Xebia

IaC with Pulumi and GitHub Actions

Microsoft Solutions Partner

GitHub Verified Partner

Till Spindler
DevOps & Agile Coach

@spindev

https://blog.spindev.org

>15 years in software development with Microsoft stack

>10 years
ALM & DevOps

> 8 years
Scrum & Agile

Microsoft & GitHub
Certified Trainer

Agenda

01 Infrastructure as Code (IaC)

02 Pulumi

03 Demo / C# Pulumi static website

04 GitHub Actions

05 Demo / Create a GitHub Actions Workflow

06 Demo / Advanced example

IaC / Overview

- IaC is the practice of managing infrastructure in a declarative manner using code.
- Instead of manually configuring infrastructure, IaC allows you to define the desired state of your infrastructure in code.
- This code can then be version controlled, tested, and deployed in a repeatable and consistent manner.
- IaC tools can be used to manage a wide range of infrastructure, including servers, networks, and cloud resources.

IaC / Benefits

Increased efficiency and consistency

01

Improved collaboration and version control

02

Reduced risk of human error

03

Faster time to deployment

04

Improved scalability and flexibility

05

IaC / Different representatives

Terraform

- Open-source tool from HashiCorp
- Uses a declarative language to define infrastructure
- Supports a wide range of cloud providers and services
- Provides a plan command to preview changes before applying them
- Has a large and active community of users and contributors

Ansible

- Open-source tool from Red Hat
- Uses a procedural language to define infrastructure
- Supports a wide range of cloud providers and services
- Provides a dry-run mode to preview changes before applying them
- Has a large and active community of users and contributors

Pulumi / Overview

- is an open-source tool for managing cloud infrastructure.
- unlike other IaC tools, Pulumi uses familiar programming languages like JavaScript, Python, and Go to define infrastructure.
- Support of a wide range of cloud providers and services, including AWS, Azure, Google Cloud, and Kubernetes.
- provides a preview command to preview changes before applying them and supports incremental updates to infrastructure.
- Uses Stacks to support multiple environments (like dev, staging, prod, …)
- has a growing community of users and contributors and is backed by venture capital firms like Accel and Madrona Venture Group.

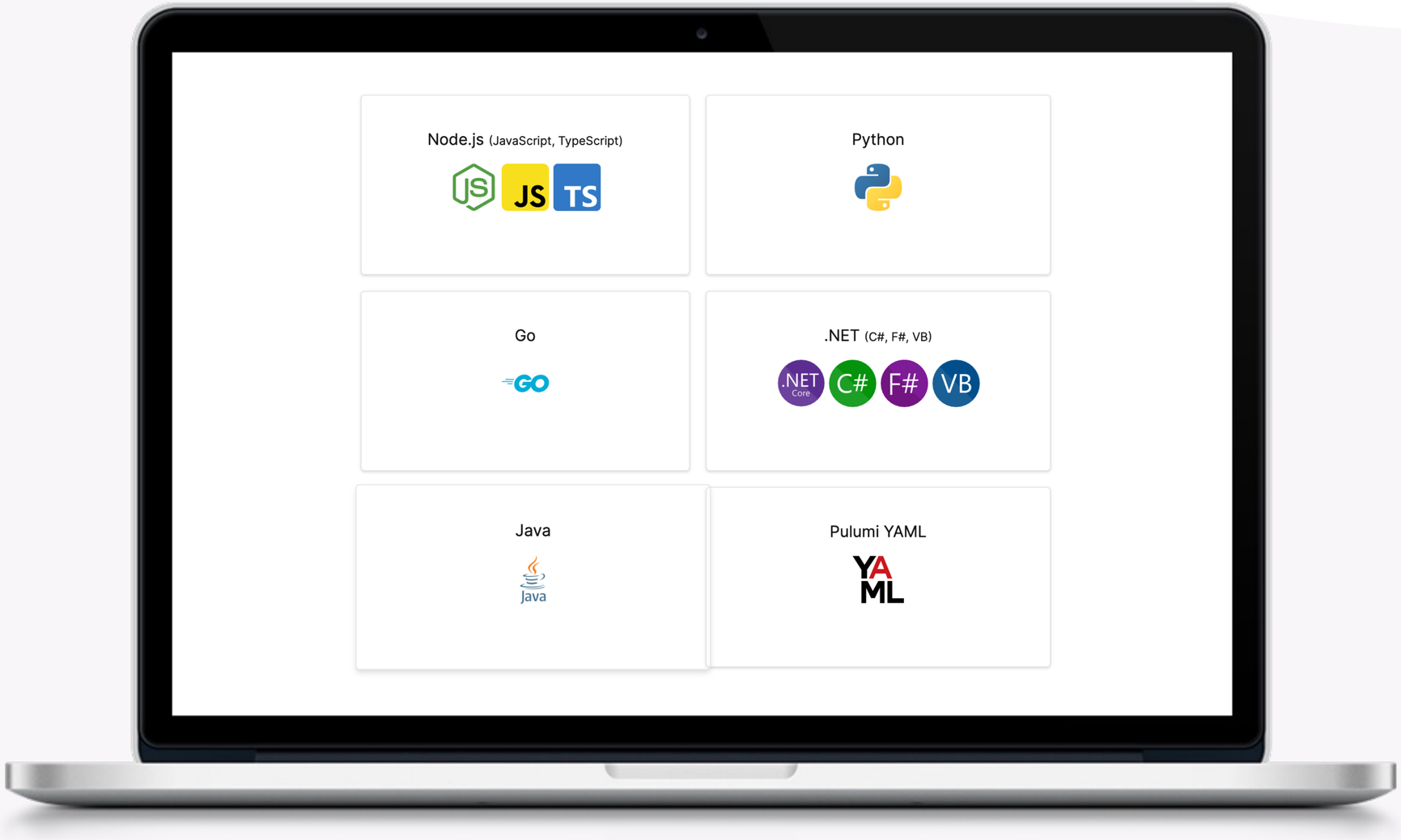
Pulumi / Clouds

Featured Packages

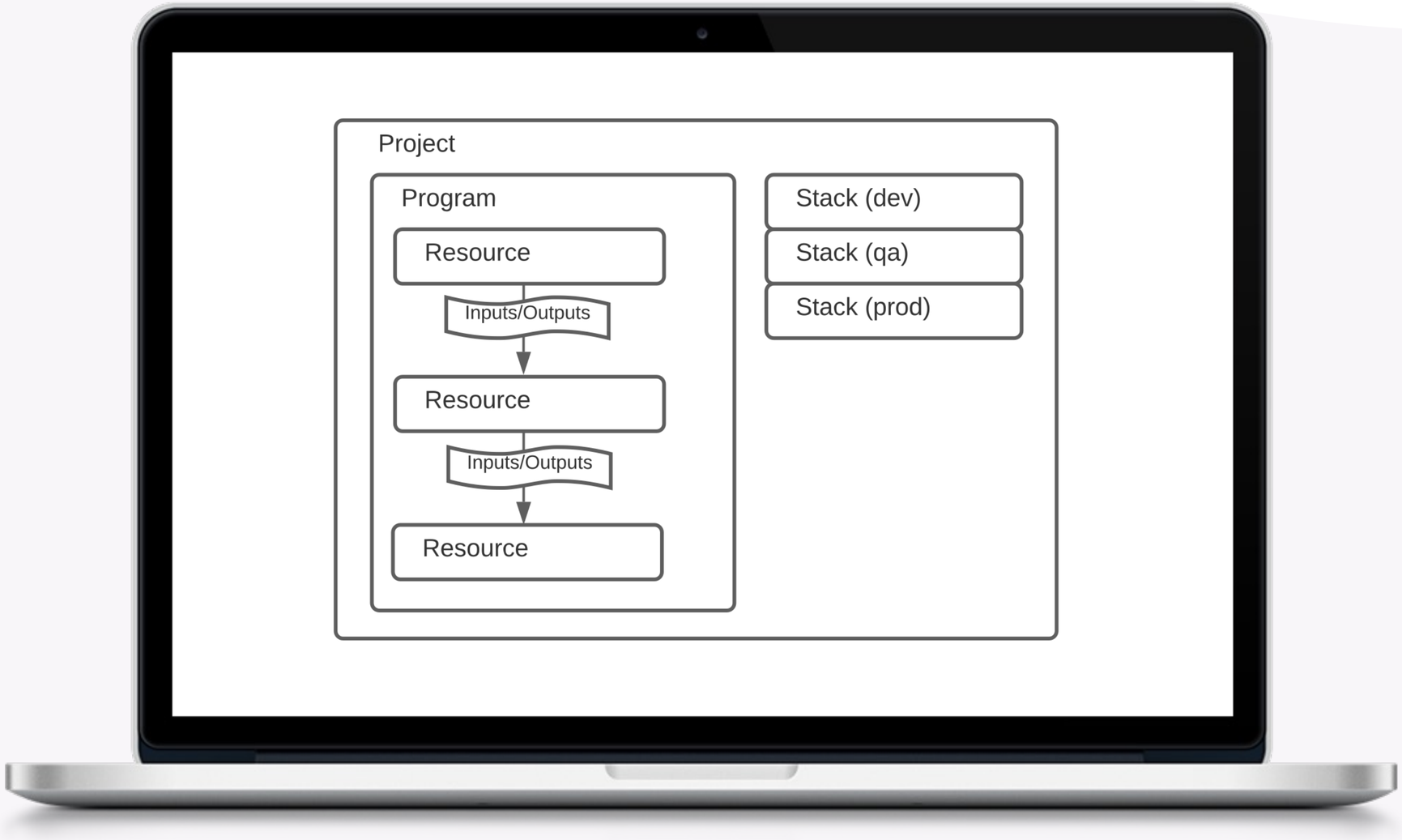
aws	Microsoft Azure	Google Cloud	kubernetes
AWS Classic	Azure Native	Google Cloud Classic	Kubernetes
v6.7.0	v2.14.0	v6.67.0	v4.5.3

More than
130 packages
and growing.

Pulumi / Languages & SDKs



Pulumi / Concepts



Pulumi / Stacks

- Stacks in Pulumi are like isolated, independently configurable instances of your Pulumi program.
- They are used to denote different phases of development (like staging, production) or feature branches (like feature-x, feature-y).
- Each stack has its own configuration file (Pulumi.<stack-name>.yaml) where you can set configuration values specific to that stack.
- You can switch between stacks using the **pulumi stack select <stack-name>** command.
- You can list all stacks using the **pulumi stack ls** command.

Pulumi / Getting started

1

Create an account for Pulumi cloud

2

Install the Pulumi CLI via download or package manager like Homebrew or Chocolatey

1

Create a new project with the command

- **pulumi new csharp –y** (starting from scratch) or
- **pulumi new** (starts an assistant and you can select from multiple templates)

Demo / Static Website on Azure with C#

GitHub Actions / Overview

GitHub Actions is a generic workflow engine (more than 35 trigger)

Allows directly referencing actions living in GitHub Repositories

Support for a lot of programming languages and frameworks

Fully integrated with all other GitHub features (like GitHub Packages, GitHub Pages, etc.)

Write your own Actions using one of the types

- JavaScript / TypeScript
- Docker
- Composite

GitHub Actions / Runner & Images

GitHub Hosted Runners & Self-Hosted runners available

Scaling Self-Hosted runner with Action Runner Controller (ARC)

Runner images available for GitHub Hosted Runner

Linux (Ubuntu)

Windows

MacOS

GitHub Actions / Getting started

Create a YAML
workflow file
in your repository
in the folder
.github/workflows

Add the basic
elements
- Name
- on (trigger)
- Jobs and Steps

Demo / Create a GitHub Actions Workflow

Demo / Advanced example

What do we have:

An ASPNET MVC application

What do we want to achieve:

Running the application on Azure Container Apps

How to this:

- Pulumi project with the required infrastructure
- Dockerize the MVC application
- GitHub Actions workflow to deploy it to Azure as part of CI/CD
- OIDC connection between GitHub Actions & Azure

Xebia

Q&A

Thanks for listening & any questions?