AWSUGMM - Meetup #1

23 June 2019, Seedspace Yangon

AWS USER GROUP MYANMAR

Launched on 1 May, 2019

700+ Community Members

/awsugmm	/awsugmm
/awsusergroupmyanmar	/awsugmm

Amazon Web Service နဲ့ပက်သတ်တဲ့ Knowledge များနှင့် နည်းပညာ သစ်များကို User များ ခင်မင်ရင်းနှီးစွာ လေ့လာဝေမျှ ပေးနိုင် ရန်။

Monthly Meetups

Annual Conference

Random Hackathons

Thank You To...

seedspace

aws
USER GROUP
MYANMAR

Meetup #1

1 : 15 pm - 1 : 30 pm	registration
1 : 30 pm - 1: 45 pm	awsugmm introduction
1 : 45 pm - 2 : 15 pm	aws global infrastructure and core services
2 : 15 pm - 2 : 30 pm	*tea break*
2 : 30 pm - 3 : 15 pm	cloud architecture best practice on aws
3 : 15 pm - 3 : 30 pm	*networking session*

Cloud Architecture

Best Practice on AWS

Phyo Min Htun @ AWS User Group Myanmar

@Channel - https://phyominhtun.slack.com
@linkedin - https://www.linkedin.com/in/phyominhtun/
@Key - pub 2048R/90373DB4 2015-10-21
uid Phyo Min Htun (Phyo Min Htun PGP Key ...) <phyominhtun@protonmail.com>
sub 2048R/DEE94BB9 2015-10-21

AWS Global Infrastructure

Regions

21

Availability Zones

65+

Points of Presence

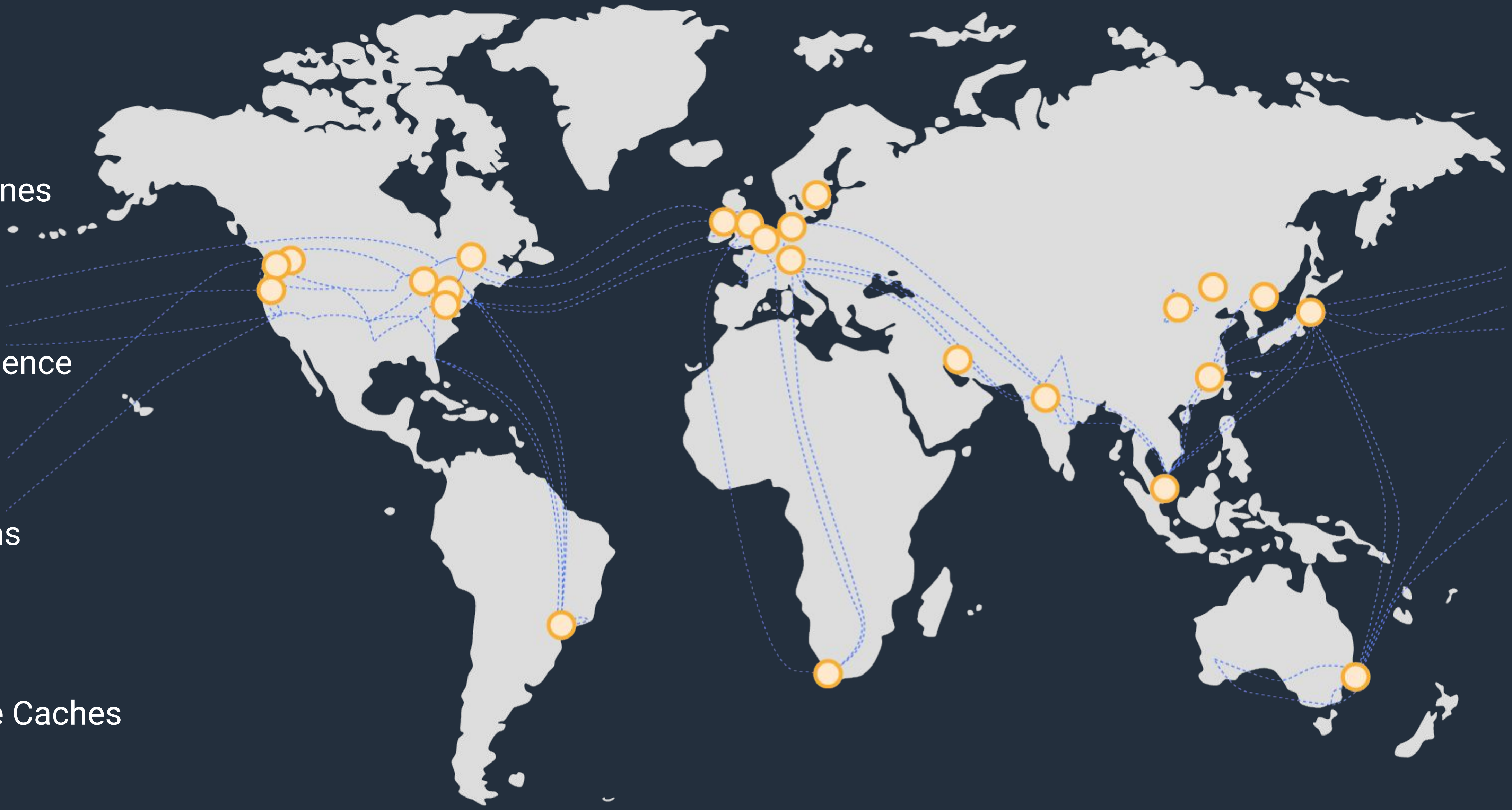
180

Edge Locations

169

Regional Edge Caches

11



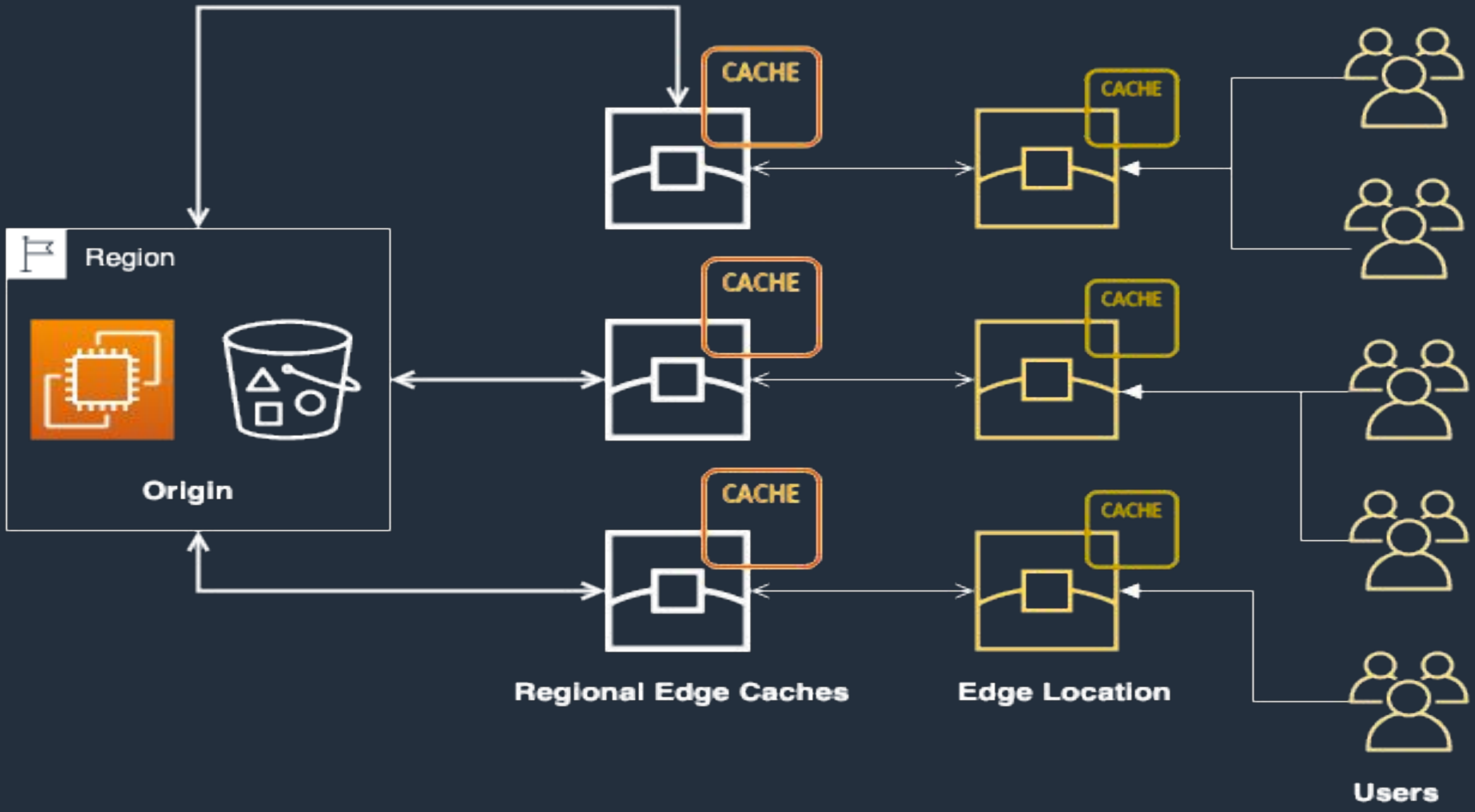
AWS USER GROUP MYANMAR

Benefit of AWS Global Infrastructure

- ❖ Increase Availability & Services
- ❖ Go global in minutes
- ❖ Disaster Recovery

Regional Edge Caches

- AWS Announced , November 2016
- Reduce the load of origin
- Improve performance for viewers
- Reduce origin costs
- Larger cache-width than edge locations


CACHE
CACHE
CACHE
CACHE
CACHE
CACHE
Region
Origin
Regional Edge Caches
Edge Location
Users

General Design Principles

- Stop guessing your capacity needs
- Test systems at production scale
- Automation and Orchestration
- Easy to change Architecture
- Drive architectures using data
- Improve through game days

Cloud Architecture

Best Practices on AWS

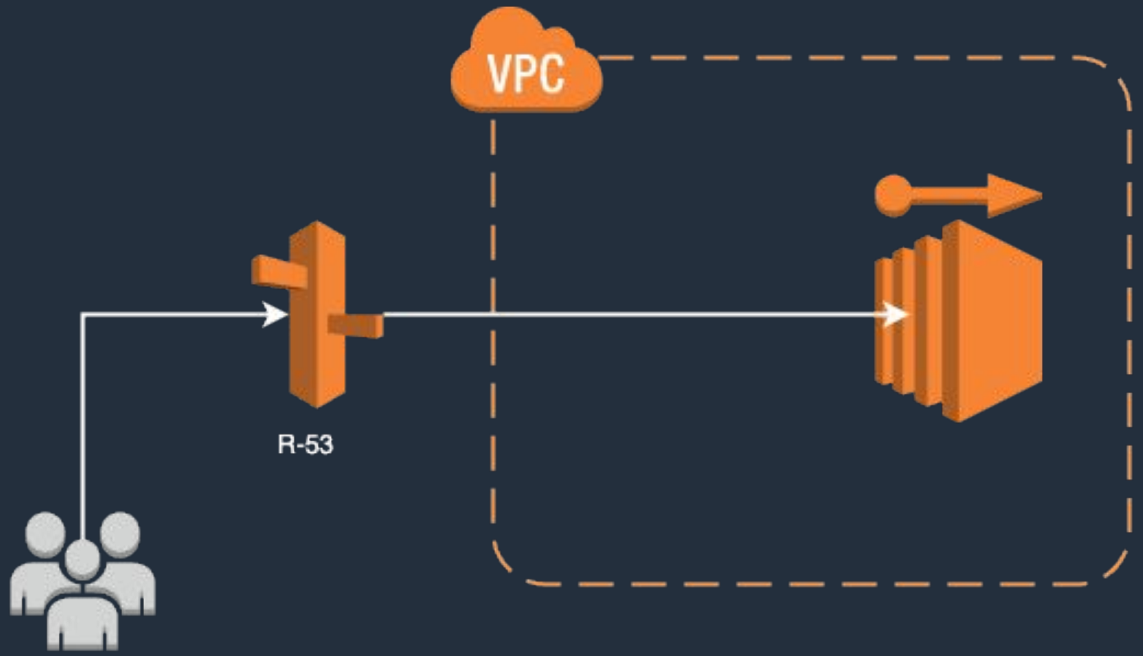
AWS USER GROUP MYANMAR

1. Design for failure
2. Build Security in every layer
3. Leverage different storage options
4. Implement elasticity
5. Think Parallel
6. Loose coupling
7. Don’t fear constraints

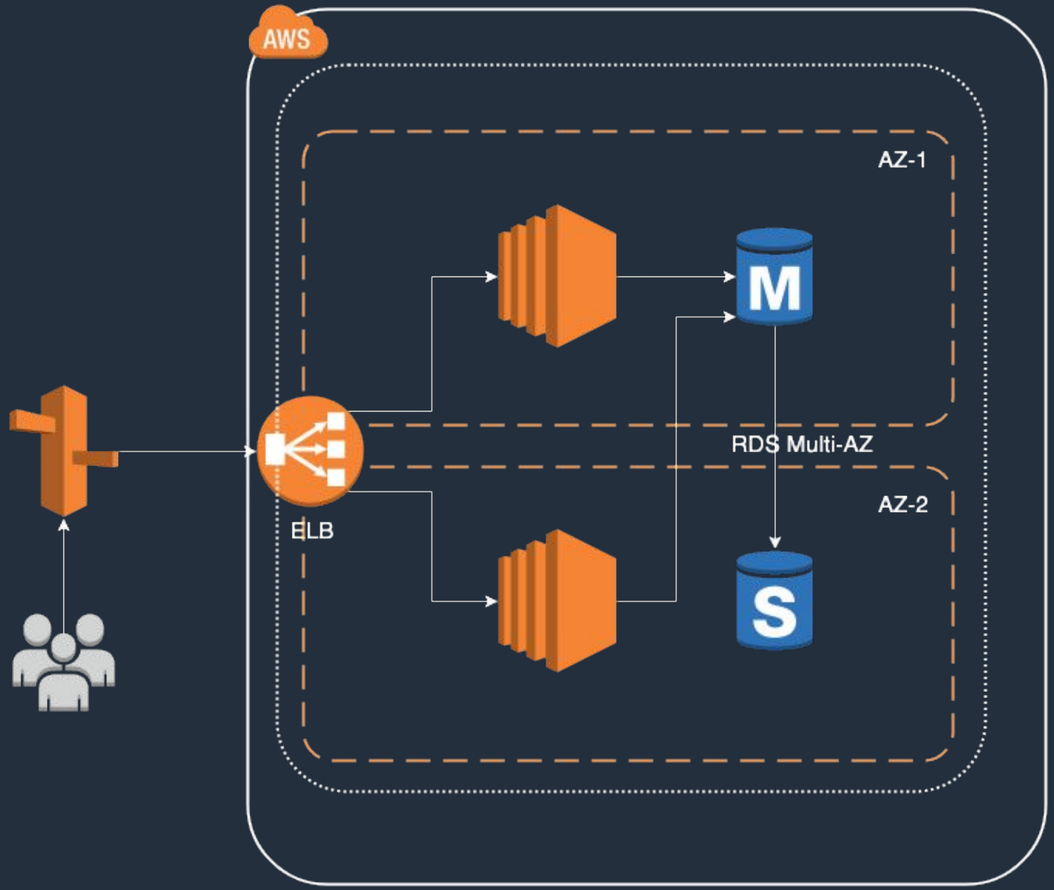
1. Design for failure

“Everything fails all the times”
Werner Vogels, CTO Amazon.com

- ❖ Amazon Route 53 for DNS
- ❖ Single Amazon EC2 Server
  - ➢ Web
  - ➢ DB
  - ➢ and other services.
- ❖ Single Elastic IP Address



- Amazon Route 53 for DNS
- Add Another Ec2 Instance in another AZ
- RDS Multi-AZ Deployment
- Use Elastic Load Balancing (ELB) Instead of Single Elastic IP Address



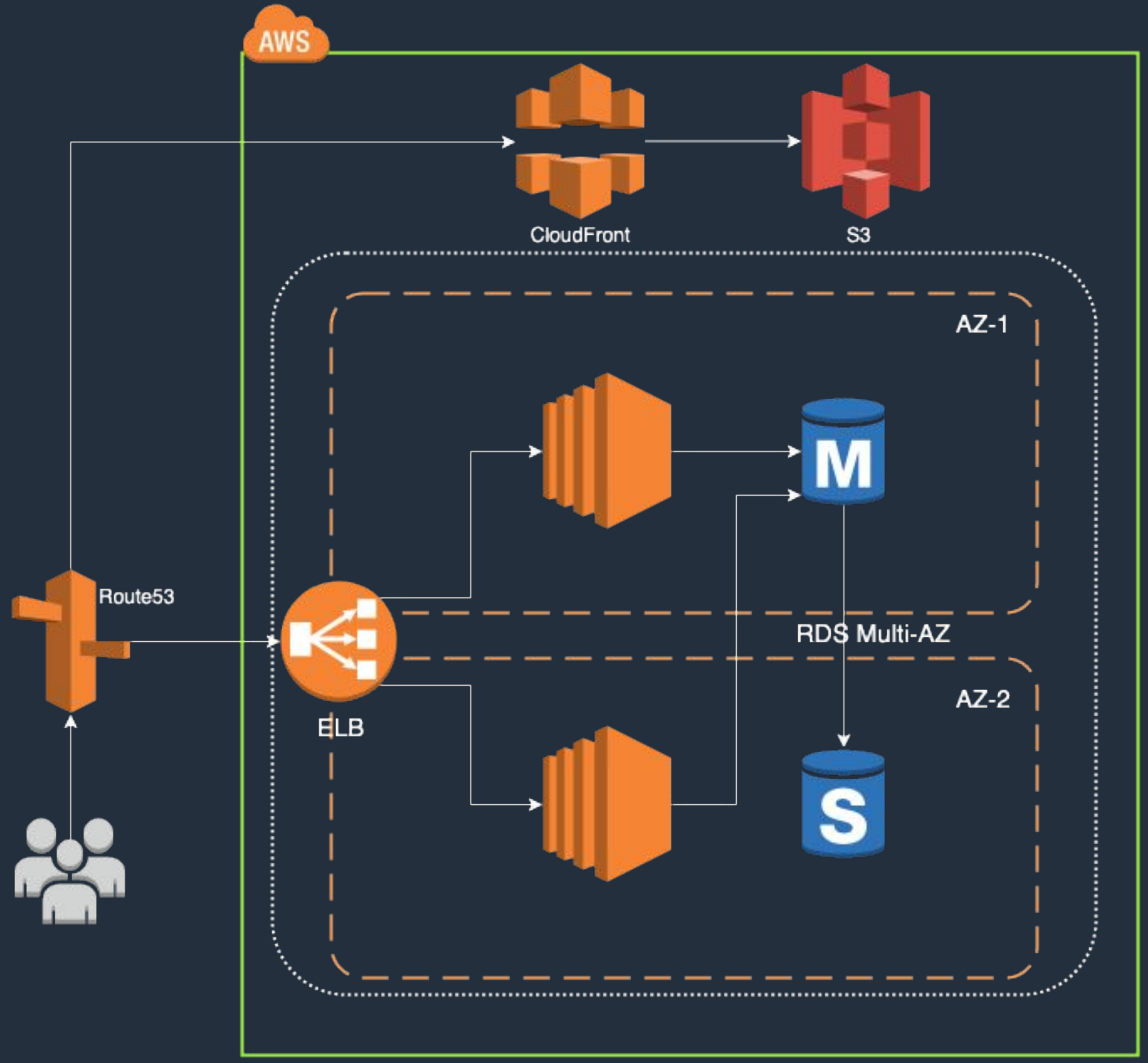
2 . Build Security in every layer

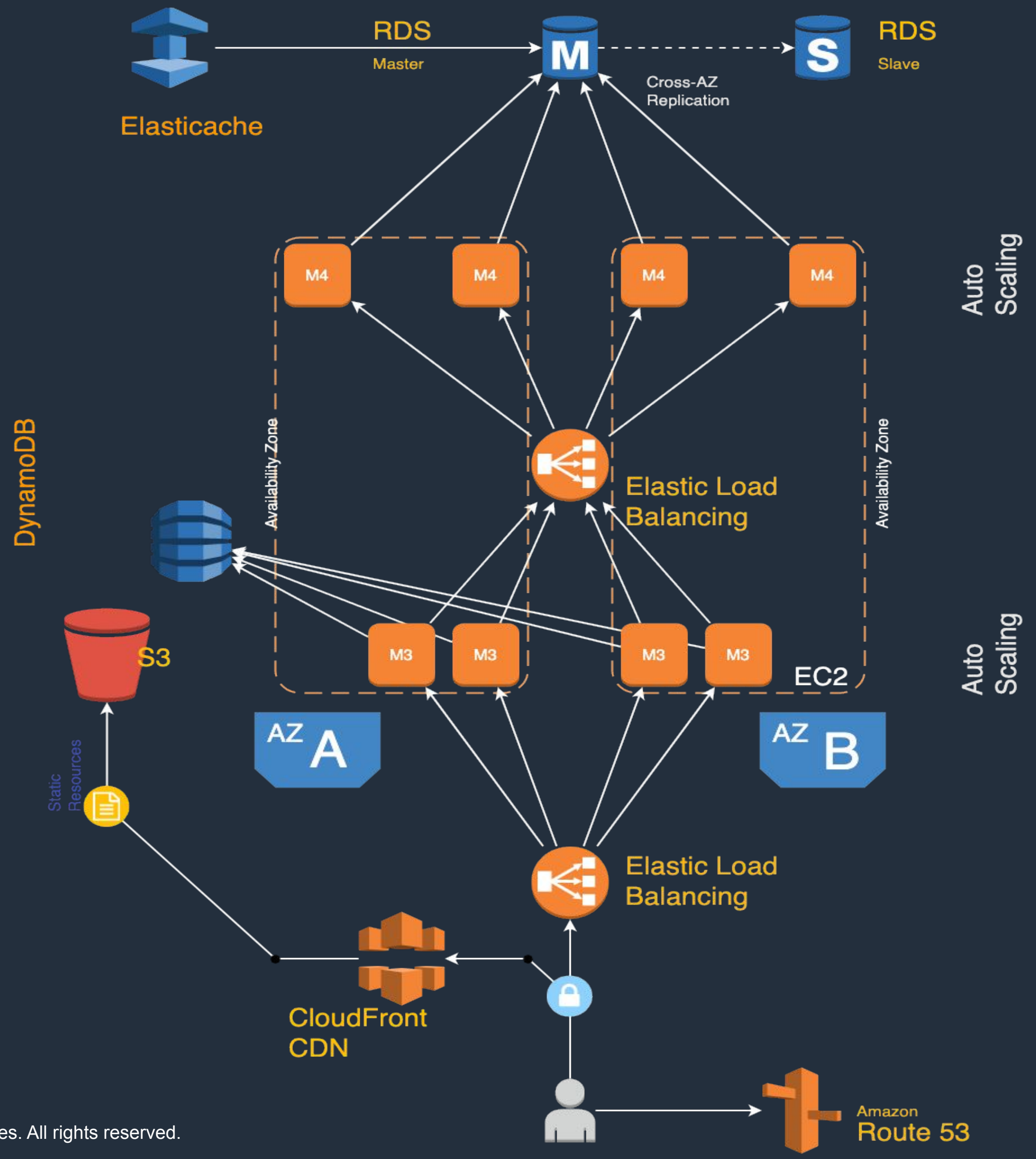
- Principle of least privilege.
  - Amazon Identity and Access Management (IAM)
    - users, groups, roles and policies.

      https://github.com/phyominhtun1990/aws-policies
- Restricted with Security Group and Network ACL.
- Use Multi-Factor Authentication (MFA).
- Encrypt data in transit and at rest.
  - Amazon KMS

3 . Leverage different storage options

Static Content to Amazon S3 and Amazon CloudFront





RDS
Master
M
S
RDS
Slave
Cross-AZ Replication
Elasticache
M4
M4
M4
M4
Auto Scaling
DynamoDB
Availability Zone
Elastic Load Balancing
Availability Zone
M3
M3
M3
M3
S3
EC2
Auto Scaling
AZ A
AZ B
Static Resources
Elastic Load Balancing
CloudFront CDN
Amazon Route 53

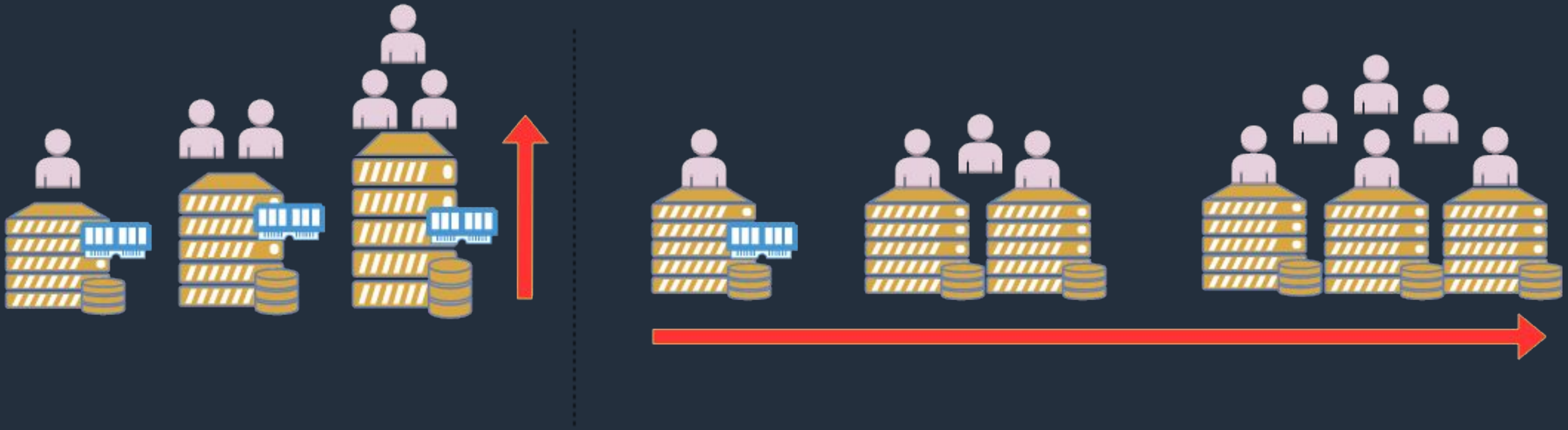
- Elastic Block Storage (EBS)
  - Solid-State Drives - SSD
    - General Purpose SSD (gp2)
    - Provisioned IOPs SSD (io1)
  - Hard Disk Drives - HDD
    - Throughput Optimized HDD (st1)
    - Cold HDD (sc1)

- Simple Storage Service (S3) Storage Class
  - Standard
  - Standard Infrequent Access (Standard-IA)
  - One Zone Infrequent Access (One Zone-IA)
  - Glacier & Deep Archive

- Amazon Elasticache
- Amazon DynamoDB
- Amazon Elastic File System

4 . Implement elasticity

Scaling Vertically and Horizontally

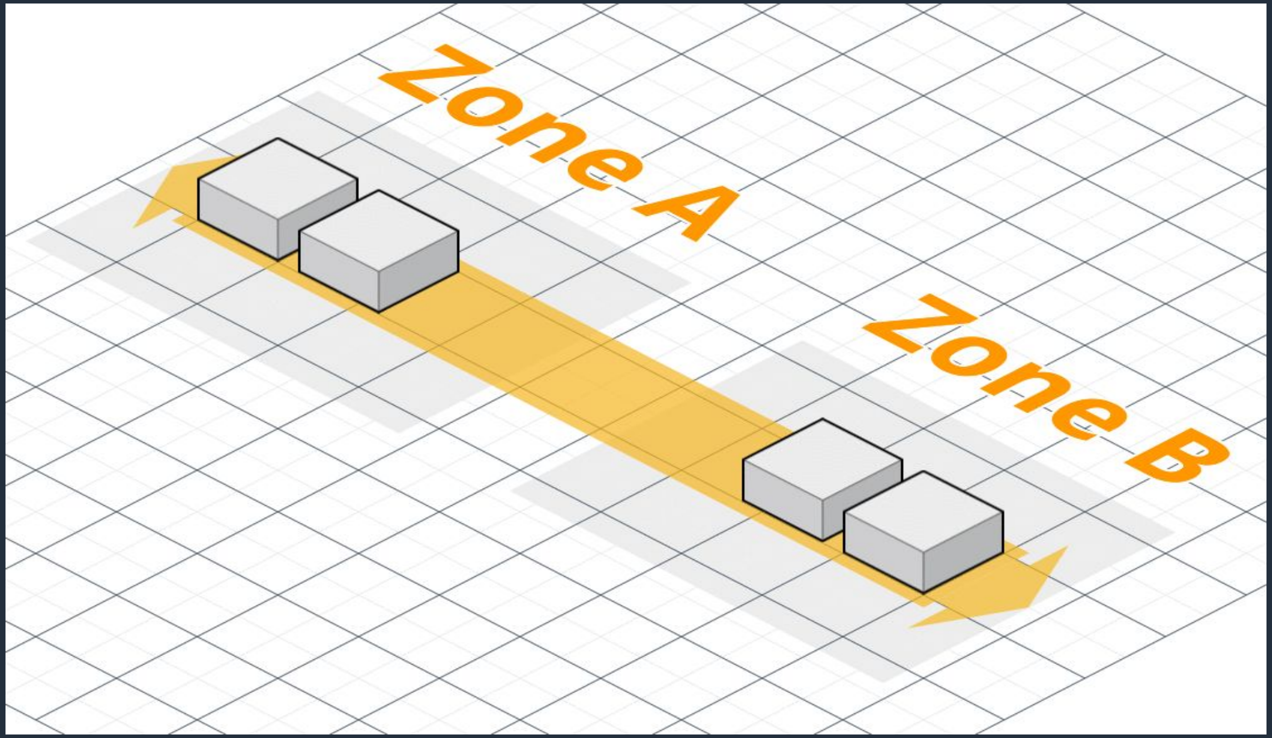


AWS Auto Scaling

Auto Scaling ensures Amazon EC2 instances are sufficient to run your application.

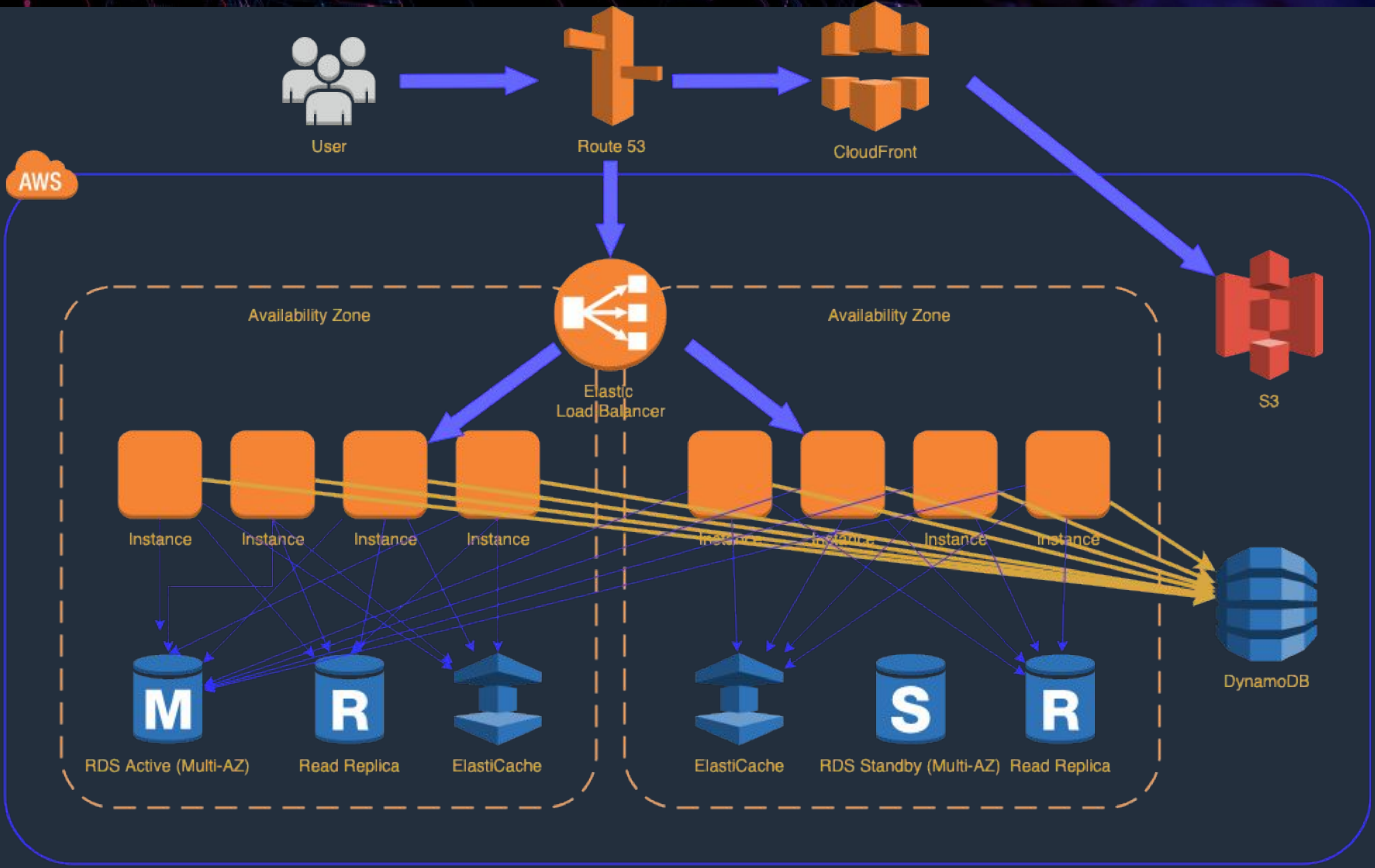
It allows you to

- Dynamically changes in load.
- Prevent over provisioning.
- Can be Integrate with ELB and EC2.
- Metrics via AWS Cloud Watch.



5 . Think Parallel

- Experiment with different parallel architecture
- AWS Pay-as-you-go model
- Multithreading and concurrent requests to cloud services
- Use Elastic Load Balancing to distribute load
- AWS Lambda - Run thousand of functions run in parallel


User
Route 53
CloudFront
AWS
S3
Availability Zone
Elastic Load Balancer
Availability Zone
Instance
Instance
Instance
Instance
Instance
Instance
Instance
Instance
DynamoDB
M
R
S
R
RDS Active (Multi-AZ)
Read Replica
ElastiCache
ElastiCache
RDS Standby (Multi-AZ)
Read Replica

6 . Loose Coupling

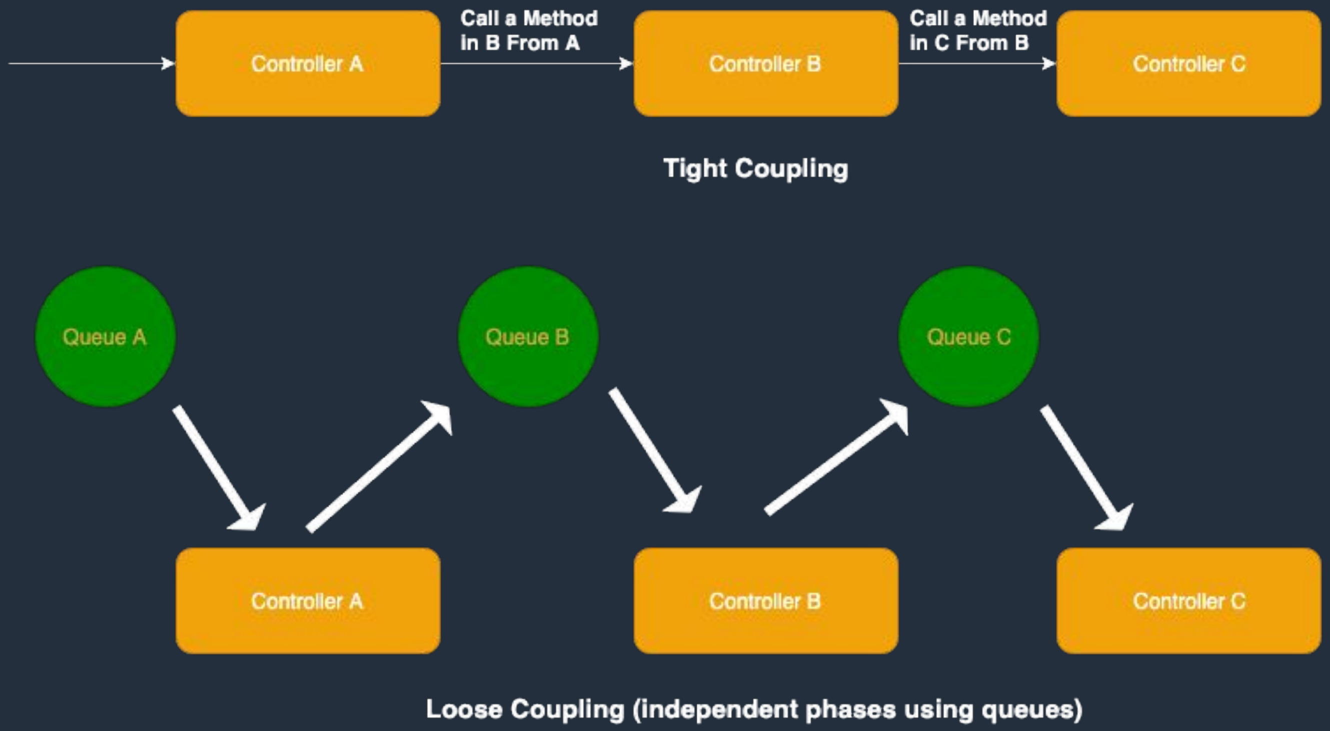
- Design architectures with independent components
- Each component can scale independently if needed
- Failure in an element does not affect the rest of a system
- Recovery from a failure is easier comparing a tightly coupled complex system

Amazon Simple Queue Service (SQS)

AWS Announced , 2004

Scalable , fully managed message queuing service by AWS that offer a simple, low-cost way to decouple cloud components.

“The more loosely system components are coupled, the larger they scale”



7. Don’t fear constraints

Rethink traditional architectural constraints

Need More RAM?

- ❖ Consider distributing load across machines or cache

Need Better IOPS for database?

- ❖ Multiple Read replicas or DB Clustering
- ❖ PIOPS Type

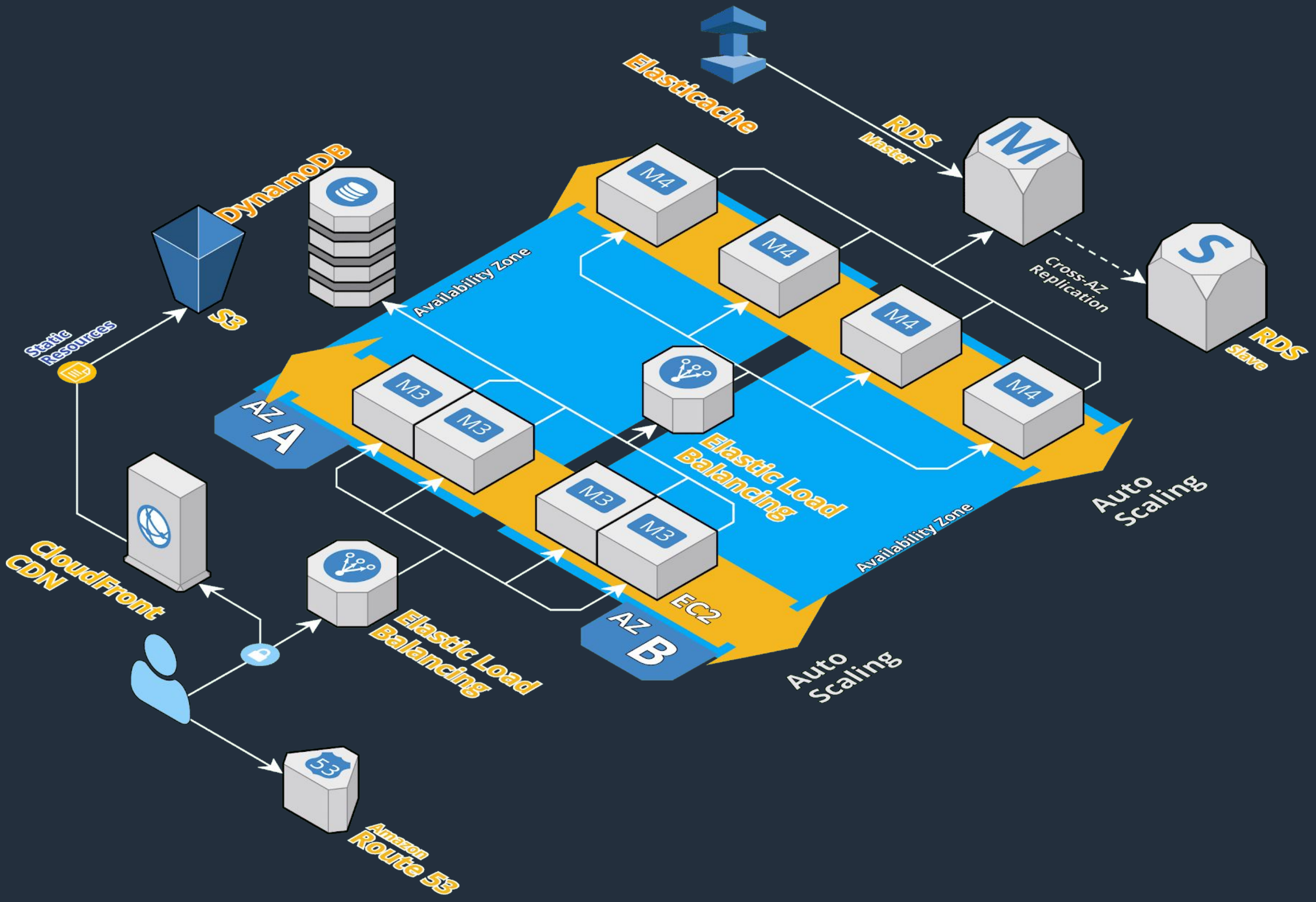
Hardware failed or corrupted?

- ❖ Replace

Cost effective disaster recovery strategy?

- ❖ Use Route-53 for failover strategy

Recap - High Availability Web App Architecture



AWS User Group Myanmar

https://aws.amazon.com/developer/community/usergroups/asia-pacific/

https://www.facebook.com/awsugmm/

https://www.facebook.com/groups/AWSusergroupmyanmar/