aws | pulumi

Implementing cloud native infrastructure across your organization

with AWS and Pulumi

Table of contents

The challenge of delivering cloud infrastructure

With the popularity of cloud-first development continuing to grow, organizations need to rethink and redefine their cloud strategies, using a consistent approach to deliver cloud native applications and infrastructure. In doing so, organizations must consider how their tools will change and how their software delivery methods will evolve. Unifying cloud development methodologies and DevOps is a crucial component to fully embracing the cloud, but comes with three distinct challenges: keeping up with the pace of change, shifting to infrastructure in motion, and moving away from domain-specific languages (DSLs).

The cloud is no longer a new concept, but it has undergone distinct evolutions that has shifted the way developers think about developing and deploying cloud applications.

1. **CLOUD-BASED ON-DEMAND VIRTUAL MACHINES:**
   Infrastructure planning shifted from CapEx to OpEx focused, with the ability to provision cloud-based virtual machines on demand.

2. **CONTAINERS AND KUBERNETES:**
   Moving past VMs, the current wave of cloud computing is offering new strategies for improving cloud native approaches including infrastructure, abstraction, OS isolation, and a model for highly distributed applications.

3. **SERVERLESS AND MANAGED SERVICES:**
   The upcoming wave of cloud computing is offering a stateless mesh of on-demand and infinitely scalable services, with options for arbitrary code execution.

The rapid pace of evolution of the cloud, combined with the shift to ephemeral infrastructure, and the connection of application code and infrastructure code, demands a different view of cloud Development and DevOps.

The challenge of delivering cloud infrastructure

When infrastructure was at rest, configuration management and convergence-based infrastructure as code tools were the preferred way of managing the typical infrastructure provisioning lifecycle. But with the rise of cloud native development processes, infrastructure has increasingly become 'in motion', where provisioning is tied to individual applications and infrastructure doesn't warrant continued configuration management. To support the shift towards increasingly codified provisioning of short-lived infrastructure, organizations need comprehensive tools that will connect application code and infrastructure code in a logical and expressive way. This is crucial to development and operations teams as application code increasingly defines the infrastructure requirements for specific cloud applications.

The final challenge organizations are often faced with in adapting their cloud-first strategy is eliminating the use of DSLs. In the case of infrastructure as code, YAML is one of the more popular DSLs. Unfortunately, it lacks the features and tooling necessary to make it effective, and encourages poor programming practices. Instead, YAML should be considered a bytecode, intended to be an output format of another program. DevOps teams have become familiar with DSL, but are now overwhelmed by the sheer number of DSLs required for cloud-based applications. To reduce the impact DSLs have on development and DevOps teams, organizations need to evaluate mature tools that will provide the consistent and consolidated experience development teams expect.

Considering these challenges, organizations need a comprehensive solution that provides the right set of tools, libraries, runtime, and service developers and DevOps teams require, in the programming languages they prefer.

Pulumi on Amazon Web Services (AWS)

As an AWS Partner Network (APN) Advanced Technology Partner, Pulumi provides a cloud native programming model for AWS to create containers, serverless functions, and infrastructure, enabling the delivery of cloud native infrastructure as code using real programming languages.

REAL CODE

Pulumi allows your developers to leverage their favorite languages and tools for provisioning cloud infrastructure including code completion, error checking, versioning, and integrated development environment (IDE) support.

EPHEMERAL INFRASTRUCTURE

Pulumi helps maintain successful infrastructure deployments by efficiently building, updating, and destroying cloud resources as needed.

REUSABLE COMPONENTS

Pulumi enables you to build repeatable practices through versioned code packages and deploy them across your team with ease.

What Pulumi does

Pulumi has a developer-centric view of the world, and strives to bring consistency and efficiency to the developer experience.

Pulumi's cloud native development platform provides a single, consistent model for infrastructure provisioning, using familiar and powerful general-purpose languages, to provide the necessary capabilities to meet continuous delivery requirements for modern cloud applications.

PULUMI HELPS YOU:

- Use familiar programming languages, such as JavaScript and Python, to provide familiar and powerful programming concepts to infrastructure as code challenges.
- Build a library of code packages to enhance efficiency when implementing standard policies, network best practices, and more.
- Adopt best practices for becoming a deployment-focused organization, rather than configuration-focused, to continuously deliver new cloud native infrastructure.

Pulumi provides a single, consistent platform for the delivery of cloud resources of any kind.

Features

For Developers

FAMILIAR LANGUAGES

Define cloud application and services as code in languages that you're familiar with.

PROGRAMMING MODEL

Easily deliver container, AWS Lambda, and data-based services with a programming model unique to your organization.

REUSABLE PACKAGES

Reuse code packages to easily design, share, and deploy cloud resources, patterns, and stacks.

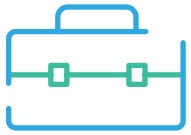
For DevOps

DEPLOY CONTINUOUSLY

Deploy your apps and infrastructure consistently and reliably across all of your cloud native environments.

FAMILIAR WORKFLOW

Easily manage and integrate your containers into existing delivery processes.



ENTERPRISE GRADE

Leverage preferred tools, languages, and practices to deliver new applications and enterprise-grade workflows.

How it works

1 CODE

Code with chosen language using Pulumi packages, custom components, and chosen languages and IDEs.

```
$> pulumi new
```

2 DEPLOY

Deploy to AWS, creating a full initial stack.

```
$> pulumi update
```

3 MANAGE

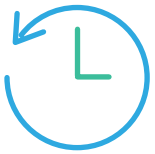
Manage with additional updates to create 'diffs' of existing stack and recreate the necessary resources.

```
$> pulumi up
```

EXAMPLE WORKFLOW WITH JAVASCRIPT/TYPESCRIPT

Benefits

Pulumi helps you get your code to AWS faster and more efficiently.

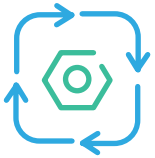


FASTER TIME TO MARKET

Pulumi removes redundancy, helping your organization deploy new infrastructure and applications, faster.

PATH TO SERVERLESS

Microservices enable easy and scalable websites, event streaming, and processing across your entire cloud environment.



CI/CD CAPABILITIES

By connecting application and infrastructure code to preferred tools and practices, Pulumi provides the CI/CD functions needed for cloud native applications.

COST SAVINGS

Reducing lines of code allows organizations to run their code locally and save on the resources needed to maintain it.

Customer success story: Learning Machine

Learning Machine Improved time to ship by implementing a container-based infrastructure with Pulumi.

CHALLENGES

Learning Machine's environment was supplemented by ad-hoc Bash scripts resulting in 25,000 lines of code that were difficult to understand and maintain.

OPPORTUNITIES

1. Pulumi offered cloud native infrastructure as code.
2. The Learning Machine team was able to leverage programming languages they were most familiar with
3. Pulumi was able to dedicate resources to advise, plan, and provide

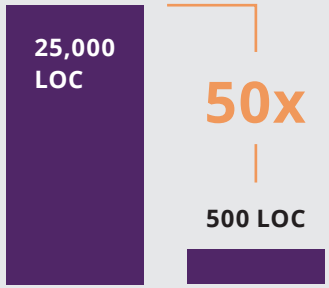
PROCESS

- Pulumi helped Learning Machine modify their 25,000 lines of code to a manageable 500 lines
- Learning Machine leveraged Amazon Elastic Container Service (Amazon ECS) to run and scale their containerized applications
- AWS Fargate was introduced to Learning Machine's new environment as the compute engine for Amazon ECS

OUTCOMES

Learning Machine gained:

- Trimmed code down to 500 lines of JavaScript
- Reduced customer provisioning time
- Significant cost savings



25,000 Lines of CloudFormation reduced to 500 Lines of Java script

New customer provisioning time reduced from 3 weeks to 1 hour

Getting started with AWS and Pulumi

Learn about the different ways you can program the cloud using Pulumi with these 12 bite-size code snippets:

INFRASTRUCTURE

- Declare cloud infrastructure using real language
- Make a reusable component out of your cloud infrastructure

SERVERLESS

- Go serverless without the YAML
- Capture state in your serverless functions, such as real AWS Lambdas
- Simple serverless cron jobs
- Run express-like serverless SPAs and REST APIs are near zero cost

CONTAINERS

- Deploy production containers without the fuss
- Use containers without Dockerfiles
- Invoke a long-running container as a task

GENERAL TIPS AND TRICKS

- Use code to avoid hard-coding configuration
- Use configuration to enable multi-instantiation and code reuse
- Give your components runtime APIs

```
// Deploy a prebuilt container image
// to AWS Fargate

import * as cloud from "@pulumi/cloud";

let nginx = new cloud.Service("nginx", {
        image: "nginx",
        ports: [{port: 80}],
        replicas: 2,
});

export let url = nginx.defaultEndpoint;

// Deploy a custom container image based on nginx
// to AWS Fargate

import * as cloud from "@pulumi/cloud";

let nginx = new cloud.Service("nginx", {
        build: ".",
        ports: [{port: 80}],
        replicas: 2,
});

export let url = nginx.defaultEndpoint
```

Do more with AWS and Pulumi

Getting started with Pulumi and Kubernetes

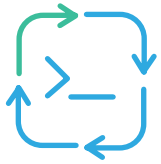
Learn how you can use Pulumi to build and deploy Kubernetes applications using cloud native infrastructure as code with 11 bite-size code snippets:

CONFIG AS REAL CODE

- Define Kubernetes applications in a real language
- More expressiveness, less boilerplate
- Inject envoy sidecars using abstraction
- Adopt existing Kubernetes YAML
- Programmatically deploy Helm Charts as code

MULTI-CLOUD INFRASTRUCTURE

- Declare cloud resources alongside Kubernetes resources
- Provision and use Kubernetes resources
- Build and deploy container images alongside configuration updates



SOFTWARE DELIVERY AS CODE

- Robust and repeatable deployments, with a notion of “done”
- Trigger cascading rollouts from dependent updates
- Staged application rollouts gated by Prometheus checks

```
// Deploy simple app to Kubernetes

import * as k8sjs from "./k8sjs";

let redisMaster = new k8sjs.ServiceDeployment("redis-master", {
       image: "k8s.gcr.io/redis:e2e",
       ports: [ 6379 ],
});

let redisSlave = new k8sjs.ServiceDeployment("redis-slave", {
       image: "gcr.io/google_samples/gb-redisslave:v1",
       ports: [ 6379 ],
});

let frontend = new k8sjs.ServiceDeployment("frontend", {
       replicas: 3
       image: "gcr.io/google_samples/gb-frontend:v4",
       ports: [ 80 ],
       loadBalancer: true,
});

export let frontendIp = frontend.ipAddress;
```

Do more with Kubernetes and Pulumi

Additional Resources

Pulumi works with your favorite language.

1 INSTALL PULUMI

- Setup
- Configure

2 TRY OUR TUTORIALS

- Quickstarts
- Tour

3 READ THE DOCS

- Reference
- Examples