

INFRASTRUCTURE as CODE



**Why infrastructure-as-code
matters: a short story.**

You are starting a
new project



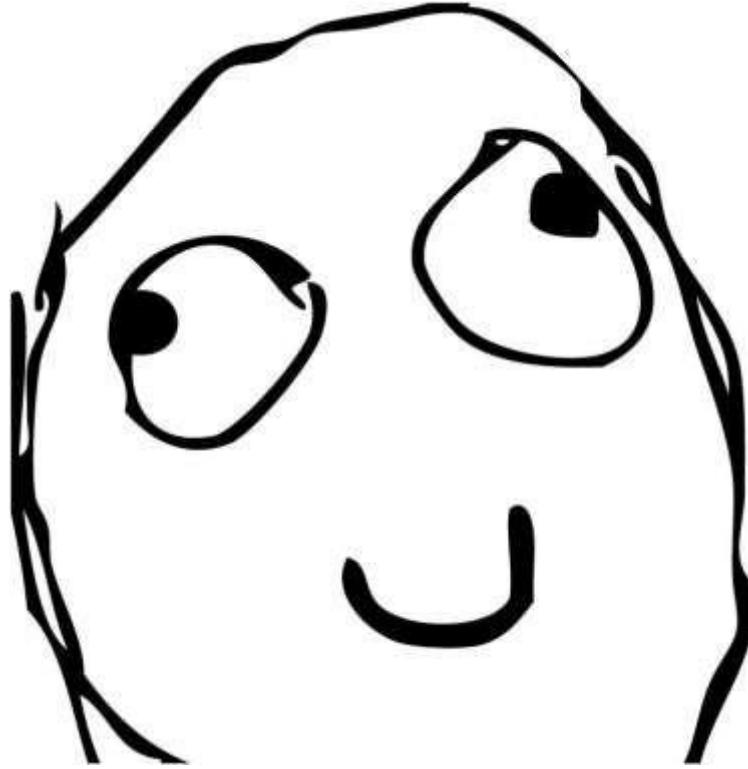
I know, I'll use Ruby on Rails!

```
> gem install rails
```

```
> gem install rails  
Fetching: i18n-0.7.0.gem (100%)  
Fetching: json-1.8.3.gem (100%)  
Building native extensions. This could take a while...  
ERROR: Error installing rails:  
ERROR: Failed to build gem native extension.
```

```
    /usr/bin/ruby1.9.1 extconf.rb  
creating Makefile
```

```
make  
sh: 1: make: not found
```



Ah, I just need to install make

```
> sudo apt-get install make
```

```
...  
Success!
```

```
> gem install rails
```

```
> gem install rails
Fetching: nokogiri-1.6.7.2.gem (100%)
Building native extensions. This could take a while...
ERROR: Error installing rails:
ERROR: Failed to build gem native extension.
```

```
/usr/bin/ruby1.9.1extconf.rb
checking if the C compiler accepts... yes
Building nokogiri using packaged libraries.
Using mini_portile version2.0.0.rc2
checking for gzopen() in -lz... no
zlib is missing; necessary for building libxml2
*** extconf.rb failed ***
```



Hmm. Time to visit StackOverflow.

```
> sudo apt-get install zlib1g-dev
```

```
...  
Success!
```

```
> gem install rails
```

```
> gem install rails
```

```
Building native extensions. This could take a while...
```

```
ERROR: Error installing rails:
```

```
ERROR: Failed to build gem native extension.
```

```
    /usr/bin/ruby1.9.1extconf.rb
```

```
checking if the C compiler accepts... yes
```

```
Building nokogiri using packaged libraries.
```

```
Using mini_portile version2.0.0.rc2
```

```
checking for gzopen() in -lz... yes
```

```
checking for iconv... yes
```

```
Extracting libxml2-2.9.2.tar.gz into tmp/x86_64-pc-linux-gnu/ports/libxml2/2.9.2... OK
```

```
*** extconf.rb failed ***
```



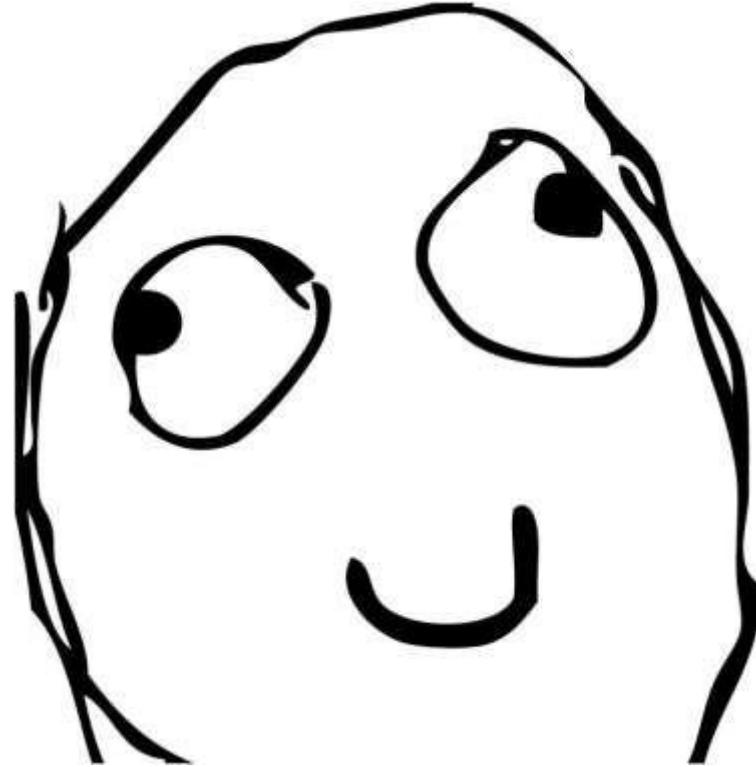
y u never install correctly?

**(Spend 2 hours trying random
StackOverflow suggestions)**

```
> gem install rails
```

```
> gem install rails
```

```
...  
Success!
```



Finally!

```
> rails new my-project  
> cd my-project  
> rails start
```

```
> rails new my-project  
> cd my-project  
> rails start
```

```
/source/my-project/bin/spring:11:in `<top (required)>':  
undefined method `path_separator' for Gem::Module  
(NoMethodError)  
from bin/rails:3:in `load'  
from bin/rails:3:in `<main>'
```



FFFFFFF
FFFFFFF
FFFFFFF
FFFFF
FFFFF
FFFFF
UUUU
UUUU
UUUU-

Eventually, you get it working

Now you have to deploy your
Rails app in **production**

AWS Services Edit

Amazon Web Services

Compute:

- EC2 Virtual Servers in the Cloud
- EC2 Container Service Run and Manage Docker Containers
- Elastic Beanstalk Run and Manage Web Apps
- Lambda Run Code in Response to Events

Storage & Content Delivery

- S3 Scalable Storage in the Cloud
- CloudFront Global Content Delivery Network
- Elastic File System PREVIEW Fully Managed File System for EC2
- Glacier Archive Storage in the Cloud
- Import/Export Snowball Large-Scale Data Transport
- Storage Gateway Hybrid Storage Integration

Database

- RDS Managed Relational Database Service
- DynamoDB Managed NoSQL Database
- ElastiCache

Developer Tools

- CodeCommit Store Code in Private Git Repositories
- CodeDeploy Automate Code Deployments
- CodePipeline Release Software using Continuous Delivery

Management Tools

- CloudWatch Monitor Resources and Applications
- CloudFormation Create and Manage Resources with Templates
- CloudTrail Track User Activity and API Usage
- Config Track Resource Inventory and Changes
- OpsWorks Automate Operations with Chef
- Service Catalog Create and Use Standardized Products
- Trusted Advisor Optimize Performance and Security

Security & Identity

- Identity & Access Management Manage User Access and Encryption Keys
- Directory Service Host and Manage Active Directory
- Inspector PREVIEW

Internet of Things

- AWS IoT Connect Devices to the Cloud

Mobile Services

- Mobile Hub PREVIEW Build, Test, and Monitor Mobile Apps
- Cognito User Identity and App Data Synchronization
- Device Farm Test Android, FireOS, and iOS Apps on Real Devices in the Cloud
- Mobile Analytics Collect, View and Export App Analytics
- SNS Push Notification Service

Application Services

- API Gateway Build, Deploy and Manage APIs
- AppStream Low Latency Application Streaming
- CloudSearch Managed Search Service
- Elastic Transcoder Easy-to-Use Scalable Media Transcoding
- SES Email Sending and Receiving Service
- SQS Message Queue Service
- SWF

Resource Groups

A resource group is a collection of resources or more tags. Create a group for each production environment in your account.

Create a Group Tag Editor

Additional Resources

Getting Started

Read our documentation or view our tutorials about AWS.

AWS Console Mobile App

View your resources on the go with our app, available from Amazon Appstore, Google Play, and iTunes.

AWS Marketplace

Find and buy software, launch with 1-Click.

AWS re:Invent Announcements

Explore the next generation of AWS cloud services and what's new.

Service Health

You use the AWS Console to deploy an EC2 instance

```
> ssh ec2-user@ec2-12-34-56-78.compute-1.amazonaws.com
```



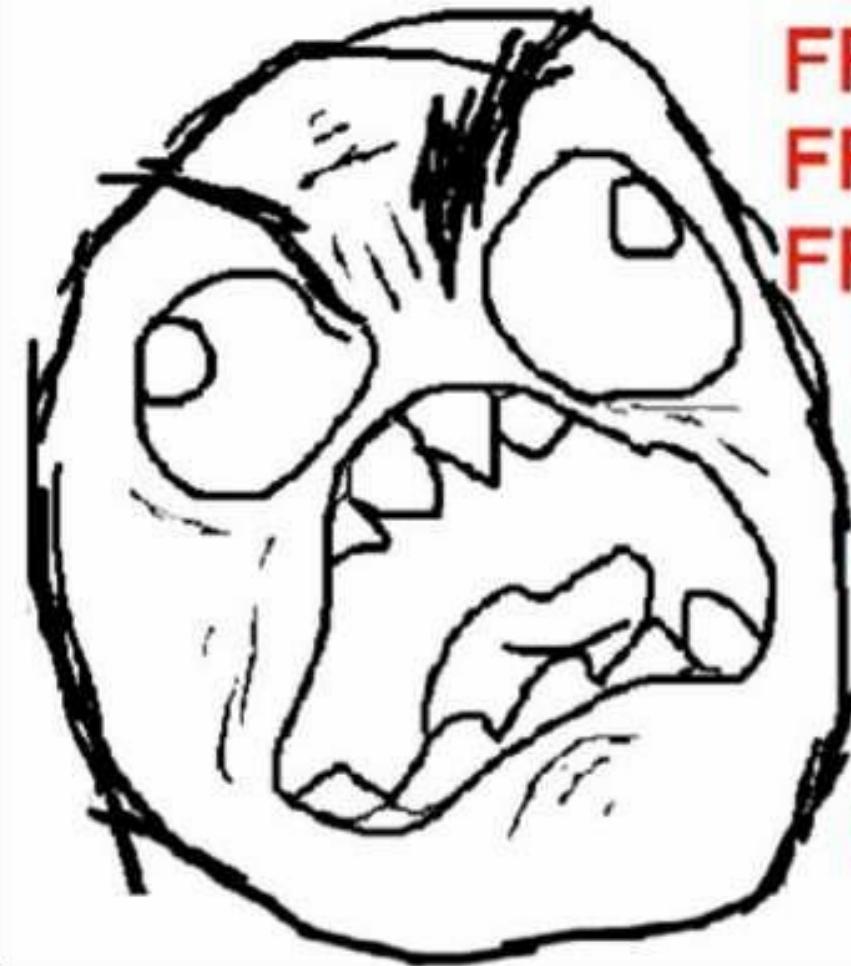
```
[ec2-user@i p- 172- 31- 61- 204 ~]$ gem install rails
```

```
> ssh ec2-user@ec2-12-34-56-78.compute-1.amazonaws.com
```



```
[ec2-user@i p- 172- 31- 61- 204 ~]$ gem install rails  
ERROR: Error installing rails:  
ERROR: Failed to build gem native extension.
```

```
/usr/bin/ruby1.9.1 extconf.rb
```



FFFFFFF
FFFFFFF
FFFFFFF
FFFFF
FFFFF
FFFFF
UUUU
UUUU
UUUU-

Eventually you get it working

Critical Ruby On Rails Issue Threatens 240,000 Websites

Bug allows attackers to execute arbitrary code on any version of Ruby published in the last six years.

All versions of the open source Ruby on Rails Web application framework released in the past six years have a critical vulnerability that an attacker could exploit to execute arbitrary code, steal information from databases and crash servers. As a result, all Ruby users should immediately upgrade to a newly released, patched version of the software.

That warning was sounded Tuesday in a [Google Groups](#) post made by Aaron Patterson, a key Ruby programmer. "Due to the critical nature of this vulnerability, and the fact that portions of it have been disclosed publicly, all users running an affected release should either upgrade or use one of the work arounds immediately," he wrote. The patched versions of Ruby on Rails (RoR) are 3.2.11, 3.1.10, 3.0.19 and 2.3.15.

As a result, [more than 240,000 websites](#) that use Ruby on Rails Web applications are at risk of being exploited by attackers. [High-profile websites](#)

**Now you urgently have to update
all your Rails installs**


```
> bundle update rails
```

```
Building native extensions. This could take a while...
```

```
ERROR: Error installing rails:
```

```
ERROR: Failed to build gem native extension.
```

```
    /usr/bin/ruby1.9.1extconf.rb
```

```
checking if the C compiler accepts... yes
```

```
Building nokogiri using packaged libraries.
```

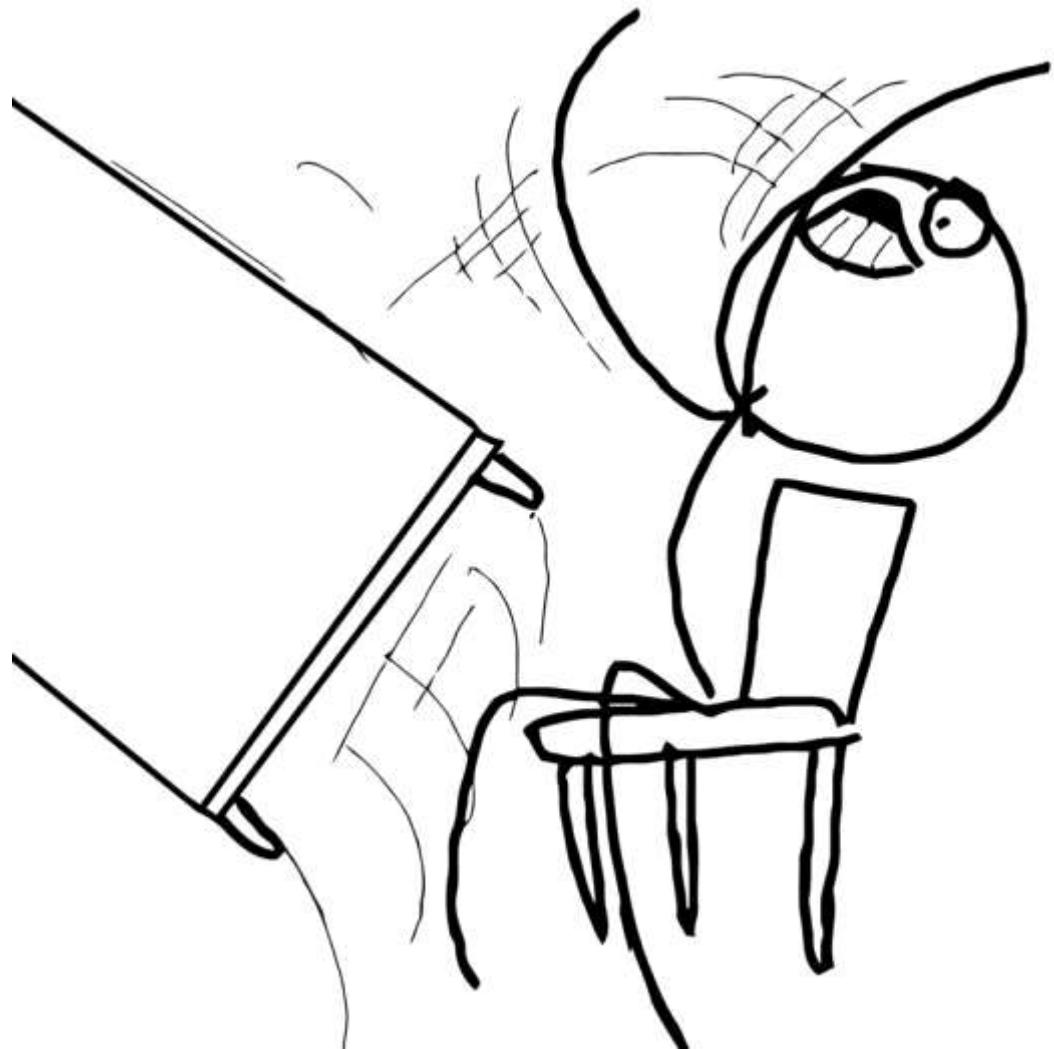
```
Using mini_portile version2.0.0.rc2
```

```
checking for gzopen() in -lz... yes
```

```
checking for iconv... yes
```

```
Extracting libxml2-2.9.2.tar.gz into tmp/x86_64-pc-linux-gnu/ports/libxml2/2.9.2... OK
```

```
*** extconf.rb failed ***
```



The problem isn't Rails

```
> ssh ec2-user@ec2-12-34-56-78.compute-1.amazonaws.com
```



```
[ec2-user@i p- 172- 31- 61- 204 ~]$ gem install rails
```

The problem is that you're
configuring servers manually

AWS Services Edit

Amazon Web Services

Compute

- EC2 Virtual Servers in the Cloud
- EC2 Container Service Run and Manage Docker Containers
- Elastic Beanstalk Run and Manage Web Apps
- Lambda Run Code in Response to Events

Storage & Content Delivery

- S3 Scalable Storage in the Cloud
- CloudFront Global Content Delivery Network
- Elastic File System PREVIEW Fully Managed File System for EC2
- Glacier Archive Storage in the Cloud
- Import/Export Snowball Large-Scale Data Transport
- Storage Gateway Hybrid Storage Integration

Database

- RDS Managed Relational Database Service
- DynamoDB Managed NoSQL Database
- ElastiCache

Developer Tools

- CodeCommit Store Code in Private Git Repositories
- CodeDeploy Automate Code Deployments
- CodePipeline Release Software using Continuous Delivery

Management Tools

- CloudWatch Monitor Resources and Applications
- CloudFormation Create and Manage Resources with Templates
- CloudTrail Track User Activity and API Usage
- Config Track Resource Inventory and Changes
- OpsWorks Automate Operations with Chef
- Service Catalog Create and Use Standardized Products
- Trusted Advisor Optimize Performance and Security

Security & Identity

- Identity & Access Management Manage User Access and Encryption Keys
- Directory Service Host and Manage Active Directory
- Inspector PREVIEW

Internet of Things

- AWS IoT Connect Devices to the Cloud

Mobile Services

- Mobile Hub BETA Build, Test, and Monitor Mobile Apps
- Cognito User Identity and App Data Synchronization
- Device Farm Test Android, FireOS, and iOS Apps on Real Devices in the Cloud
- Mobile Analytics Collect, View and Export App Analytics
- SNS Push Notification Service

Application Services

- API Gateway Build, Deploy and Manage APIs
- AppStream Low Latency Application Streaming
- CloudSearch Managed Search Service
- Elastic Transcoder Easy-to-Use Scalable Media Transcoding
- SES Email Sending and Receiving Service
- SQS Message Queue Service
- SWF

Resource Groups

A resource group is a collection of resources or more tags. Create a group for each production environment in your account.

Create a Group Tag Editor

Additional Resources

Getting Started ? Read our documentation or view our tutorials about AWS.

AWS Console Mobile App iOS View your resources on the go with our app, available from Amazon Appstore, Google Play, and iTunes.

AWS Marketplace ? Find and buy software, launch with 1-Click.

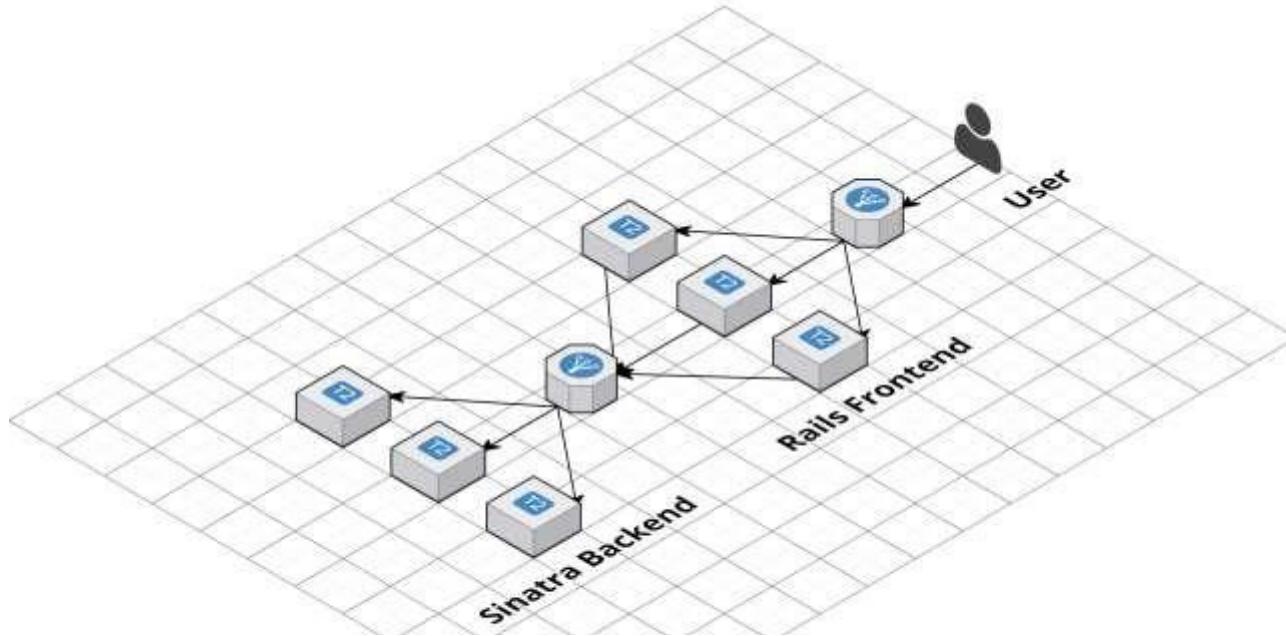
AWS re:Invent Announcements ? Explore the next generation of AWS cloud services and what's new.

Service Health

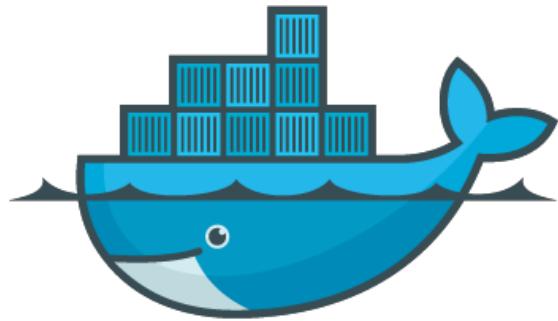
And that you're deploying infrastructure manually

A better alternative: **infrastructure-as-code**

In this talk, we'll go through a
real-world example:



We'll configure & deploy two
microservices on Amazon ECS



docker



TERRAFORM

**With two infrastructure-as-code
tools: Docker and Terraform**

Outline

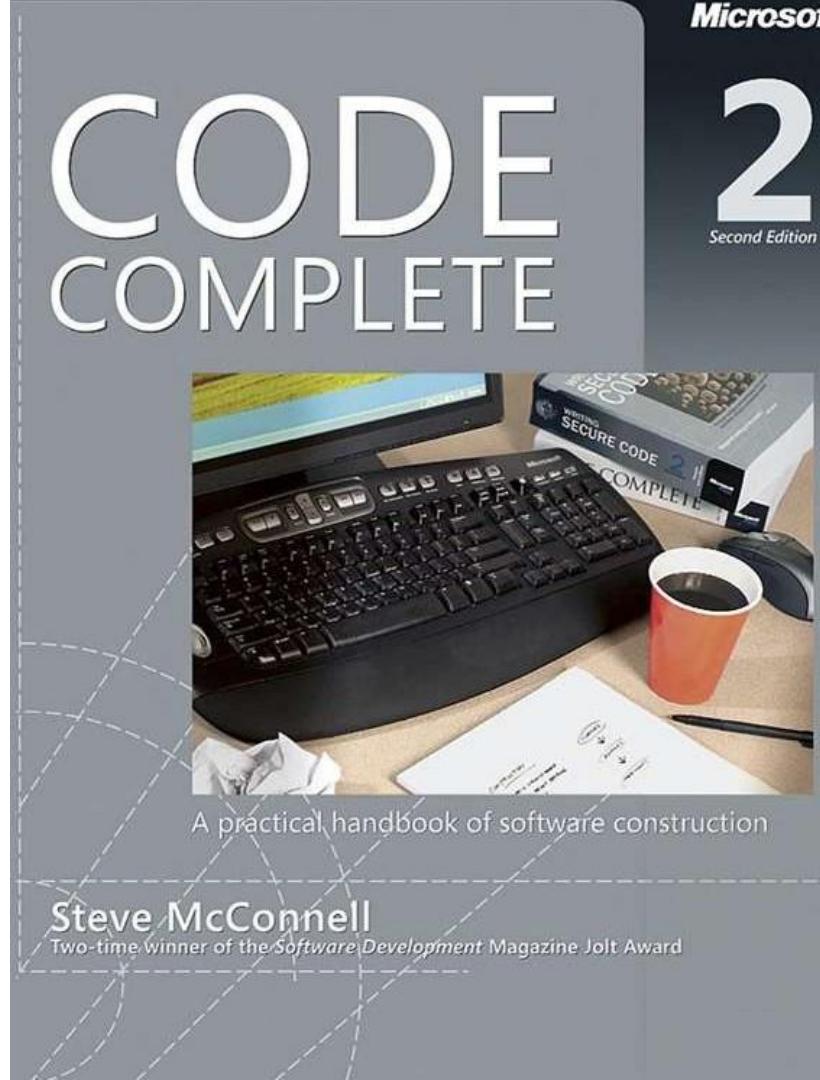
1. Microservices
2. Docker
3. Terraform
4. Recap

Outline

1. Microservices
2. Docker
3. Terraform
4. ECS
5. Recap

**Code is the enemy: the more you
have, the slower you go**

Project Size	Bug Density
Lines of code	Bugs per thousand lines of code
< 2K	0 – 25
2K – 6K	0 – 40
16K – 64K	0.5 – 50
64K – 512K	2 – 70
> 512K	4 – 100



**As the code grows, the number of
bugs grows even faster**

“Software development doesn't happen in a chart, an IDE, or a design tool; it happens in your head.”

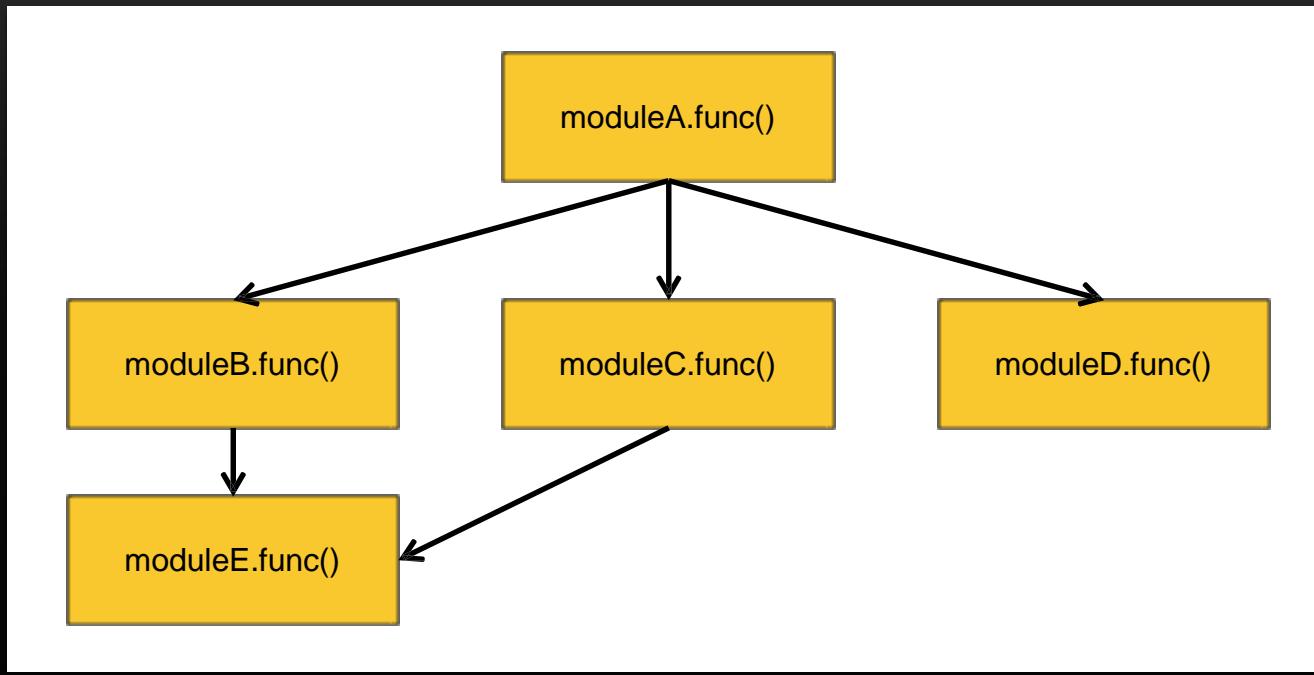
Practices of an Agile Developer



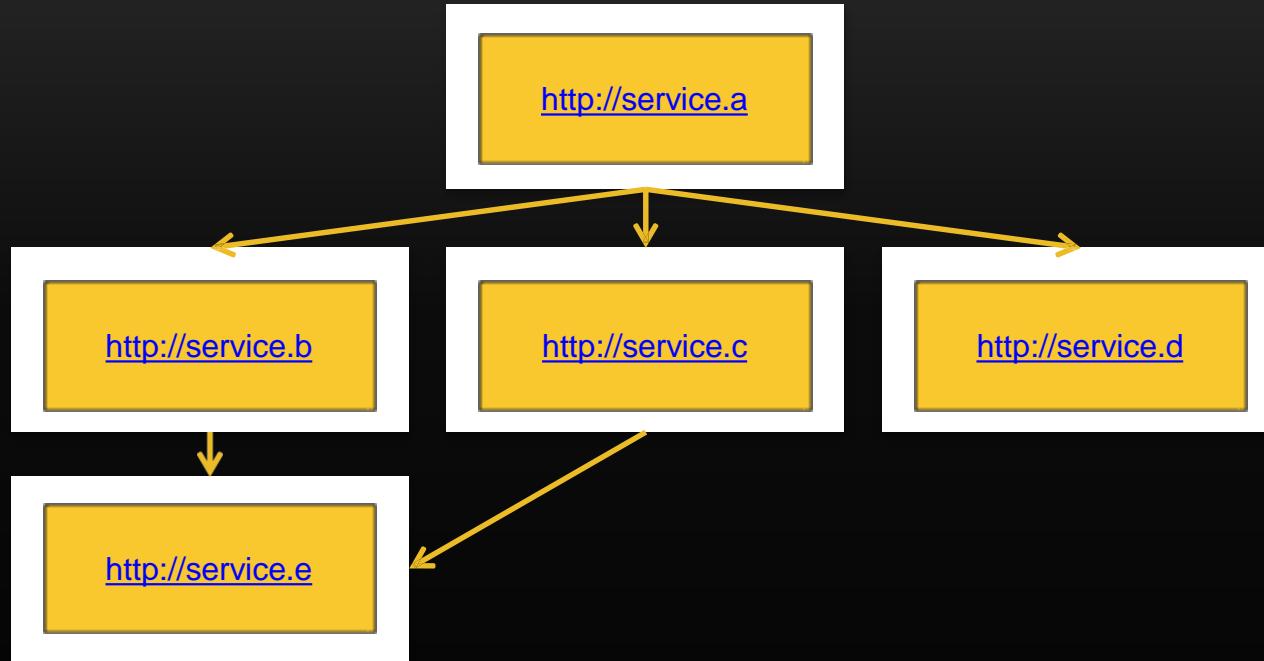
Venkat Subramaniam
Andy Hunt

The mind can only handle so
much complexity at once

**One solution is to break the code
into microservices**



In a monolith, you use function calls within one process



**With services, you pass messages
between processes**

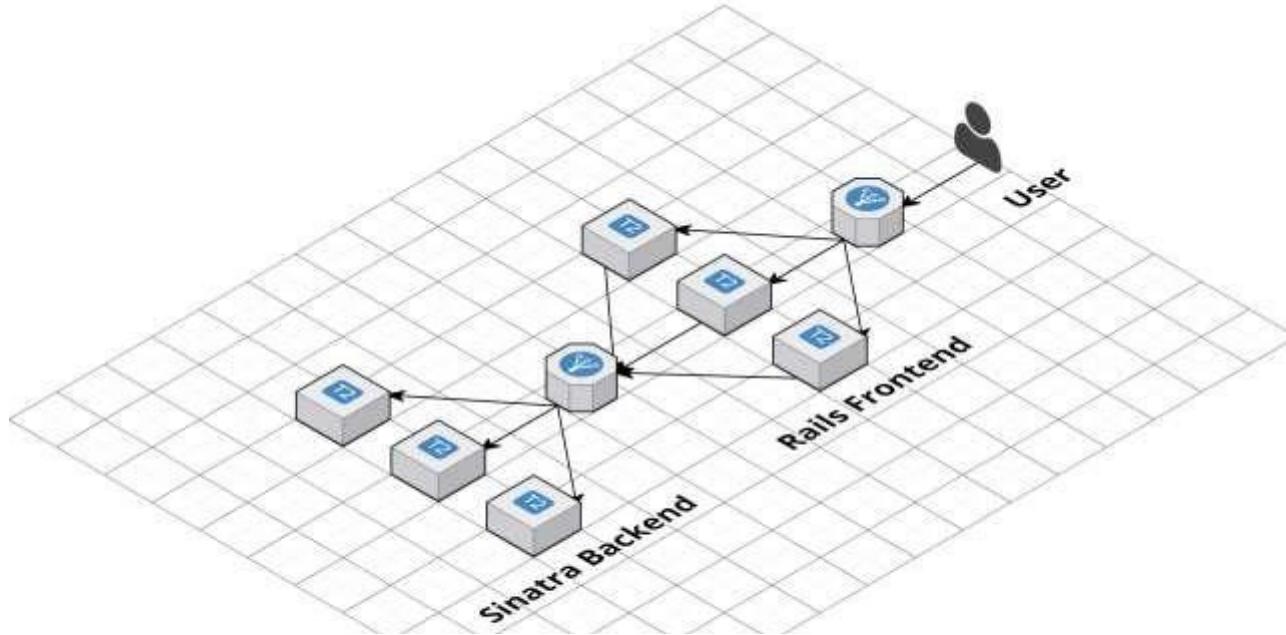
Advantages of services:

- 1. Isolation**
- 2. Technology agnostic**
- 3. Scalability**

Disadvantages of services:

- 1. Operational overhead**
- 2. Performance overhead**
- 3. I/O, error handling**
- 4. Backwards compatibility**
- 5. Global changes, transactions,
referential integrity all very hard**

For more info, see: Splitting Up a
Codebase into Microservices and
Artifacts



For this talk, we'll use two example microservices

```
require 'sinatra'  
  
get "/" do  
  "Hello, World!"  
end
```

A **sinatra backend** that returns
“Hello, World”

```
class ApplicationController <ActionController::Base
  def index
    url = URI.parse(backend_addr)
    req = Net::HTTP::Get.new(url.to_s)
    res = Net::HTTP.start(url.host, url.port) { |http|
      http.request(req)
    }
    @ext = res.body
  end
end
```

A **rails frontend that calls the sinatra backend**

```
<h1>Rails Frontend</h1>
<p>
  Response from the backend: <strong><%= @text %></strong>
</p>
```

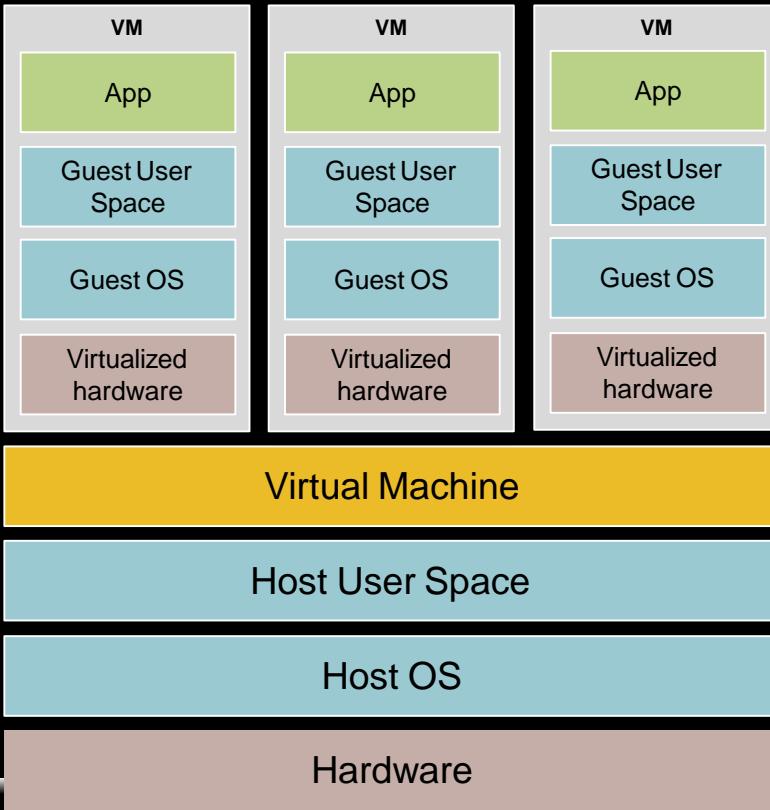
And renders the response as
HTML

Outline

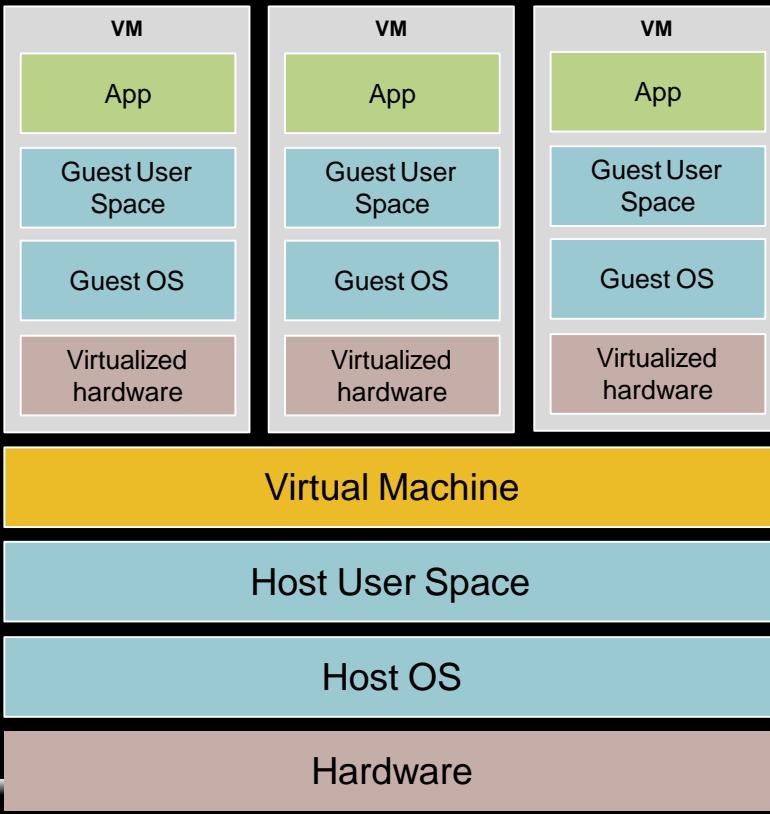
1. Microservices
2. Docker
3. Terraform
4. ECS
5. Recap

Docker allows you to build and
run code in containers

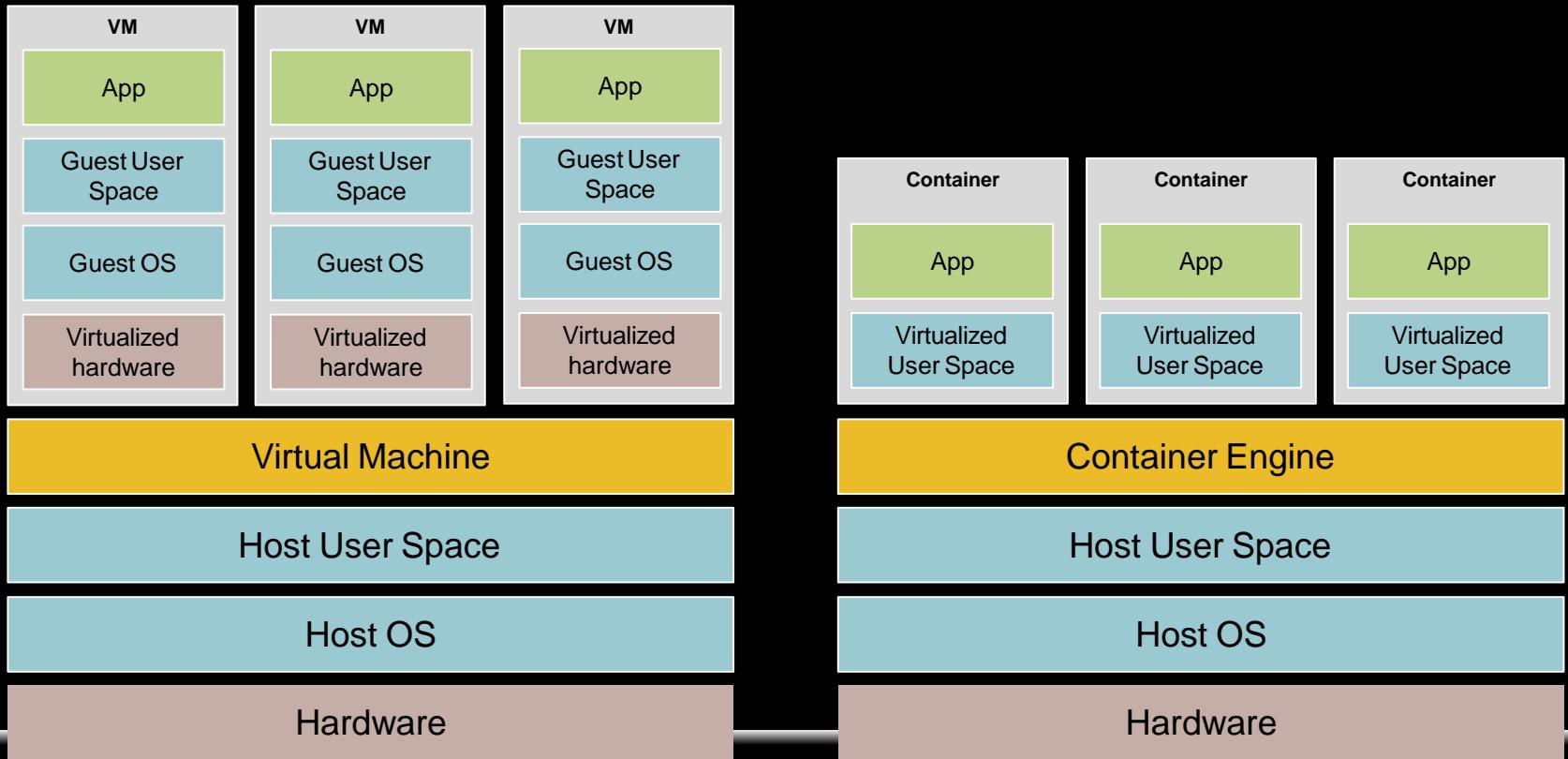
Containers are like **lightweight**
Virtual Machines (VMs)



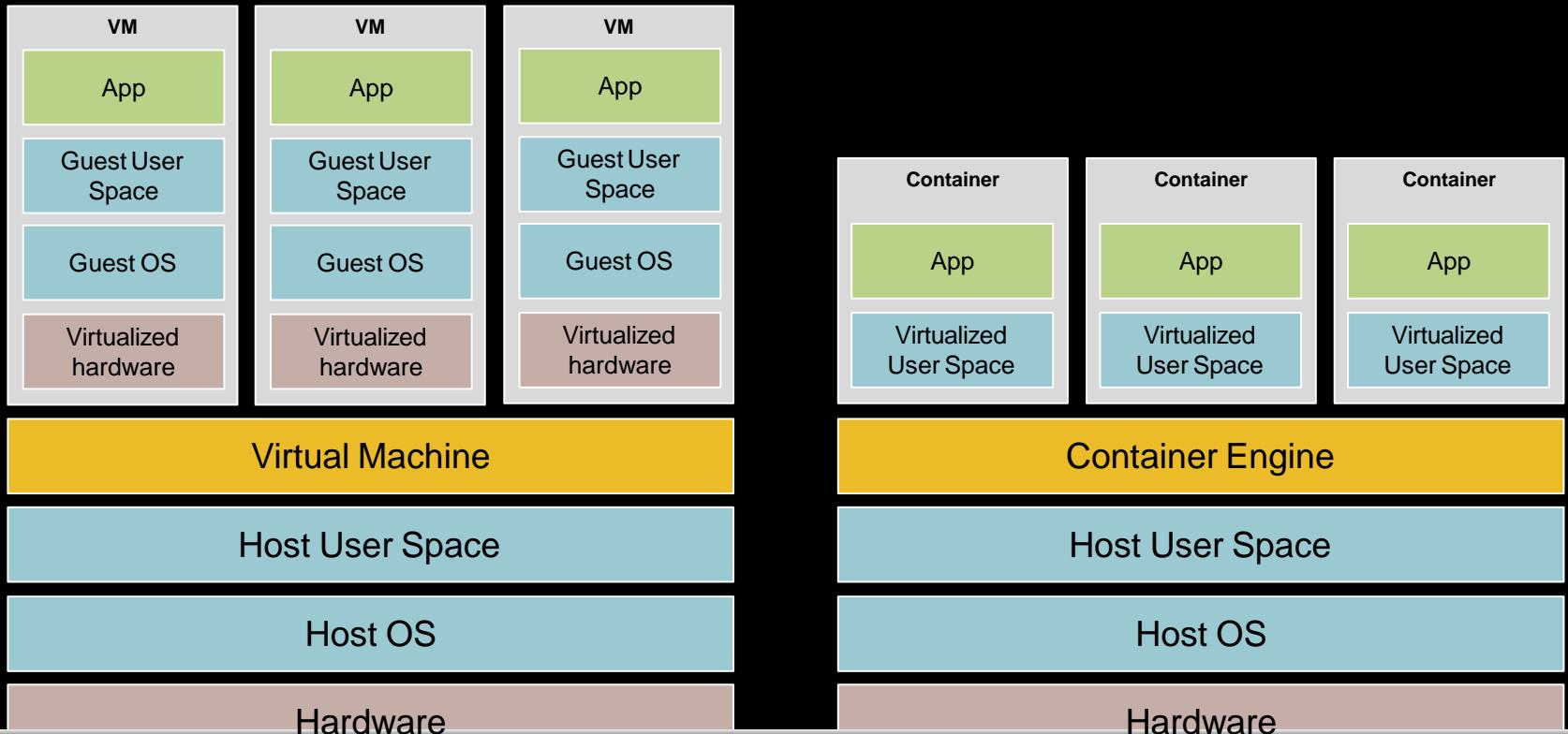
VMs virtualize the **hardware** and run an entire guest OS on top of the host OS



This provides good isolation, but lots of
CPU, memory, disk, & startup **overhead**



Containers virtualize User Space (shared memory, processes, mount, network)



Isolation isn't as good but much less CPU, memory, disk, startup overhead

```
> docker run -it ubuntu bash
```

```
root@12345:/# echo "I'm in $(cat /etc/issue)"
```

```
I'm in Ubuntu 14.04.4 LTS
```

Running **Ubuntu** in a Docker container

```
> time docker run ubuntu echo "Hello, World"
```

```
Hello, World
```

```
real 0m0.183s
```

```
user 0m0.009s
```

```
sys 0m0.014s
```

Containers boot **very quickly.
Easily run a dozen at once.**

You can define a Docker image
as code in a Dockerfile

```
FROM gliderlabs/alpine:3.3

RUN apk --no-cache add ruby ruby-dev
RUN gem install sinatra --no-ri --no-rdoc

RUN mkdir -p /usr/src/app
COPY . /usr/src/app
WORKDIR /usr/src/app

EXPOSE 4567
CMD ["ruby", "app.rb"]
```

Here is the Dockerfile for the
Sinatra backend

```
FROM gliderlabs/alpine:3.3

RUN apk --no-cache add ruby ruby-dev
RUN gem install sinatra --no-ri --no-rdoc

RUN mkdir -p /usr/src/app
COPY . /usr/src/app
WORKDIR /usr/src/app

EXPOSE 4567
CMD ["ruby", "app.rb"]
```

It specifies dependencies, code, config, and how to run the app

```
> docker build -t briki98/sinatra-backend .
```

```
Step 0 : FROM gliderlabs/alpine:3.3
```

```
---> 0a7e169bce21
```

```
( ... )
```

```
Step 8 : CMD ruby app.rb
```

```
---> 2e243eba30ed
```

```
Successfully built 2e243eba30ed
```

Build the Docker image

```
> docker run -it -p 4567:4567 brikis98/sinatra-backend
INFO  WEBrick 1.3.1
INFO  ruby 2.2.4 (2015-12-16) [x86_64-linux-musl]
== Sinatra (v1.4.7) has taken the stage on 4567 for
development with backup from WEBrick
INFO  WEBrick::HTTPServer#start: pid=1 port=4567
```

Run the Docker image

```
> docker push briki98/sinatra-backend
```

```
The push refers to a repository [docker.io/briki98/sinatra-  
backend] (len: 1)
```

```
2e243eba30ed: Image successfully pushed
```

```
7e2e0c53e246: Image successfully pushed
```

```
919d9a73b500: Image successfully pushed
```

(...)

```
v1: digest: sha256:09f48ed773966ec7fe4558 size: 14319
```

You can share your images by
pushing them to Docker Hub

**Now you can reuse the same
image in dev, stg, prod, etc**

```
> docker pull rails:4.2.6
```

And you can **reuse** images created by others.

```
FROM rails :4.2.6
```

```
RUN mkdir -p /usr/src/app
```

```
COPY . /usr/src/app
```

```
WORKDIR /usr/src/app
```

```
RUN bundle install
```

```
EXPOSE 3000
```

```
CMD [ "rails", "start" ]
```

The **rails-frontend** is built on top of
the **official rails Docker image**



No more insane install procedures!

```
rails_frontend:  
  image: brikis98/rails-frontend  
  ports:  
    - "3000:3000"  
  links:  
    - sinatra_backend:sinatra_backend
```

```
sinatra_backend:  
  image: brikis98/sinatra-backend  
  ports:  
    - "4567:4567"
```

Define your entire dev stack as code with docker-compose

```
rails_frontend:  
  image: brikis98/rails-frontend  
  ports:  
    - "3000:3000"  
  links:  
    - sinatra_backend
```

```
sinatra_backend:  
  image: brikis98/sinatra-backend  
  ports:  
    - "4567:4567"
```

Docker links provide a simple service discovery mechanism

```
> docker-compose up
```

```
Starting infrastructureascode talk_sinatra_backend_1
```

```
Recreating infrastructureascode talk_rails_frontend_1
```

```
sinatra_backend_1 | INFO  WEBrick 1.3.1
```

```
sinatra_backend_1 | INFO  ruby 2.2.4 (2015-12-16)
```

```
sinatra_backend_1 | Sinatra has taken the stage on 4567
```

```
rails_frontend_1 | INFO  WEBrick 1.3.1
```

```
rails_frontend_1 | INFO  ruby 2.3.0 (2015-12-25)
```

```
rails_frontend_1 | INFO  WEBrick::HTTPServer#start: port=3000
```

Run your entire dev stack with one command

Advantages of Docker:

- 1. Easy to create & share images**
- 2. Images run the same way in all environments (dev, test, prod)**
- 3. Easily run the entire stack in dev**
- 4. Minimal overhead**
- 5. Better resource utilization**

Disadvantages of Docker:

- 1. Maturity. Ecosystem developing very fast, but still a ways to go**
- 2. Tricky to manage persistent data in a container**
- 3. Tricky to pass secrets to containers**

Outline

1. Microservices
2. Docker
3. Terraform
4. ECS
5. Recap

Terraform is a tool for
provisioning infrastructure

PROVIDERS

- Atlas
- AWS
- Azure (Service Management)
- Azure (Resource Manager)
- Chef
- CenturyLinkCloud
- CloudFlare
- CloudStack
- Consul
- Datadog
- DigitalOcean
- DNSMadeEasy
- DNSSimple
- Docker
- Dyn

PROVIDERS

Terraform is used to create, manage, and manipulate infrastructure resources. Examples of resources include physical machines, VMs, network switches, containers, etc. Almost any infrastructure noun can be represented as a resource in Terraform.

Terraform is agnostic to the underlying platforms by supporting providers. A provider is responsible for understanding API interactions and exposing resources. Providers generally are an IaaS (e.g. AWS, DigitalOcean, GCE, OpenStack), PaaS (e.g. Heroku, CloudFoundry), or SaaS services (e.g. Atlas, DNSimple, CloudFlare).

Use the navigation to the left to read about the available providers.

Terraform supports many providers (cloud agnostic)

[» DOCUMENTATION HOME](#)

AWS PROVIDER

▶ EC2 RESOURCES

- aws_ami
- aws_ami_copy
- aws_ami_from_instance
- aws_app_cookie_stickiness_policy
- aws_autoscaling_group
- aws_autoscaling_lifecycle_hook
- aws_autoscaling_notification
- aws_autoscaling_policy
- aws_autoscaling_schedule
- aws_ebs_volume
- aws_elb

AWS_INSTANCE

Provides an EC2 instance resource. This allows instances to be created, updated, and deleted. Instances also support [provisioning](#).

Example Usage

```
# Create a new instance of the 'ami-408c7f28' (Ubuntu 14.04) on an
# t1.micro node with an AWS Tag naming it "HelloWorld"
provider "aws" {
  region = "us-east-1"
}

resource "aws_instance" "web" {
  ami = "ami-408c7f28"
  instance_type = "t1.micro"
  tags {
    Name = "HelloWorld"
  }
}
```

And many **resources** for each provider

You define infrastructure as code
in Terraform **templates**

```
provider "aws" {  
    region = "us-east-1"  
}  
  
resource "aws_instance" "example" {  
    ami = "ami-408c7f28"  
    instance_type = "t2.micro"  
}
```

This **template** creates a single EC2 instance in AWS

```
> terraform plan
+ aws_instance.example
  ami:                      ""  => "ami-408c7f28"
  instance_type:             ""  => "t2.micro"
  key_name:                  ""  => "<computed>"
  private_ip:                ""  => "<computed>"
  public_ip:                 ""  => "<computed>"
```

Plan: 1 to add, 0 to change, 0 to destroy.

Use the **plan command to see
what you're about to deploy**

```
> terraform apply
aws_instance.example: Creating...
  ami:          "" => "ami-408c7f28"
  instance_type: "" => "t2.micro"
  key_name:      "" => "<computed>"
  private_ip:    "" => "<computed>"
  public_ip:     "" => "<computed>"
aws_instance.example: Creation complete
```

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.

Use the **apply command to apply
the changes**

EC2 Management Console

https://console.aws.amazon.com/ec2/v2/home?region=us-east-1#instances:search=i-48a2fbcb;sort=instanceId

AWS Services Edit N. Virginia Support

EC2 Dashboard Events Tags Reports Limits

Instances

- Launch Instance
- Connect
- Actions

search : i-48a2fbcb Add filter

1 to 1 of 1

Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS
	i-48a2fbcb	t2.micro	us-east-1d	running	2/2 checks ...	None	ec2-54-84-21-59.compu...

Public DNS: ec2-54-84-21-59.compute-1.amazonaws.com

Feedback English © 2008 - 2016, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use

A screenshot of the AWS EC2 Management Console. The left sidebar shows navigation options like EC2 Dashboard, Instances (which is selected), and Images. The main content area displays a table of instances. One instance is listed: Name i-48a2fbcb, Instance ID i-48a2fbcb, Instance Type t2.micro, Availability Zone us-east-1d, Instance State running, Status Checks 2/2 checks ..., Alarm Status None, and Public DNS ec2-54-84-21-59.compute-1.amazonaws.com. A search bar at the top is set to search for the instance ID. The bottom of the page shows the public DNS again and links for Feedback, English language selection, and legal notices.

Now our EC2 instance is running!

```
resource "aws_instance" "example" {
    ami = "ami-408c7f28"
    instance_type = "t2.micro"
    tags{
        Name = "terraform-example"
    }
}
```

Let's give the EC2 instance a **tag** with a readable name

```
> terraform plan  
~ aws_instance.example  
  tags.#: "0" => "1"  
  tags.Name: "" => "terraform-example"
```

Plan: 0 to add, 1 to change, 0 to destroy.

Use the **plan command again to verify your changes**

```
> terraform apply
aws_instance.example: Refreshing state...
aws_instance.example: Modifying...
  tags.#:    "0" => "1"
  tags.Name: ""  => "terraform-example"
aws_instance.example: Modifications complete

Apply complete! Resources: 0 added, 1 changed, 0 destroyed.
```

Use the **apply** command again to
deploy those changes

EC2 Management Console

https://console.aws.amazon.com/ec2/v2/home?region=us-east-1#instances:search=terraform-example;sort=instanceId

AWS Services Edit N. Virginia Support

EC2 Dashboard Events Tags Reports Limits

Instances

- Launch Instance
- Connect
- Actions

search : terraform-example Add filter

1 to 1 of 1

Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS
terraform-example	i-f3d58c70	t2.micro	us-east-1d	running	Initializing	None	ec2-54-88-184-

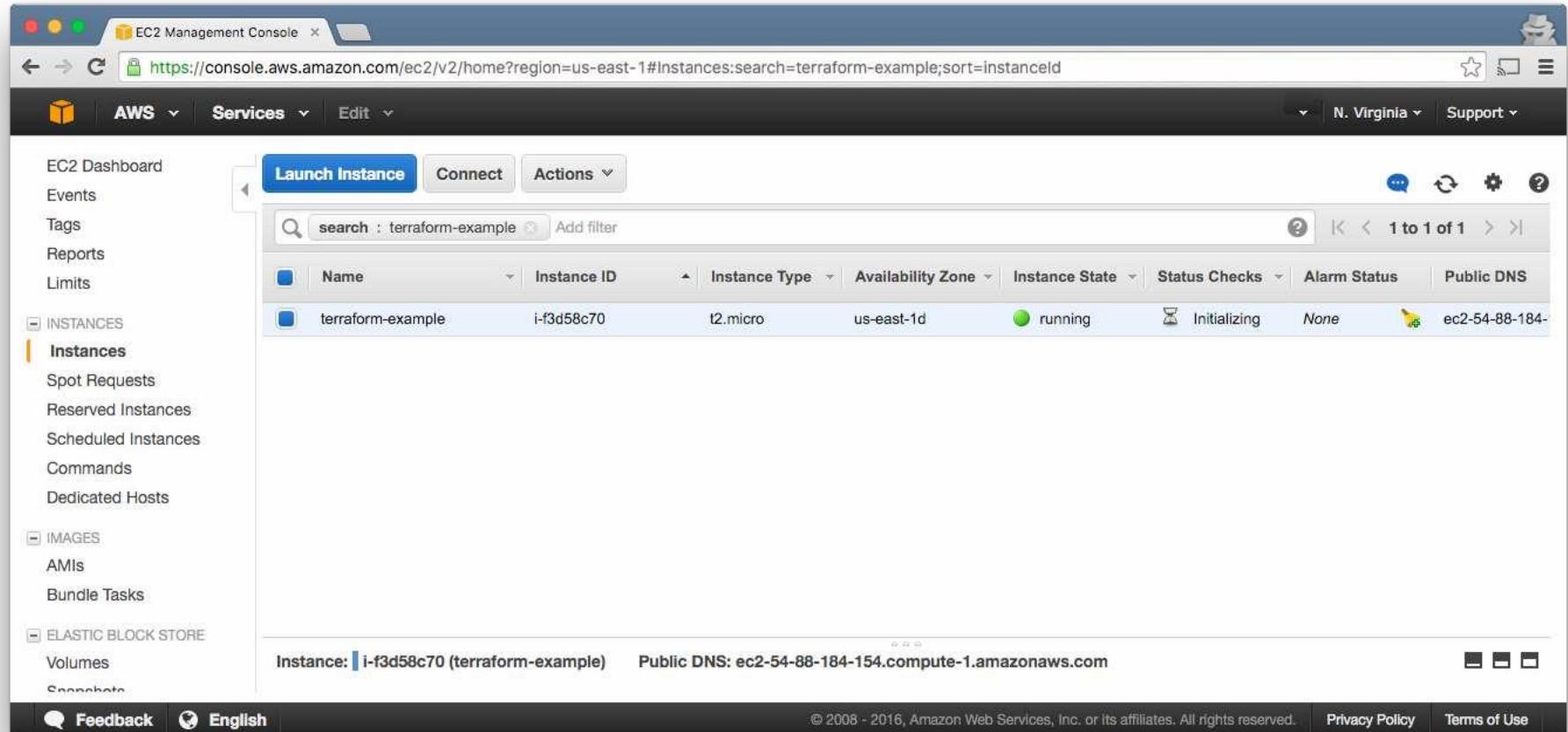
SPOT Requests Reserved Instances Scheduled Instances Commands Dedicated Hosts

AMIs Bundle Tasks

Volumes Snapshots

Instance: i-f3d58c70 (terraform-example) Public DNS: ec2-54-88-184-154.compute-1.amazonaws.com

Feedback English © 2008 - 2016, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use



Now our EC2 instance has a tag!

```
resource "aws_elb" "example" {
    name = "example"
    availability_zones = ["us-east-1a", "us-east-1b"]
    instances=["${aws_instance.example.id}"]
    listener {
        lb_port = 80
        lb_protocol = "http"
        instance_port = "${var.instance_port}"
        instance_protocol = "http"
    }
}
```

Let's add an **Elastic Load Balancer (ELB)**.

```
resource "aws_elb" "example" {
    name = "example"
    availability_zones = ["us-east-1a", "us-east-1b"]
    instances=["${aws_instance.example.id}"]
    listener {
        lb_port = 80
        lb_protocol = "http"
        instance_port = "${var.instance_port}"
        instance_protocol = "http"
    }
}
```

**Terraform supports variables,
such as var.instance_port**

```
resource "aws_elb" "example" {
    name = "example"
    availability_zones = ["us-east-1a", "us-east-1b"]
    instances=["${aws_instance.example.id}"]
    listener {
        lb_port = 80
        lb_protocol = "http"
        instance_port = "${var.instance_port}"
        instance_protocol = "http"
    }
}
```

As well as **dependencies** like
`aws_instance.example.id`

```
resource "aws_elb" "example" {  
    name = "example"  
    availability_zones = ["us-east-1a", "us-east-1b"]  
    instances=["${aws_instance.example.id}"]  
    listener {  
        lb_port = 80  
        lb_protocol = "http"  
        instance_port = "${var.instance_port}"  
        instance_protocol = "http"  
    }  
}
```

It builds a dependency graph and applies it in parallel.

EC2 Management Console

https://console.aws.amazon.com/ec2/v2/home?region=us-east-1#LoadBalancers:search=example

AWS Services Edit

jim.brikman@atomic-squirrel ... N. Virginia Support

EC2 Dashboard Events Tags Reports Limits

INSTANCES

- Instances
- Spot Requests
- Reserved Instances
- Scheduled Instances
- Commands
- Dedicated Hosts

IMAGES

- AMIs
- Bundle Tasks

ELASTIC BLOCK STORE

- Volumes
- Snapshots

Create Load Balancer Actions

Filter: example

Load Balancer Name	DNS Name	Port Configuration	Availability Zones	Instance Count	Health Check
example	example-1273249555.us-eas...	80 (HTTP) forwarding to 8080 (HTTP)	us-east-1a, us-east-1b	1 Instance	TCP:8080

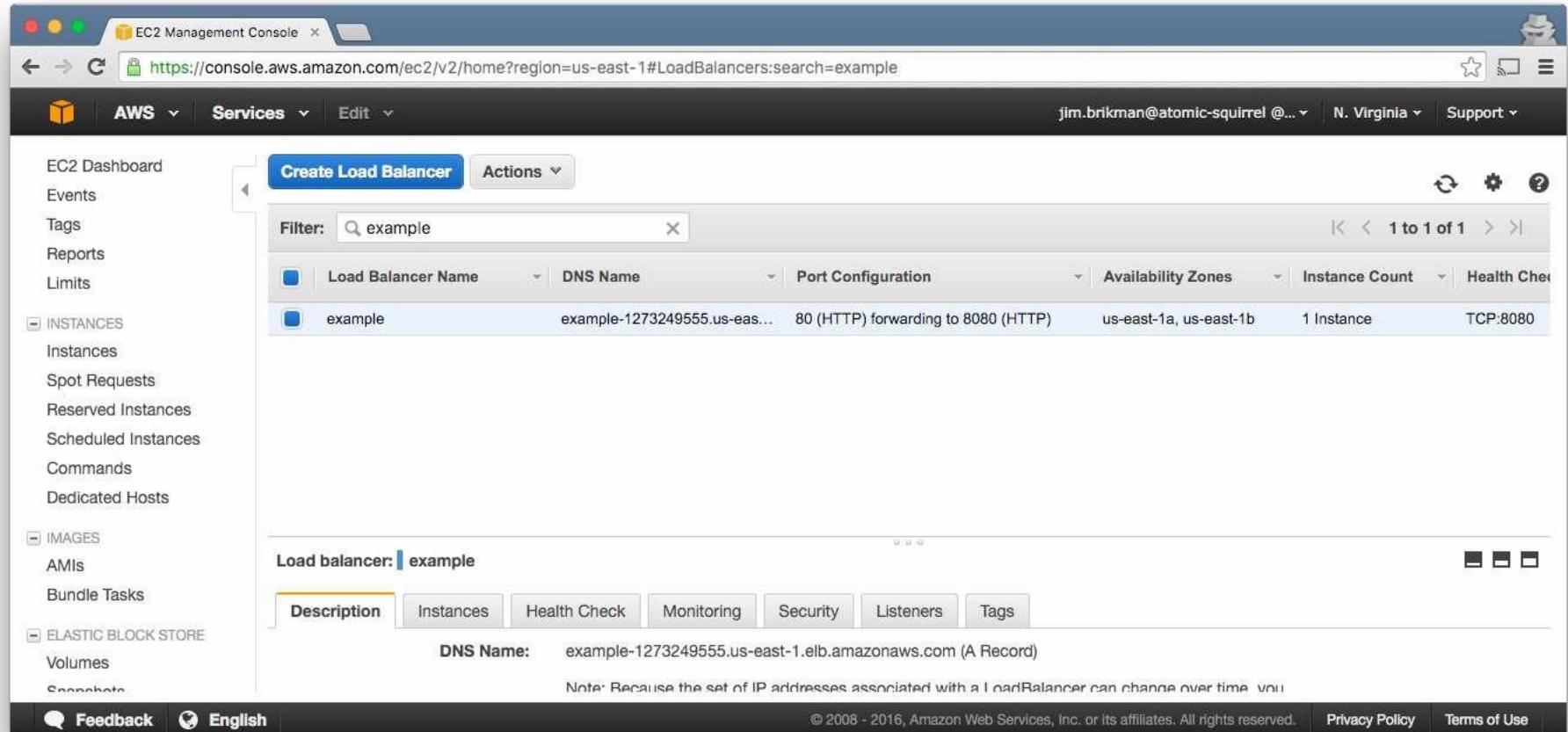
Load balancer: example

Description Instances Health Check Monitoring Security Listeners Tags

DNS Name: example-1273249555.us-east-1.elb.amazonaws.com (A Record)

Note: Because the set of IP addresses associated with a LoadBalancer can change over time, you

Feedback English © 2008 - 2016, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use



After running apply, we have an ELB!

```
> terraform destroy
aws_instance.example: Refreshing state... (ID: i-f3d58c70)
aws_elb.example: Refreshing state... (ID: example)
aws_elb.example: Destroying...
aws_elb.example: Destruction complete
aws_instance.example: Destroying...
aws_instance.example: Destruction complete
```

Apply complete! Resources: 0 added, 0 changed, 2 destroyed.

Use the **destroy command to
delete all your resources**

Advantages of Terraform:

- 1. Concise, readable syntax**
- 2. Reusable code: inputs, outputs, modules**
- 3. Plan command!**
- 4. Cloud agnostic**
- 5. Very active development**

Disadvantages of Terraform:

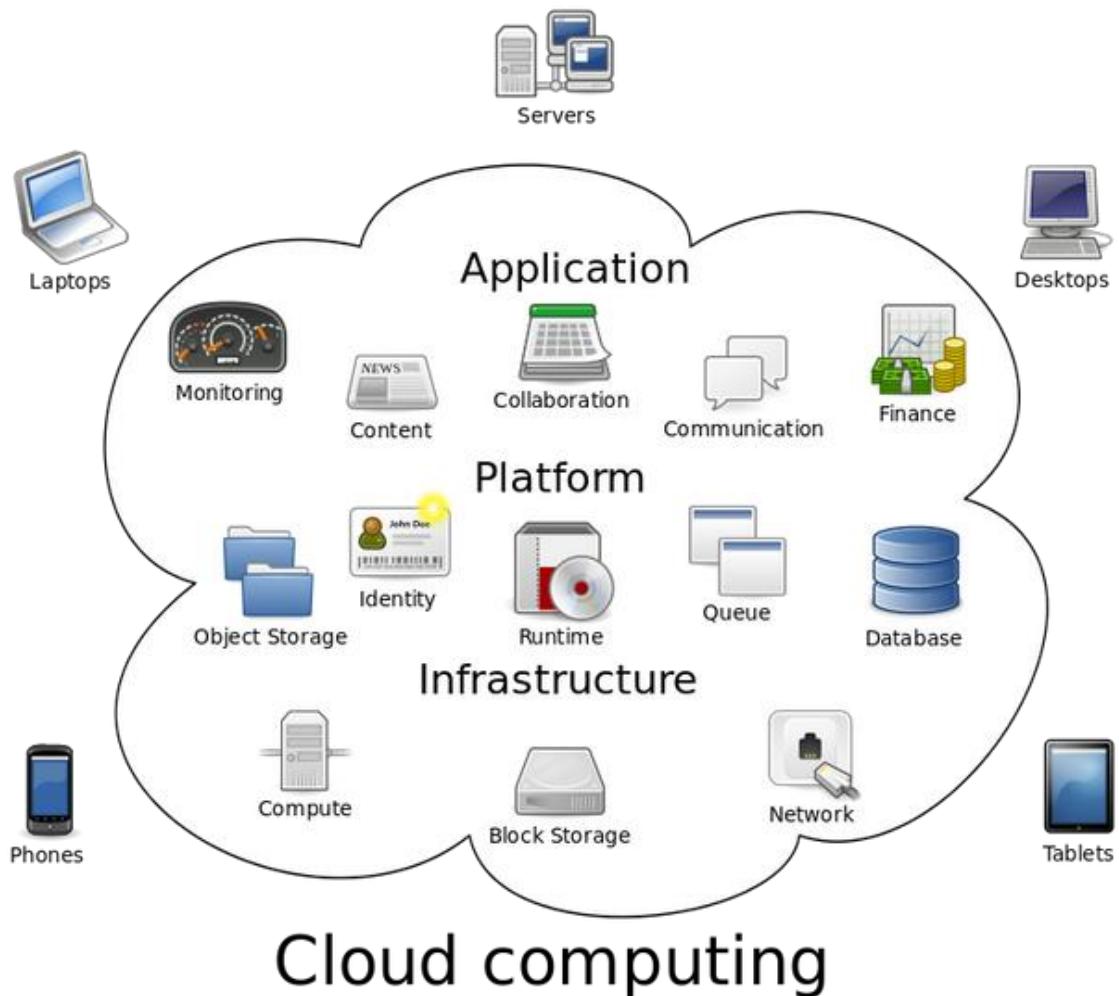
1. Maturity
2. Collaboration on Terraform state is hard (but terragrunt makes it easier)
3. No rollback
4. Poor secrets management

Outline

1. Microservices
2. Docker
3. Terraform
4. Recap

Benefits of infrastructure-as-code:

- 1. Reuse**
- 2. Automation**
- 3. Version control**
- 4. Code review**
- 5. Testing**
- 6. Documentation**





Questions?