



# Pulumi CloudNative Prague

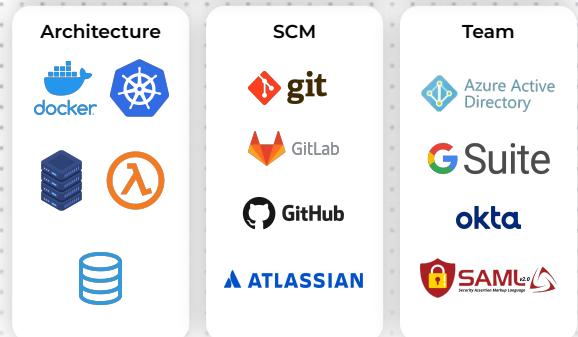
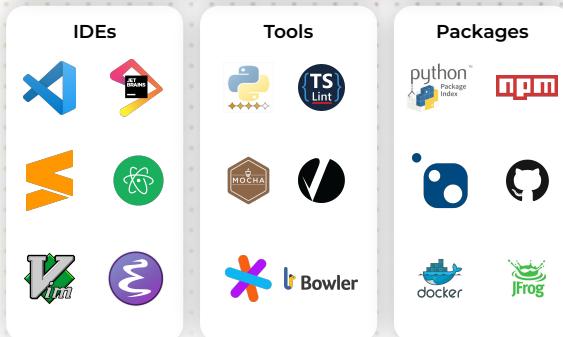
Ringo De Smet  
Customer Experience Architect

19 June 2023



# Introduction

# Your Cloud, Your Language, Your Way



Build

Deploy

Manage

# Multi-Cloud Infrastructure as Code

BUILD, DEPLOY, AND MANAGE ANYWHERE WITH A STANDARD WORKFLOW

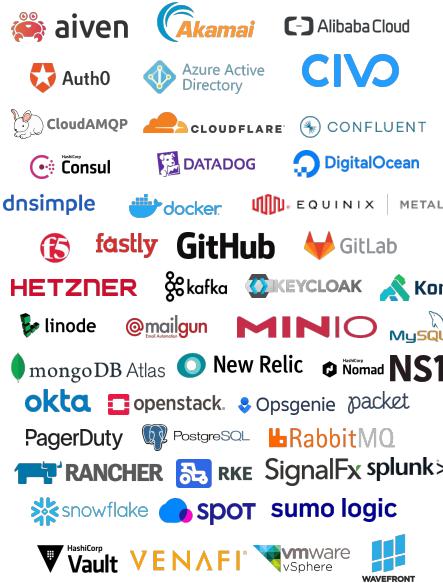
## Core Features:

- Any Cloud, Any Language
- State Management
- Secrets Management
- Guaranteed Preview Plans
- CI/CD Integrations
- Webhooks
- REST API
- Automation API
- Dashboards and Reports

## Clouds



## Infra Providers

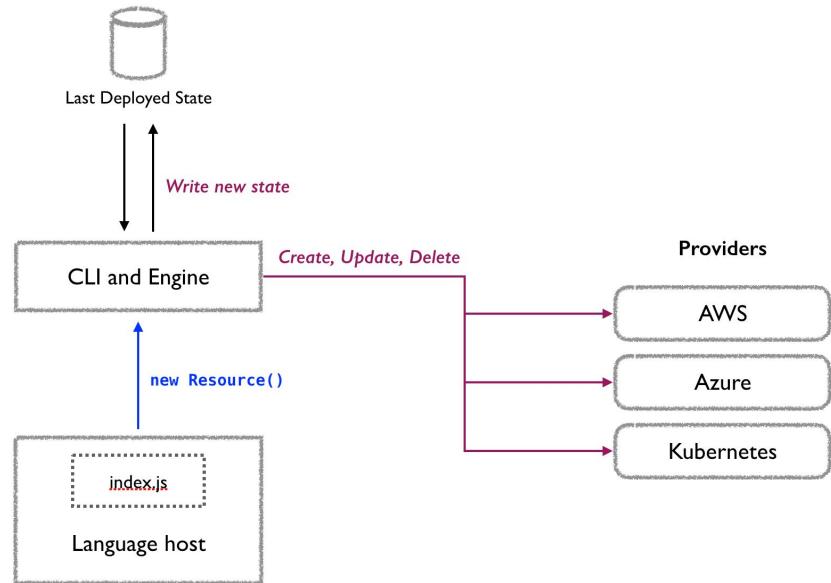


## Cloud Native



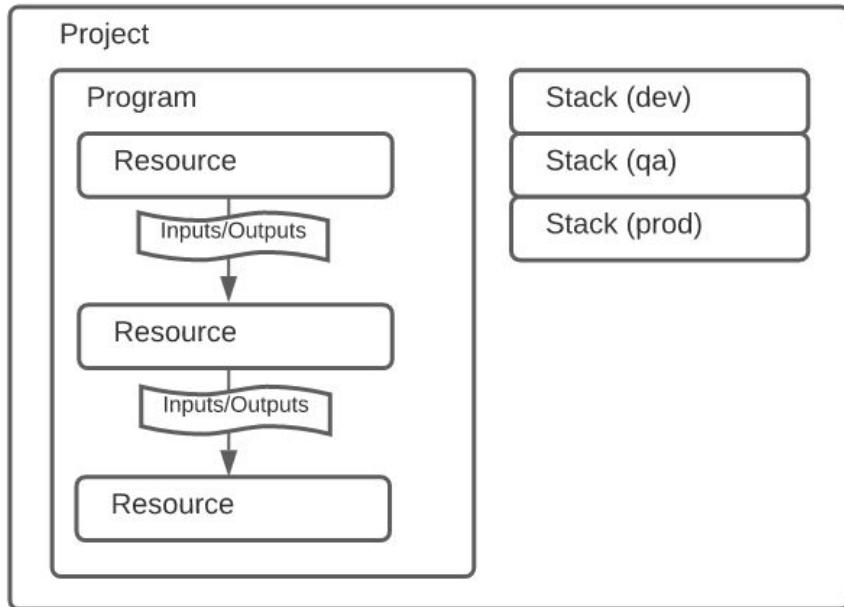
# Pulumi - High Level View

- Pulumi Architecture
  - Language Host
  - CLI & Engine
  - Providers
  - State Backend
    - Pulumi Service
    - Cloud Storage (S3, ...)



# Pulumi - High Level View

- Automate your infrastructure setup with 1 or more Pulumi projects
- A project contains
  - a program: this is your “template” from which you can create stacks
  - the stacks: each stack is an “instance” of the program
- One project can use info from another project as input



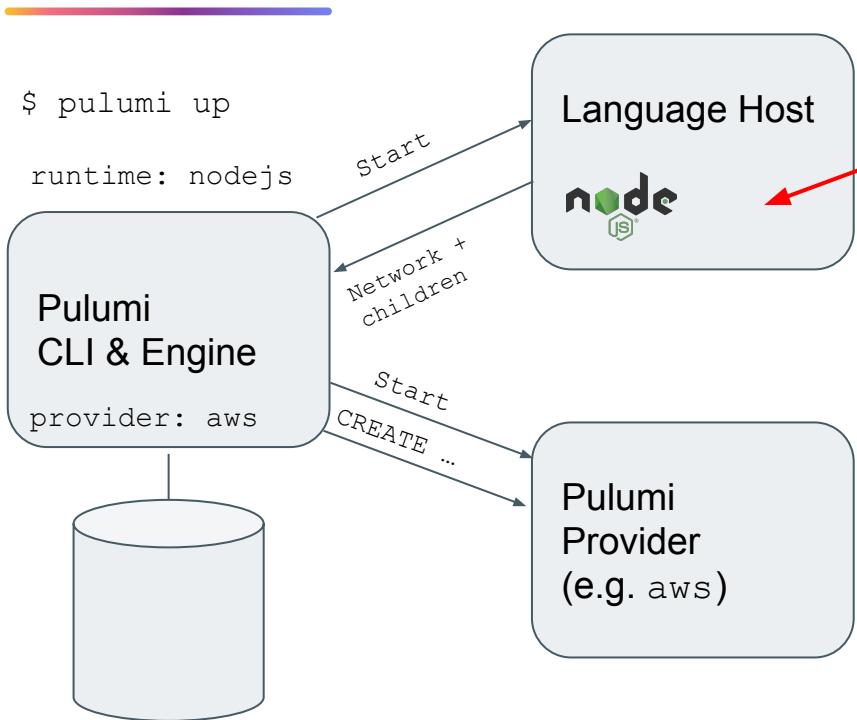


# Demo: Simple Program



# What about a real usecase?

# Pulumi - ComponentResource



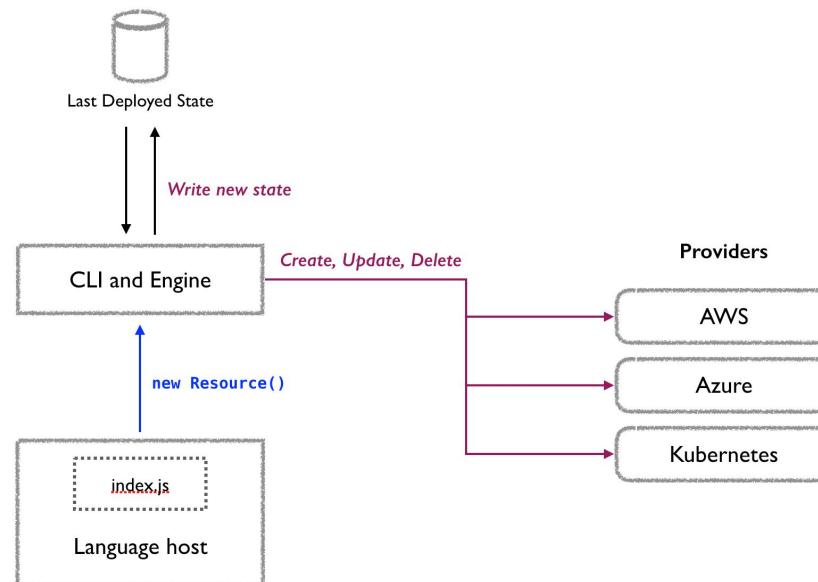
```
export interface NetworkArgs {
  cidr: pulumi.Input<string>
};

export class Network extends pulumi.ComponentResource {
  private _vpc: aws.ec2.Vpc;
  private _subnet: aws.ec2.Subnet;
  private _securityGroup: aws.ec2.SecurityGroup;

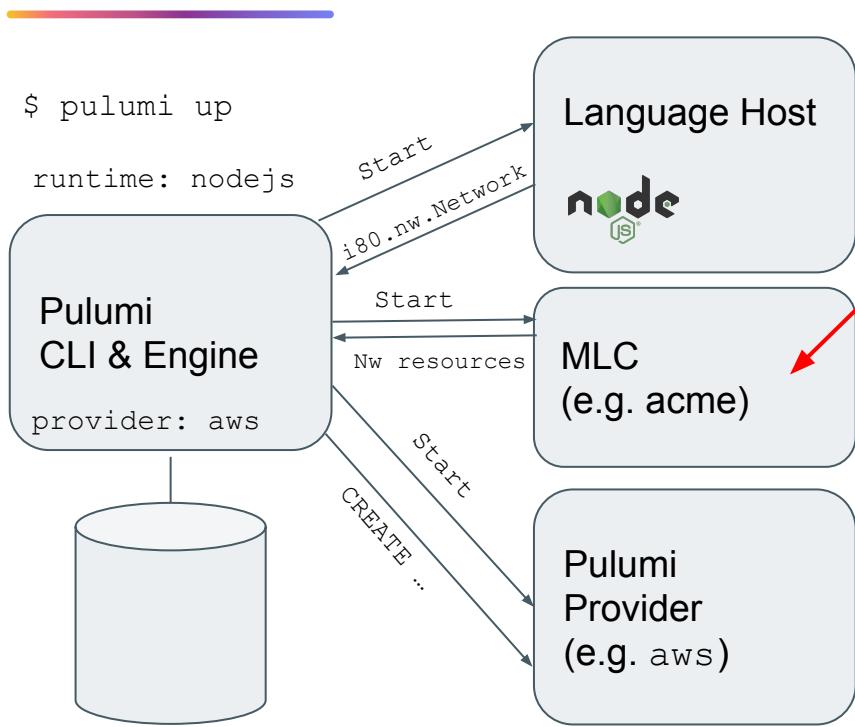
  constructor(name: string, args: NetworkArgs, opts?: pulumi.ComponentResourceOptions) {
    super("org:network:Network", name, {}, opts);
    this._vpc = new aws.ec2.Vpc(name, {
      cidrBlock: args.cidr,
      tags: {
        Name: name
      }
    });

    this._subnet = new aws.ec2.Subnet(name, {
      cidrBlock: args.cidr,
      vpcId: this._vpc.id,
      tags: {
        Name: name
      }
    });
}
```

# Pulumi - Multi Language Components



# Pulumi - Multi Language Component



```
export interface NetworkArgs {
  cidr: pulumi.Input<string>
};

export class Network extends pulumi.ComponentResource {
  private _vpc: aws.ec2.Vpc;
  private _subnet: aws.ec2.Subnet;
  private _securityGroup: aws.ec2.SecurityGroup;

  constructor(name: string, args: NetworkArgs, opts?: pulumi.ComponentResourceOptions) {
    super("org:network:Network", name, {}, opts);
    this._vpc = new aws.ec2.Vpc(name, {
      cidrBlock: args.cidr,
      tags: {
        Name: name
      }
    });

    this._subnet = new aws.ec2.Subnet(name, {
      cidrBlock: args.cidr,
      vpcId: this._vpc.id,
      tags: {
        Name: name
      }
    });
}
```



# Demo: VPC + Cluster



# Policies: Simple Program + TypeScript Policies



# Policies: VPC/Cluster + Policies using OPA Rego

Converting Full Terraform Programs to Pulumi

https://www.pulumi.com/blog/converting-full-terraform-programs-to-pulumi/

Introducing Review Stacks: Temporary, on-demand environments for reviewing changes in pull requests.  
[Read the blog.](#)

Star 16,295 Slack Docs Registry Pulumi AI Pulumi Cloud

Pulumi Blog Product Why Pulumi? Pricing Docs Learn Try Pulumi Cloud

## Converting Full Terraform Programs to Pulumi

Posted on Monday, Jun 12, 2023

Justin Van Patten

Over the last 2 years, we've seen an increasing trend of cloud development teams migrating to Pulumi from Terraform. These teams often have experience with and meaningful investment in Terraform, but have also typically run into limits of expressivity, productivity, scalability, or reliability with their existing tools. One of the first questions we hear when they decide to move to Pulumi is "how will I migrate my existing Terraform projects over?".

Today, we're excited to announce new support for converting whole Terraform projects to Pulumi via the `pulumi convert` command in the Pulumi CLI. The new Terraform converter includes support for Terraform modules, core features of Terraform 1.4, and the majority of Terraform built-in functions, converting to Pulumi TypeScript, Python, Go, or C#. The new converter can significantly reduce the amount of time it takes to migrate Terraform to Pulumi. Let's dig in to learn more about the new converter and how to use it.

Historically, we have offered a separate `tf2pulumi` tool to convert small snippets of Terraform to Pulumi. The new converter is no longer a separate tool. As of v3.71.0, you can run the new converter directly from the Pulumi CLI with the `pulumi convert --from terraform` command. And you can convert more than small snippets – the new converter supports converting full Terraform programs.

The new support in `pulumi convert` builds upon Pulumi's [CrossCode](#) foundations for providing universal infrastructure as code support across a wide variety of programming languages and conversion tooling between them. It also introduces a new concept of `converter` plugin in the Pulumi engine, which allows conversion tools from other Infrastructure as Code platforms to be integrated into the same `pulumi convert` experience in the future, both as part of the core project, as well as by other ecosystem partners and contributors.

Several common use cases are supported via the new `pulumi convert --from terraform` support in the Pulumi CLI:

- Converting your organization's existing Terraform projects to Pulumi
- Converting your organization's existing Terraform modules to Pulumi, to be consumed as part of existing

Share this post

Twitter LinkedIn

PulumiUP The Global Infrastructure as Code Conference Watch On-Demand

Recent Posts

- Introducing Azure Native 2.0
- Review Stacks: Collaborate in the Cloud
- Property Search: Enhanced Resource Search in Pulumi Cloud
- Announcing the Terraform Migration Offer
- Enhanced search & Navigation: The new Pulumi Docs experience
- Converting Full Terraform Programs to Pulumi
- Announcing OIDC Support for



# Thank you!

- <https://www.pulumi.com/docs>
- <https://www.pulumi.com/blog>
- <https://www.pulumi.com/registry>

