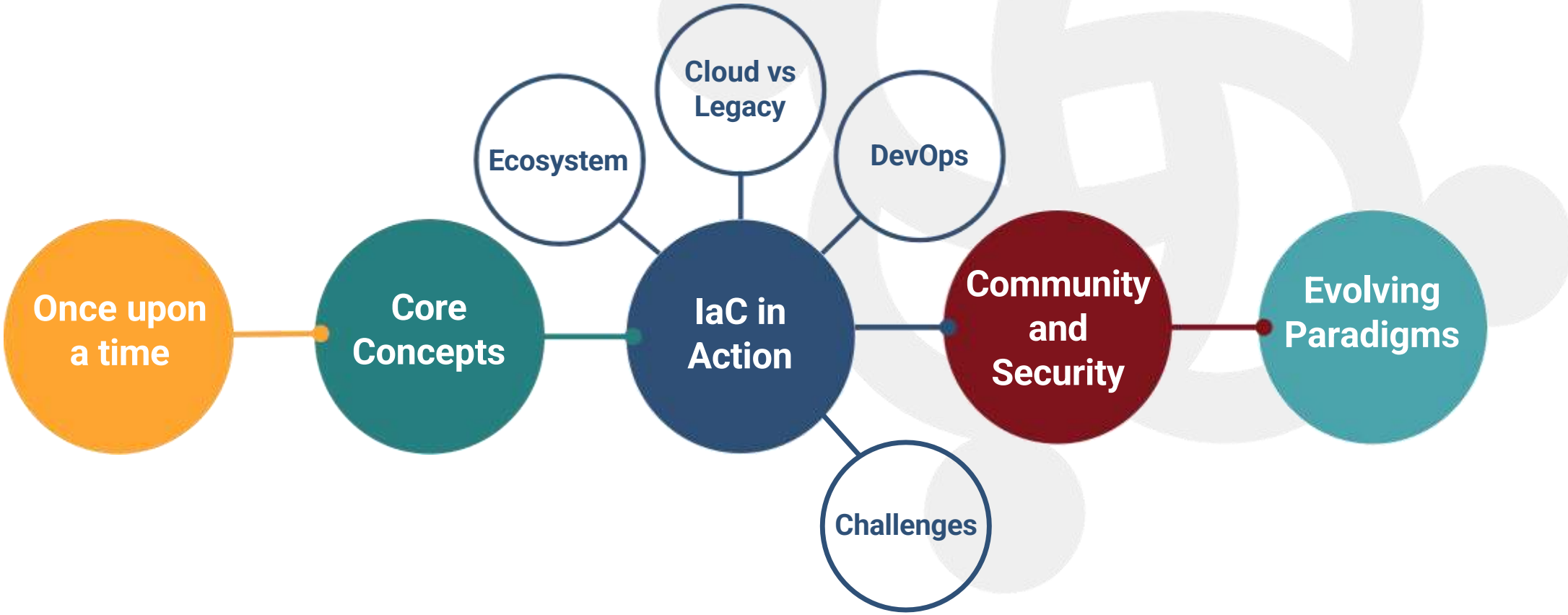
DevOps Workshop Series

Infrastructure as Code

Our agenda

You will learn:

- Why and how we got to Infrastructure as Code
- Fundamental concepts of IaC including tools available
- How does IaC look like in action, and how it relates to practices and use cases.
- The role of community in IaC and security implication
- GitOps and other evolving paradigms



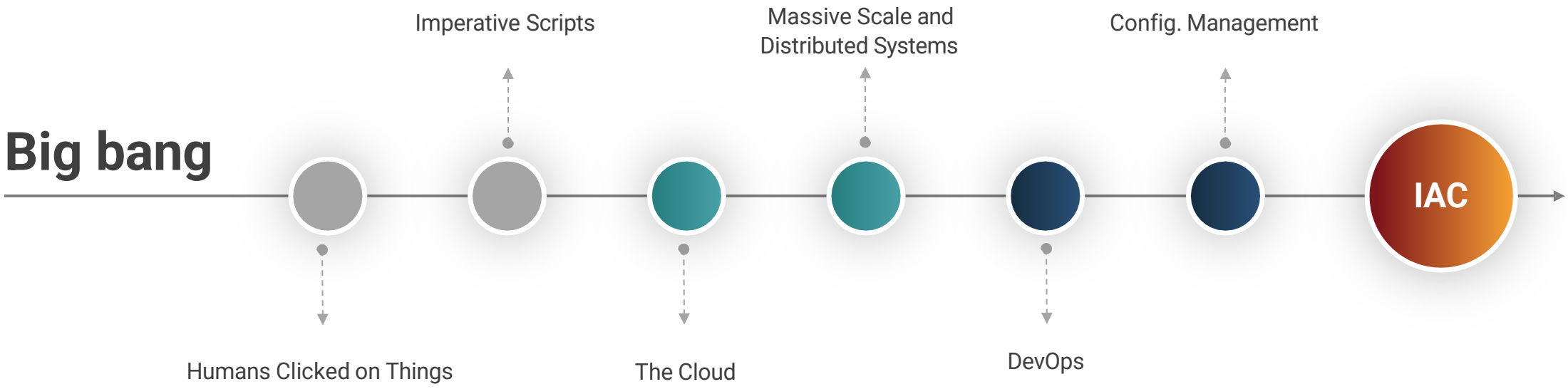
Infrastructure as Code

The practice of designing, provisioning, managing and operating infrastructure (networks, storage, compute clusters, load balancer, etc), by writing code and applying the same patterns and processes as used for other forms of source code (e.g. software)

ONCE UPON A TIME

THE STORY OF INFRASTRUCTURE AS CODE

A timeline to IaC



Infrastructure as Code fundamentals

- Objectives of IaC
- Source Control
- Idempotency
- Modularity

What is IaC looking to accomplish

CONSISTENCY

Environments are created from a declared blueprint reducing and making drift visible.

ONE TRUTH

Code represents the single source of truth. Repositories and version control provide history.

DE-RISK

Less opportunity for human error. Ability to inspect and secure changes before being applied.

TRACEABILITY

Visibility of change owners, timelines and testing procedures. Versioning and simplified reverts.

COLLABORATE

Work can be shared, peer reviewed and distributed. Common language and constructs.

EFFICIENCY

Speed to provision and recover. Reproducibility enables teams to manage huge scale.

Version control

The beauty of everything as code

- **Versions**: Snapshots over time of your infrastructure evolution.
- **Comparable**: Easily identifiable changes by comparing versions.
- **Collaborative**: Multiple team members (or teams altogether!) can collaborate on new versions of your infrastructure.
- **Lifecycle**: Develop, review, test, deploy, promote – apply SDLC to your infrastructure.

DevOps Institute

ADVANCING THE HUMANS OF DEVOPS

Idempotency

The result will be always the same, independent of the number of instances or iterations in which the process is executed.

Idempotency

```
resource "random_string" "random" {
    length = 10
}

resource "aws_instance" "not_idempotent" {
  ami           = "am-123456"
  instance_type = "t2.micro"
  name          = ${random_string.random.result}
}
```

```
resource "aws_instance" "idempotent" {
  ami           = "am-123456"
  instance_type = "t2.micro"
  name          = "not_random_name"
}
```

Modularity

Reusability and extensibility

Modules declare specific components, can be reused and extended. Infrastructure is not monolithic, components are required and consumed by services, there's a clear view unto service to infra dependencies.

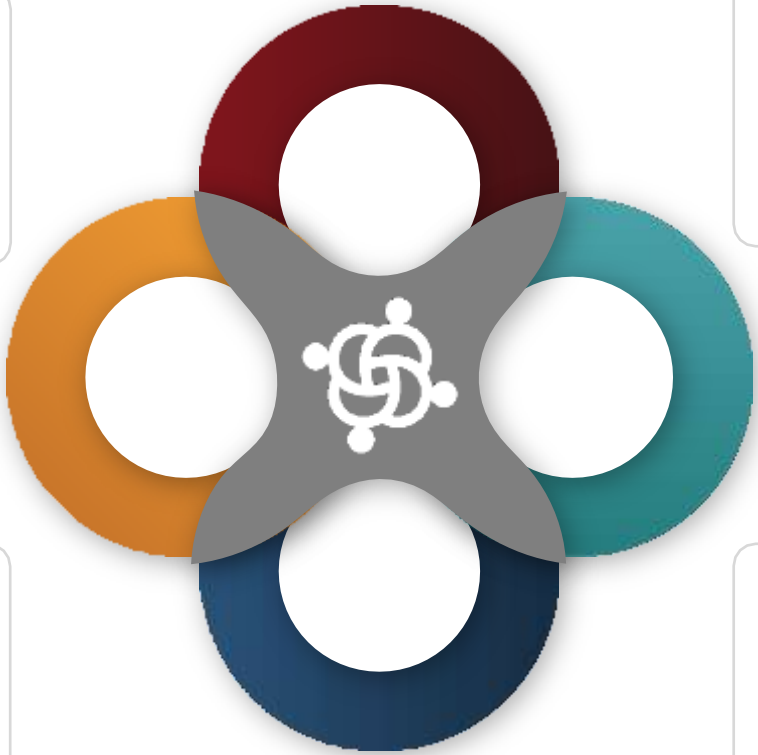
CI/CD of Infrastructure as Code

MULTISTAGE

Your infrastructure implementation and changes can be consistently applied and tested across the various stages of your application delivery

HYBRID ENVIRONMENTS

Whether on public cloud, private cloud, on premises or any combination, you can continuously integrate and deploy infrastructure.



TESTING AND VALIDATION

Automated and continuous testing becomes available for continuous validation of stability and reliability of infrastructure changes.

AUTOMATED

Human intervention becomes dramatically reduced in the integration, delivery and operation of infrastructure.

IaC IN ACTION

How does Infrastructure as Code apply and operate

Ecosystem

Growing number of tools, implementation paradigms and community contributed modules.

Ansible

Community

Ansible Galaxy

- Built with Python, uses Playbooks of tasks and modules to build and configure infrastructure and resources.
- Uses YAML and JSON for playbook declaration.
- Mostly used for Push operations over SSH, although a Pull implementation model from Git is also possible.
- Idempotency must be carefully implemented when using some modules.

Chef

Community

Chef Supermarket

- Built with Ruby, uses a custom DSL (very Rubyesque) to program recipes and cookbooks.
- Ruby blocks can be embedded in recipes which provides interesting possibilities.
- Client/Server model use primarily, though it can also run standalone as “chef-solo”.
- More imperative and programming oriented.

Puppet

Community

Puppet Forge

Custom declarative language, requires limited programming experience (also supports a Ruby DSL).

Client/Server Architecture, agent (client) pulls changes from a central server.

Model driven and relatively leans to configuration management.

Terraform

Community

Terraform Registry

Very much focused on Infrastructure as Code provisioning, not configuration management

Declarative, using HCL, a custom declarative syntax (can optionally use JSON)

Users use providers and declare resources, data sources and other primitives to define a desired state.

Providers abstract access to Cloud and other APIs.

AWS CloudFormation

Specific to AWS (although you can program custom resources that can be used, with some effort, to manage/query external resources).

Uses YAML and JSON to declare Stack templates.

Tightly integrated with AWS, clear visibility of resources and nested stacks, and easy IAM centric controls.

Pulumi

Provides libraries and SDKs for a variety of languages, you can use existing development environments to program your infrastructure.

Interesting combination of imperative and declarative approach, Pulumi programs are used to declare infrastructure.

Wide variety of providers outside of clouds: Network, Monitoring, Databases, VCS, etc.

Offers free managed service for state management.

AWS CDK

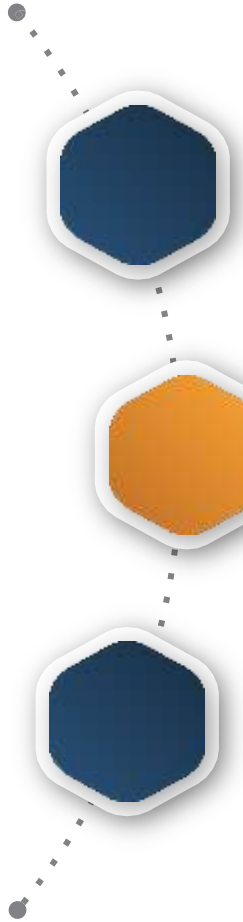
Currently support TypeScript, JavaScript, Python, Java and C # (Go is in developer preview).

AWS specific, uses CloudFormation under the hood.

Cdk tf provides simplified access to Terraform constructs, Cdk8s allows quick programming of Kubernetes primitives.

Infrastructure as Code: clouds, networks, legacy and in-between

Networks as Code



Software Defined Networks

Virtualization and container orchestration platforms enable the definition of virtual, software defined networks without the need to configure appliances.

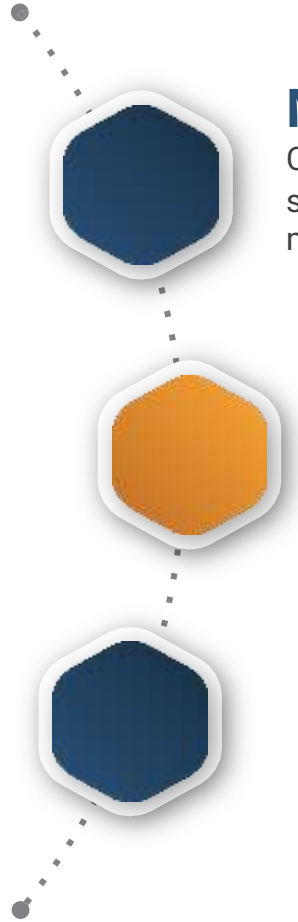
Support for Network Hardware

For network appliances, some tools (such as Ansible) allow for the declaration of a desired configuration state and the implementation of CI/CD of config changes.

Hybrid Environments

Choosing the right tool, you can provision, configure and operate hybrid environments, that may include different types of network architectures, devices and topologies.

Legacy and bare metal



Metal as a Service

Canonical's MaaS and Equinix Tinkerbell are examples of solutions that allow for programmatic and automated management of bare metal machines.

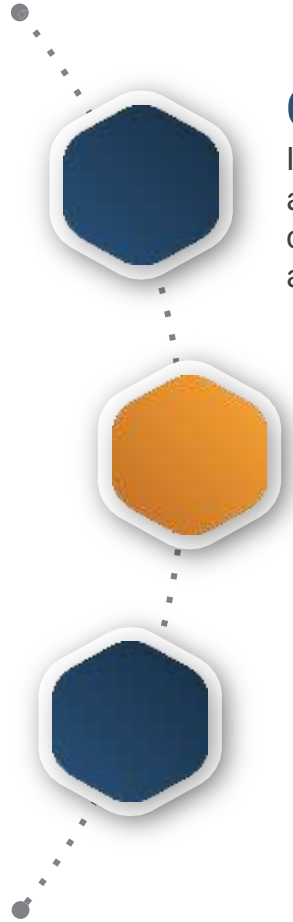
Legacy should not be overlooked

Legacy systems and infrastructure can be imported or codified to integrate into new IaC pipelines when new changes are applied.

Storage as well

Databases, network file systems, object storage and basically any other storage repository should also be declared and configured using infrastructure as code.

Cloud and world of APIs



Cloud Native

IaC simplifies the implementation of cloud native architectures, by enabling automation, abstracting the creation of managed resources, and allowing for iterative and ongoing re-architecting.

Massive Scale

Managing configuration and provisioning infrastructure at the massive and growing scale of current day solutions would be impossible for human operators to perform by hand.

Give me all your APIs

The clouds and a new breed of managed service providers have accelerated IaC growth by enabling API based access and control to most of their services and features.

Infrastructure as Code and its role in DevOps

IaC and DevOps core tenets

Infrastructure as Code is a critical practice to adopt in DevOps oriented organizations.

It aligns with the core tenets:

- Fast feedback cycles
- Shared ownership
- Everything automated

Challenges

IaC is not without barriers to overcome and problems to solve.

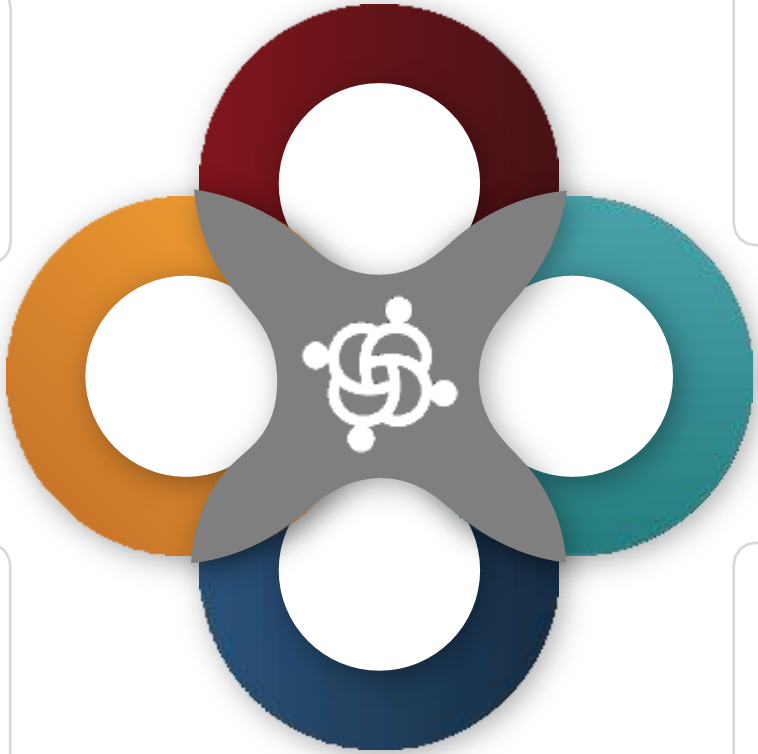
Challenges to overcome

EXISTING INFRASTRUCTURE

It's not that it can't be created and managed as code, it's that it existed already and was created manually.

I'M NOT USED TO THIS!

Sysadmins, Network Engineers, DBA and many roles will need to learn new skills and adopt a new way of doing things



A LEARNING CURVE

It will take getting used to and starting from code will make things harder for some, before making them easier.

PICK A TOOL, ANY TOOL

We saw a handful of tools, and that's just the tip of the iceberg, which one to pick?

Community and security

Great power, great responsibility

Why community and security in the same section?

Community contribution provides simplified and convenient means to access infrastructure blueprints and reusable modules, but there is a risk that software development also faces vulnerabilities out of copy-paste infrastructure.

GitOps and evolving paradigms

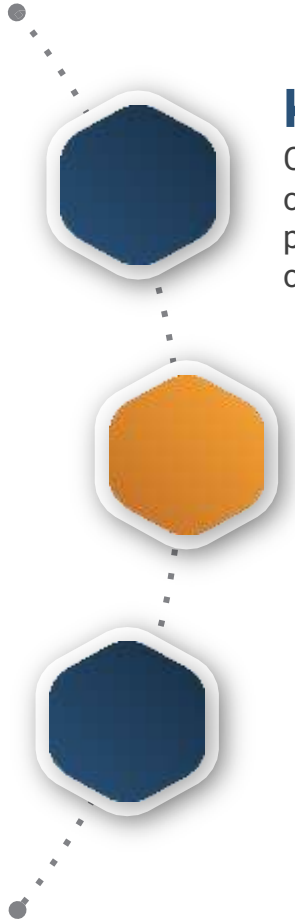
New technologies and better ways in which to apply existing tools are continuously evolving.

GitOps – declarative continuous reconciliation

GitOps applies to infrastructure as code, but doesn't stop there, it looks to consolidate the ways by which we declare and deploy infrastructure, with those that we use to declare and deploy our software solutions.

- Desired state declared in code
- Desired state is versioned and immutable
- Software agents continuously reconcile
- You only operate systems using these principles

Evolving paradigms



Kubernetes

Containers and container orchestrators have dramatically changed the landscape of infrastructure as code, providing an API driven, declarative abstraction away from compute

Infrastructure as Software

The impact of further blurred lines between team roles can not be understated,

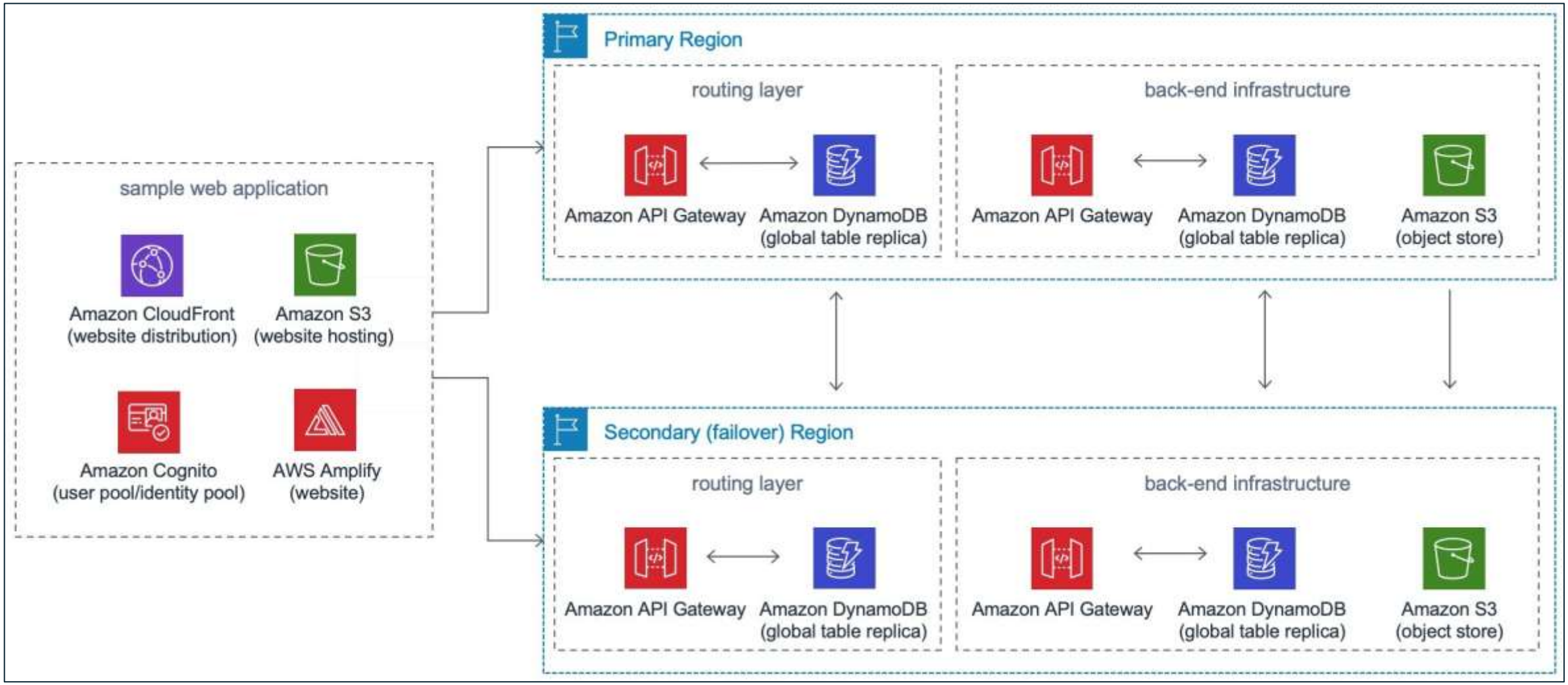
Serverless

How does infrastructure as code look like when there's no infrastructure?

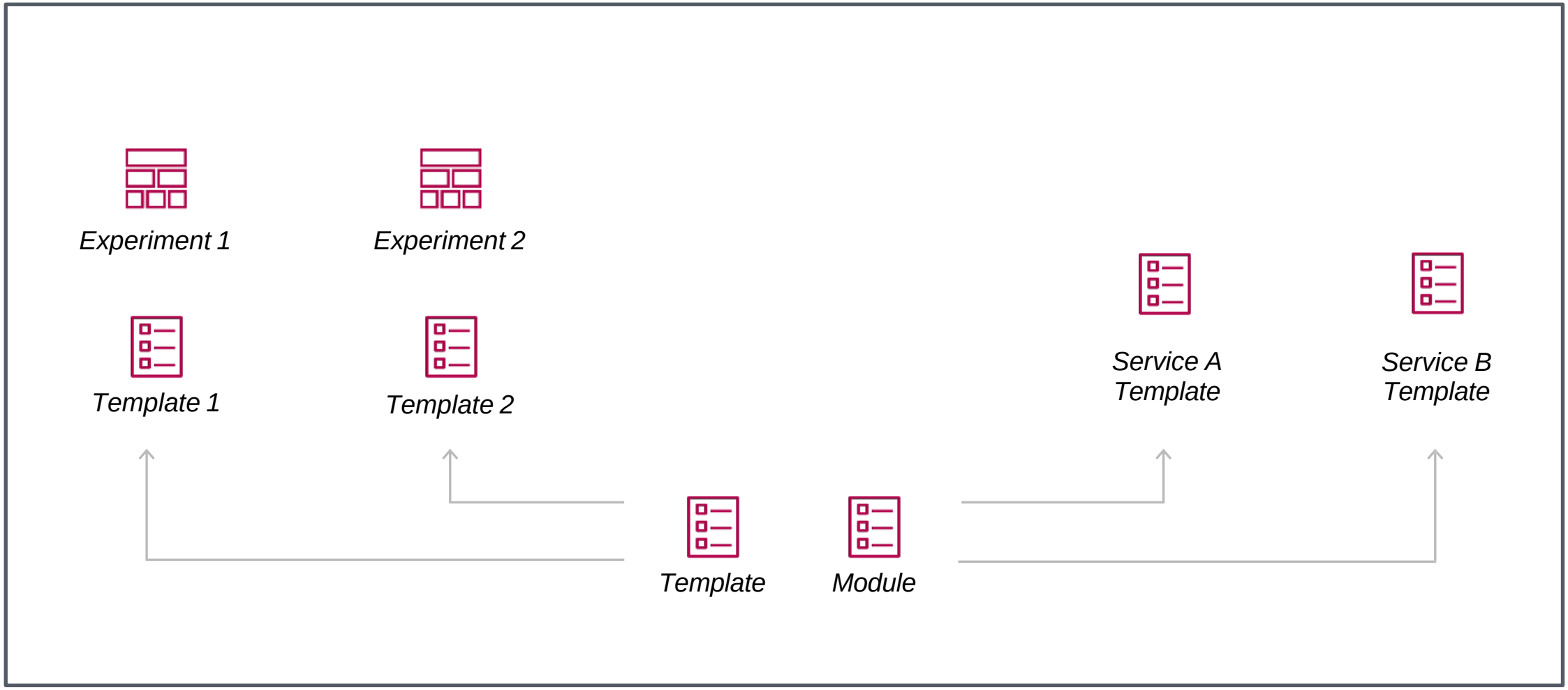
Agenda

- *Speed of deployment*
- *Experiment easy and more*
- *IaC on AWS*
- *AWS CloudFormation*
- *AWS CDK*

Speed of deployment



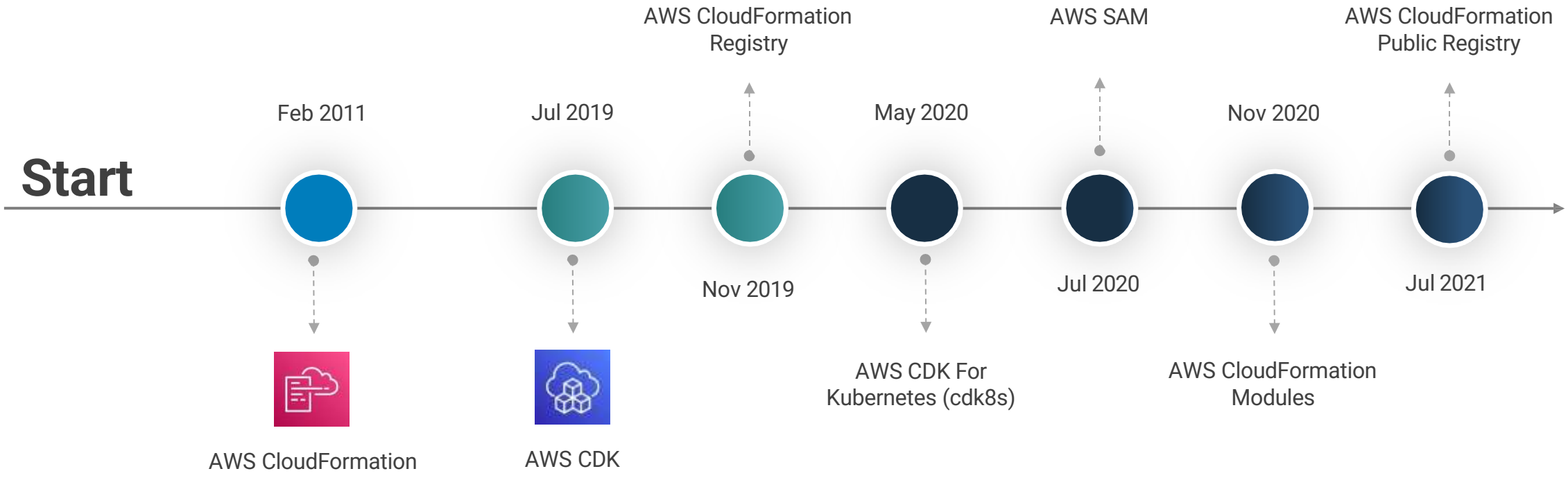
Experiment easy and more



Infrastructure as Code (IaC) on AWS

AWS DevOps-focused way of creating and maintaining infrastructure

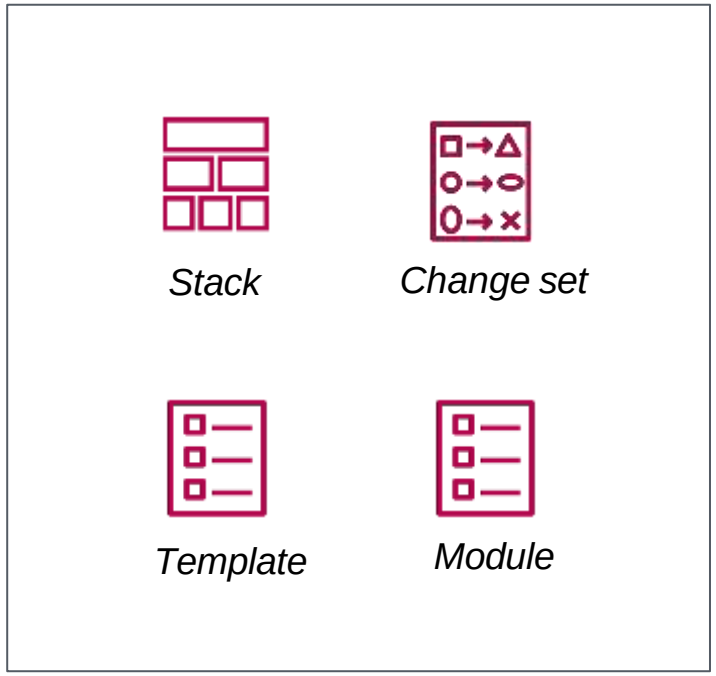
IaC on AWS



AWS CloudFormation

Reliability, reusability, extensibility

- Dependency management
- Rollback to previous state
- Cross-account and cross-region management
- Extensibility with CloudFormation Public Registry



AWS SAM (Serverless Application Model)

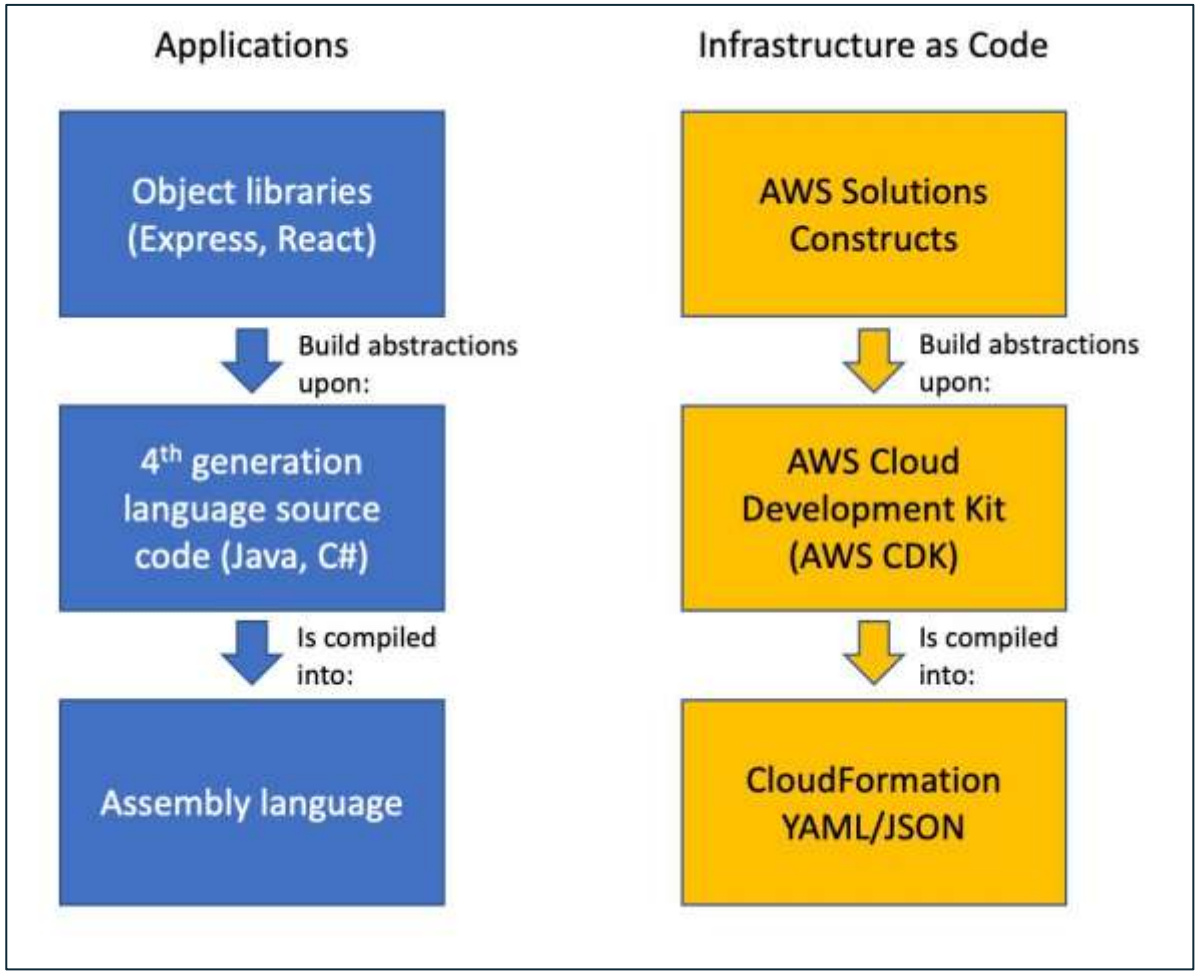
Serverless simplicity

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
Resources:
  LambdaFunctionOverHttps:
    Type: AWS::Serverless::Function
    Properties:
      Handler: index.handler
      Runtime: nodejs12.x
      Policies: AmazonDynamoDBFullAccess
      Events:
        HttpPost:
          Type: Api
          Properties:
            Path: '/DynamoDBOperations/DynamoDBManager'
            Method: post
```

AWS CDK (Cloud Development Kit)

Comfort for developers

- *TypeScript, Java, .NET, Python*
- *Leveraging your IDE*
- *Utilizing CloudFormation*
- *AWS Solution Constructs*

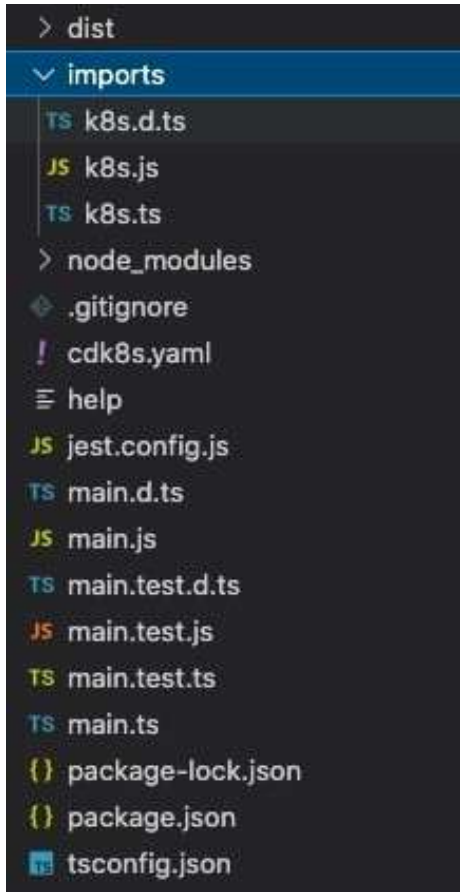


AWS CDK for Kubernetes (cdk8s)

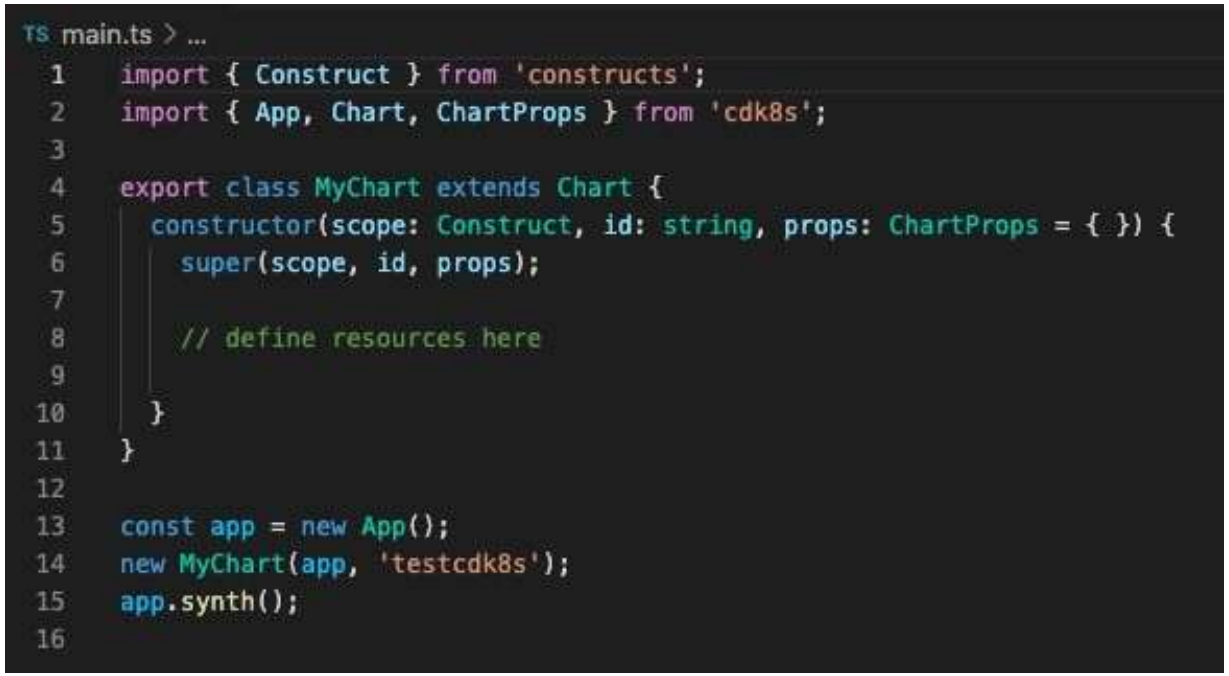
How CDK solves challenges with AWS CloudFormation for Kubernetes

- *Works for any cluster*
- *Imperative approach to declarative state*
- *Use any Kubernetes API version and custom resources*

cdk8s – diving deep



Pic 1: Structure of a new project



Pic 2: Main app file with a single empty chart

cdk8s –Kubernetes API objects

Pic 1: Kubernetes API imported objects

```
main.ts > MyChart > constructor > spec
1   import { Construct } from 'constructs';
2   import { App, Chart, ChartProps } from 'cdk8s';
3
4   // imported constructs
5   import { KubeDeployment, KubeService, IntOrString } from './imports/k8s';
6
7
8   export class MyChart extends Chart {
9     constructor(scope: Construct, id: string, props: ChartProps = { }) {
10      super(scope, id, props);
11
```

Pic 2: Import of Kubernetes API objects to use inside the file

```
12    const label = { app: 'hello-k8s' };
13
14    new KubeService(this, 'service', {
15      spec: {
16        type: 'LoadBalancer',
17        ports: [ { port: 80, targetPort: IntOrString.fromNumber(8080) } ],
18        selector: label
19      }
20    });
```

Pic 3: KubeService – we use Kubernetes API object to define our service at port 8080

```
22    new KubeDeployment(this, 'deployment', {
23      spec: {
24        replicas: 2,
25        selector: {
26          matchLabels: label
27        },
28        template: {
29          metadata: { labels: label },
30          spec: {
31            containers: [
32              {
33                name: 'hello-kubernetes',
34                image: 'paulbouwer/hello-kubernetes:1.7',
35                ports: [ { containerPort: 8080 } ]
36              }
```

Pic 3: KubeDeployment – we use Kubernetes API object to define our deployment with 2 replicas from specific image

cdk8s – YAML generated

dist > ! hello.k8s.yaml

```yaml
apiVersion: v1
kind: Service
metadata:
  name: hello-service-c8c17160
spec:
  ports:
    - port: 80
      targetPort: 8080
  selector:
    app: hello-k8s
  type: LoadBalancer
---
```

Pic 1: Our service in the generated YAML

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: hello-deployment-c8c7fda7
spec:
  replicas: 2
  selector:
    matchLabels:
      app: hello-k8s
  template:
    metadata:
      labels:
        app: hello-k8s
    spec:
      containers:
        - image: paulbouwer/hello-kubernetes:1.7
          name: hello-kubernetes
          ports:
            - containerPort: 8080
```

Pic 2: Our deployment in the generated YAML

Enhance your IaC practice with AWS Partners

AWS CloudFormation

Amazon Elastic Compute Cloud (Amazon EC2)

Amazon Simple Storage Service (Amazon S3)

Amazon DynamoDB

Amazon Simple Notification Service (Amazon SNS)

AWS Cloud API

HashiCorp Terraform

puppet

Quali

TREND MICRO

CORESTACK

CHEF Progress

Available in aws marketplace