und Verantwortung

Version 1.2 wurde am 15. August 2026 erstellt. Vor produktiven Änderungen die aktuelle Resource Specification, API-Referenz, Berechtigungen, Quotas und das erzeugte Change Set prüfen.

OeXYZ.
Systeme verstehen. Komplexität reduzieren. Dinge bauen, die bleiben.