SERVERLESS + CONTAINERS = MODERN CLOUD APPLICATIONS

Donna Malayeri

Product Manager, Pulumi

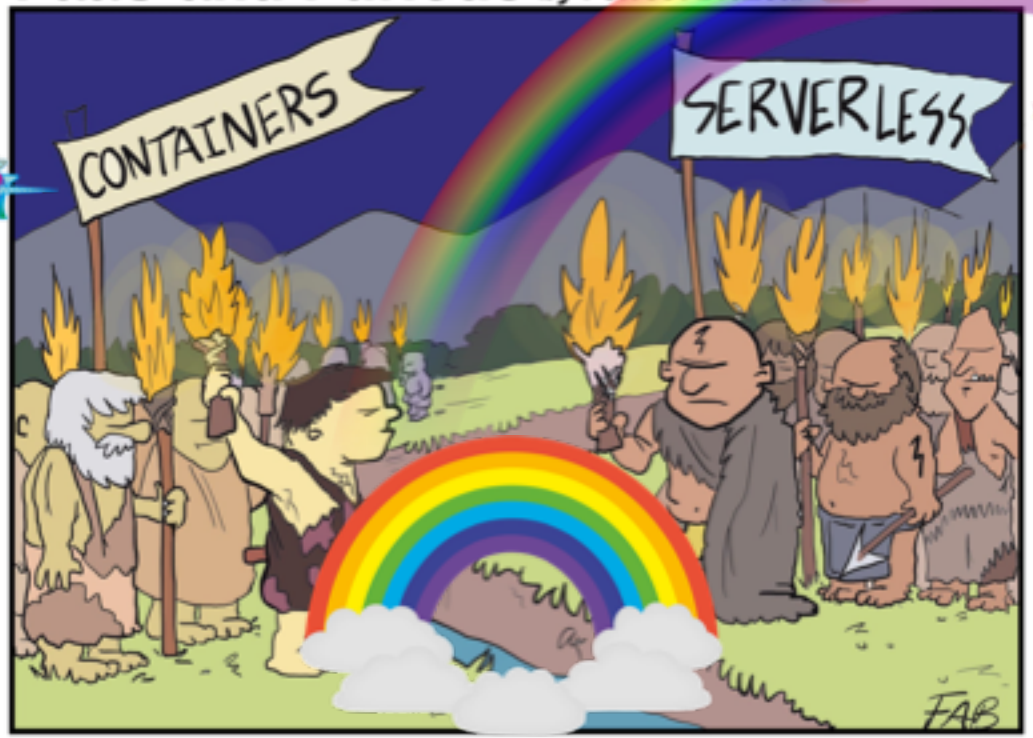
@PulumiCorp

@lindydonna

pulumi

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

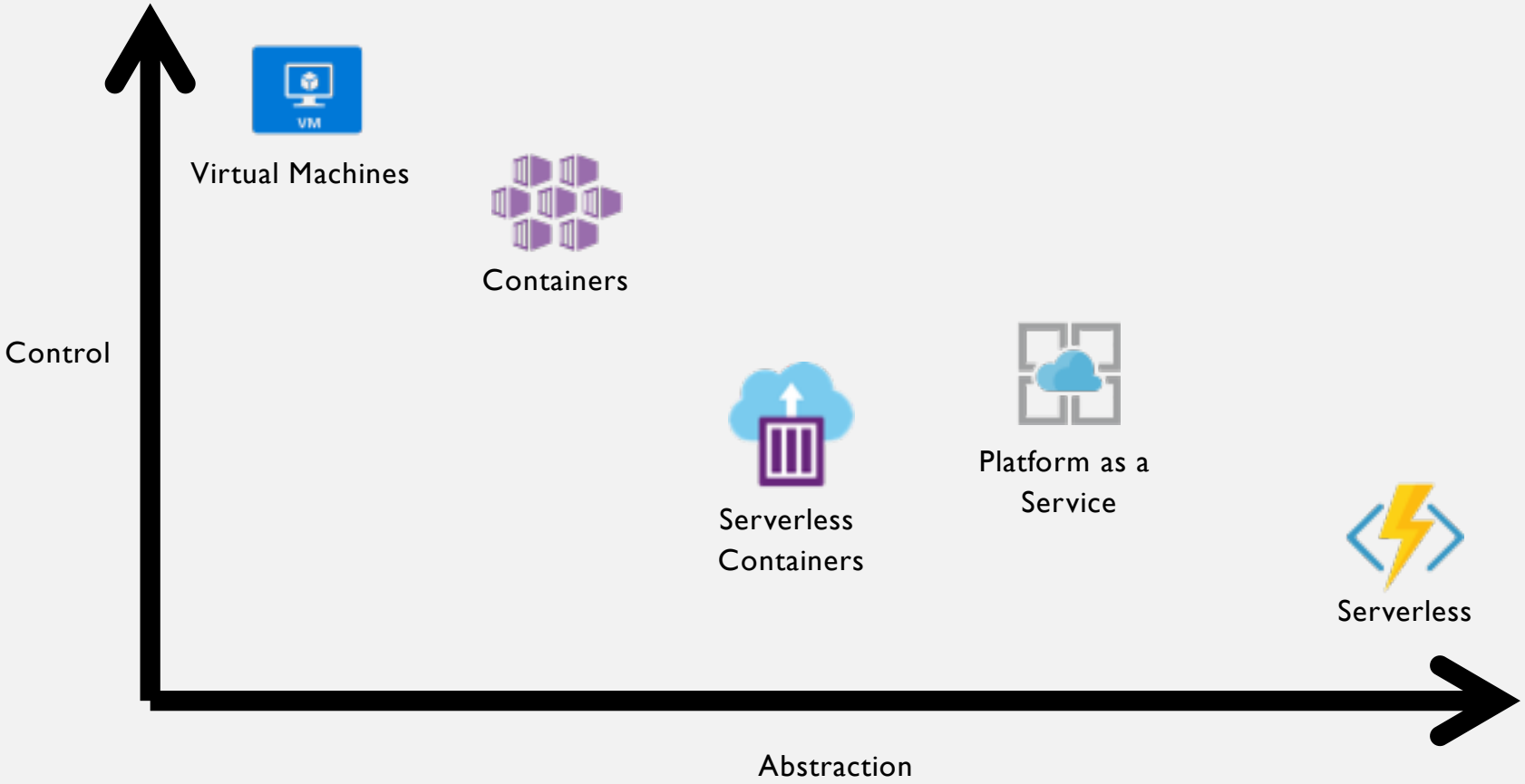
PROGRAMMING IS ABOUT ABSTRACTION

- 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

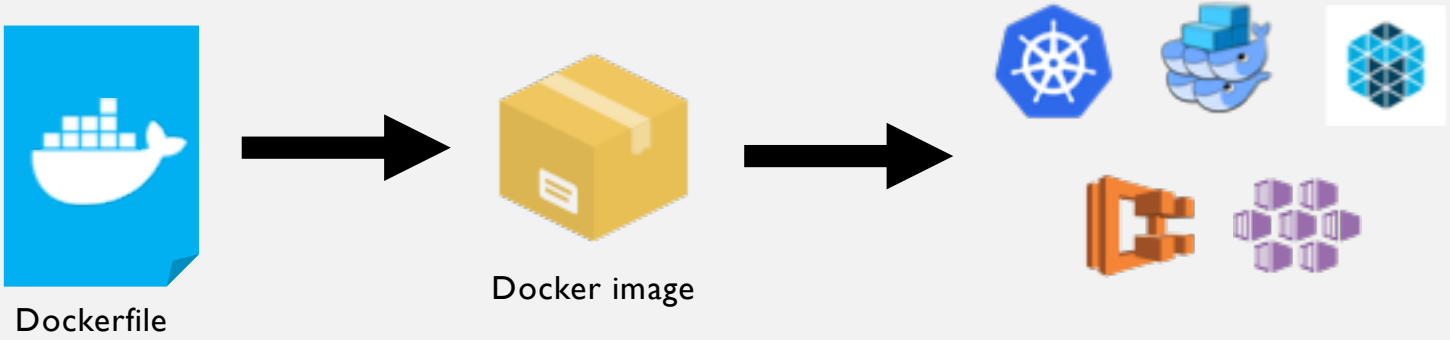
The cloud landscape



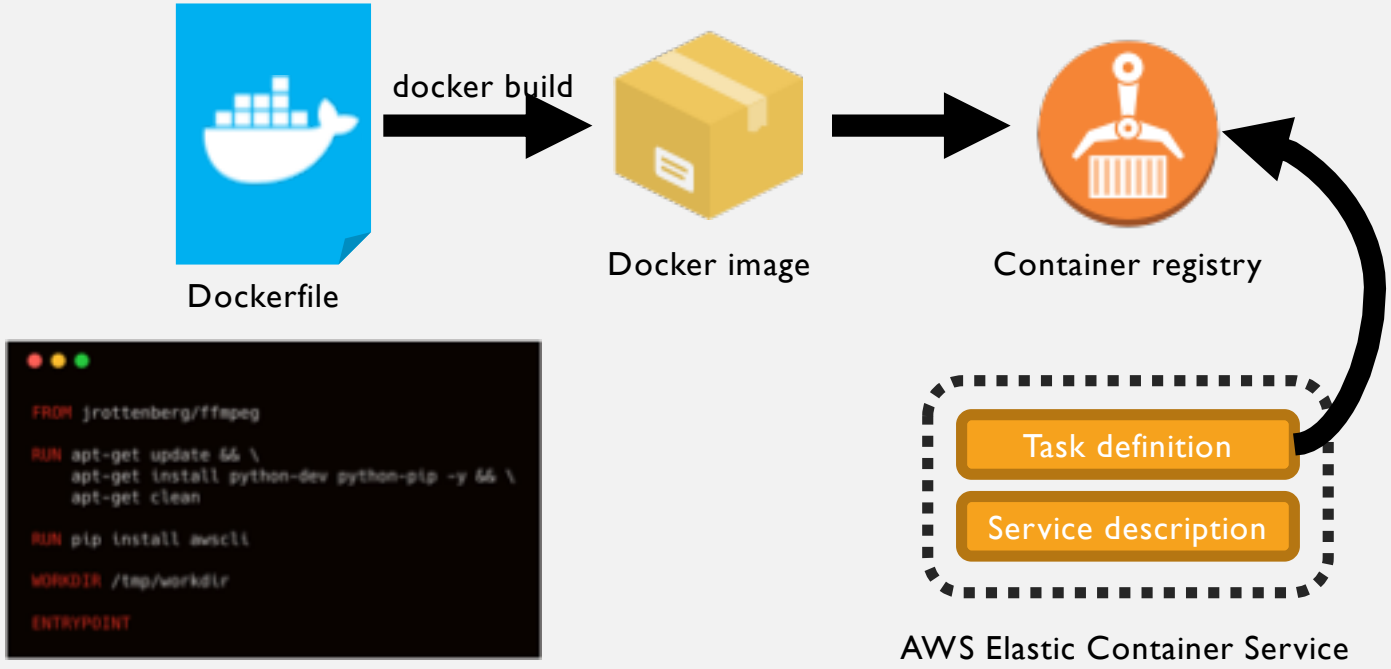
IN THE EARLY DAYS OF CLOUD, THERE WERE ONLY VIRTUAL MACHINES

- 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?

CONTAINERS REDUCE COMPLEXITY



CONTAINERS

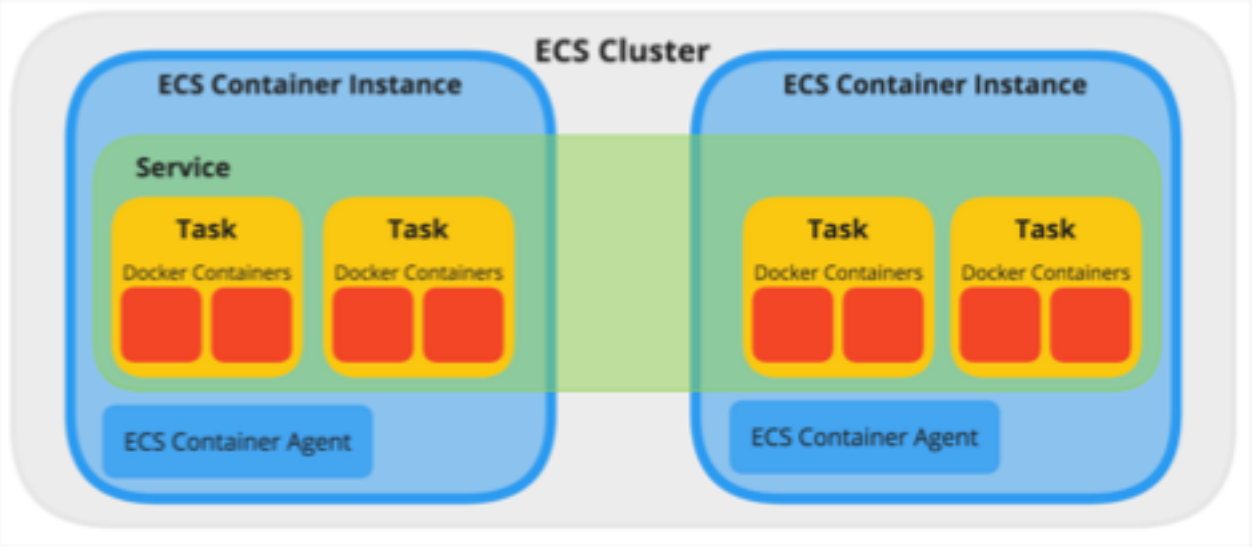

docker build
Dockerfile
Docker image
Container registry
FROM jrottenberg/ffmpeg
RUN apt-get update && \
apt-get install python-dev python-pip -y && \
apt-get clean
RUN pip install awscli
WORKDIR /tmp/workdir
ENTRYPOINT
Task definition
Service description
AWS Elastic Container Service

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

ECS Service Description

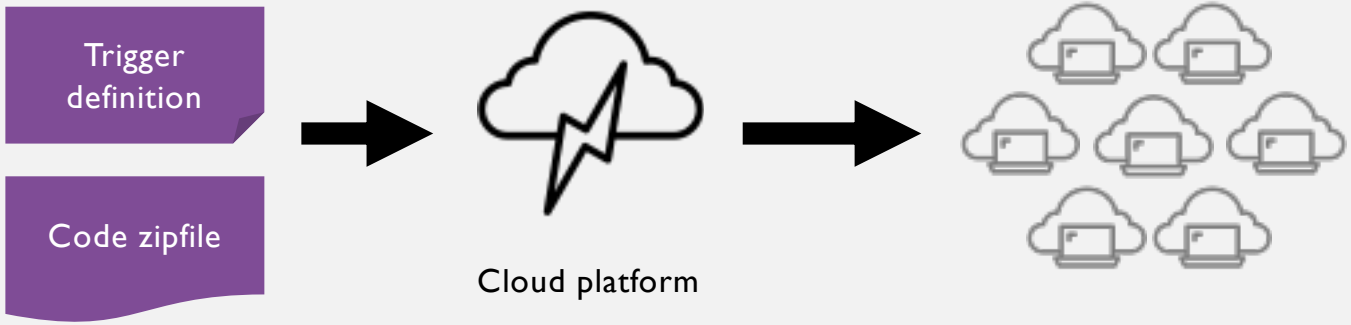


Graphic: https://medium.freecodecamp.org/amazon-ecs-terms-and-architecture-807d8c4960fd

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



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?

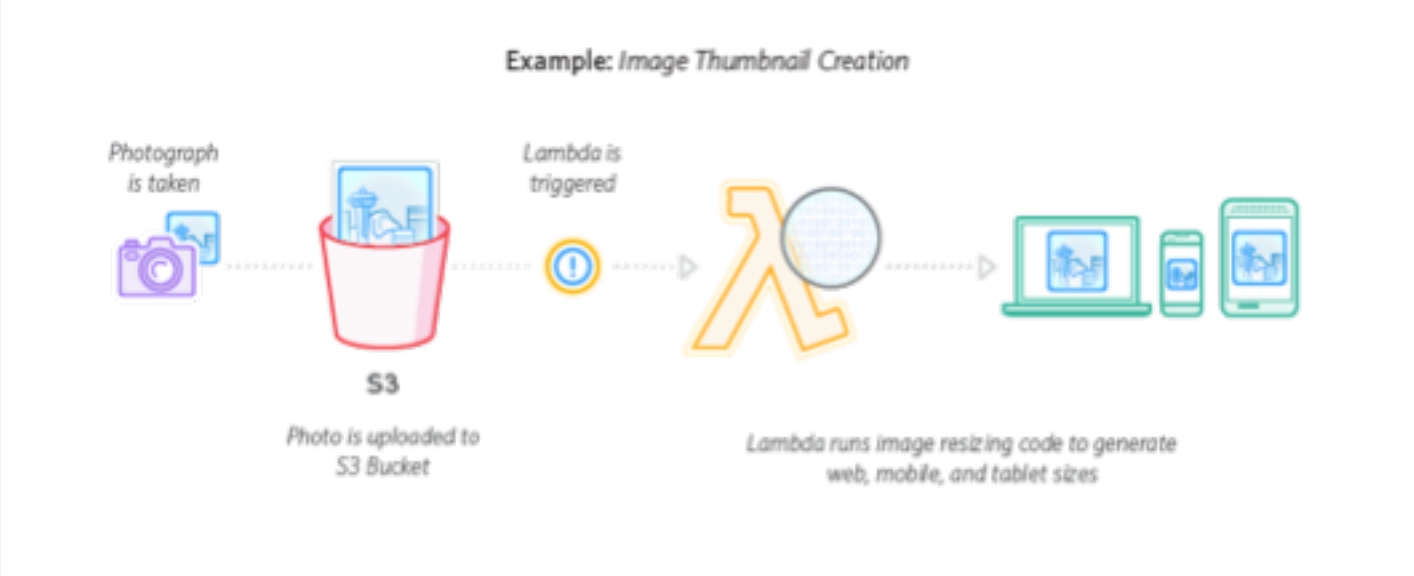
- Reduce operational overhead
- Faster time to market
- Focus on business value

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

SCHEDULED TASKS

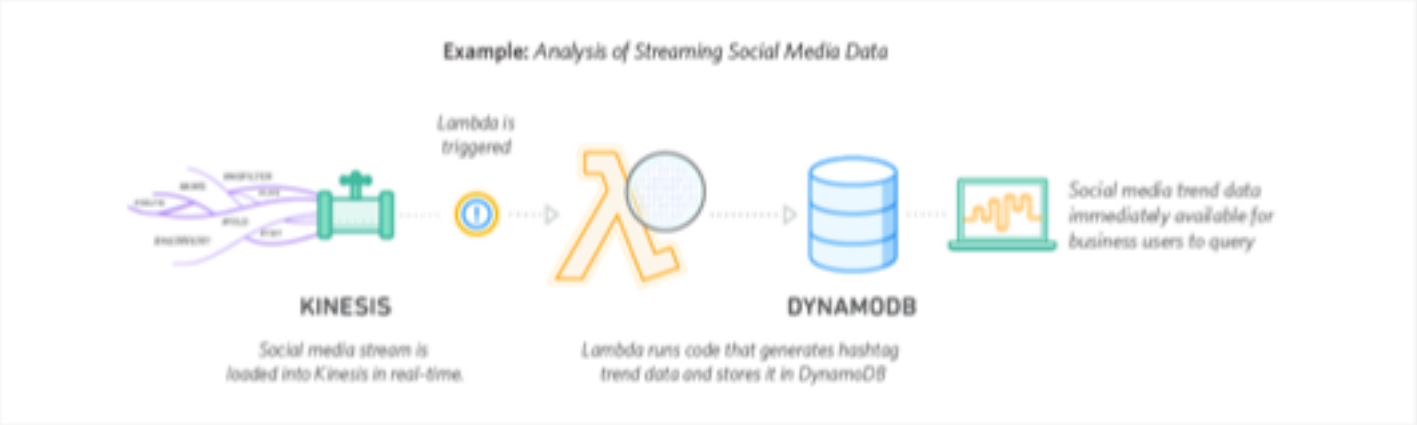
https://functions.azure.com

CREATE IMAGE THUMBNAIL



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

ANALYZE SOCIAL MEDIA STREAM



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

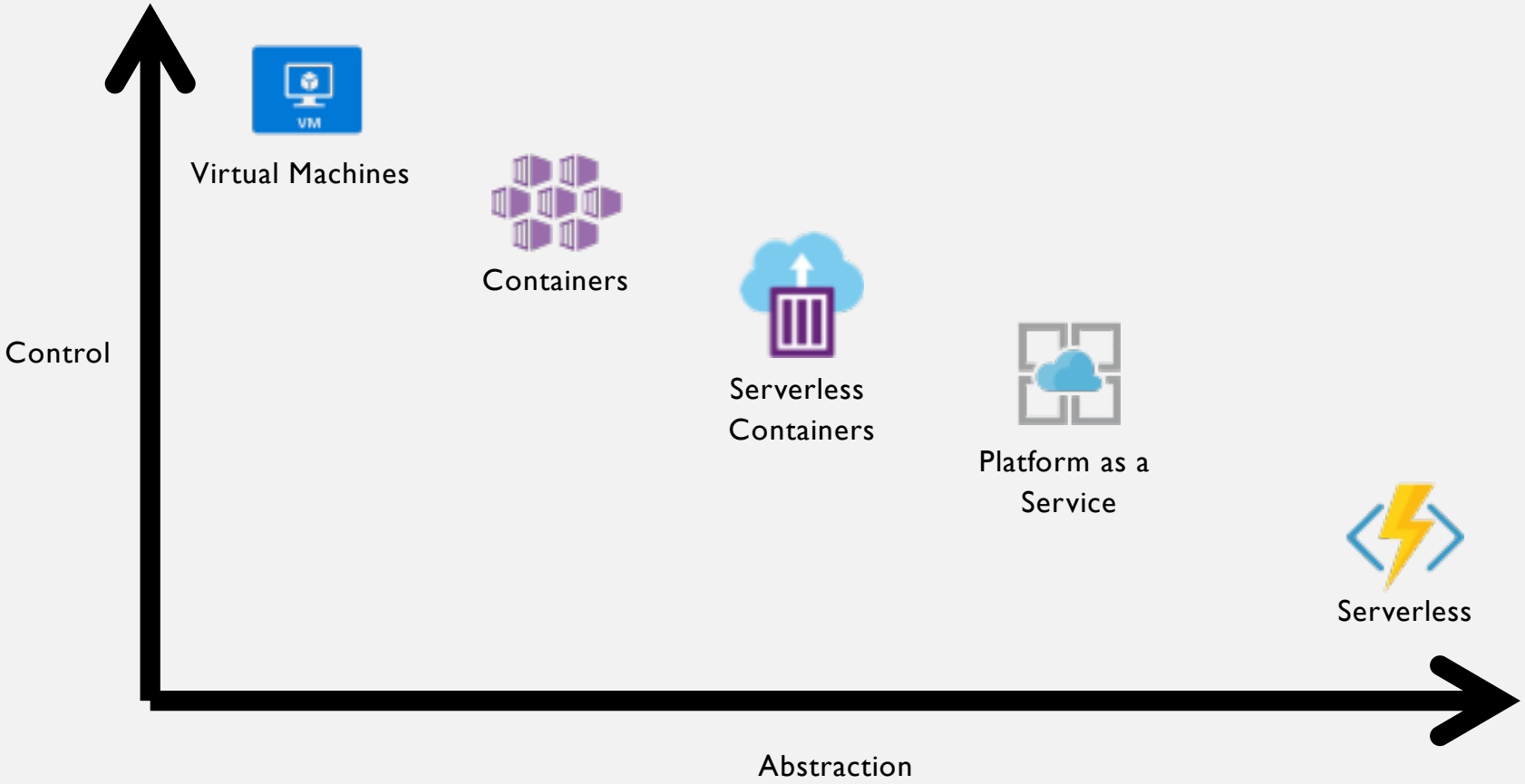
- Azure Container Instances
- AWS Fargate

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

CONTAINERS AND SERVERLESS

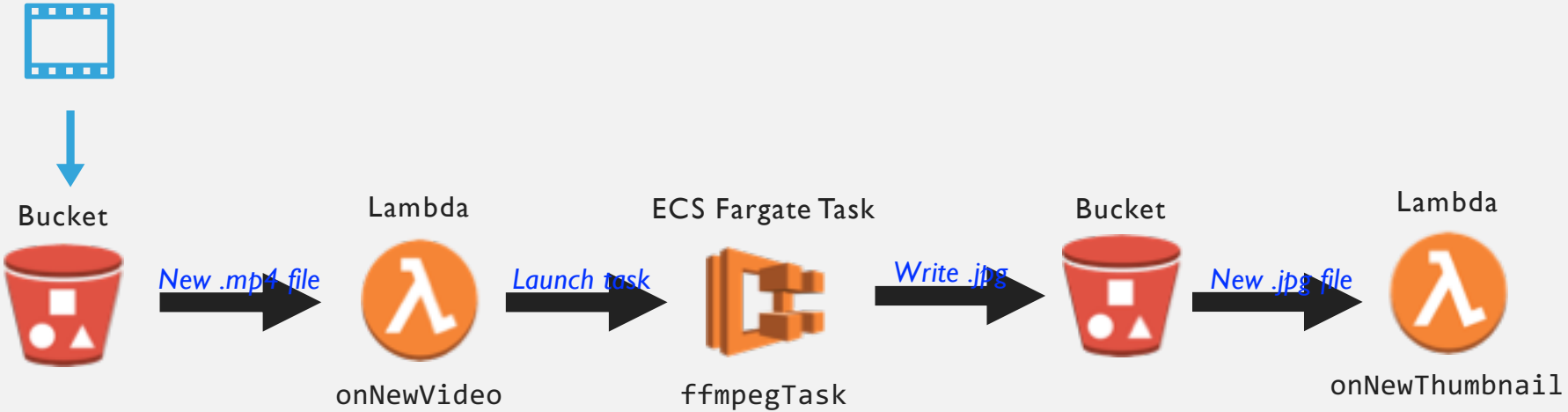
- Use containers for control over the execution environment
  - Customize software and physical servers
  - Great for long-running compute
- Use serverless for event-based compute that scales on demand
  - Less to manage
  - Less to configure

The cloud landscape

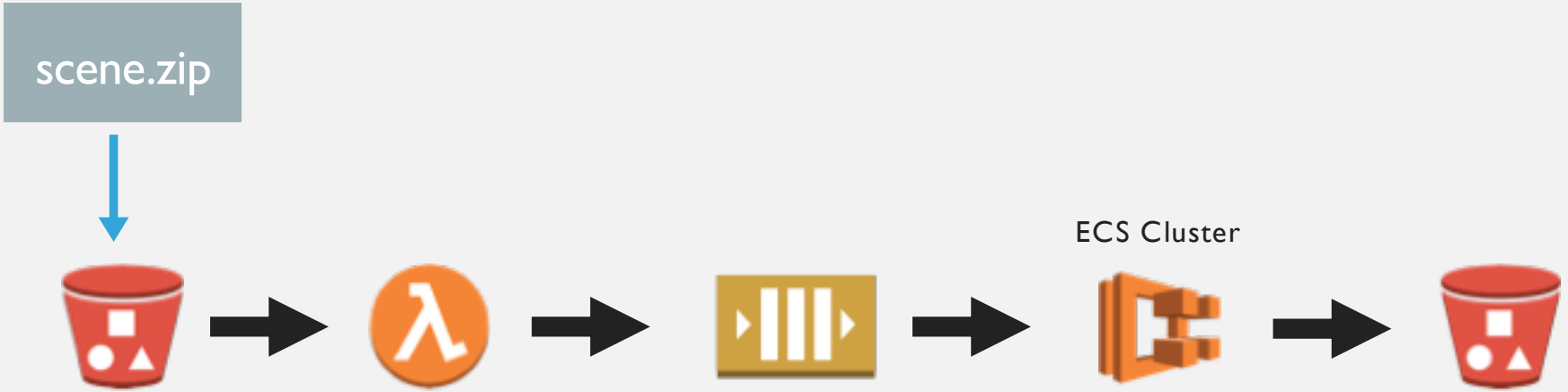


COMBINING THE TWO

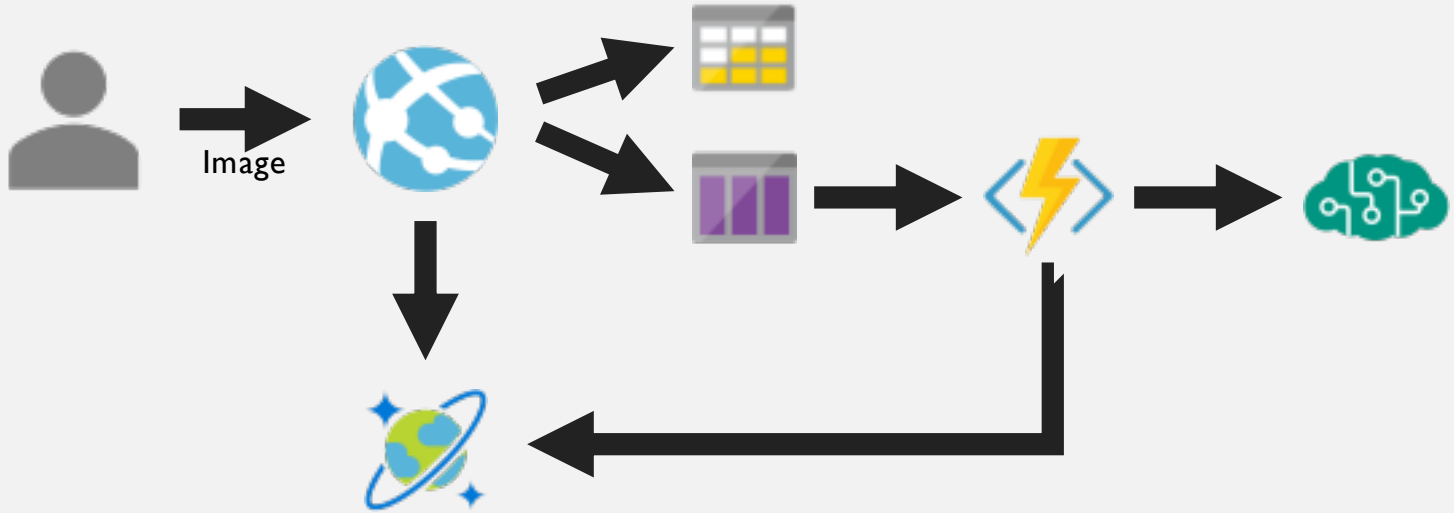
EXAMPLE: VIDEO THUMBNAILER



EXAMPLE: RAY TRACING



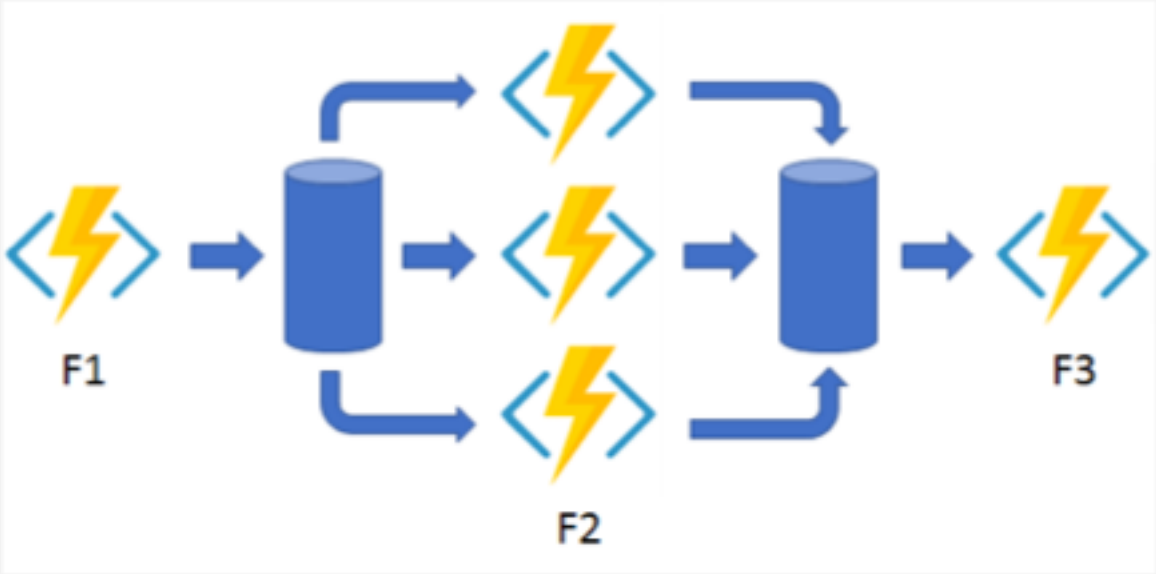
EXAMPLE: CONTENT MODERATION



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

AZURE RESOURCE MANAGER

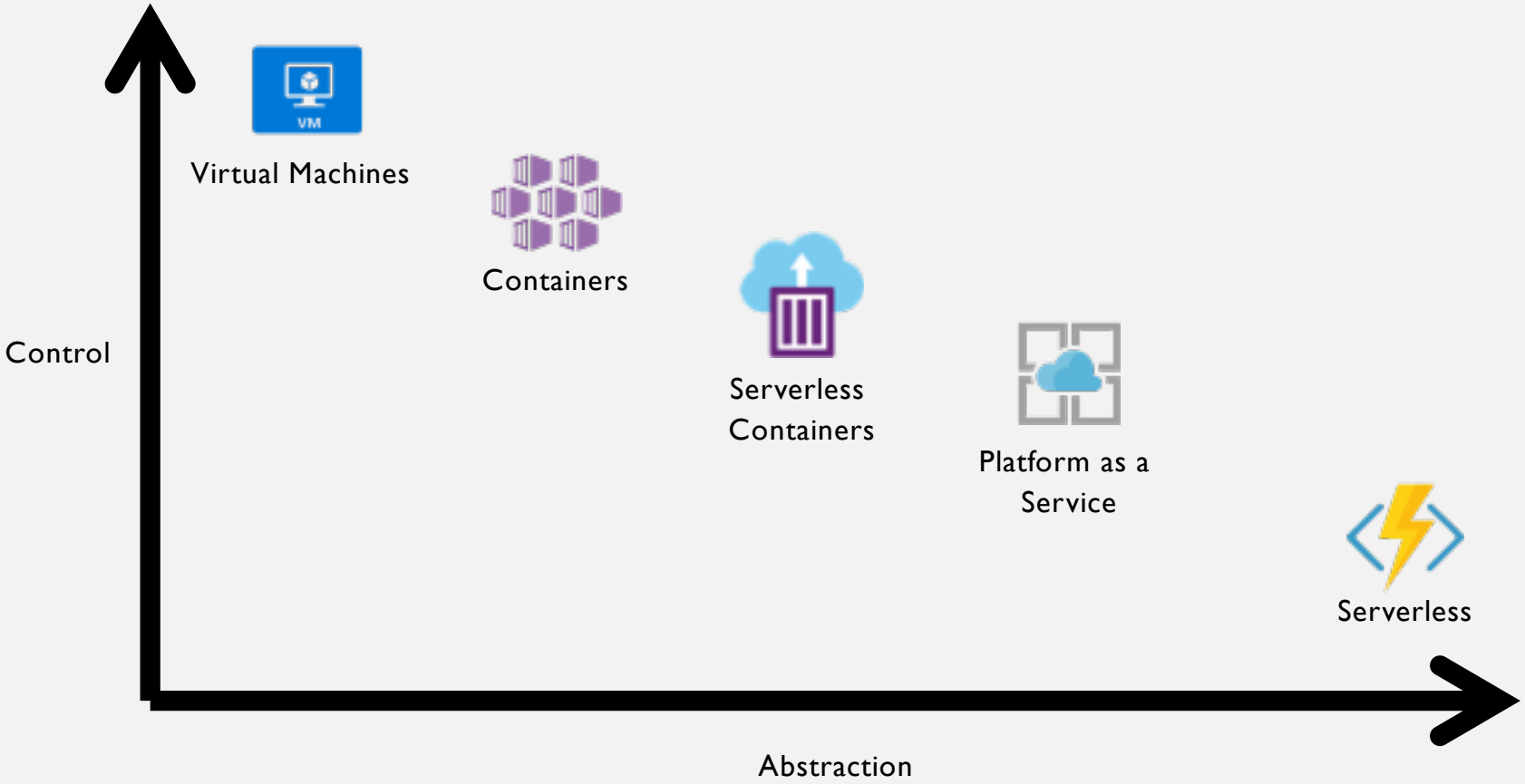
GOOGLE CLOUD DEPLOYMENT MANAGER

TOOLS ALSO PROVIDE ABSTRACTION

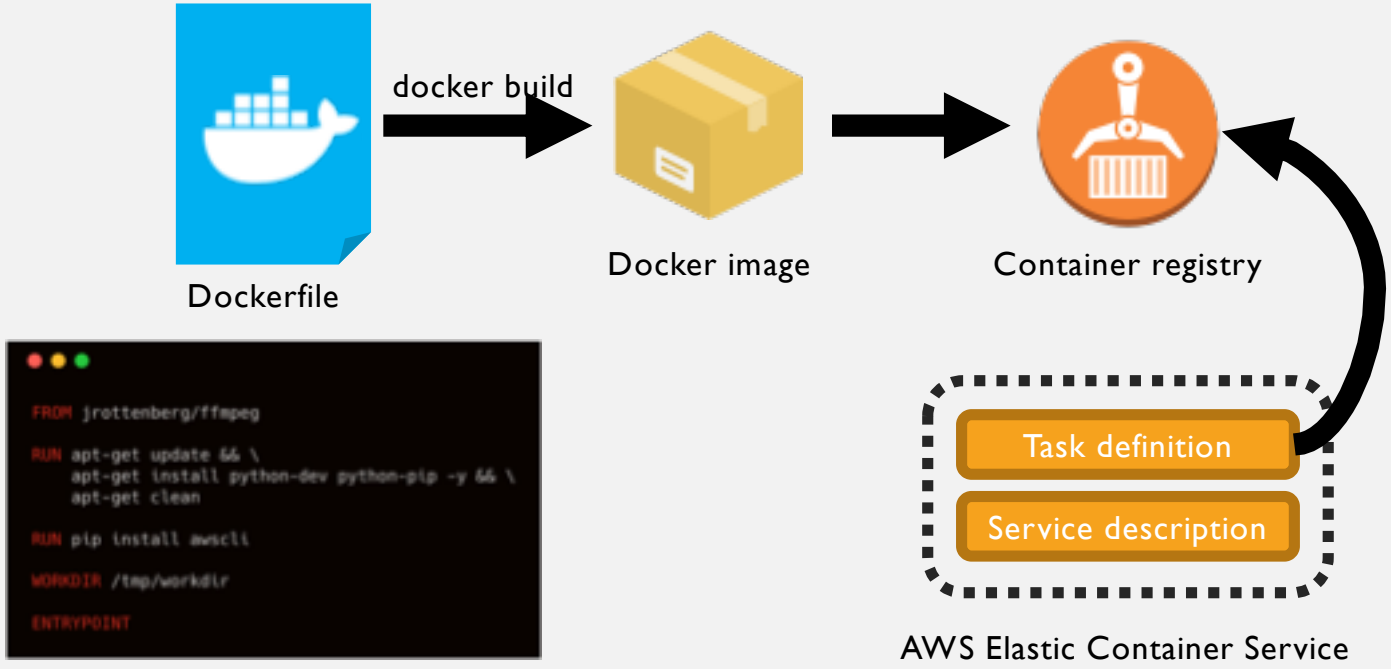
- Use Terraform modules
- Use Serverless Framework plugins or components
- Use Pulumi components

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

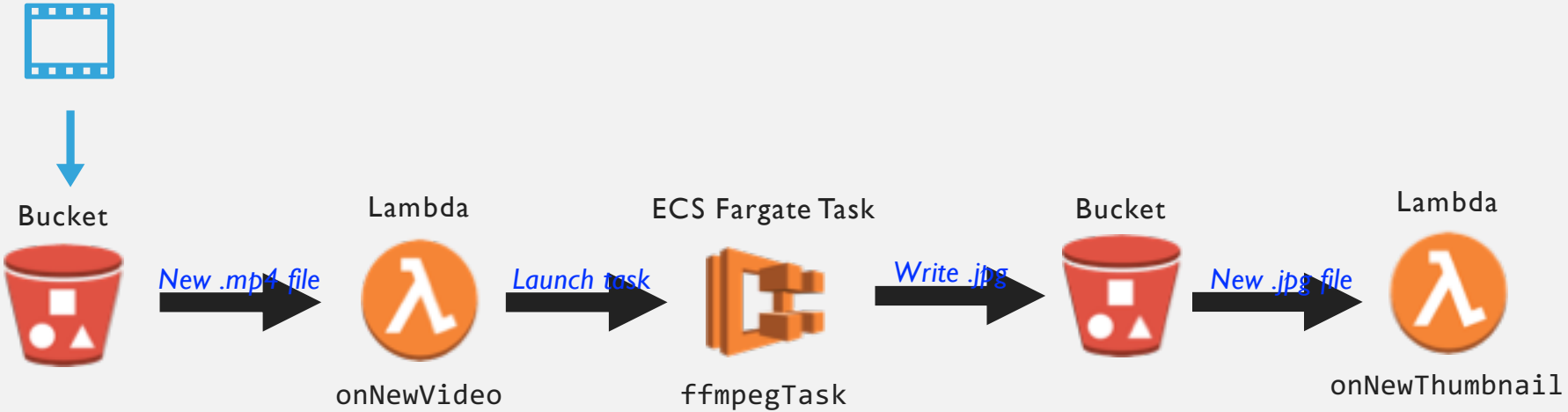
The cloud landscape



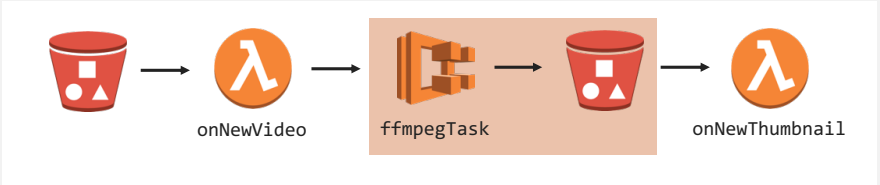
CONTAINERS


docker build
Dockerfile
Docker image
Container registry
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
Task definition
Service description
AWS Elastic Container Service

EXAMPLE: VIDEO THUMBNAILER



DEFINING THE APP IN PULUMI



Dockerfile

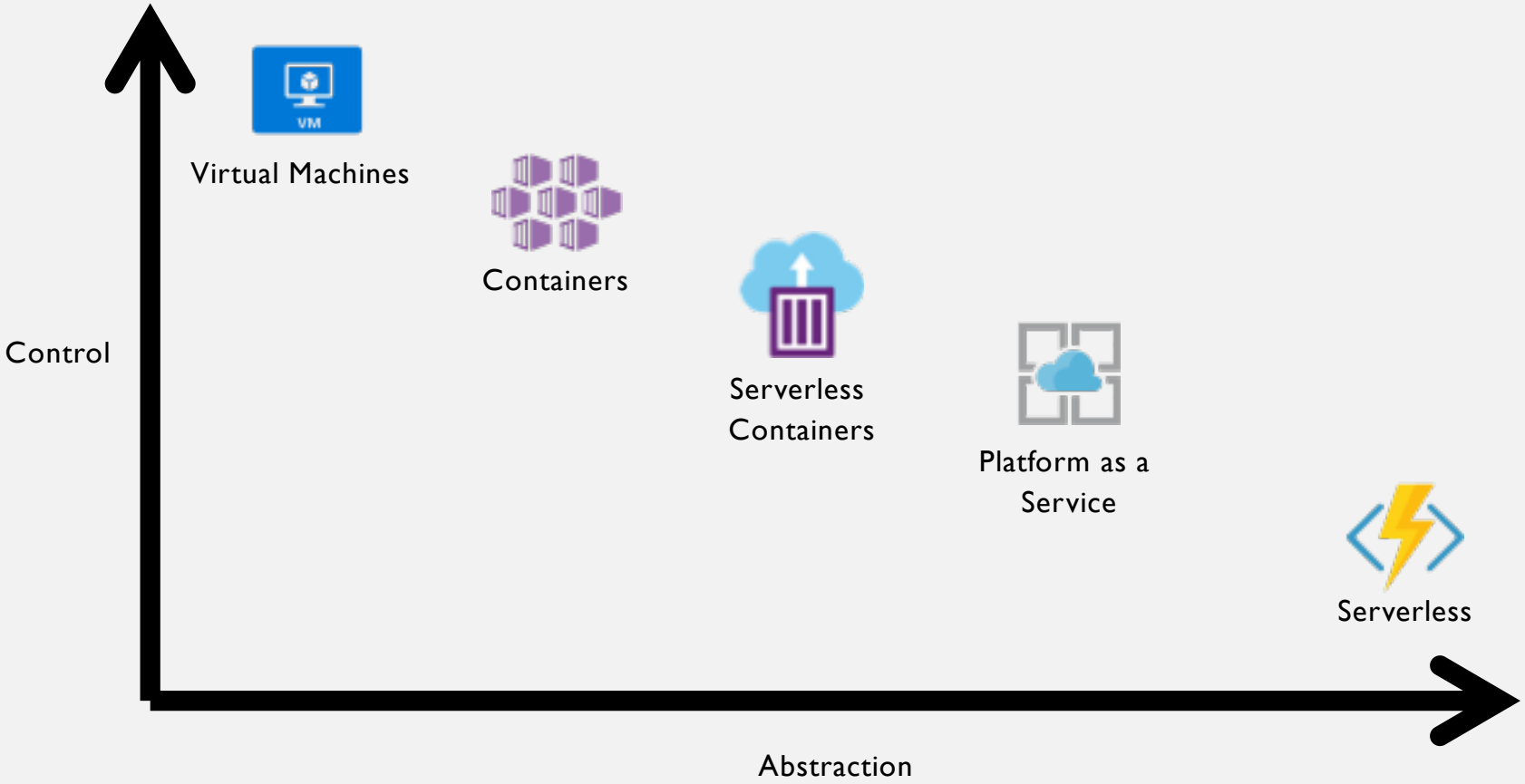
```
FROM jrottenberg/ffmpeg

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("onNewVideo", async (bucketArgs) => {
  const file = bucketArgs.key;
  const framePos = ... // extract timestamp 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" });
```

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



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?

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