stackbuilders

# Lift & Shift

## Our Journey to Serverless

I'm Nicolas Vivar

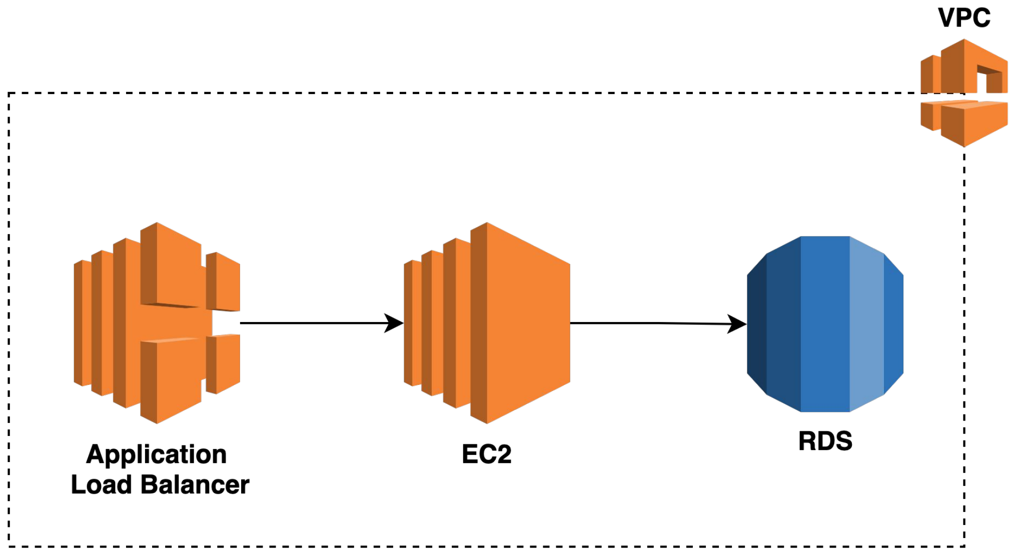
DevOps / SRE Engineer

Stack Builders

Quito - Ecuador

## A couple of years ago...

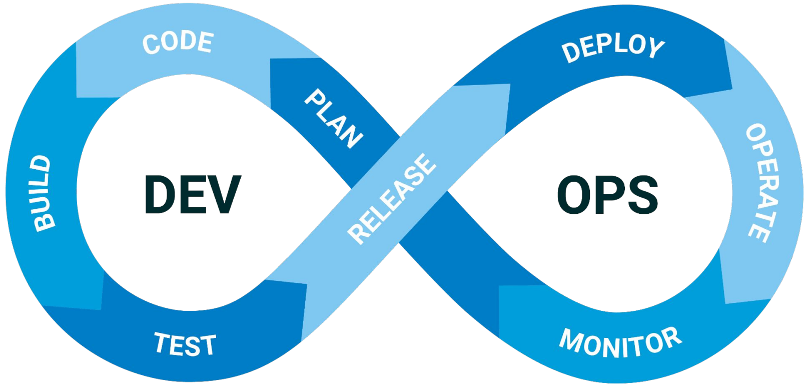
- Basic Infrastructure ALB - EC2 - RDS



## Everything under control

- ○ Infrastructure as Code - Terraform
- ○ Configuration Management - Ansible
- ○ CI/CD pipelines based on Gitflow

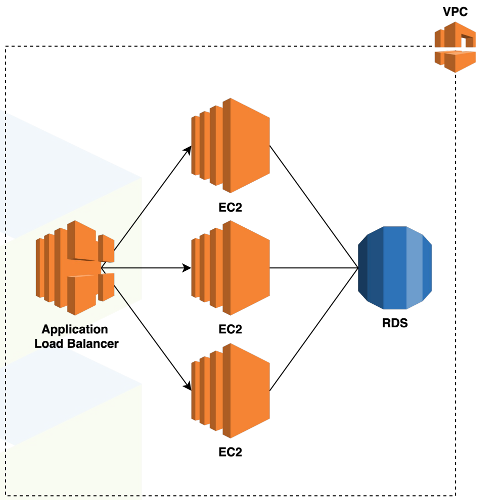
- Implemented DevOps pipeline



## Time came up and new technologies arrived!

Keep the current infrastructure

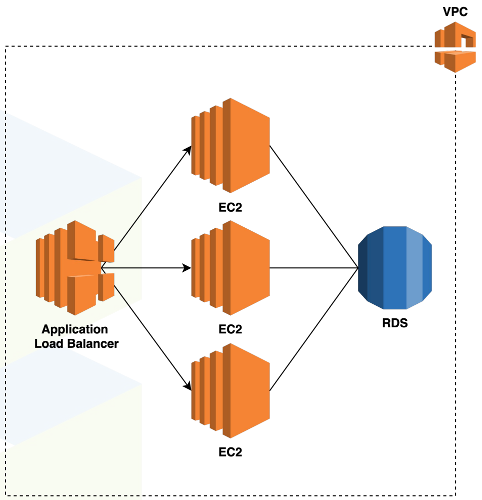
- Traditional Horizontal and Vertical scaling.
- Increase resources.
- Add more nodes

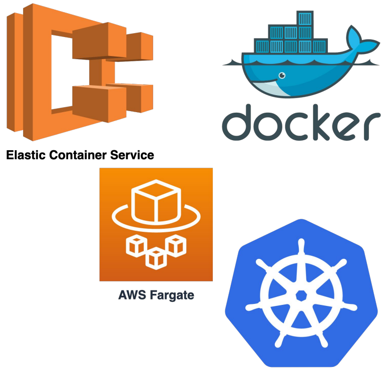


## Time came up and new technologies arrived!

Keep the current infrastructure
- Traditional Horizontal and Vertical scaling.
- Increase resources.
- Add more nodes

Rebuild the infrastructure
- Containers
- Orchestration
- Serverless





## Which tools? Why Serverless?

- Simplicity
- Scalability
- Security
- Cost-effective
- Docker friendly

# Which tools? Why Serverless?

- Simplicity
- Scalability
- Security
- Cost-effective
- Docker friendly

Elastic Container Service

ECS

Native Docker management and orchestration service

## Which tools? Why Serverless?

- Simplicity
- Scalability
- Security
- Cost-effective
- Docker friendly

Elastic Container Service

ECS

Native Docker management and orchestration service

Fargate

- Reduce maintenance burden
- Pay for the usage
- Friendly auto scaling configuration

## Lift and Shift - Big Changes Small Steps

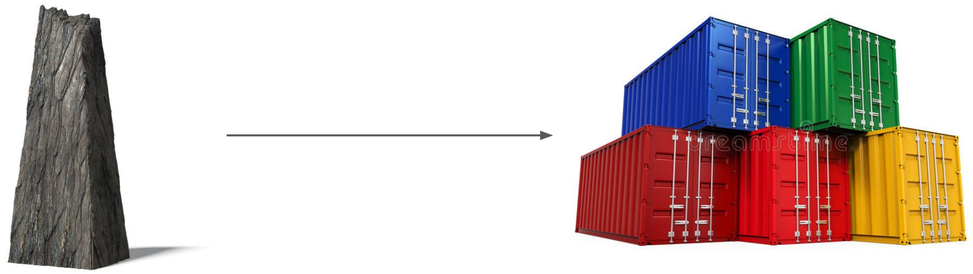
- Incremental changes.
- Small steps to minimize impact.

## Lift and Shift - Big Changes Small Steps

- Incremental changes.
- Small steps to minimize impact.
- Maximize the operational time of the application.
- Detect issues early in the workflow and reduce bottlenecks.

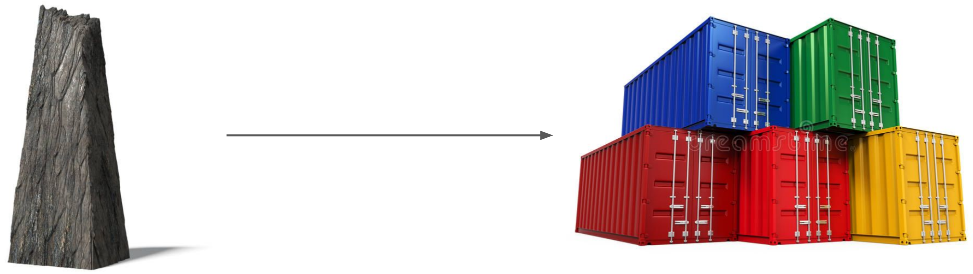
## Not everything is perfect...

- Monolith applications to container-based architecture



## Not everything is perfect…

- Monolith applications to container-based architecture
  - Compatibility
  - Data storage
  - Performance



## And...

- Clients' expectations
  - Deliver value
  - Fit the budget
  - Business growth

## Lessons Learned

- Manage persistent data. Containers are ephemeral.
- Always share context with the Dev team. Collaboration is crucial.

## Lessons Learned

- Manage persistent data. Containers are ephemeral.
- Always share context with the Dev team. Collaboration is crucial.
- Improve the monitoring. Application's health and limits.
- Avoid reactive scalability.

## Lessons Learned

- Manage persistent data. Containers are ephemeral.
- Always share context with the Dev team. Collaboration is crucial.
- Improve the monitoring. Application's health and limits.
- Avoid reactive scalability.
- Focus on fast reproducibility.
- Make everything to be resilient.

## Thanks!

@nickovivar

/nickovivar