Pulumi

Modern Infrastructure as Code. Open source, any cloud, any language.

What is Pulumi?

Pulumi is an open source infrastructure as code tool for creating, deploying, and managing cloud infrastructure. Pulumi works with traditional infrastructure like VMs, networks, and databases, in addition to modern architectures, including containers, Kubernetes clusters, and serverless functions.

Pulumi enables developers to write code in their favorite language, such as **TypeScript**, **JavaScript**, **Python**, **Go** and **.NET (C#, F#, VB)**. This enables modern approaches to cloud applications and infrastructure without needing to learn yet-another YAML or DSL dialect. This unlocks abstractions and reuse as well as access to your favorite IDEs, refactoring, and testing tools. Master one toolchain and set of frameworks, and go to any cloud — **AWS, Azure, GCP**, or **Kubernetes** — with ease.

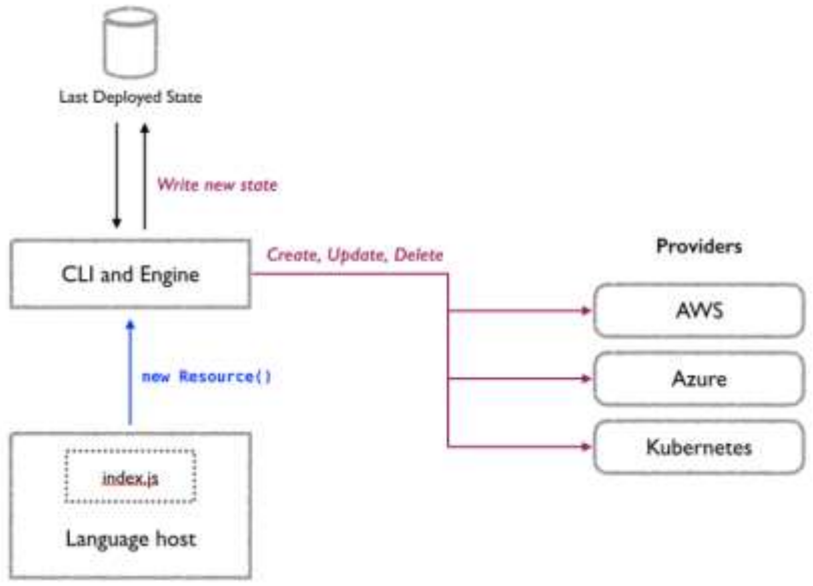
Who Develops It?

Pulumi is developed by venture backed startup in Seattle whose mission is to enable every person to harness the power of the cloud. First preview release was in 2018 and v1.0 was debuted only in September of 2019.

Pulumi founder and CEO is **Joe Duffy**, who has extensive open-source experience (he built the team at Microsoft that took .NET open source), so it’s no surprise that Pulumi, too, has a number of open-source components.

The service has the backing of **Madrona Venture Group** and **Tola Capital**, with Madrona’s S. Somasegar joining its board of directors.

How Pulumi Works?



Pulumi uses a **desired state model** for managing infrastructure. A Pulumi program is executed by a **language host** to compute a desired state for a stack’s infrastructure. The **deployment engine** compares this desired state with the stack’s current state and determines what resources need to be created, updated or deleted.

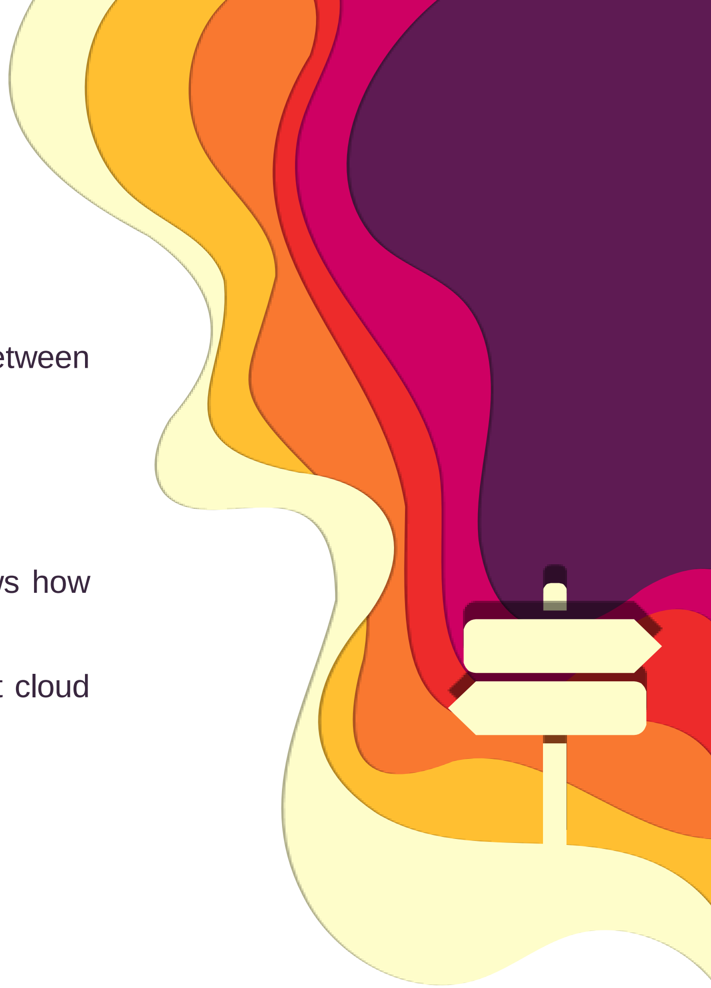
Pulumi Program Structure

Pulumi programs are structured as projects and stacks. The distinction between them is:

Program: a collection of files written in your chosen programming language

Project: a directory containing a program, with metadata, so Pulumi knows how to run it

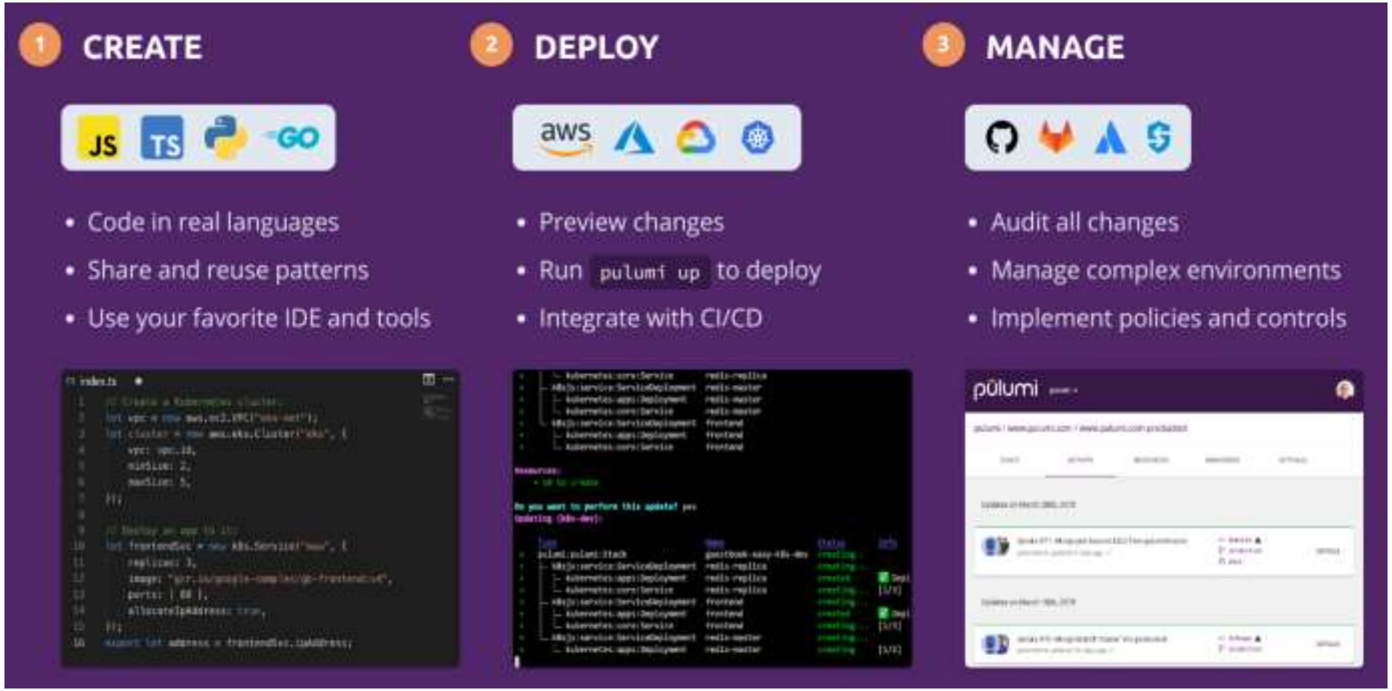
Stack: an instance of your project, each often corresponding to a different cloud environment



Code Structure

Pulumi.dev.yaml	Pulumi Stack definition
Pulumi.yaml	Pulumi Project config
__main__.py	Pulumi Program (Python)
some.py	Pulumi Program (could be imported in __main__.py)
requirements.txt	Dependency file

How To Launch This Code?



1 CREATE

- Code in real languages
- Share and reuse patterns
- Use your favorite IDE and tools

```
index.ts
// Create a Kubernetes cluster.
let vpc = new awsx.ec2.Vpc("eks-net");
let cluster = new aws.eks.Cluster("eks", {
    vpc: vpc.id,
    minSize: 2,
    maxSize: 5,
});

// Deploy an app to it:
let frontendSvc = new k8s.Service("svc", {
    replicas: 3,
    image: "gcr.io/google-samples/gb-frontend:v4",
    ports: [ 80 ],
    allocateIpAddress: true,
});
export let address = frontendSvc.ipAddress;
```

2 DEPLOY

- Preview changes
- Run `pulumi up` to deploy
- Integrate with CI/CD

```
 +    │  └─ kubernetes:core:Service          redis-replica
 +    ├─ k8sjs:service:ServiceDeployment     redis-master
 +    │  ├─ kubernetes:apps:Deployment       redis-master
 +    │  └─ kubernetes:core:Service          redis-master
 +    └─ k8sjs:service:ServiceDeployment     frontend
 +       ├─ kubernetes:apps:Deployment       frontend
 +       └─ kubernetes:core:Service          frontend

Resources:
    + 10 to create

Do you want to perform this update? yes
Updating (k8s-dev):

     Type                                    Name                     Status       Info
 +   pulumi:pulumi:Stack                     guestbook-easy-k8s-dev   creating...
 +   ├─ k8sjs:service:ServiceDeployment      redis-replica            creating...
 +   │  ├─ kubernetes:apps:Deployment        redis-replica            created      ✓ Depl
 +   │  └─ kubernetes:core:Service           redis-replica            creating...  [1/3]
 +   ├─ k8sjs:service:ServiceDeployment      frontend                 creating...
 +   │  ├─ kubernetes:apps:Deployment        frontend                 created      ✓ Depl
 +   │  └─ kubernetes:core:Service           frontend                 creating...  [1/3]
 +   └─ k8sjs:service:ServiceDeployment      redis-master             creating...
 +      └─ kubernetes:apps:Deployment        redis-master             creating     [1/3]
```

3 MANAGE

- Audit all changes
- Manage complex environments
- Implement policies and controls

pulumi

[illegible]

Setup Pulumi

```
$ curl -fsSL https://get.pulumi.com | sh
$ sudo vim /etc/environment
PATH="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/
usr/local/games:/home/user/.pulumi/bin"
$ source /etc/environment

$ pulumi version
v2.1.0

# apt install build-essential python3 python3-dev python3-venv

$ export AWS_ACCESS_KEY_ID=AKIA1234563J76A
$ export AWS_SECRET_ACCESS_KEY=/xLmpmdp1V3abcdefghklmnopabcdefg2nKRDKO
```

Example #1: S3 Bucket Creating

```
$ mkdir quickstart
$ cd quickstart

$ pulumi new aws-python
<answer on the questions>

$ ls
Pulumi.dev.yaml  Pulumi.yaml  __main__.py  __pycache__  requirements.txt

$ python3 -m venv venv
$ source venv/bin/activate
(venv) $ pip3 install -r requirements.txt
```

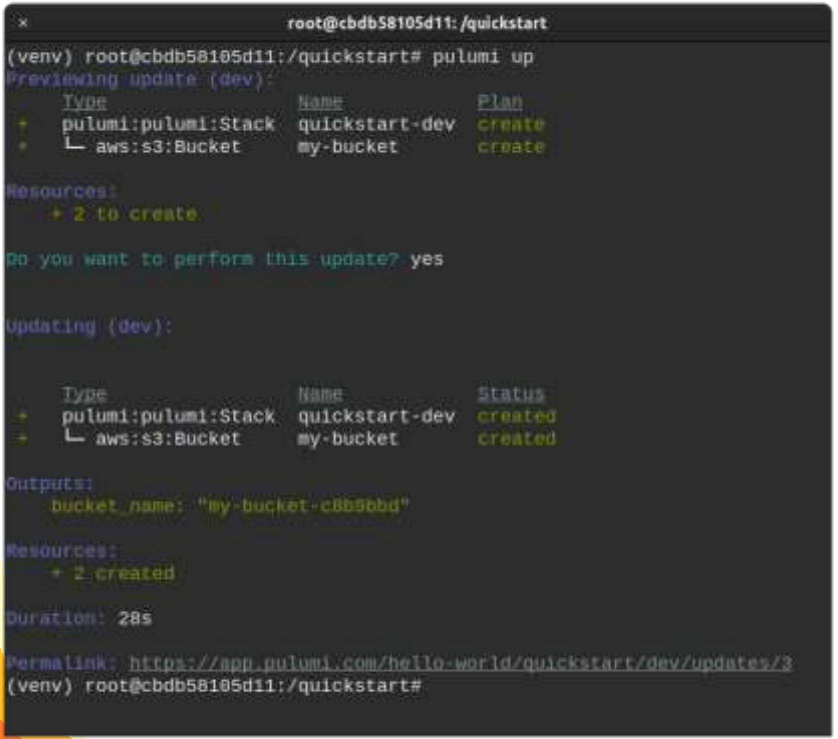
Example #1: S3 Bucket Creating

```
$ cat Pulumi.dev.yaml
config:
  aws:region: us-east-1
```

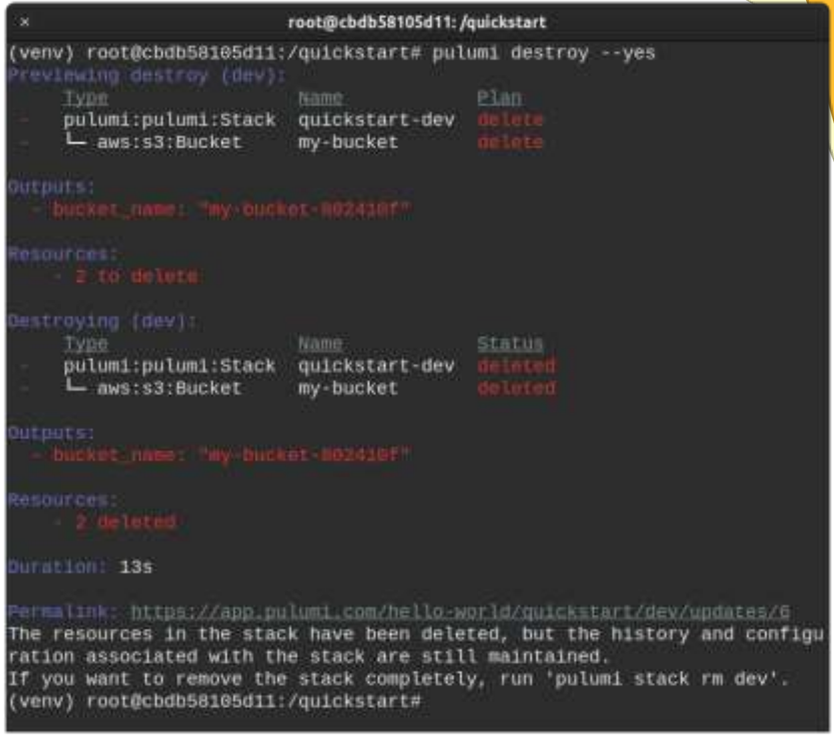
```
$ cat Pulumi.yaml
name: quickstart
runtime: python
description: A minimal AWS Python
Pulumi program
```

```
$ cat __main__.py
import pulumi
from pulumi_aws import s3

# Create an AWS resource (S3 Bucket)
bucket = s3.Bucket('my-bucket')

# Export the name of the bucket
pulumi.export('bucket_name',
bucket.id)
```

Example #1: S3 Bucket Creating

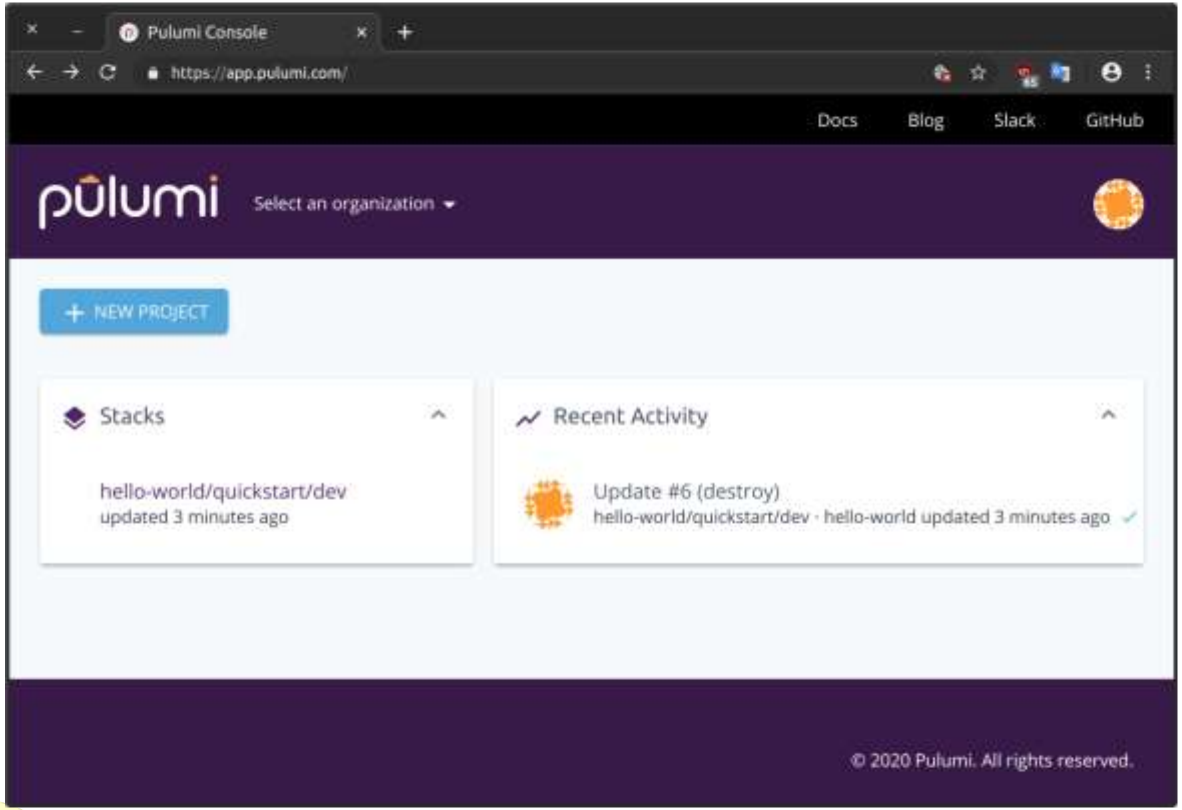


```
root@cbdb58105d11: /quickstart
(venv) root@cbdb58105d11:/quickstart# pulumi up
Previewing update (dev):
     Type                    Name             Plan
 +   pulumi:pulumi:Stack     quickstart-dev   create
 +   └─ aws:s3:Bucket        my-bucket        create

Resources:
    + 2 to create

Do you want to perform this update? yes


Updating (dev):


     Type                    Name             Status
 +   pulumi:pulumi:Stack     quickstart-dev   created
 +   └─ aws:s3:Bucket        my-bucket        created

Outputs:
    bucket_name: "my-bucket-c8b9bbd"

Resources:
    + 2 created

Duration: 28s

Permalink: https://app.pulumi.com/hello-world/quickstart/dev/updates/3
(venv) root@cbdb58105d11:/quickstart#
```



```
root@cbdb58105d11: /quickstart
(venv) root@cbdb58105d11:/quickstart# pulumi destroy --yes
Previewing destroy (dev):
     Type                    Name             Plan
 -   pulumi:pulumi:Stack     quickstart-dev   delete
 -   └─ aws:s3:Bucket        my-bucket        delete

Outputs:
  - bucket_name: "my-bucket-802410f"

Resources:
    - 2 to delete

Destroying (dev):
     Type                    Name             Status
 -   pulumi:pulumi:Stack     quickstart-dev   deleted
 -   └─ aws:s3:Bucket        my-bucket        deleted

Outputs:
  - bucket_name: "my-bucket-802410f"

Resources:
    - 2 deleted

Duration: 13s

Permalink: https://app.pulumi.com/hello-world/quickstart/dev/updates/6
The resources in the stack have been deleted, but the history and configu
ration associated with the stack are still maintained.
If you want to remove the stack completely, run 'pulumi stack rm dev'.
(venv) root@cbdb58105d11:/quickstart#
```

Example #1: S3 Bucket Creating



hello-world/quickstart/dev
updated 3 minutes ago

Update #6 (destroy)
hello-world/quickstart/dev · hello-world updated 3 minutes ago

© 2020 Pulumi. All rights reserved.

Example #2: Networks and Fargate

3 public subnets and 3 private subnets across 3 AZs in us-east-1 region

ECS Fargate cluster on top of these networks

https://github.com/ipeacocks/pulumi-aws-example

I like it. When to migrate?

It’s really quite promising product and I wish it good luck…but take into consideration next facts:

* it’s very **young product**: first public version was presented in 2018 only!

* **not enough documentation.** Good examples your can find in GitHub repositories but very often on JavaScript/TypeScript only. It’s really hard to find how to use Pulumi or how to organize code/dir structure with it.

* **no locks** with 3rd-party storage for states. Those locks which help you to avoid running the same code at the same time. But maybe it’s not a huge problem because it’s a general purpose language and you can use mature libraries for doing that by yourself.

* as Pulumi uses general purpose languages your teammates **could create a real monster** even worse than with Terraform! So you definitely need to have rules how code needs to look in your team.

* your team **needs to know any normal language** and its structures. But its learning definitely worth time investment.

Example #2: Networks and Fargate

https://medium.com/@_ipeacocks/pulumi-vpc-and-fargate-configuration-fd60f1b053ea

https://www.pulumi.com/docs/get-started/aws/install-pulumi/

https://www.reddit.com/r/devops/comments/bcdwsn/pulumi/

https://www.pulumi.com/docs/intro/concepts/programming-model/

https://techcrunch.com/2018/06/18/pulumi-wants-to-let-you-manage-your-infrastructure-with-code/

http://joeduffyblog.com/2018/06/18/hello-pulumi/

https://thenewstack.io/pulumi-uses-real-programming-languages-to-enforce-cloud-best-practices/

https://medium.com/@kscloud/how-to-program-infrastructure-with-pulumi-part-1-a47d5edb913f

https://medium.com/@kscloud/how-to-program-infrastructure-with-pulumi-part-2-3d7d64e69146

https://itnext.io/infrastructure-as-code-using-pulumi-to-provision-and-bootstrap-a-gcp-instance-318f06c61a03

Thank you!

Any Questions?