Hands-On Infrastructure as Code Workshop

Joe Duffy <joe@pulumi.com> @funcOfJoe
Paul Stack <paul@pulumi.com> @stack72
Mikhail Shilkov <code@mikhail.io> @MikhailShilkov

<City>
<Date>

Course Outline

Course Outline

Part 1 — Infrastructure as Code Concepts (90m)

- overview
- one end to end lab

Part 2 — Modern Cloud Architectures (90m)

- four labs:
  - VMs
  - containers (on ECS)
  - containers (on Kubernetes)
  - serverless

Infrastructure as Code Concepts

30m Overview
45m Hands-On Labs
15m Break

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 pets.
- containers and serverless are usually cattle.

Configuration ⇔ Provisioning

Scenario: need to upgrade web server from V1 to V2.

The configuration way

- Chef, Puppet, Ansible, Salt, etc — upgrade server in place
- What if it fails part-way through? Have we tested all combinations?

The provisioning way

- deploy a new instance, update dependants,
- Testable in advance. If it fails, we won't have deployed anything.

Can be used together!

Today, we will focus on provisioning.

IaC Landscape

Cloud provider tools

- AWS CloudFormation and CDK
- Azure Resource Manager (ARM) Templates
- Google Deployment Manager

Cloud independent tools

- Kubernetes YAML
- Helm YAML/templates
- HashiCorp Terraform
- **Pulumi (what we will be using today)**

Infrastructure as Code (IaC)

Declare infrastructure as “code,” using:

- markup languages (YAML/JSON).
- markup templating (pre-/post-processing).
- domain-specific languages (DSLs).
- general purpose (“real”) languages.

Benefits:

- automatable.
- repeatable.
- review and version like code (often in Git).

Infrastructure as Code (JSON)

```
{
  "AWSTemplateFormatVersion" : "2010-09-09",
  "Resources" : {
    "EC2Instance" : {
      "Type" : "AWS::EC2::Instance",
      "Properties" : {
        "InstanceType" : "t2.micro",
        "SecurityGroups" : [ { "Ref" : "InstanceSecurityGroup" } ],
        "ImageId" : { "ami-0080e4c5bc078760e" },
      }
    },
    "InstanceSecurityGroup" : {
      "Type" : "AWS::EC2::SecurityGroup",
      "Properties" : {
        "GroupDescription" : "Enable HTTP over port 80",
        "SecurityGroupIngress" : [{
          "IpProtocol" : "tcp", "FromPort" : "80", "ToPort" : "80", "CidrIp" : "0.0.0.0/0"
        }]
      }
    }
  }
}
```

Infrastructure as Code (DSL)

```
provider aws {
  region = "eu-central-1"
}

resource "aws_security_group" "web_sg" {
  description = "Enable HTTP over port 80"
  ingress {
    protocol    = "tcp"
    from_port   = 80
    to_port     = 80
    cidr_blocks = [ "0.0.0.0/0" ]
  }
}

resource "aws_instance" "web" {
  ami           = "ami-0080e4c5bc078760e"
  instance_type = "t2.micro"
  security_groups = [ "$(aws_security_group.web_sg.id)" ]
}
```

Infrastructure as Code (Language)

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

let group = new aws.ec2.SecurityGroup("web-sg", {
    description: "Enable HTTP over port 80",
    ingress: [
        { protocol: "tcp", fromPort: 80, toPort: 80, cidrBlocks: ["0.0.0.0/0"] },
    ],
});

let server = new aws.ec2.Instance("web", {
    instanceType: "t2.micro",
    securityGroups: [ group.id ],
    ami: "ami-0080e4c5bc078760e",
});
```

Infrastructure as Code (Language 💡)

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

let group = new aws.ec2.SecurityGroup("web-sg", {
    description: "Enable HTTP over port 80",
    ingress: [
        { protocol: "tcp", fromPort: 80, toPort: 80, cidrBlocks: ["0.0.0.0/0"] },
    ],
});

for (let az in aws.getAvailabilityZones().names) {
    let server = new aws.ec2.Instance(`web-${az}`, {
        instanceType: "t2.micro",
        securityGroups: [ group.id ],
        ami: "ami-0080e4c5bc078760e",
        availabilityZone: az,
    });
}
```

Using Real Languages

Full power of real languages:

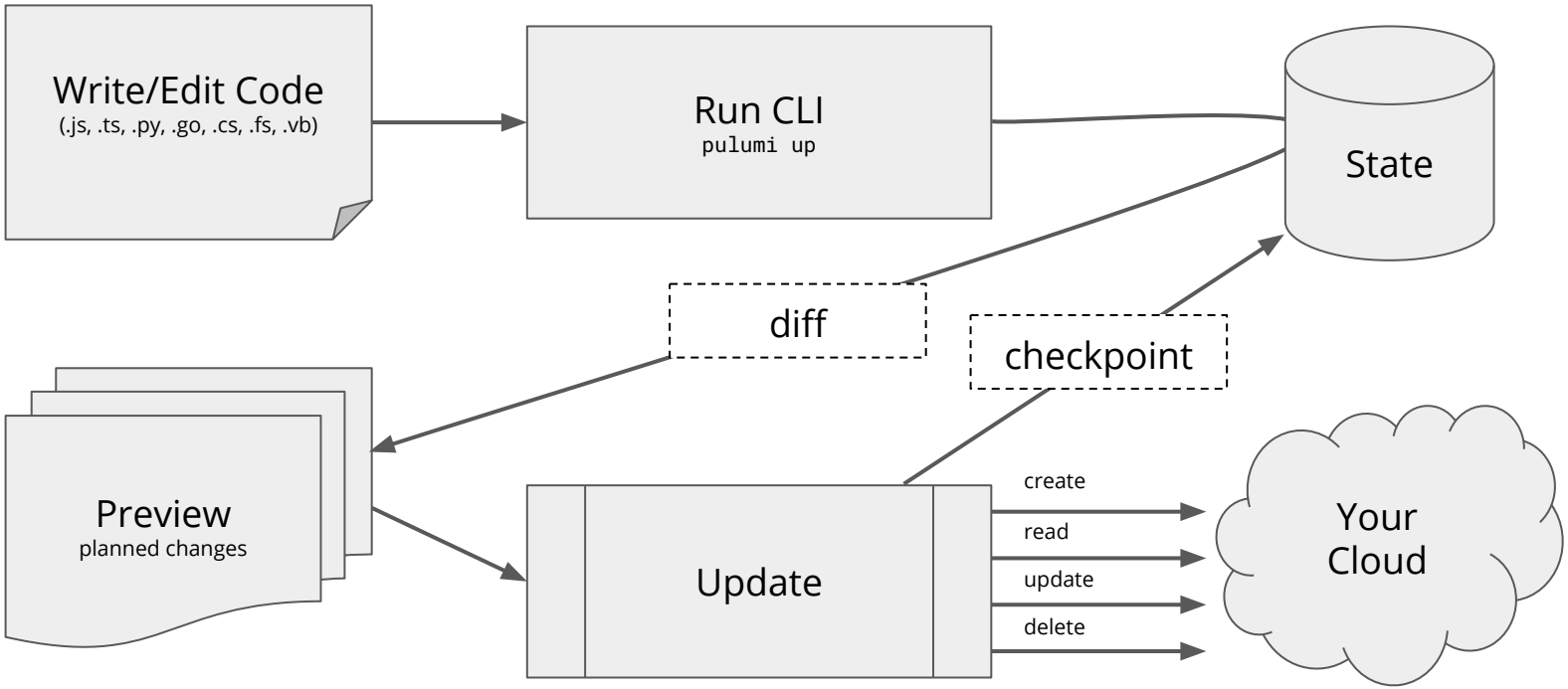
- control flow: loops, conditionals.
- abstraction and reuse: functions, classes, packages.
- share and reuse, don’t copy and paste.
- ****still declarative** ⇐ *important***

Leverage existing tools, communities, and best practices:

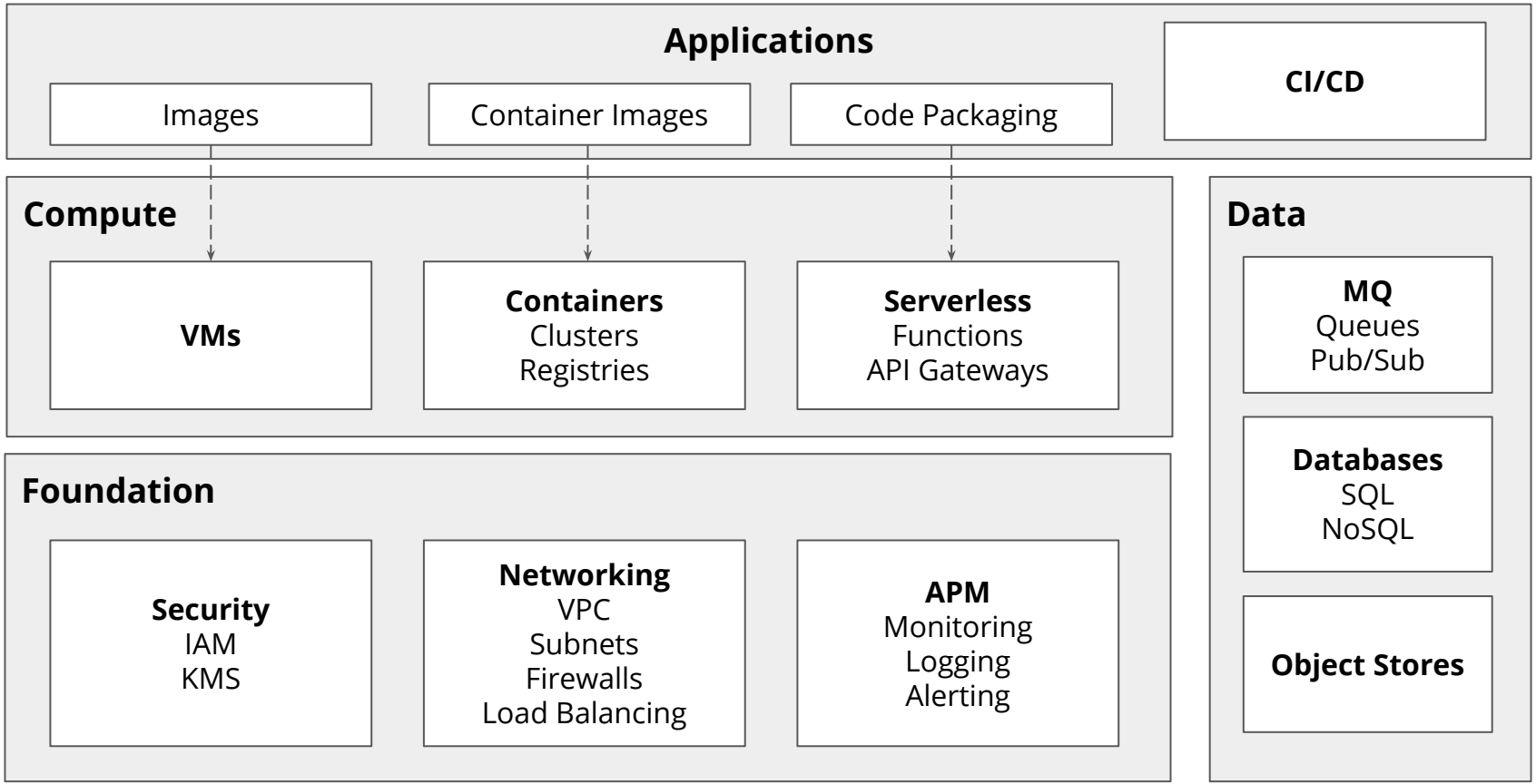
- authoring: IDEs, linters, test frameworks, etc.
- online communities, training, books, knowledge bases.

Easier for developers and infrastructure engineers to collaborate.

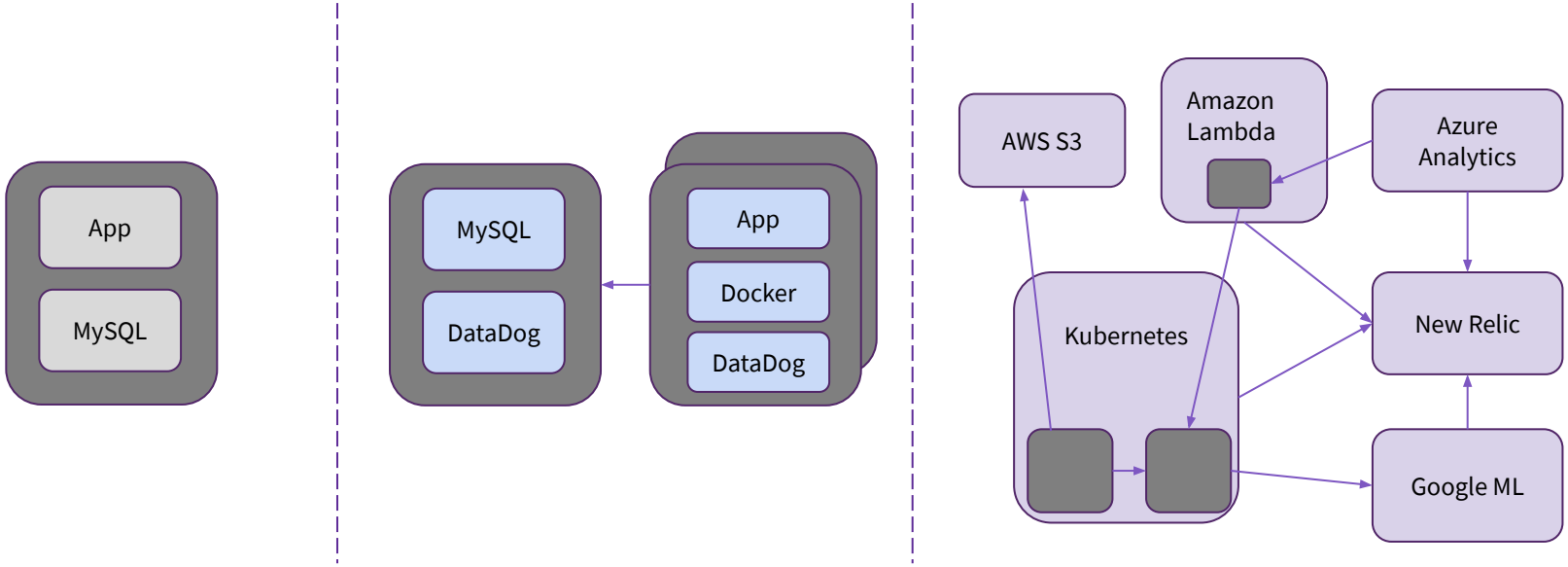
Pulumi Workflow



Infrastructure Landscape



Cloud Transition



EARLY CLOUD
Mostly Static
N-Tier Apps
VMs
Private Cloud

CURRENT CLOUD
Partly Dynamic
Less Monolithic
VMs and Containers*
Hybrid - Public/Private

FUTURE CLOUD
Fully Dynamic
Hyper-Connected Services
Containers and Serverless
Mostly Public Cloud

*Experimentation

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

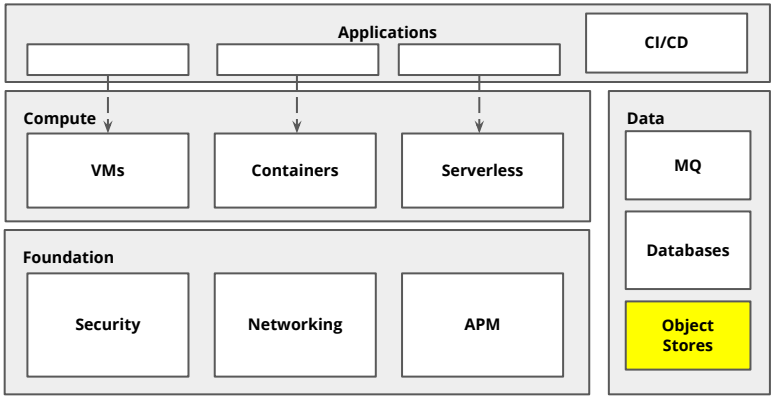
Hands-On Labs Part 1

30m Overview
45m Hands-On Labs
15m Break

- Creating a New Project
- Configuring AWS
- Provisioning Infrastructure
- Updating Your Infrastructure
- Making Your Stack Configurable
- Creating a Second Stack
- Destroying Your Infrastructure

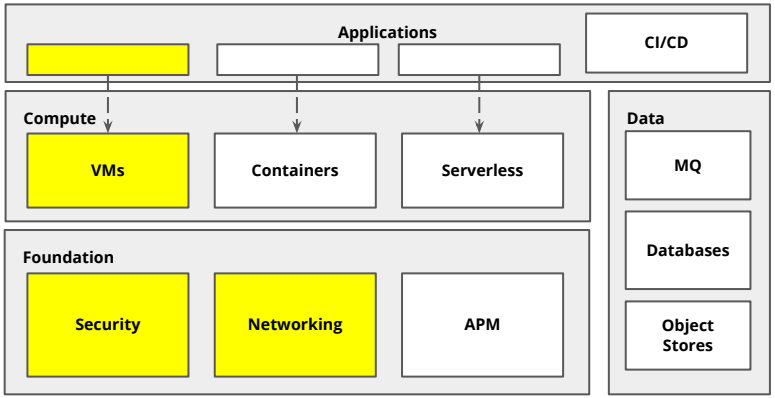
Lab 1 — Modern Infrastructure as Code

- Create a New Project
- Configure AWS
- Provision Infrastructure
- Update Infrastructure
- Make Your Stack Configurable
- Create a Second Stack
- Destroy Your Infrastructure



Lab 2.1 — Provisioning EC2 Virtual Machines

- Create a VM and Access It
- Scale Out Multiple VMs, One Per Availability Zone
- Load Balance Traffic Across Them

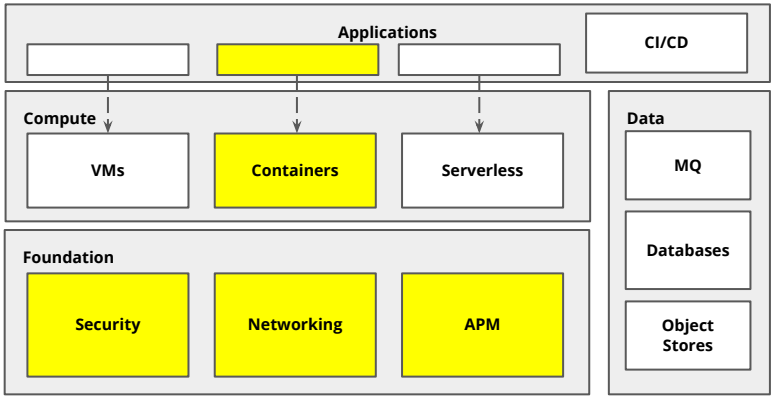


Lab 2.2 — Deploying Containers to ECS "Fargate"

- Create an ECS Cluster
- Create a Load-Balanced “Nginx” Service
- Build and Publish a Custom, Private Container Image
- Do a Rolling Deployment
- Scale Out the Service

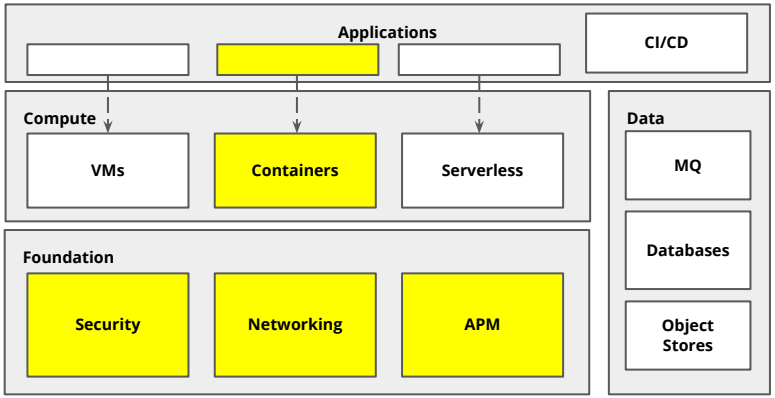
Special prerequisites:

- Docker



Lab 2.3 — Deploying Containers to a Kubernetes Cluster

- Connect to a Kubernetes Cluster
- Create a Kubernetes Namespace
- Create a Kubernetes Deployment
- Create a Load-Balanced Kubernetes Service
- Do a Rolling Deployment
- Scale Out the Service

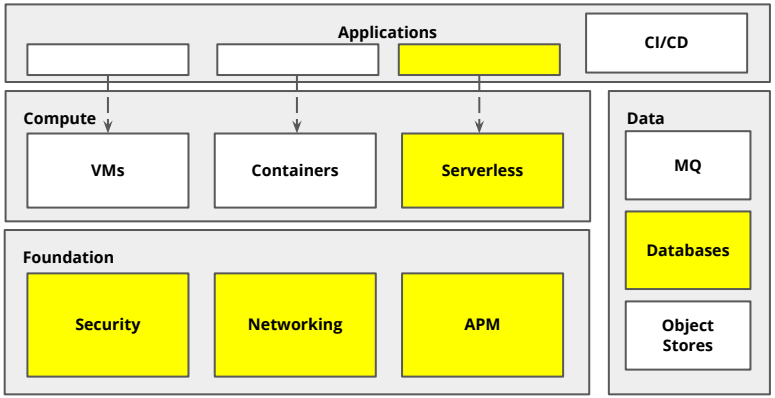


Special prerequisites:

- kubectl
- aws-iam-authenticator
- KUBECONFIG=iac-workshop-creds.tgz/kubeconfig

Lab 2.4 — Using AWS Lambda for Serverless Patterns

1. Create a Serverless DynamoDB Table
2. Create IAM Policies
3. Create a Serverless AWS Lambda-Based API Gateway
4. Refactor to Use Simpler “Inline” Application Code Approach



Prerequisites

Download credentials from: **http://<snip/>**

From https://github.com/pulumi/infrastructure-as-code-workshop/blob/master/labs/00-installing-prerequisites.md ...

Make sure you have:

- AWS CLI: https://docs.aws.amazon.com/cli/latest/userguide/cli-chap-install.html
  - Run **`aws configure`** to input credentials
  - Choose **`eu-central-1`** as your AWS region
- Pulumi: https://www.pulumi.com/docs/get-started/install/
- Node.js: https://nodejs.org/en/download/
- An editor of choice (VS Code works great https://code.visualstudio.com/download)

For 2nd set of labs, if you plan to do Docker or Kubernetes tutorials:

- Docker CE: https://docs.docker.com/install/
- Kubectl CLI: https://kubernetes.io/docs/tasks/tools/install-kubectl/
- AWS IAM Authenticator: https://docs.aws.amazon.com/eks/latest/userguide/install-aws-iam-authenticator.html

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

Hands-On Labs Part 1

30m Overview
70m Hands-On Labs
15m Break

- Creating a New Project
- Configuring AWS
- Provisioning Infrastructure
- Updating Your Infrastructure
- Making Your Stack Configurable
- Creating a Second Stack
- Destroying Your Infrastructure

Break

30m Overview
45m Hands-On Labs
15m Break

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

Hands-On Labs Part 2

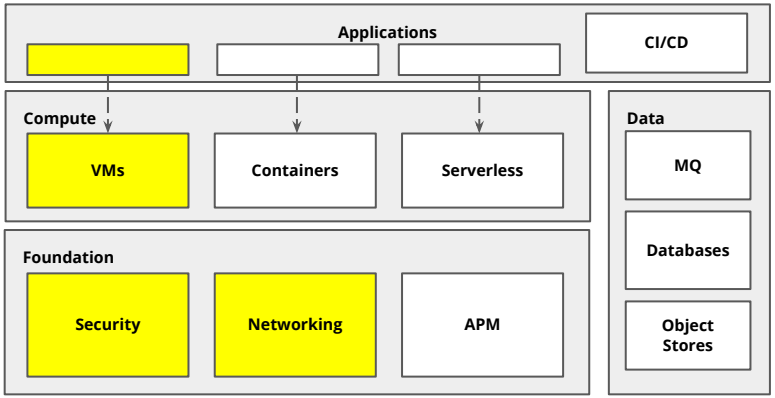
80m Hands-On Labs

10m Recap

- Provisioning EC2 Virtual Machines
- Deploying Containers to ECS “Fargate”
- Deploying Containers to a Kubernetes Cluster
- Using AWS Lambda for Serverless Application Patterns

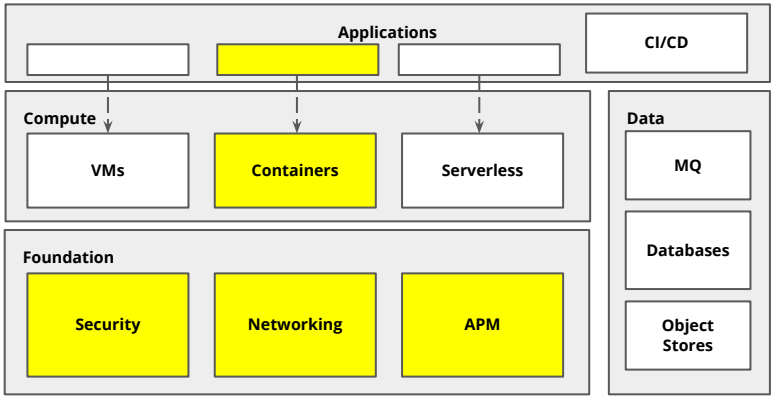
Lab 2.1 — Provisioning EC2 Virtual Machines

- Create a VM and Access It
- Scale Out Multiple VMs, One Per Availability Zone
- Load Balance Traffic Across Them



Lab 2.2 — Deploying Containers to ECS "Fargate"

- Create an ECS Cluster
- Create a Load-Balanced “Nginx” Service
- Build and Publish a Custom, Private Container Image
- Do a Rolling Deployment
- Scale Out the Service

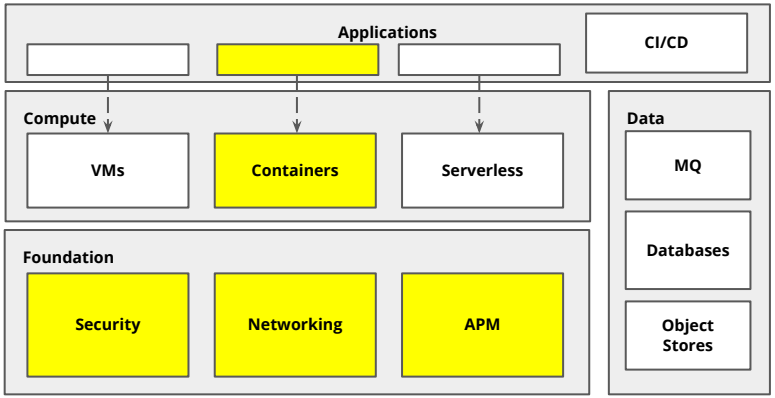


Special prerequisites:

- Docker

Lab 2.3 — Deploying Containers to a Kubernetes Cluster

- Connect to a Kubernetes Cluster
- Create a Kubernetes Namespace
- Create a Kubernetes Deployment
- Create a Load-Balanced Kubernetes Service
- Do a Rolling Deployment
- Scale Out the Service

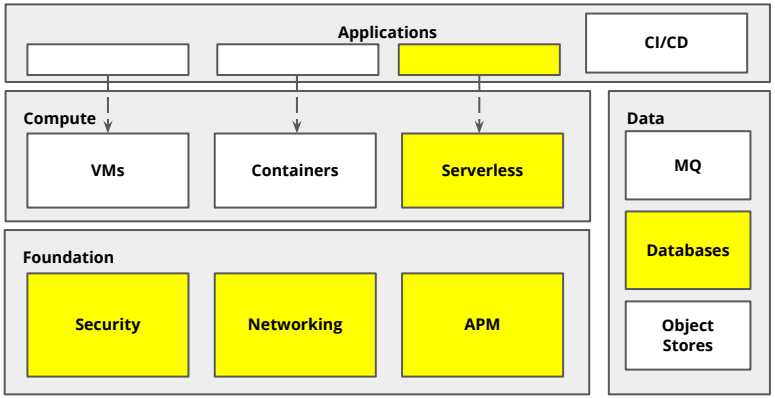


Special prerequisites:

- kubectl
- aws-iam-authenticator
- KUBECONFIG=iac-workshop-creds.tgz/kubeconfig

Lab 2.4 — Using AWS Lambda for Serverless Patterns

1. Create a Serverless DynamoDB Table
2. Create IAM Policies
3. Create a Serverless AWS Lambda-Based API Gateway
4. Refactor to Use Simpler “Inline” Application Code Approach



Recap

80m Hands-On Labs

10m Recap

Finishing Labs

All available on GitHub
https://github.com/pulumi/infrastructure-as-code-workshop

- Infrastructure as Code Lab
- Modern Application Architecture Labs
  - Provisioning EC2 Virtual Machines
  - Deploying Containers to Elastic Container Service (ECS) "Fargate"
  - Deploying Containers to a Kubernetes Cluster
  - Using AWS Lambda for Serverless Application Patterns

Possible Next Steps

Complete additional labs (all open source).

- Additional providers:
  - AWS, Azure, GCP, DigitalOcean
  - Kubernetes
  - vSphere, OpenStack, F5 BigIP
  - Datadog, NewRelic, GitHub, GitLab
  - more....
- Use secrets management.
- Multi-project infrastructure architectures.
- Import some existing infrastructure.
- Convert some HCL to Pulumi! **https://github.com/pulumi/tf2pulumi**
- Policy as Code.
- Continuous delivery (e.g., triggered by Git commit).

Q&A

Thank you.

Pulumi documentation: **https://pulumi.com/docs**
- Getting started: **https://www.pulumi.com/docs/get-started/**
- 100+ additional tutorials: **https://www.pulumi.com/docs/tutorials/**

Pulumi examples repo: **https://github.com/pulumi/examples**
Community Slack: **https://slack.pulumi.com**
Workshop: **https://github.com/pulumi/infrastructure-as-code-workshop**

Joe Duffy <joe@pulumi.com> @funcOfJoe
Paul Stack <paul@pulumi.com> @stack72
Mikhail Shilkov <code@mikhail.io> @MikhailShilkov