Power to the people
Combine imperative with declarative aspects

Ringo De Smet

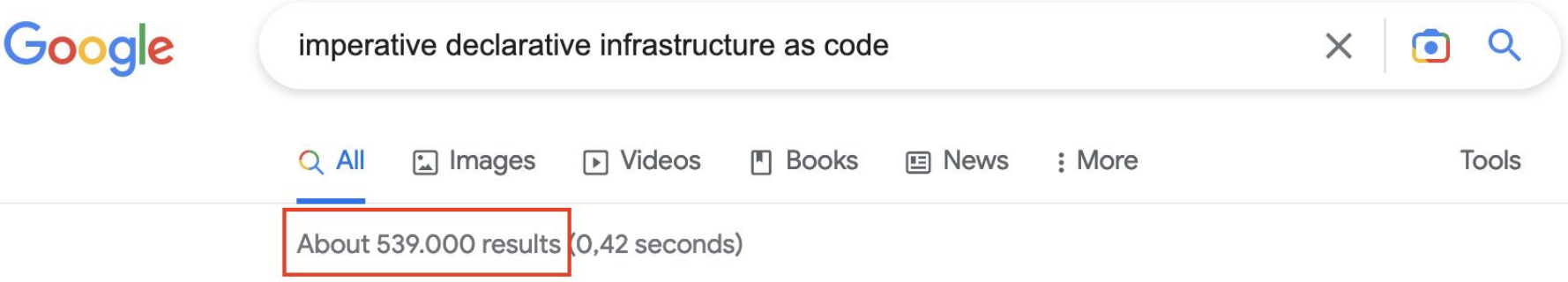
@ZiggyTheHamster@ruby.social / Keit... @ZiggyTh... · Mar 1, 2022

Replying to @ZiggyTheHamster @briggsl and 2 others

HCL can be represented in JSON and your Terraform project can be parsed with jq. This is not possible with Pulumi because Pulumi is imperative, not declarative. CloudFormation is another declarative example; CDK imperative.


Denis Makogon
@denis_makogon
On the other side that's what CDKs like #Pulumi CDK or #Terraform CDK exist. They attempt to solve that long-running battle "DECLARATIVE VS. IMPERATIVE".
12:16 AM · Jan 22, 2023 · 124 Views
parsed with
imperative, not
example; CDK imperative.


Denis Makogon
Faith Tosin.
@CapitalFAITH
Replying to @Joe_Stead
Multi-cloud argument is truly bullshit but one of the real advantages of Terraform over Pulumi is availablility of talents. Most Cloud/Devops engineers already know Terraform, still prefer declarative approach to infrastructure as code over imperative approach.


Google
imperative declarative infrastructure as code
All
Images
Videos
Books
News
More
Tools
About 539.000 results (0,42 seconds)

Adam Jacob
@adamhjk

Here’s a hot take for you: things can be both declarative and imperative *at the same time*. Trying to turn those properties into marketing wedges makes you ignorant, not smart. :)

7:26 PM · Jul 30, 2022

Both terms refer to how the user provides direction to the automation platform.

With an **imperative** tool, you define the steps to execute in order to reach the desired solution.

With a **declarative** tool, you define the desired state of the final solution, and the automation platform determines how to achieve that state.

Source: https://www.linode.com/blog/devops/declarative-vs-imperative-in-iac/

What is the behaviour when running multiple times?

Imperative

Run #1

all steps are executed and resources are created

Run #2

All steps are executed and resources are created *again*

Declarative

Run #1

All resources are declared and converged

Run #2

All resources are declared again, the tool remembers these are already converged, and does nothing

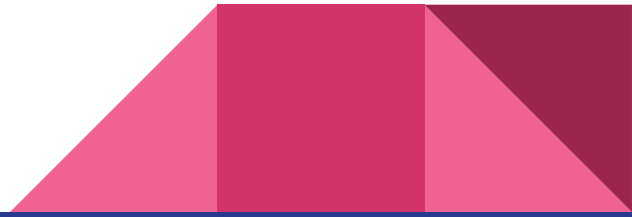
Imperative & Declarative aspects of tools in our toolbox

Chef

Helm

Pulumi

Dagger



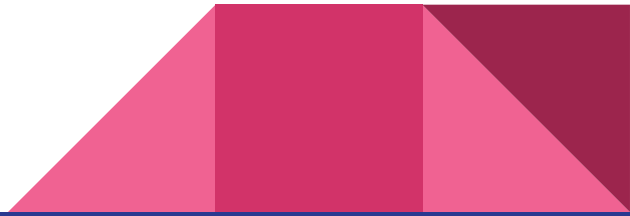
Chef

Short recap

- DSL in Ruby for configuration management

- Resource
- Recipe
- Cookbook

- And some more



Chef - single resource

```
package 'Install Apache' do
  case node[:platform]
  when 'redhat', 'centos'
    package_name 'httpd'
  when 'ubuntu', 'debian'
    package_name
 'apache2'
  end
end
```

Chef - collection of resources

```
packages = [ 'apache2', 'rails']

packages.each |pkg| do

  package "Install #{pkg}" do
    package_name pkg
  end

end
```

Chef - remote collection of resources

```
URI.open("http://server.somewhere.com/package-list.txt") |f| do

  f.each_line |pkg| do

    package "Install #{pkg}" do
      package_name pkg
    end

  end

end
```

Helm

```
{{- if .Values.server.service.enabled -}}
apiVersion: v1
kind: Service
metadata:
{{- if .Values.server.service.annotations }}
  annotations:
{{ toYaml .Values.server.service.annotations | indent 4 }}
{{- end }}
  labels:
    {{- include "prometheus.server.labels" . | nindent 4 }}
{{- if .Values.server.service.labels }}
{{ toYaml .Values.server.service.labels | indent 4 }}
{{- end }}
  name: {{ template "prometheus.server.fullname" . }}
{{ include "prometheus.namespace" . | indent 2 }}
spec:
{{- if .Values.server.service.clusterIP }}
  clusterIP: {{ .Values.server.service.clusterIP }}
{{- end }}
{{- if .Values.server.service.externalIPs }}
  externalIPs:
{{ toYaml .Values.server.service.externalIPs | indent 4 }}
```

Source: https://github.com/prometheus-community/helm-charts/blob/main/charts/prometheus/templates/service.yaml

Helm has over 60 available functions. Some of them are defined by the Go template language itself. Most of the others are part of the Sprig template library. We'll see many of them as we progress through the examples.

> While we talk about the "Helm template language" as if it is Helm-specific, it is actually a combination of the Go template language, some extra functions, and a variety of wrappers to expose certain objects to the templates. Many resources on Go templates may be helpful as you learn about templating.

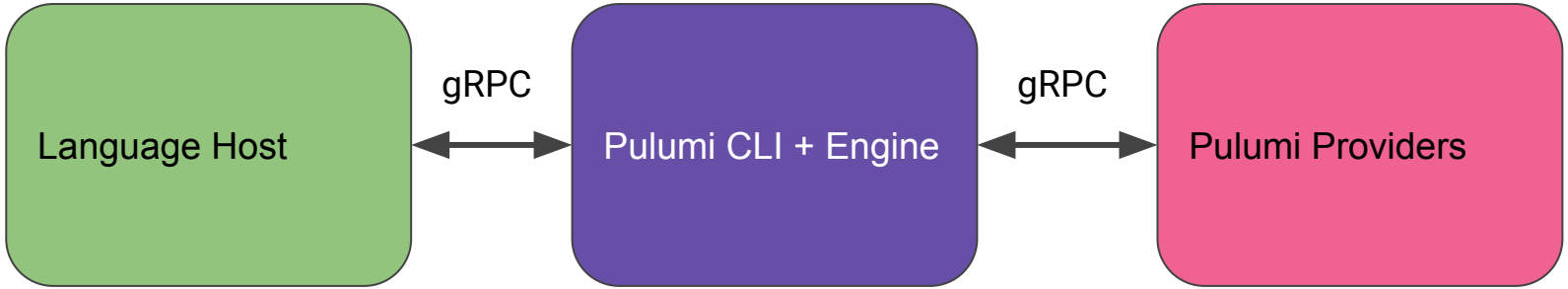
Source: https://helm.sh/docs/chart_template_guide/functions_and_pipelines/

- Cryptographic and Security
- Date
- Dictionaries
- Encoding
- File Path
- Kubernetes and Chart
- Logic and Flow Control
- Lists
- Math
- Network
- Reflection
- Regular Expressions
- Semantic Versions
- String
- Type Conversion
- URL
- UUID

Source: https://helm.sh/docs/chart_template_guide/function_list/

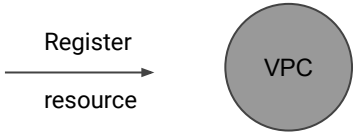
Pulumi

Pulumi Architecture



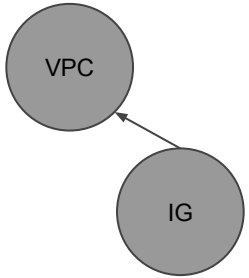
```
import * as aws from "@pulumi/aws";

const vpc = new aws.ec2.Vpc("myvpc", {
    cidrBlock: "10.0.0.0/16",
});

const internetGateway = new aws.ec2.InternetGateway("myinternetgateway", {
    vpcId: vpc.id,
});

const publicRouteTable = new aws.ec2.RouteTable("myroutetable", {
    routes: [
        {
            cidrBlock: "0.0.0.0/0",
            gatewayId: internetGateway.id,
        },
    ],
    vpcId: vpc.id,
});

const securityGroup = new aws.ec2.SecurityGroup("mysecuritygroup", {
    ingress: [
        { protocol: "tcp", fromPort: 80, toPort: 80, cidrBlocks: ["0.0.0.0/0"] },
    ],
    vpc: vpc.id,
});
```


```
3   const vpc = new aws.ec2.Vpc("myvpc", {
4       cidrBlock: "10.0.0.0/16",
5   });
```




```
3   const vpc = new aws.ec2.Vpc("myvpc", {
4       cidrBlock: "10.0.0.0/16",
5   });
```

Register resource

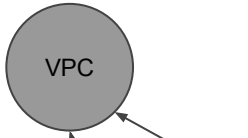



```
7   const internetGateway = new aws.ec2.InternetGateway("myinternetgateway", {
8       vpcId: vpc.id,
9   });
```

Register resource

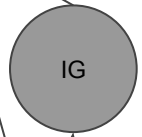

```
3   const vpc = new aws.ec2.Vpc("myvpc", {
4       cidrBlock: "10.0.0.0/16",
5   });
```

Register resource



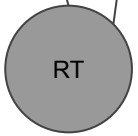

```
7   const internetGateway = new aws.ec2.InternetGateway("myinternetgateway", {
8       vpcId: vpc.id,
9   });
```

Register resource



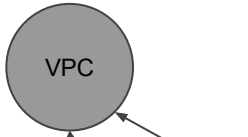

```
11  const publicRouteTable = new aws.ec2.RouteTable("myroutetable", {
12      routes: [
13          {
14              cidrBlock: "0.0.0.0/0",
15              gatewayId: internetGateway.id,
16          },
17      ],
18      vpcId: vpc.id,
19  });
```

Register resource



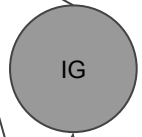

```
3   const vpc = new aws.ec2.Vpc("myvpc", {
4       cidrBlock: "10.0.0.0/16",
5   });
```

Register resource



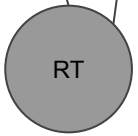

```
7   const internetGateway = new aws.ec2.InternetGateway("myinternetgateway", {
8       vpcId: vpc.id,
9   });
```

Register resource



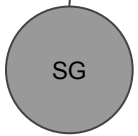

```
11  const publicRouteTable = new aws.ec2.RouteTable("myroutetable", {
12      routes: [
13          {
14              cidrBlock: "0.0.0.0/0",
15              gatewayId: internetGateway.id,
16          },
17      ],
18      vpcId: vpc.id,
19  });
```

Register resource




```
21  const securityGroup = new aws.ec2.SecurityGroup("mysecuritygroup", {
22      ingress: [
23          { protocol: "tcp", fromPort: 80, toPort: 80, cidrBlocks: ["0.0.0.0/0"] },
24      ],
25      vpc: vpc.id,
26  });
```

Register resource

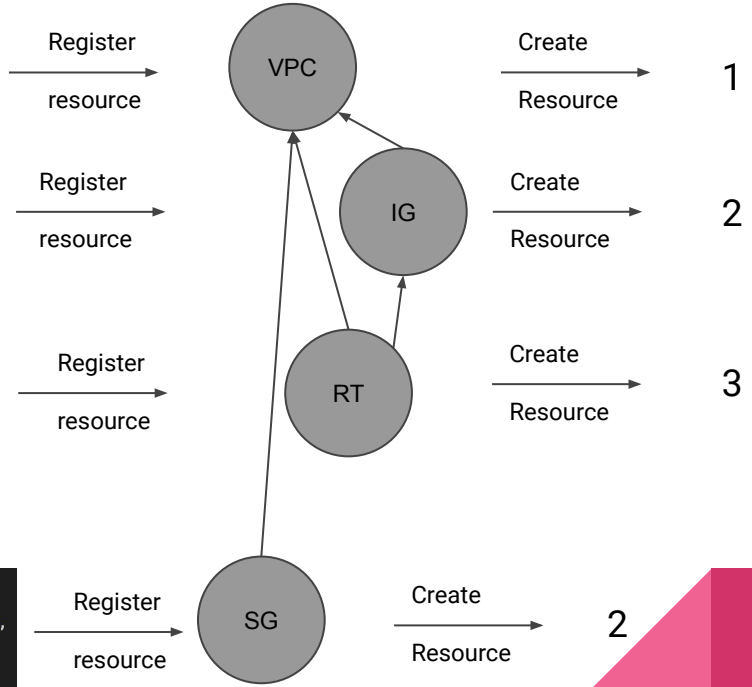



```
3   const vpc = new aws.ec2.Vpc("myvpc", {
4       cidrBlock: "10.0.0.0/16",
5   });
```

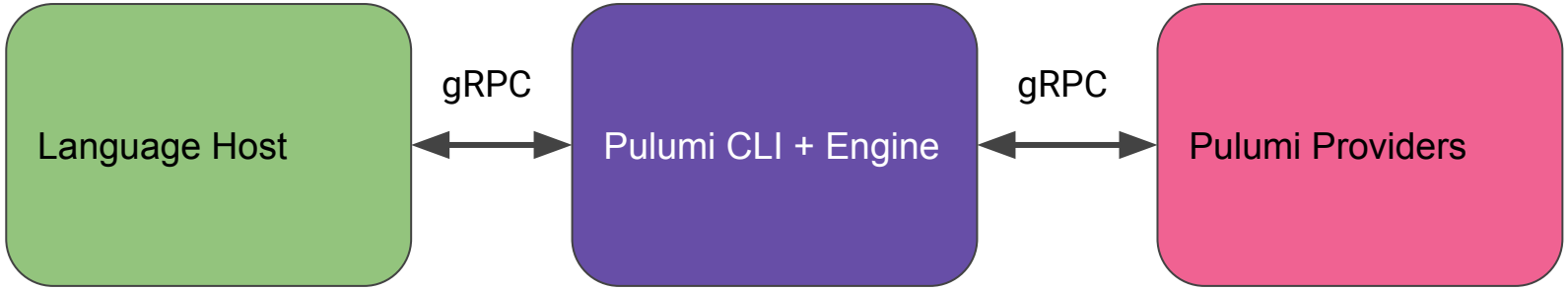
```
7   const internetGateway = new aws.ec2.InternetGateway("myinternetgateway", {
8       vpcId: vpc.id,
9   });
```

```
11  const publicRouteTable = new aws.ec2.RouteTable("myroutetable", {
12      routes: [
13          {
14              cidrBlock: "0.0.0.0/0",
15              gatewayId: internetGateway.id,
16          },
17      ],
18      vpcId: vpc.id,
19  });
```

```
21  const securityGroup = new aws.ec2.SecurityGroup("mysecuritygroup", {
22      ingress: [
23          { protocol: "tcp", fromPort: 80, toPort: 80, cidrBlocks: ["0.0.0.0/0"] },
24      ],
25      vpc: vpc.id,
26  });
```



Pulumi Architecture



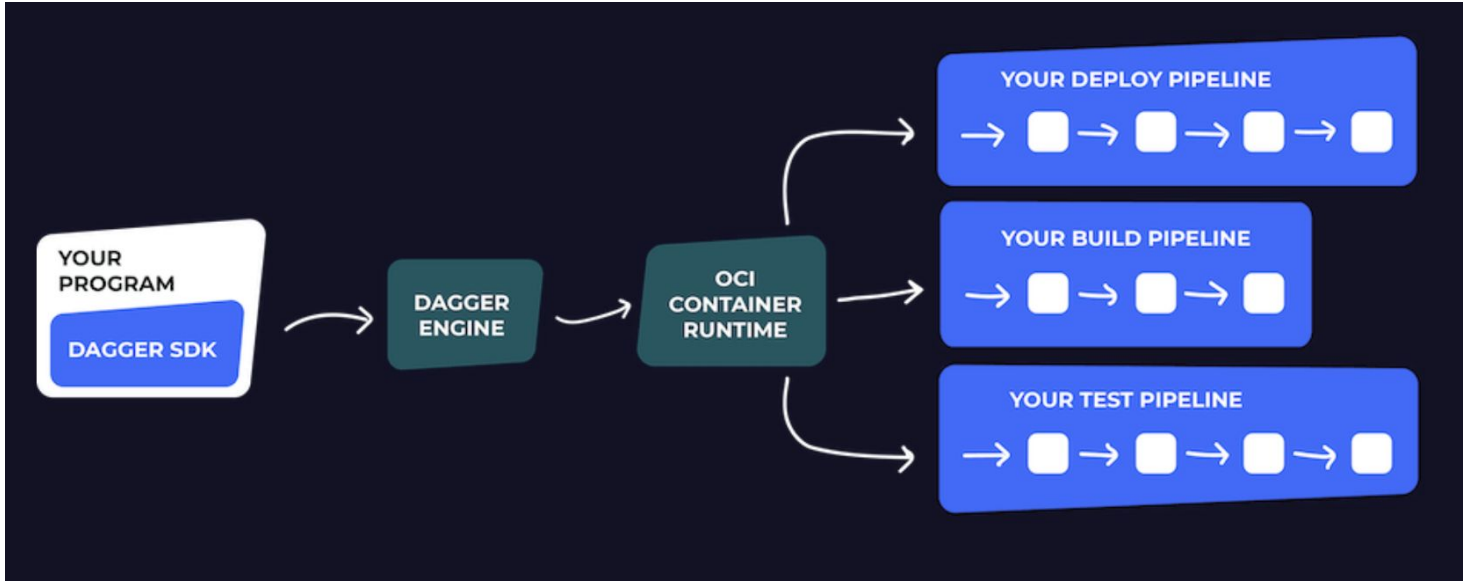
An imperative language setup drives a …

… declarative engine, which uses …

… imperative calls to fulfil the desired state

Dagger

Dagger (High Level) Architecture



```
import Client, { connect } from "@dagger.io/dagger"

// initialize Dagger client
connect(async (client: Client) => {
  // Set Node versions against which to test and build
  const nodeVersions = ["12", "14", "16"]

  // get reference to the local project
  const source = await client.host().directory(".", ["node_modules/"]).id()

  // for each Node version
  for (const nodeVersion of nodeVersions) {
    // get Node image
    const node = client.container().from(`node:${nodeVersion}`)

    // mount cloned repository into Node image
    const runner = client
      .container(node)
      .withMountedDirectory("/src", source)
      .withWorkdir("/src")
      .withExec(["npm", "install"])

    // run tests
    await runner.withExec(["npm", "test", "--", "--watchAll=false"]).exitCode()

    // build application using specified Node version
    // write the build output to the host
    await runner
      .withExec(["npm", "run", "build"])
      .directory("build/")
      .export(`./build-node-${nodeVersion}`)
  }
}, {LogOutput: process.stdout})
```

Why do we want both aspects?

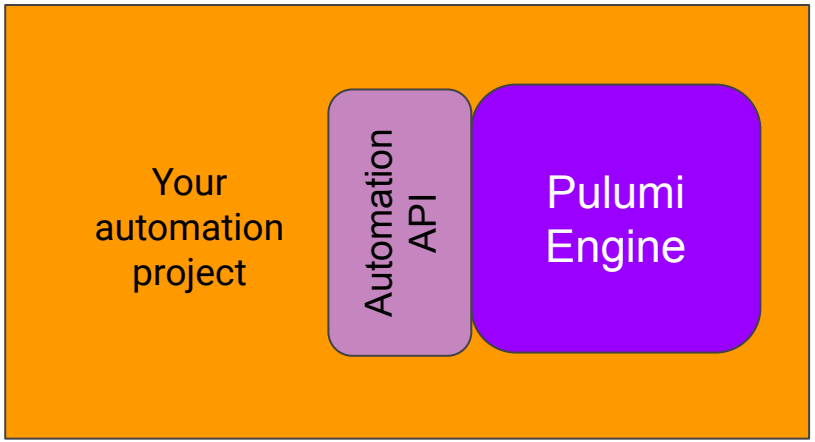
Integration!

```
import * as ec2 from 'aws-cdk-lib/aws-ec2';
import * as pulumi from '@pulumi/pulumi';
import * as pulumicdk from '@pulumi/cdk';
import * as ecs from 'aws-cdk-lib/aws-ecs';
import * as ecs_patterns from 'aws-cdk-lib/aws-ecs-patterns';
import { Construct } from 'constructs';
import { Stack, Duration, CfnOutput } from 'aws-cdk-lib';
import {remapCloudControlResource } from './adapter';

class FargateStack extends pulumicdk.Stack {

    loadBalancerDNS: pulumi.Output<string>;

    constructor(id: string, options?: pulumicdk.StackOptions) {
        super(id, { ...options, remapCloudControlResource });

        // Create VPC and Fargate Cluster
        const vpc = new ec2.Vpc(this, 'MyVpc', { maxAzs: 2 });
        const cluster = new ecs.Cluster(this, 'fargate-service-autoscaling', { vpc });
```

```
1  // Copyright 2016-2020, Pulumi Corporation.  All rights reserved.
2
3  import * as k8s from "@pulumi/kubernetes";
4
5  // Deploy the bitnami/wordpress chart.
6  const wordpress = new k8s.helm.v3.Chart("wpdev", {
7      version: "15.0.5",
8      chart: "wordpress",
9      fetchOpts: {
10         repo: "https://charts.bitnami.com/bitnami",
11     },
12 });
13
14 // Get the IP address once the chart is deployed and ready.
15 export let wordpressIP = wordpress.ready.apply(ready => wordpress
16     .getResourceProperty("v1/Service", "default", "wpdev-wordpress", "status")
17     .apply(status => status.loadBalancer.ingress[0].ip));
```

Pulumi Automation API

Allows you to embed Pulumi in bigger automations



```
28      // Create our stack
29      const args: InlineProgramArgs = {
30          stackName: "dev",
31          projectName: "inlineNode",
32          program: pulumiProgram
33      };
34
35      // create (or select if one already exists) a stack that uses our inline program
36      const stack = await LocalWorkspace.createOrSelectStack(args);
37
38      console.info("successfully initialized stack");
39      console.info("installing plugins...");
40      await stack.workspace.installPlugin("aws", "v4.0.0");
41      console.info("plugins installed");
42      console.info("setting up config");
43      await stack.setConfig("aws:region", { value: "us-west-2" });
44      console.info("config set");
45      console.info("updating stack...");
46      const upRes = await stack.up({ onOutput: console.info });
47      console.log(`update summary: \n${JSON.stringify(upRes.summary.resourceChanges, null, 4)}`);
48      console.log(`website url: ${upRes.outputs.websiteUrl.value}`);
```

```
13  const run = async () => {
14      // This is our pulumi program in "inline function" form
15      const pulumiProgram = async () => {
16          // Create a bucket and expose a website index document
17          const siteBucket = new s3.Bucket("s3-website-bucket", {
18              website: {
19                  indexDocument: "index.html",
20              },
21          });
22
23          return {
24              websiteUrl: siteBucket.websiteEndpoint,
25          };
26      };
```

Thank you!

@ringods

@ringods@mastodon.social