



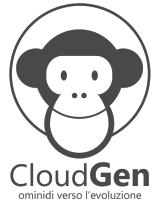
TRA POCO INIZIA IL LIVE



CloudGen



Microsoft



TOPIC

Deploy Azure with Pulumi



Andrea Ceroni



@andrekiba

andrea.ceroni@gmail.com

<https://github.com/andrekiba>

<https://www.linkedin.com/in/andreaceroni/>



pulumi



Modern Infrastructure as Code.

Declare cloud infrastructure using real languages.
Enable developers and operators to work better together.

Problemi



necessità di poter unire e far collaborare sempre di più il mondo dev e quello ops

anche se l'infrastruttura non è particolarmente complicata se occorre replicarla in diversi ambienti (prod, dev, staging...) può essere un discreto effort per la parte IT

nei diversi ambienti potremmo volere configurazioni differenti

chi si ricorda come ricreare il server?! 😞

se devo migrare ambiente cosa succede? ...tipicamente problemi 😊

Perchè Infrastructure as Code?



stato desiderato **descrivo cosa voglio, non come ottenerlo**

riutilizzabile e ripetibile stesso risultato tutte le volte di cui ne ho bisogno
(anche solo di alcune parti)

versioning dell'infrastruttura esattamente come per il codice delle app

sicura e veloce abbatte drasticamente i tempi di provisioning e con molti meno errori

documentata perchè il file in cui scrivo la configurazione rimane consultabile
ed è la "source of truth"

Management API



portali web

CLI

SDK

Management API (es: Azure Resource Manager)

Azure

AWS

GCP

...

```
az storage account create
--name devopsstorage
--resource-group g6
--location westeurope
--sku LRS
```

possibilità di riutilizzare lo script in momenti successivi e per ambienti diversi

lo scripting è però molto imperativo, occorre descrivere passo passo tutto quello che vogliamo sia presente nella nostra infrastruttura finale, in ordine rigoroso

non facile, dipende sia dallo stato attuale che da quello a cui si vuole arrivare

in caso di errore a metà cosa succede? 😞

SDK (es: Azure SDK for .NET)



```
public class AzureFunctionDeployment : BaseDeployment
{
    protected override Task ExecuteCreateAsync()
    {
        var definition = Azure
            .WithSubscription(Options.SubscriptionId)
            .AppServices.FunctionApps
            .Define(AppName)
            .WithRegion(Options.Region);

        var create = definition.WithExistingResourceGroup(
            Options.ResourceGroupName)

        return create.CreateAsync();
    }
}
```

Stato desiderato



ho un grafo dove i nodi rappresentano le risorse e gli archi le dipendenze

le dipendenze sono importanti perchè definiscono l'ordine in cui le varie risorse verranno create/modificate

lo strumento deve quindi permettermi di esprimere lo stato finale e poi occuparsi effettivamente di renderlo effettivo

la migrazione dovrebbe essere “indolore” indipendentemente dallo stato attuale in cui l’infrastruttura si trova



Cloud Formation, ARM Templates, Google Cloud Deployment Manager terribilmente verbosi

esistono diversi tool a supporto
ma non un vero e proprio intellisense

non c'è alcun controllo a compile time,
sono solo file di testo

diversi a seconda del provider

devono mantenere compatibilità e quindi evolvono poco velocemente

si traducono in chiamate REST, difficile avere una preview di quello che accade

```
"resources": [
  {
    "apiVersion": "06.06.06",
    "type": "Microsoft.Storage/storageAccounts",
    "name": "devopsstorage",
    "location": "westeurope",
    "sku": {"name": "Standard LRS" },
    "kind": "Storage",
    "properties": { ... },
  }
]
```

multi-cloud (imparato per un cloud sarà più o meno simile per un altro)

la definizione delle risorse è cmq differente (HCL è un DSL)

linguaggio proprietario che non è uno dei linguaggi tipici con cui sviluppiamo codice tutti i giorni

rimane cmq un file di testo, simile a JSON

vasto numero di provider...più meno per qualsiasi cosa 😊

```
No:  
| Body:  
| | Wants:  
| | | To:  
| | | | Write:  
| | | | - YAML
```

```
# 😊 Why YAML is the right devops technology for you 😊  
#  
# - 100% test coverage, always compiles just fine with no errors or warnings, always shippable  
# - no enforced error handling during development because runtime "panic at the disco" in production is dope  
# - "something broke" is way better than stack traces with line numbers  
# - you need to burn hours as part of setting up a new CI pipeline  
# - safe choice with unquestionable industry adoption, "used by kubernetes"  
# - is marginally better than windows.ini  
# - unlike json [1], YAML supports comments  
# - you need a super safe way to "execute this code"  
  
# 🚫 wait a sec, did you say "executable yaml"?? 🚫  
# - https://ruby-doc.org/stdlib-2.4.0/libdoc/yaml/rdoc/YAML.html#module-YAML-label-Security  
# - https://www.php.net/manual/en/function.yaml-parse.php#refsect1-function.yaml-parse-notes  
# - https://lgtm.com/blog/swagger\_snakeyaml\_CVE-2017-1000207\_CVE-2017-1000208  
# - https://github.com/yaml/pyyaml/wiki/PyYAML-yaml.load\(input\)-Deprecation  
  
# 🇳🇴 Anyone who uses YAML long enough will eventually get burned when attempting to abbreviate Norway 🇳🇴  
# `NO` is parsed as a boolean type, which with the YAML 1.1 spec, there are 22 options to write "true" or "false."  
# You have to wrap "NO" in quotes to get the expected result.  
NI: Nicaragua  
NL: Netherlands  
NO: Norway # 🇳🇴!
```

Loop, dipendenze, condizioni ...



si usa ad esempio `Ref` per esprimere le dipendenze in AWS CloudFormation

esprimere condizioni e variabili nei template ARM non è così facile

oppure pensiamo ai loop di Terraform dove di fatto esiste uno pseduo-linguaggio per poterli definire e usare

se ho configurazioni diverse dell'infrastruttura per ambienti diversi è complicato avere un buon risultato in modo semplice

difficile astrarre e creare componenti

difficile condividere parti riutilizzabili (moduli in Terraform)

difficili da testare, l'unico modo è provare a deployare e vedere cosa succede 😊

Esempi



```
"Resources": {
    "MyEC2Instance": { ... }
    "MyEIP": {
        "Type": "AWS::EC2::EIP",
        "Properties": {
            "InstanceId": {
                "Ref": "MyEC2Instance"
            }
        }
    }
}

"resources": [
{
    "condition": "[equals(parameters('newOrExisting'), 'new')]",
    "apiVersion": "06.06.06",
    "type": "Microsoft.Storage/storageAccounts",
    "name": "devopsstorage",
    "location": "westeurope",
    "sku": {"name": "Standard LRS" },
    "kind": "Storage",
    "properties": { ... },
}
]
```

```
foo {
    files = ["index.html", "404.html"]
}

resource "aws_s3_bucket_object" "files" {
    count = "${length(foo.files)}"
    key   = "${foo.files[count.index]}"
    bucket = "${aws_s3_bucket.examplebucket.id}"
    source = "./src/${foo.files[count.index]}"
}
```

OK ... quindi cosa vorrei?



scriptabile

riproducibile

multi-cloud

permetta di esprimere stato desiderato

ma con un vero linguaggio di programmazione

con che linguaggio?

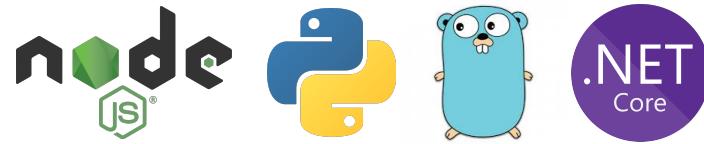
sarebbe bello se potessi farlo in C# o TypeScript o ...

e poter quindi sfruttare tutto quello che un linguaggio ci mette a disposizione
costrutti, cicli, IDE, code completion/intellisense, compilatore, astrazioni ...

free e open source per uso individuale (a pagamento per uso team la parte SAAS)

engine sviluppato in Go e api in TypeScript

supporta vari cloud provider e diversi linguaggi
TypeScript, Javascript, Python, C#, F#, Go



il codice sembra imperativo ma in realtà stiamo solo descrivendo cosa vogliamo

l'engine si occuperà di allinerare lo stato in seguito a modifiche nel codice
non dobbiamo scrivere il codice che migra tra diverse versioni di infra

è codice...non codice strano in qualche tipo di file di configurazione! 😊

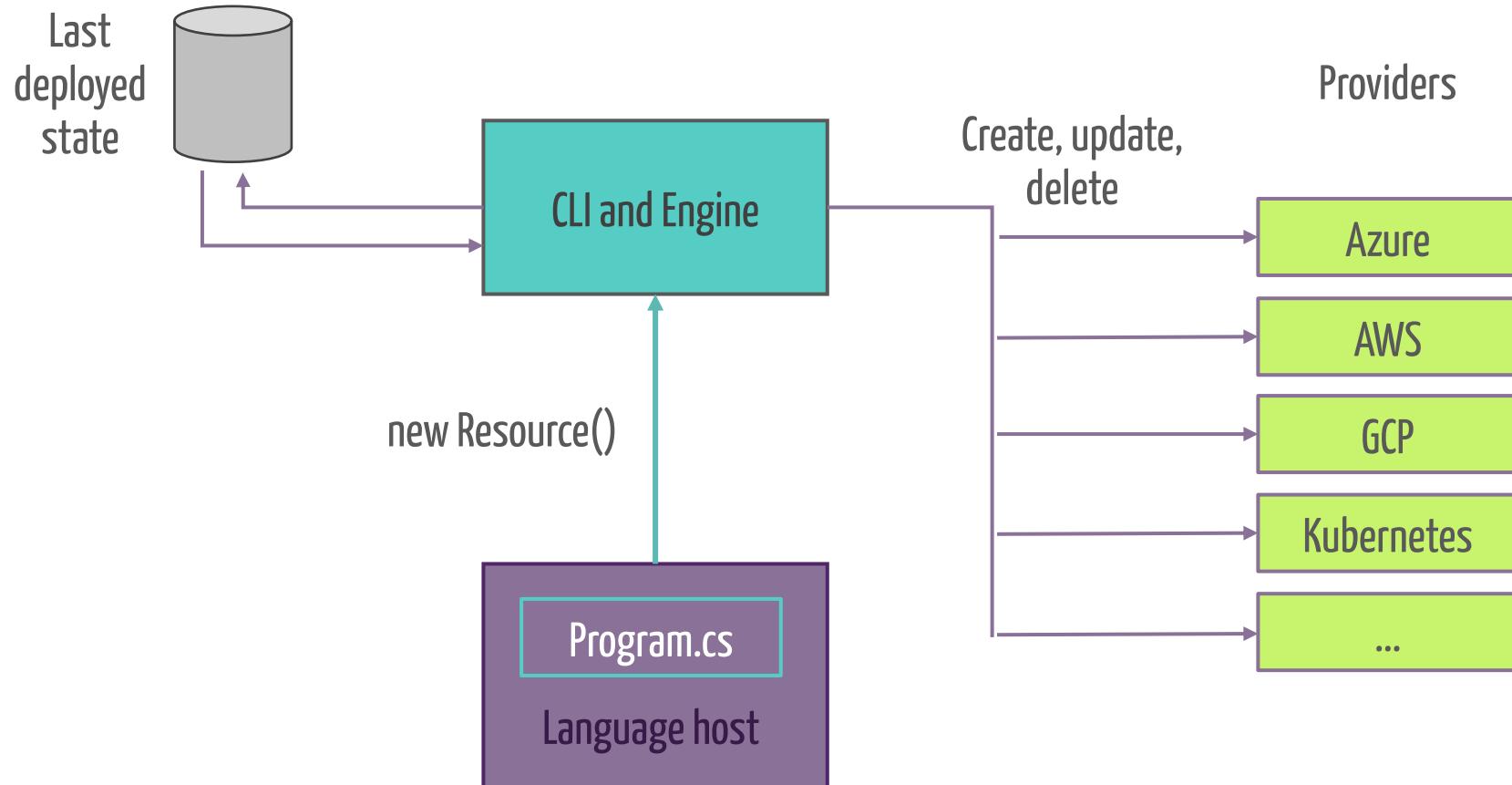
Esempio



```
var resourceGroup = new ResourceGroup("globalazure-rg");
var storageAccount = new Account("globalazurrest", new AccountArgs
{
    ResourceGroupName = resourceGroup.Name,
    AccountReplicationType = "LRS",
    AccountTier = "Standard",
    //EnableHttpsTrafficOnly = true
});
```

non sto ancora fisicamente creando nulla
sto di fatto creando il grafo di risorse e dipendenze che compongono l'architettura

Come funziona



Componenti e astrazione



programming model

blend tra infrastruttura e codice

componenti riutilizzabili (astraiendo tutto quello che non dobbiamo setizzare)
molto potente perchè possiamo di fatto utilizzare OOP o FP per sviluppare quello che ci serve

rende il codice molto compatto

per creare un custom component è sufficiente estendere Resource di Pulumi

posso anche creare un custom provider estendendo Provider

Project pulumi new azure-csharp

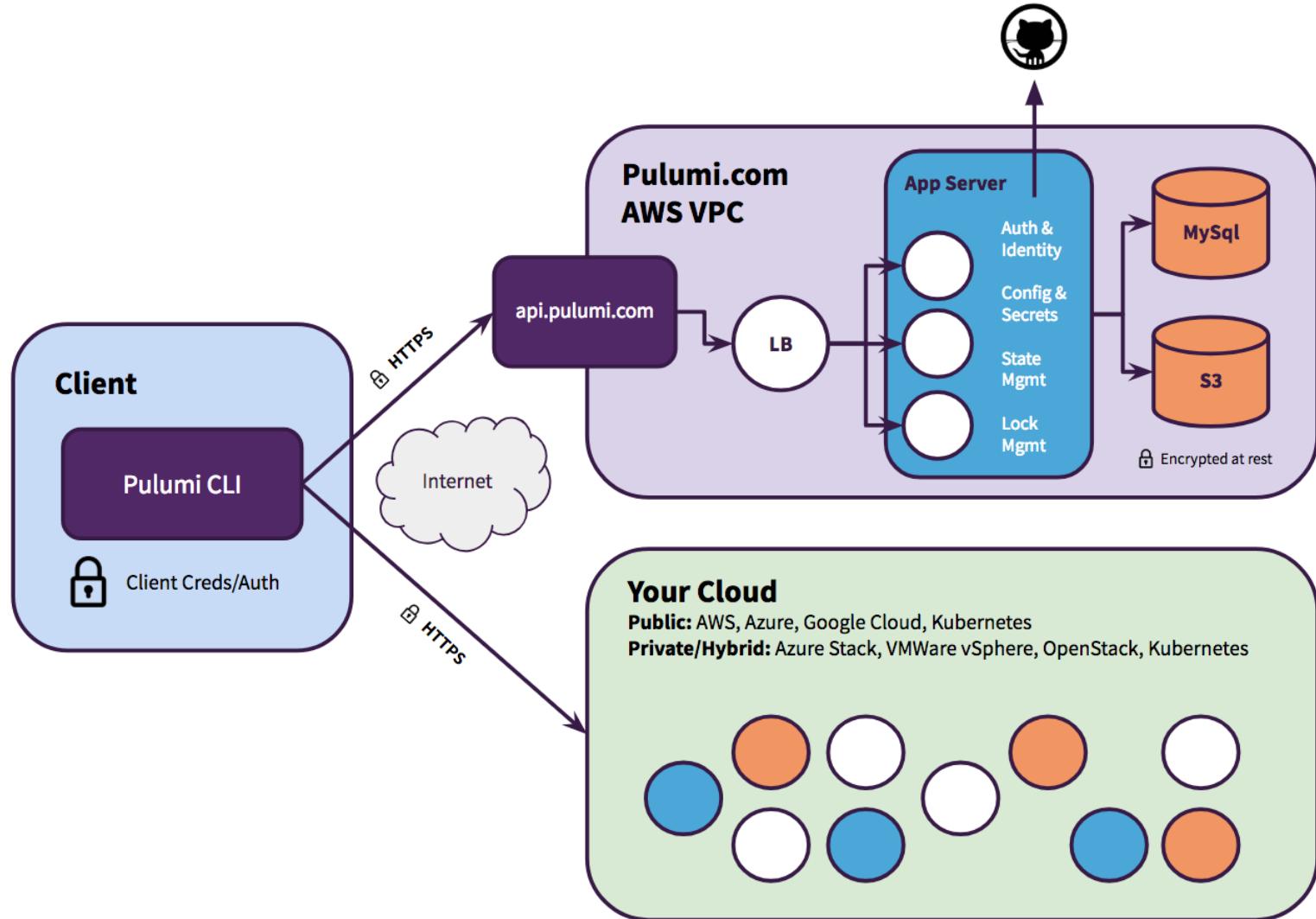
definisce un singolo gruppo di deploy
può contenere uno o più stack
contiene la definizione dell'infrastruttura

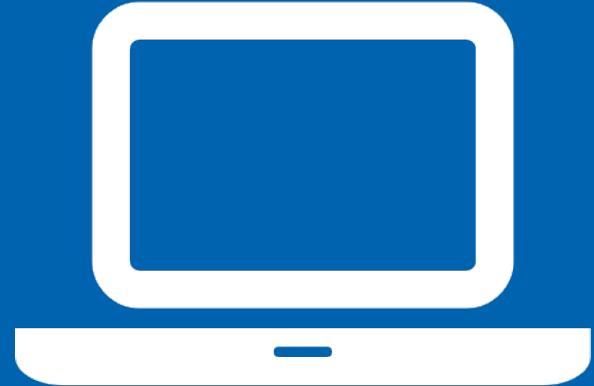
Stack pulumi stack init dev

configurazione specifica
rappresenta un singolo deploy del progetto

State pulumi state

mantiene lo stato attuale dell'infrastruttura e lo confronta con quello desiderato





Demo

pulumi stack init Q&A

pulumi up



<https://github.com/andrekiba/VirtualGAB20>

<https://www.pulumi.com>

<https://www.pulumi.com/docs/>

<https://github.com/pulumi/pulumi>

<https://github.com/pulumi/examples>

<https://github.com/pulumi/infrastructure-as-code-workshop>

<https://www.pulumi.com/blog/>

<https://www.youtube.com/channel/UC2Dhyn4Ev52YSbcfnfP0Mw>

Thanks!



andrea.ceroni@elfo.net

andrea.ceroni@gmail.com

[@andrekeiba6](https://twitter.com/andrekeiba)

[http://github.com/andrekeiba](https://github.com/andrekeiba)

[http://www.linkedin.com/in/andreaceroni](https://www.linkedin.com/in/andreaceroni)