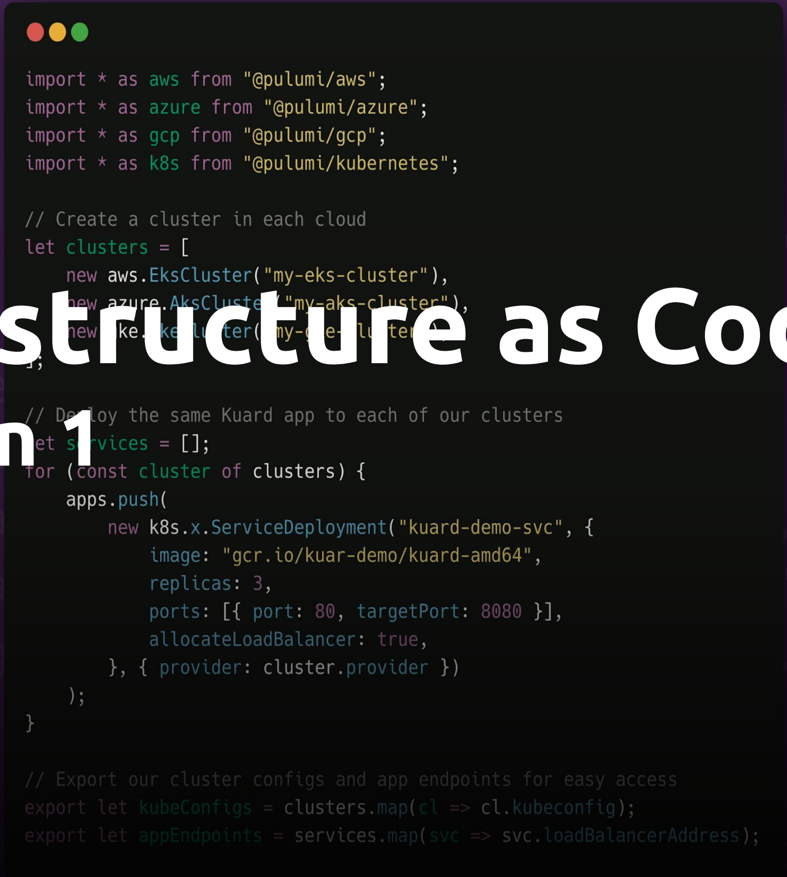
pulumi

Infrastructure as Code

Session 1



```
import * as aws from "@pulumi/aws";
import * as azure from "@pulumi/azure";
import * as gcp from "@pulumi/gcp";
import * as k8s from "@pulumi/kubernetes";

// Create a cluster in each cloud
let clusters = [
    new aws.EksCluster("my-eks-cluster"),
    new azure.AksCluster("my-aks-cluster"),
    new gke.GkeCluster("my-gke-cluster"),
];

// Deploy the same Kuard app to each of our clusters
let services = [];
for (const cluster of clusters) {
    apps.push(
        new k8s.x.ServiceDeployment("kuard-demo-svc", {
            image: "gcr.io/kuar-demo/kuard-amd64",
            replicas: 3,
            ports: [{ port: 80, targetPort: 8080 }],
            allocateLoadBalancer: true,
        }, { provider: cluster.provider })
    );
}

// Export our cluster configs and app endpoints for easy access
export let kubeConfigs = clusters.map(cl => cl.kubeconfig);
export let appEndpoints = services.map(svc => svc.loadBalancerAddress);
```

Session 1 Overview

Intro (slides) - 30 mins

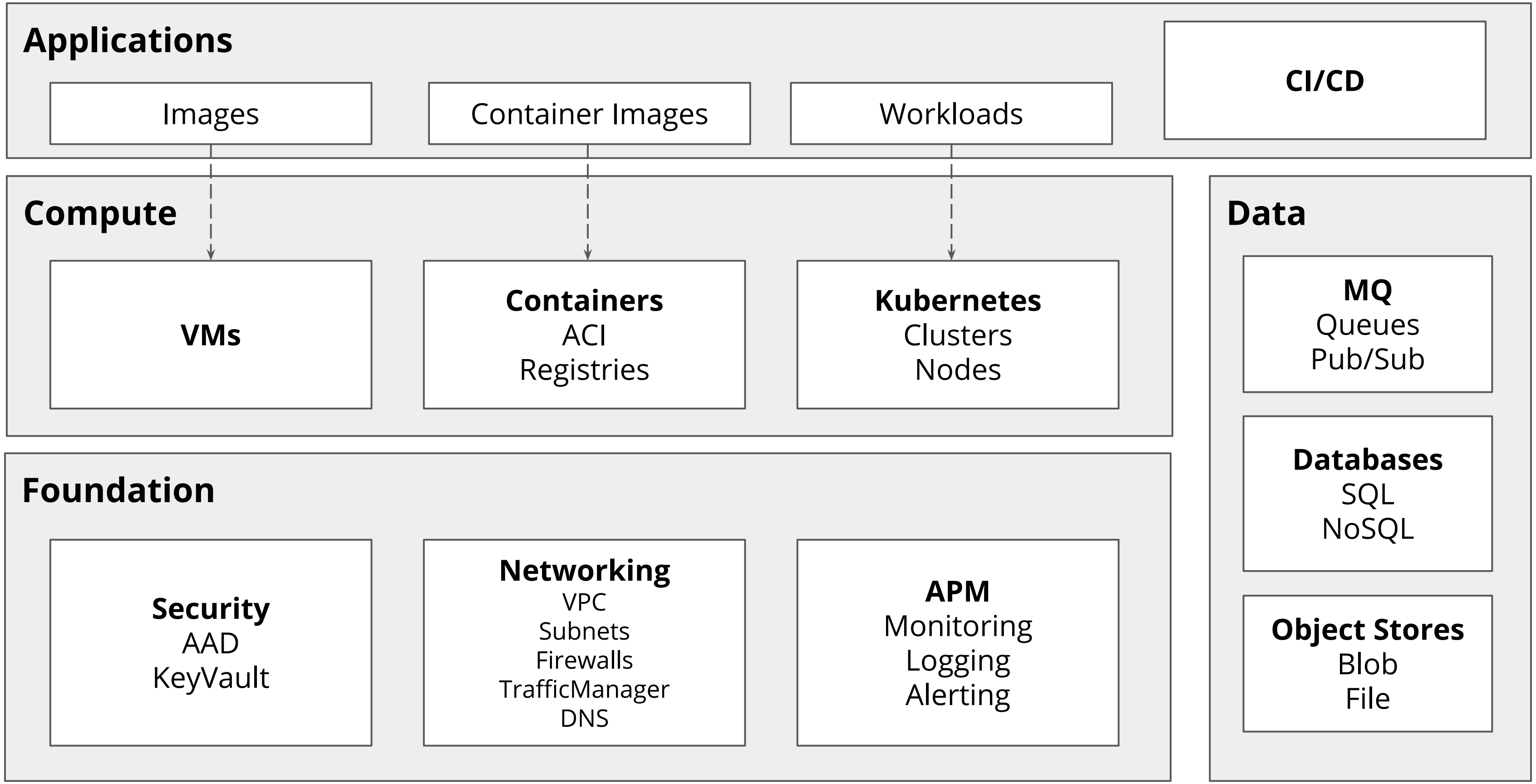
Pulumi Getting Started (hands-on) - 45 mins

- Azure login
- Pulumi SaaS login
- New project and get deployed

Wrap up and Q&A - 15 mins

Infrastructure as Code Concepts

Infrastructure Landscape



Concepts

So, your application needs infrastructure resources (VM, database, cluster, queue, etc). How do you create and manage them?

- **Manual**: point and click to create/modify resources in the console.
- **Ad-hoc automation**: CLI commands or scripts to create/modify resources.
- **Infrastructure as code**:
  - **Provisioning**: declaratively create/modify resources.
  - **Configuration**: change state of an existing resource post-provisioning.

Philosophical difference between immutable and mutable infrastructure (cattle vs pets).

- VMs are usually roses (pets)
- Containers and serverless are usually corn (cattle)

Infrastructure as Code (JSON)

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
    "location": {
      "type": "string",
      "defaultValue": "[resourceGroup().location]"
    }
  },
  "resources": [
    {
      "type": "Microsoft.Storage/storageAccounts",
      "apiVersion": "2019-04-01",
      "name": "mystorage1234bead",
      "location": "[parameters('location')]",
      "sku": {
        "name": "Standard_LRS"
      },
      "kind": "StorageV2",
      "properties": {
        "supportsHttpsTrafficOnly": true
      }
    }
  ]
}
```

Infrastructure as Code (Language)

```
from pulumi_azure_native import storage

# Create an Azure resource (Storage Account)
account = storage.StorageAccount('sa',
    resource_group_name=resource_group.name,
    sku=storage.SkuArgs(
        name=storage.SkuName.STANDARD_LRS,
    ),
    kind=storage.Kind.STORAGE_V2)
```

Infrastructure as Code (Language)

```
from pulumi_azure_native.storage import storage_account
from pathlib import Path

account = storage.StorageAccount('sa',
    resource_group_name=resource_group.name,
    sku=storage.SkuArgs(name=storage.SkuName.STANDARD_LRS),
    kind=storage.Kind.STORAGE_V2)

container = storage.BlobContainer('container',
    resource_group_name=resource_group.name,
    account_name=account.name,
    container_name='files')

files = Path('my_directory/')
for file in files.iterdir():
    blob = storage.Blob('blob',
        account_name=account.name,
        container_name=container.name,
        name=file,
        type='Block',
        source=pulumi.FileAsset(file))
```

Using Real Languages

Full power of real languages

- Control flow: loops, conditionals.
- Abstraction and reuse: functions, classes, packages.
- Share and reuse, don't copy and paste.
- ****Still a declarative engine**** ⇐ ***important***

Leverage existing tools, communities, and best practices

- Authoring: IDEs, linters, test frameworks, etc.
- Online communities, training, books, knowledge bases.

Easier for developers and infrastructure engineers to collaborate.

What is Pulumi?

The Pulumi Platform

Create, deploy, and manage modern cloud apps and infrastructure

PULUMI SDK

Open Source Tools and Framework

A multi-language, multi-cloud open source SDK.

Create modern cloud native software using containers, serverless, and hosted infrastructure.

The power of infrastructure as code, the flexibility of real languages you know and love, with reusable packages.

PULUMI CONSOLE

Team and Enterprise SaaS

Community, Team, and Enterprise tiers for any team.

Continuously deliver and manage cloud apps and infrastructure with necessary controls and guardrails.

Developers and Operators meet on common ground with flexible workflows and policies for teams of all sizes.

Pulumi Console

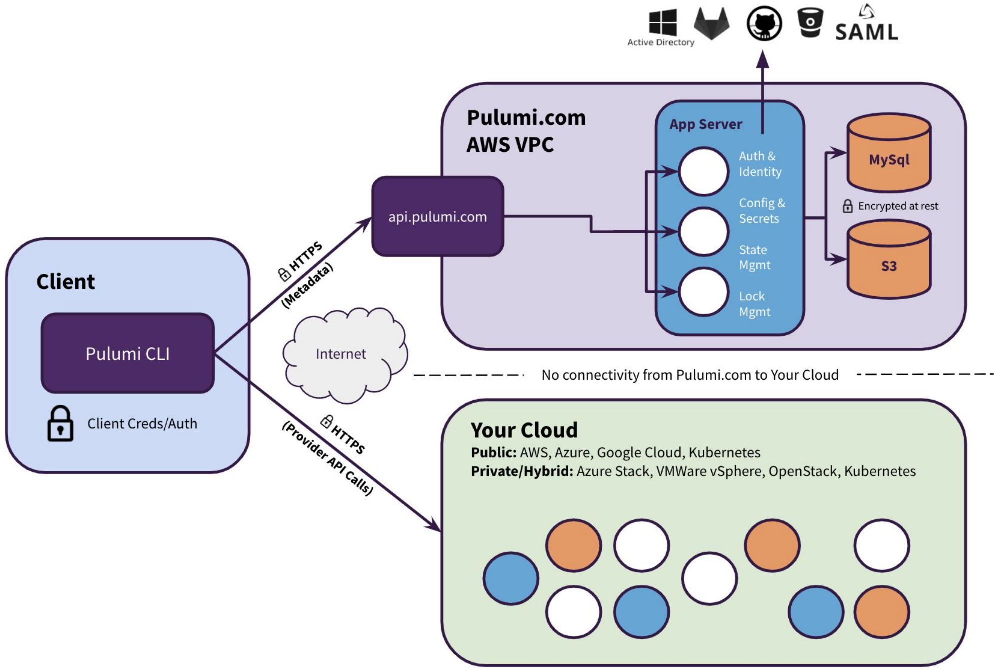
Pulumi Console

FOR INDIVIDUALS, TEAMS AND ENTERPRISES

Production Ready:
Highly available hosted infrastructure across multiple regions/AZs.

Modern & Flexible:
Built on Docker, AWS, and Pulumi itself.

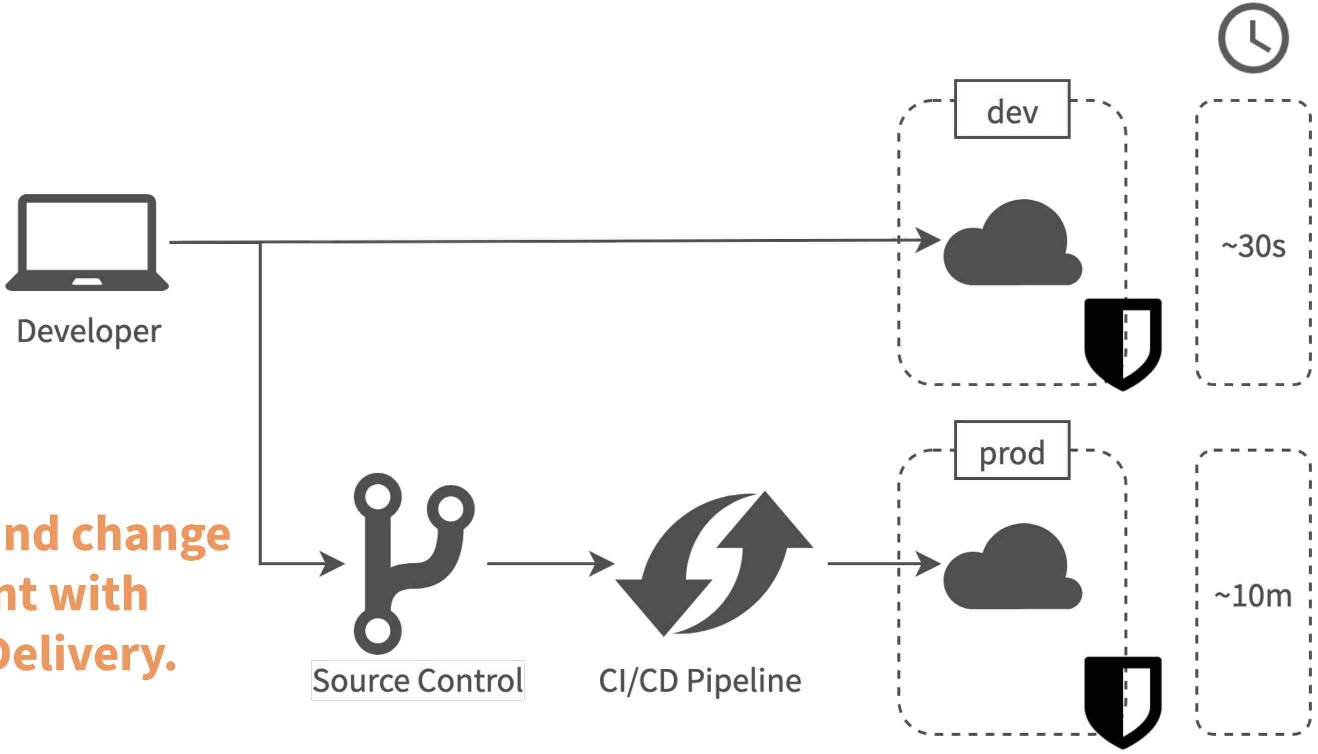
Easy & Fast:
Robust state and lock management for teams of all sizes.



Developer Deployment Process

Allow developers to iterate quickly on dev environments.

Gain visibility and change management with Continuous Delivery.



Exercises

Prerequisites

Goal: Have a working Pulumi-Azure-Python environment

- Install Pulumi
  - Mac: **`brew install pulumi`**
  - Windows: **`choco install pulumi`**
  - Mac|Linux: **`curl -fsSL https://get.pulumi.com | sh`**
- Install Python3 (3.6+)
- Install az command line and login
  - **`az login`**

Exercise (Done Together as a Class)

1. Create a folder:
   - `mkdir test-project && cd test-project`
2. Create a simple project
   - `pulumi new azure-python`
3. Review the code
4. Deploy the stack
   - `pulumi up`
5. Familiarize yourself with the console view of the project and stack
6. Destroy the stack
   - `pulumi destroy`