(ISC)²®

SECURITY CONGRESS | LATIN AMERICA 2016

ADVANCING SECURITY LEADERS

#ISC2CongressLATAM

latamcongress.isc2.org

Serverless Applications, Containers and Security

Anderson Dadario, CISSP, CSSLP
dadario.com.br | @andersonmvd
Founder of Gauntlet.io

Cool stuff we'll talk about

- Containers
- Serverless Applications

Who am I?

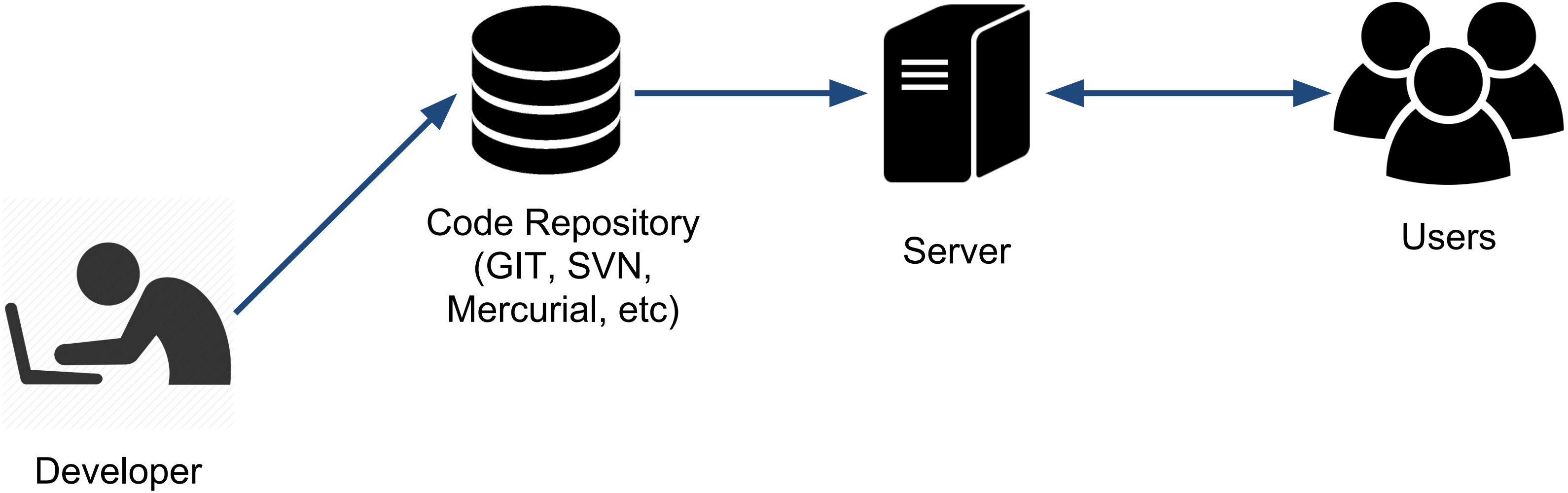
Official (ISC)^2 CSSLP Instructor

Founder of Gauntlet.io

Loves Software {Engineering,Security}

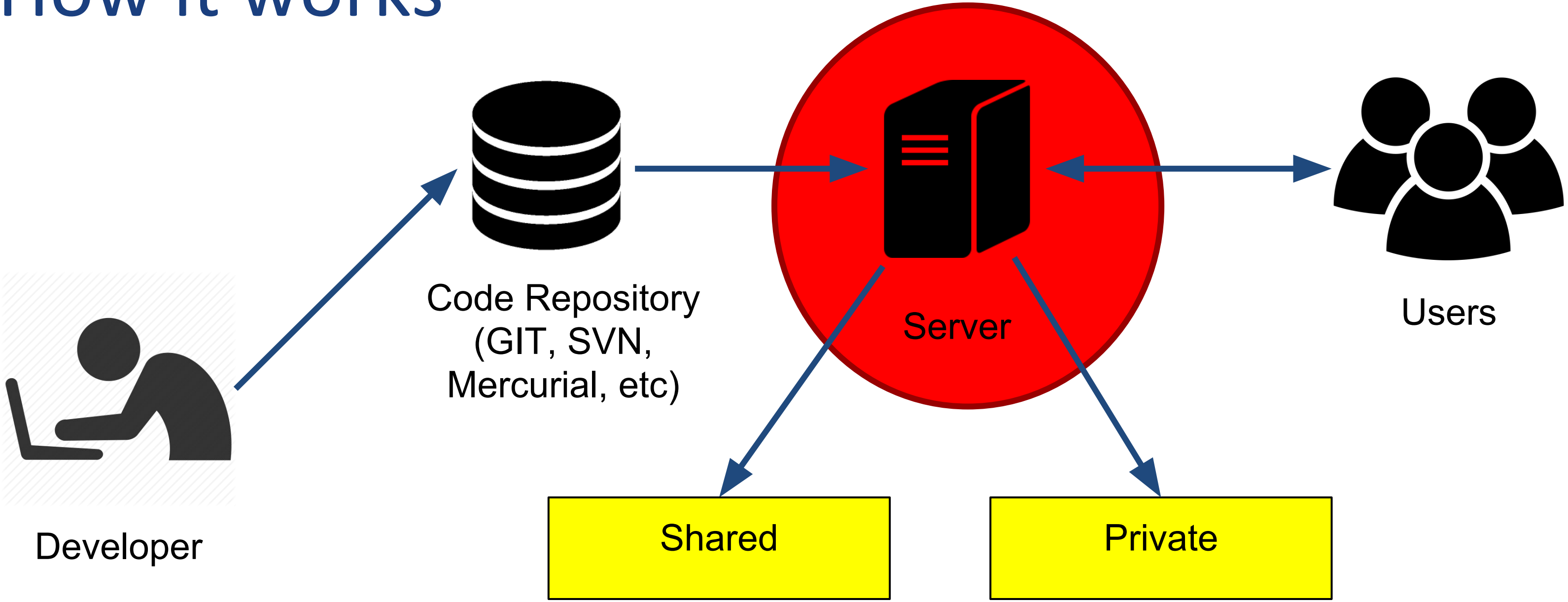
Simple deployment flow

How it works



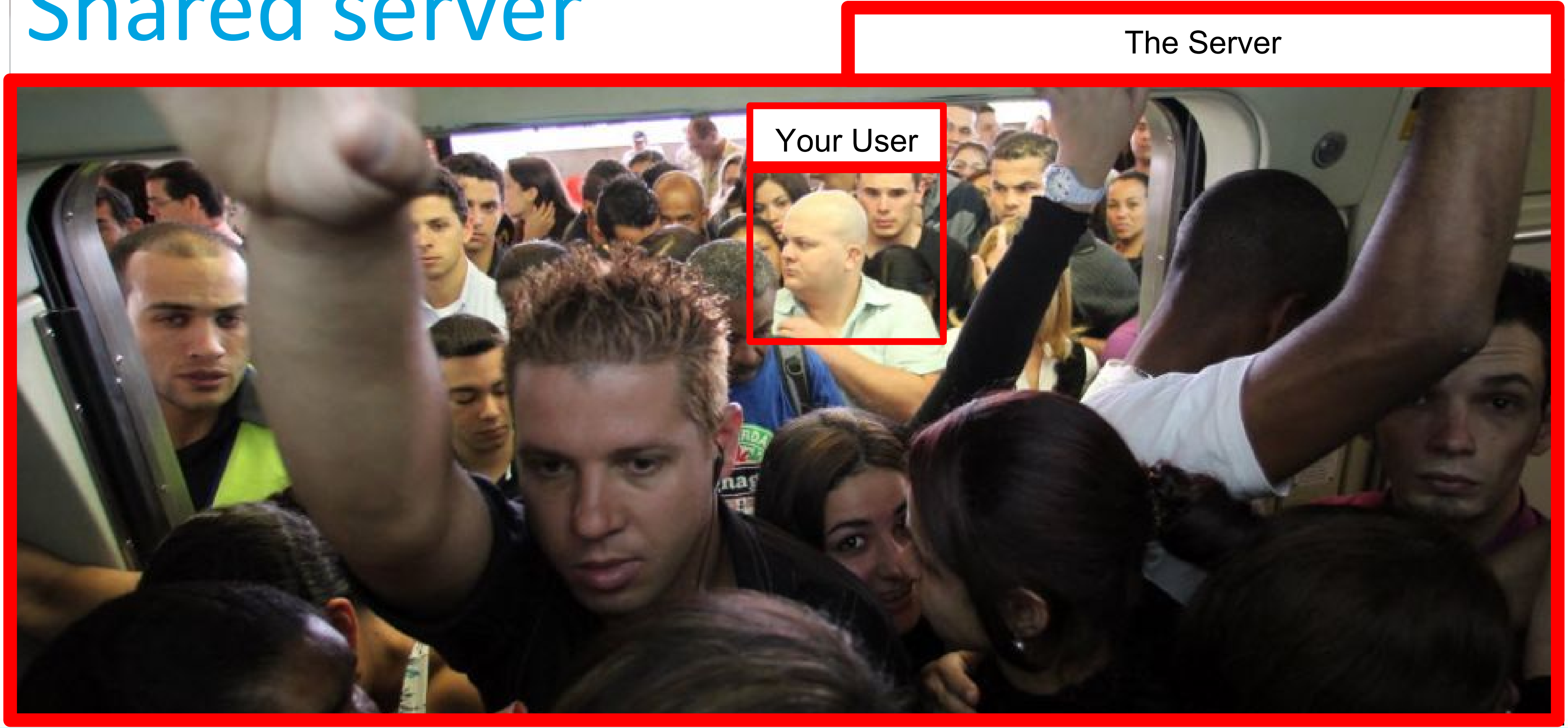
Simple deployment flow

How it works

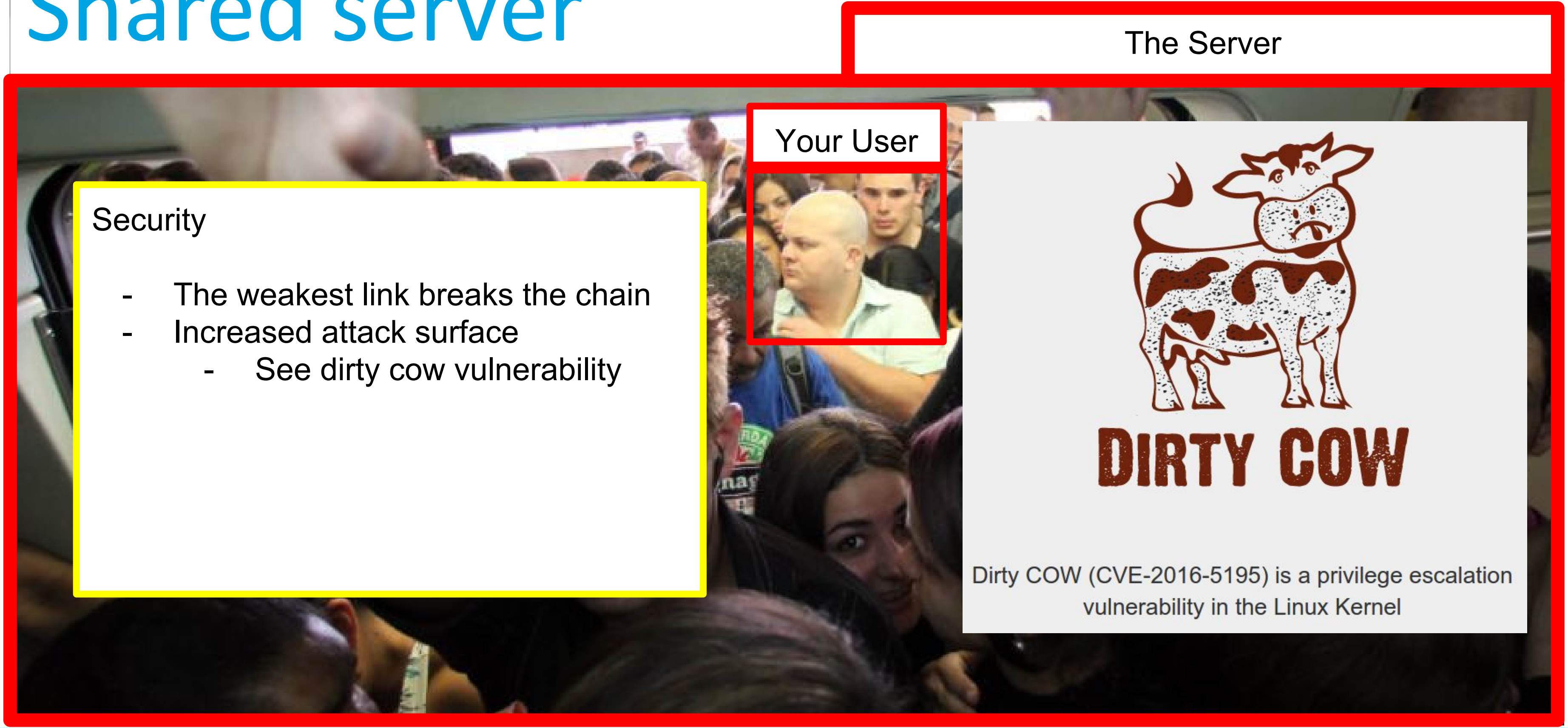


Shared server

Shared server



Shared server



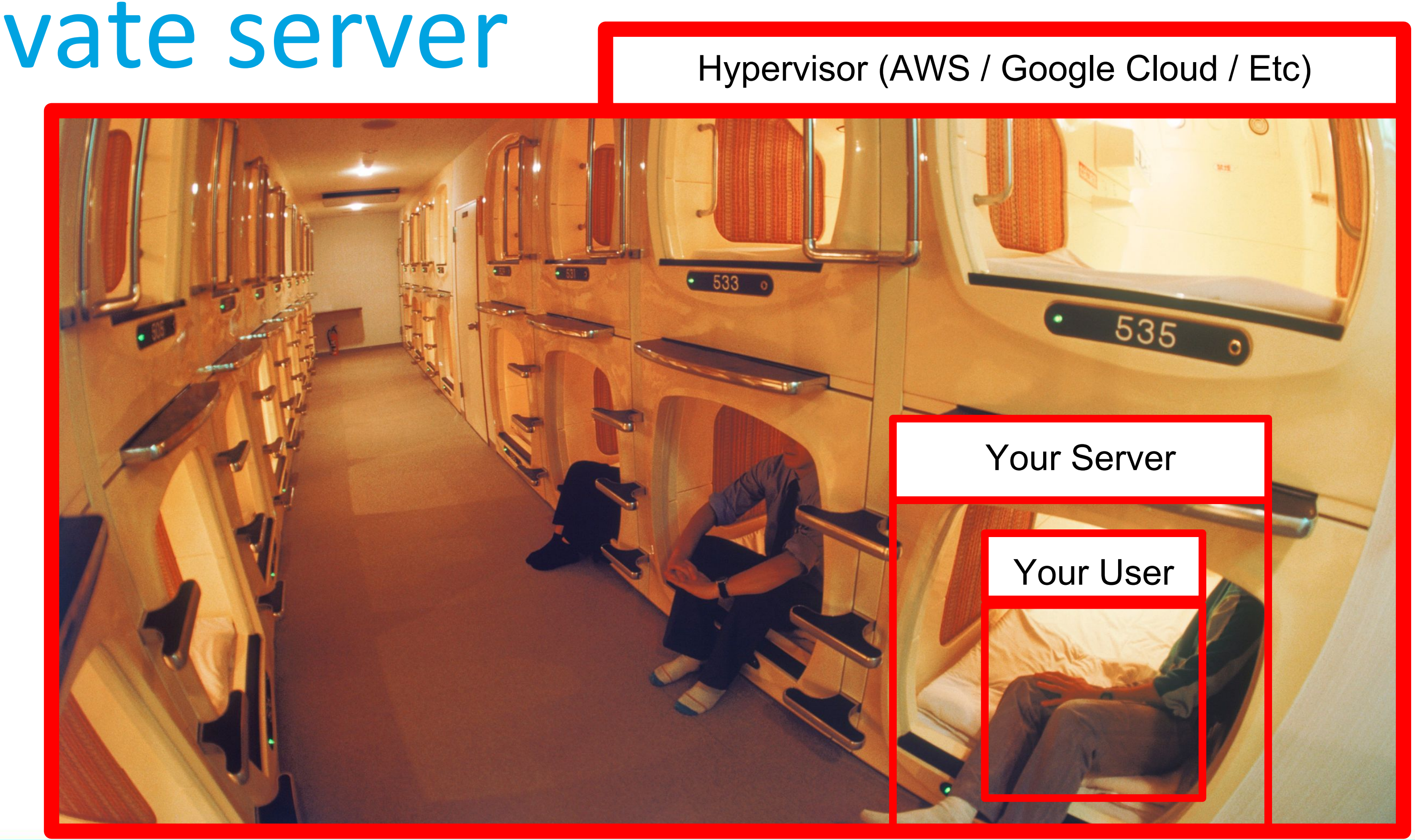
Security

- The weakest link breaks the chain
- Increased attack surface
  - See dirty cow vulnerability

Dirty COW (CVE-2016-5195) is a privilege escalation vulnerability in the Linux Kernel

Private server

Private server

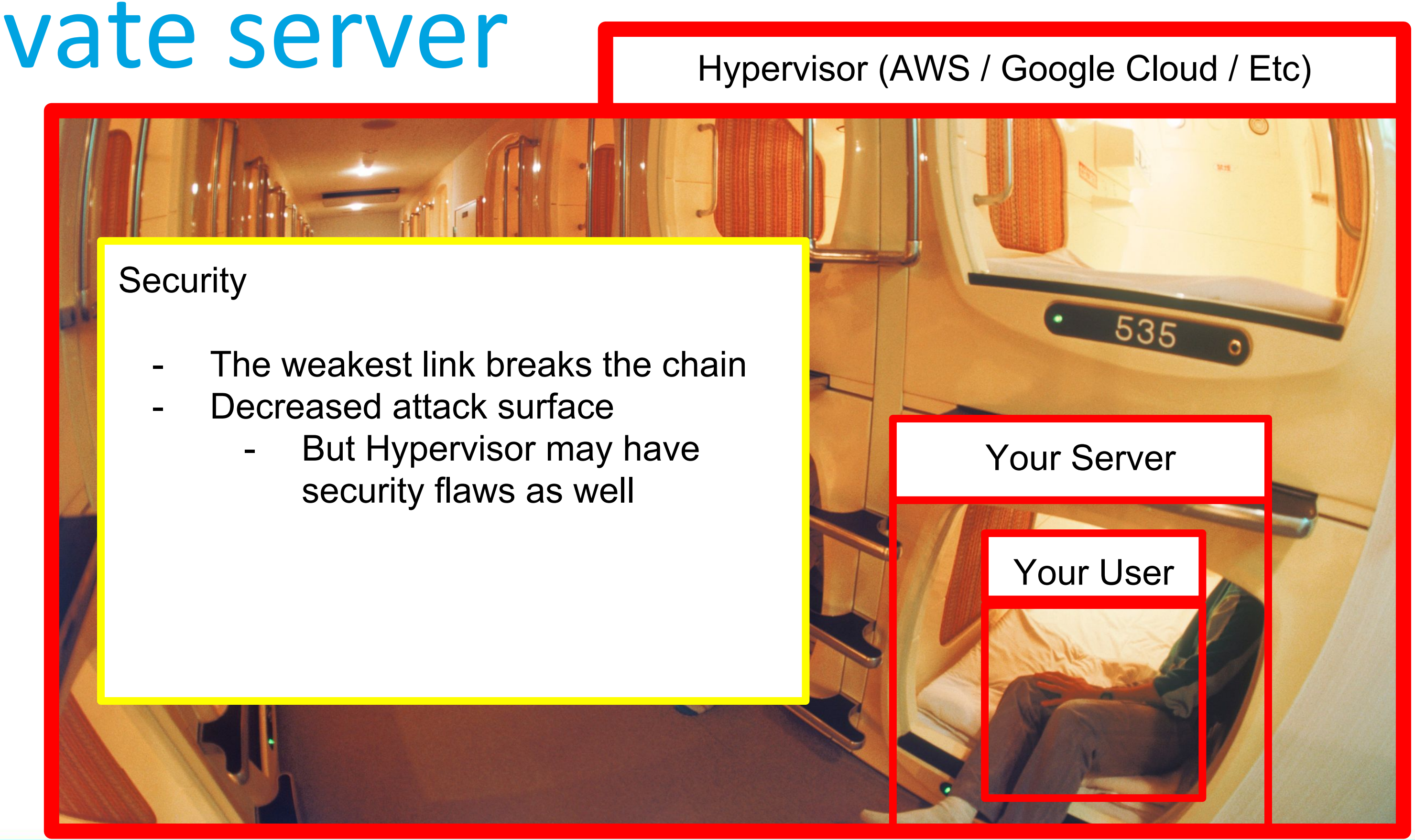


Hypervisor (AWS / Google Cloud / Etc)

Your Server

Your User

Private server



Security

- The weakest link breaks the chain
- Decreased attack surface
  - But Hypervisor may have security flaws as well

Setting up your private rerver

Manually connect
through SSH & Go Nuts

Setting up your private rerver

Security

- It’s Ad hoc, non repeatable
- Requires a different security approach for every server, as they’re likely to be different
- Creates high technical debt

Manually connect through SSH & Go Nuts

Use CM Tools

… OR …

CHEF™
ANSIBLE
puppet

Use Configuration
Management (CM) Tools

Use CM Tools

Security

- Chef & Puppet use agents & servers that need to be secured / tested
- Server recipes make the process repeatable
- Security needs to audit every recipe

HEF™

ANSIBLE

puppet

Use Configuration Management (CM) Tools

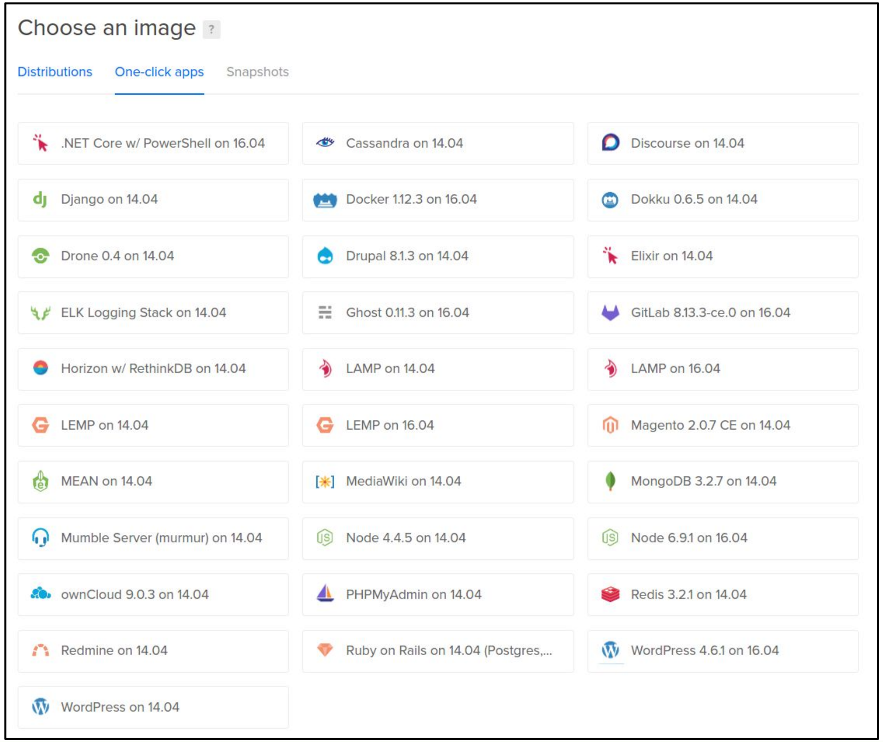
Ansible example

```
---
- hosts: server
  sudo: yes
  sudo_user: root
  tasks:

  - name: install mysql-server
    apt: name=mysql-server state=present
update_cache=yes

  - name: Ensure mysql is running
    service: name=mysql state=started
```

That’s the Infrastructure as Code (IaC) concept.

Use cloud provider server images

… OR …



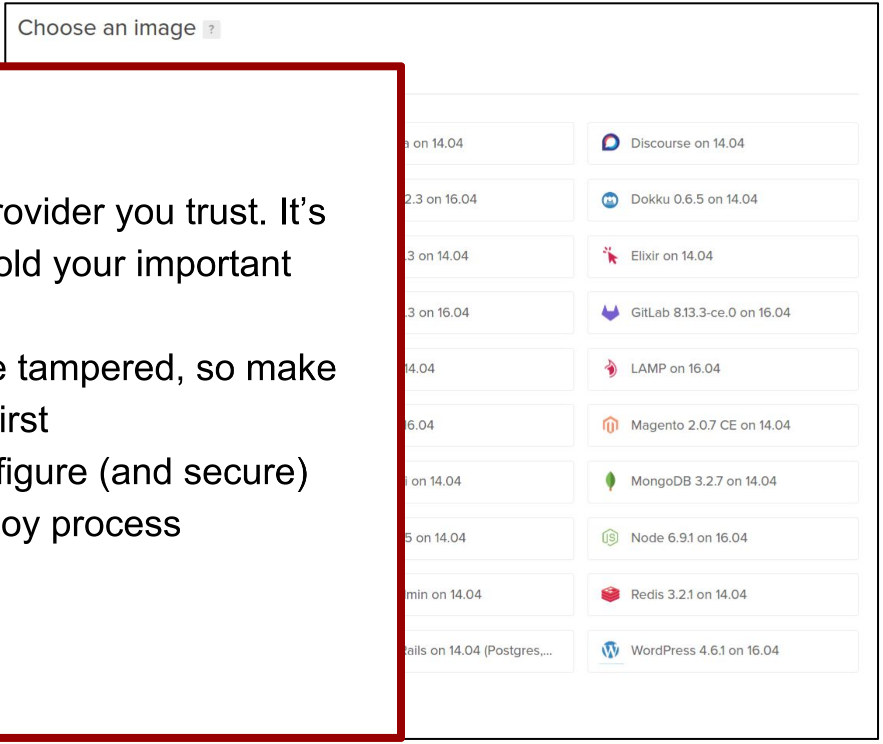
Choose an image ?

.NET Core w/ PowerShell on 16.04	Cassandra on 14.04	Discourse on 14.04
Django on 14.04	Docker 1.12.3 on 16.04	Dokku 0.6.5 on 14.04
Drone 0.4 on 14.04	Drupal 8.1.3 on 14.04	Elixir on 14.04
ELK Logging Stack on 14.04	Ghost 0.11.3 on 16.04	GitLab 8.13.3-ce.0 on 16.04
Horizon w/ RethinkDB on 14.04	LAMP on 14.04	LAMP on 16.04
LEMP on 14.04	LEMP on 16.04	Magento 2.0.7 CE on 14.04
MEAN on 14.04	MediaWiki on 14.04	MongoDB 3.2.7 on 14.04
Mumble Server (murmur) on 14.04	Node 4.4.5 on 14.04	Node 6.9.1 on 16.04
ownCloud 9.0.3 on 14.04	PHPMyAdmin on 14.04	Redis 3.2.1 on 14.04
Redmine on 14.04	Ruby on Rails on 14.04 (Postgres,...	WordPress 4.6.1 on 16.04
WordPress on 14.04		

Use cloud provider server images

Security

- Only go to a cloud provider you trust. It’s like Google, they’ll hold your important assets and data
- Their images may be tampered, so make sure you trust them first
- You still need to configure (and secure) your application deploy process
- Enable 2FA



Choose an image ?

a on 14.04 | Discourse on 14.04
2.3 on 16.04 | Dokku 0.6.5 on 14.04
.3 on 14.04 | Elixir on 14.04
.3 on 16.04 | GitLab 8.13.3-ce.0 on 16.04
14.04 | LAMP on 16.04
6.04 | Magento 2.0.7 CE on 14.04
i on 14.04 | MongoDB 3.2.7 on 14.04
5 on 14.04 | Node 6.9.1 on 16.04
min on 14.04 | Redis 3.2.1 on 14.04
ails on 14.04 (Postgres,... | WordPress 4.6.1 on 16.04

Use a Platform as a Service (PaaS)

… OR …

Platform as a Service (PaaS) -
(Send them your code and let
them manage the servers)

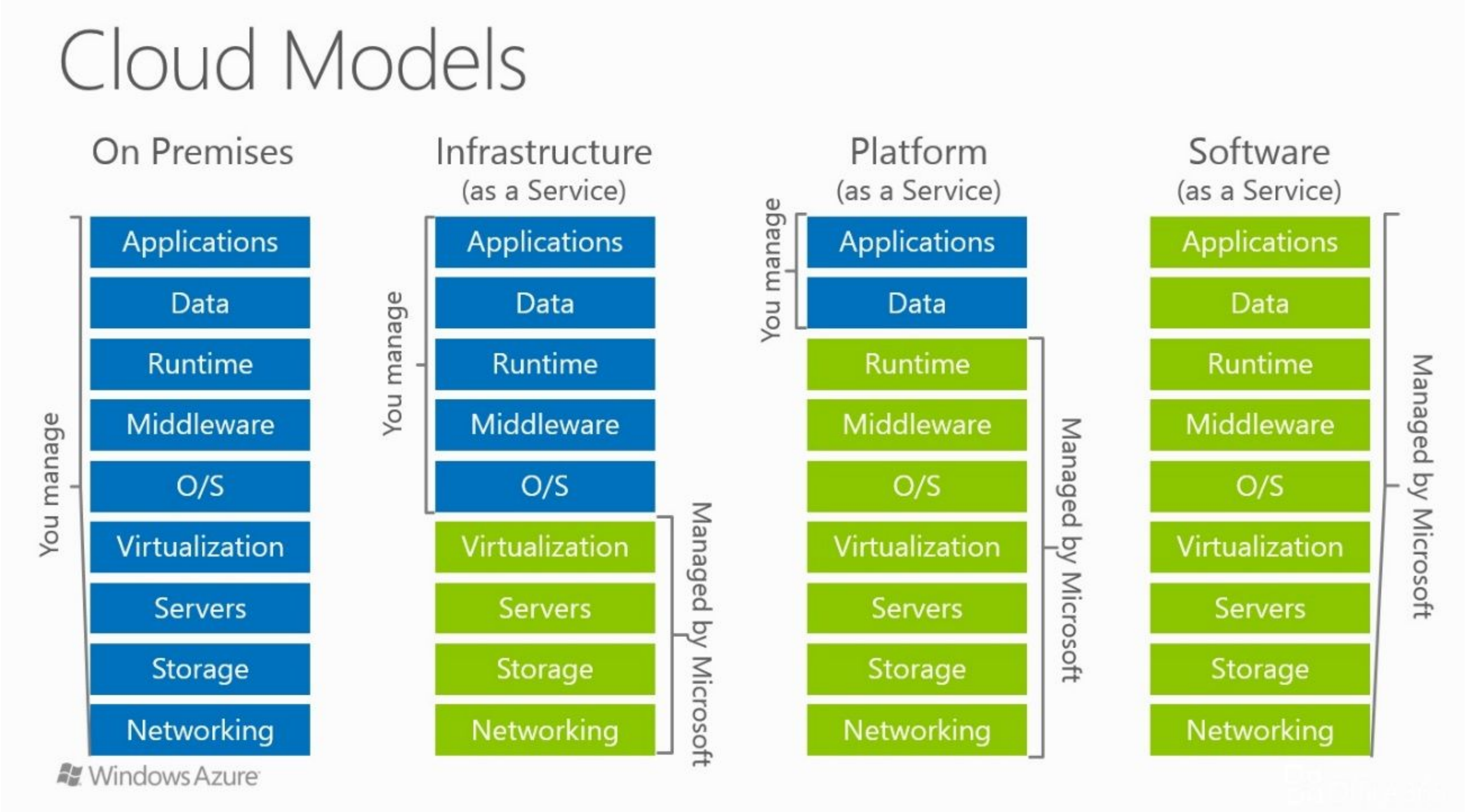
Use a Platform as a Service (PaaS)

Security

- It’s all about trust and Vendor Risk Assessment. You’re as secure as your PaaS provider as long as you don’t mess up with your personal security (leak passwords, use weak passwords, etc)
- Review every server requirements, read everything they offer for security
- Enable 2FA

heroku

n as a Service (PaaS) -
em your code and let
manage the servers)



Cloud Models

On Premises

You manage

- Applications
- Data
- Runtime
- Middleware
- O/S
- Virtualization
- Servers
- Storage
- Networking

Infrastructure (as a Service)

You manage

- Applications
- Data
- Runtime
- Middleware
- O/S

Managed by Microsoft

- Virtualization
- Servers
- Storage
- Networking

Platform (as a Service)

You manage

- Applications
- Data

Managed by Microsoft

- Runtime
- Middleware
- O/S
- Virtualization
- Servers
- Storage
- Networking

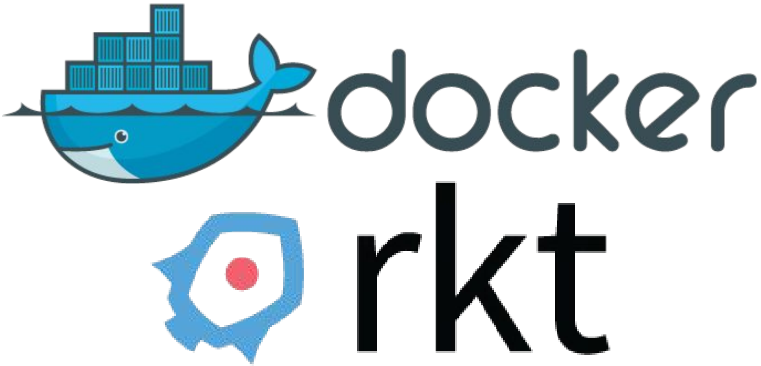
Software (as a Service)

Managed by Microsoft

- Applications
- Data
- Runtime
- Middleware
- O/S
- Virtualization
- Servers
- Storage
- Networking

Or go with Containers

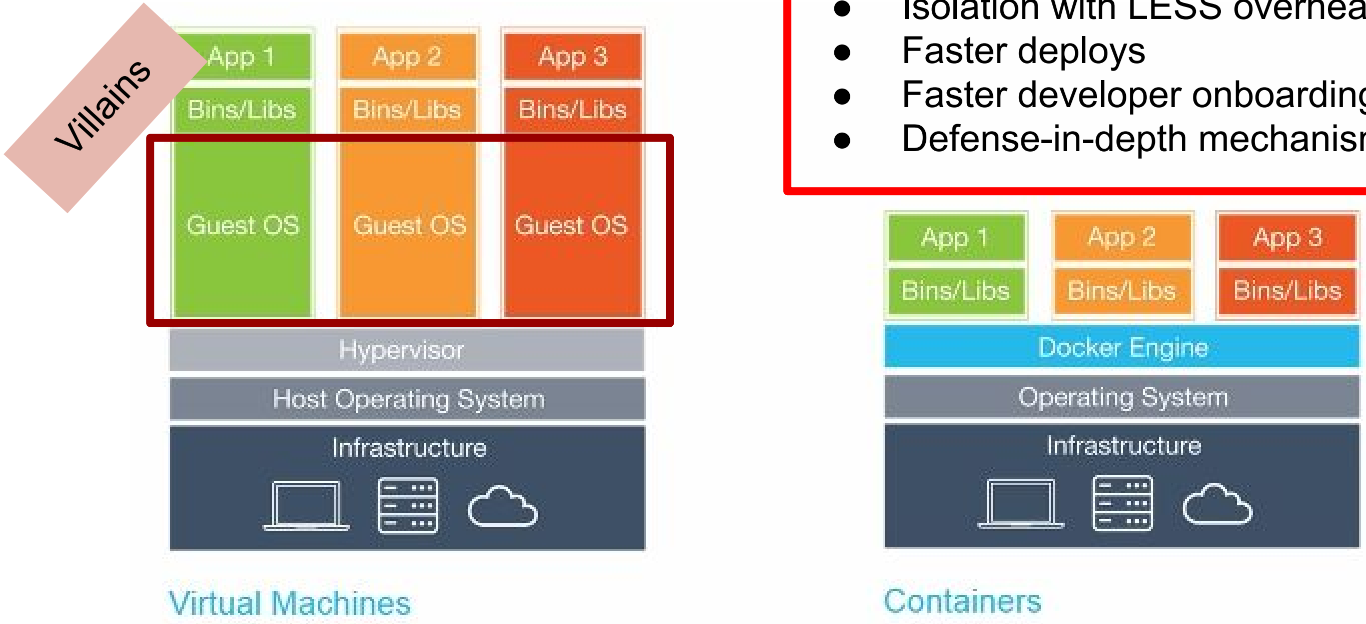
… OR …



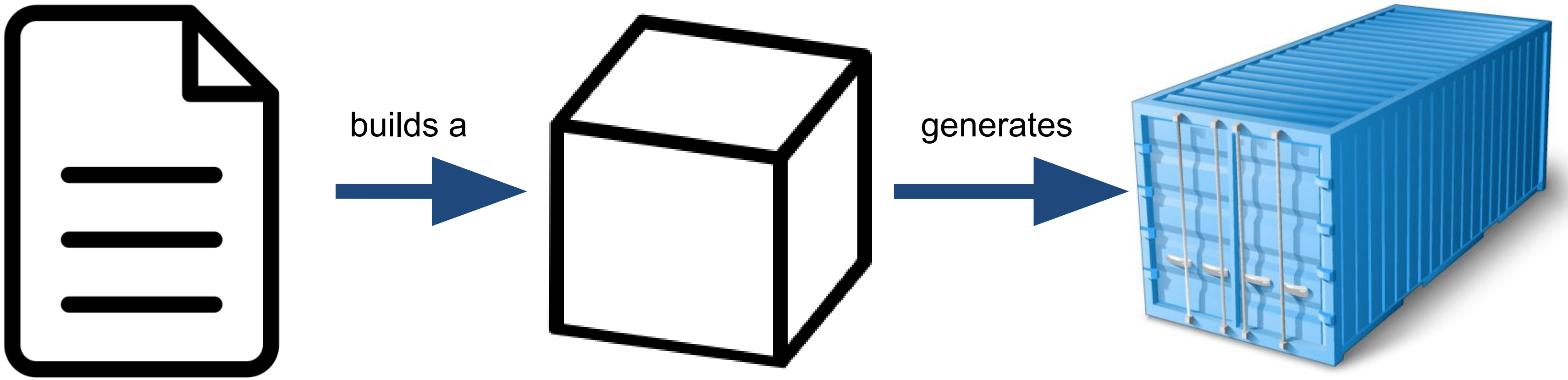
Containers

It’s like Debian’s *chroot* on steroids!

- Isolation with LESS overhead
- Faster deploys
- Faster developer onboarding
- Defense-in-depth mechanism



How a container is born



builds a

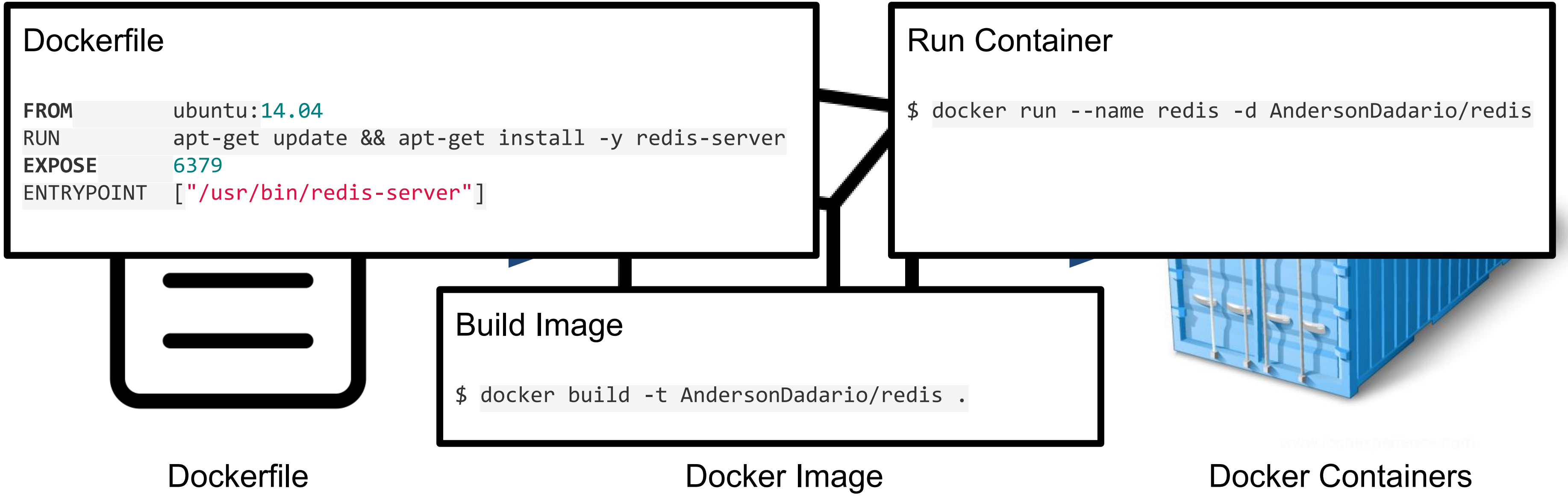
generates

Dockerfile

Docker Image

Docker Containers

How a container is born



Dockerfile

```
FROM        ubuntu:14.04
RUN         apt-get update && apt-get install -y redis-server
EXPOSE      6379
ENTRYPOINT  ["/usr/bin/redis-server"]
```

Build Image

```
$ docker build -t AndersonDadario/redis .
```

Run Container

```
$ docker run --name redis -d AndersonDadario/redis
```

What is in a container

Docker Containers

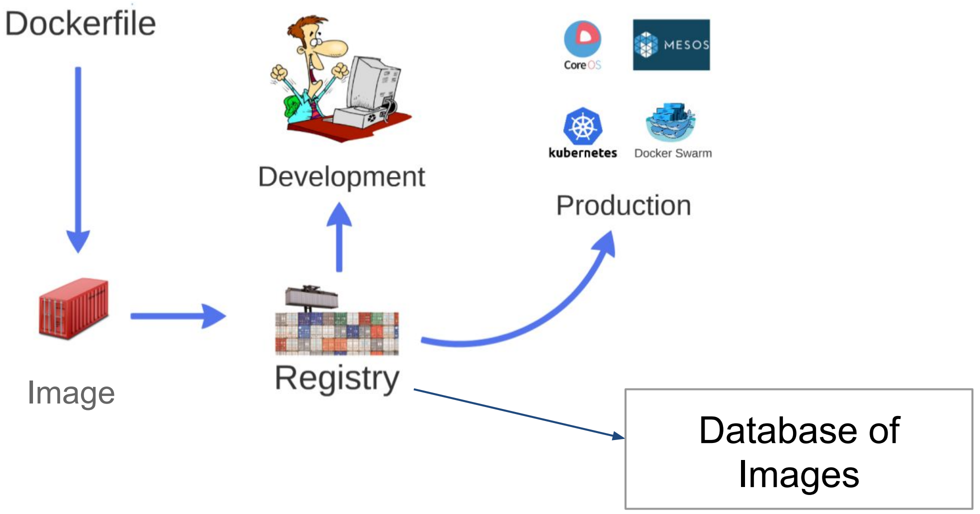
A container is a server with its own:

- IP Address
- Network
- File system
- Etc

Two major ways to use containers

- Microservices
  - Thin Containers
  - One process per Container
- Monolithic
  - Fat Containers
  - Multiple processes per Container
  - Looks like a Virtual Machine (VM)

Typical flow with Containers



Database of Images

What can you do about security?

- Keep Docker Up-to-date
- Harden Docker Daemon
- Don’t run containers as root
  - And don’t let users easily become root
    - Remove SUID flags / SUDO
- Don’t put sensitive files in your container
- Audit Dockerfiles

What can you do about security?

- Reduce Container Capabilities
- Avoid images without Dockerfile
  - As you can’t check what’s inside so easily, e.g., look for backdoors
- Only install an image if you trust
  - If possible verify
- Limit Container Usage (Memory/Etc) Anti-DoS

What can you do about security?

- Run security scanners on images
  - Docker Security Scanning (Paid)
  - CoreOS Clair (Open Source)
- Consider using a Container Security Platform
  - Evaluated by Gartner:
    - Aqua Security, CloudPassage, Docker, Magnetic.io, Twistlock, Weaveworks

Enough about containers

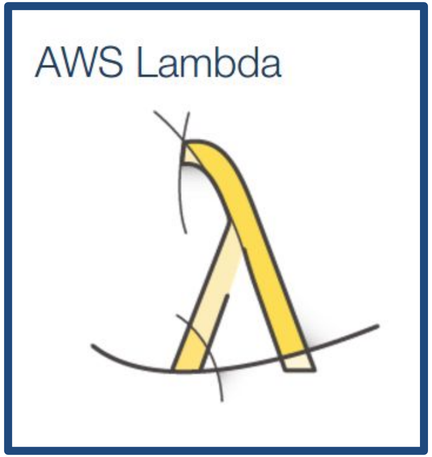
Time is finite after all

.

.

Show me the Serverless Stuff

Serverless Applications

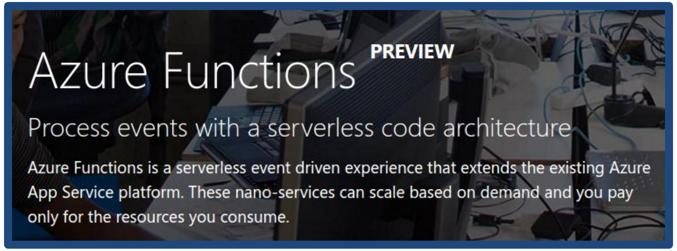


CLOUD FUNCTIONS ALPHA

A serverless platform for building event-based microservices

IBM Bluemix OpenWhisk

Execute code on demand in a highly scalable serverless environment



Azure Functions PREVIEW

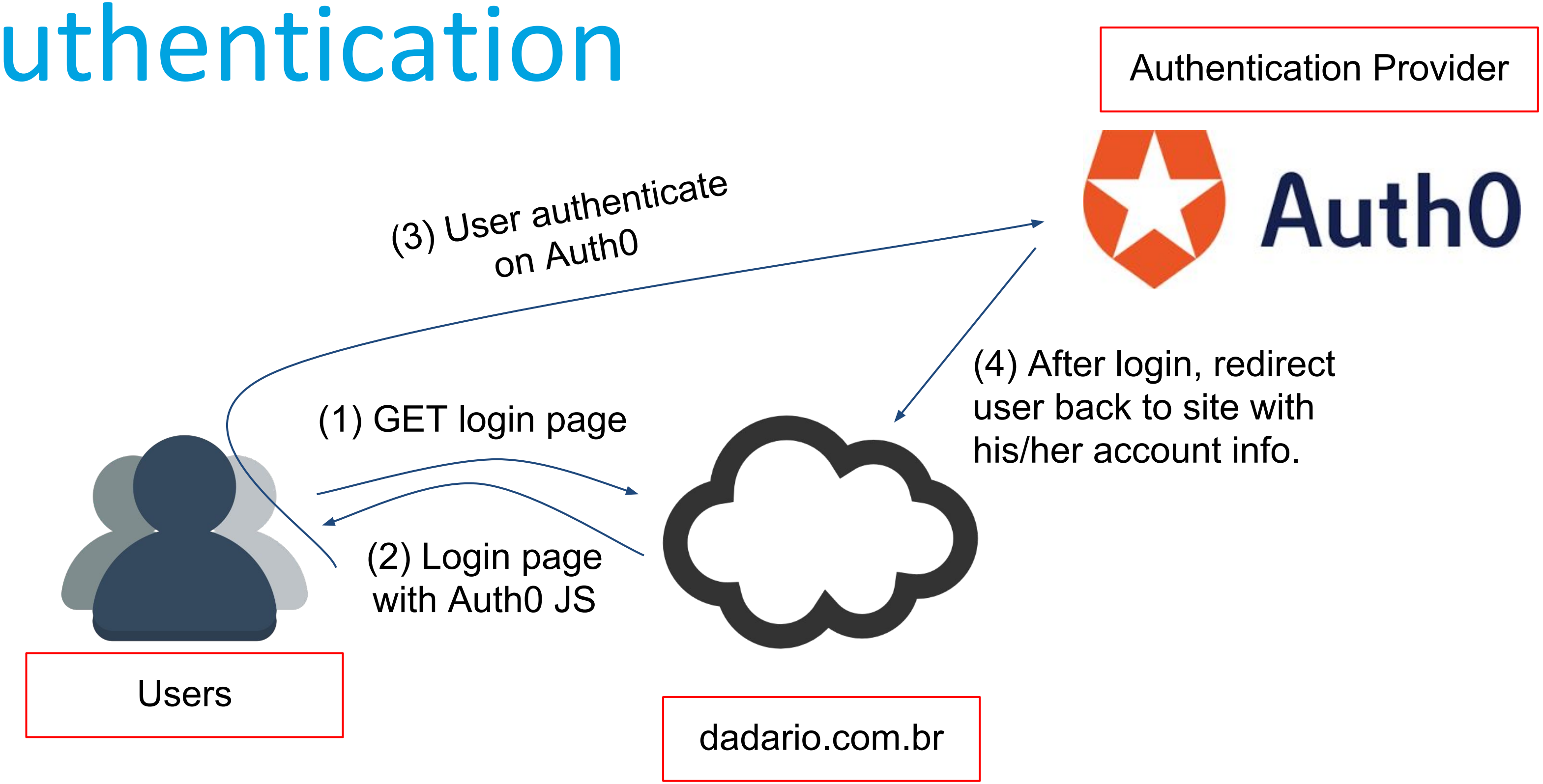
Process events with a serverless code architecture

Azure Functions is a serverless event driven experience that extends the existing Azure App Service platform. These nano-services can scale based on demand and you pay only for the resources you consume.

The Challenge

- Case Study: Dadario’s Learning Platform
- Build a Serverless Application with:
    - Authentication
    - Authorization
    - Payment Processing

Authentication



Authentication Provider

(3) User authenticate on Auth0

(4) After login, redirect user back to site with his/her account info.

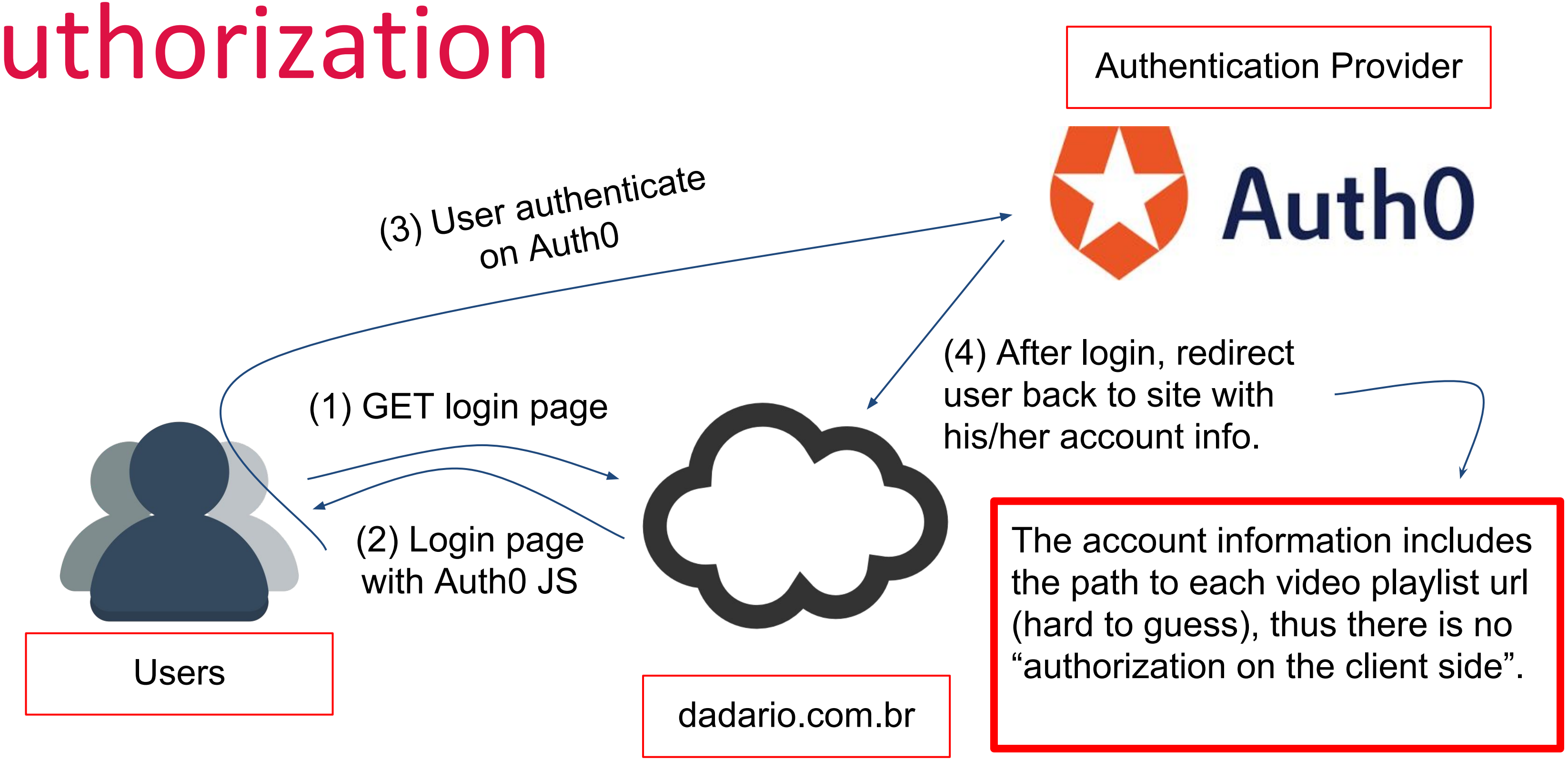
(1) GET login page

(2) Login page with Auth0 JS

Users

dadario.com.br

Authorization



Authentication Provider

Auth0

(3) User authenticate on Auth0

(1) GET login page

(2) Login page with Auth0 JS

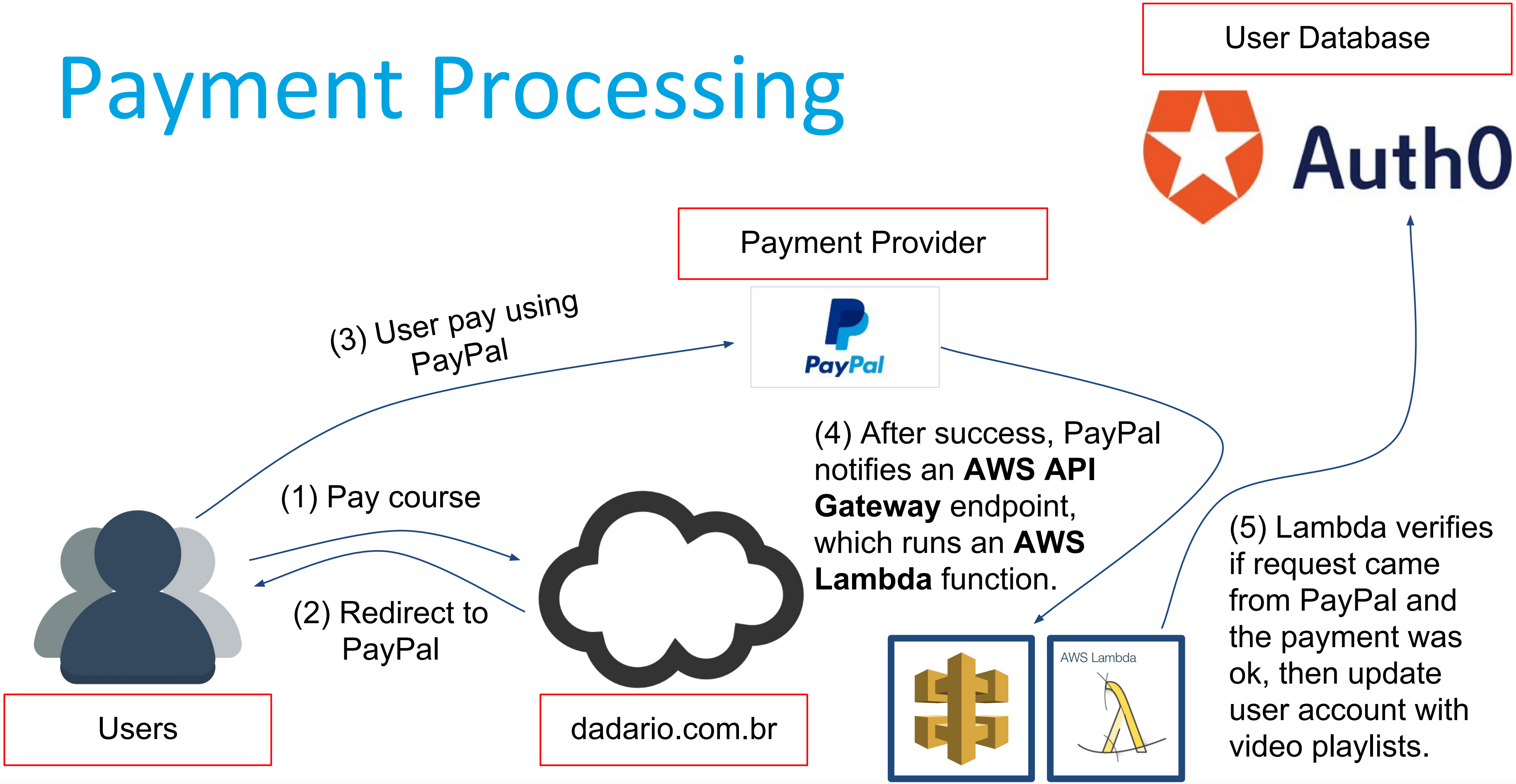
(4) After login, redirect user back to site with his/her account info.

Users

dadario.com.br

The account information includes the path to each video playlist url (hard to guess), thus there is no “authorization on the client side”.

Payment Processing



User Database

Auth0

Payment Provider

PayPal

(3) User pay using PayPal

(1) Pay course

(2) Redirect to PayPal

Users

dadario.com.br

(4) After success, PayPal notifies an **AWS API Gateway** endpoint, which runs an **AWS Lambda** function.

(5) Lambda verifies if request came from PayPal and the payment was ok, then update user account with video playlists.

But there are Limitations

- Few languages available
- Can’t include all libraries
- Need automation to organize multiple functions

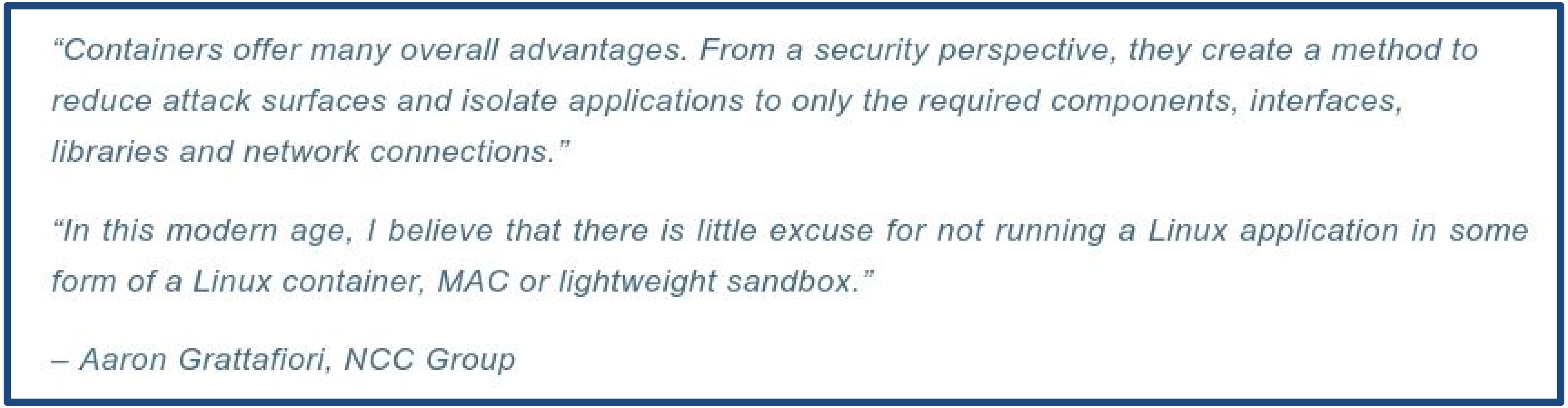
For some needs it’s a great solution.

And how does Security fits in?

- Vendor Risk Assessment
- Security Configuration
- Code Review
- Fuzzing
- Note: Pentest requires Vendor approval

Takeaways

- Development is changing, thus Security must catch up
- Your applications are safer in Containers
- Serverless Applications are real and growing



"Containers offer many overall advantages. From a security perspective, they create a method to reduce attack surfaces and isolate applications to only the required components, interfaces, libraries and network connections."

"In this modern age, I believe that there is little excuse for not running a Linux application in some form of a Linux container, MAC or lightweight sandbox."

– Aaron Grattafiori, NCC Group

Resources, References & Tools

Docker

https://docs.docker.com/docker-cloud/builds/image-scan/

https://github.com/coreos/clair

https://github.com/CenturyLinkLabs/dockerfile-from-image

https://www.sumologic.com/blog-devops/securing-docker-containers/

https://www.ctl.io/developers/blog/post/tutorial-protecting-sensitive-info-docker

Serverless

https://github.com/apex/apex

https://github.com/serverless/serverless

Thank you

Anderson Dadario

dadario.com.br | @andersonmvd

Founder of Gauntlet.io

Slides are available for free on https://dadario.com.br/slides