

@andresgz00

DESPLIEGUE DE APLICACIONES PARA ALTA DISPONIBILIDAD







Andres Gonzalez

Director of Research and Development at inQbation
Labs

About Me

- Desarrollador de Software.
- 15+ años desarrollo web.
- Cofounder Django Cali.
- Python/Django Evangelist.
- Emprendedor.
- Software developer at [swapps](#)
-  @andresgz00
-  andresgz

Agenda

- Requerimientos
- Consideraciones de diseño.
- Arquitectura para alta disponibilidad.
- Ejemplos de arquitecturas.

Diseño de la aplicación

Lo básico

- Evitar Consultas lentas a la BD.
- Evitar operaciones de lectura lenta (Archivos, API).
- La Caché es tu amiga.
- Evitar timeouts por procesamiento: Encoladores.
- Menor cantidad de peticiones a recursos (JS, CSS).
- Minify lo que más se pueda.

Consultas lentas

```
# :(
def check_for_me(self):
    users = Users.objects.filter(role='developer')
    for el in users:
        # This would execute a db query
        # each iteration
        prof = el.profile
        if prof.real_name is 'Andres':
            return True
    return False
```

```
# :)
def check_for_me(self):
    # This is 'lazy'
    users = Users.objects.filter(role='developer')
    users = users.select_related('profile')
    for el in users:
        # This would NOT execute an additional db query
        prof = el.profile
        if prof.real_name is 'Andres':
            return True
    return False
```

Es suficiente?

12Factor.net

//

- Use **declarative** formats for setup automation.
- Have a **clean contract** with the underlying operating system, offering **maximum portability** between execution environments;
- Are suitable for **deployment** on modern **cloud platforms**,
- **Minimize divergence** between development and production, enabling **continuous deployment** for maximum agility;
- And can **scale up** without significant changes to tooling, architecture, or development practices.



En pocas palabras...

Codebase: Código base seguido por Control de versiones, Muchos deployments

Dependencias: Declarar dependencias explícitamente

Configuración: Almacenar configuración en el entorno

Backing Services: Treat backing services as attached resources

Desarrollar, liberar, ejecutar: Separar las etapas de desarrollo.

Procesos: Ejecutar aplicaciones como uno o más procesos "stateless".

Port binding: Servicios via "port binding"

Concurrencia: Scale out via the process model

Desechabilidad: Fast startup, graceful shutdown

Dev/prod parity: Desarrollo, staging, y producción iguales.

Escenario 1:

- 1 Servidor
- Guardando imágenes en el sistema de archivos local.
- Obtienes mucho tráfico
- Qué haces?

Ni siquiera pienses en guardar cosas en el servidor de la aplicación!



Por qué?

- Escalamiento Vertical no es una buena solución a largo plazo
- Redundancia: Realmente queremos todo en un único servidor
- "State is your worst enemy"
- "Kill all the servers!"
- Freedom
- Containers

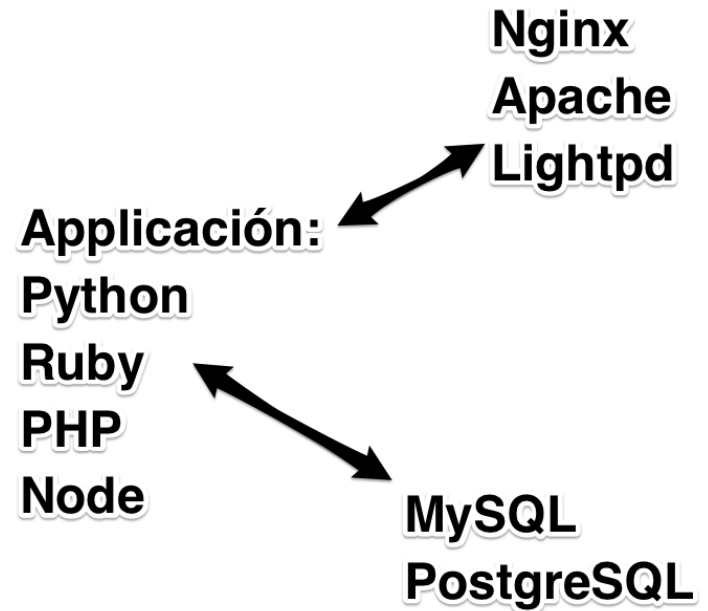
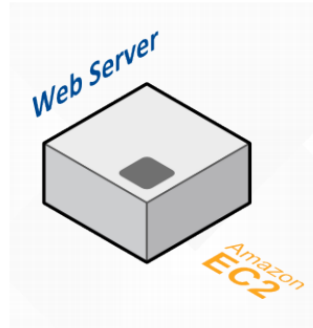
Construyamos una aplicación

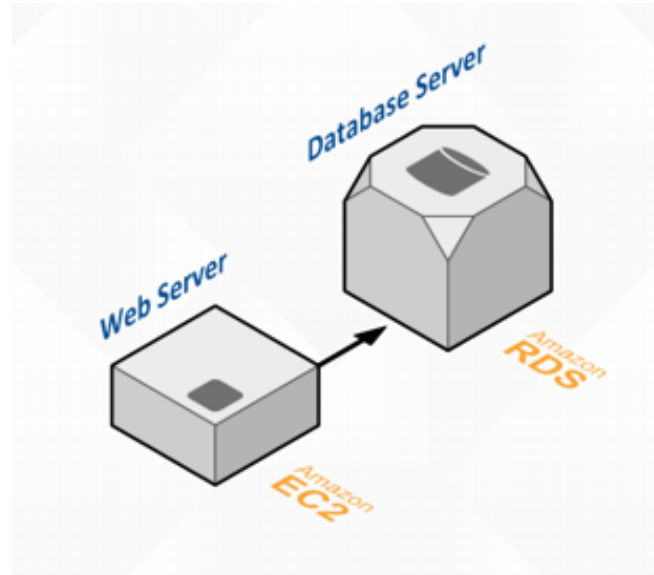
Altamente disponible!

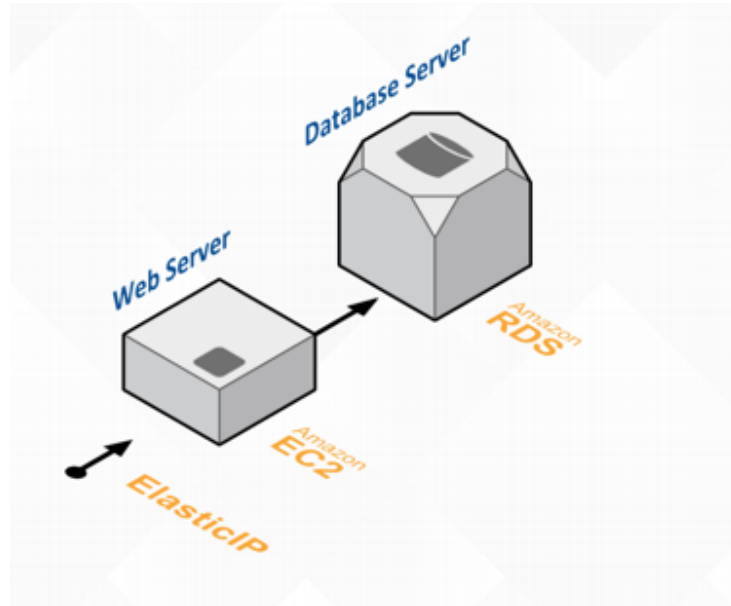
Consideraciones

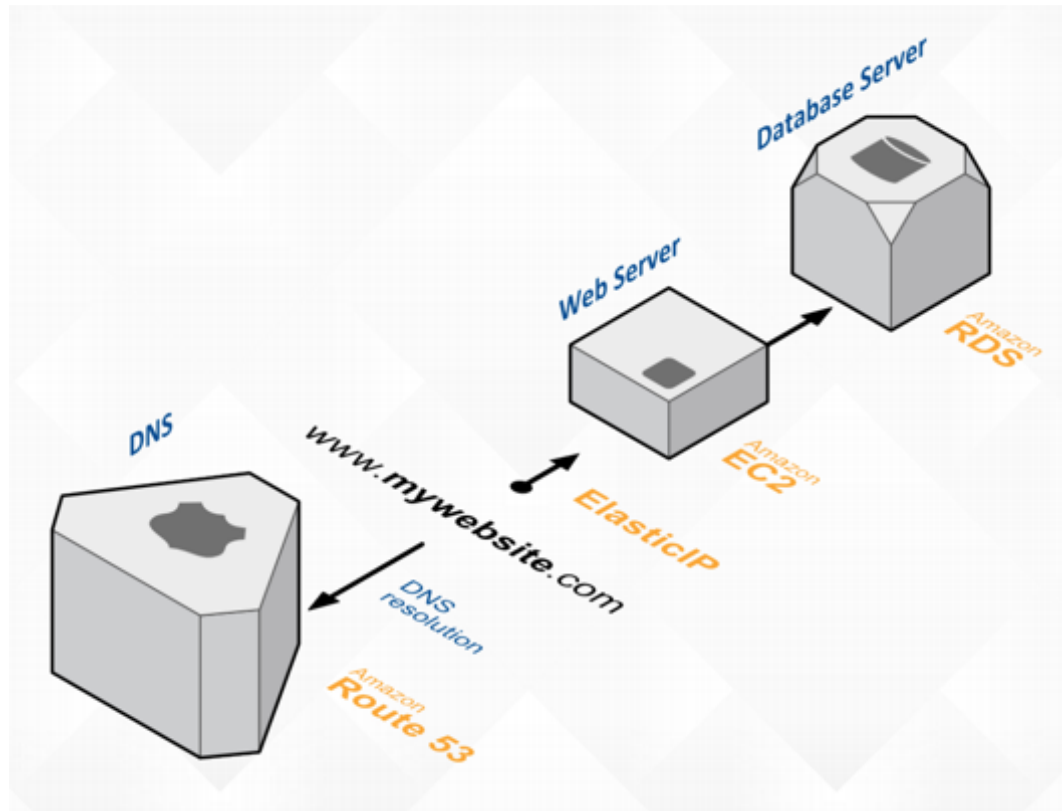
1. Diseñar para fallas.
2. Múltiples Zonas de disponibilidad.
3. Escalamiento.
4. Auto-Curación
5. Desacoplamiento.

Single Server









1. Diseñar para fallas

"Everything fails all the time"

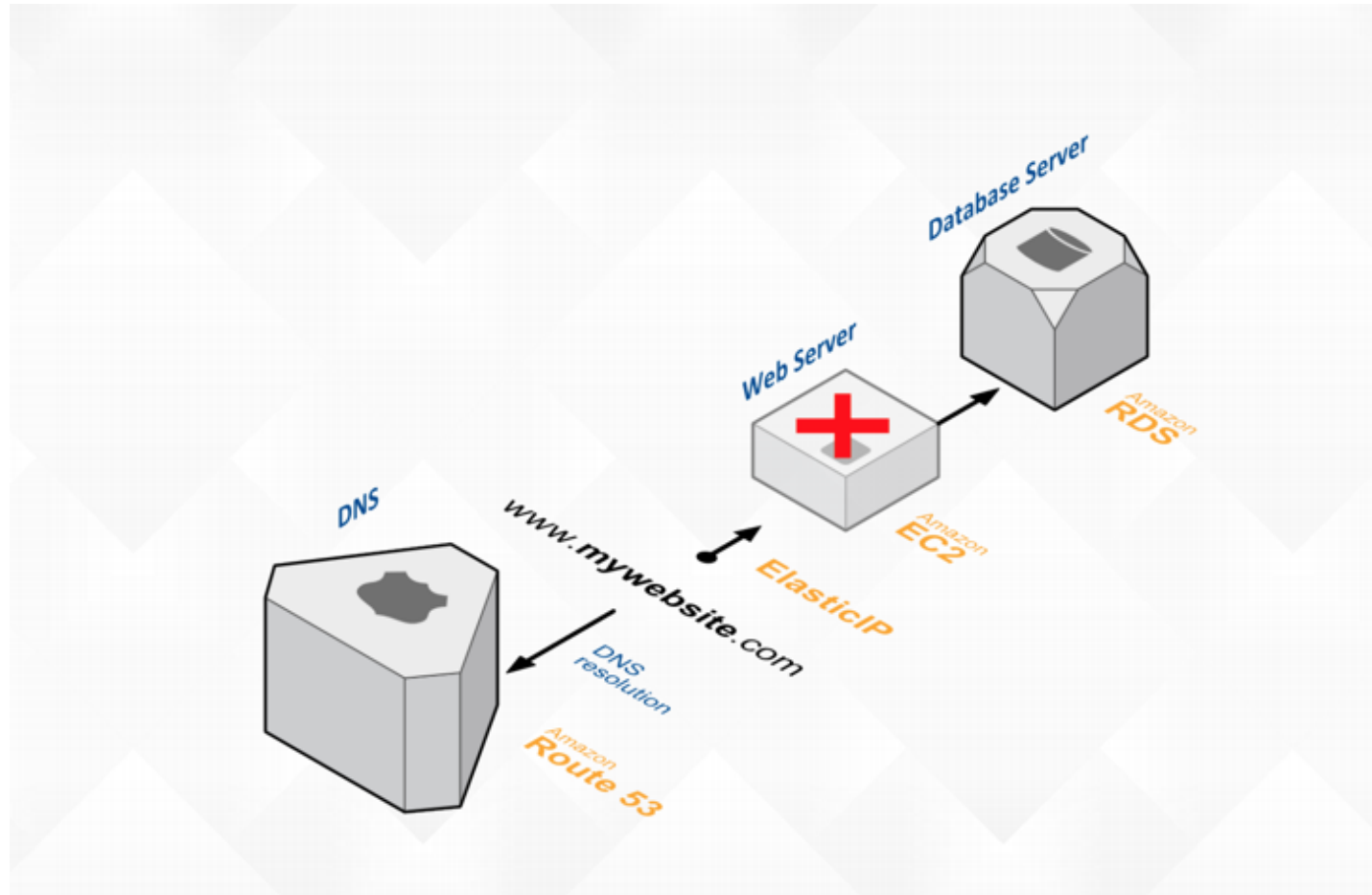
Werner Vogels
CTO de Amazon

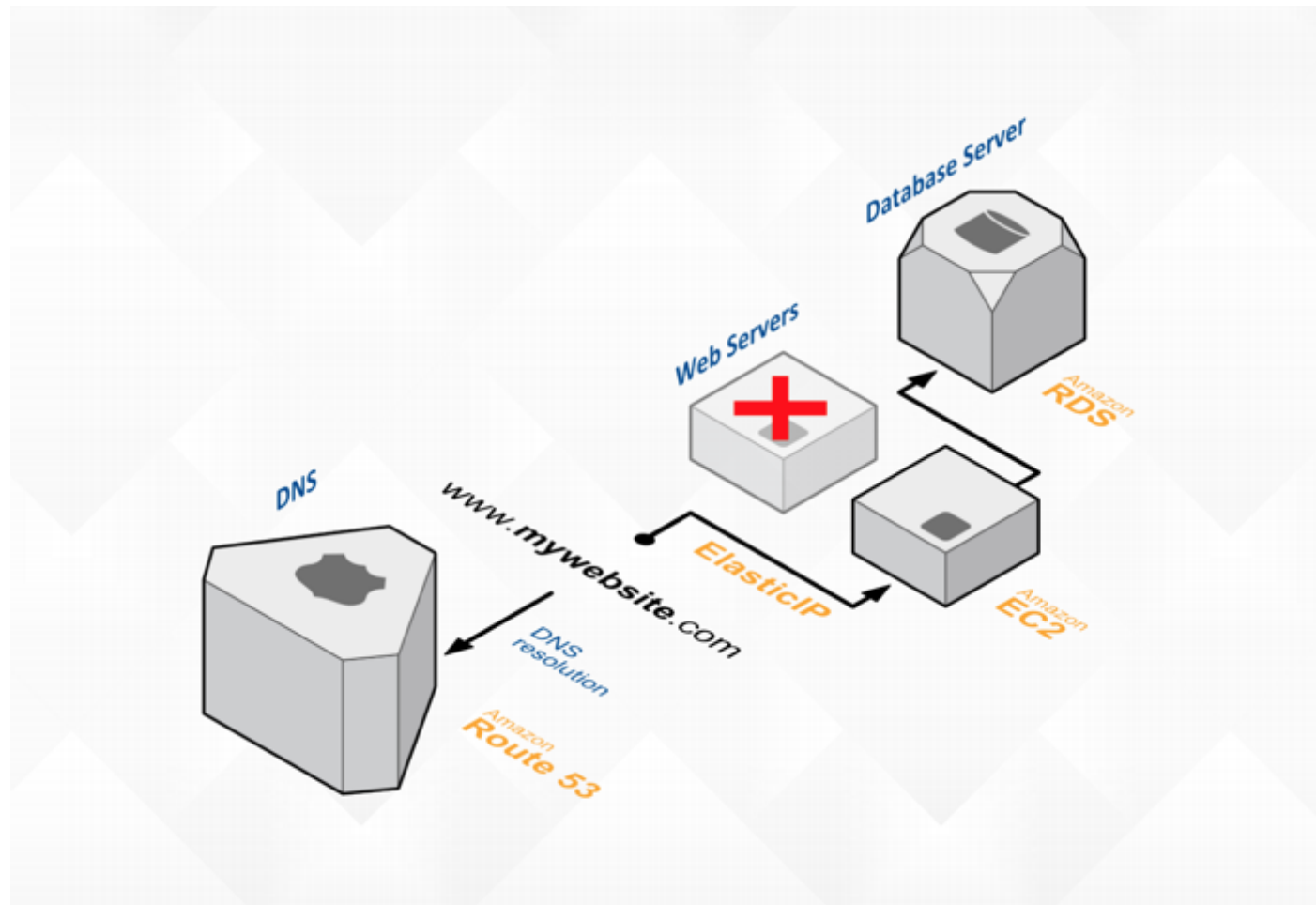
Consideraciones

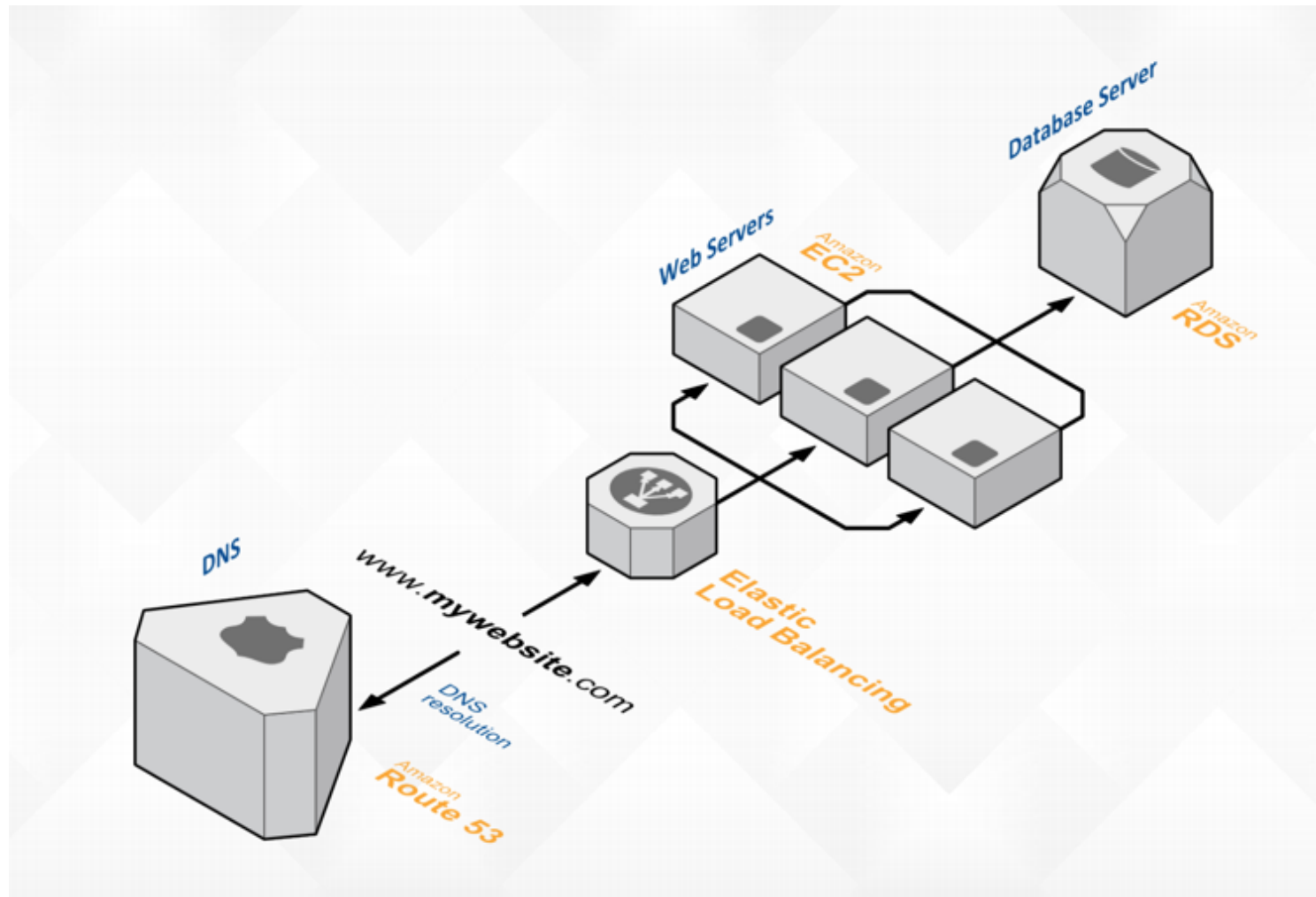
- Evitar los puntos de falla únicos.
- Asumir que todo falla.

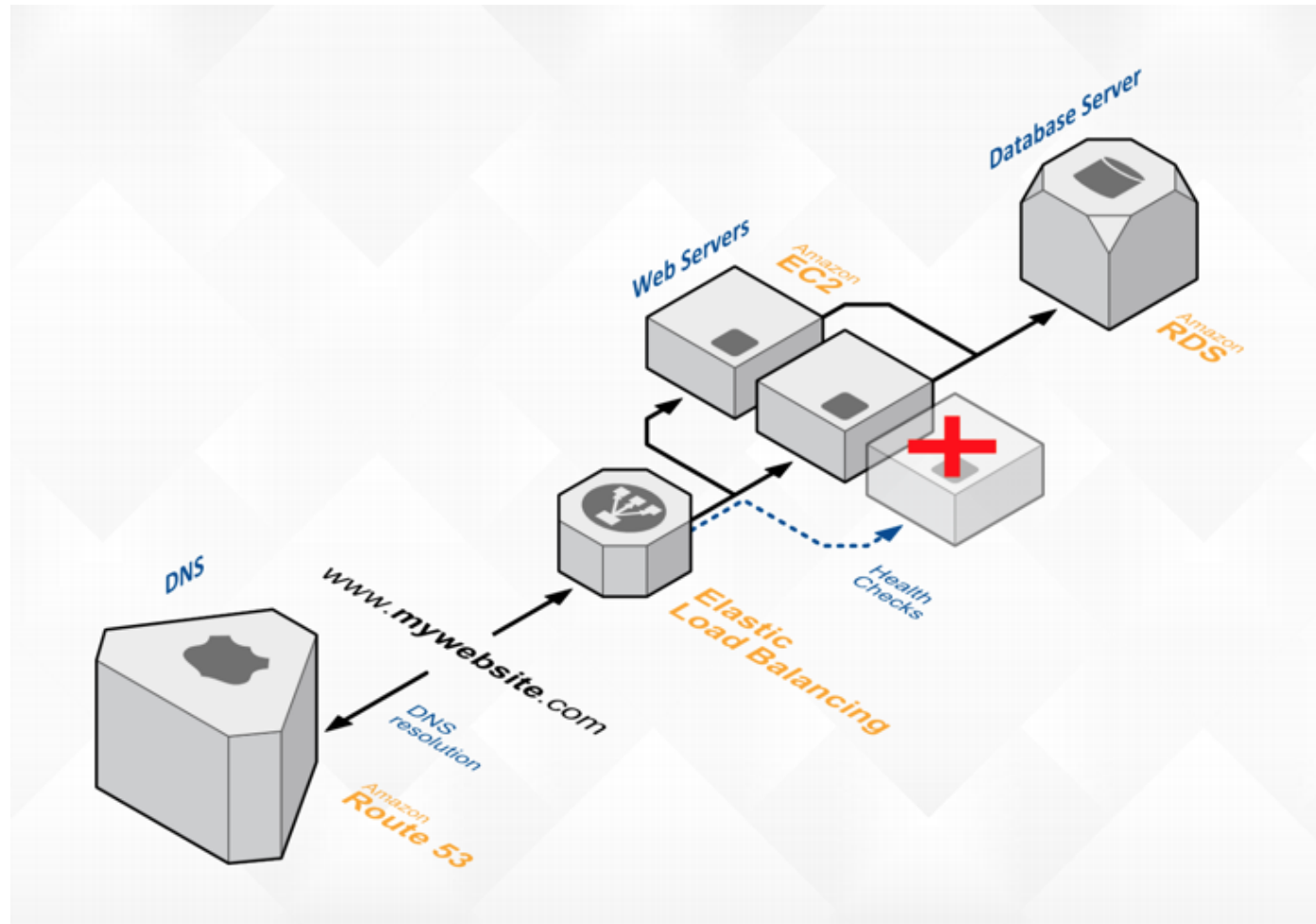
META

Las aplicaciones deben continuar
funcionando

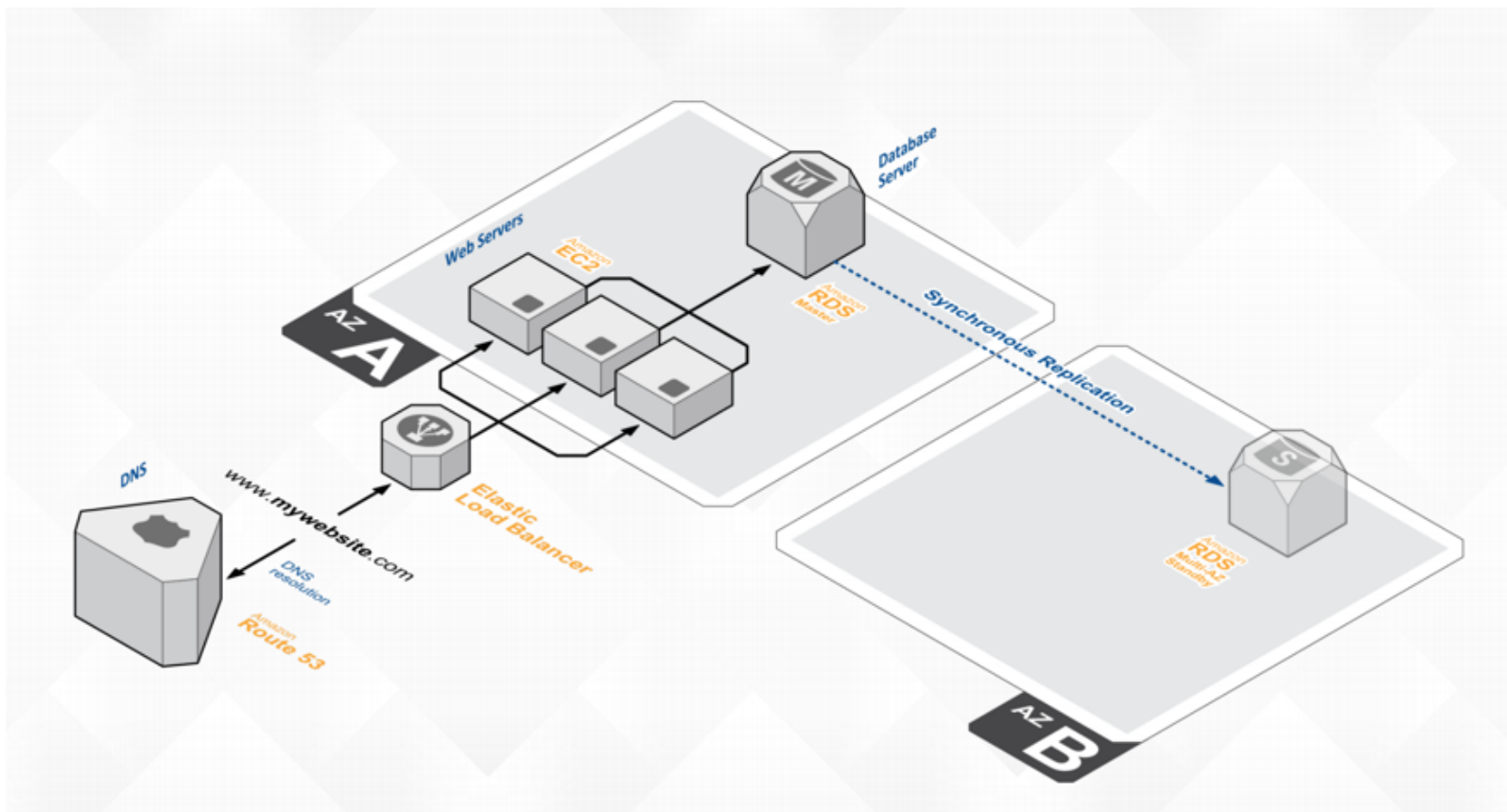


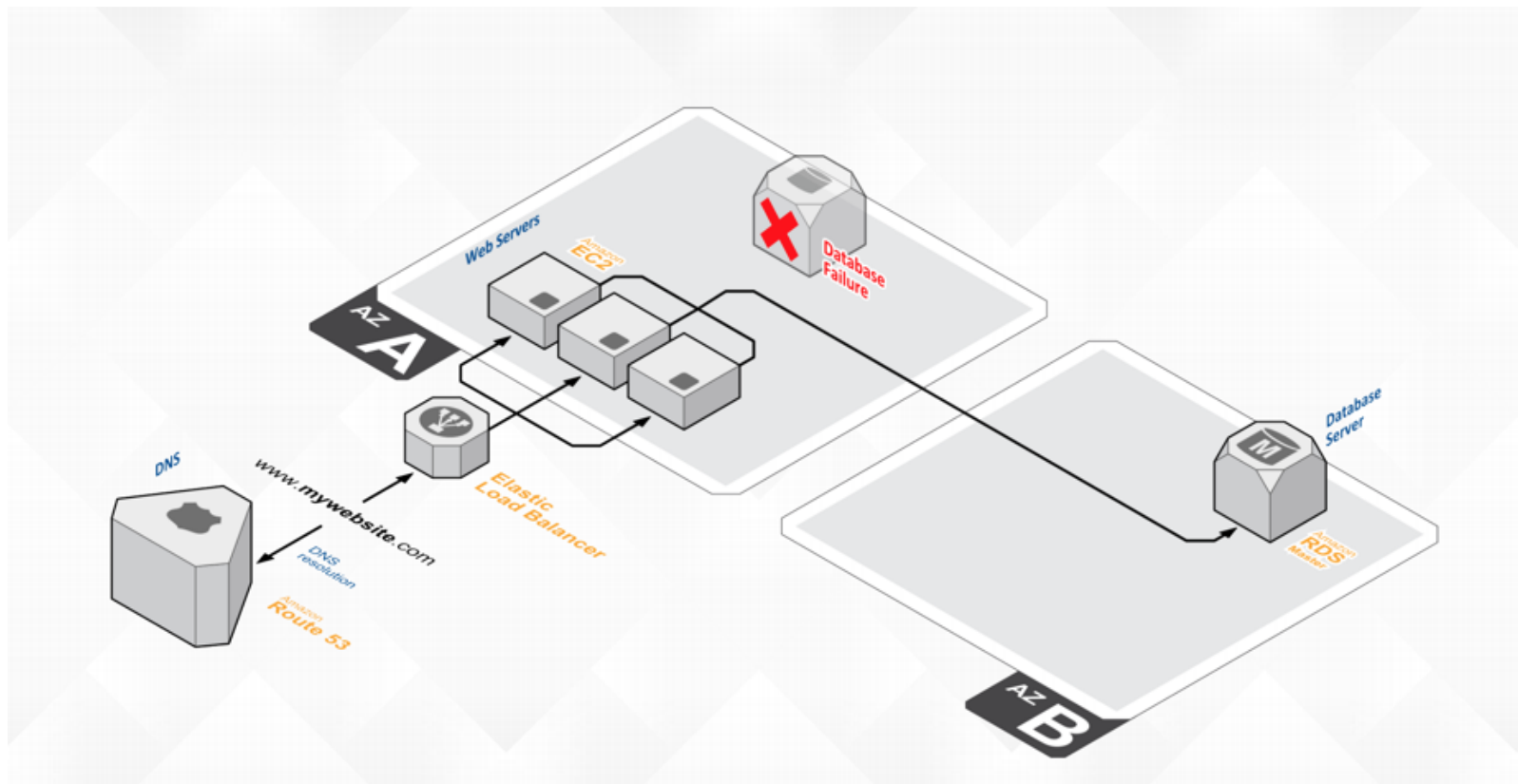




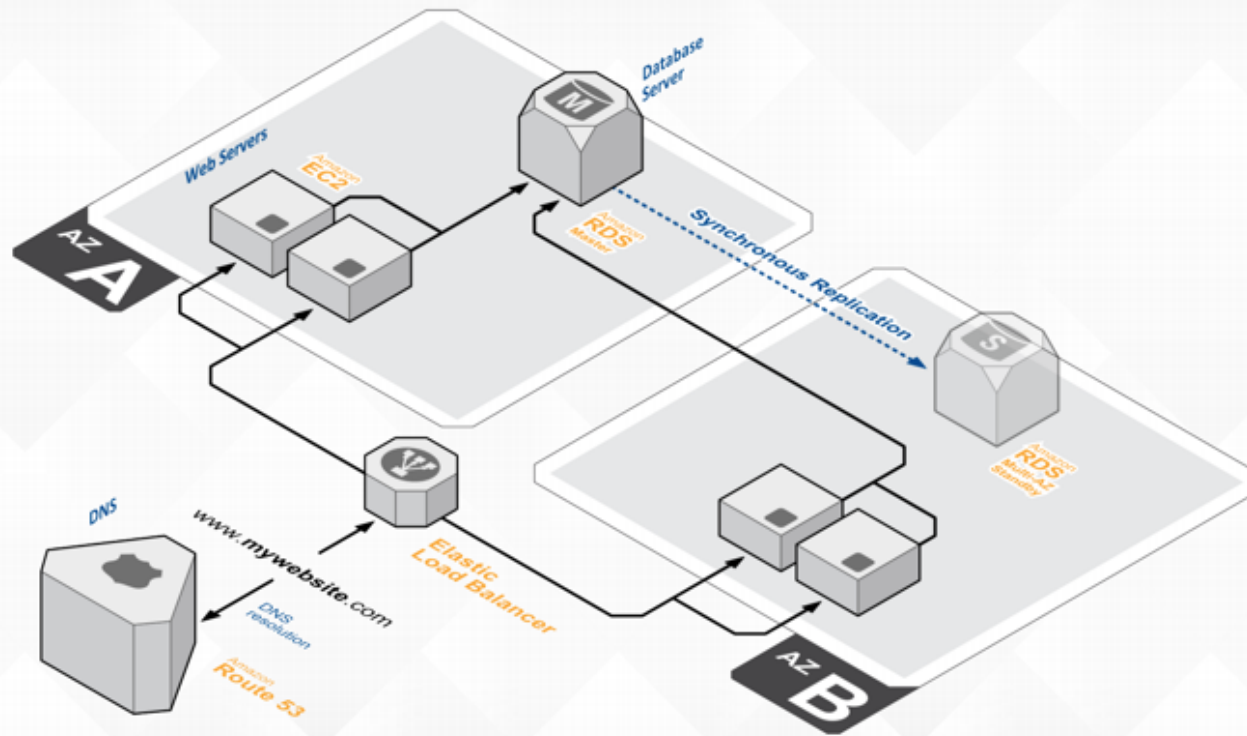


2. Múltiples Zonas de disponibilidad

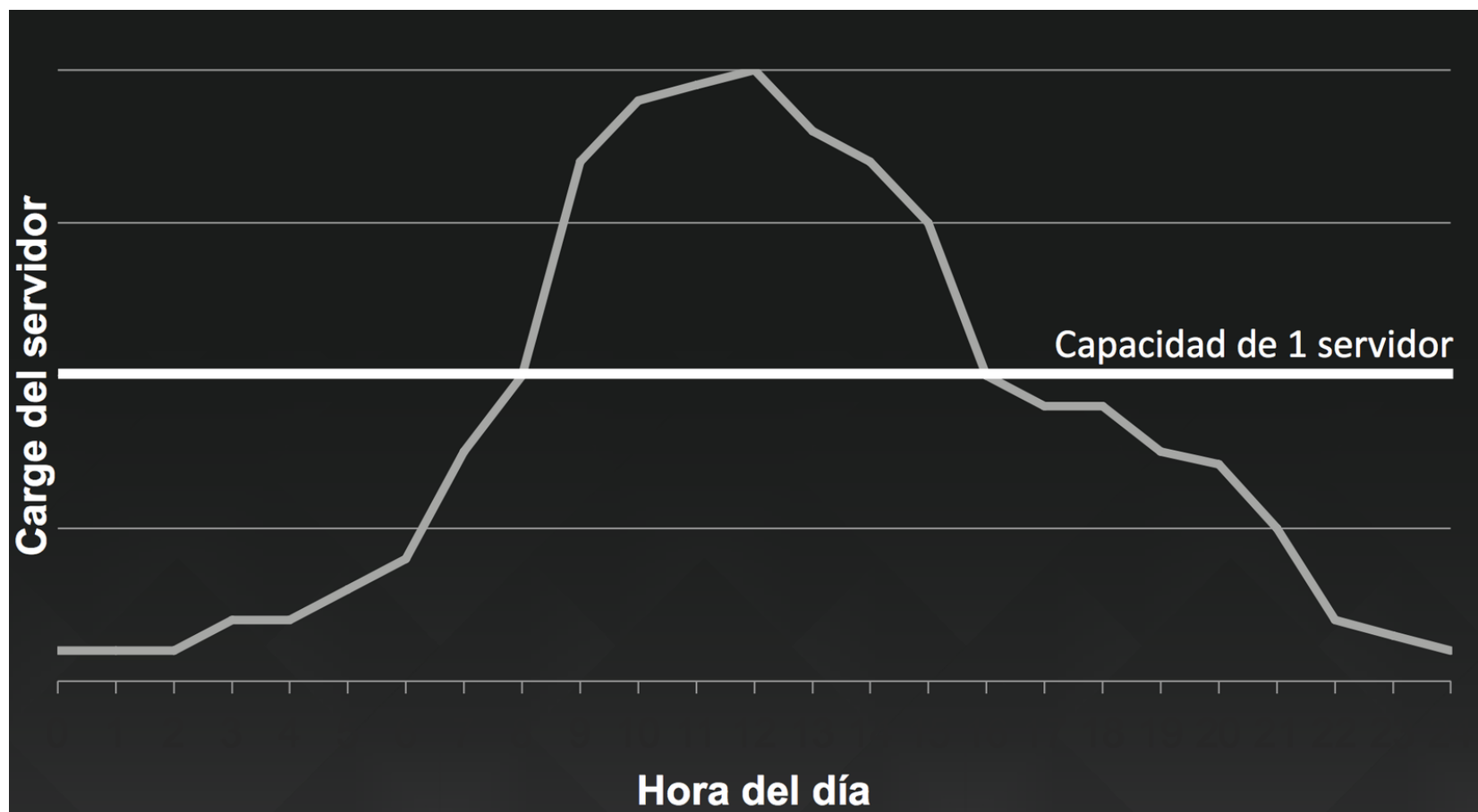




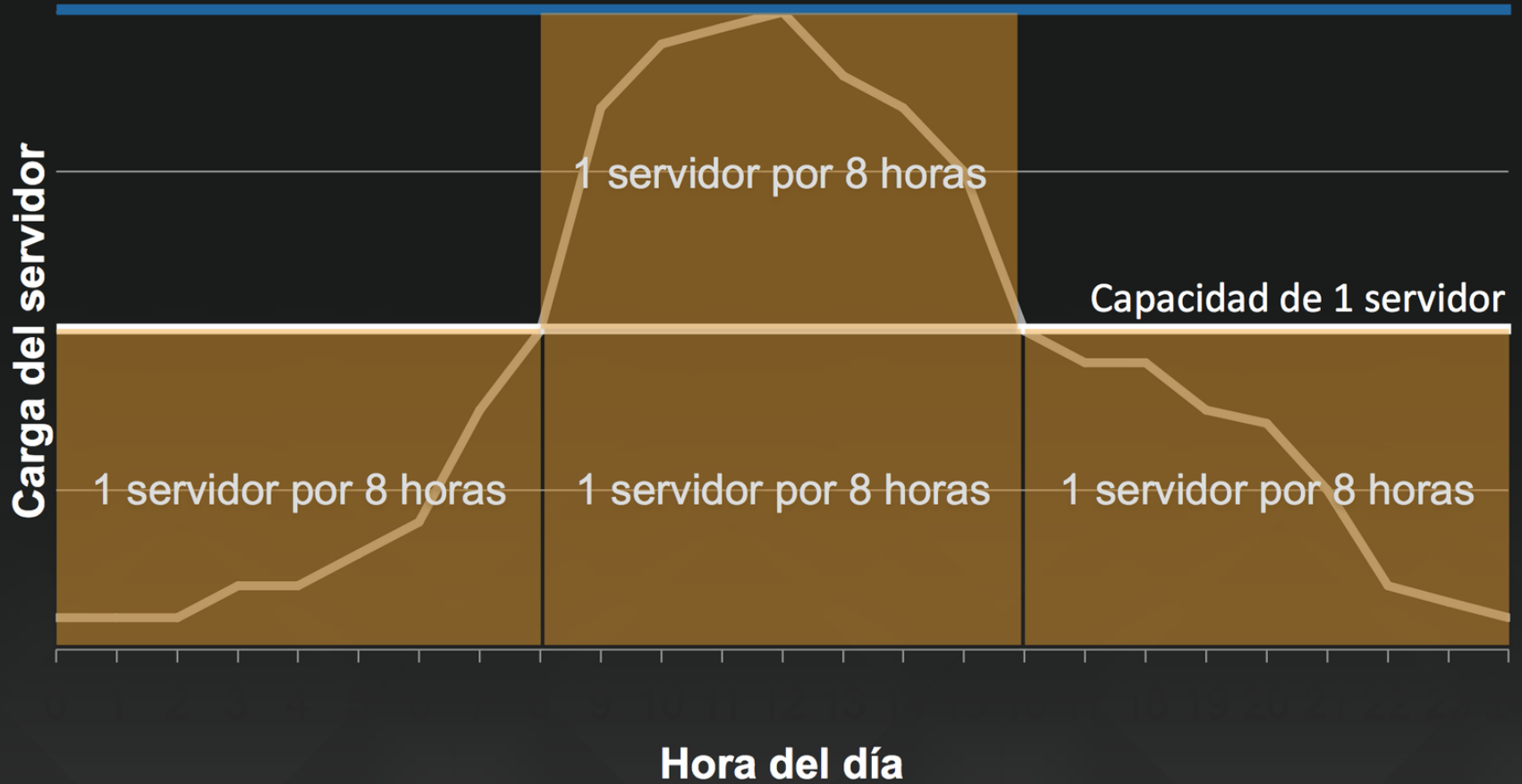
Multiple Availability Zones



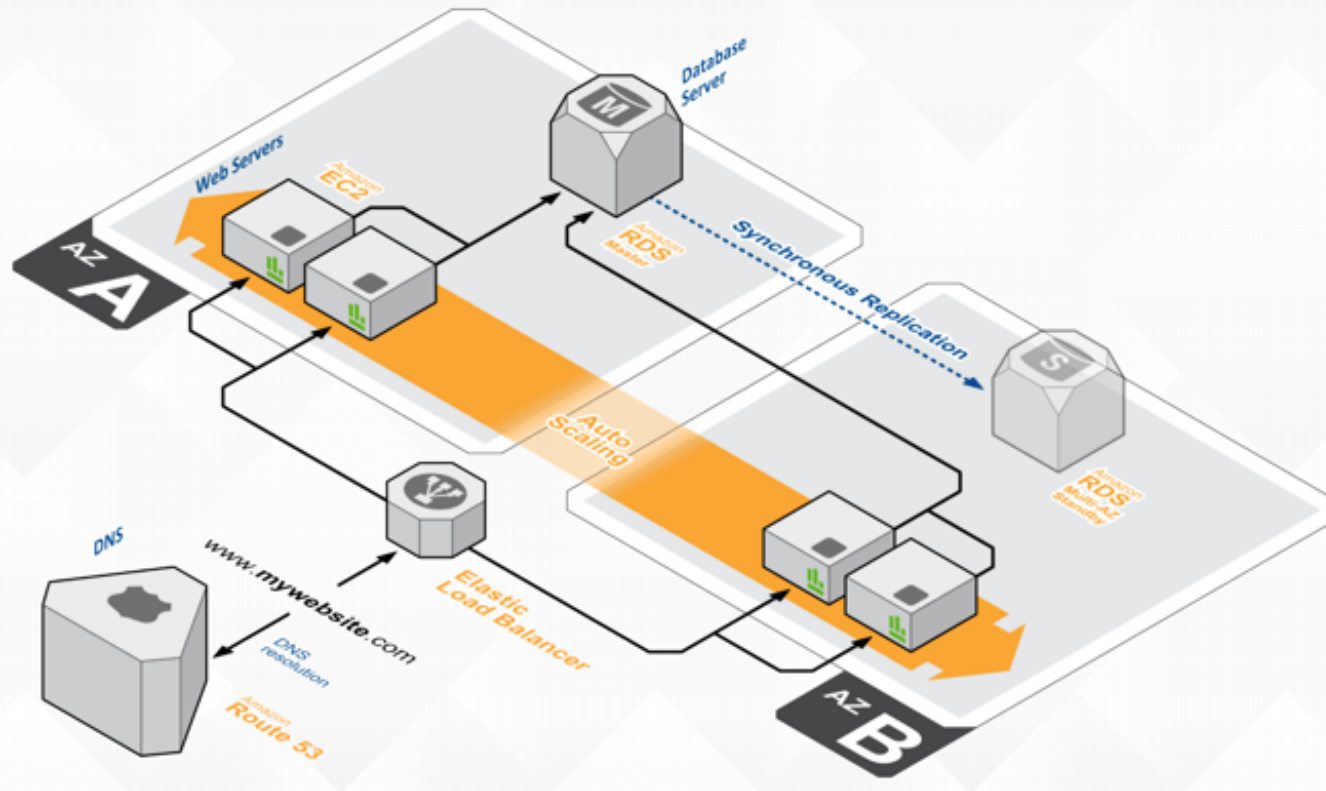
3. Escalamiento

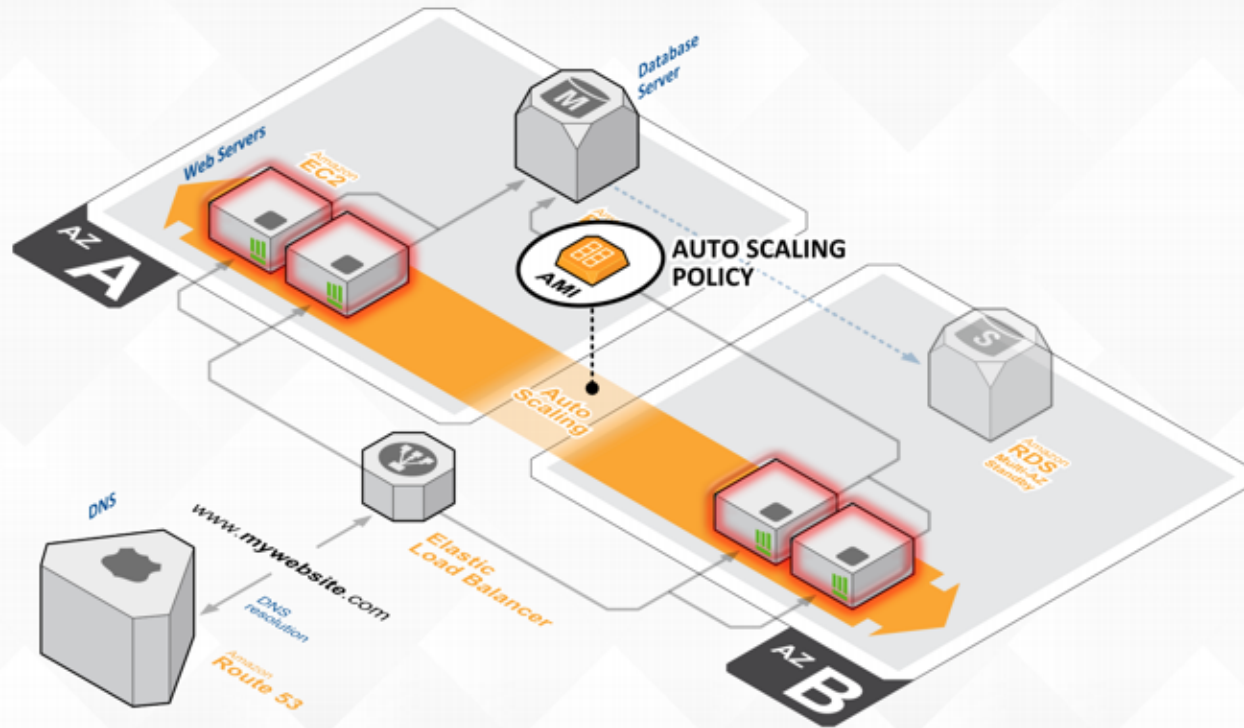


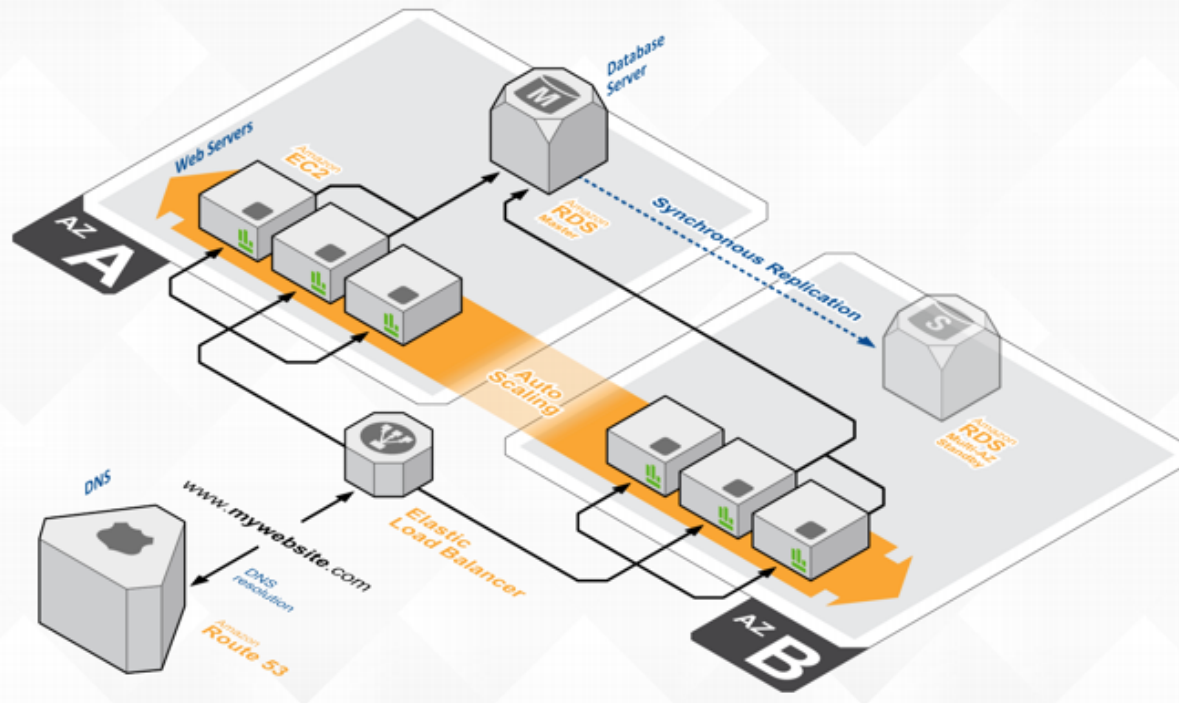
Capacidad requerida en esquema tradicional



AutoEscalamiento





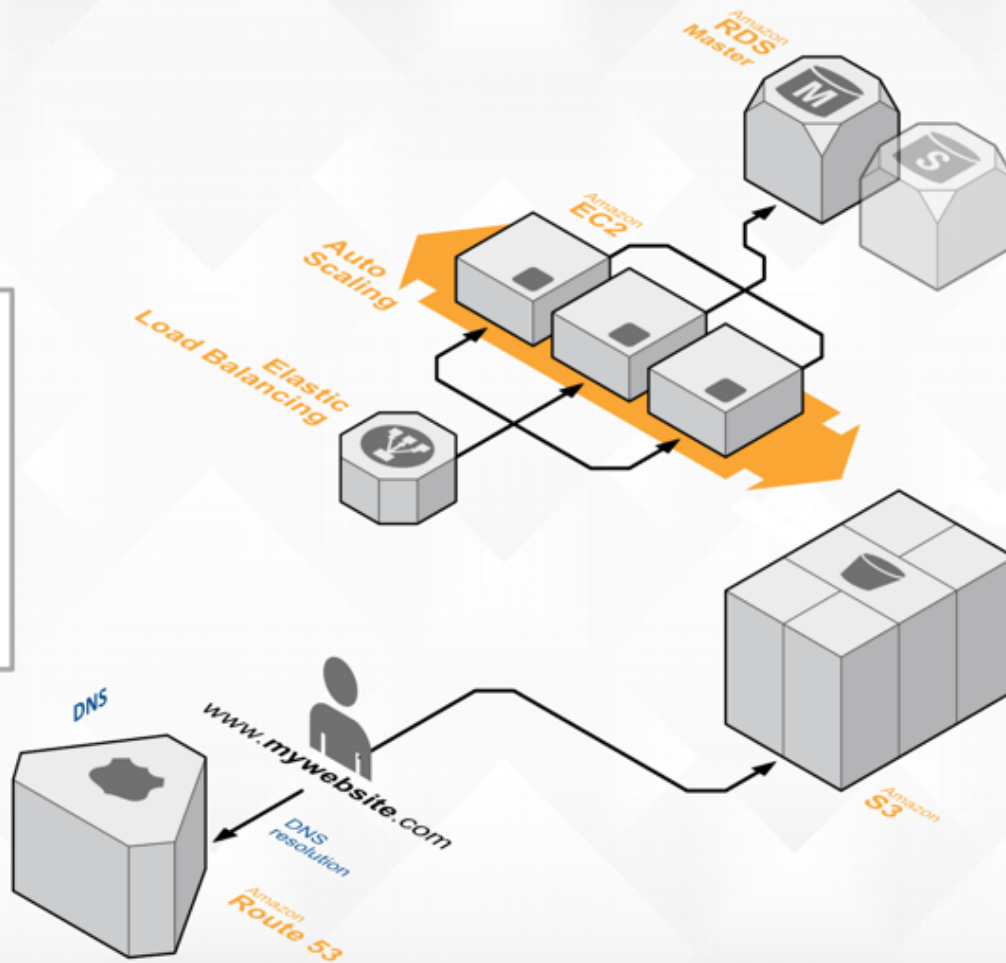


3. Auto-curación

RECORD SETS

Alias target to:
elastic_load_balancer
weight=0

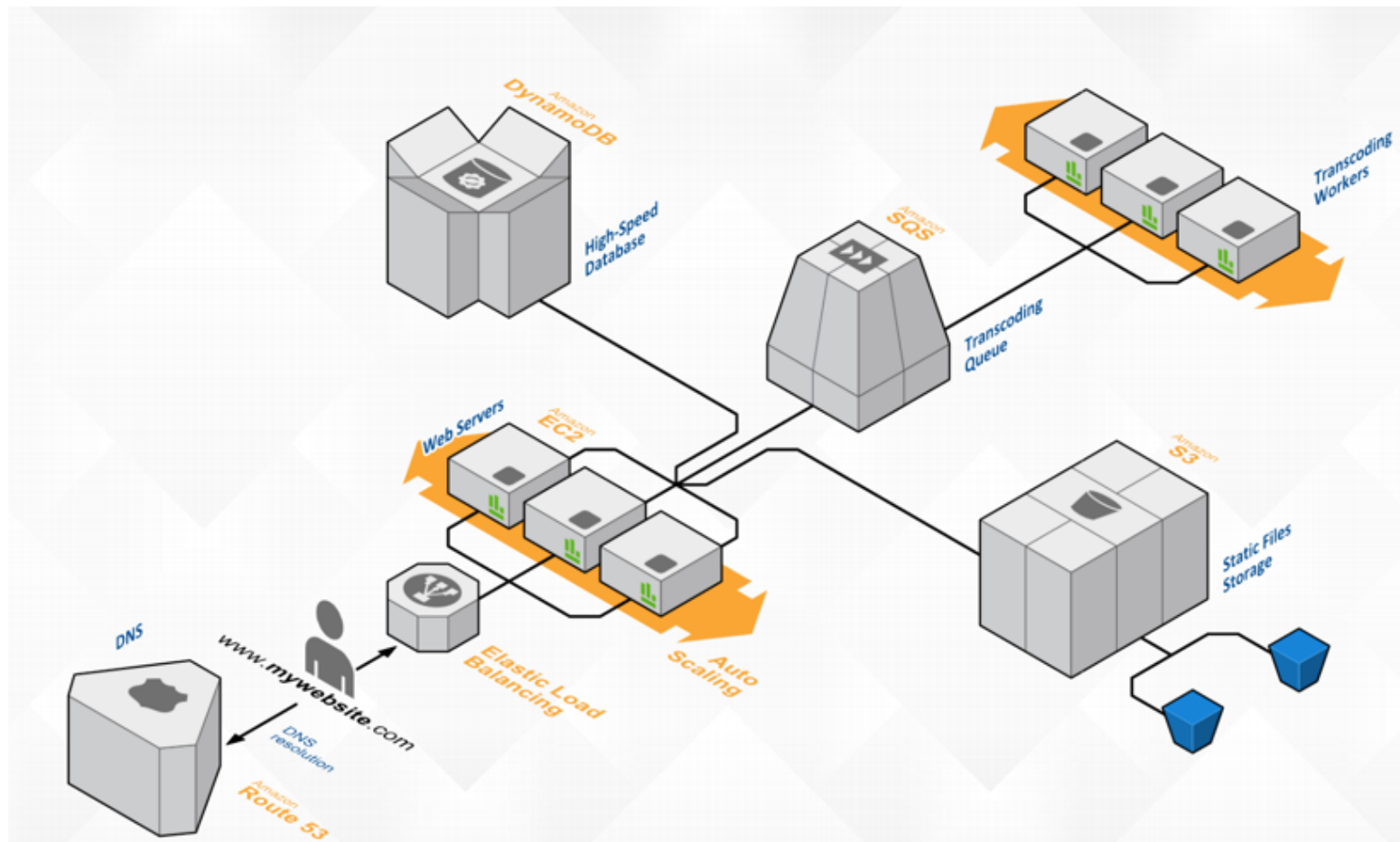
Sitio web en S3
weight=255



5. Desacoplamiento

Un sistema desacoplado es más fácil de escalar

Entre menos acoplados estén los componentes, más fácilmente se pueden escalar y más tolerantes a fallas se vuelven



Gracias!

@calidevco

Referencias

- <http://aws.amazon.com/es/architecture/>
- <http://slides.com/juliangindi/deck#/10>
- <http://es.slideshare.net/AmazonWebServices/presentations>