

# INFRASTRUCTURE as CODE

The background of the slide is a faded architectural blueprint. It shows various technical drawings, including floor plans, elevations, and sections of a building. Labels like 'ENTRY', 'BATH NO. 2', and 'KITCHEN' are visible, along with numerous dimension lines and annotations. The blueprint is rendered in a light blue/gray tone, providing a technical and structural context for the 'Infrastructure as Code' theme.

Running Microservices on AWS with Docker<sub>1</sub>

# Microservices with Docker



**Rajesh Kumar**

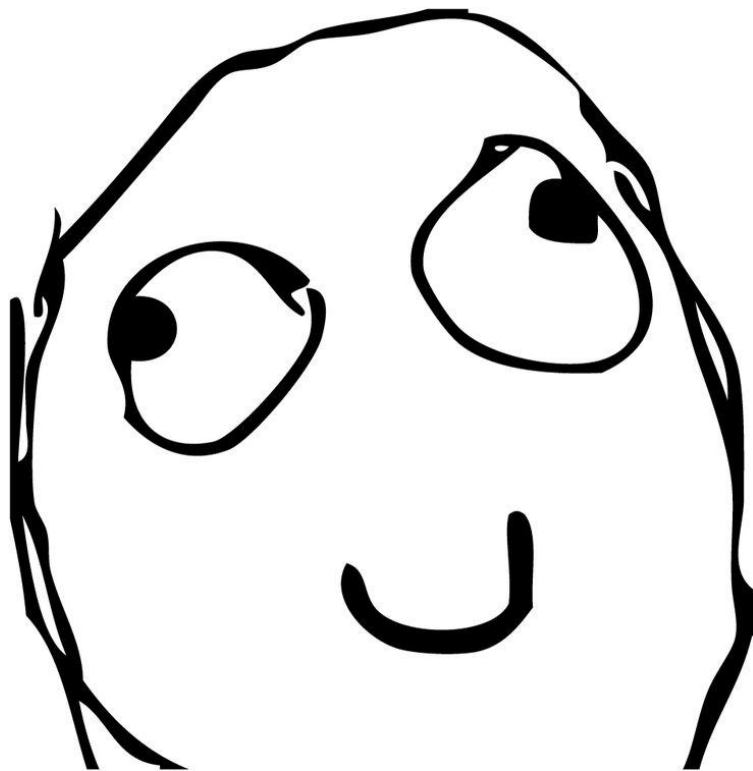
**DevOps Architect**

**@RajeshKumarIN | [www.RajeshKumar.xyz](http://www.RajeshKumar.xyz)**

---

**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:
```

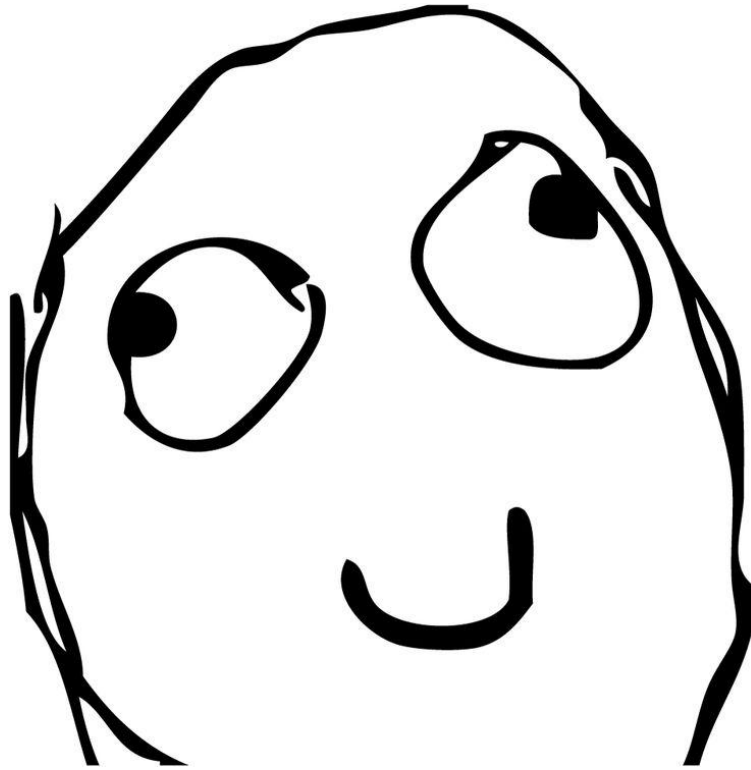
```
ERROR: Failed to build gem native extension.
```

```
      /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/ruby1.9.1 extconf.rb
```

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

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

```
Using mini_portile version 2.0.0.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/ruby1.9.1 extconf.rb
```

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

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

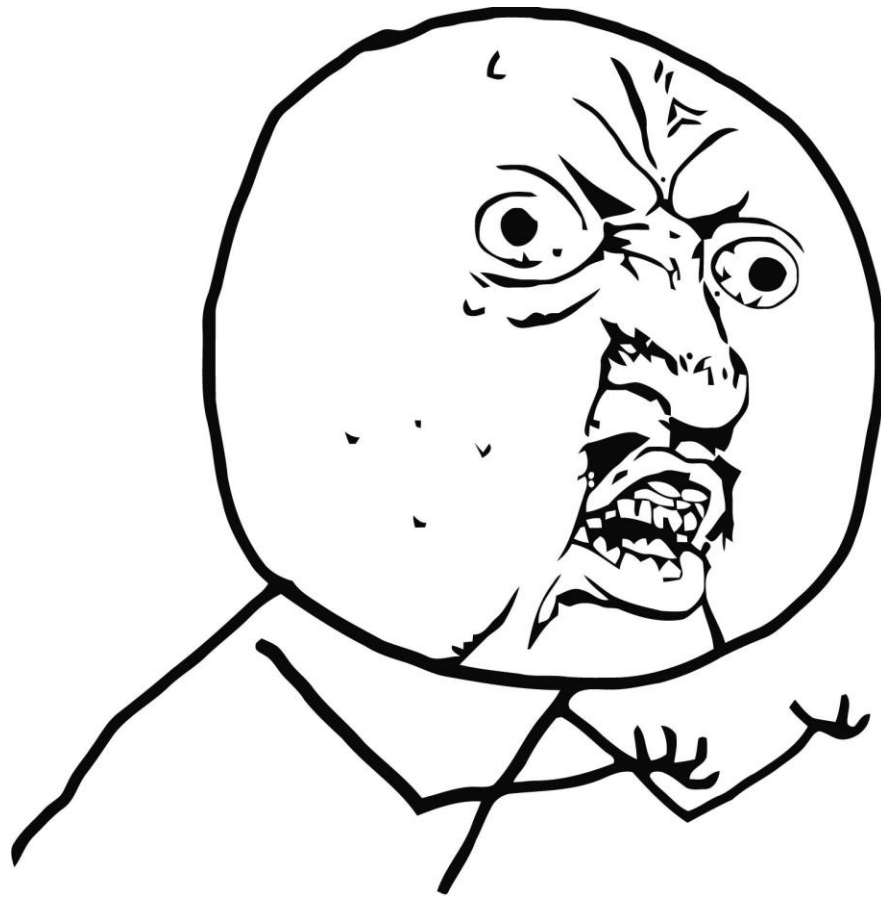
```
Using mini_portile version 2.0.0.rc2
```

```
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 ***
```



**nokogiri y u never install correctly?**



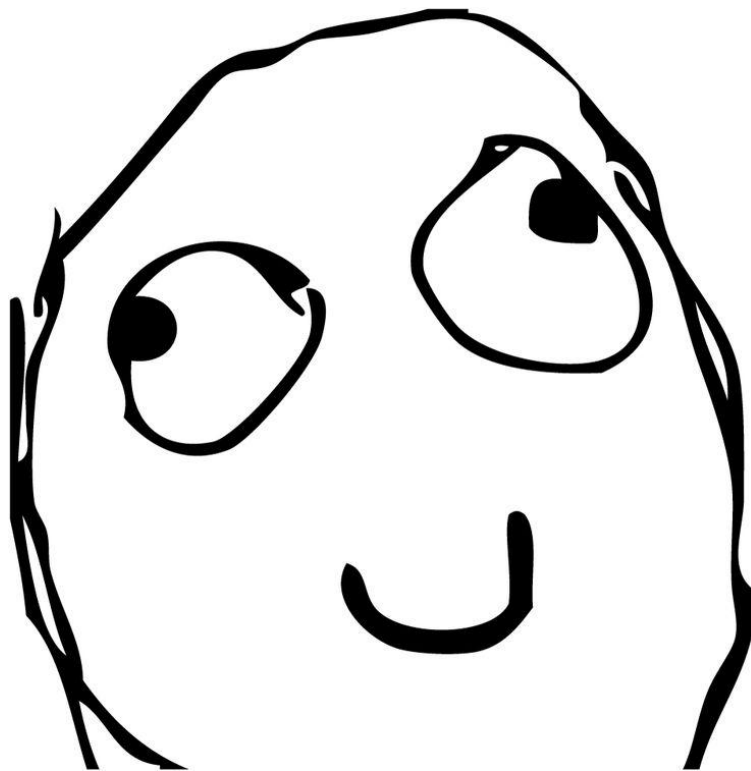
**(Spend 2 hours trying random  
StackOverflow suggestions)**

```
> gem install rails
```

```
> gem install rails
```

```
...
```

```
Success!
```

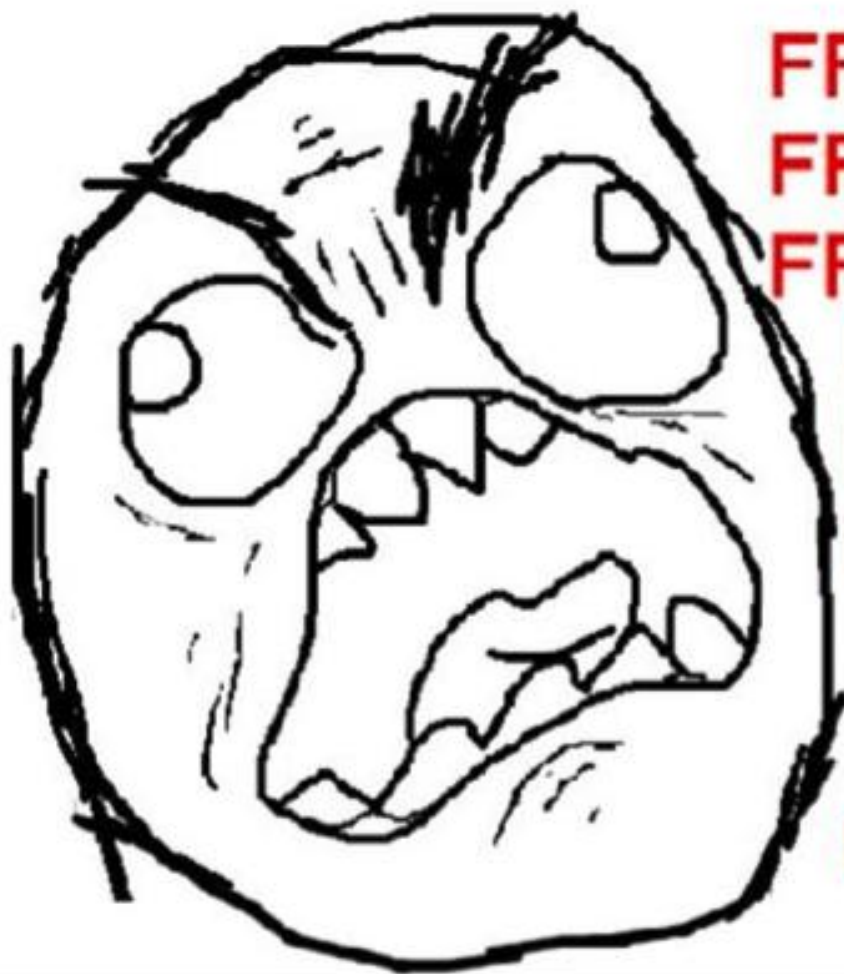


**Finally!**

```
> 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>'
```



FFFFFFFF  
FFFFFFFF  
FFFFFFFF  
FFFFF  
FFFFF  
FFFFF  
UUUU  
UUUU  
UUUU-

**Eventually**, you get it working



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

## 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 &amp; 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**  
In-Memory Cache
- Redshift**  
Fast, Simple, Cost-Effective Data Warehousing
- DMS** **PREVIEW**  
Managed Database Migration Service

## Networking

- VPC**  
Isolated Cloud Resources
- Direct Connect**  
Dedicated Network Connection to AWS
- Route 53**  
Scalable DNS and Domain Name Registration

## 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 &amp; Identity

- Identity & Access Management**  
Manage User Access and Encryption Keys
- Directory Service**  
Host and Manage Active Directory
- Inspector** **PREVIEW**  
Analyze Application Security
- WAF**  
Filter Malicious Web Traffic
- Certificate Manager**  
Provision, Manage, and Deploy SSL/TLS Certificates

## 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**  
Workflow Service for Coordinating Application Components

## Enterprise Applications

- WorkSpaces**  
Desktops and Applications in the Cloud
- WorkDocs**  
Secure Enterprise Document and Image Storage
- WorkMail**  
Secure Email and Calendaring Service

## Resource Groups

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

[Create a Group](#)[Tag Editor](#)

## Additional Resources

[Getting Started](#)

Read our documentation or view our training resources about AWS.

[AWS Console Mobile App](#)

View your resources on the go with our mobile app, available from [Amazon Appstore](#), [Google Play](#), and [iTunes](#).

[AWS Marketplace](#)

Find and buy software, launch with 1-click, and get 12 months free.

[AWS re:Invent Announcements](#)

Explore the next generation of AWS cloud services.

## Service Health

✓ All services operating normally.

Updated: Jan 31 2016 18:27:00 GMT-0500

[Service Health Dashboard](#)

# You use the AWS Console to deploy an EC2 instance

www.scmGalaxy.com

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

```
  _ | _ | _ )  
 _ | ( _ /   Amazon Linux AMI  
__ | \__ | __ |
```

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

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

```
  _ | _ | _ )  
 _ | ( / Amazon Linux AMI  
__ | \__ |__ |
```

```
[ec2-user@ip-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
```



FFFFFFFF  
FFFFFFFF  
FFFFFFFF  
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/ruby1.9.1 extconf.rb
```

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

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

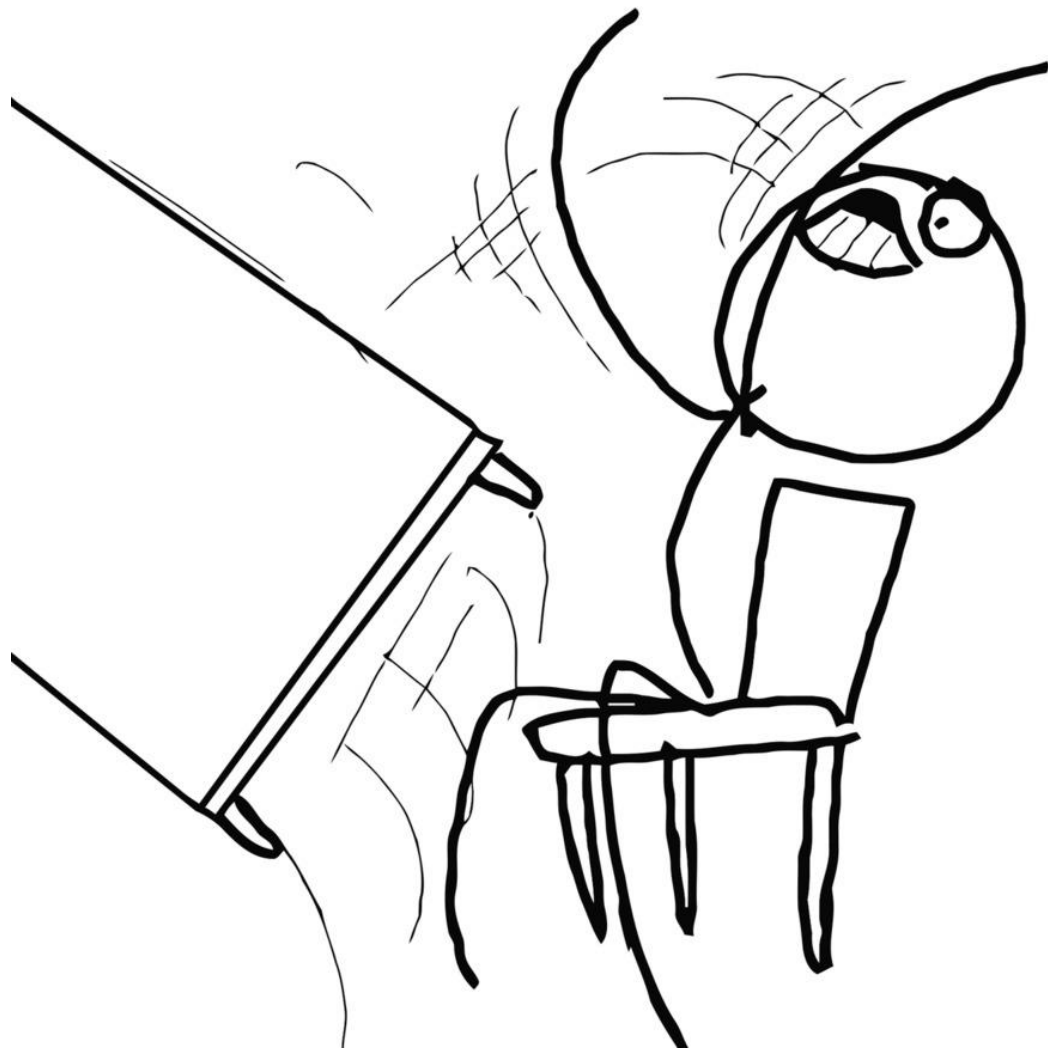
```
Using mini_portile version 2.0.0.rc2
```

```
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
```

```
  _ | _ | _ )  
  _ | ( _ /  Amazon Linux AMI  
  _ | \ _ | _ |
```

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

**The problem is that you're  
configuring servers manually**

## 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 &amp; 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**  
In-Memory Cache
- Redshift**  
Fast, Simple, Cost-Effective Data Warehousing
- DMS** **PREVIEW**  
Managed Database Migration Service

## Networking

- VPC**  
Isolated Cloud Resources
- Direct Connect**  
Dedicated Network Connection to AWS
- Route 53**  
Scalable DNS and Domain Name Registration

## 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 &amp; Identity

- Identity & Access Management**  
Manage User Access and Encryption Keys
- Directory Service**  
Host and Manage Active Directory
- Inspector** **PREVIEW**  
Analyze Application Security
- WAF**  
Filter Malicious Web Traffic
- Certificate Manager**  
Provision, Manage, and Deploy SSL/TLS Certificates
- Amazon Macie**  
Discover and Protect Sensitive Data in AWS
- Managed Hadoop Framework**  
Run Hadoop on Amazon EMR
- Data Pipeline**  
Orchestration for Data-Driven Workflows
- Amazon SageMaker**  
Train, Tune, and Deploy Machine Learning Models
- Machine Learning**  
Build Smart Applications Quickly and Easily

## 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**  
Workflow Service for Coordinating Application Components

## Enterprise Applications

- WorkSpaces**  
Desktops in the Cloud
- WorkDocs**  
Secure Email and Calendar Service
- WorkMail**  
Secure Email and Calendar Service

## Resource Groups

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

[Create a Group](#)[Tag Editor](#)

## Additional Resources

[Getting Started](#)

Read our documentation or view our training materials about AWS.

[AWS Console Mobile App](#)

View your resources on the go with our mobile app, available from [Amazon Appstore](#), [Google Play](#), and [iTunes](#).

[AWS Marketplace](#)

Find and buy software, launch with 1-click, and get support.

[AWS re:Invent Announcements](#)

Explore the next generation of AWS cloud services.

## Service Health

✓ All services operating normally.

Updated: Jan 31 2016 18:27:00 GMT-0500

[Service Health Dashboard](#)

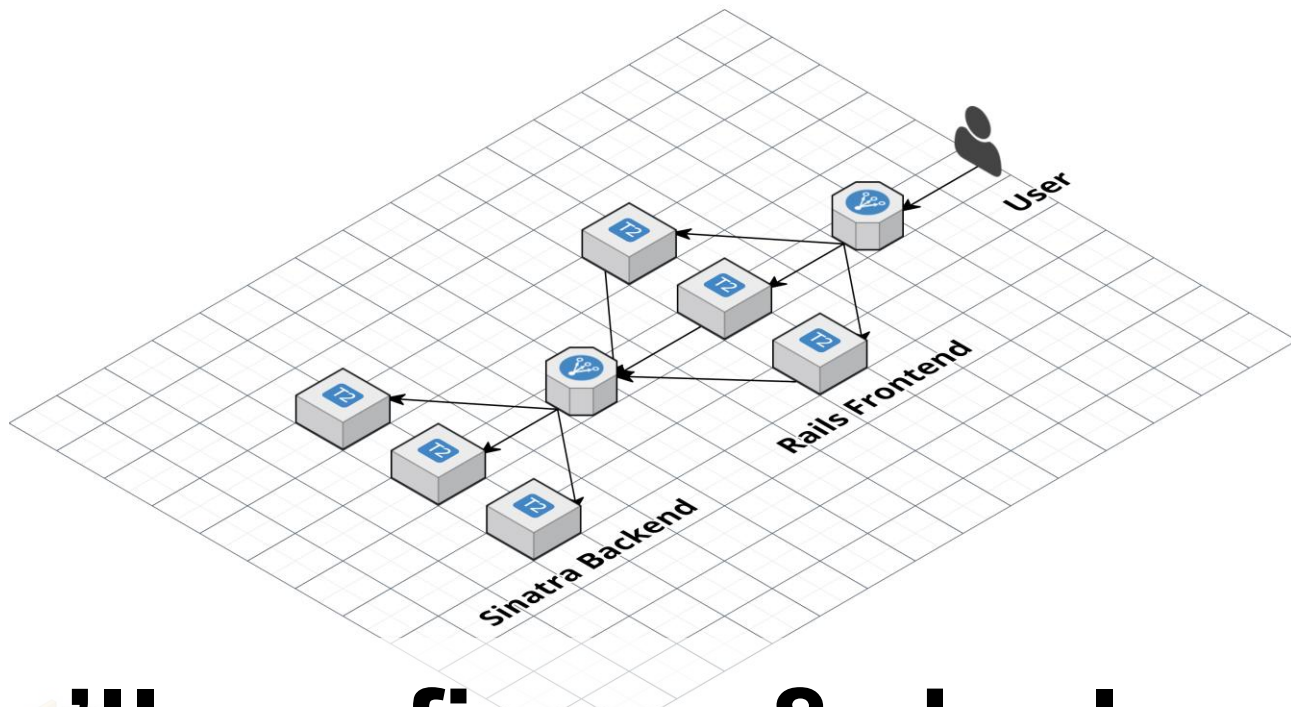
# And that you're deploying infrastructure manually

www.scmGalaxy.com

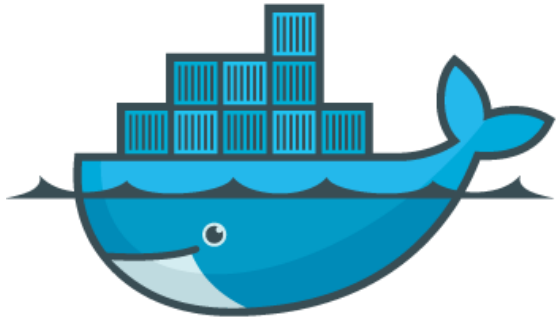
**A better alternative: infrastructure-  
as-code**

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





**We'll configure & deploy two microservices on AWS**



docker



TERRAFORM

**With infrastructure-as-code tools:**  
**Docker**

# Outline

1. **Microservices**
2. **Docker**
3. **Recap**

# Outline

1. **Microservices**
2. Docker
3. Recap

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



<b>Project Size</b> Lines of code	<b>Bug Density</b> 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

# CODE COMPLETE

**Microsoft**

**2**  
Second Edition



A practical handbook of software construction

**Steve McConnell**

Two-time winner of the *Software Development Magazine* Jolt Award

**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



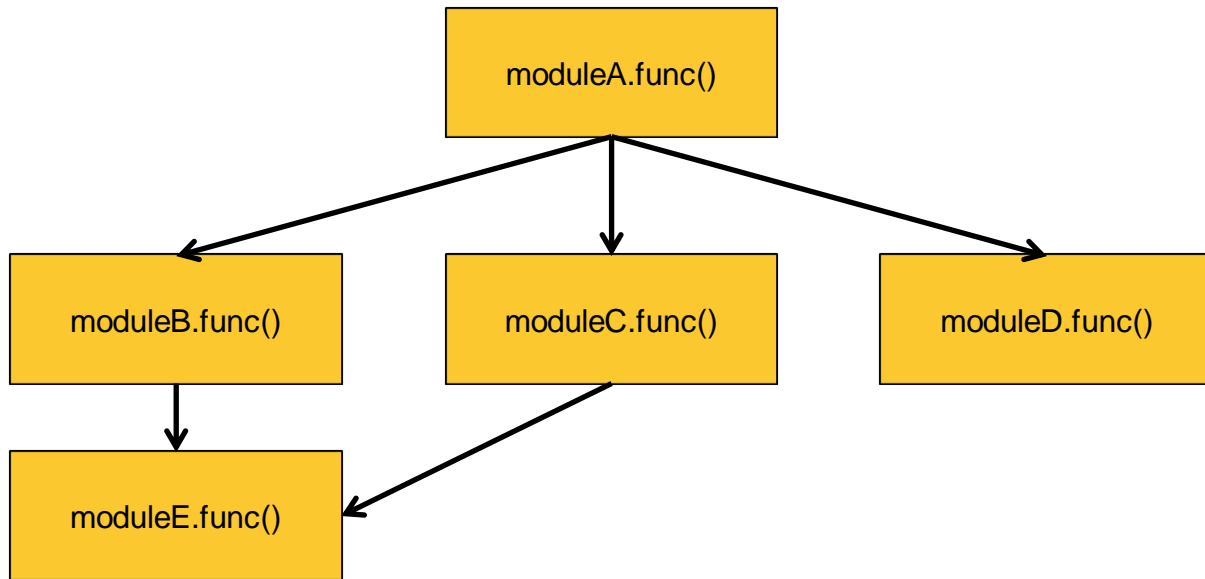
Venkat Subramaniam

Andy Hunt

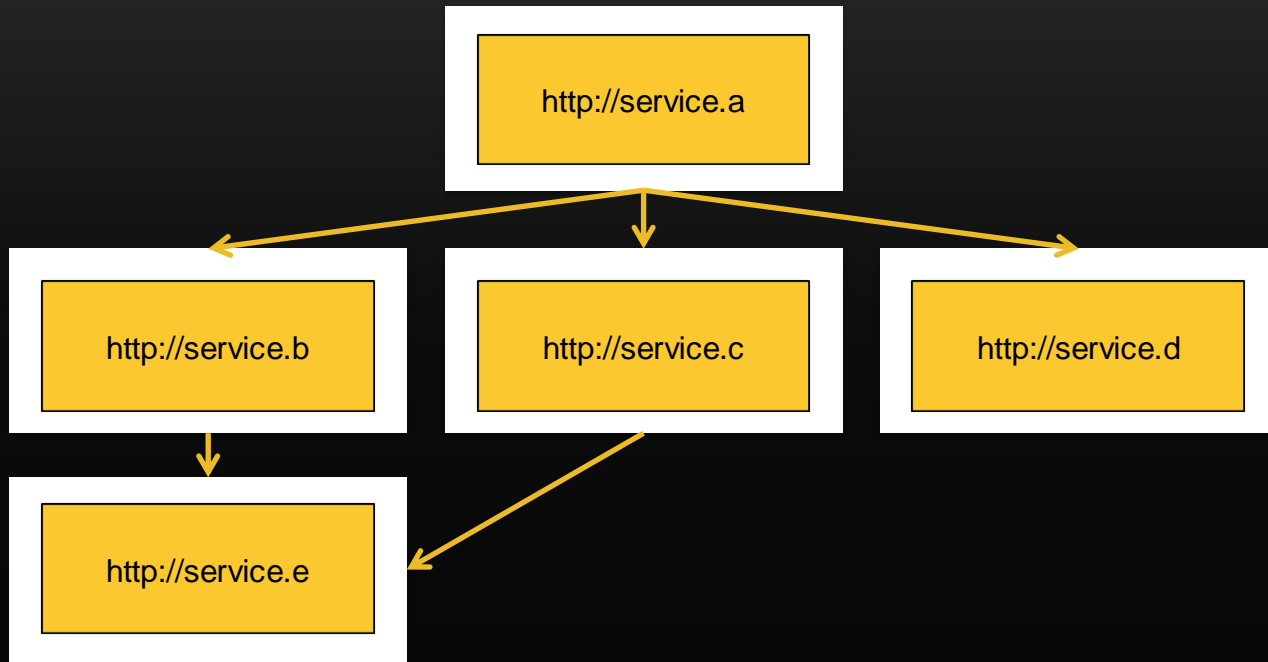


**The mind can only handle so  
much complexity at once**

**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**

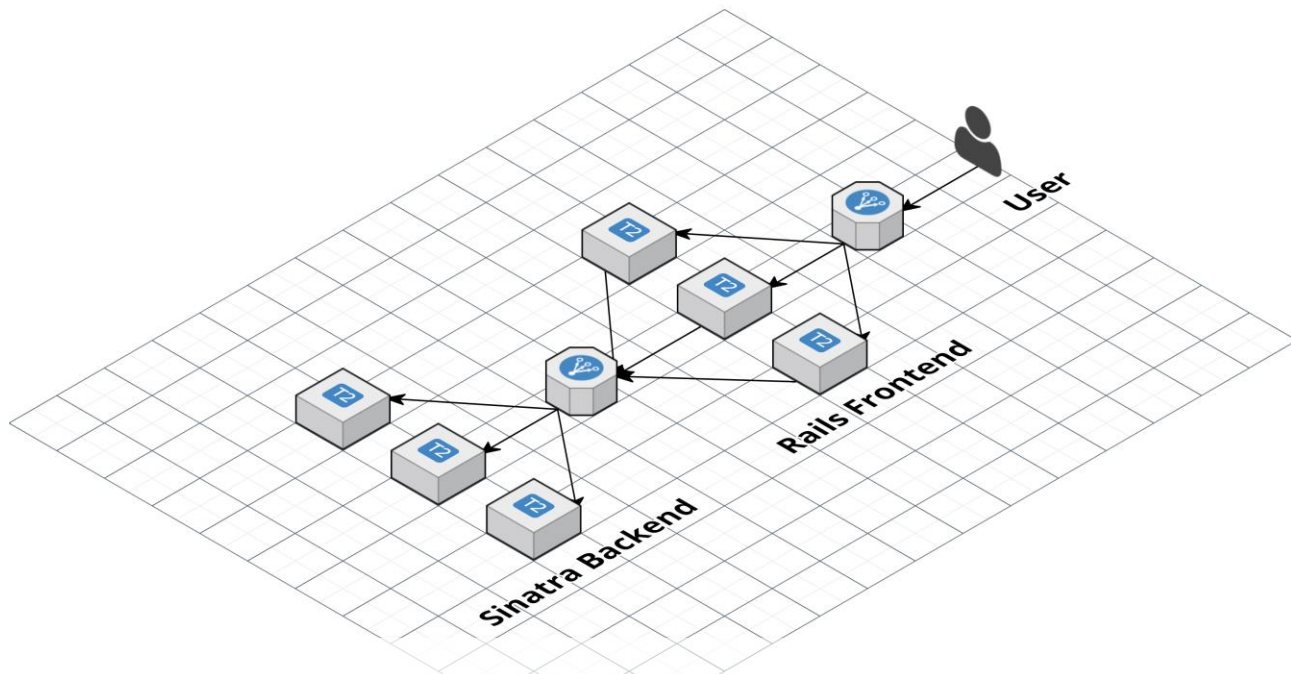
# Advantages of services:

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)
    }
    @text = 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>
```

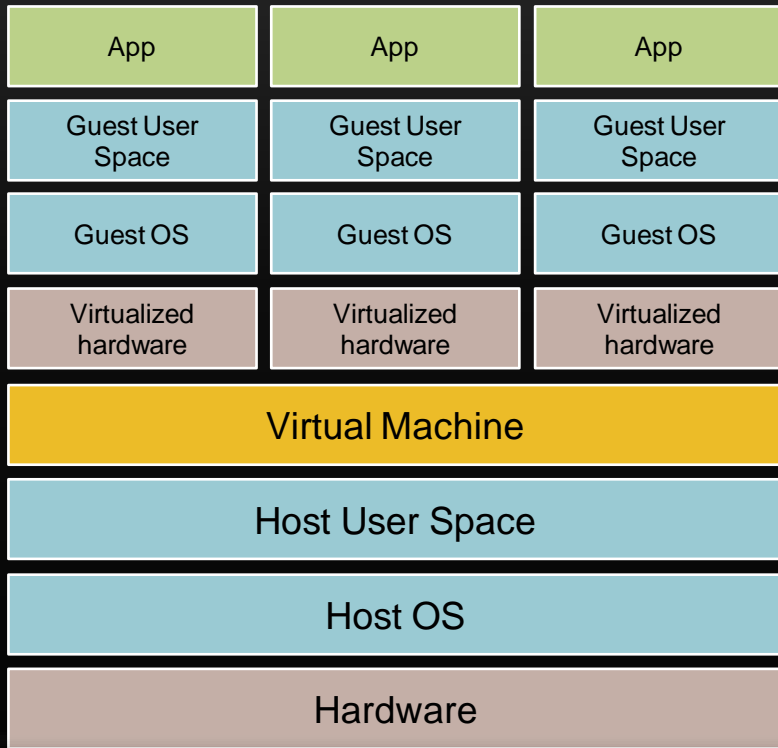
And renders the response as  
**HTML**

# Outline

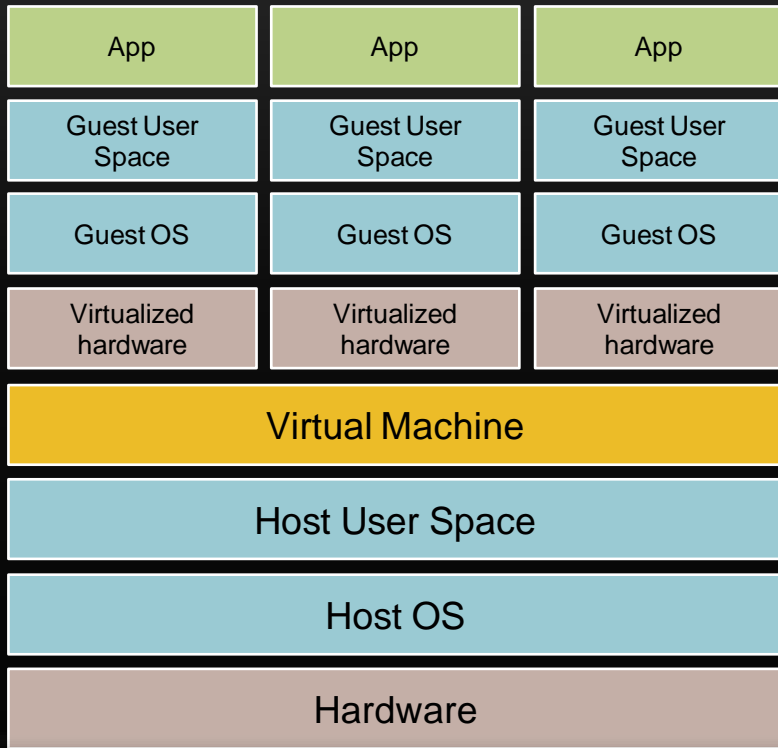
1. Microservices
2. **Docker**
3. Terraform
4. ECS
5. Recap

**Docker allows you to build and  
run code in containers**

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

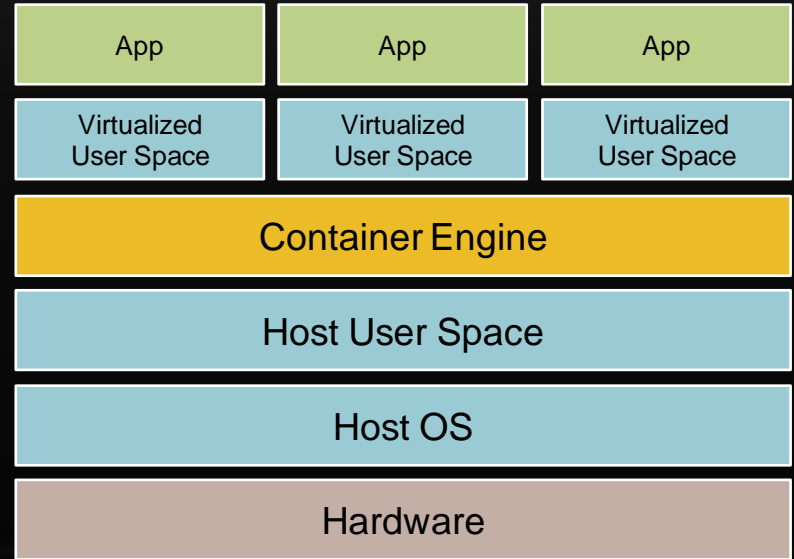
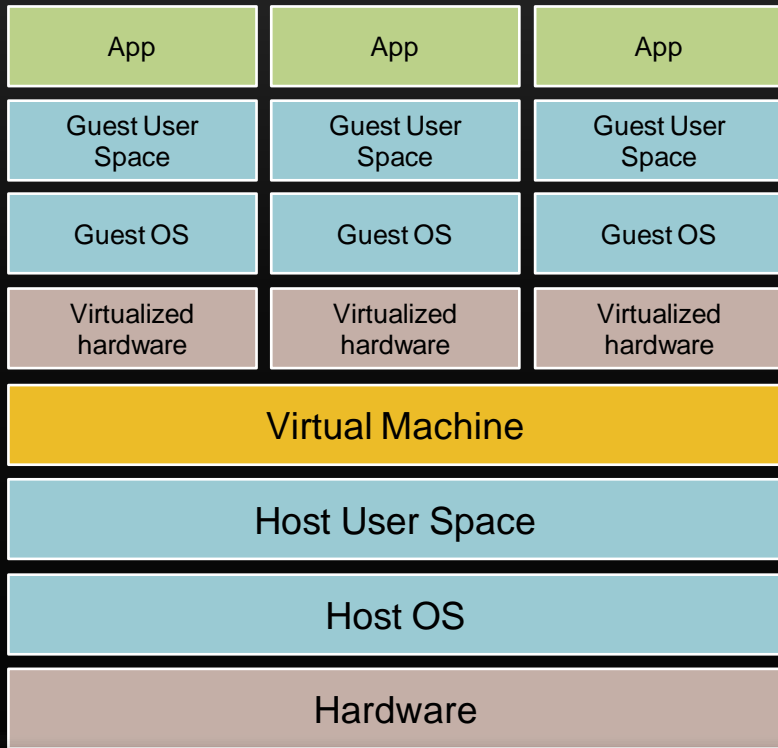


**VMs virtualize the **hardware** and run an entire guest OS on top of the host OS**

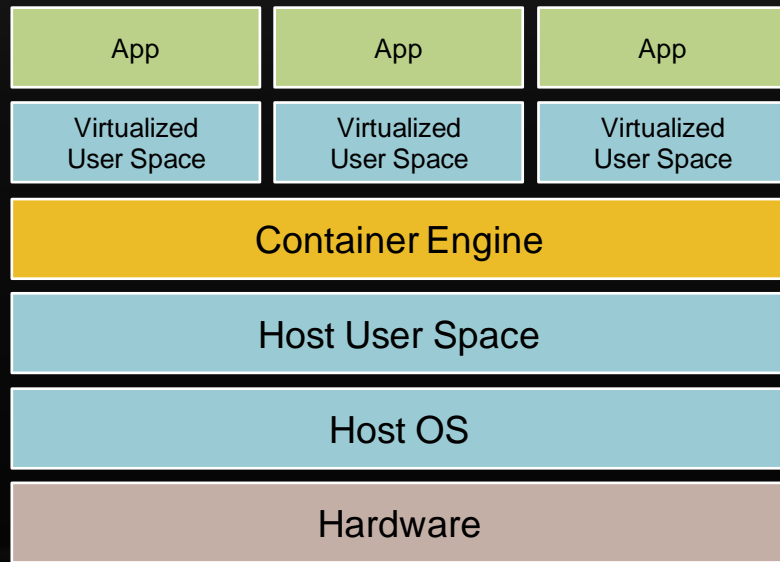
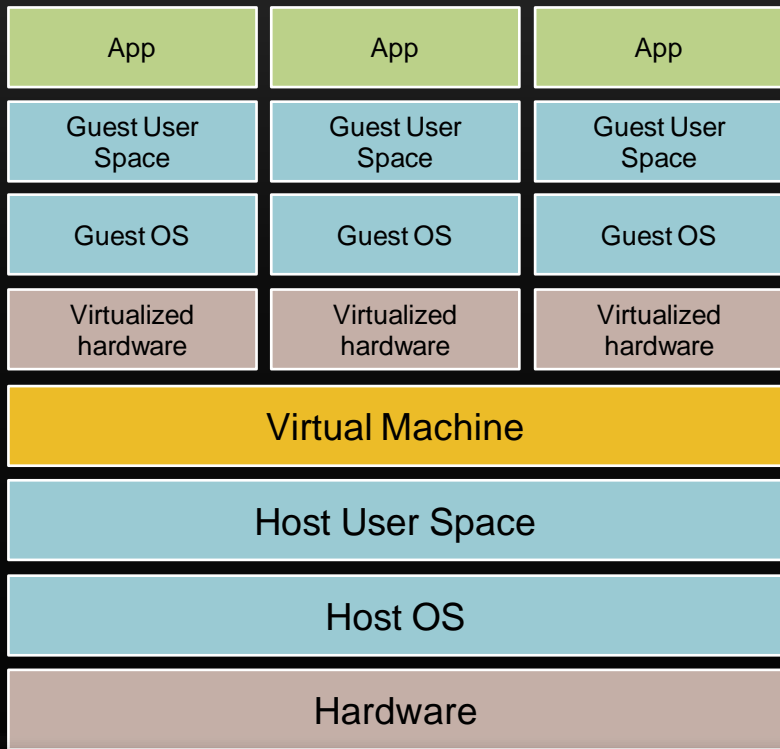


This provides good isolation, but lots of CPU, memory, disk, & startup **overhead**





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 brikis98/sinatra-backend .
```

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

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

```
(...)
```

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

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

```
Successfully built 2e243eba30ed
```

# Building a 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
```

## Running a Docker image

```
> docker push brikis98/sinatra-backend
```

```
The push refers to a repository [docker.io/brikis98/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

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

**Docker links** provide a simple  
service discovery mechanism



```
> docker-compose up
```

```
Starting infrastructureascodetalk_sinatra_backend_1
```

```
Recreating infrastructureascodetalk_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**

# Benefits of infrastructure-as-code:

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

**Slides and code from this talk:**  
**[ybrikman.com/speaking](https://ybrikman.com/speaking)**

# Questions?

