

Infrastructure as Code (IaC) Getting Started Guide: Terraform Edition

Infrastructure as Code (IaC) Getting Started Guide

Table of Contents

- [Introduction](#)
- [Core Concepts](#)
 - [Providers / Services](#)
 - [Resources](#)
 - [Modules / Nested Stacks / Constructs](#)
 - [State Management](#)
- [Terraform](#)
 - [Installation](#)
 - [Basic Syntax and Structure](#)
 - [Creating and Managing Resources](#)
 - [Version Control](#)
 - [Testing Methods](#)
- [AWS CloudFormation](#)
 - [Installation & Setup](#)
 - [YAML Templates](#)
 - [Stack Management](#)
 - [Version Control](#)
 - [Testing Methods](#)
- [AWS Cloud Development Kit \(CDK\)](#)
 - [Installation & Setup](#)
 - [Basic Syntax \(TypeScript Example\)](#)
 - [Creating and Deploying Stacks](#)
 - [Version Control](#)
 - [Testing Methods](#)
- [Conclusion](#)

Introduction [🔗](#)

Infrastructure as Code (IaC) is the practice of managing and provisioning computing infrastructure using machine-readable configuration files, rather than through physical hardware configuration or interactive configuration tools.

Terraform by HashiCorp is one of the most popular cross-platform IaC tools. AWS also provides two robust native options: AWS CloudFormation and AWS Cloud Development Kit (CDK).

Core Concepts [🔗](#)

Providers / Services [🔗](#)

Defines which platform or cloud provider the configuration will manage.

Resources [🔗](#)

Infrastructure components like virtual machines, databases, or networking elements.

Modules / Nested Stacks / Constructs [🔗](#)

Reusable infrastructure definitions, grouped for scalability and reusability.

State Management [🔗](#)

Tracks the real-world infrastructure resources so changes can be applied incrementally.

Terraform [🔗](#)

Installation [🔗](#)

1. Visit: [🔗 Install | Terraform | HashiCorp Developer](#)
2. Download and install.
3. Verify with: `terraform -version`

Basic Syntax and Structure [🔗](#)

Terraform uses .tf files written in HCL (HashiCorp Configuration Language).

```
1 provider "aws" {
2   region = "us-east-1"
3 }
4
5 resource "aws_instance" "example" {
6   ami           = "ami-0c55b159cbf1f0"
7   instance_type = "t2.micro"
8 }
```

Creating and Managing Resources [🔗](#)

```
1 terraform init
2 terraform plan
3 terraform apply
4 terraform destroy
```

Version Control [🔗](#)

Use Git to store .tf files. Exclude terraform.tfstate and .terraform/.

Testing Methods [🔗](#)

- **Terraform validate**: Check syntax correctness.
 - **Terraform plan**: Preview infrastructure changes.
 - **Terratest** (Go-based): Automate infrastructure testing.
-

AWS CloudFormation [↗](#)

Installation & Setup [↗](#)

CloudFormation is built into AWS; no installation required. Use via Console, AWS CLI, or SDK.

YAML Templates [↗](#)

Example (YAML):

```
1 Resources:
2   MyInstance:
3     Type: AWS::EC2::Instance
4     Properties:
5       InstanceType: t2.micro
6       ImageId: ami-0c55b159cbfafa1f0
```

Stack Management [↗](#)

```
1 aws cloudformation create-stack --stack-name myStack --template-body file://template.yaml
2 aws cloudformation update-stack
3 aws cloudformation delete-stack
```

Version Control [↗](#)

Track .yaml or .json templates in Git.

Testing Methods [↗](#)

- **cfn-lint**: Linter for template syntax.
- **TaskCat**: AWS tool to test templates by deploying in multiple regions.
- **Manual validation**: Use Change Sets to preview changes.

AWS Cloud Development Kit (CDK) [↗](#)

Installation & Setup [↗](#)

```
1 npm install -g aws-cdk
2 cdk --version
```

Basic Syntax (TypeScript Example) [↗](#)

```
1 import * as cdk from 'aws-cdk-lib';
2 import { Instance, InstanceType, MachineImage } from 'aws-cdk-lib/aws-ec2';
3
4 const app = new cdk.App();
5 const stack = new cdk.Stack(app, 'MyStack');
6
7 new Instance(stack, 'MyInstance', {
8   instanceType: InstanceType.of('t2', 'micro'),
9   machineImage: MachineImage.latestAmazonLinux(),
```

```
10 });
```

Creating and Deploying Stacks [↗](#)

```
1 cdk init app --language=typescript
2 cdk synth
3 cdk deploy
4 cdk destroy
```

Version Control [↗](#)

Store CDK code (TypeScript, Python, etc.) in your Git repo.

Testing Methods [↗](#)

- **cdk synth**: Validates the CloudFormation output.
- **Jest (TypeScript) or PyTest (Python)**: Unit test constructs.
- **cdk diff**: Shows difference between deployed stack and local code.

Conclusion [↗](#)

IaC enables repeatable, auditable, and efficient infrastructure deployment.

- **Terraform**: Best cross-platform, declarative tool.
- **CloudFormation**: Native AWS declarative option.
- **CDK**: Code-first approach to AWS infrastructure.

Learning all three prepares you to document and manage IaC for multiple environments with a modern DevOps mindset.