Pulumi

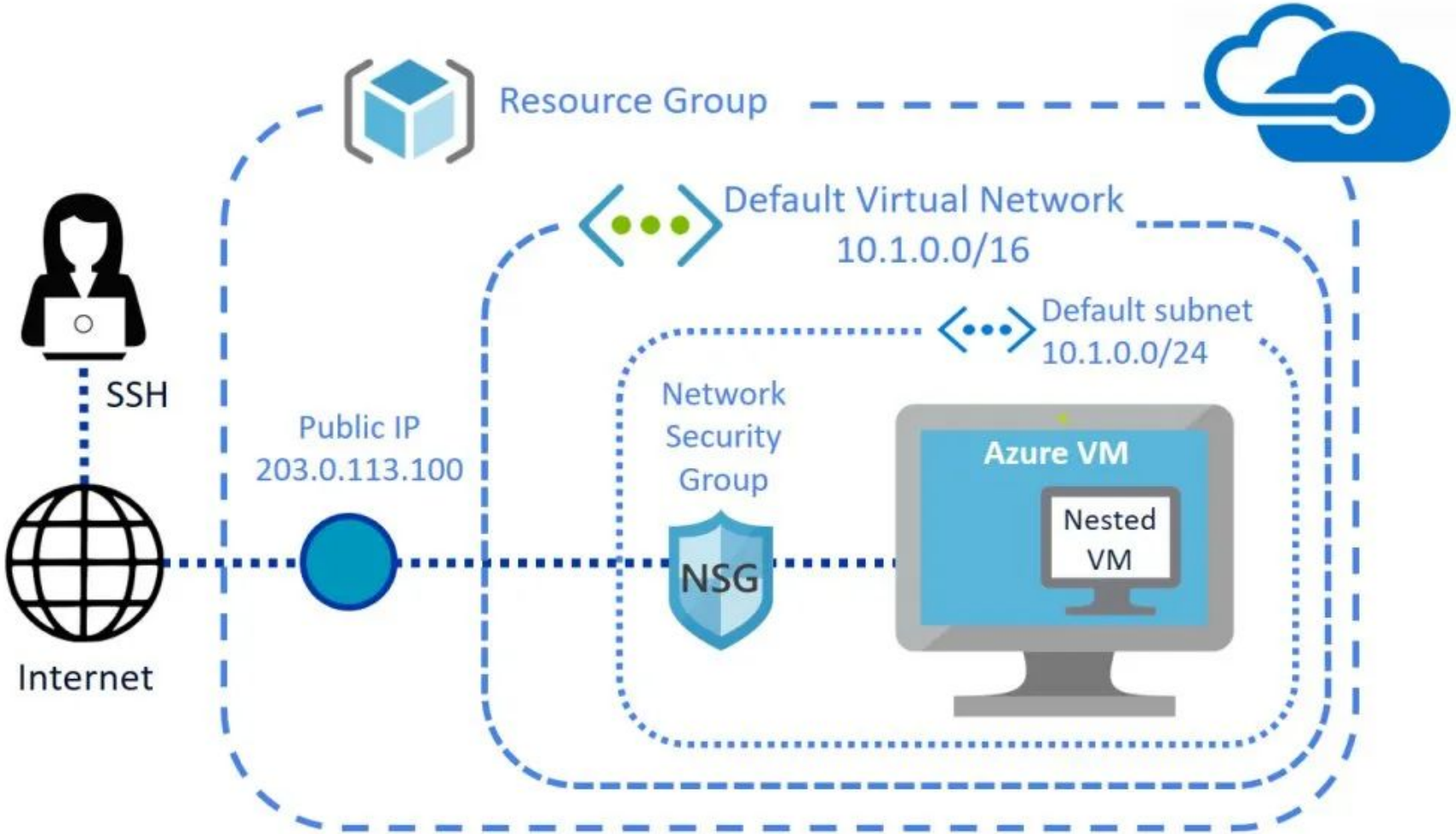
Architecture overview

Anirudh

contents

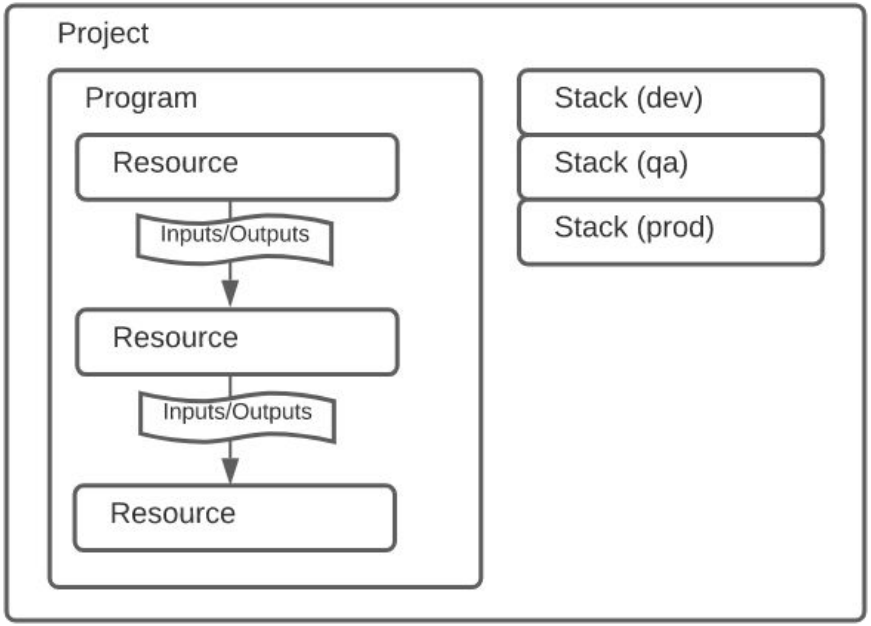
Introduction

- Infrastructure as code
- Creating, deploying, and managing cloud infrastructure
- Open source - Github
- VMs, networks, and databases
- Containers, Kubernetes clusters, and serverless functions
- CLI, runtime, libraries, and a hosted service - Pulumi


Resource Group
Default Virtual Network
10.1.0.0/16
Default subnet
10.1.0.0/24
SSH
Public IP
203.0.113.100
Network
Security
Group
NSG
Azure VM
Nested
VM
Internet

Projects

- Folder with Pulumi.yaml file




```
name: pypulumi_az
runtime:
  name: python
  options:
    virtualenv: venv
description: azure
infra in python using
pulumi
```

Stacks

- Every program must have a stack
- A Project can have multiple stacks
- Each stack is isolated from other stacks - How?
- Each stack is independently configurable - How and Why?
- Eg : phases of development (development, staging, and production)
    - feature branches (such as feature-x-dev)

Config stack

- Pulumi.<stackname>.yaml
- key-value pairs
- CLI - config set/get
- Programming model - config object

```
config:
  azure-native:location:
westus
  azure-py-webserver:password:
    secure:
AAABAME7tmNvU3guvVN2IbNWsux07J
i/u9RfSNHw6hRWPQyA6sfW/A==
  azure-py-webserver:username:
webmaster
```

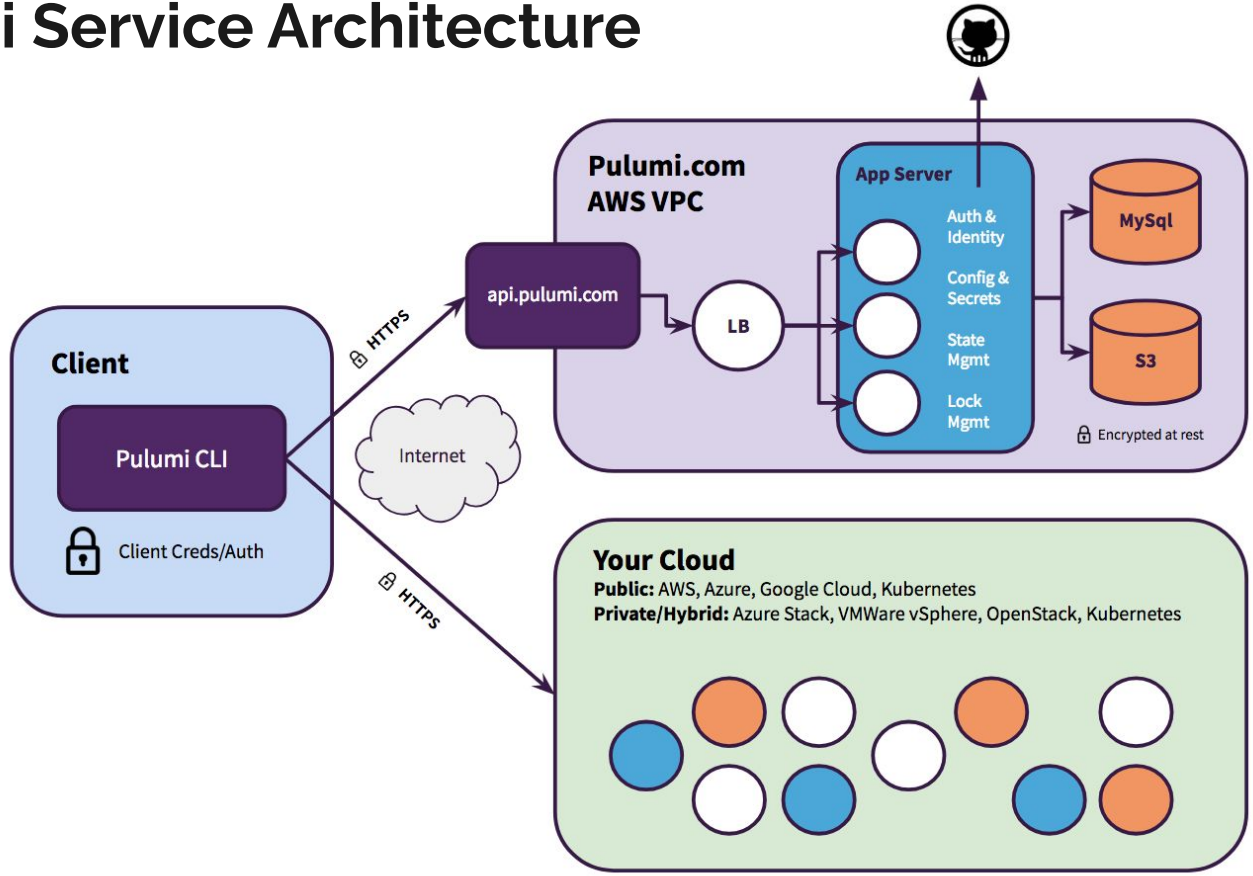
State and Backend

- Pulumi program - desired state of your infrastructure
- Metadata about your infrastructure - state
- Each stack has its own state
- diff your program's goal state against the last known update, recover from failure, and destroy resources accurately

Backend - Pulumi Service

- Store the state in a backend
- Pulumi Service or
- Self-Managed: a manually managed object store, AWS S3, Azure Blob Storage, your local filesystem
- cloud credentials - managed by CLI

Pulumi Service Architecture



Resources

- fundamental components of infrastructure
- compute instance, storage bucket, or database instance
- All infra resources are one of two subclasses:
  - CustomResource: managed by a resource providers -AWS, Azure,GCP
  - ComponentResource: logical grouping of other resources that creates a larger, higher-level abstraction

Custom Resources

- From Pulumi SDK libraries for aws,azure,gcp,etc.
- import the relevant library package like any other library
- A custom resource's desired state is declared by constructing an instance of the resource
- res = Resource(name, args, options)

Eg: a vnet custom resource

```
# virtualNetworkResource
virtual_network = network.VirtualNetwork("virtualNetwork",
    address_space = network.AddressSpaceArgs(
        address_prefixes=["10.0.0.0/16"],
    ),
    virtual_network_name="test-vnet",
    location=resource_group.location,
    resource_group_name = resource_group.name)
```

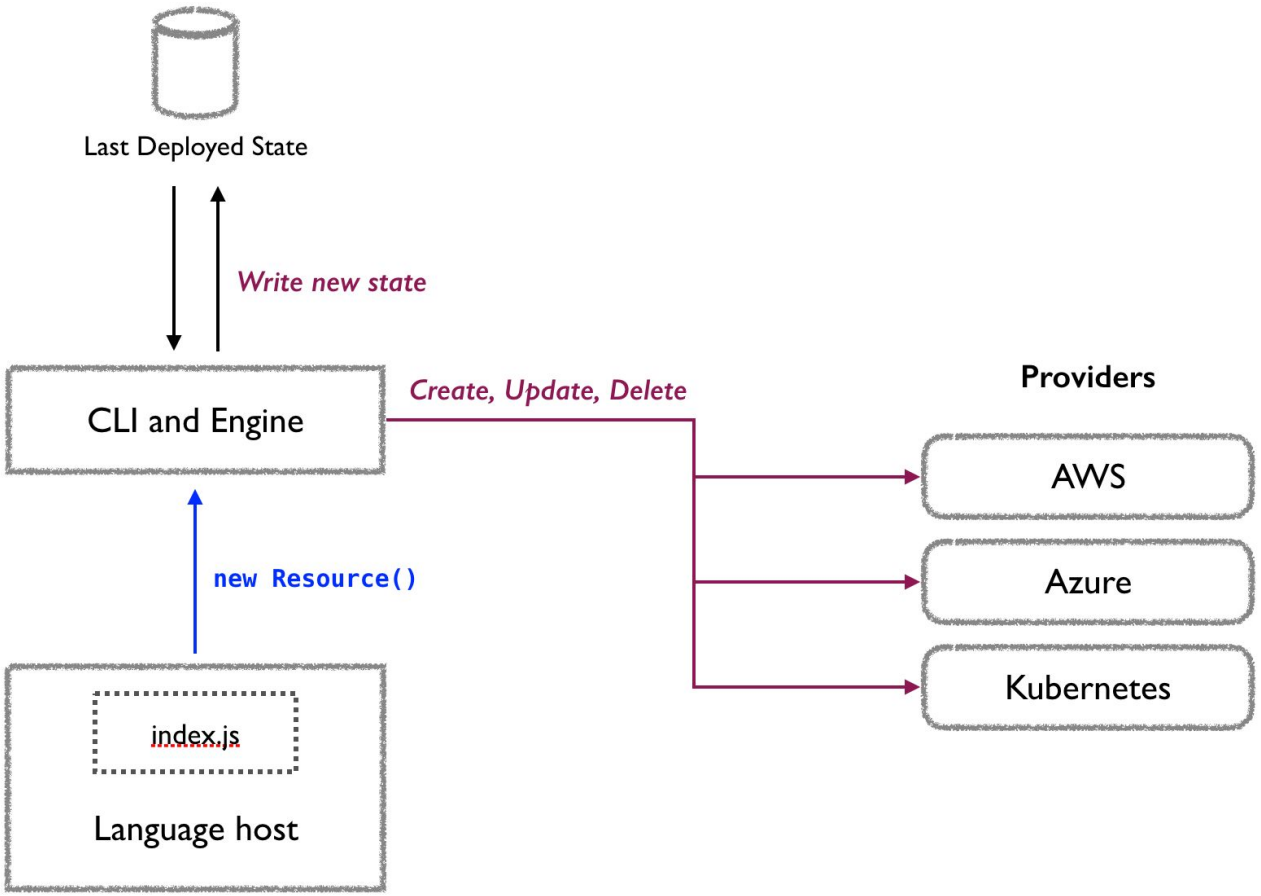
Resource object

- res = Resource(name, args, options)
- name - **logical name**
  - unique across resources of the same kind in a stack
  - **physical name** - assigned by your infrastructure's cloud provider
  - Pulumi **auto-names** physical resources by default - logical name+Random Chars eg: VirtualNetwork-d7c2fa0
  - Override with setting it in args
  - **variable names** assigned to resource objects are not used - res in first line

Arguments and Options

- Resource's argument parameters differ by resource type.
- All resource constructors accept an options argument that provide resource options
  - Aliases, custom timeout …..
  - Same options for all types of resources

Pulumi Up


Last Deployed State
Write new state
CLI and Engine
Create, Update, Delete
Providers
AWS
Azure
Kubernetes
new Resource()
index.js
Language host

- Language Host :
  - Language executor - launch the runtime - comes with CLI
  - Language runtime - detect resource registrations and sends request back to the deployment engine - comes from package manager - pypi,npm
- Deployment Engine -
  - consults the existing state - backend like pulumi service
  - Asks the resource provider in order to create it
  - Comes with CLI
- Resource Providers
  - resource plugin - binary used by the engine to manage a resource
  - SDK - provides bindings for each type of resource -comes from PyPi,npm

Putting it all together

```
# Create an Azure Resource Group
resource_group = resources.ResourceGroup("resourceGroup_py",
    location="eastus",
    resource_group_name="myResourceGroup")
```

- Pulumi up
  - Pulumi CLI - Python language host to exec program
  - language host- resource registration request to the deployment engine
  - Engine - resource plugin to create the resource and the plugin uses the SDK