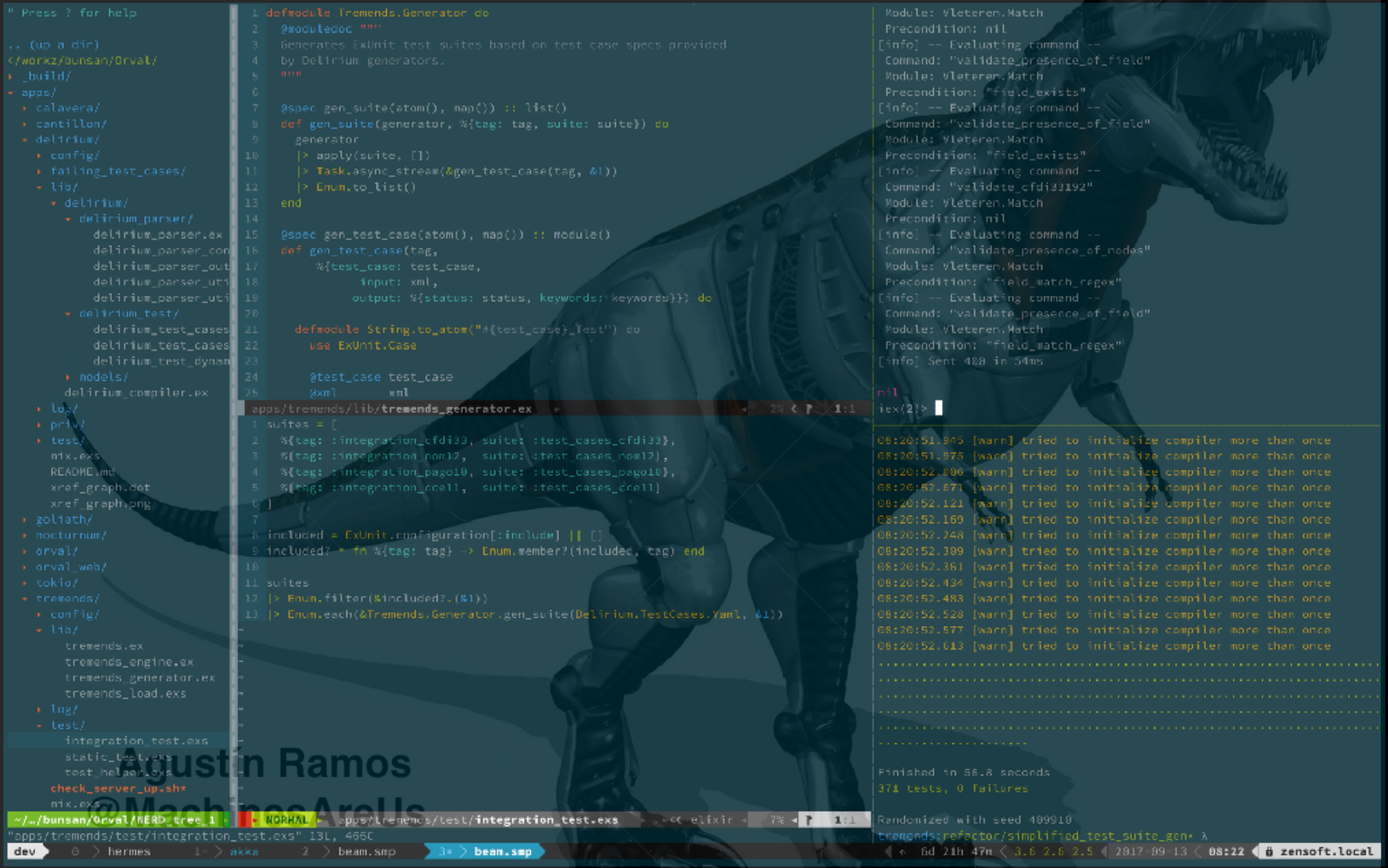
Kubernetes Workshop

Agustín Ramos
@MachinesAreUs

Noviembre 2017

Agustín Ramos
@MachinesAreUs



```
" Press ? for help

.. (up a dir)
</workz/bunsan/Orval/
▸ _build/
▾ apps/
  ▸ calavera/
  ▸ cantillon/
  ▾ delirium/
    ▸ config/
    ▸ failing_test_cases/
    ▾ lib/
      ▾ delirium/
        ▾ delirium_parser/
            delirium_parser.ex
            delirium_parser_con
            delirium_parser_out
            delirium_parser_uti
            delirium_parser_uti
        ▾ delirium_test/
            delirium_test_cases
            delirium_test_cases
            delirium_test_dynam
        ▸ models/
        delirium_compiler.ex
    ▸ log/
    ▸ priv/
    ▸ test/
      mix.exs
      README.md
      xref_graph.dot
      xref_graph.png
  ▸ goliath/
  ▸ nocturnum/
  ▸ orval/
  ▸ orval_web/
  ▸ tokio/
  ▾ tremends/
    ▸ config/
    ▾ lib/
        tremends.ex
        tremends_engine.ex
        tremends_generator.ex
        tremends_load.exs
    ▸ log/
    ▾ test/
        integration_test.exs
        static_test.exs
        test_helper.exs
      check_server_up.sh*
      mix.exs
~/…/bunsan/Orval/NERD_tree_1
```

```elixir
defmodule Tremends.Generator do
  @moduledoc """
  Generates ExUnit test suites based on test case specs provided
  by Delirium generators.
  """

  @spec gen_suite(atom(), map()) :: list()
  def gen_suite(generator, %{tag: tag, suite: suite}) do
    generator
    |> apply(suite, [])
    |> Task.async_stream(&gen_test_case(tag, &1))
    |> Enum.to_list()
  end

  @spec gen_test_case(atom(), map()) :: module()
  def gen_test_case(tag,
        %{test_case: test_case,
               input: xml,
              output: %{status: status, keywords: keywords}}) do

    defmodule String.to_atom("#{test_case}_Test") do
      use ExUnit.Case

      @test_case test_case
      @xml        xml
```

```
apps/tremends/lib/tremends_generator.ex        2% ⁋  1:1
```

```elixir
suites = [
  %{tag: :integration_cfdi33, suite: :test_cases_cfdi33},
  %{tag: :integration_nom12,  suite: :test_cases_nom12},
  %{tag: :integration_pago10, suite: :test_cases_pago10},
  %{tag: :integration_ccell,  suite: :test_cases_ccell}
]

included = ExUnit.configuration[:include] || []
included? = fn %{tag: tag} -> Enum.member?(included, tag) end

suites
|> Enum.filter(&included?.(&1))
|> Enum.each(&Tremends.Generator.gen_suite(Delirium.TestCases.Yaml, &1))
```

```
NORMAL  apps/tremends/test/integration_test.exs   << elixir   7%  ⁋  1:1
"apps/tremends/test/integration_test.exs" 13L, 466C
```

```
 Module: Vleteren.Match
 Precondition: nil
[info] -- Evaluating command --
 Command: "validate_presence_of_field"
 Module: Vleteren.Match
 Precondition: "field_exists"
[info] -- Evaluating command --
 Command: "validate_presence_of_field"
 Module: Vleteren.Match
 Precondition: "field_exists"
[info] -- Evaluating command --
 Command: "validate_cfdi33192"
 Module: Vleteren.Match
 Precondition: nil
[info] -- Evaluating command --
 Command: "validate_presence_of_nodes"
 Module: Vleteren.Match
 Precondition: "field_match_regex"
[info] -- Evaluating command --
 Command: "validate_presence_of_field"
 Module: Vleteren.Match
 Precondition: "field_match_regex"
[info] Sent 400 in 54ms

nil
iex(2)>
```

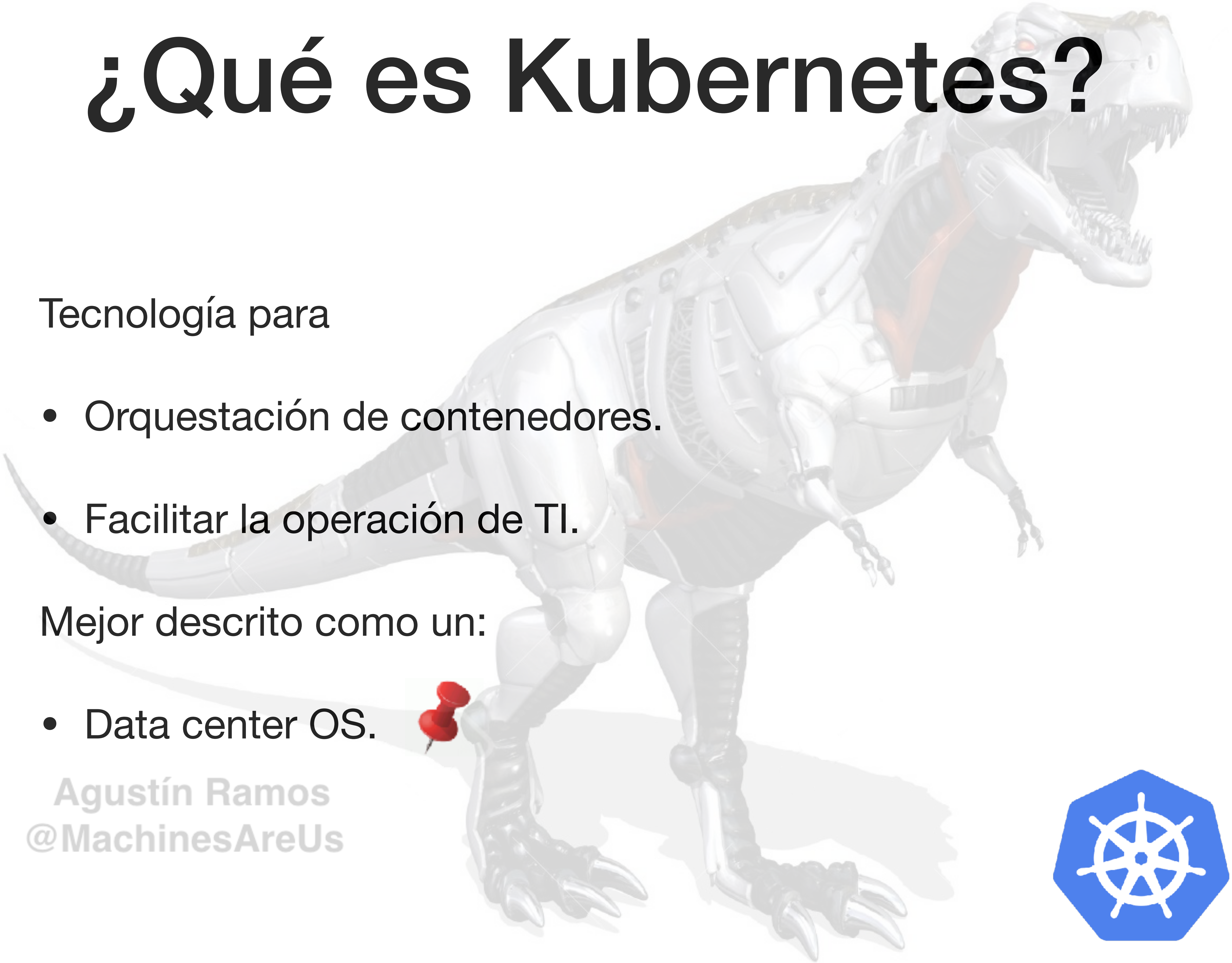
```
08:20:51.945 [warn] tried to initialize compiler more than once
08:20:51.975 [warn] tried to initialize compiler more than once
08:20:52.006 [warn] tried to initialize compiler more than once
08:20:52.071 [warn] tried to initialize compiler more than once
08:20:52.121 [warn] tried to initialize compiler more than once
08:20:52.169 [warn] tried to initialize compiler more than once
08:20:52.248 [warn] tried to initialize compiler more than once
08:20:52.309 [warn] tried to initialize compiler more than once
08:20:52.361 [warn] tried to initialize compiler more than once
08:20:52.434 [warn] tried to initialize compiler more than once
08:20:52.483 [warn] tried to initialize compiler more than once
08:20:52.528 [warn] tried to initialize compiler more than once
08:20:52.577 [warn] tried to initialize compiler more than once
08:20:52.613 [warn] tried to initialize compiler more than once
...........................................................................
...........................................................................
...........................................................................
...........................................................................
...........................................................................
.....................

Finished in 56.8 seconds
371 tests, 0 failures

Randomized with seed 409910
tremends:refactor/simplified_test_suite_gen• λ
```

```
dev   0 > hermes   1- > akka   2 > beam.smp   3* > beam.smp        6d 21h 47m < 3.6 2.6 2.5 < 2017-09-13 < 08:22 < 🔒 zensoft.local
```

Agustín Ramos
@MachinesAreUs

Agustín Ramos
@MachinesAreUs

¿Qué es Kubernetes?



Tecnología para

- Orquestación de contenedores.
- Facilitar la operación de TI.

Mejor descrito como un:

- Data center OS.

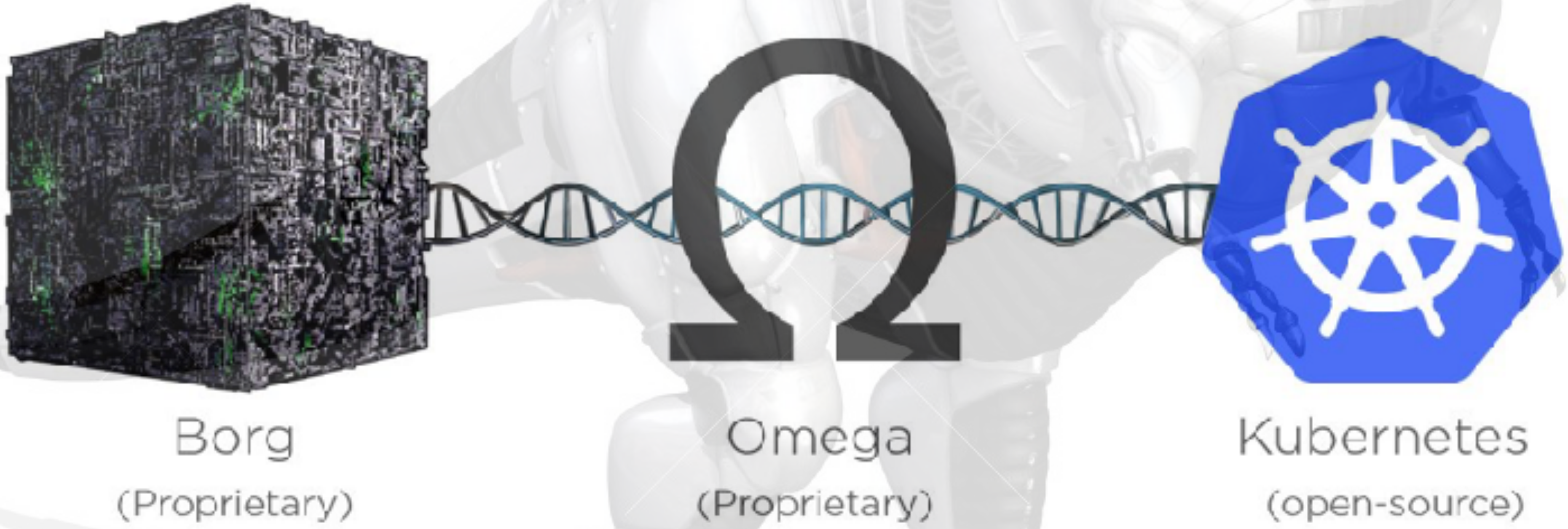
¿De dónde salió?

Google

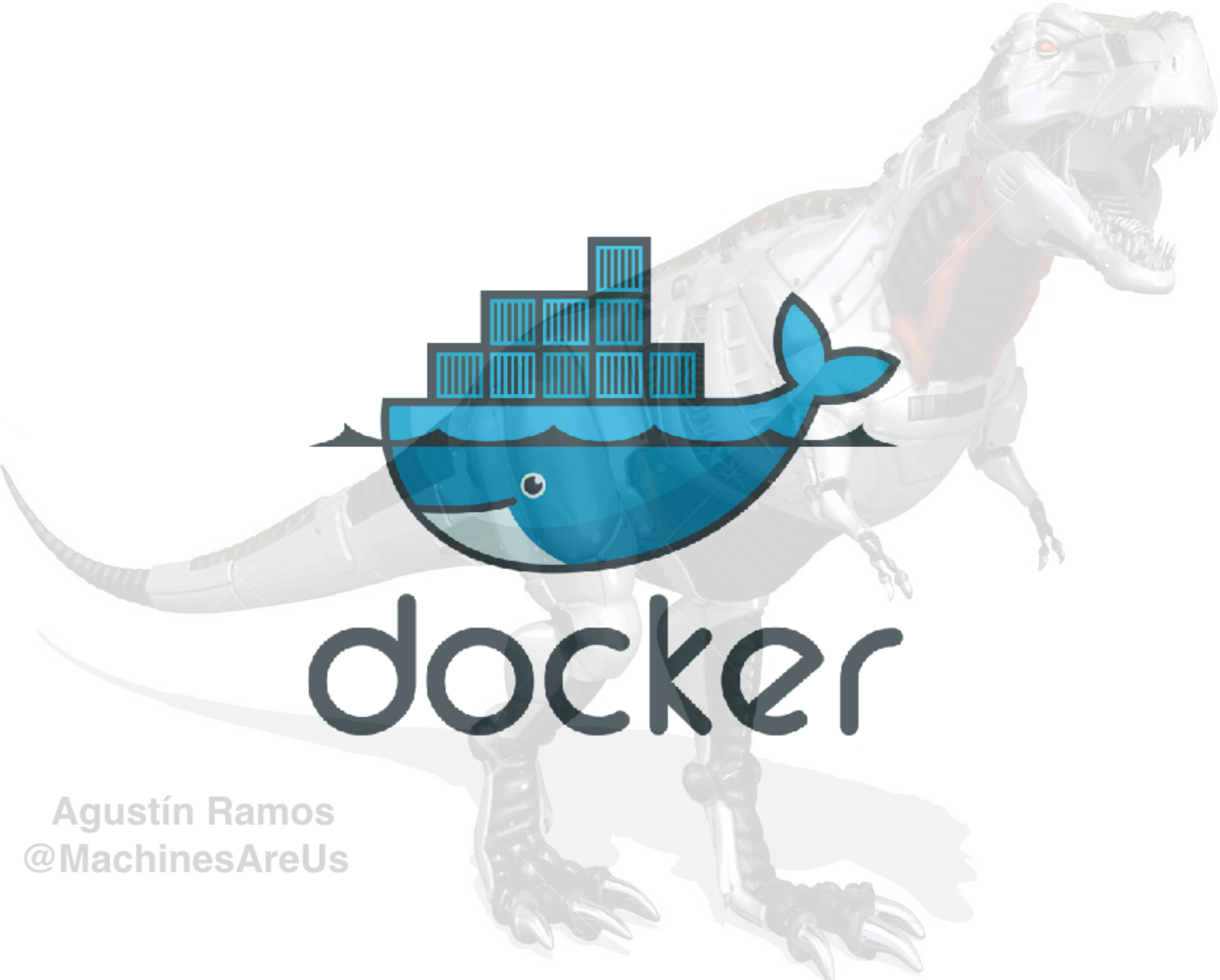
- Open source en verano del 2014.
  - Donado a CLOUD NATIVE COMPUTING FOUNDATION
- Google ha estado ejecutando sus sistemas sobre tecnología de contenedores por 15+ años.

¿De dónde salió?

Pero comparte ADN con otros dos sistemas propietarios.

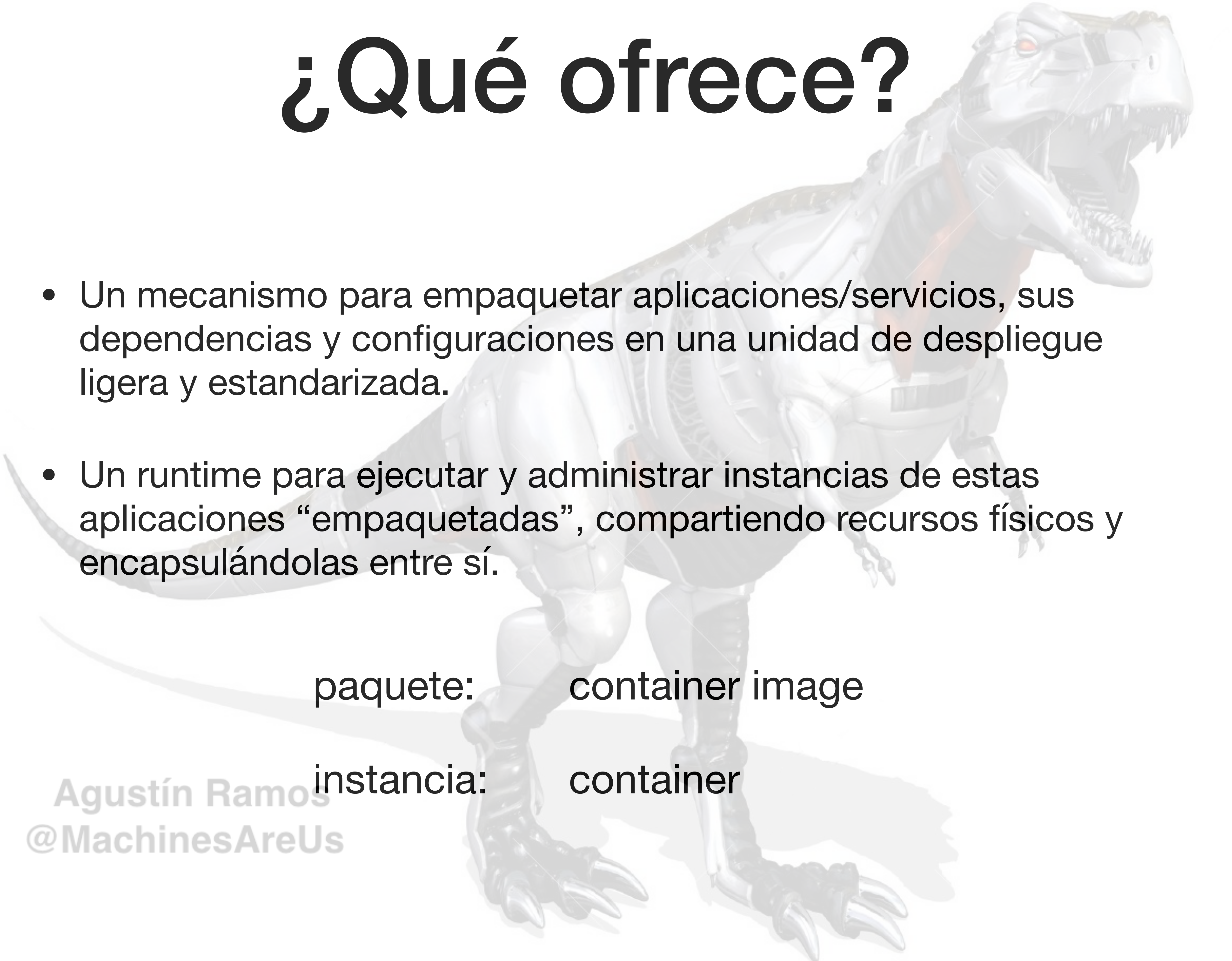


http://j.mp/k8history



Agustín Ramos
@MachinesAreUs

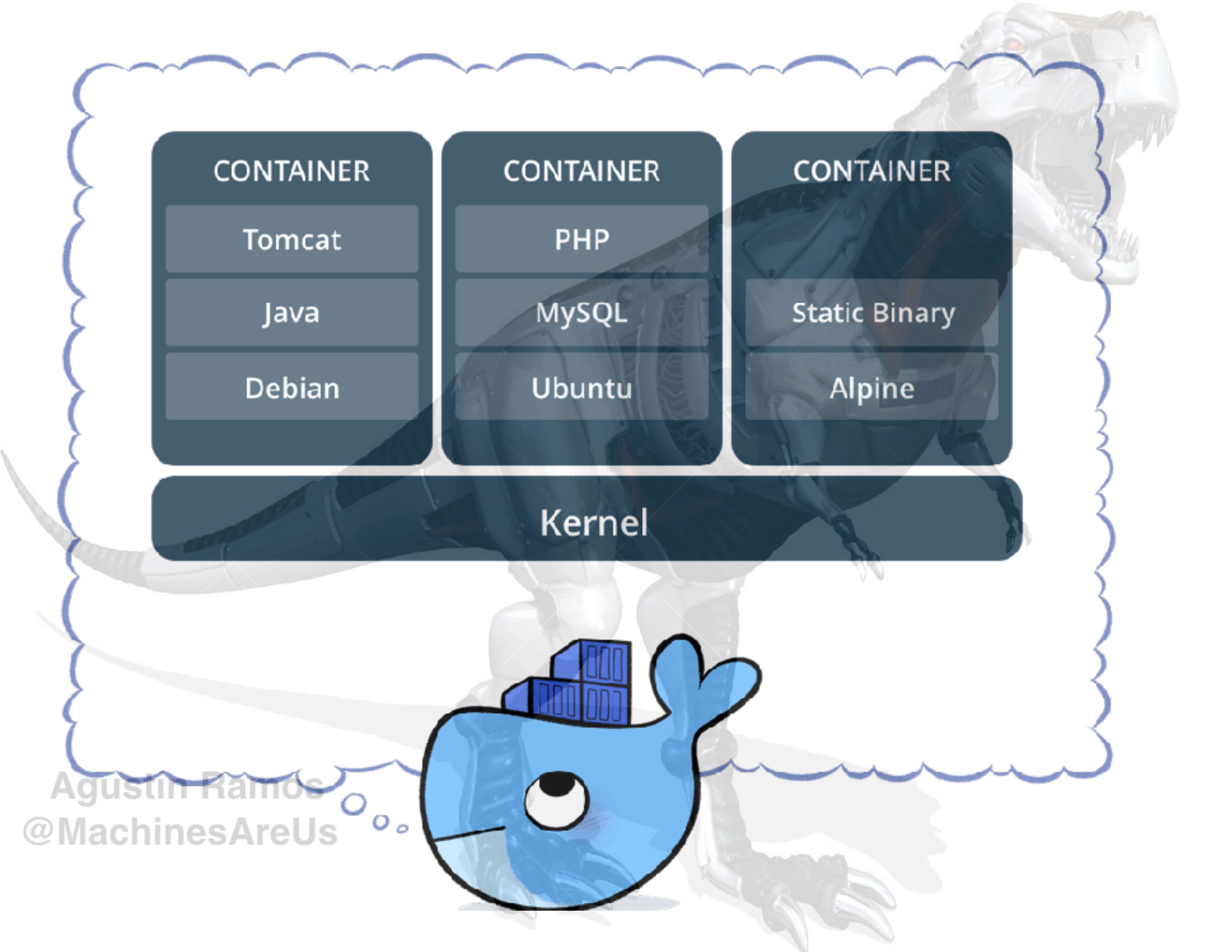
¿Qué ofrece?



- Un mecanismo para empaquetar aplicaciones/servicios, sus dependencias y configuraciones en una unidad de despliegue ligera y estandarizada.

- Un runtime para ejecutar y administrar instancias de estas aplicaciones “empaquetadas”, compartiendo recursos físicos y encapsulándolas entre sí.

paquete:	container image
instancia:	container



CONTAINER

Tomcat

Java

Debian

CONTAINER

PHP

MySQL

Ubuntu

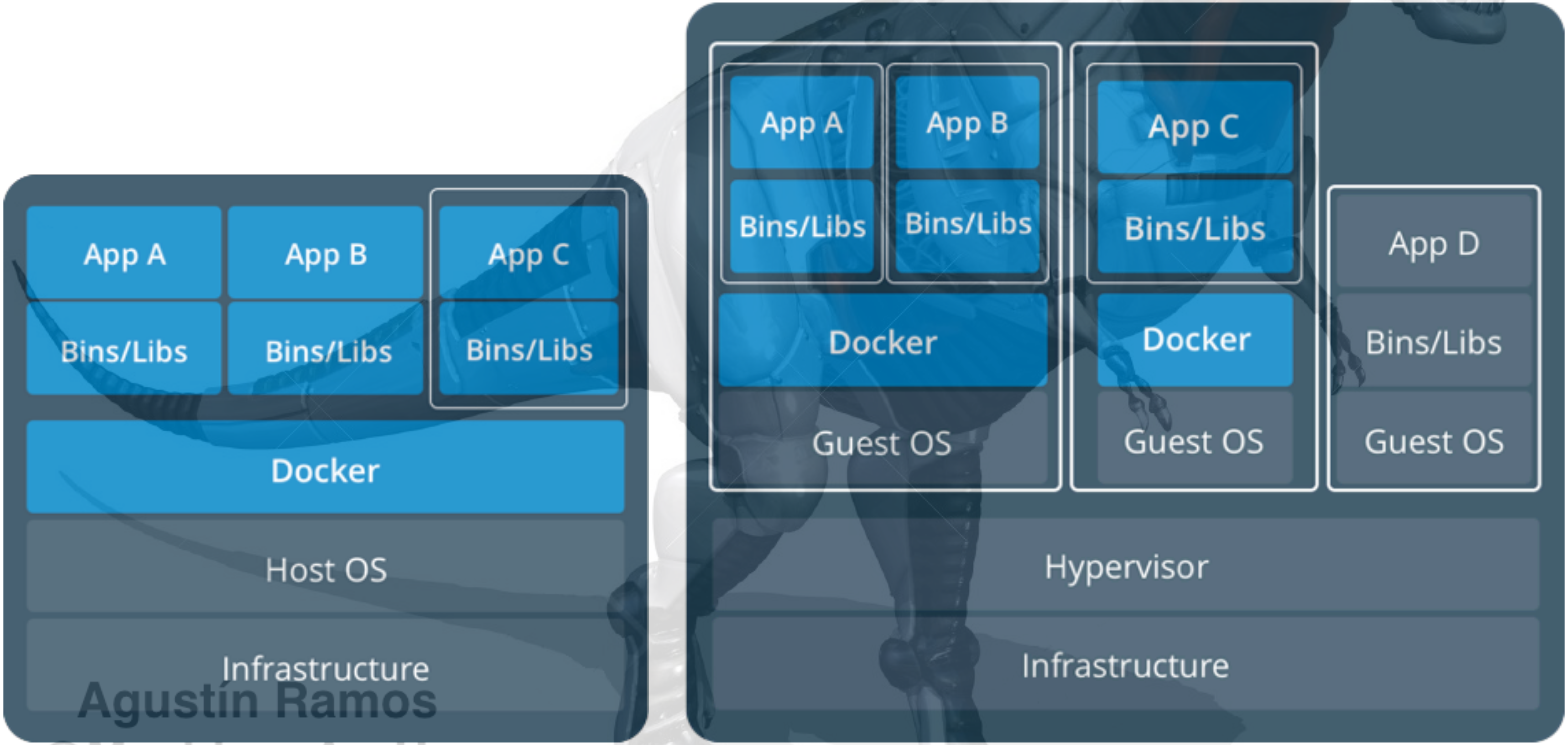
CONTAINER

Static Binary

Alpine

Kernel

Opciones de despliegue

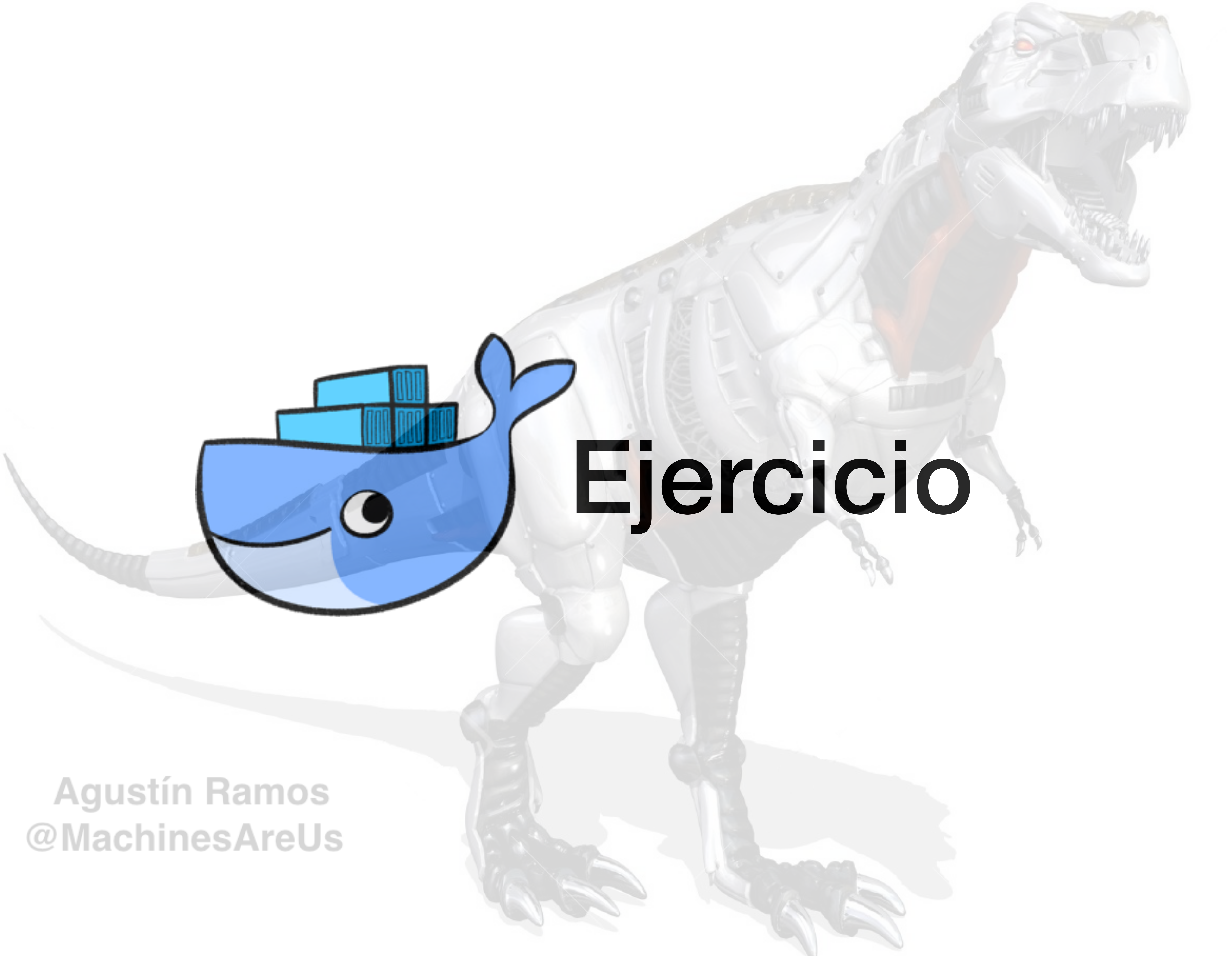


App A
Bins/Libs
App B
Bins/Libs
App C
Bins/Libs
Docker
Host OS
Infrastructure

App A
Bins/Libs
App B
Bins/Libs
Docker
Guest OS
App C
Bins/Libs
Docker
Guest OS
App D
Bins/Libs
Guest OS
Hypervisor
Infrastructure

Ventajas respecto a VMs

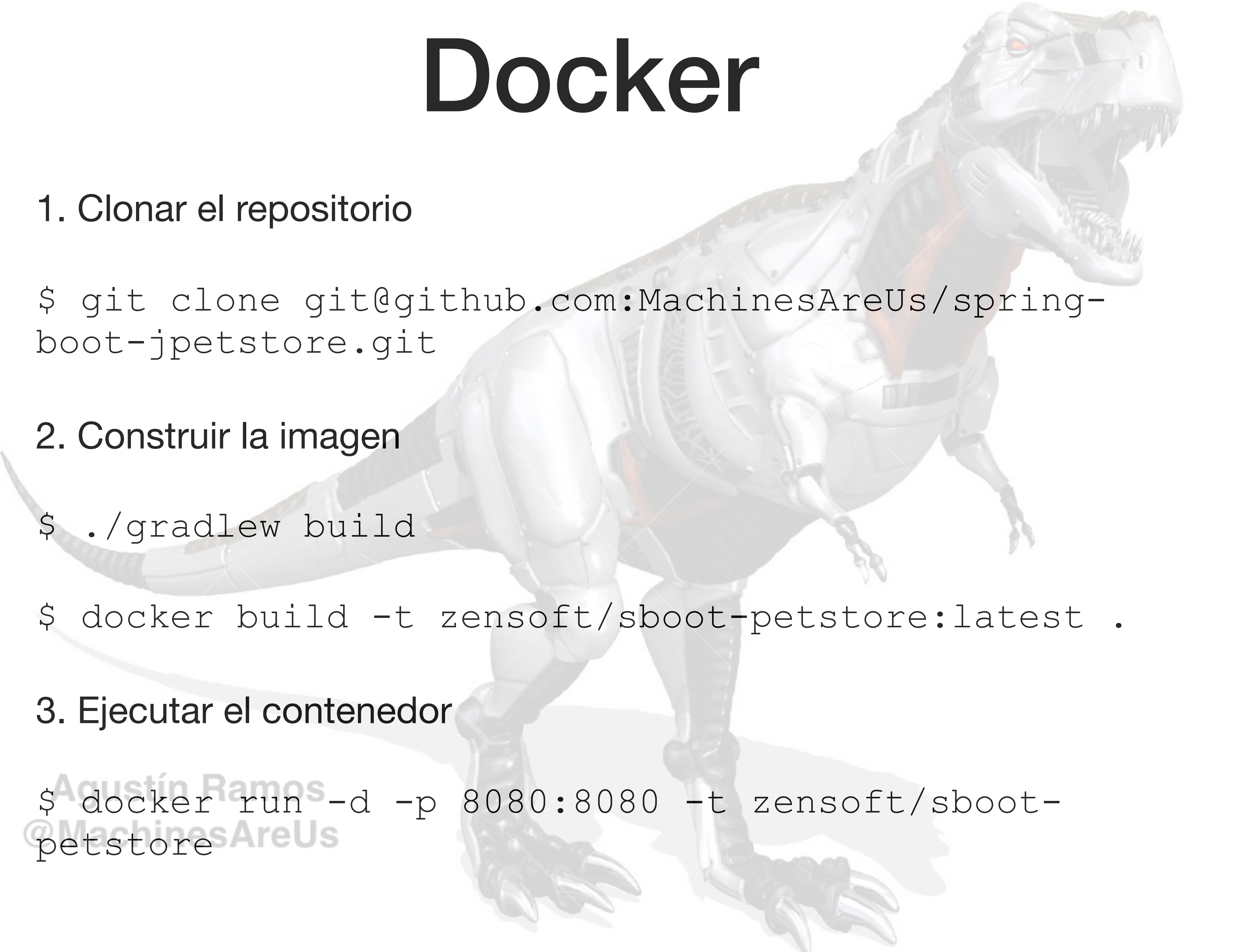
- Reutilización de la instalación del sistema operativo.
- Tiempos de creación y destrucción mucho más rápidos.
- Flexibilidad en reorganizar la configuración de un sistema.



Ejercicio

Agustín Ramos
@MachinesAreUs

Docker



1. Clonar el repositorio

```
$ git clone git@github.com:MachinesAreUs/spring-boot-jpetstore.git
```

2. Construir la imagen

```
$ ./gradlew build
```

```
$ docker build -t zensoft/sboot-petstore:latest .
```

3. Ejecutar el contenedor

```
$ docker run -d -p 8080:8080 -t zensoft/sboot-petstore
```

Docker

4. Etiquetar la imagen

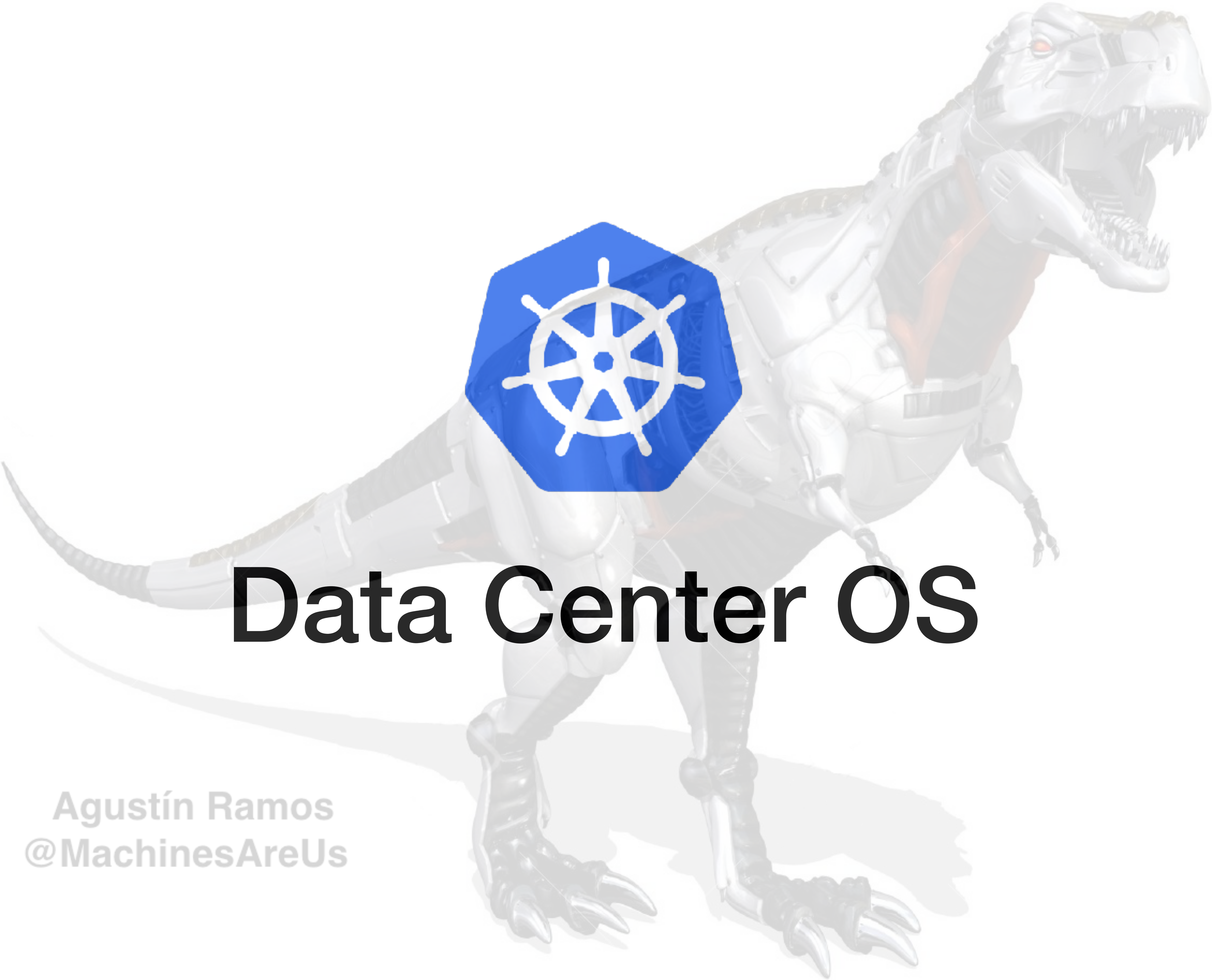
```
$ docker tag 25c8776185b6 machinesareus/
sboot-petstore:latest
```

5. Login a Docker Hub

```
$ docker login --username=machinesareus
```

6. Subir la imagen

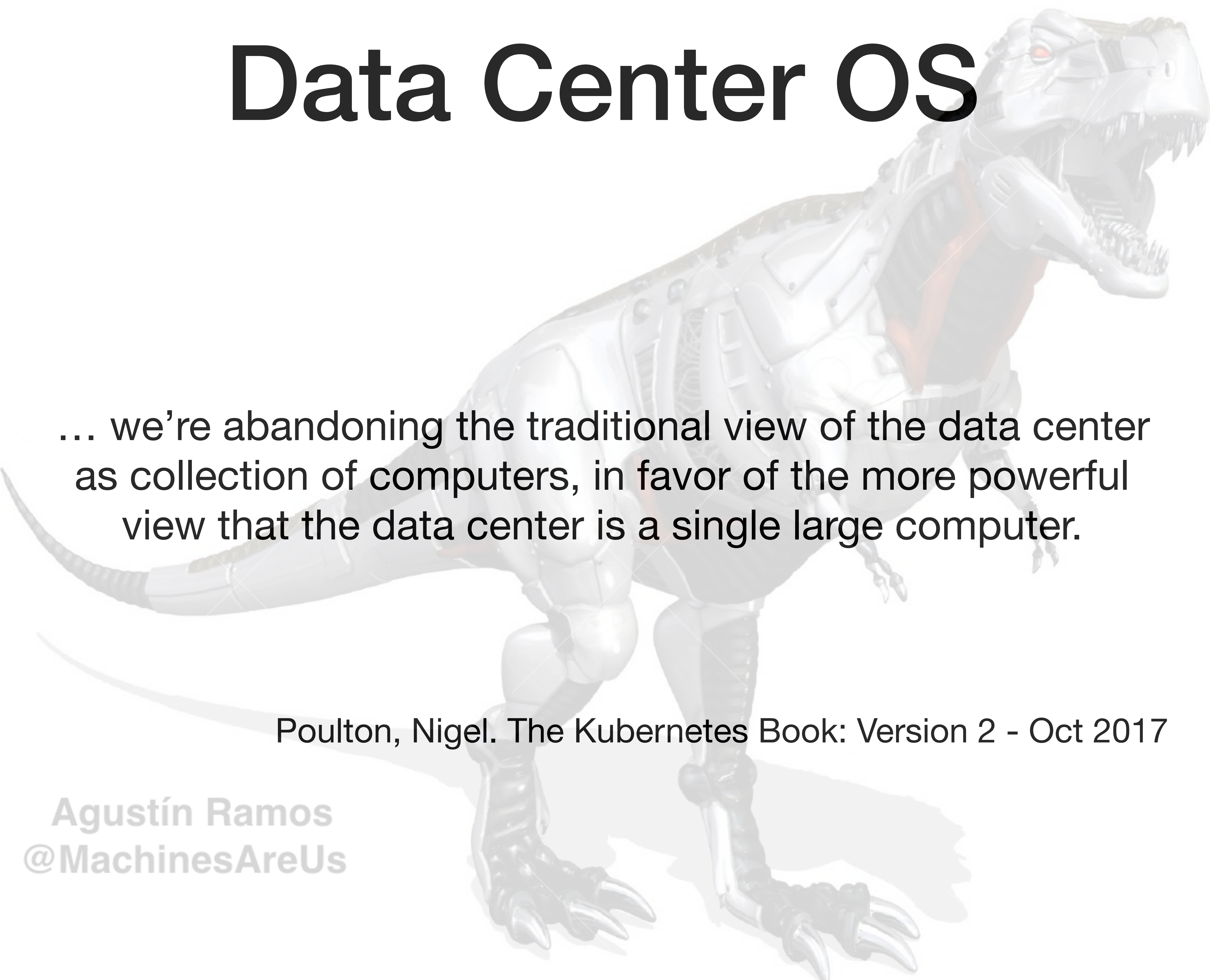
```
$ docker push machinesareus/sboot-petstore
```



Data Center OS

Agustín Ramos
@MachinesAreUs

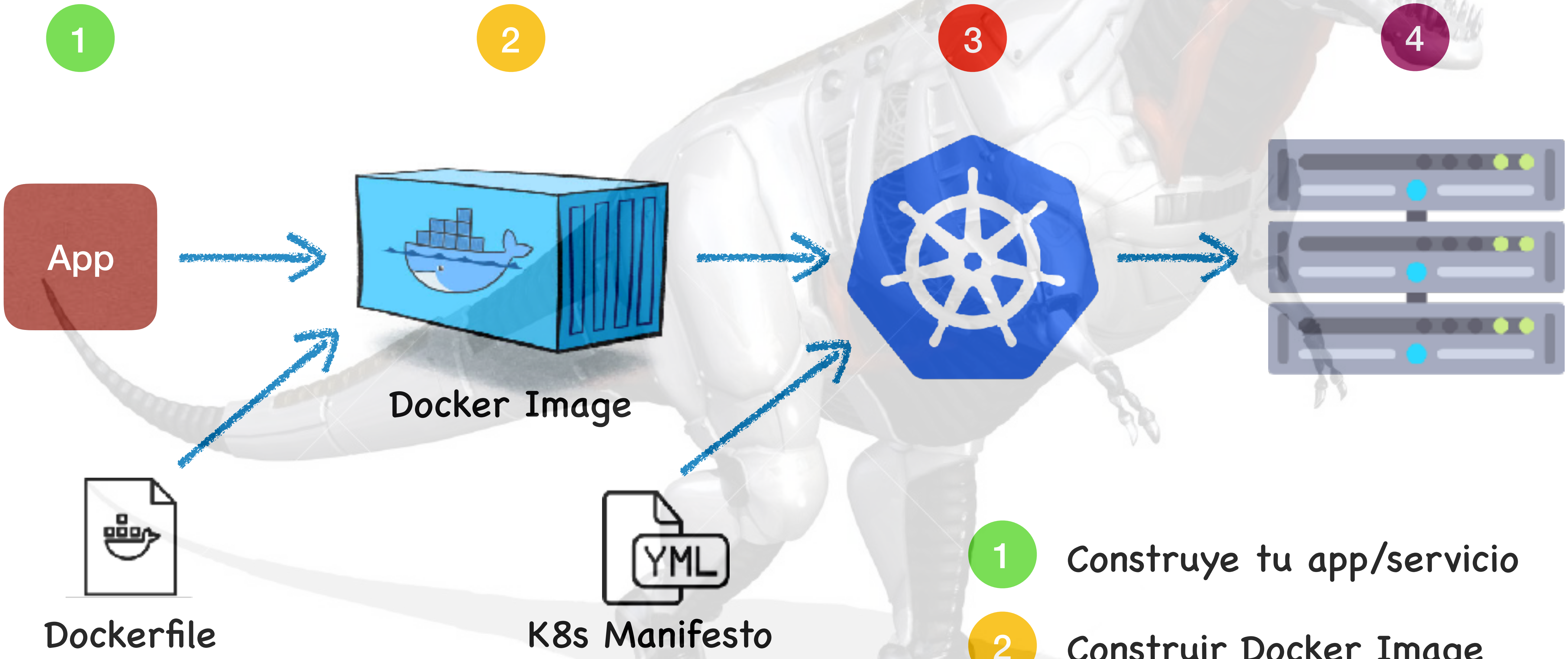
Data Center OS



… we’re abandoning the traditional view of the data center as collection of computers, in favor of the more powerful view that the data center is a single large computer.

Poulton, Nigel. The Kubernetes Book: Version 2 - Oct 2017

Flujo básico

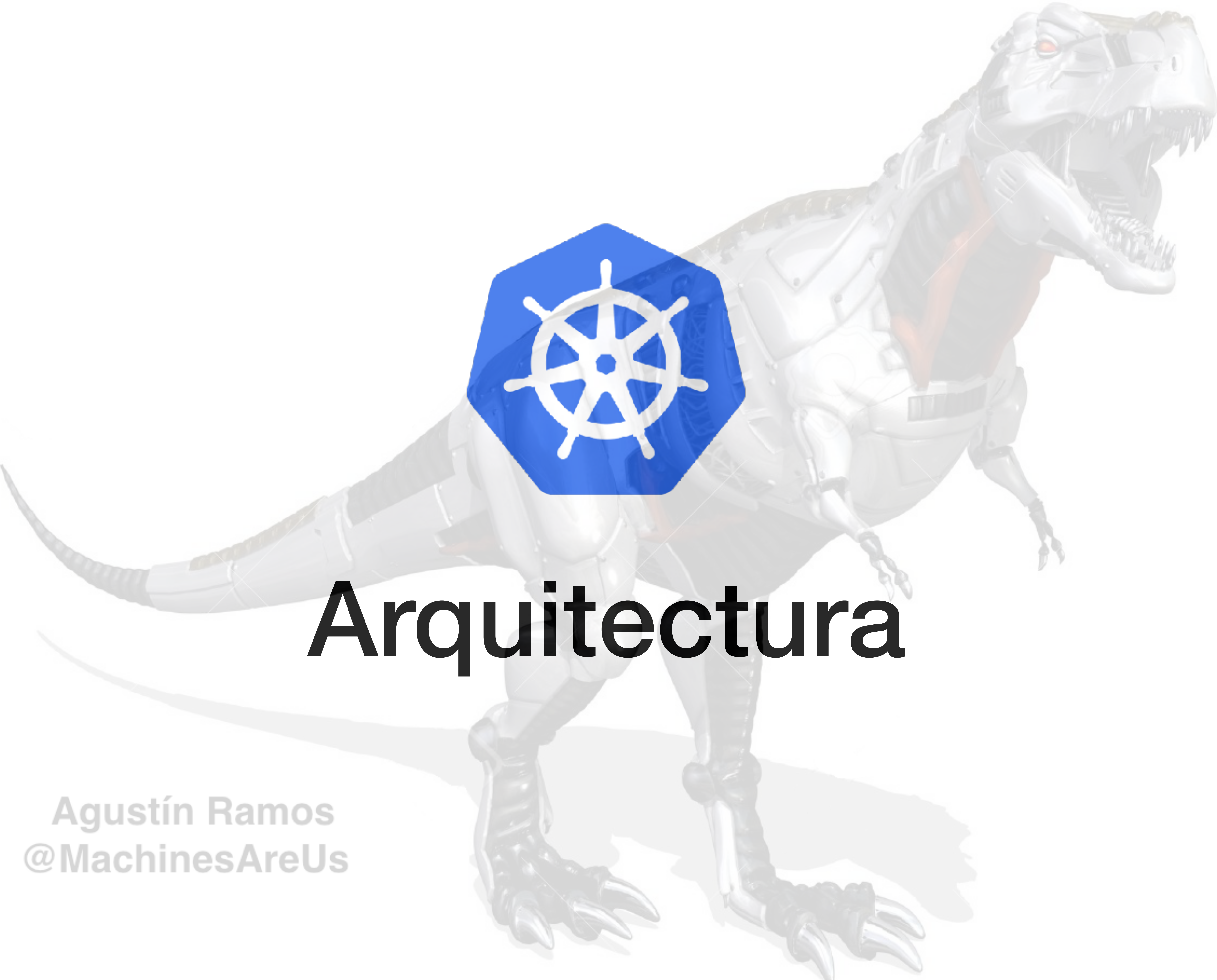


1 Construye tu app/servicio

2 Construir Docker Image

3 Despliega en K8s

4 Orquestación



Arquitectura

Agustín Ramos
@MachinesAreUs

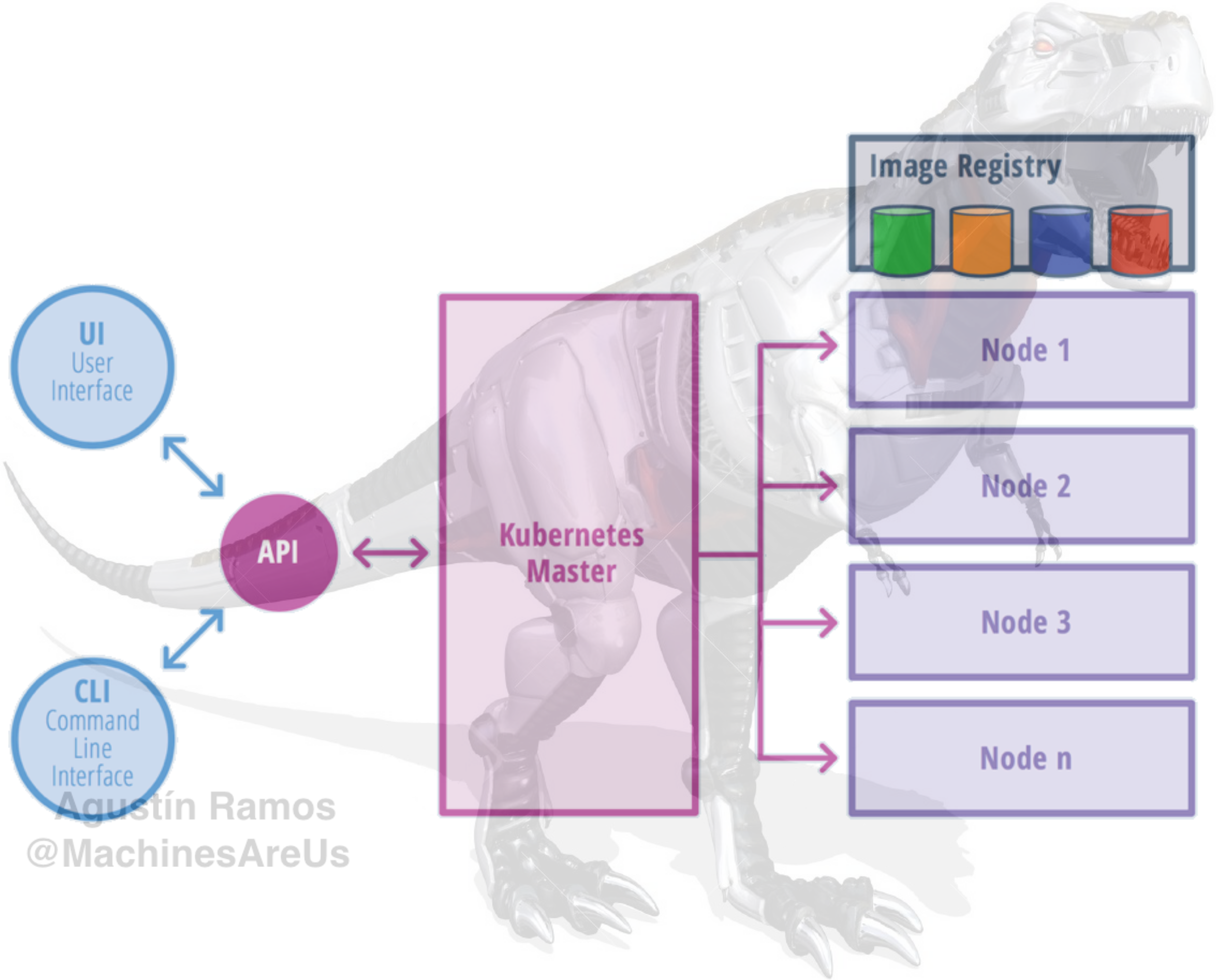


Image Registry

UI
User
Interface

CLI
Command
Line
Interface

API

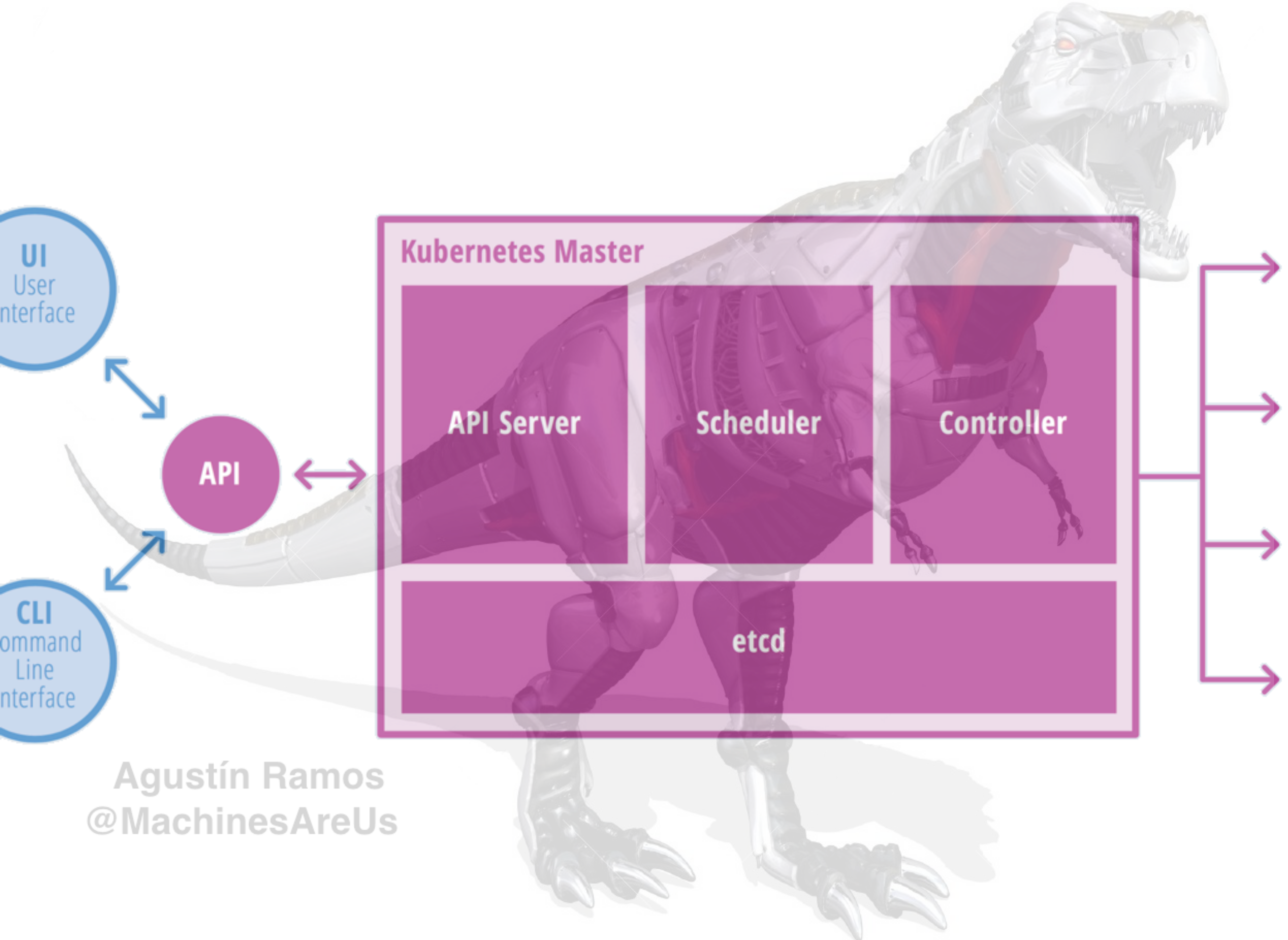
Kubernetes
Master

Node 1

Node 2

Node 3

Node n



UI
User
nterface

API

Kubernetes Master

API Server

Scheduler

Controller

etcd

CLI
ommand
Line
nterface

Agustín Ramos
@MachinesAreUs

Componentes de un Master

- API Server

  Es el punto de entrada de un cluster. Define todas las operaciones posibles.

- Scheduler

  Se encarga de asignar pods (contenedores) a los diferentes nodos del cluster.

- Controllers

  Nodes, Replication, Endpoints, Service Account & Token

- etcd

  Es un storage distribuido de tipo KV.

  Kubernetes lo usa para guardar sus configuraciones.

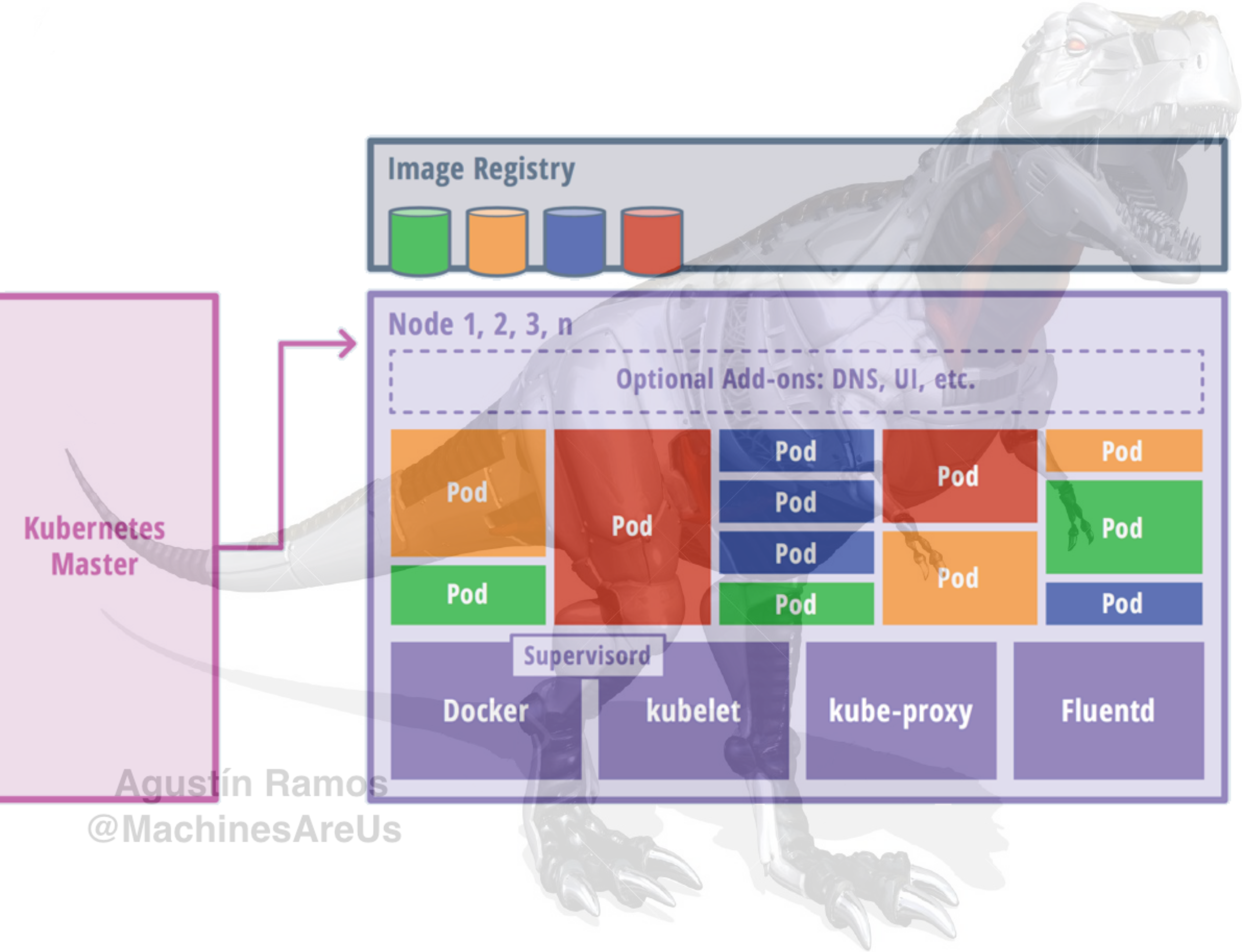


Image Registry
Node 1, 2, 3, n
Optional Add-ons: DNS, UI, etc.
Kubernetes Master
Pod
Pod
Pod
Pod
Pod
Pod
Pod
Pod
Pod
Pod
Pod
Pod
Pod
Supervisord
Docker
kubelet
kube-proxy
Fluentd

Componentes de un Node

- Docker

    Es el runtime de los contenedores.

- kubelet

    Es el “manager” de un nodo.

- kube-proxy

    Se encarga de la conectividad. Administra redes.

- fluentd

    Se encarga del bitacoreo

¿Sobre qué infraestructura puede correr?

Google Cloud Platform

amazon web services

Azure

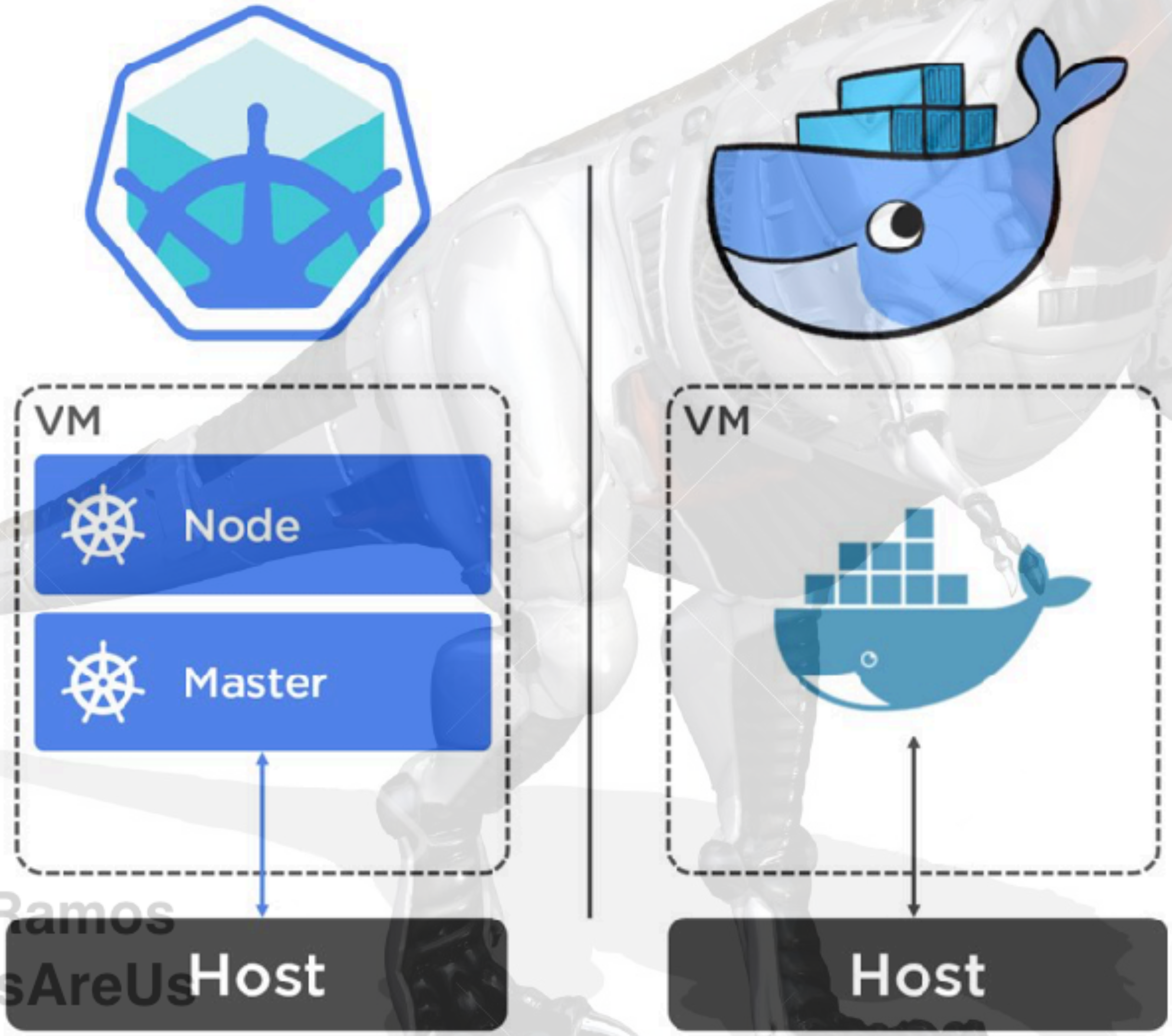
DigitalOcean

Tu propia infraestructura / data center

minikube

Agustín Ramos
@MachinesAreUs

Es setup es parecido a Docker



minikube 101

1. Iniciar minikube

```
$ minikube start —memory=4000 —vm-driver=xhyve
```

2. Crear un deployment con la imagen ***echoserver***

```
$ kubectl run hello-minikube --image=gcr.io/
google_containers/echoserver:1.4 --port=8080
```

3. Esperar a que el pod esté disponible

```
$ kubectl get pods
```

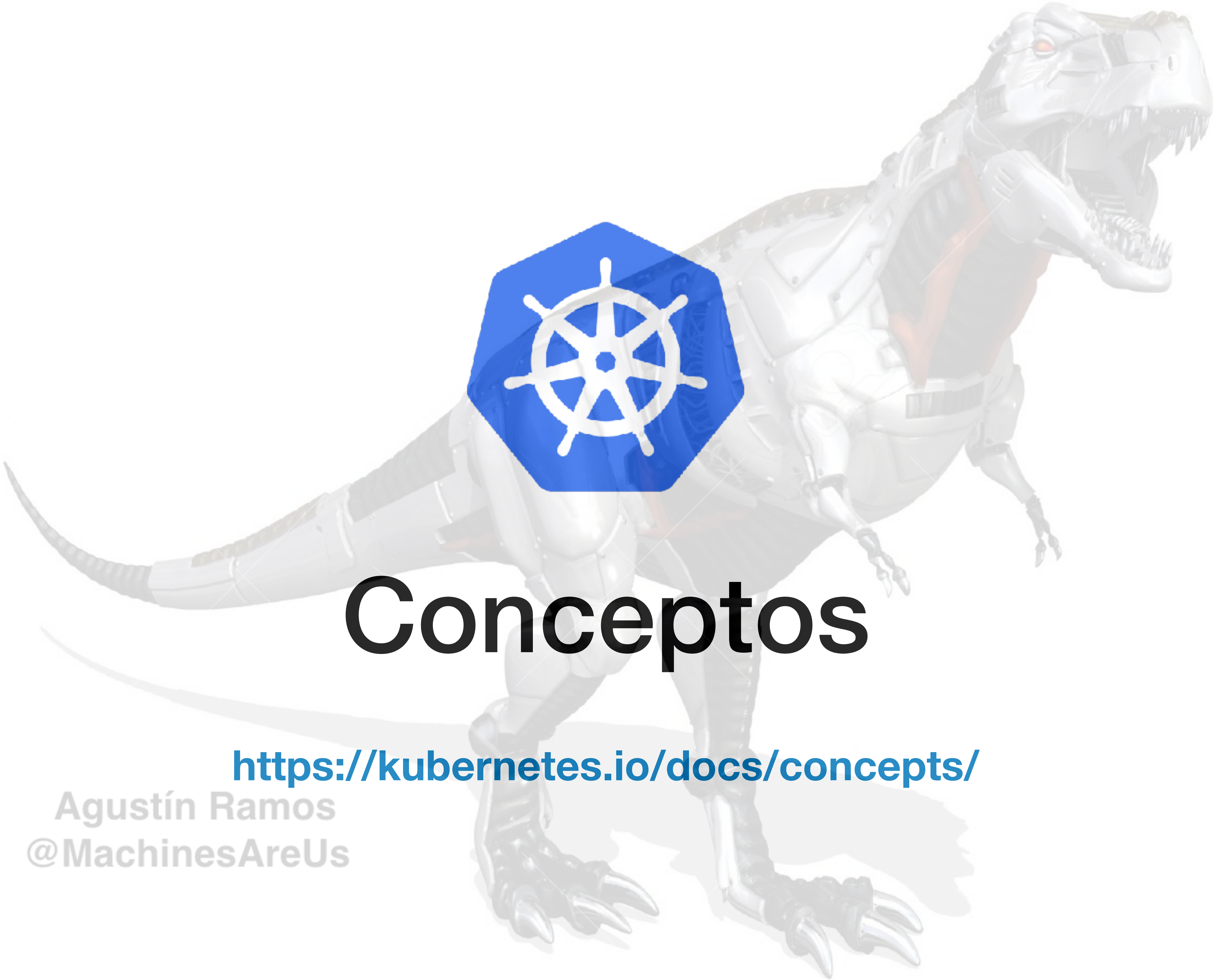
minikube 101

4. Exponer el deployment como servicio

```
$ kubectl expose deployment hello-minikube
—type=NodePort
```

5. Consultar el servicio

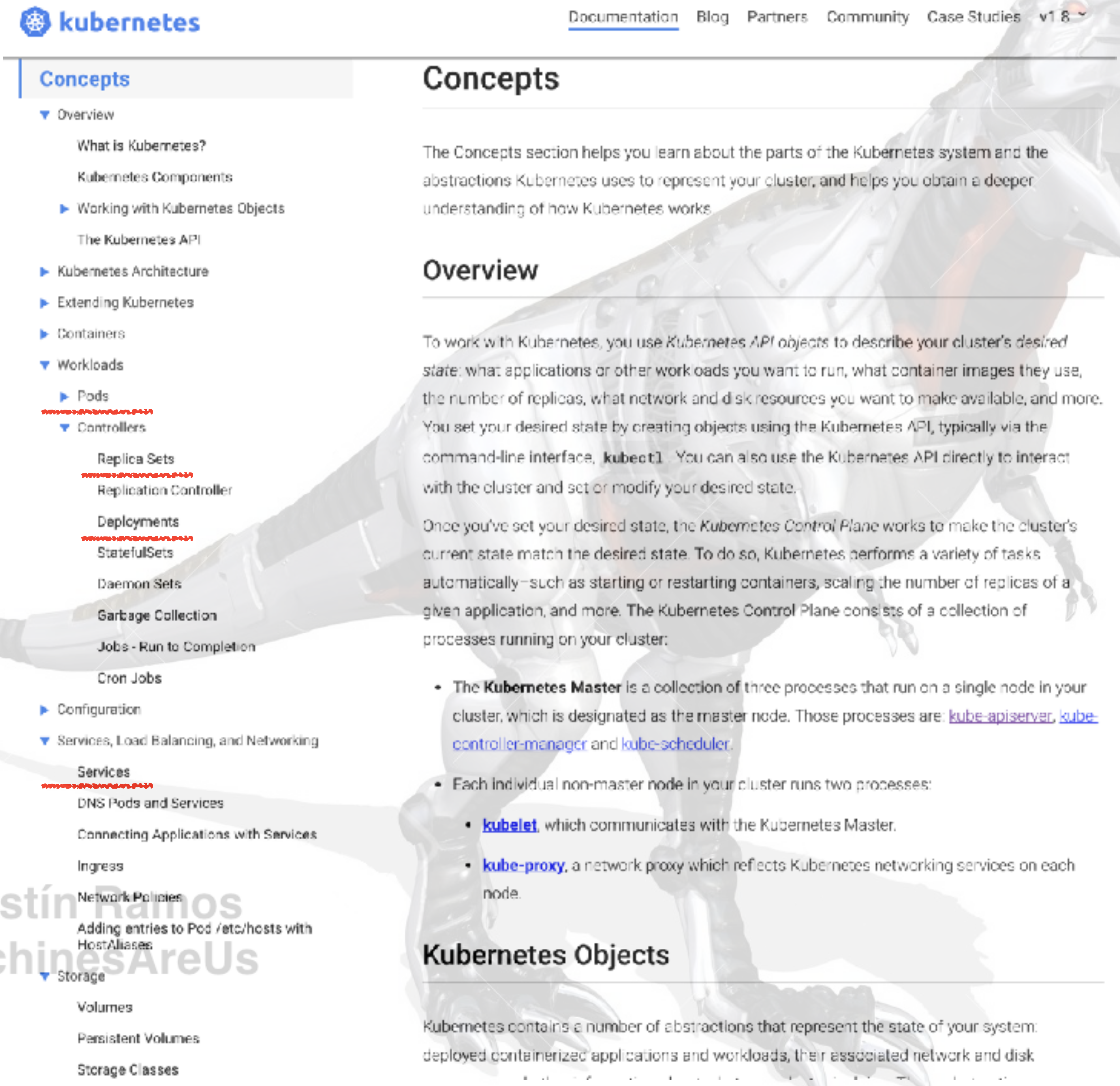
```
$ curl $(minikube service hello-minikube --
url)
```



Conceptos

https://kubernetes.io/docs/concepts/

Agustín Ramos
@MachinesAreUs



Concepts

- Overview
  - What is Kubernetes?
  - Kubernetes Components
  - Working with Kubernetes Objects
  - The Kubernetes API
- Kubernetes Architecture
- Extending Kubernetes
- Containers
- Workloads
  - Pods
  - Controllers
    - Replica Sets
    - Replication Controller
    - Deployments
    - StatefulSets
    - Daemon Sets
    - Garbage Collection
    - Jobs - Run to Completion
    - Cron Jobs
- Configuration
- Services, Load Balancing, and Networking
  - Services
  - DNS Pods and Services
  - Connecting Applications with Services
  - Ingress
  - Network Policies
  - Adding entries to Pod /etc/hosts with HostAliases
- Storage
  - Volumes
  - Persistent Volumes
  - Storage Classes

Concepts

The Concepts section helps you learn about the parts of the Kubernetes system and the abstractions Kubernetes uses to represent your cluster, and helps you obtain a deeper understanding of how Kubernetes works.

Overview

To work with Kubernetes, you use *Kubernetes API objects* to describe your cluster's *desired state*: what applications or other workloads you want to run, what container images they use, the number of replicas, what network and disk resources you want to make available, and more. You set your desired state by creating objects using the Kubernetes API, typically via the command-line interface, `kubectl`. You can also use the Kubernetes API directly to interact with the cluster and set or modify your desired state.

Once you've set your desired state, the *Kubernetes Control Plane* works to make the cluster's current state match the desired state. To do so, Kubernetes performs a variety of tasks automatically–such as starting or restarting containers, scaling the number of replicas of a given application, and more. The Kubernetes Control Plane consists of a collection of processes running on your cluster:

- The **Kubernetes Master** is a collection of three processes that run on a single node in your cluster, which is designated as the master node. Those processes are: kube-apiserver, kube-controller-manager and kube-scheduler.
- Each individual non-master node in your cluster runs two processes:
  - **kubelet**, which communicates with the Kubernetes Master.
  - **kube-proxy**, a network proxy which reflects Kubernetes networking services on each node.

Kubernetes Objects

Kubernetes contains a number of abstractions that represent the state of your system: deployed containerized applications and workloads, their associated network and disk

Conceptos Básicos

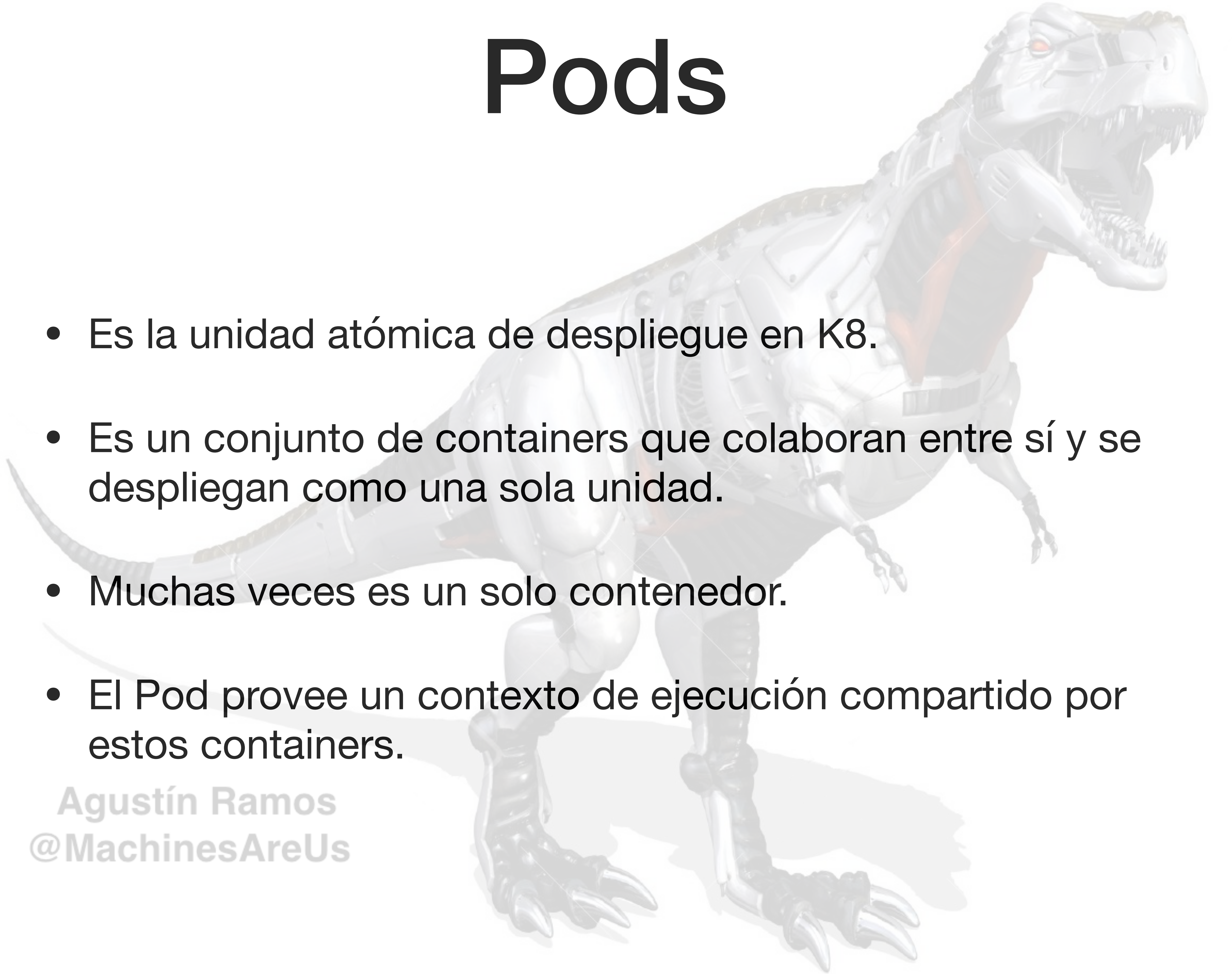
Services

Deployments

Replica Sets

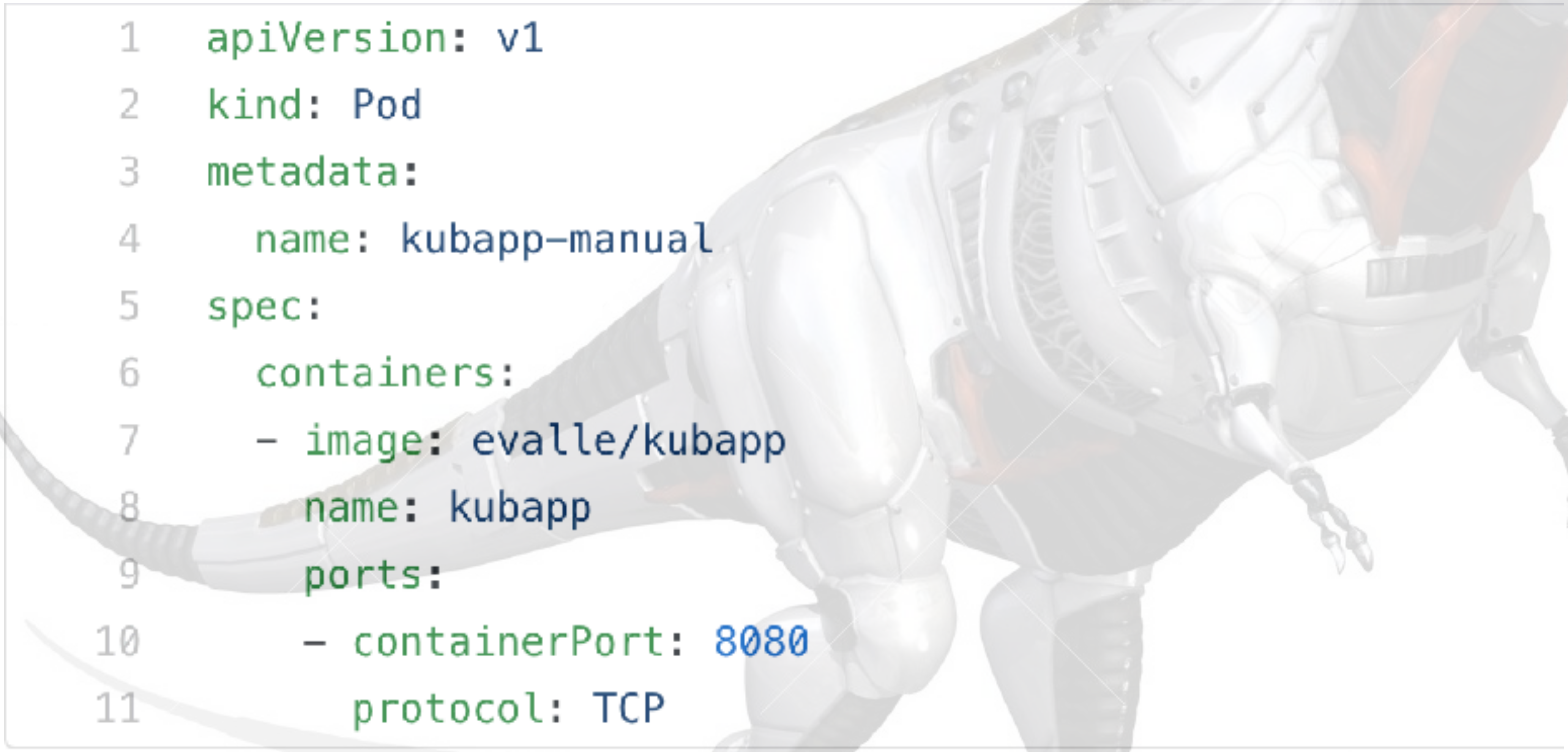
Pods

Pods



- Es la unidad atómica de despliegue en K8.
- Es un conjunto de containers que colaboran entre sí y se despliegan como una sola unidad.
- Muchas veces es un solo contenedor.
- El Pod provee un contexto de ejecución compartido por estos containers.

Pods - Demo



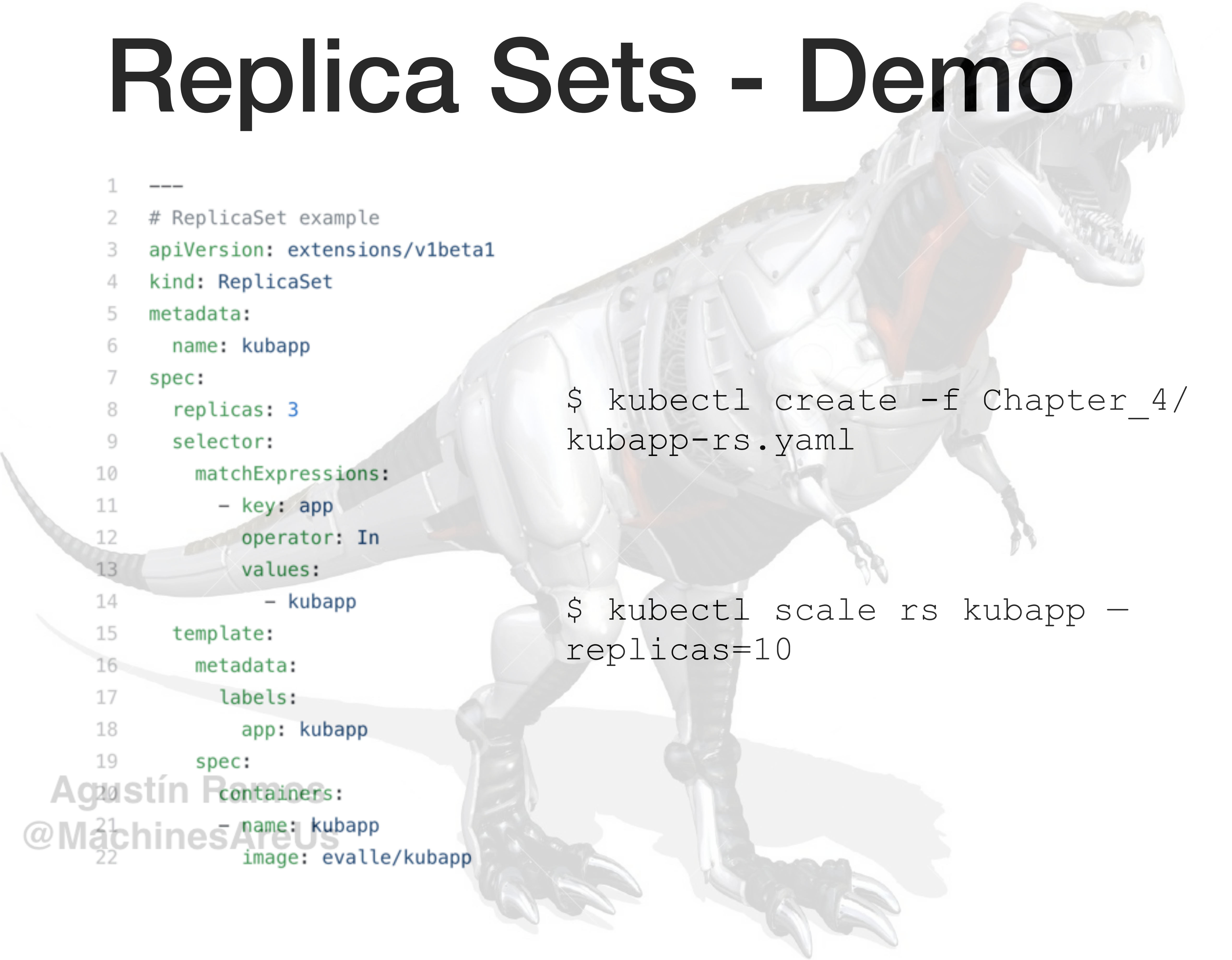
```yaml
apiVersion: v1
kind: Pod
metadata:
  name: kubapp-manual
spec:
  containers:
  - image: evalle/kubapp
    name: kubapp
    ports:
    - containerPort: 8080
      protocol: TCP
```

```
$ kubectl create -f Chapter_3/nodejs_app.yaml
```

Replica Sets

- Se encargan de crear un “cluster” de pods (réplicas)
- Se define un estado deseado, y el RS se encarga de mantener este estado.
- Si se muere un pod, el RS crea uno nuevo en su lugar.

Replica Sets - Demo



```
---
# ReplicaSet example
apiVersion: extensions/v1beta1
kind: ReplicaSet
metadata:
  name: kubapp
spec:
  replicas: 3
  selector:
    matchExpressions:
      - key: app
        operator: In
        values:
          - kubapp
  template:
    metadata:
      labels:
        app: kubapp
    spec:
      containers:
      - name: kubapp
        image: evalle/kubapp
```

```
$ kubectl create -f Chapter_4/
kubapp-rs.yaml
```

```
$ kubectl scale rs kubapp —
replicas=10
```

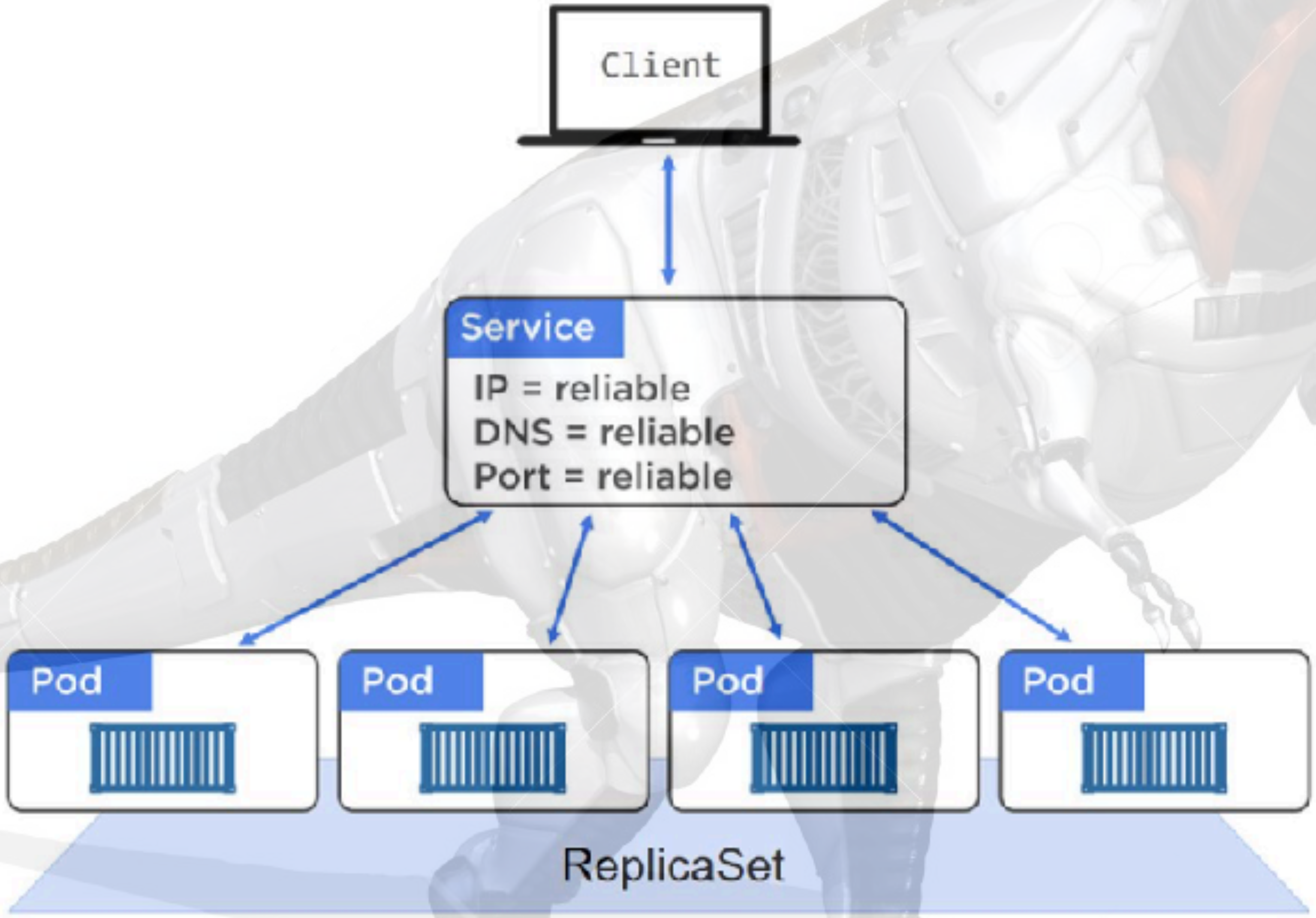
Deployment

- Es un mecanismo de realizar actualizaciones declarativas a Pods y Replica Sets.
- Fue introducido posteriormente a K8s.

Service

- Permite exponer un RS a la red, dentro o fuera del cluster.
    - Las IP’s de los Pods no son confiables.
- Tipos de servicios.
    - ClusterIP
    - NodePort
    - LoadBalancer

Service



Client

Service

IP = reliable
DNS = reliable
Port = reliable

Pod

Pod

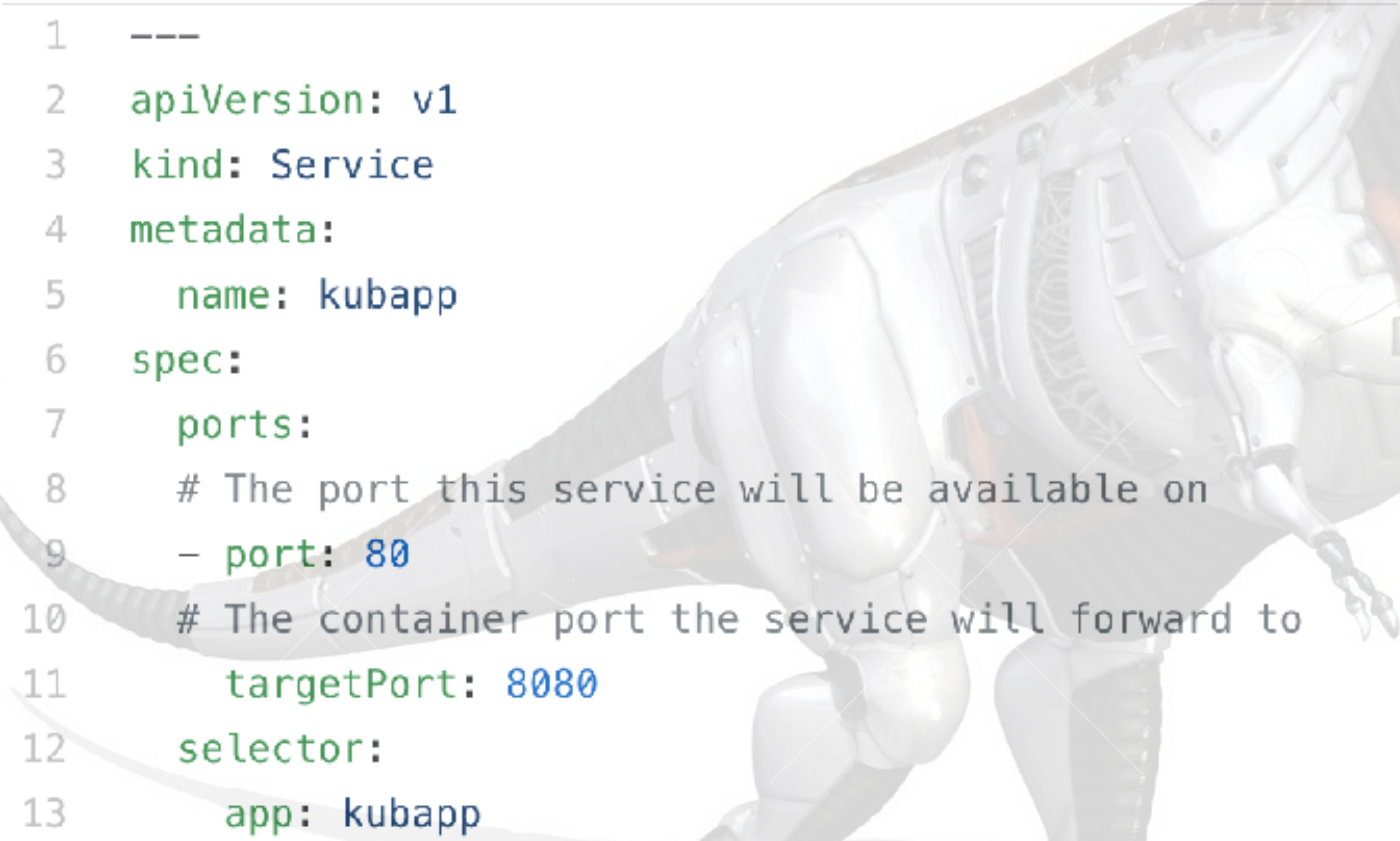
Pod

Pod

ReplicaSet

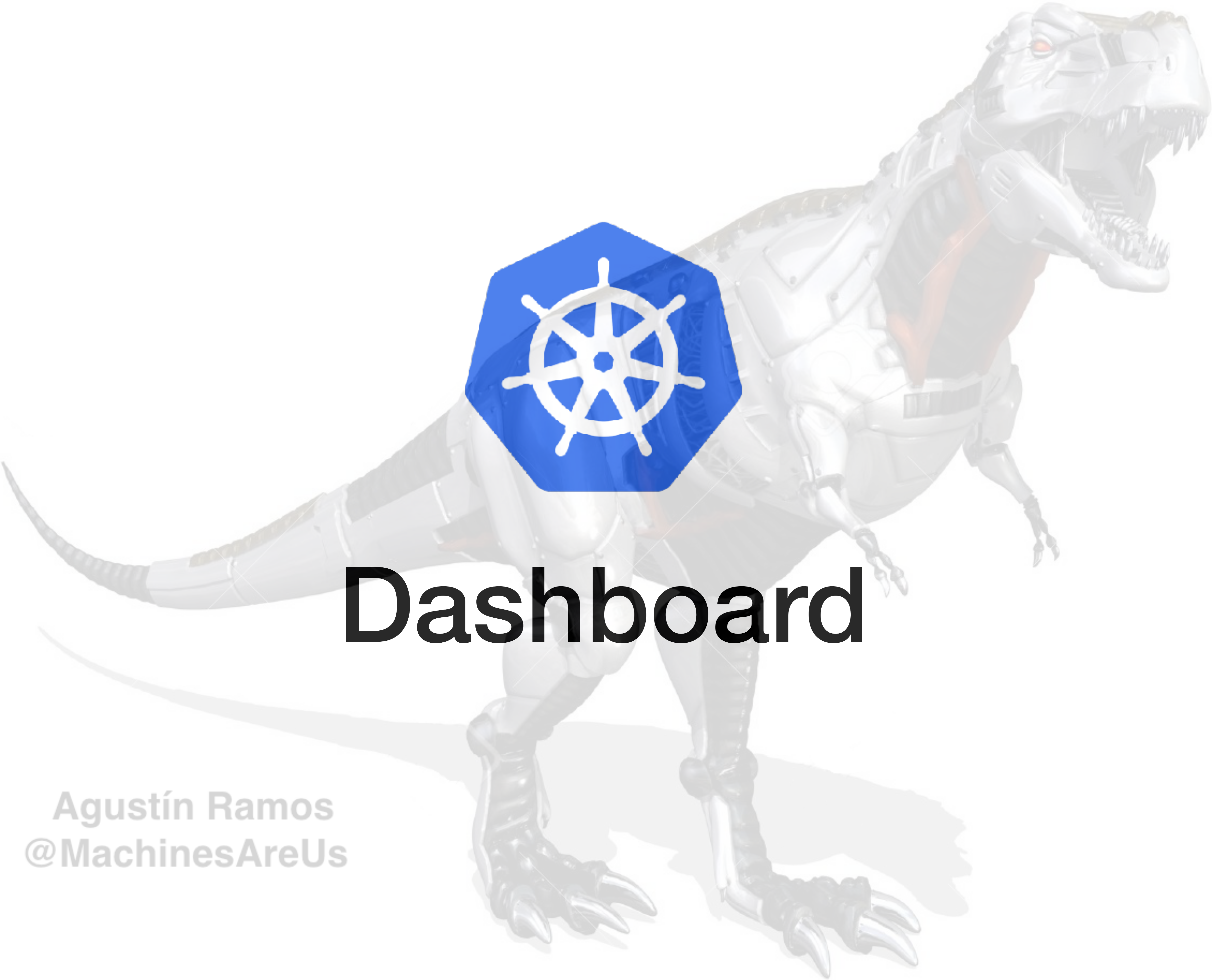
Agustín Ramos
@MachinesAreUs

Service - Demo



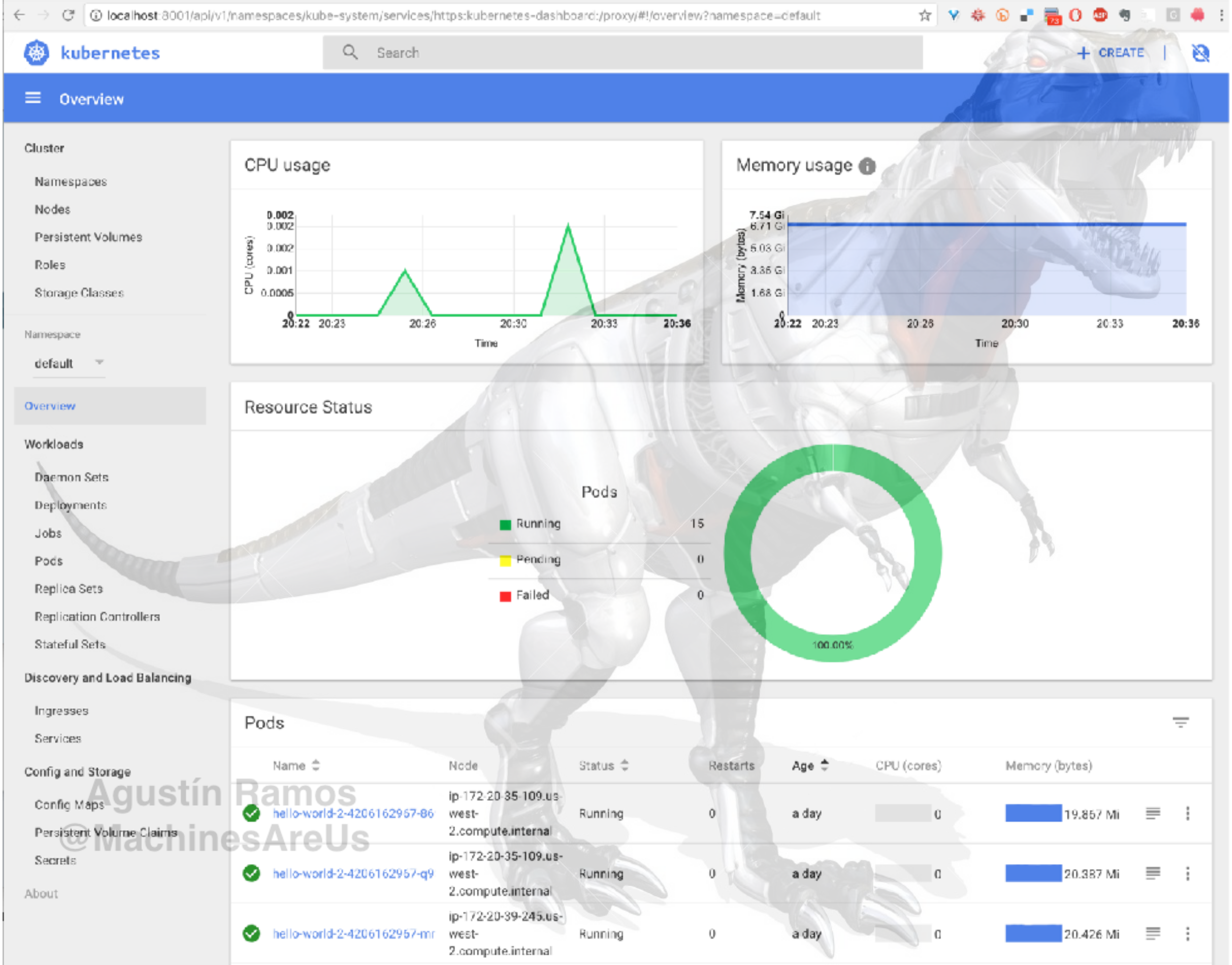
```yaml
---
apiVersion: v1
kind: Service
metadata:
  name: kubapp
spec:
  ports:
  # The port this service will be available on
  - port: 80
  # The container port the service will forward to
    targetPort: 8080
  selector:
    app: kubapp
```

```
$ kubectl create -f Chapter_5/kubapp-svc.yaml
```



Dashboard

Agustín Ramos
@MachinesAreUs



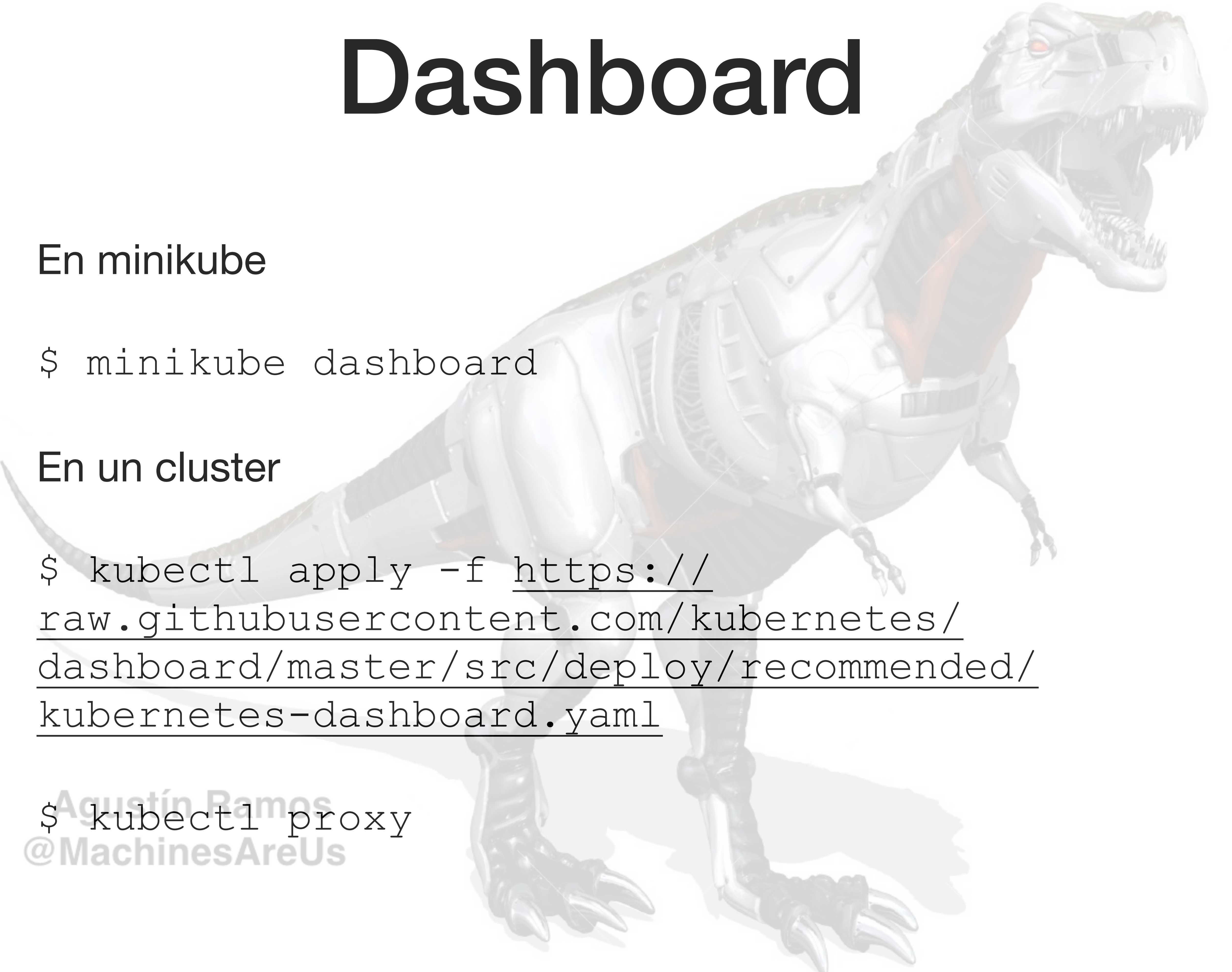
Pods

🟩 Running	15
🟨 Pending	0
🟥 Failed	0

Pods

Name	Node	Status	Restarts	Age	CPU (cores)	Memory (bytes)
✓ hello-world-2-4206162957-86	ip-172-20-35-109.us-west-2.compute.internal	Running	0	a day	0	19.857 Mi
✓ hello-world-2-4206162957-q9	ip-172-20-35-109.us-west-2.compute.internal	Running	0	a day	0	20.387 Mi
✓ hello-world-2-4206162957-mr	ip-172-20-39-245.us-west-2.compute.internal	Running	0	a day	0	20.426 Mi

Dashboard



En minikube

```
$ minikube dashboard
```

En un cluster

```
$ kubectl apply -f https://
raw.githubusercontent.com/kubernetes/
dashboard/master/src/deploy/recommended/
kubernetes-dashboard.yaml
```

```
$ kubectl proxy
```

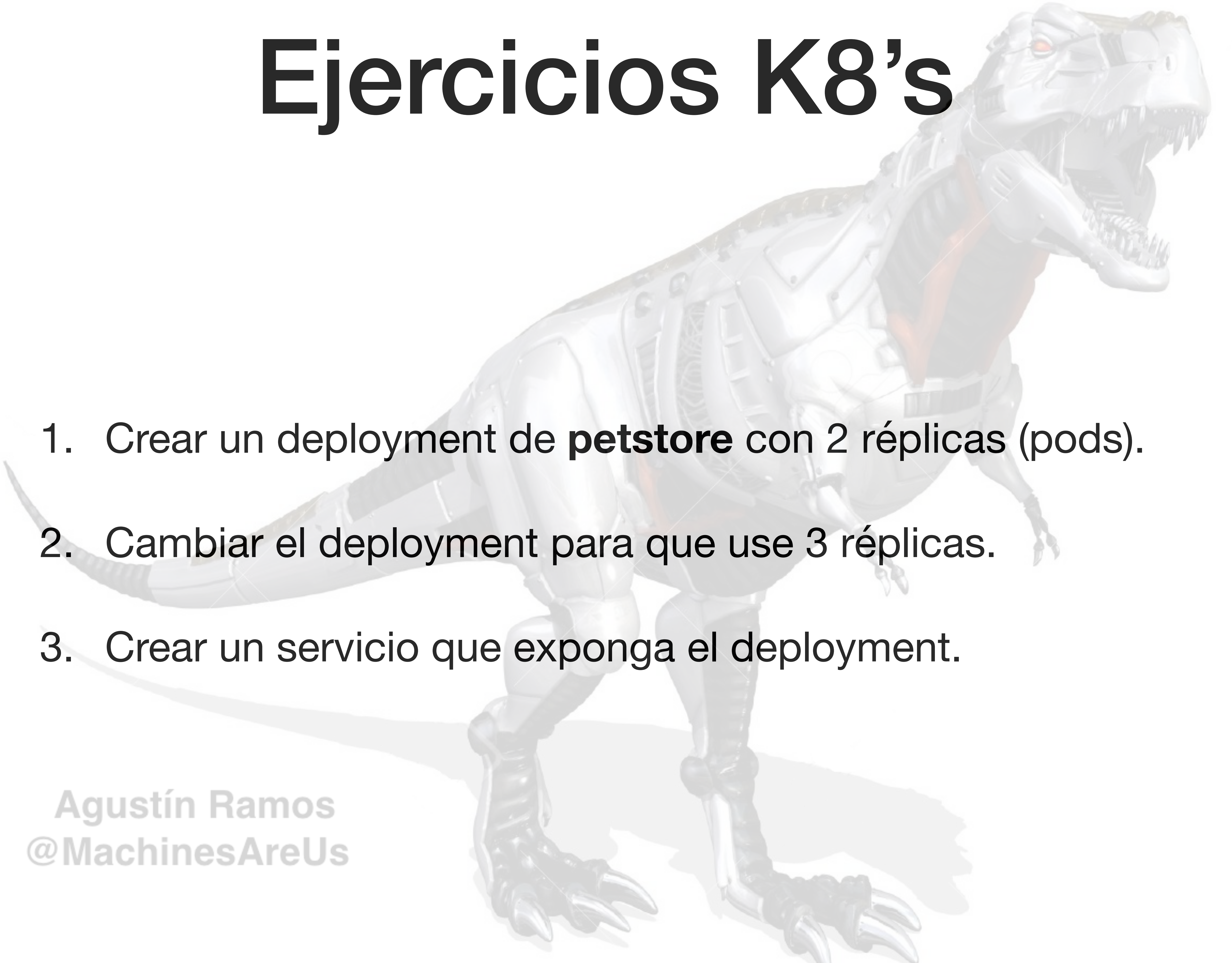
Casos de Uso

Agustín Ramos
@MachinesAreUs

Ejercicios

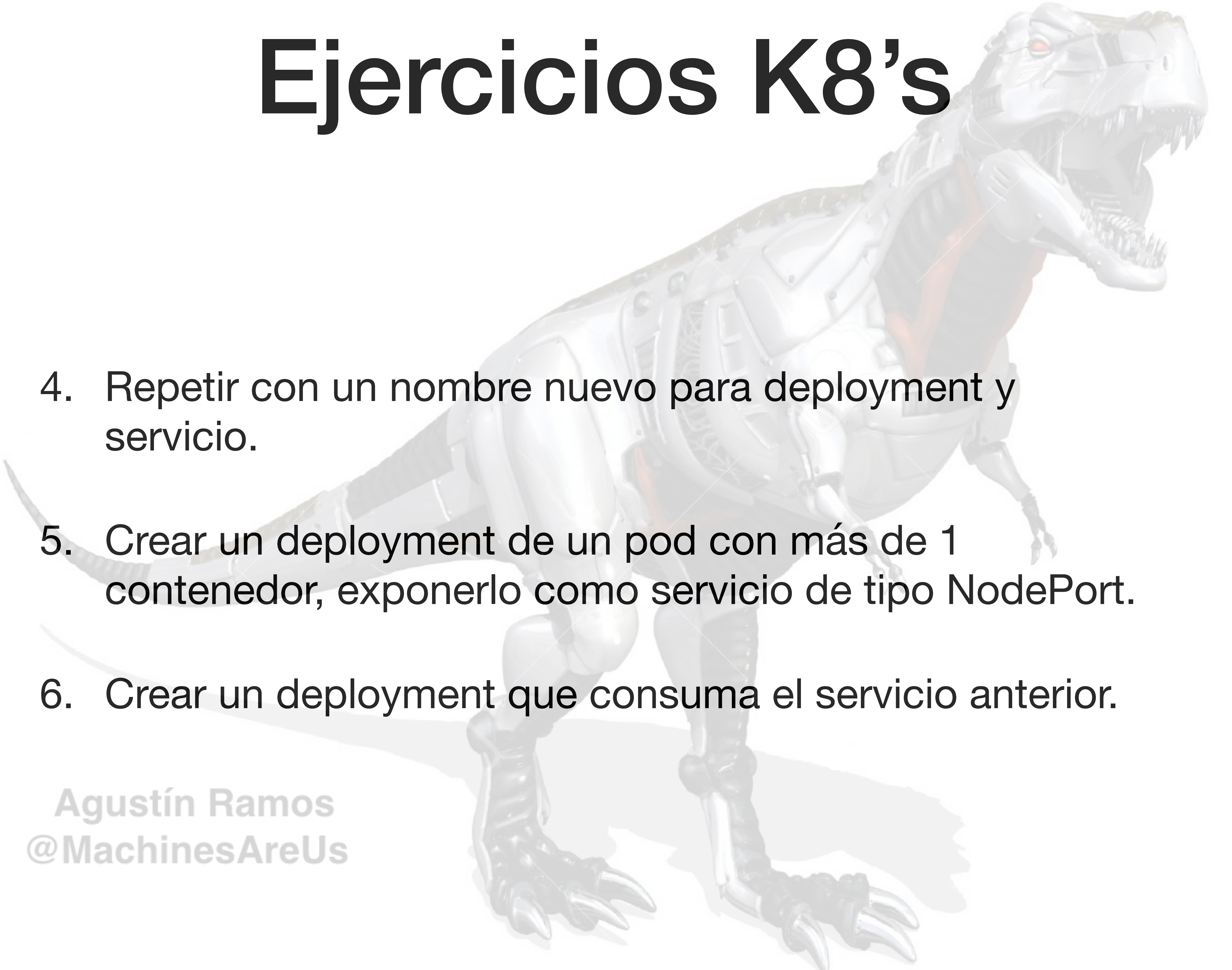
Agustín Ramos
@MachinesAreUs

Ejercicios K8’s



1. Crear un deployment de **petstore** con 2 réplicas (pods).
2. Cambiar el deployment para que use 3 réplicas.
3. Crear un servicio que exponga el deployment.

Ejercicios K8's



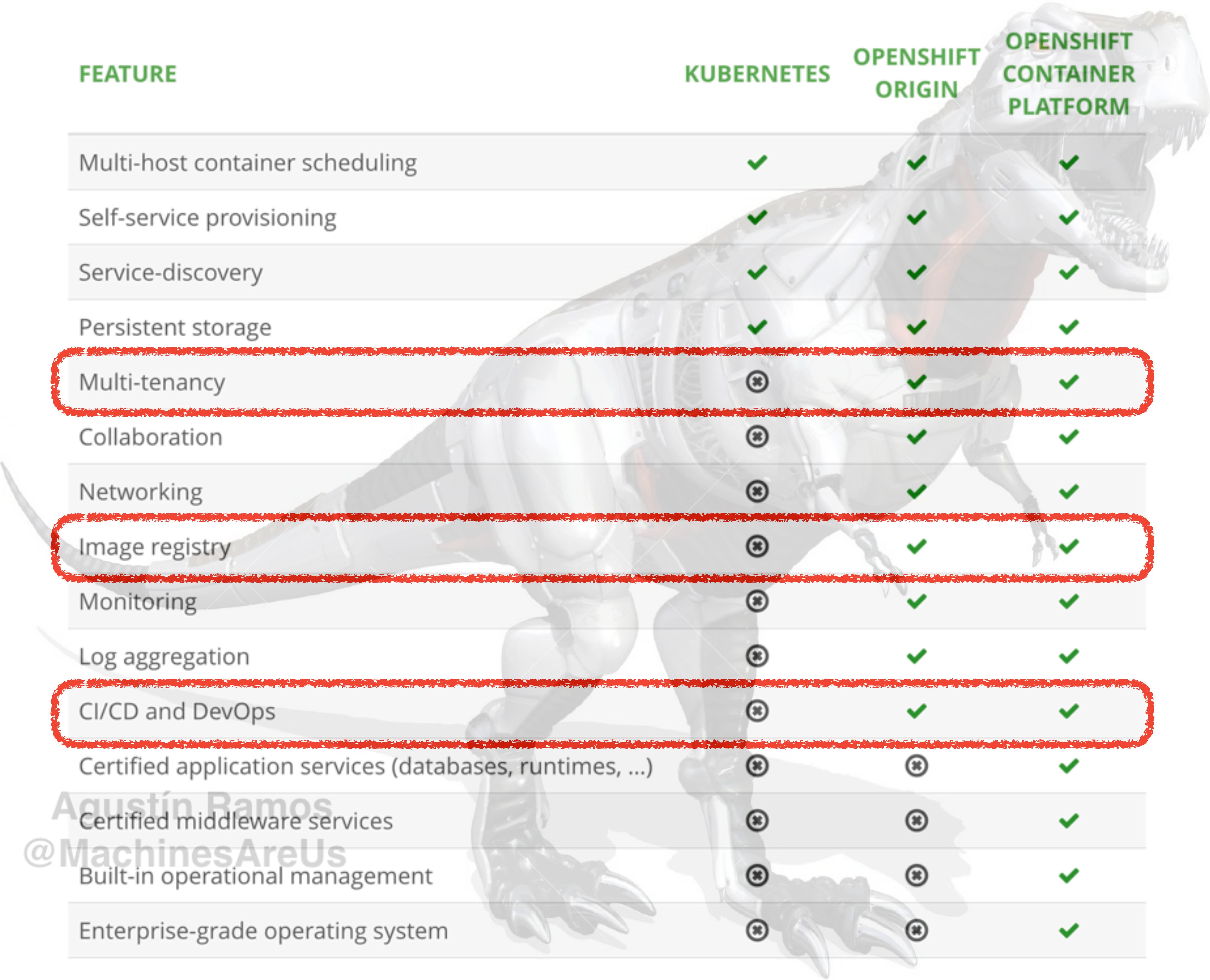
4. Repetir con un nombre nuevo para deployment y servicio.
5. Crear un deployment de un pod con más de 1 contenedor, exponerlo como servicio de tipo NodePort.
6. Crear un deployment que consuma el servicio anterior.

Distribuciones Comerciales

Agustín Ramos
@MachinesAreUs

RED HAT® OPENSHIFT

Agustín Ramos
@MachinesAreUs



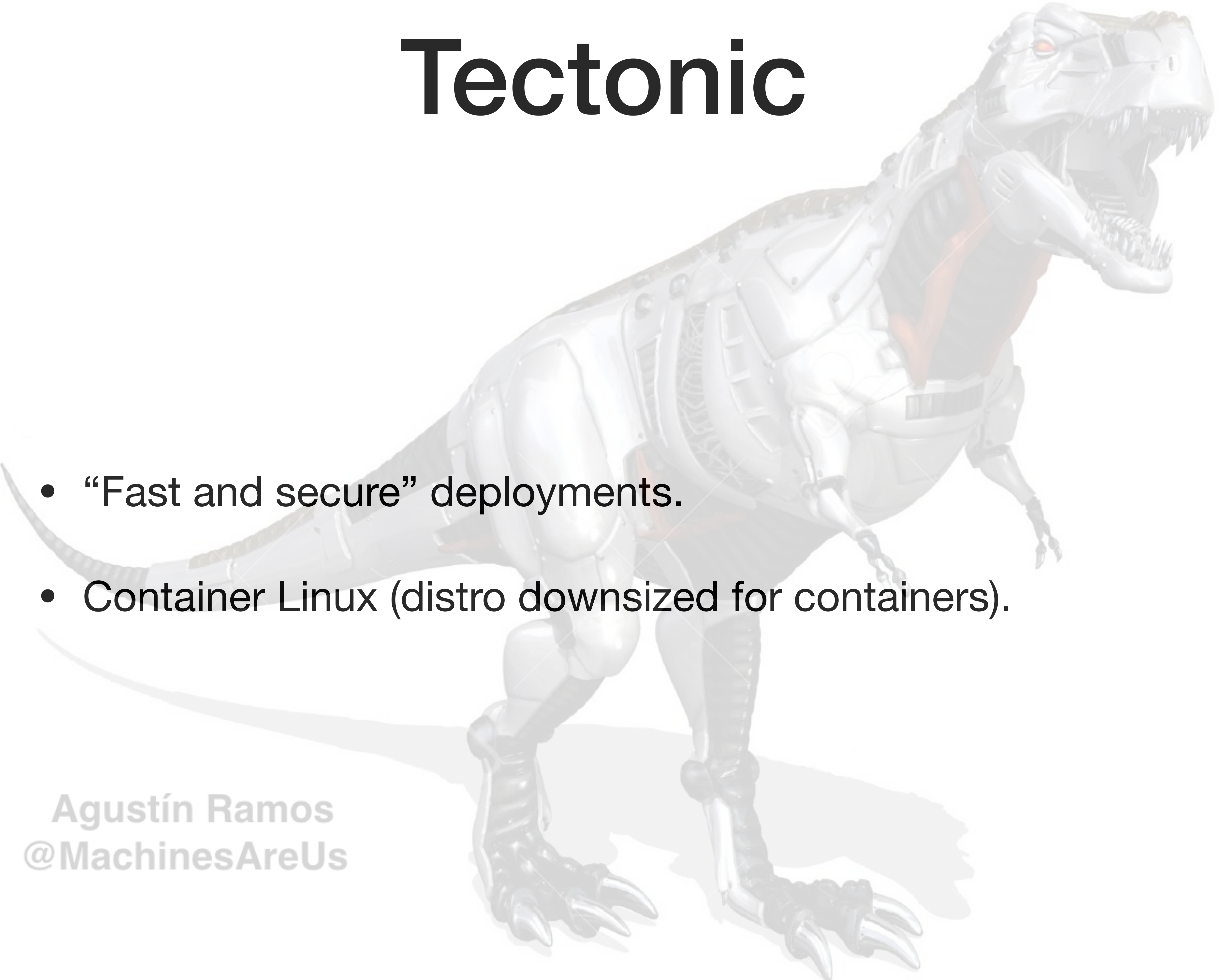
FEATURE	KUBERNETES	OPENSHIFT ORIGIN	OPENSHIFT CONTAINER PLATFORM
Multi-host container scheduling	✓	✓	✓
Self-service provisioning	✓	✓	✓
Service-discovery	✓	✓	✓
Persistent storage	✓	✓	✓
Multi-tenancy	⊗	✓	✓
Collaboration	⊗	✓	✓
Networking	⊗	✓	✓
Image registry	⊗	✓	✓
Monitoring	⊗	✓	✓
Log aggregation	⊗	✓	✓
CI/CD and DevOps	⊗	✓	✓
Certified application services (databases, runtimes, ...)	⊗	⊗	✓
Certified middleware services	⊗	⊗	✓
Built-in operational management	⊗	⊗	✓
Enterprise-grade operating system	⊗	⊗	✓

TECTONIC

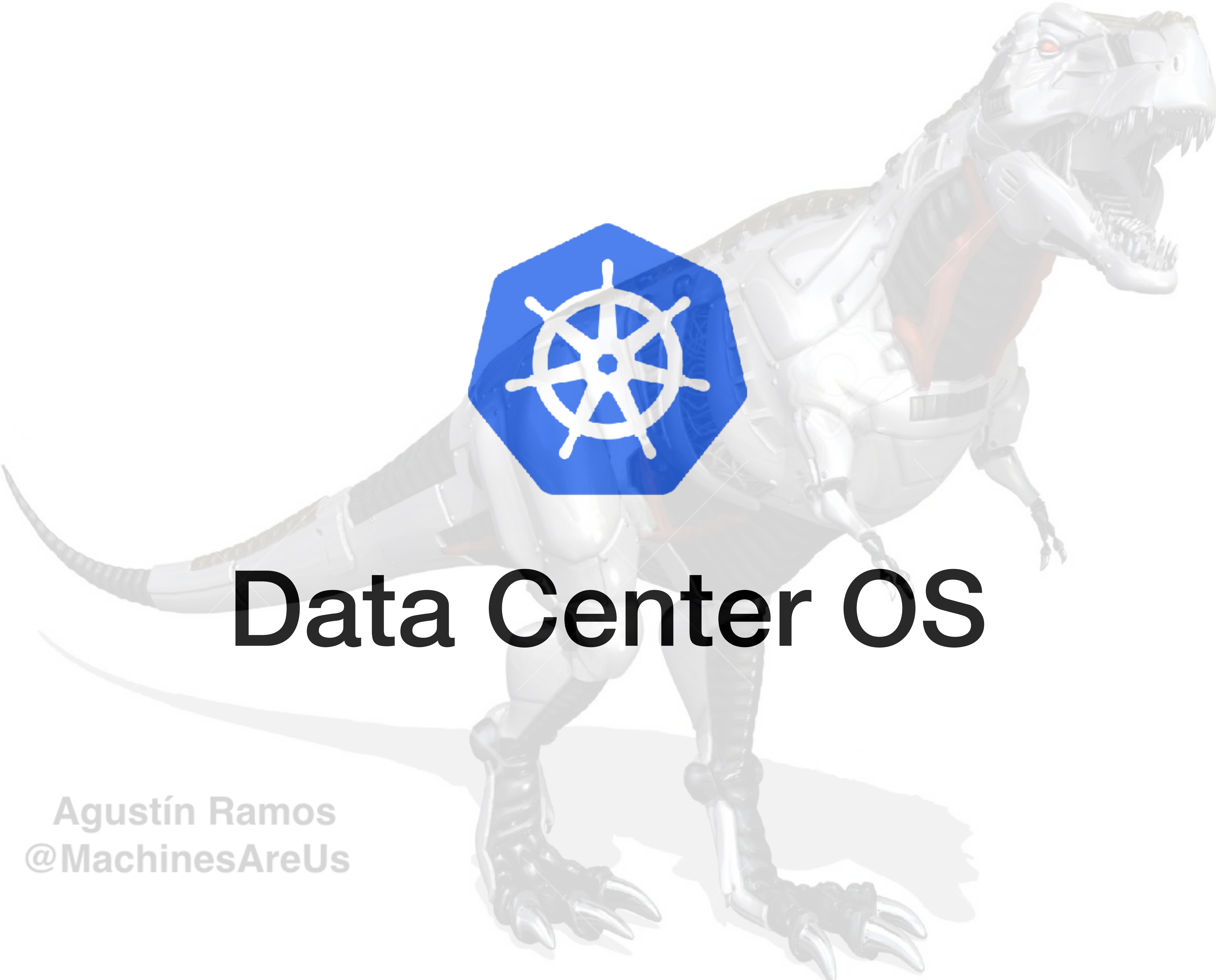
CoreOS

Agustín Ramos
@MachinesAreUs

Tectonic



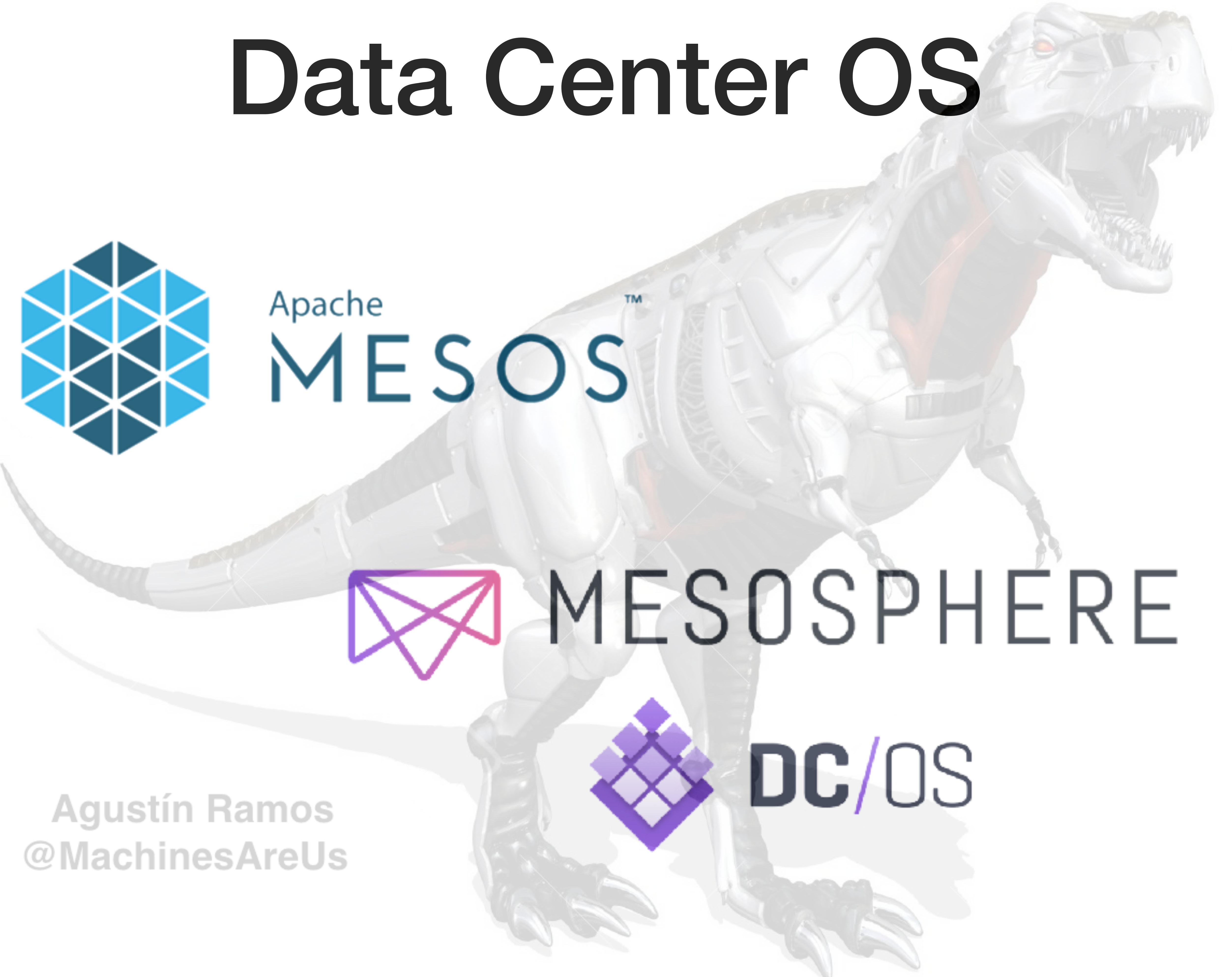
- “Fast and secure” deployments.
- Container Linux (distro downsized for containers).



Data Center OS

Agustín Ramos
@MachinesAreUs

Data Center OS

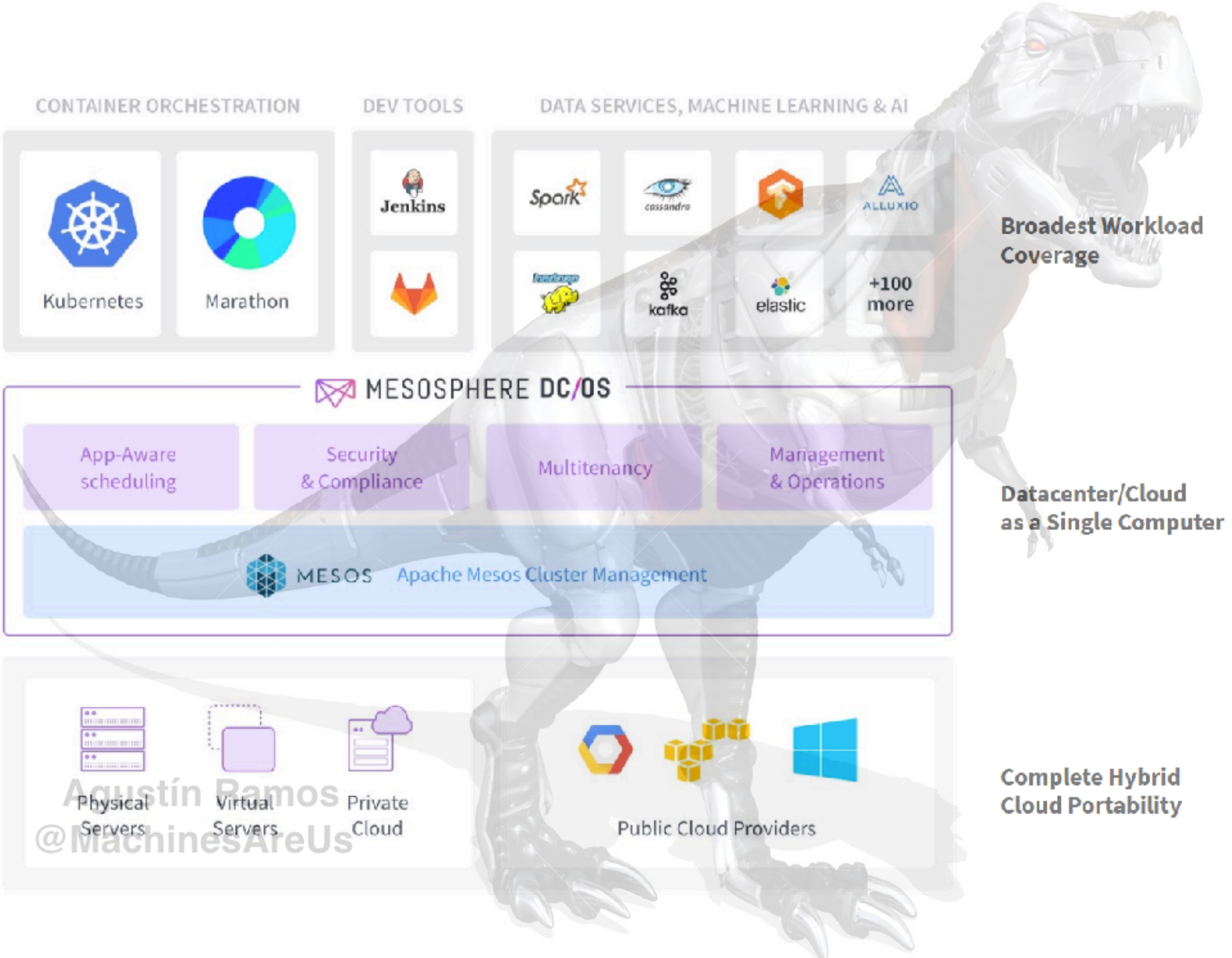


Apache MESOS™

MESOSPHERE

DC/OS

Agustín Ramos
@MachinesAreUs



CONTAINER ORCHESTRATION

Kubernetes

Marathon

DEV TOOLS

Jenkins

DATA SERVICES, MACHINE LEARNING & AI

Spark

cassandra

ALLUXIO

kafka

elastic

+100 more

Broadest Workload Coverage

MESOSPHERE DC/OS

App-Aware scheduling

Security & Compliance

Multitenancy

Management & Operations

MESOS Apache Mesos Cluster Management

Datacenter/Cloud as a Single Computer

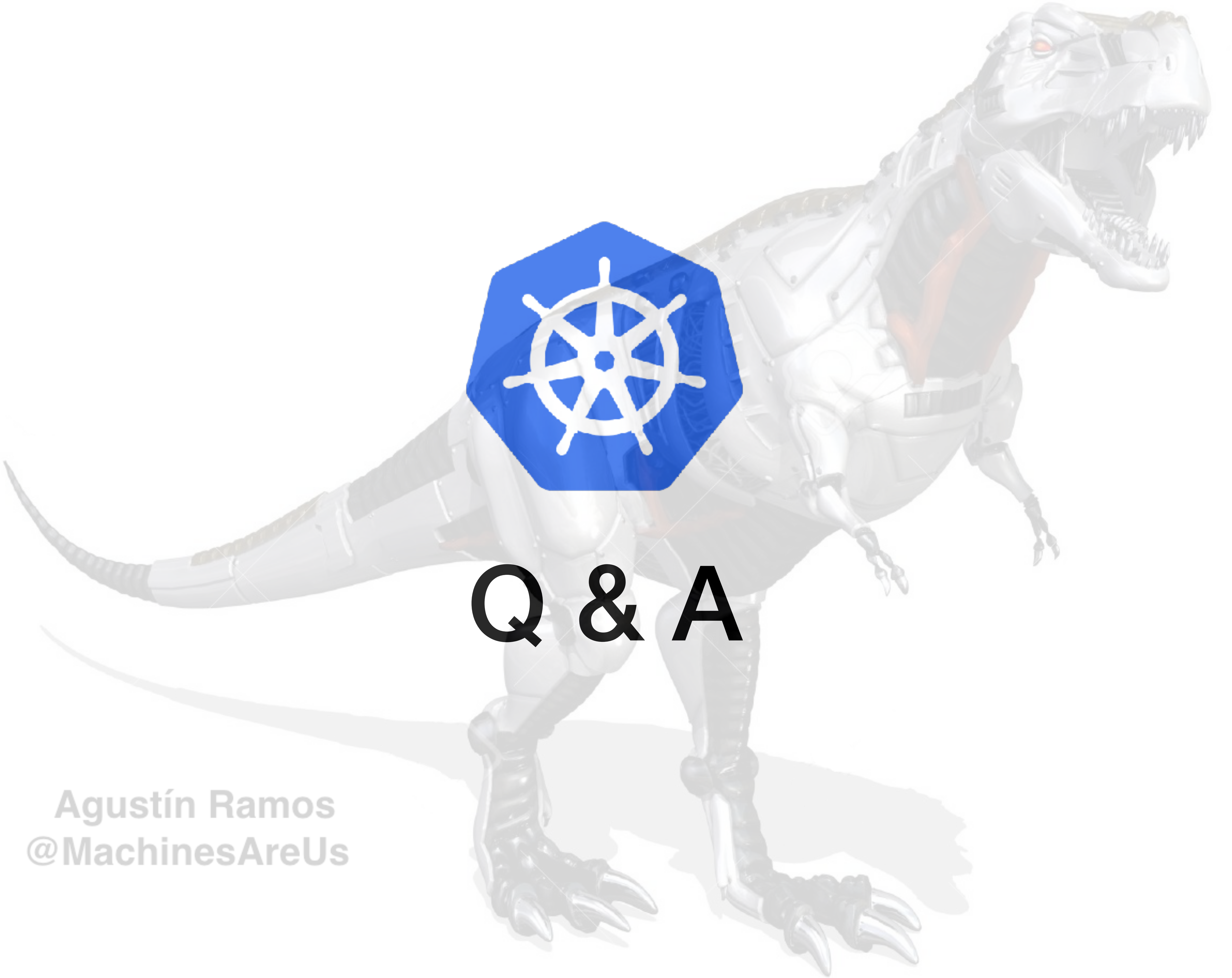
Physical Servers

Virtual Servers

Private Cloud

Public Cloud Providers

Complete Hybrid Cloud Portability



Q & A

Agustín Ramos
@MachinesAreUs