# WORDPRESS AND CONTAINERS

ALAN LOK

# WHAT ARE CONTAINERS

## Self-contained

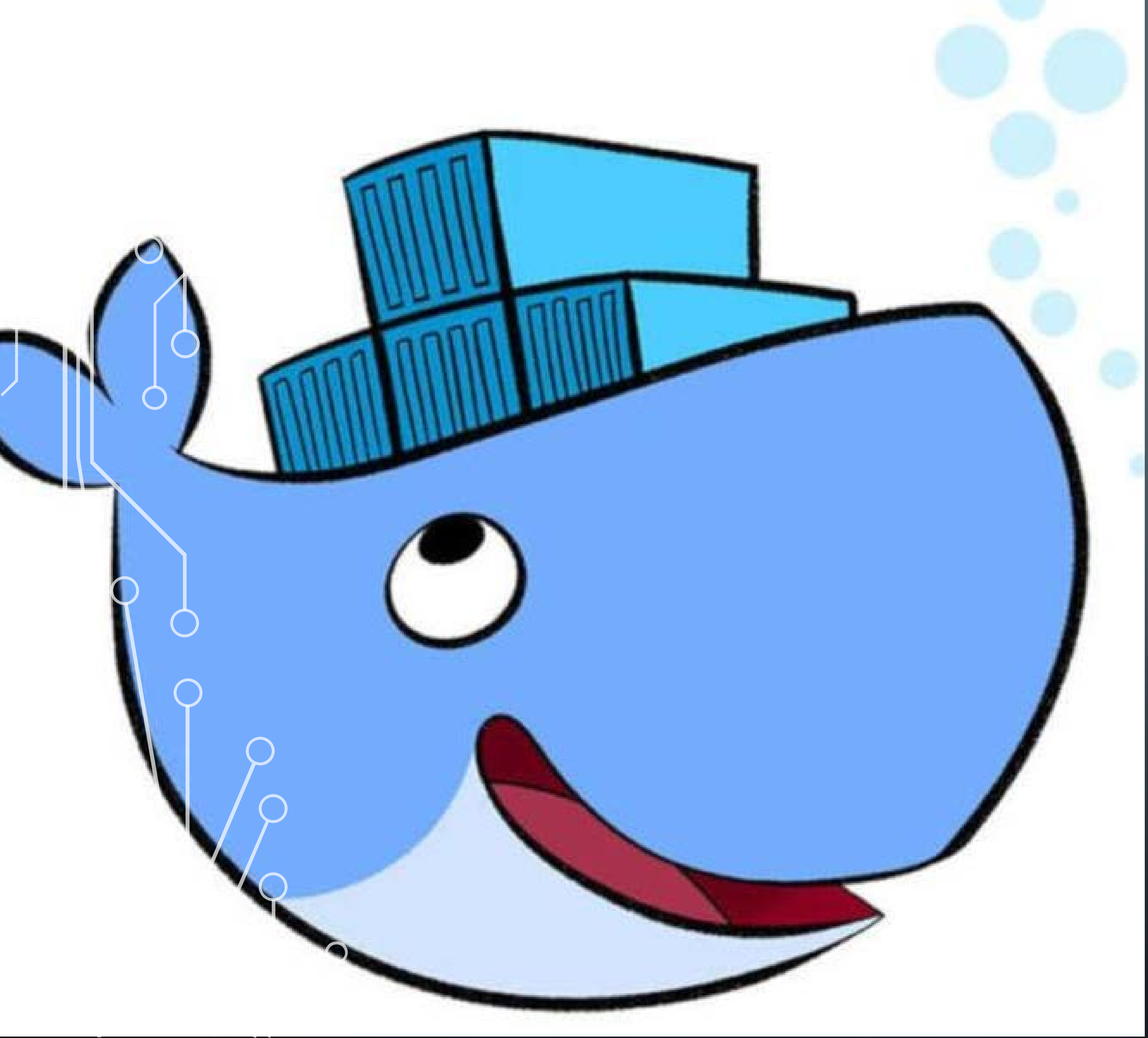
- Single file with system library and application code
- Lightweight

## Immutable

- Portable, predictable, repeatable infrastructure

## Scalable

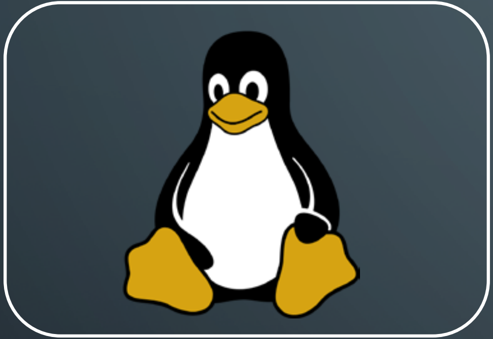
- Need more power? Add more containers*



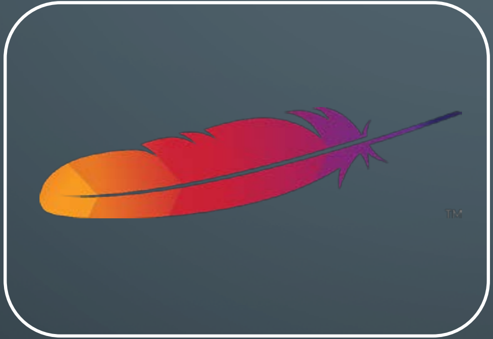
# CONTAINER MANAGEMENT

- Containers need a home (or vessel) to carry its load
- Your solution will likely need different containers (database, app, API?)
- Lots of management or orchestration tools to pick
  - Kubernetes, Docker Swarm, Mesos, Openshift, ECS to name a few

# THE LAMP STACK



Linux

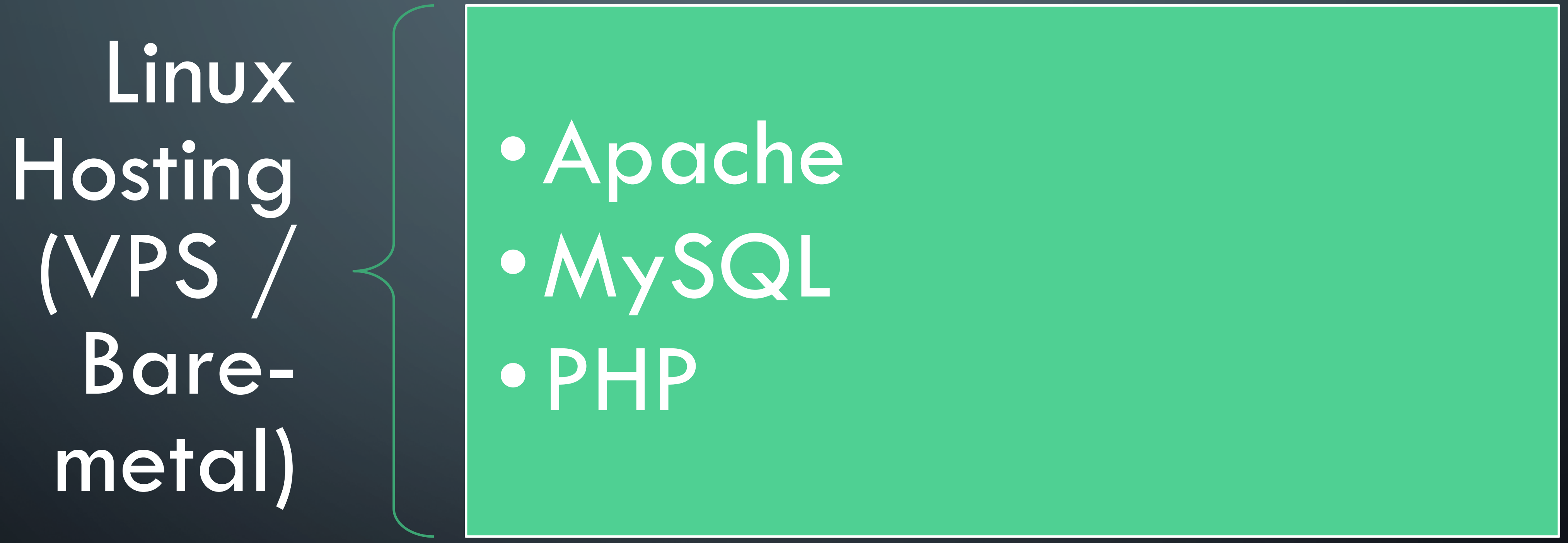


Apache

PHP

MySQL

# THE “ALL-IN-ONE” WORDPRESS LAMP STACK



Linux Hosting (VPS / Bare-metal)

- Apache
- MySQL
- PHP

# WORDPRESS IN DOCKER TERMS

Web & PHP

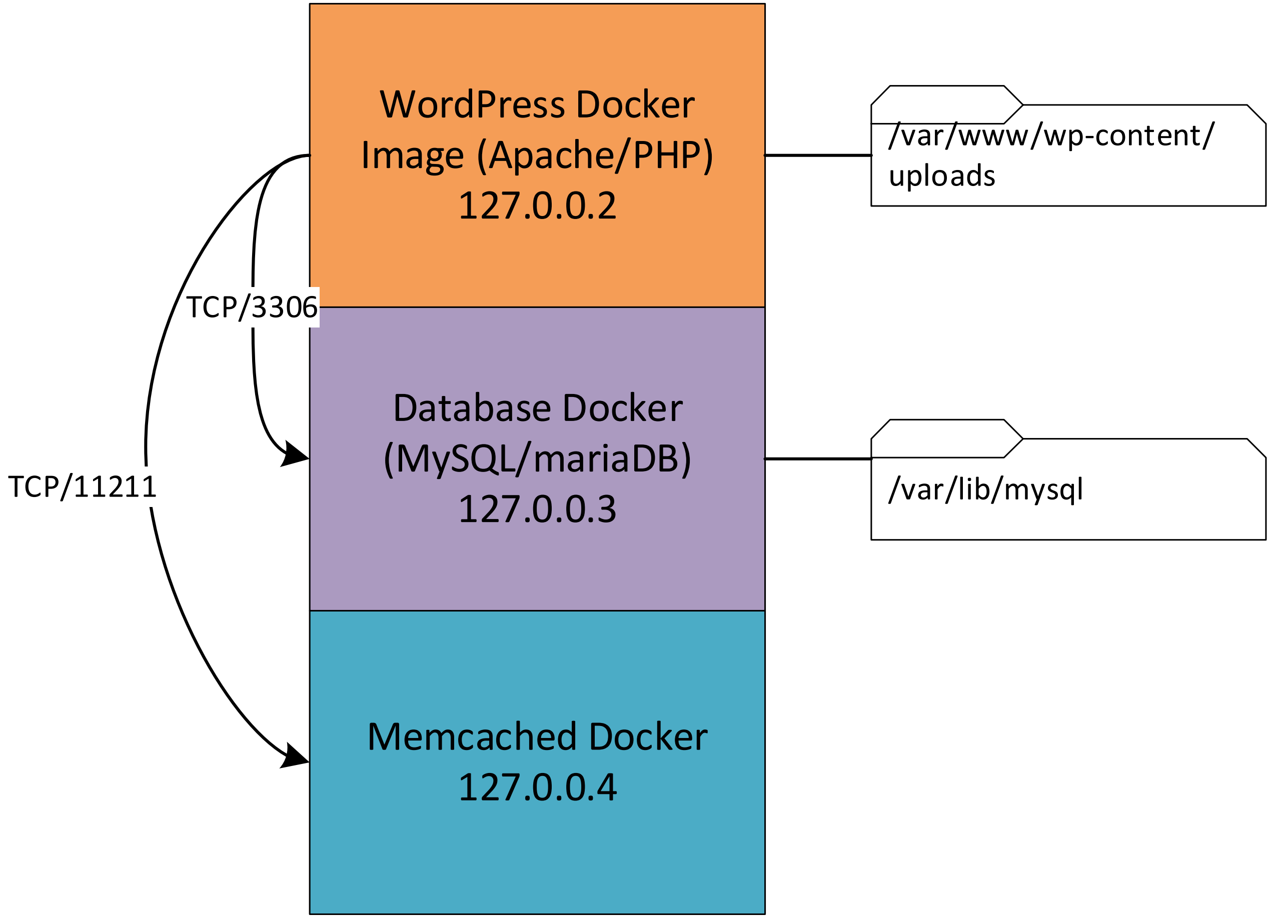
WordPress

MySQL Server

MySQL

# WHY START WITH DOCKER?

- Docker is a very *popular* container format
- Immutable infrastructure
  - Your application and its environment will be consistent and run similarly - whether it's with developer Jane, John or Janice
- Hosting mobility
  - Deploy with any provider that takes an Docker image (including AWS, Azure, and Google)
- Separating code and data
  - You can mount data to a data volume or path


WordPress Docker Image (Apache/PHP) 127.0.0.2
Database Docker (MySQL/mariaDB) 127.0.0.3
Memcached Docker 127.0.0.4
TCP/3306
TCP/11211
/var/www/wp-content/uploads
/var/lib/mysql

# A SIMPLE DOCKERIZED WORDPRESS VPS

- Single container host with shared file mounts for user data
- Each layer can change as you need to update versions of middleware or code
- This is possible with any VPS or development workstation

# DOCKER DEMO

Step 1 – pull Docker images

docker pull wordpress

docker pull mysql

Step 2 – create **stack.yml** (see right)

Step 3 – run the stack

```
docker stack deploy -c styack.yml wordpress
```

```
version: '3.1'

services:

  wordpress:
    image: wordpress:latest
    ports:
      - 8080:80
    environment:
      WORDPRESS_DB_PASSWORD: password
    volumes:
      - wpcontent:/var/www/html/wp-content

  mysql:
    image: mysql:latest
    environment:
      MYSQL_ROOT_PASSWORD: password
    volumes:
      - wpdb:/var/lib/mysql

volumes:
  wpdb:
    driver: local
  wpcontent:
    driver: local
```

# HOW ABOUT

- Making tweaks to your docker image?

```
docker container ls
docker exec -i -t <id>
/bin/bash
```

- Shutting down the deployment?
- Cloning data / editing

```
docker cp <source> <dest>
```

- Save docker image?

```
docker container commit
<id> <rep>:<tag>
```

# BUT WHAT ABOUT?

## HIGH AVAILABILITY

It can’t be down at all!

## SCALING

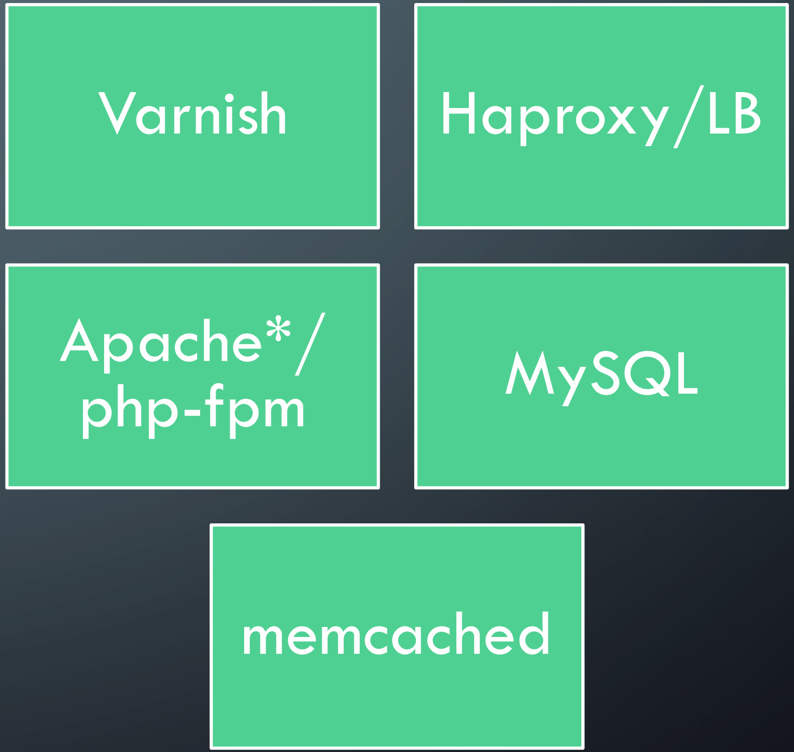
My WordPress plugin needs lots of processing power

## ORCHESTRATION

It’s *complicated*…

# DOCKER ORCHESTRATION TO THE RESCUE

- Break your layers to more manageable containers
- Use orchestration to handle service discovery and expansion
- Have more than 1 container hosts
- Can mitigate some single points of failure with LB or RR-DNS



Varnish

Haproxy/LB

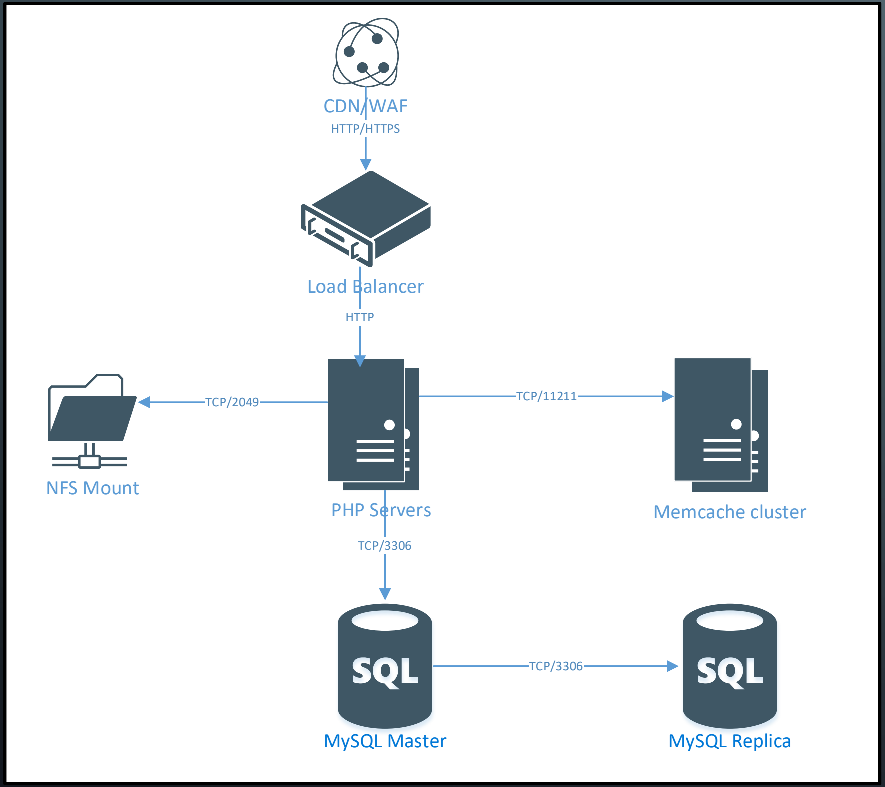
Apache*/
php-fpm

MySQL

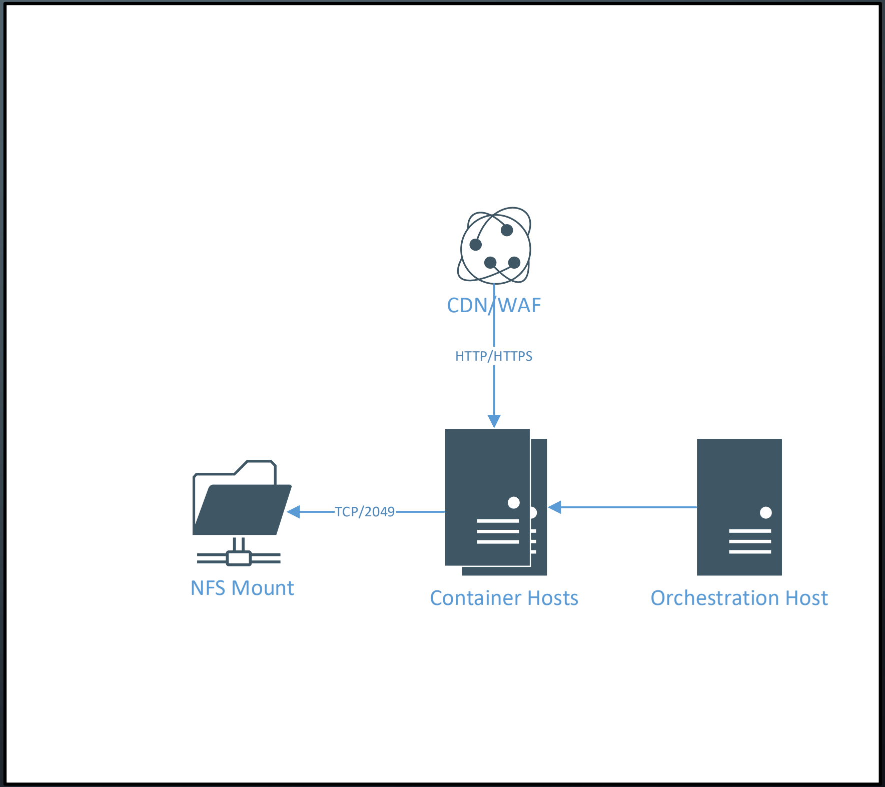
memcached

# SCALING WORDPRESS

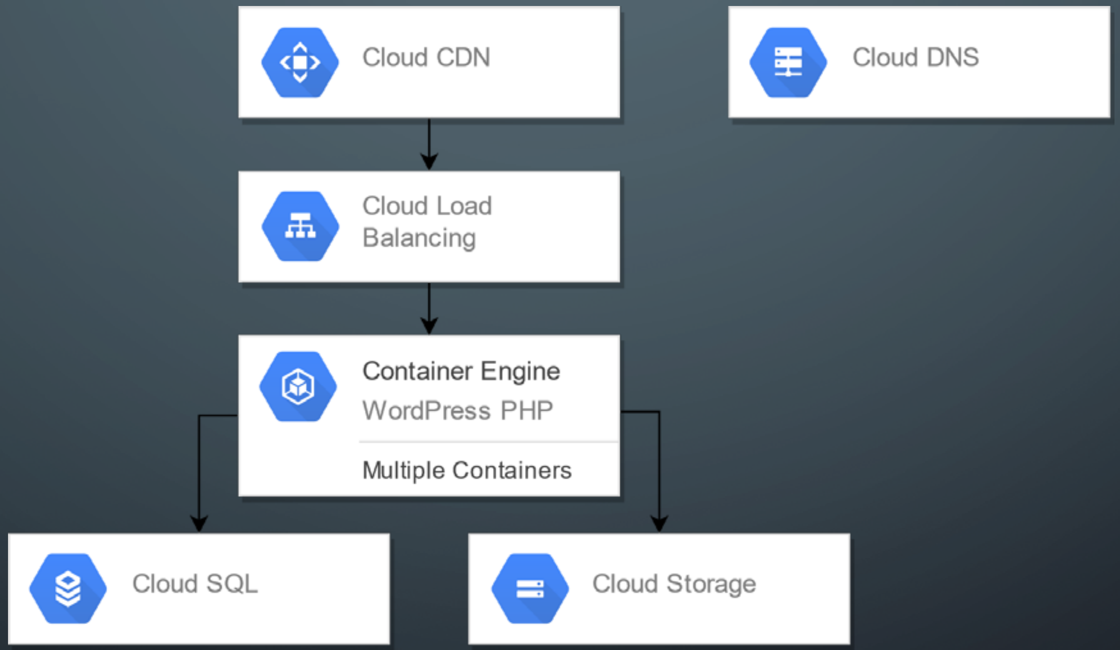
## BY COMPONENT



## CONTAINERIZED



# USING GOOGLE CLOUD COMPUTE


Cloud CDN

Cloud DNS

Cloud Load Balancing

Container Engine

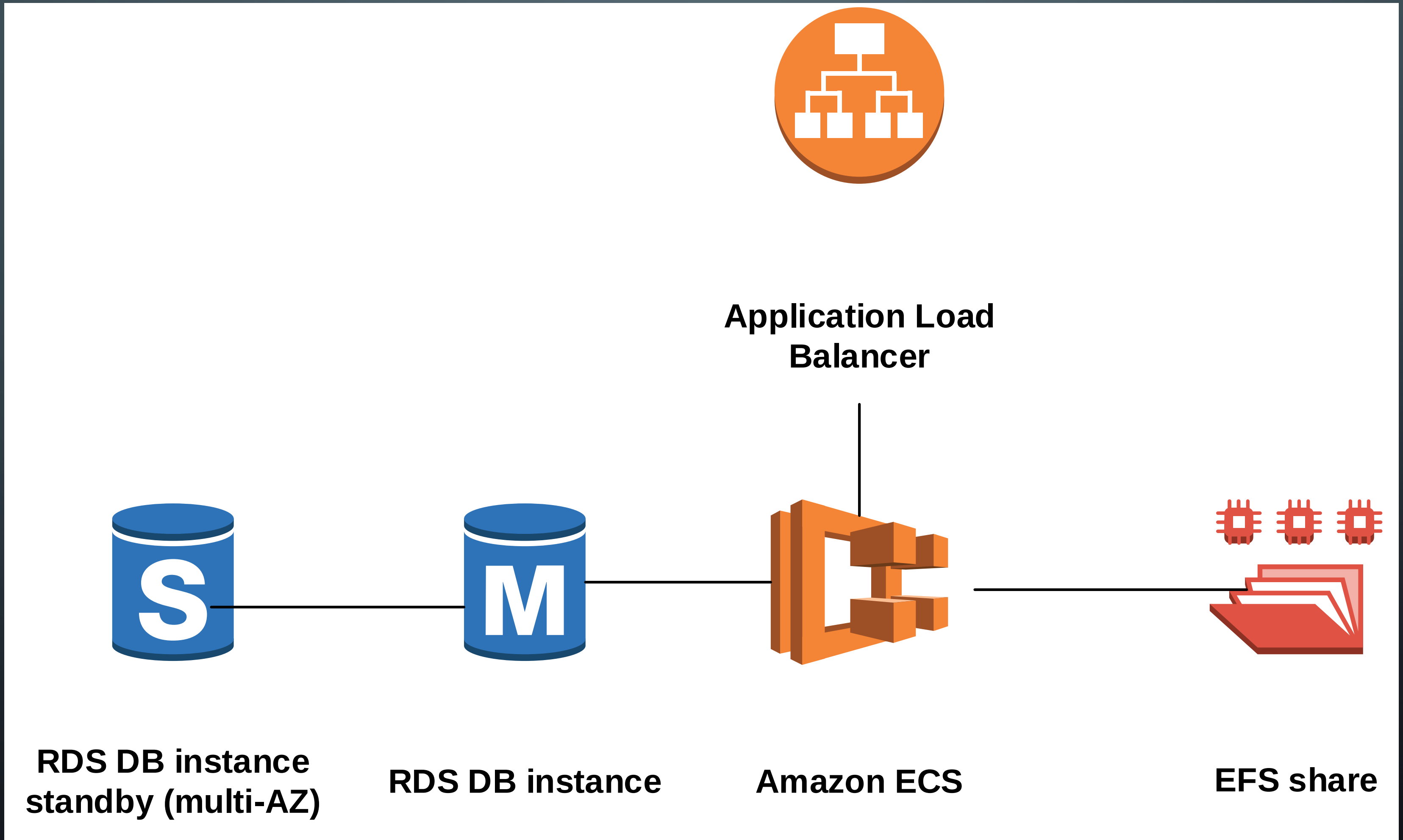
WordPress PHP

Multiple Containers

Cloud SQL

Cloud Storage

# USING AMAZON WEB SERVICES



# CONSIDERATIONS FOR BETTER-SCALED DOCKERIZED WORDPRESS

- Lock out plugin and WP auto-upgrades (use Docker images)
- Send wp-uploads using media library plugins (S3, Azure, GCP)
- Use orchestration for service discovery
- Multiple container hosts for resiliency

# THANK YOU!

@alan_lok