SERVERLESS + CONTAINERS = MODERN CLOUD APPLICATIONS

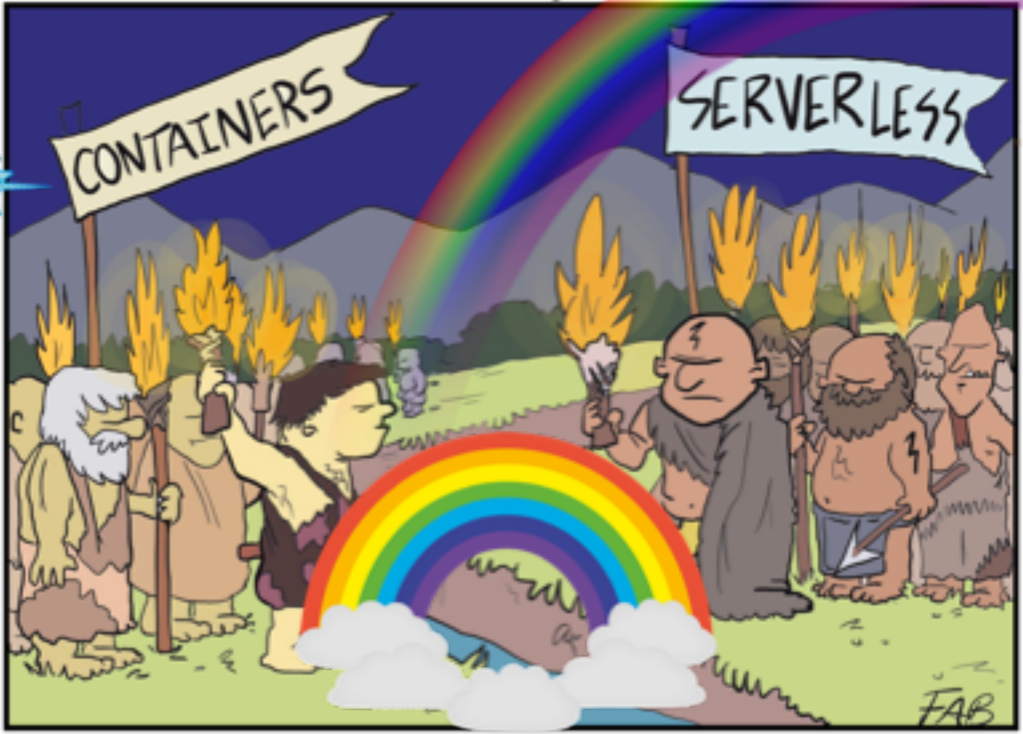
Donna Malayeri

Product Manager, Pulumi

@PulumiCorp

@lindydonna

FaaS and Furious by Forrest Brazeal A CLOUD GURU



The two tribes regarded each other suspiciously
in the glow of their brightly blazing production environments.

SERVERLESS AND CONTAINERS

- Tradeoff between control and productivity
- Containers give you full control over your compute workloads
- Serverless scales instantly and is cheaper to own and operate
- Modern applications need both compute models

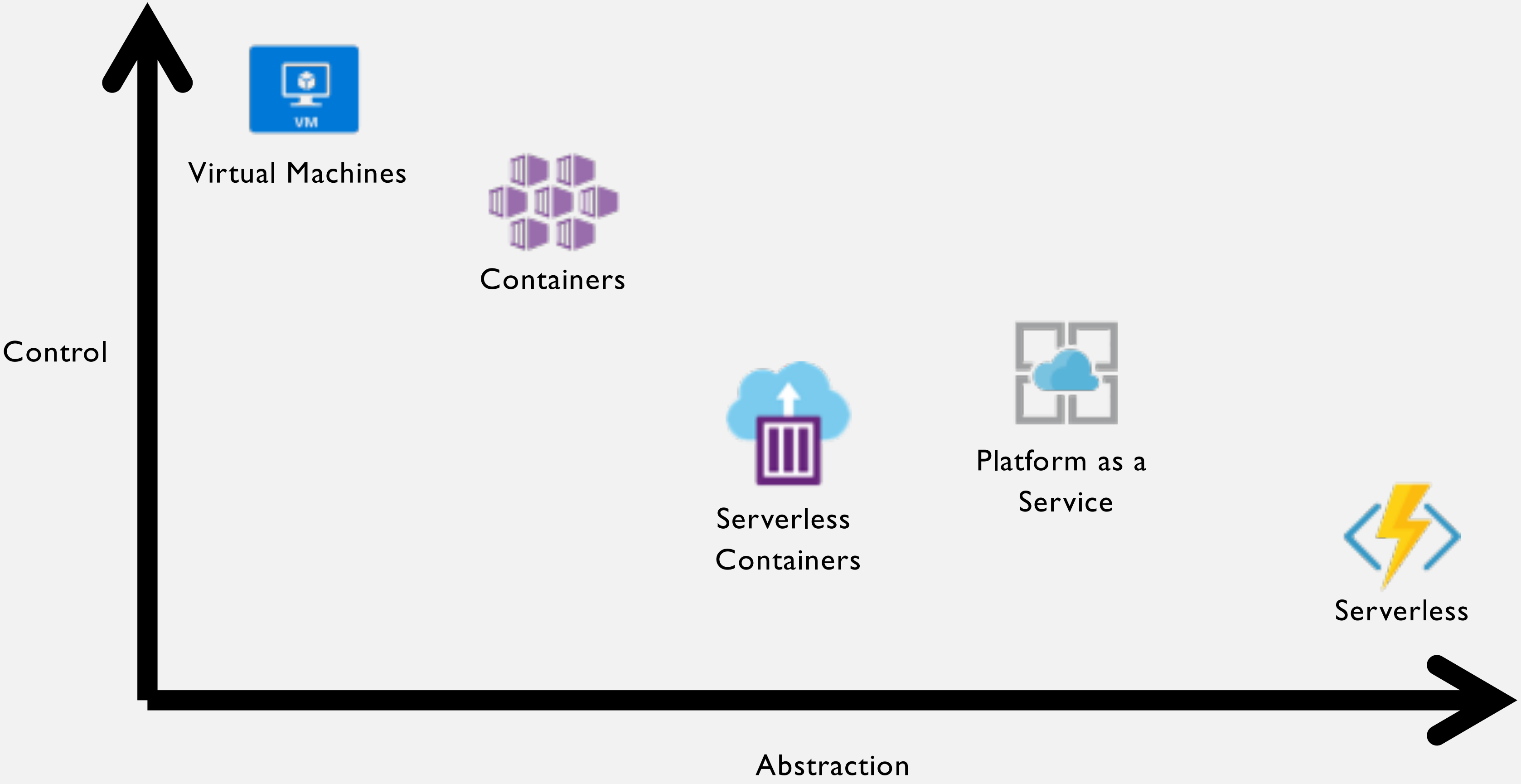
PROGRAMMING IS ABOUT ABSTRACTION

- JavaScript
- Go
- Python
- Ruby
- C#
- Java
- C/C++
- Assembly

If I have seen further it is only by standing on the shoulders of giants.

-- Isaac Newton

The cloud landscape



Virtual Machines

Containers

Serverless Containers

Platform as a Service

Serverless

Control

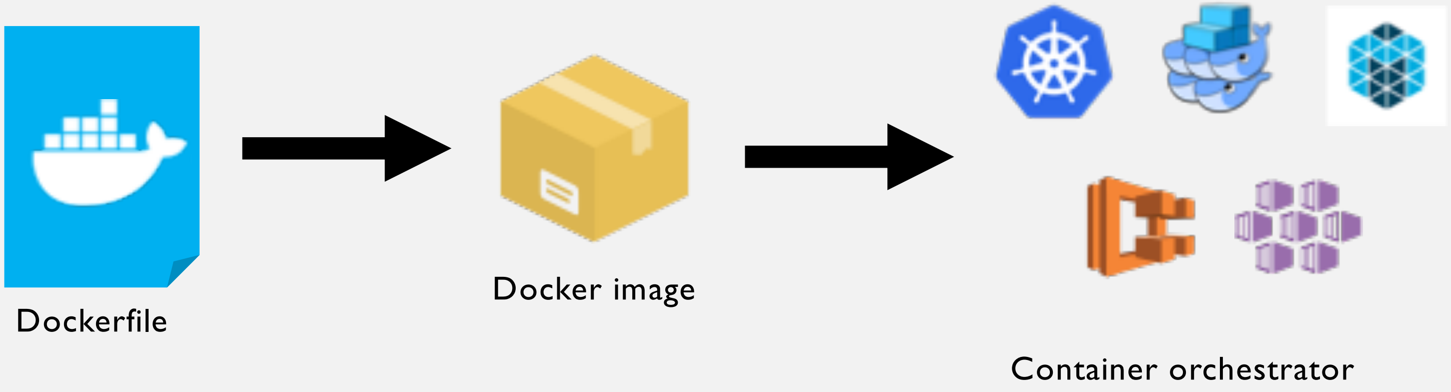
Abstraction

IN THE EARLY DAYS OF CLOUD, THERE WERE ONLY VIRTUAL MACHINES

- How often should I *patch* my server?
  - How *do* I patch?
- How do I deploy *code*?
- How *many* servers do I need?
- How can I *scale* my app?

@lindydonna

CONTAINERS REDUCE COMPLEXITY

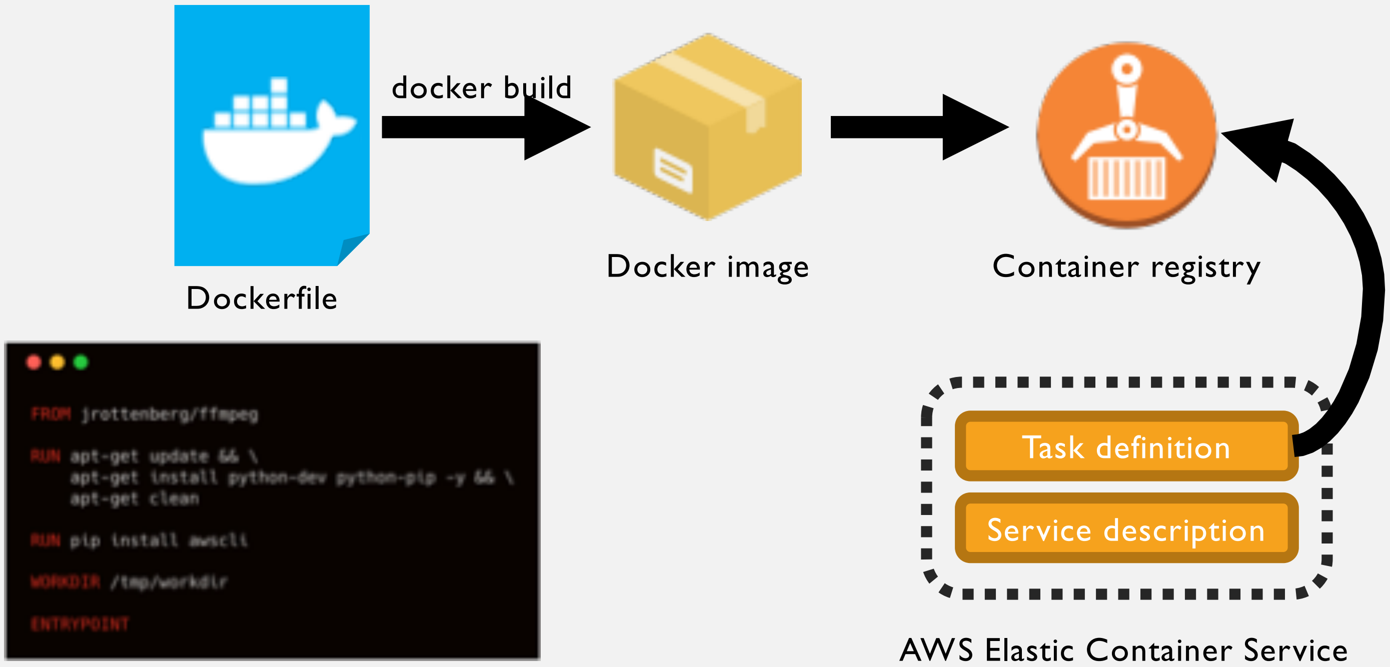


Dockerfile

Docker image

Container orchestrator

CONTAINERS



docker build

Dockerfile

Docker image

Container registry

Task definition

Service description

AWS Elastic Container Service

```
FROM jrottenberg/ffmpeg

RUN apt-get update && \
    apt-get install python-dev python-pip -y && \
    apt-get clean

RUN pip install awscli

WORKDIR /tmp/workdir

ENTRYPOINT
```

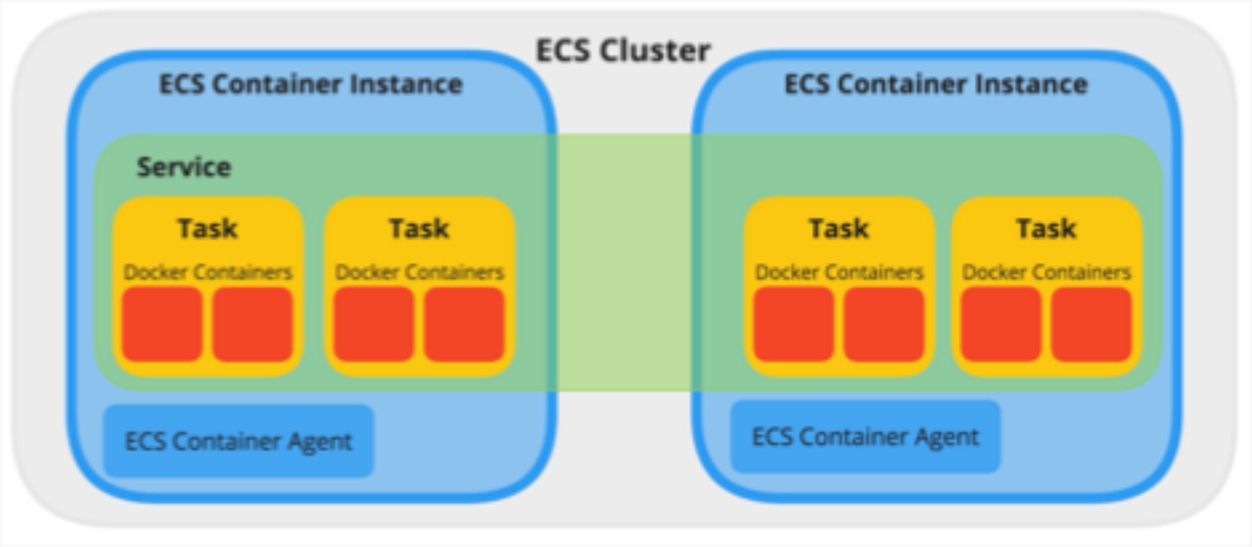
CONTAINER BENEFITS

- Abstraction for compute: containers instead of VMs
- Useful package format
- Full control over application environment
- Full control over task placement
- Control over compute resources

CONTAINERS AT RUNTIME

Task definition
Service description

ECS Service Description

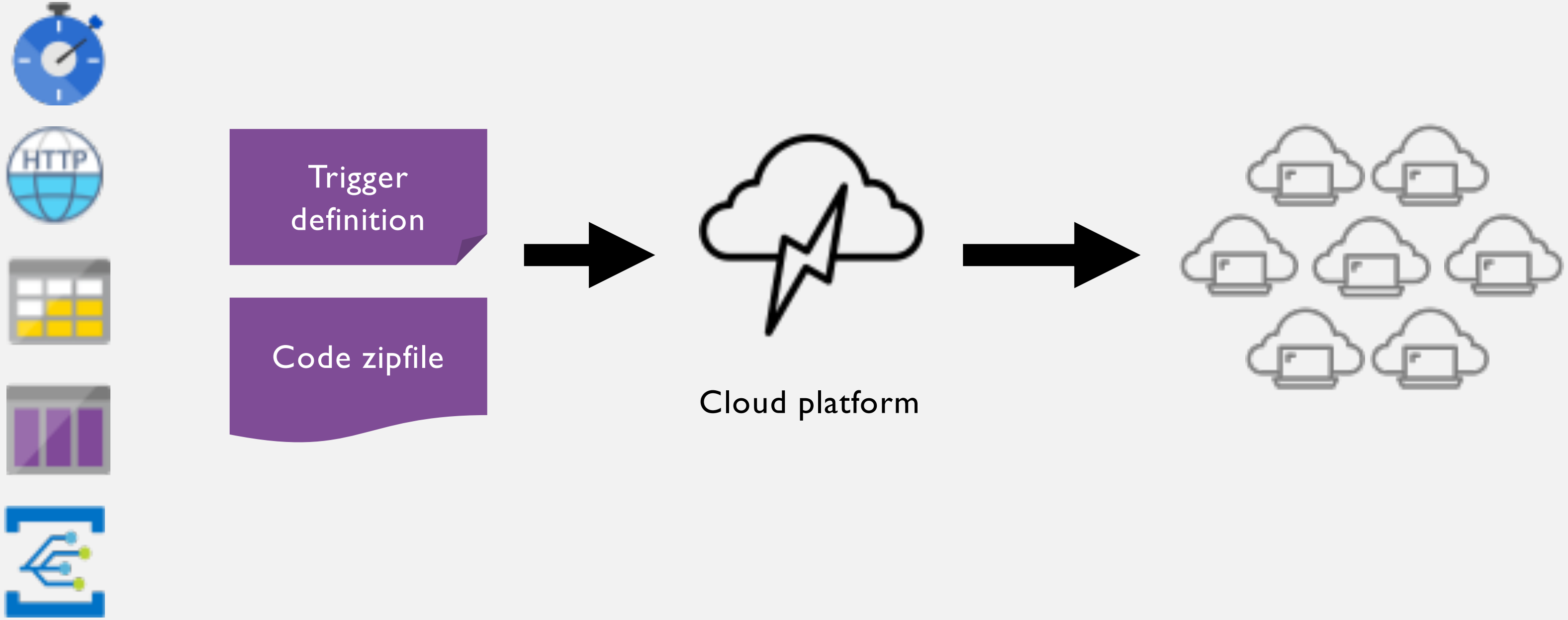


Graphic: https://medium.freecodecamp.org/amazon-ecs-terms-and-architecture-807d8c4960fd

CONTAINERS: THINGS TO MANAGE

- How often should I *update* my Dockerfile dependencies?
- How do I *build* my container images?
- How do I get my containers in *production*?
- How *many* servers do I need?
- How can I *scale* my app?

SERVERLESS: JUST PROVIDE YOUR CODE



Trigger definition

Code zipfile

Cloud platform

Cloud icons: https://www.flaticon.com/authors/payungkead

SERVERLESS

- Event-driven compute with near-instant scale
- Managed, ephemeral compute
- Never pay for idle

(Btw, there are actually servers)

AWS Lambda

Azure Functions

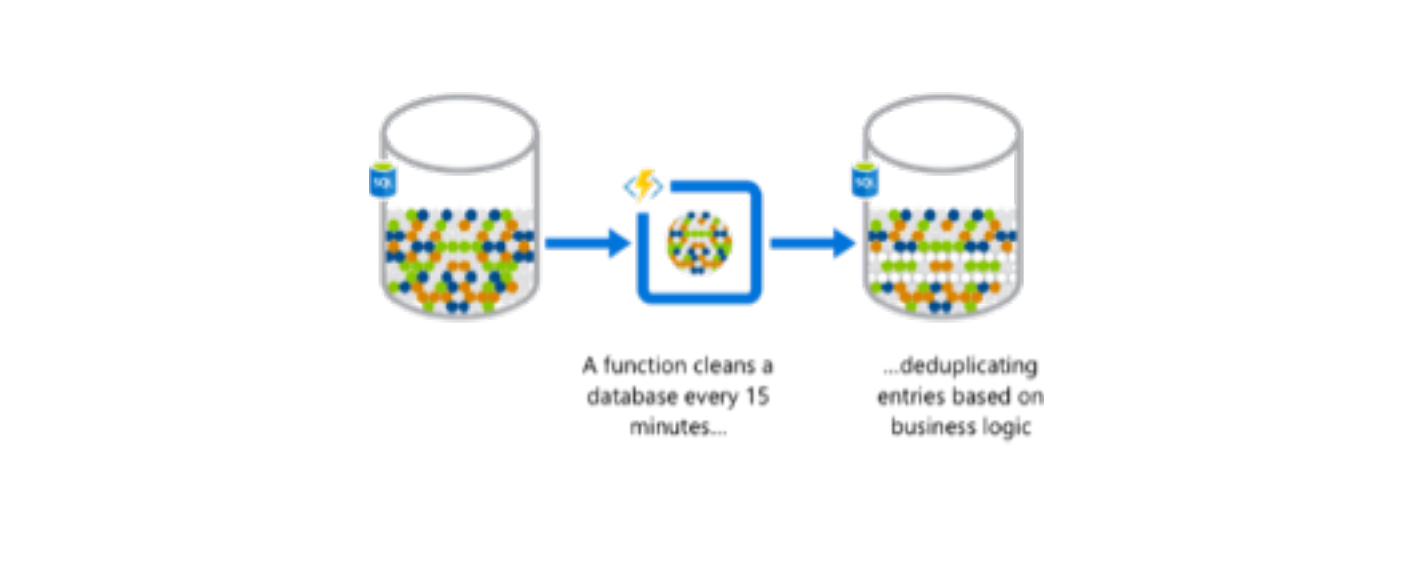
Google Cloud Functions

WHY SERVERLESS?

- Reduce operational overhead
- Faster time to market
- Focus on business value

The Serverless Spectrum https://read.acloud.guru/the-serverless-spectrum-147b02cb2292

SCHEDULED TASKS

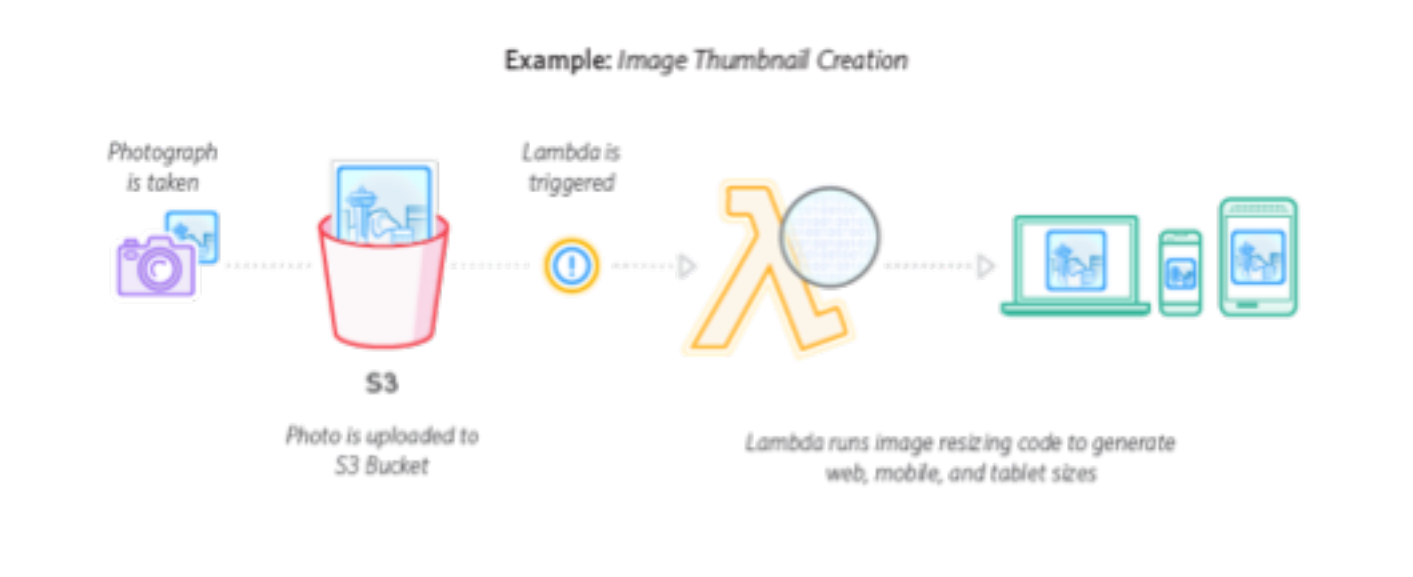


A function cleans a database every 15 minutes...

...deduplicating entries based on business logic

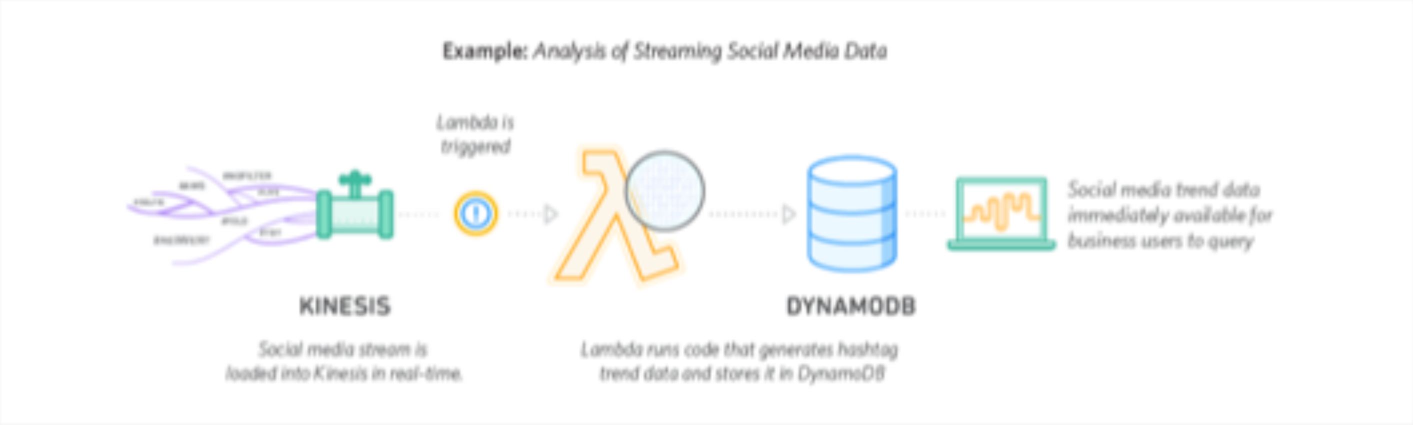
https://functions.azure.com

CREATE IMAGE THUMBNAIL


Example: Image Thumbnail Creation
Photograph is taken
Lambda is triggered
S3
Photo is uploaded to S3 Bucket
Lambda runs image resizing code to generate web, mobile, and tablet sizes

https://aws.amazon.com/lambda/

ANALYZE SOCIAL MEDIA STREAM



Example: *Analysis of Streaming Social Media Data*

Lambda is triggered

KINESIS

Social media stream is loaded into Kinesis in real-time.

DYNAMODB

Lambda runs code that generates hashtag trend data and stores it in DynamoDB

Social media trend data immediately available for business users to query

https://aws.amazon.com/lambda/

SERVERLESS CAVEATS

- Works best for event-based workloads
- Cloud vendor supports specific languages and runtimes
- Can’t customize execution environment
- Not well-suited for long-running tasks

ANALOGY: RENTING VS OWNING A BIKE

NEW CONTAINER EXECUTION MODELS

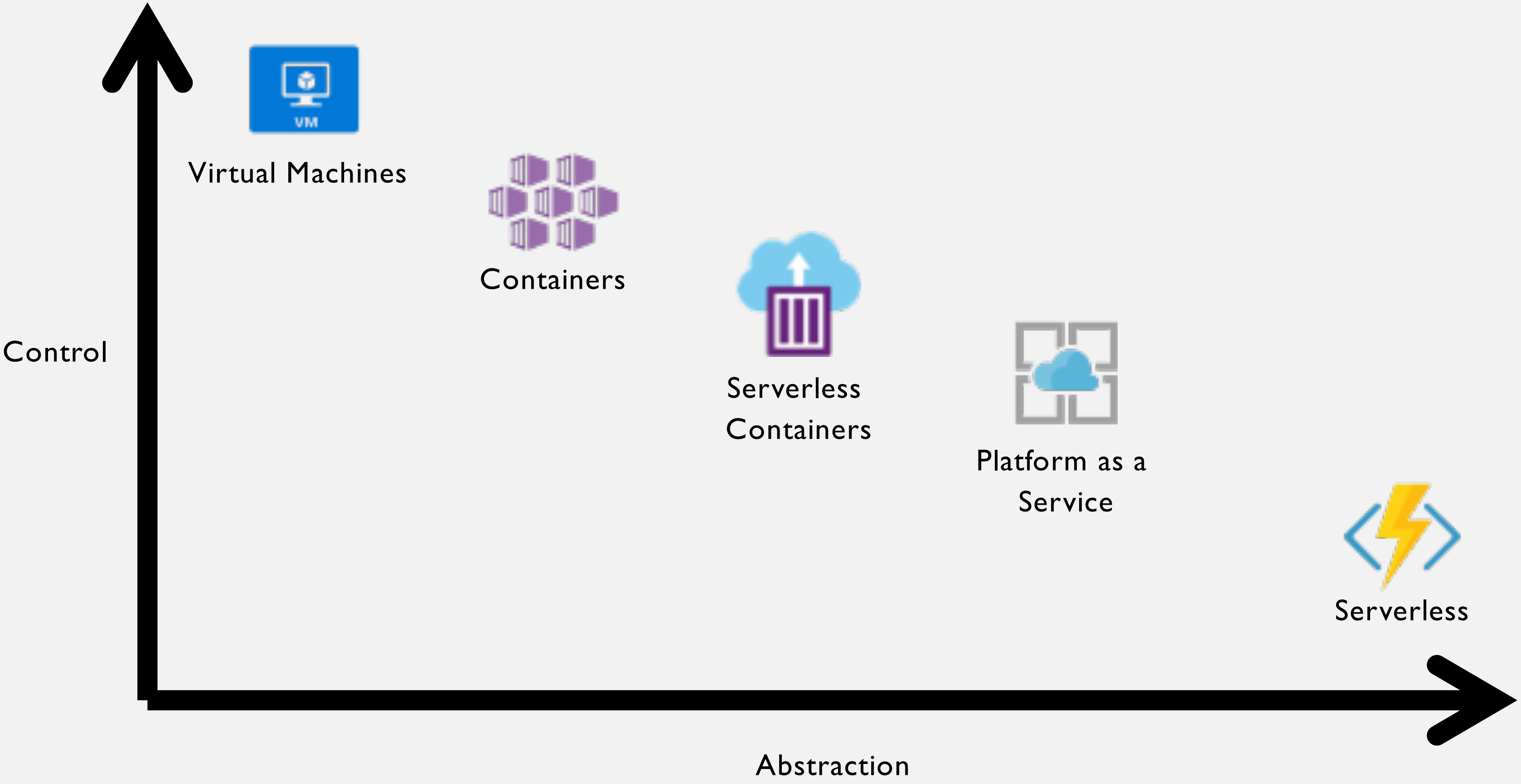
- Azure Container Instances
- AWS Fargate

- On-demand containers
- Don’t have to manage underlying cluster

CONTAINERS AND SERVERLESS

- Use containers for control over the execution environment
  - Customize software and physical servers
  - Great for long-running compute
- Use serverless for event-based compute that scales on demand
  - Less to manage
  - Less to configure

The cloud landscape



Control

Virtual Machines

Containers

Serverless Containers

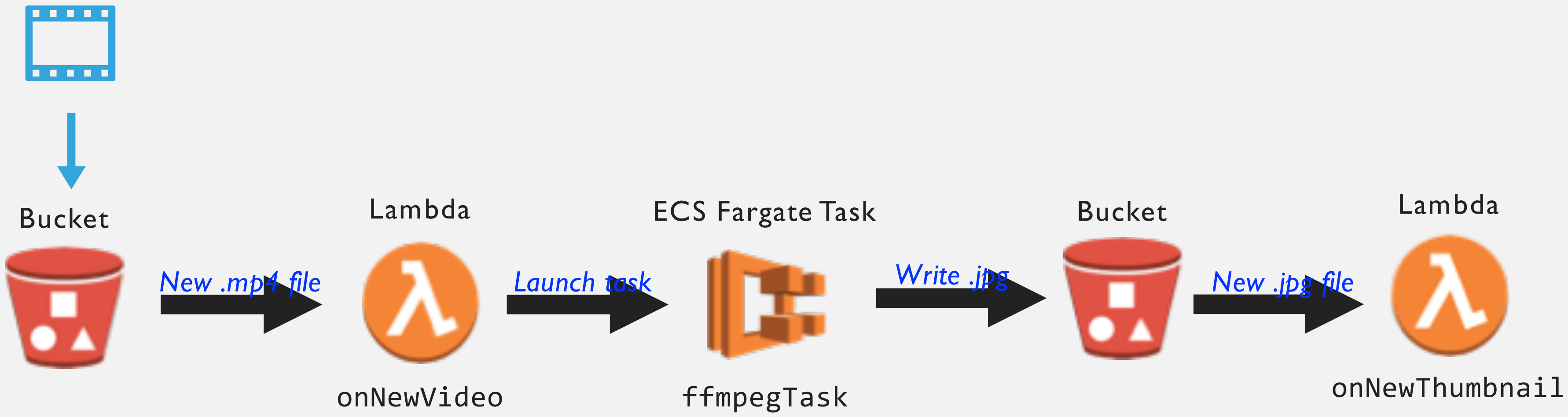
Platform as a Service

Serverless

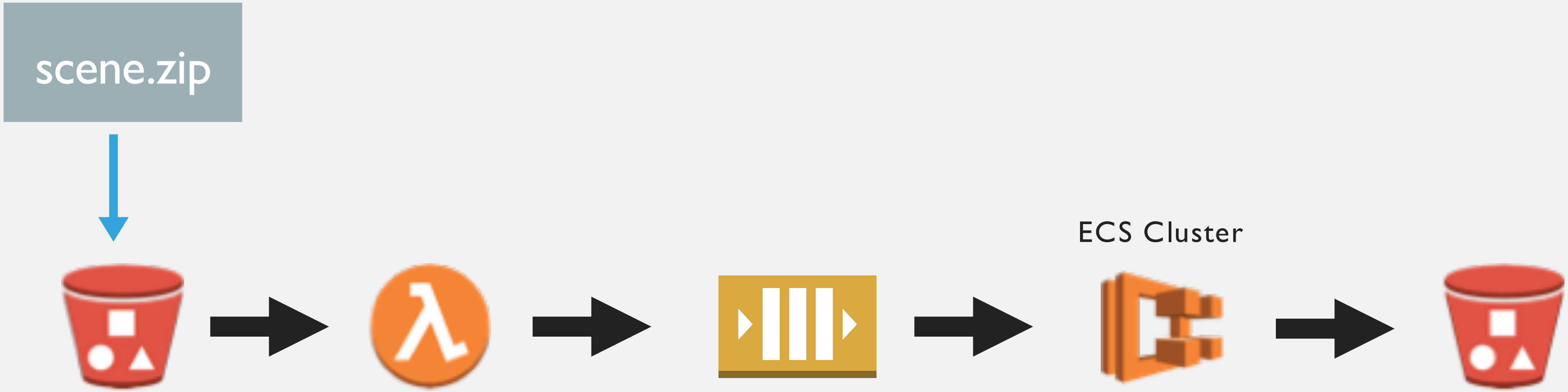
Abstraction

COMBINING THE TWO

EXAMPLE: VIDEO THUMBNAILER



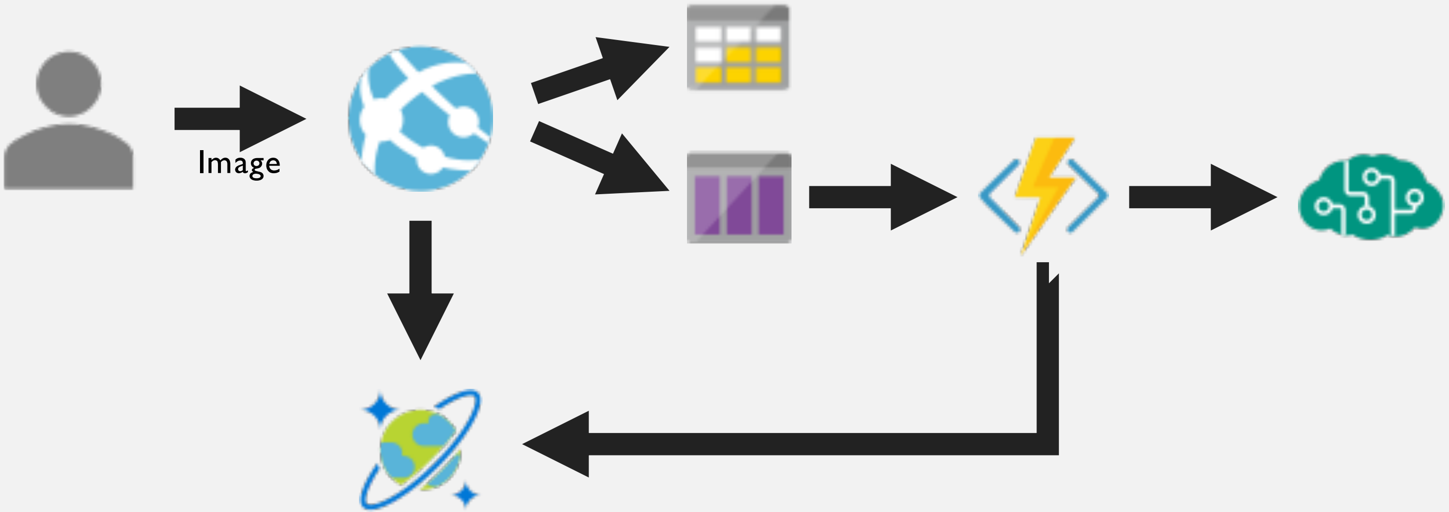
EXAMPLE: RAY TRACING



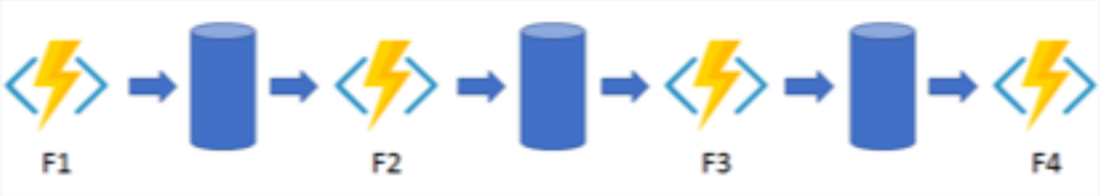
scene.zip

ECS Cluster

EXAMPLE: CONTENT MODERATION

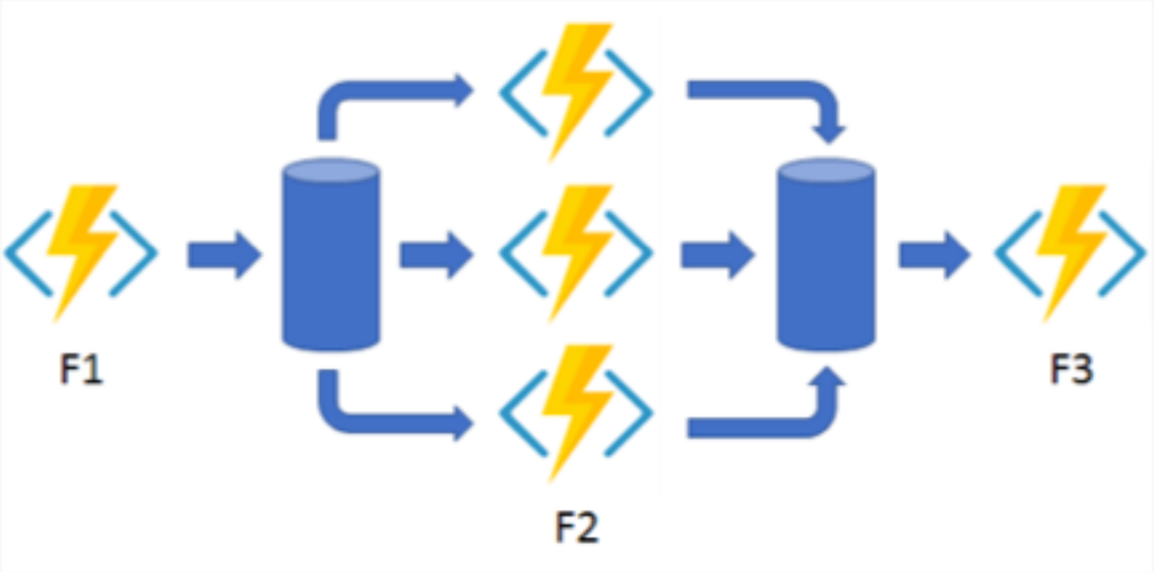


EXAMPLE: FUNCTION CHAINING



```
const df = require("durable-functions");

module.exports = df(function*(ctx) {
    const x = yield ctx.df.callActivityAsync("F1");
    const y = yield ctx.df.callActivityAsync("F2", x);
    const z = yield ctx.df.callActivityAsync("F3", y);
    return yield ctx.df.callActivityAsync("F4", z);
});
```

EXAMPLE: DURABLE FUNCTIONS



TOOLS

VENDOR DEPLOYMENT TOOLS

AWS CLOUDFORMATION

```
        Statement:
          - Action:
              - logs:CreateLogGroup
              - logs:CreateLogStream
              - logs:PutLogEvents
            Effect: Allow
            Resource: '*'
            Sid: AllowLogging
          Version: '2012-10-17'
        PolicyName: LambdaWriteCWLogs
      - PolicyDocument:
          Statement:
          - Action:
              - s3:Get*
            Effect: Allow
            Resource:
              Fn::Sub: arn:aws:s3:::${PhotoRepoS3Bucket}/*
            Sid: ReadFromPhotoRepoS3Bucket
          Version: '2012-10-17'
        PolicyName: ReadFromPhotoRepoS3Bucket
      - PolicyDocument:
          Statement:
          - Action:
              - s3:PutObject
            Effect: Allow
            Resource:
              Fn::Sub: arn:aws:s3:::${PhotoRepoS3Bucket}/*
            Sid: WriteToPhotoRepoS3Bucket
          Version: '2012-10-17'
        PolicyName: WriteToPhotoRepoS3Bucket
      - PolicyDocument:
          Statement:
          - Action:
              - dynamodb:UpdateItem
              - dynamodb:PutItem
            Effect: Allow
            Resource:
              Fn::Sub: arn:aws:dynamodb:${AWS::Region}:${AWS::AccountId}:table/${ImageMetadataDDBTable
            Sid: WriteToImageMetadataDDBTable
          Version: '2012-10-17'
        PolicyName: WriteToImageMetadataDDBTable
      - PolicyDocument:
          Statement:
          - Action:
              - rekognition:DetectLabels
            Effect: Allow
            Resource: '*'
```

AZURE RESOURCE MANAGER

```
    "imagePublisher": "Canonical",
    "imageOffer": "UbuntuServer",
    "nicName": "networkInterface1",
    "vmName": "vm1",
    "virtualNetworkName": "virtualNetwork1",
    "publicIPAddressName": "publicIp1",
    "addressPrefix": "10.0.0.0/16",
    "subnet1Name": "Subnet-1",
    "subnet1Prefix": "10.0.0.0/24",
    "vmStorageAccountContainerName": "vhds",
    "publicIPAddressType": "Dynamic",
    "storageAccountType": "Standard_LRS",
    "vnetID": "[resourceId('Microsoft.Network/virtualNetworks',variables('virtualNetworkName'))]",
    "subnet1Ref": "[concat(variables('vnetID'),'/subnets/',variables('subnet1Name'))]",
    "apiVersion": "2015-06-15"
  },
  "resources": [
    {
      "type": "Microsoft.Storage/storageAccounts",
      "name": "[variables('storageAccountName')]",
      "apiVersion": "2015-06-15",
      "location": "[resourceGroup().location]",
      "properties": {
        "accountType": "[variables('storageAccountType')]"
      }
    },
    {
      "apiVersion": "2015-06-15",
      "type": "Microsoft.Network/publicIPAddresses",
      "name": "[variables('publicIPAddressName')]",
      "location": "[resourceGroup().location]",
      "properties": {
        "publicIPAllocationMethod": "[variables('publicIPAddressType')]",
        "dnsSettings": {
          "domainNameLabel": "[parameters('dnsLabelPrefix')]"
        }
      }
    },
    {
      "apiVersion": "2015-06-15",
      "type": "Microsoft.Network/virtualNetworks",
      "name": "[variables('virtualNetworkName')]",
      "location": "[resourceGroup().location]",
      "properties": {
        "addressSpace": {
          "addressPrefixes": [
            "[variables('addressPrefix')]"
```

GOOGLE CLOUD DEPLOYMENT MANAGER

```
imports:
  - path: path/to/my_vm_template.jinja
    name: my_renamed_template.jinja
  - path: special_vm.py
```

If your template uses other templates as dependencies, import the dependent templates in your configuration as well:

```
imports:
  - path: path/to/my_vm_template.jinja
  - path: special_vm.py
  - path: base_vm.jinja
```

You can also import text files in order to inline the content. For example, if you create a file named resource_type.txt with the following string:

```
compute.v1.instance
```

Import it into your configuration and provide the content inline like so:

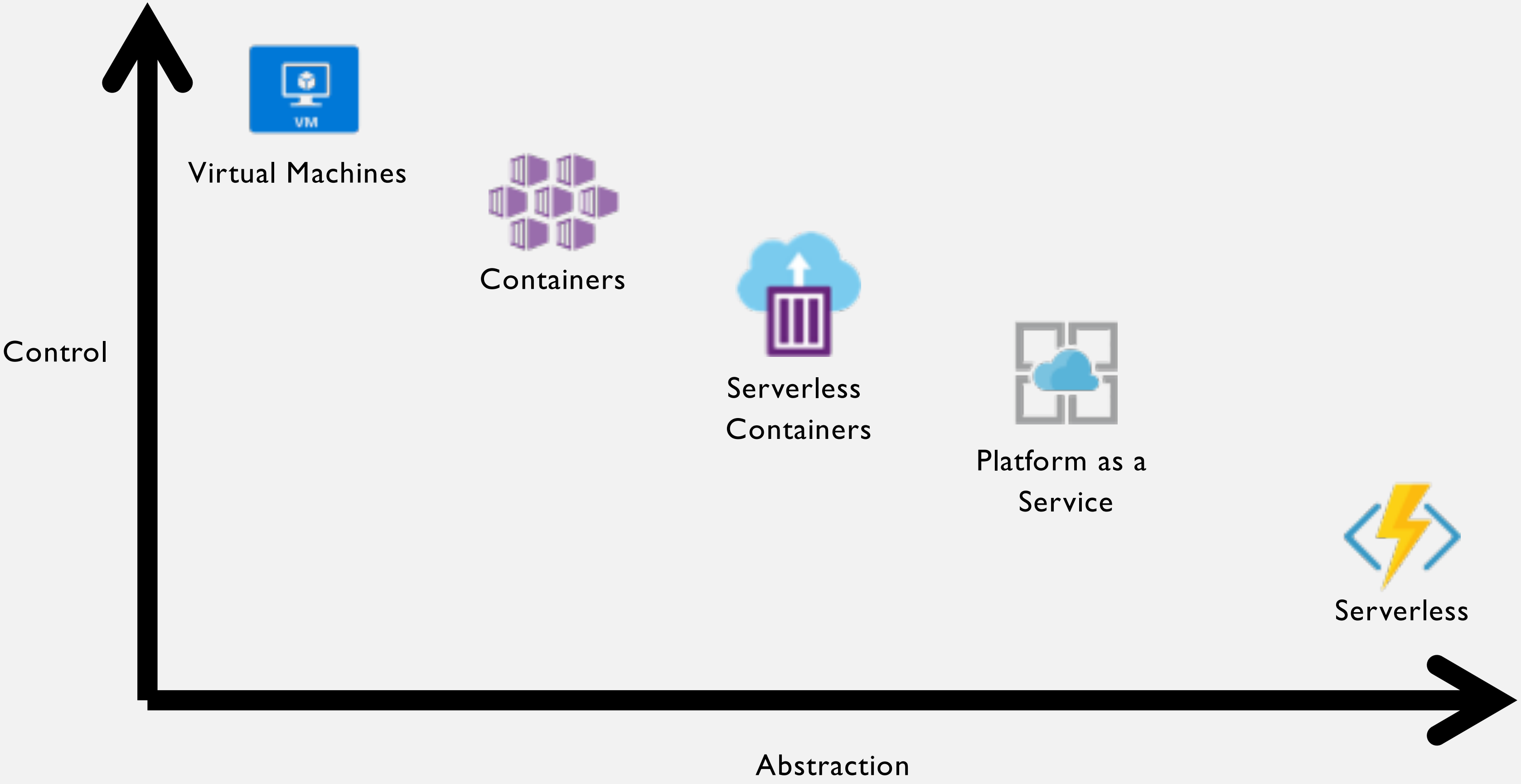
```
imports:
- path: resource_type.txt

resources:
- name: my-vm
  type: {{ imports['resource_type.txt'] }} # Resolves to "compute.v1.instance"
  properties:
    zone: us-central1-a
    machineType: zones/us-central1-a/machineTypes/f1-micro
    disks:
    - deviceName: boot
      type: PERSISTENT
      boot: true
      autoDelete: true
      initializeParams:
        sourceImage: projects/debian-cloud/global/images/family/debian-8
    networkInterfaces:
    - network: global/networks/default
      accessConfigs:
      - name: External NAT
        type: ONE_TO_ONE_NAT
```

TOOLS ALSO PROVIDE ABSTRACTION

- Use Terraform modules
- Use Serverless Framework plugins or components
- Use Pulumi components

- Examples: `github.com/lindydonna/velocity-examples`

The cloud landscape



Virtual Machines

Containers

Serverless Containers

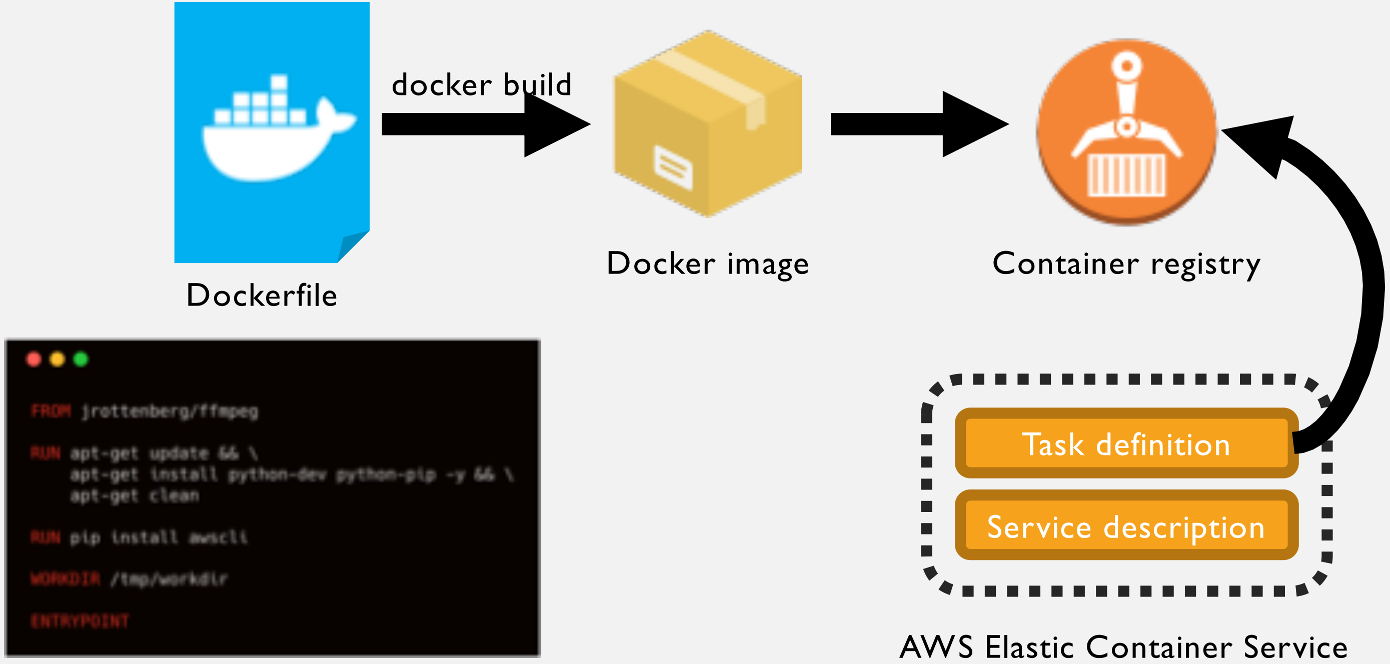
Platform as a Service

Serverless

Control

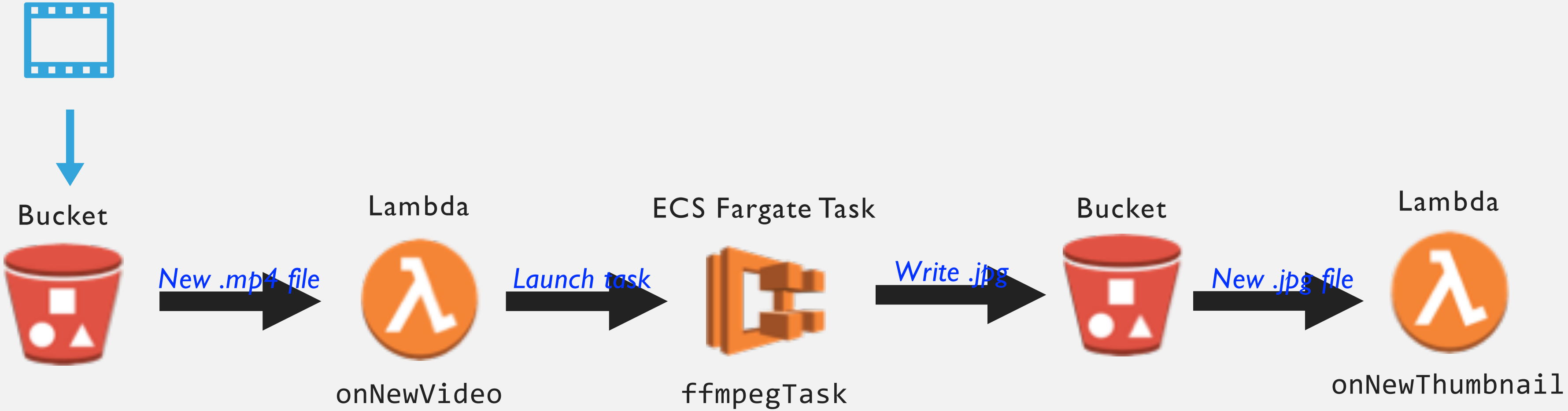
Abstraction

CONTAINERS

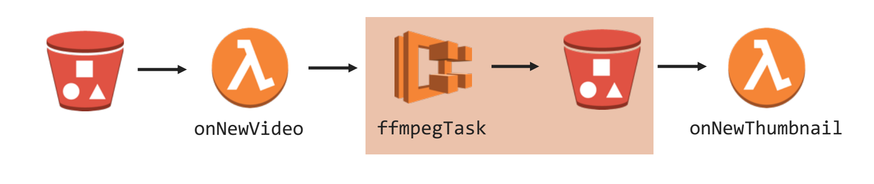


```
FROM jrottenberg/ffmpeg

RUN apt-get update && \
    apt-get install python-dev python-pip -y && \
    apt-get clean

RUN pip install awscli

WORKDIR /tmp/workdir

ENTRYPOINT
```

EXAMPLE: VIDEO THUMBNAILER



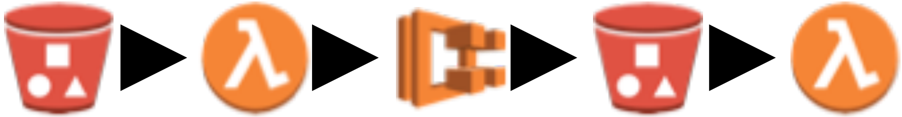
DEFINING THE APP IN PULUMI

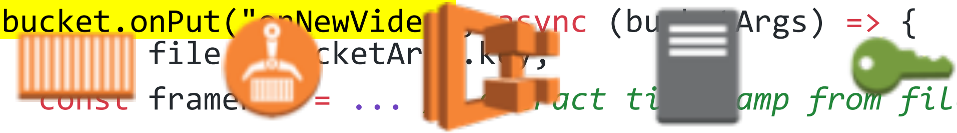


Dockerfile

```
FROM jrottenberg/ffmpeg

RUN apt-get update && \
    apt-get install python-dev python-pip -y && \
    apt-get clean

RUN pip install awscli

WORKDIR /tmp/workdir

ENTRYPOINT \
  aws s3 cp s3://${S3_BUCKET}/${INPUT_VIDEO} ./${INPUT_VIDEO} && \
  ffmpeg -i ./${INPUT_VIDEO} -ss ${TIME_OFFSET} -vframes 1 -f image2 -an -y ${OUTPUT_FILE} && \
  aws s3 cp ./${OUTPUT_FILE} s3://${S3_BUCKET}/${OUTPUT_FILE}
```

```
let bucket = new cloud.Bucket("bucket");

let ffmpegTask = new cloud.Task("ffmpegTask", {
  build: "./docker-folder",
  memoryReservation: 512,
});

bucket.onPut("onNewVide  async (bu   Args) => {
  file   cketAr  .k y,
  const frame  = ...   act ti   amp from filename

  await ffmpegTask.run({
    environment: {
      "S3_BUCKET": bucket.id.get(),
      "INPUT_VIDEO": file,
      "TIME_OFFSET": framePos,
      "OUTPUT_FILE": file + '.jpg',
    },
  });
}, { keySuffix: ".mp4" });

bucket.onPut("onNewThumbnail", async (bucketArgs) => {
  console.log(`*** New thumbnail: file ${bucketArgs.key}.`);
}, { keySuffix: ".jpg" });
```





Container image

ECR repository

ECS task

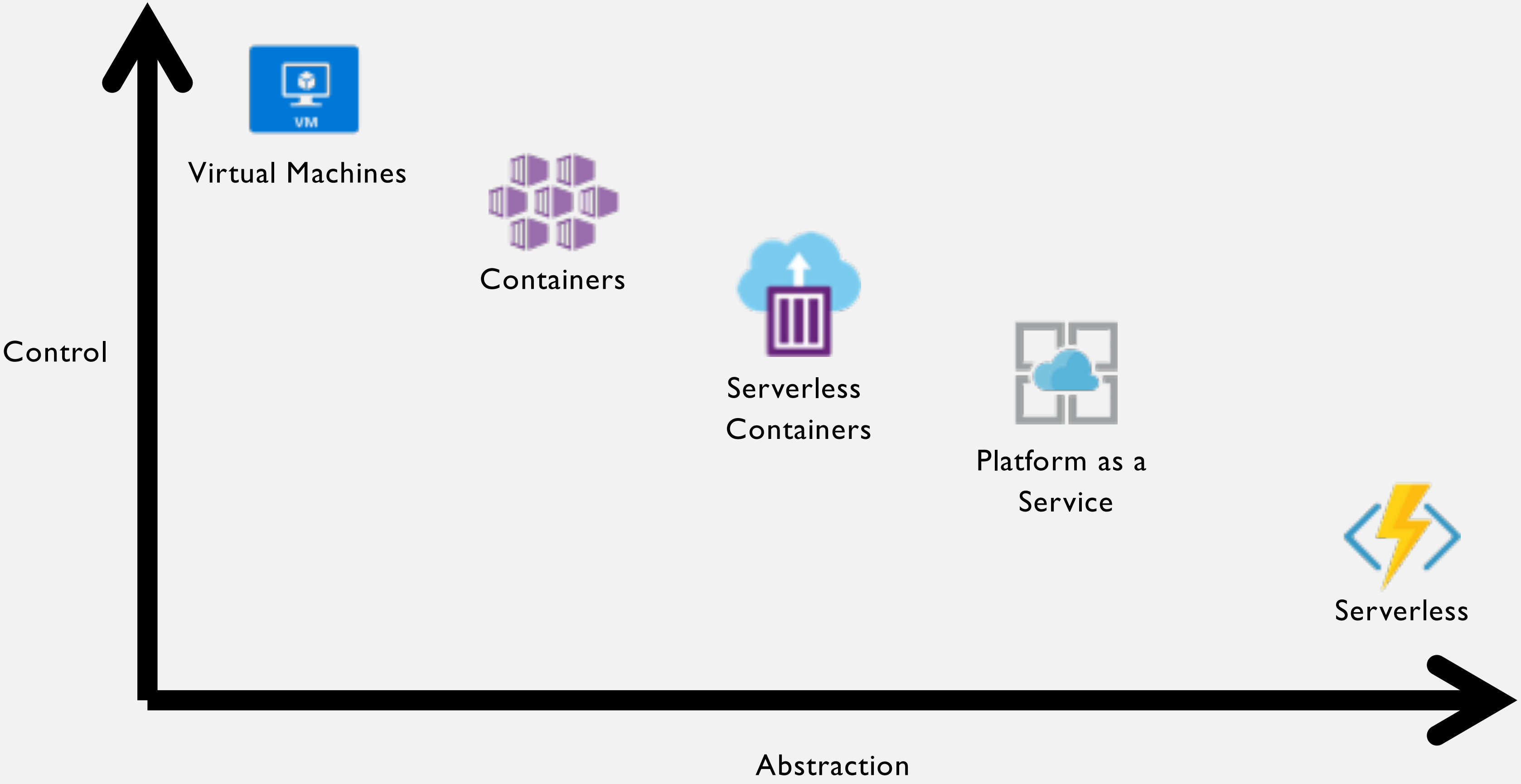
ECS cluster

IAM roles

EXAMPLE: PROVISION QUEUES

```
function createQueue(name, deadLetter) {
  return new aws.sqs.Queue(`${common.prefix}-${name}`, { ... });
}

exports.certIssuer = {
  request:    createQueue("c-i-req”, true),
  response:   createQueue("c-i-res", true),
  prepare:    createQueue("c-i-prep", true),
  initOrg:    createQueue("c-i-init-org", true),
  initOrgRes: createQueue("c-i-init-org-res", true),
  confirmTxs: createQueue("confirm-tx"),
};
```

The cloud landscape



Virtual Machines

Containers

Serverless Containers

Platform as a Service

Serverless

Control

Abstraction

CONTAINERS WITH PULUMI

- How often should I *update* my Dockerfile dependencies?
- ~~How do I *build* my container images~~?
- ~~How do I get my containers in *production*~~?
- How *many* servers do I need?
- How can I *scale* my app?

@lindydonna

SUMMARY

- Serverless and containers each have their place
- Use serverless for event-based code that needs to scale on demand
- Use containers for durable workloads, or to customize environment
- Define abstractions using infrastructure-as-code tooling

Learn more at **`pulumi.io`**

`github.com/pulumi`

@PulumiCorp

@lindydonna