Architectural Design Patterns in Cloud Computing

Cloud Best Practices Whitepaper

Prescriptive guidance to Cloud Architects

Just Search for “Cloud Best Practices” to find the link

ttp://media.amazonwebservices.com/AWS_Cloud_Best_Practices.pdf

- or -

http://aws.amazon.com/whitepapers

Scalability

Build Scalable Architecture on AWS

A scalable architecture is critical to take advantage of a scalable infrastructure

Characteristics of Truly Scalable Service

- Increasing resources results in a proportional increase in performance
- A scalable service is capable of handling heterogeneity
- A scalable service is operationally efficient
- A scalable service is resilient
- A scalable service becomes more cost effective when it grows

Cloud Architecture Lessons

using Amazon Web Services

1. Design for failure and nothing fails
2. Loose coupling sets you free
3. Implement “Elasticity”
4. Build Security in every layer
5. Don't fear constraints
6. Think Parallel
7. Leverage different storage options

1. Design for Failure

and nothing will really fail

"Everything fails, all the time"
Werner Vogels, CTO Amazon.com

Avoid single points of failure
Assume everything fails, and design backwards
Goal: Applications should continue to function even if the underlying physical hardware fails or is removed or replaced.

1. Design for Failure

and nothing will really fail

Best Practices

- Use multiple Availability Zones
- Use Elastic IP addresses
- Use Elastic Load Balancers
- Real-time monitoring with CloudWatch
- Leverage Auto Scaling groups
- Create database slaves across Azs
- Practice failures/recovery (Chaos Monkey)

1. Design for Failure

and nothing will really fail

High Availability Principles

- Focus on overall application availability, not one resource
- Scale horizontally across AZs for durability
- Start replacement instances, don’t save dying ones
- Design for eliminating the need for maintenance windows

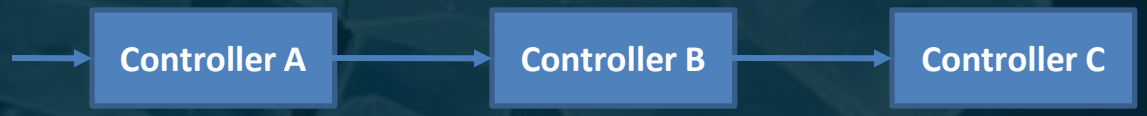
2. Loose coupling sets you free

The looser they're coupled, the bigger they scale

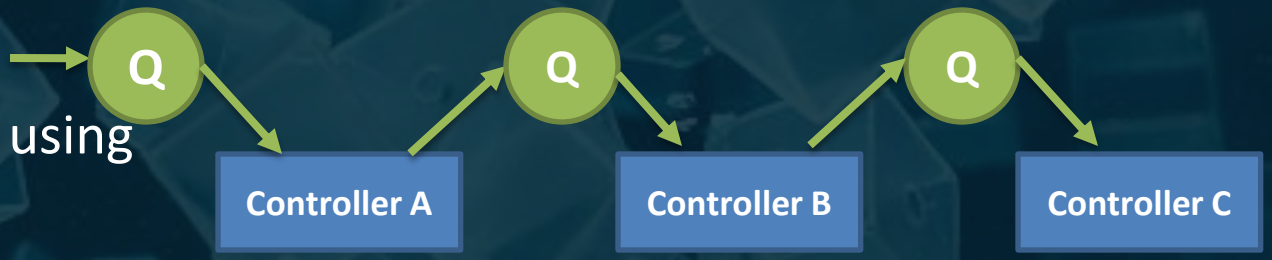
Independent components
Design everything as a Black Box
De-coupling for Hybrid models
Load-balance clusters

Use Amazon SQS as Buffers

Tight Coupling



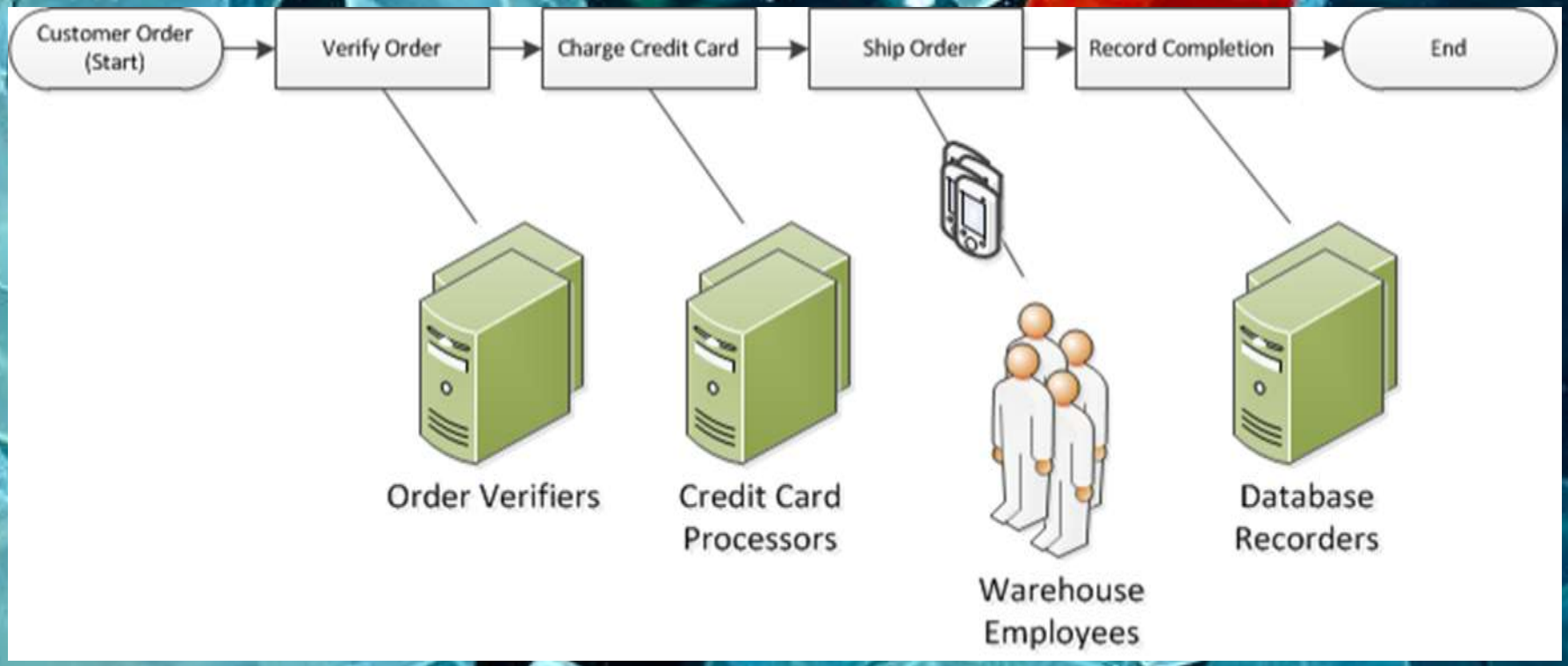
Loose Coupling using Queues



2. Loose coupling sets you free

The looser they're coupled, the bigger they scale

Use Amazon SWF for distributed, decoupled and asynchronous applications



3. Implement Elasticity

Elasticity is fundamental property of the Cloud

Don't assume health or fixed location of components
Use designs that are resilient to reboot and re-launch
Bootstrap your instances: Instances on boot will ask a question *"Who am I & what is my role?"*
Enable dynamic configuration

Use Auto-scaling (Free)
Use [Elastic] Load Balancing on multiple layers
Use configurations in SimpleDB to bootstrap instance

4. Build Security in every layer

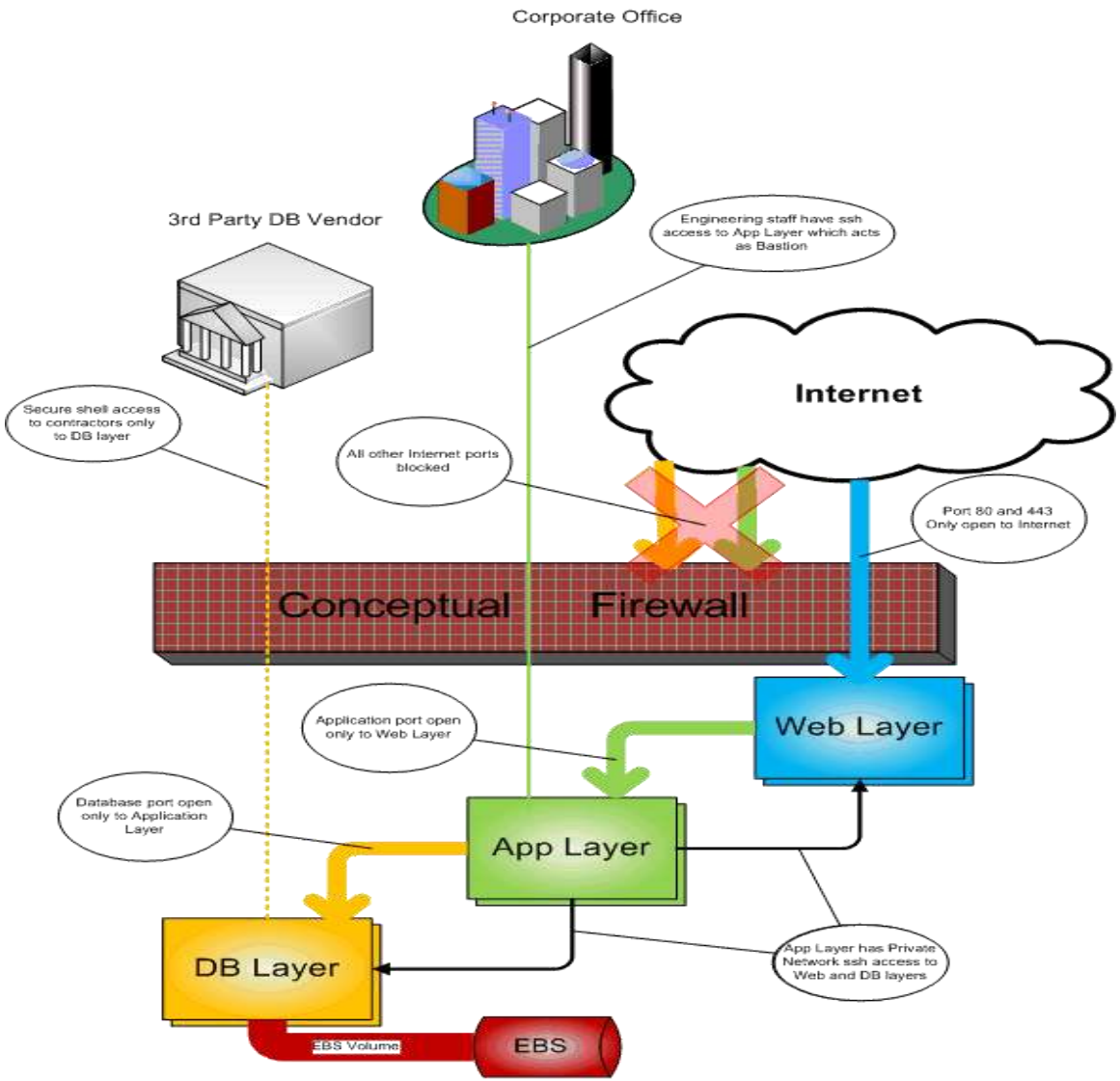
Design with Security in mind

With cloud, you lose a little bit of physical control but not your ownership

Create distinct Security Groups for each Amazon EC2 cluster
Use group-based rules for controlling access between layers
Restrict external access to specific IP ranges
Encrypt data "at-rest" in Amazon S3
Encrypt data "in-transit" (SSL)
Consider encrypted file systems in EC2 for sensitive data
Use AWS Identity & Access Management (IAM)
Use MultiFactor Authentication (MFA)

4. Build Security in every layer

Design with Security in mind



"Web" Security Group:

TCP	80	0.0.0.0/0
TCP	443	0.0.0.0/0
TCP	22	"App"

"App" Security Group:

TCP	8080	"Web"
TCP	22	172.154.0.0/16
TCP	22	"App"

"DB" Security Group:

TCP	3306	"App"
TCP	3306	163.128.25.32/32
TCP	22	"App"

5. Don't fear constraints

Re-think architectural constraints

More RAM? Distribute load across machines
Shared distributed cache

Better IOPS on my database?
Multiple read-only / sharding / DB clustering / Caching / Provisioned IOPs / SSD Instances

Your hardware failed or messed up config?
simply throw it away and switch to new hardware with no additional cost

Performance
Caching at different levels (Page, Render, DB)

6. Think Parallel

Serial and Sequential is now history

Experiment different architectures in parallel

Multi-treading and Concurrent requests to cloud services

Run parallel MapReduce Jobs

Use Elastic Load Balancing to distribute load across multiple servers

Decompose a Job into its simplest form

7. Leverage many storage options

One size DOES NOT fit all

Amazon S3: large static objects
Amazon CloudFront: content distribution
Amazon SimpleDB: simple data indexing/querying
Amazon DynamoDB: infinitely scalable NoSQL “Big Tables”
Amazon ElastiCache: Database caching
Amazon CloudSearch: fast, highly-scalable search functionality
Amazon EC2 local disc drive : transient data
Amazon EC2 High I/O Instances: high performance, local SSD-backed storage
Amazon EBS: persistent storage for any RDBMS + Snapshots on S3
Amazon EBS PIOPs: consistent, persistent storage for any RDBMS + Snapshots
Amazon RDS: RDBMS service - Automated and Managed MySQL

Applying Cloud Architecture Lessons

Moving a Web Architecture to the Cloud

Exterior Firewall Hardware or Software Solution to open standard Ports (80, 443)

Web Load Balancer Hardware or Software solution to distribute traffic over web servers

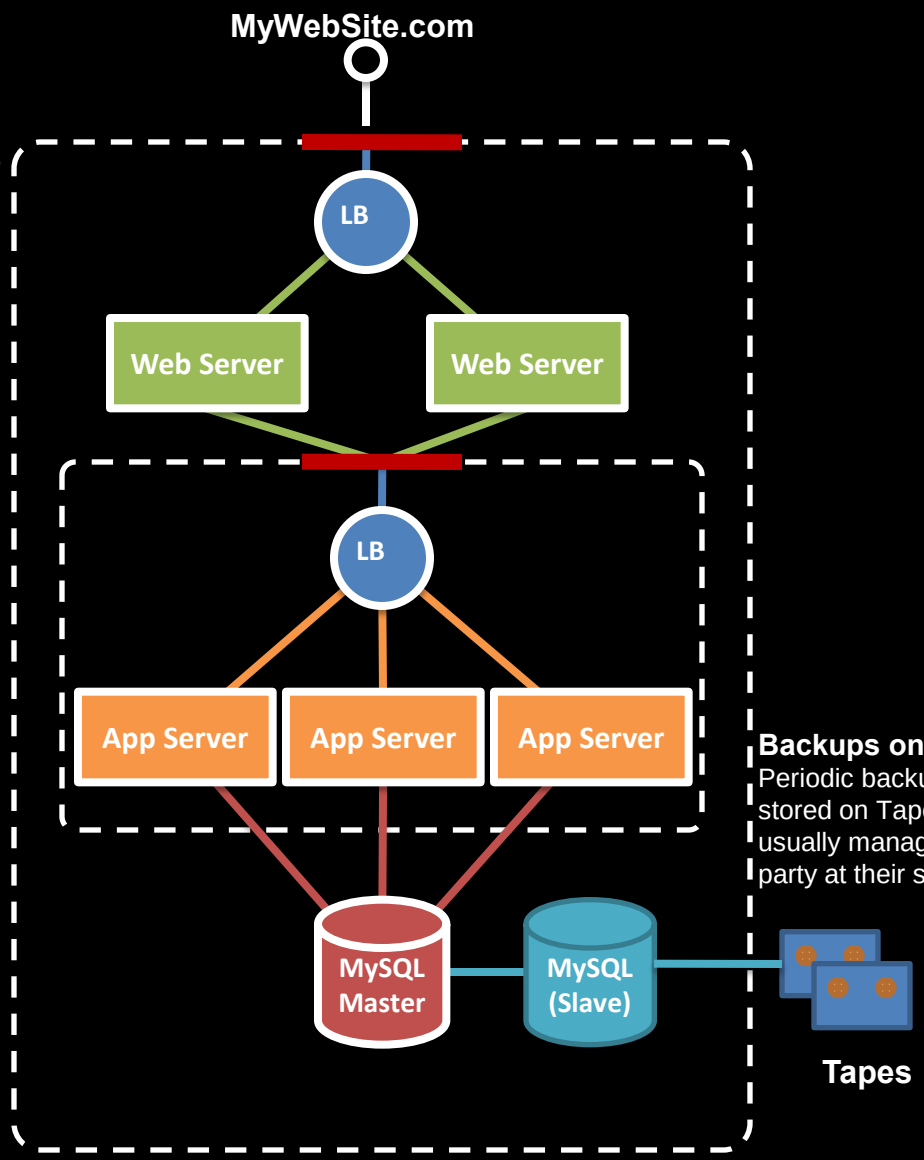
Web Tier
Fleet of machines handling HTTP requests.

Backend Firewall Limits access to application tier from web tier

App Load Balancer
Hardware or Software solution to spread traffic over app servers

App Server Tier
Fleet of machines handling Application specific workloads
Caching server machines can be implemented at this layer

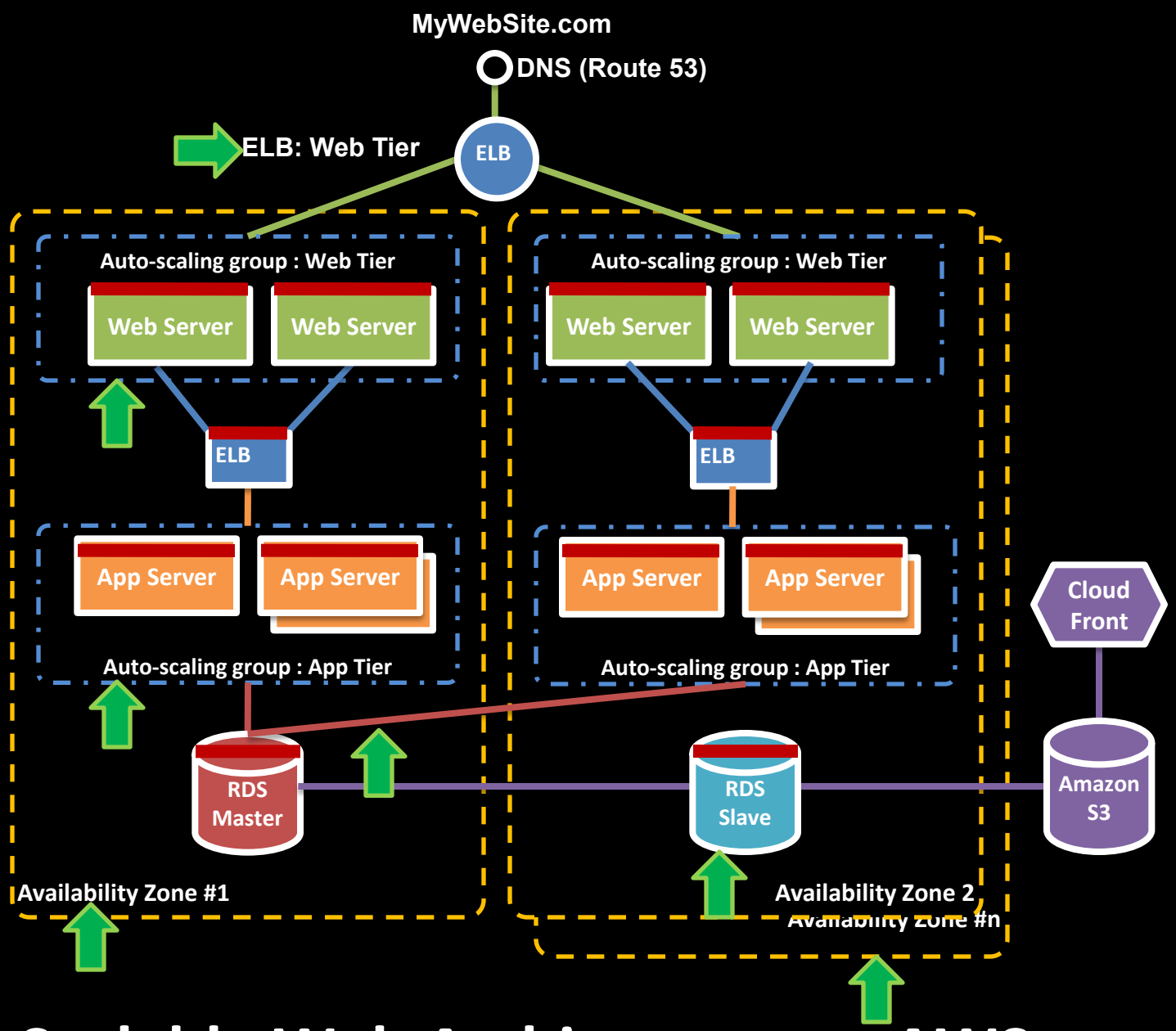
Data Tier
Database Server machines with master and local running separately, Network storage for Static objects



Backups on Tapes
Periodic backups stored on Tapes usually managed by 3rd party at their site

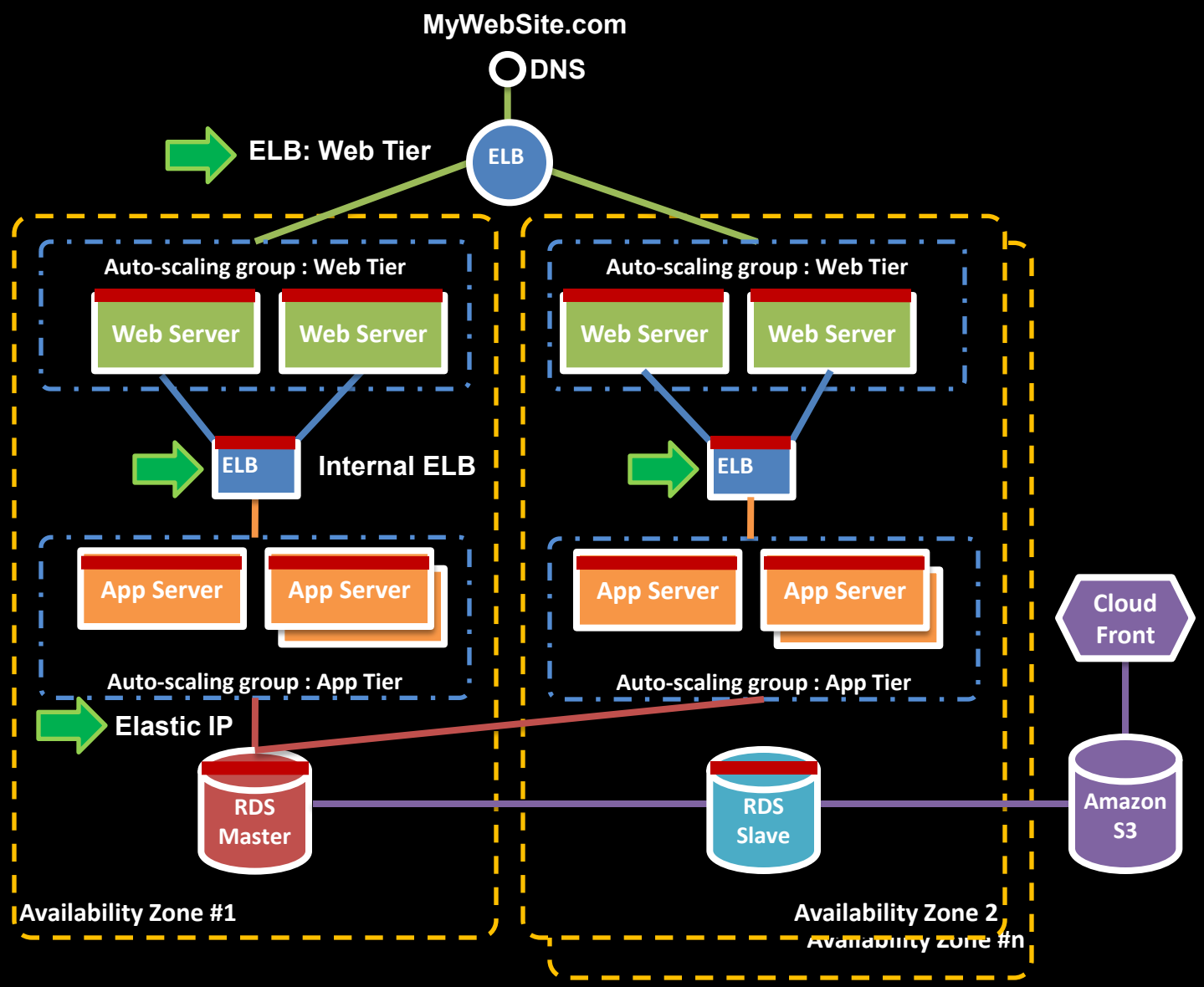
A Classic Web Architecture

Design for failure and nothing fails



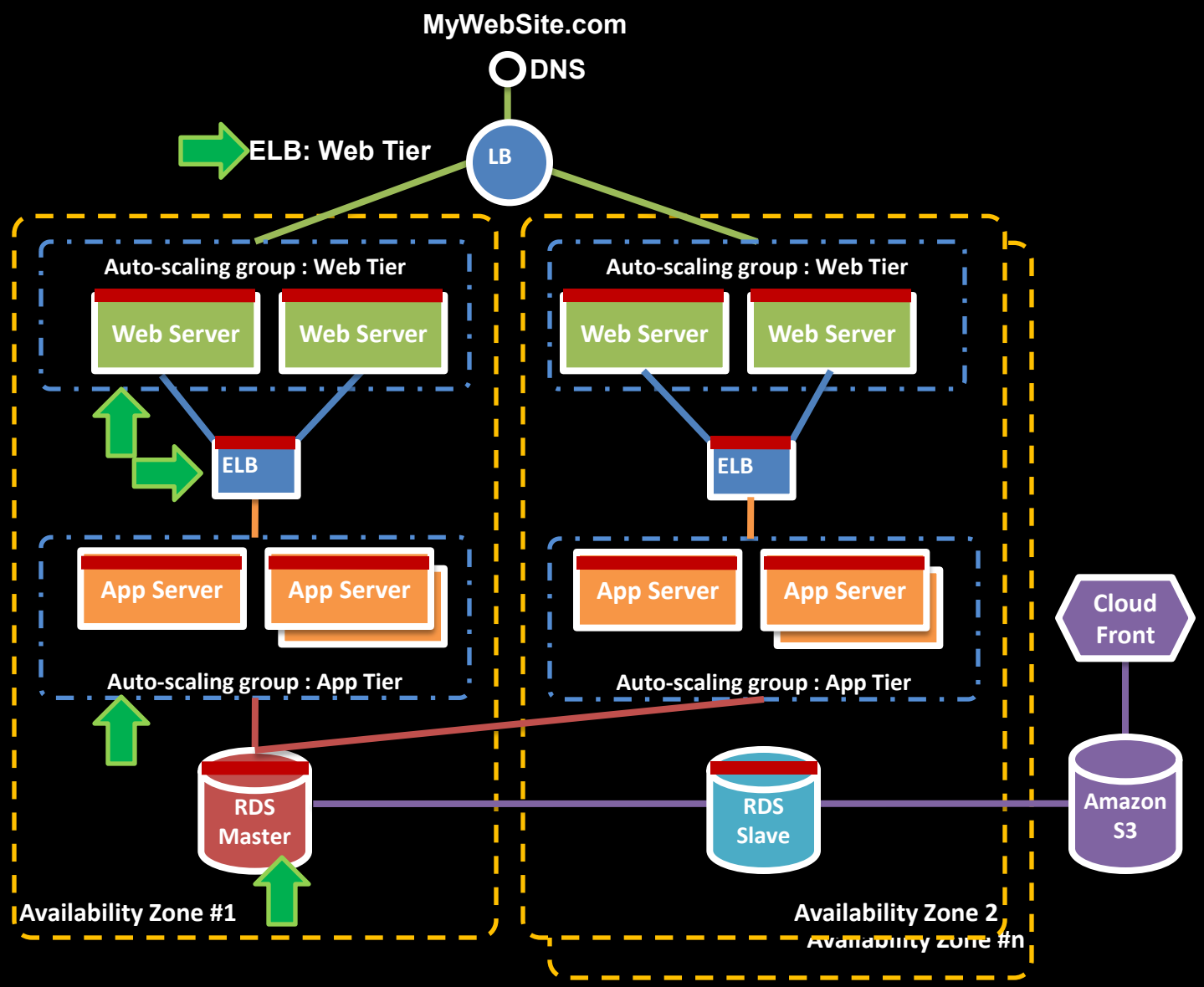
A Scalable Web Architecture on AWS

Loose coupling sets you free



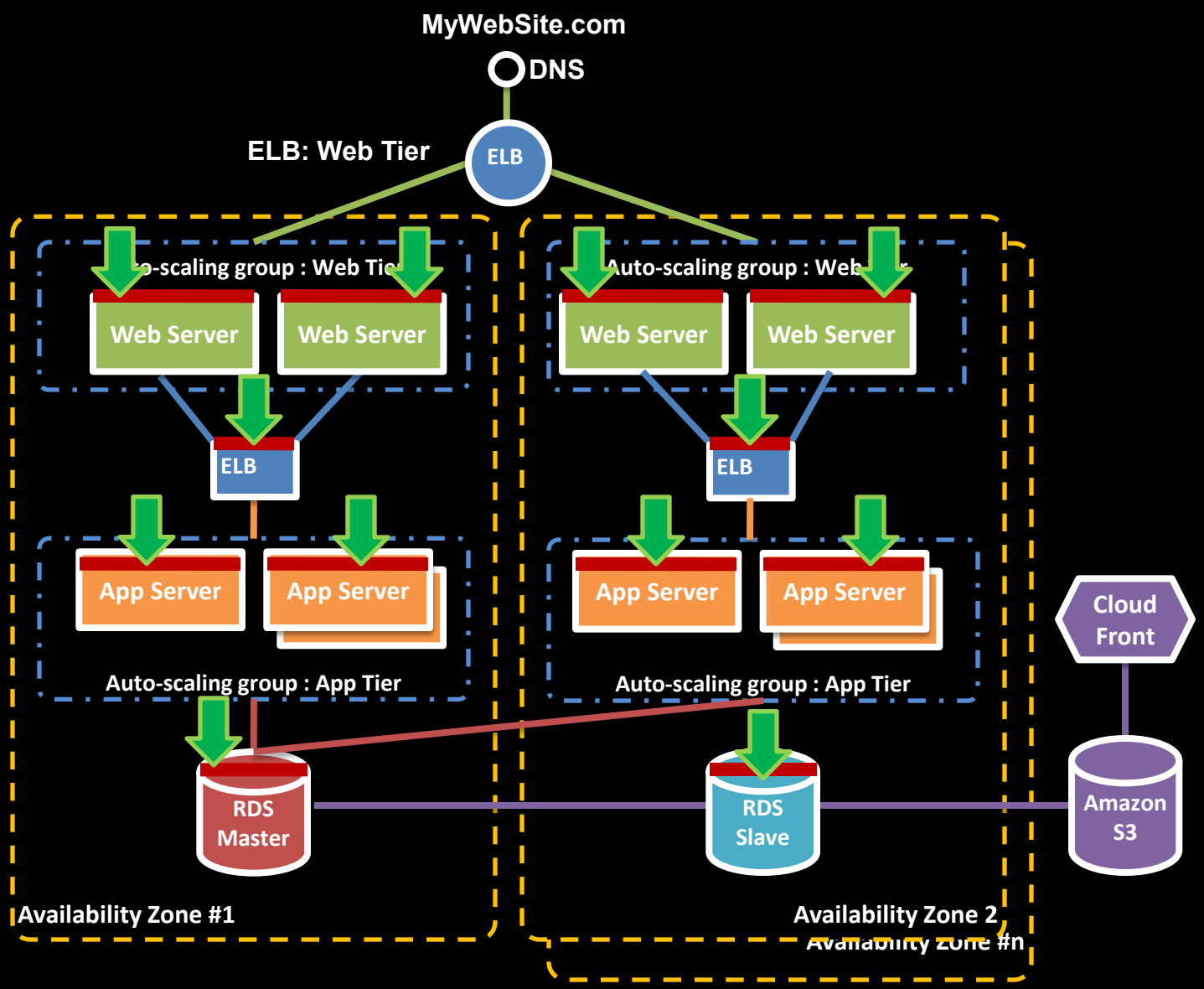
A Scalable Web Architecture on AWS

Implement elasticity



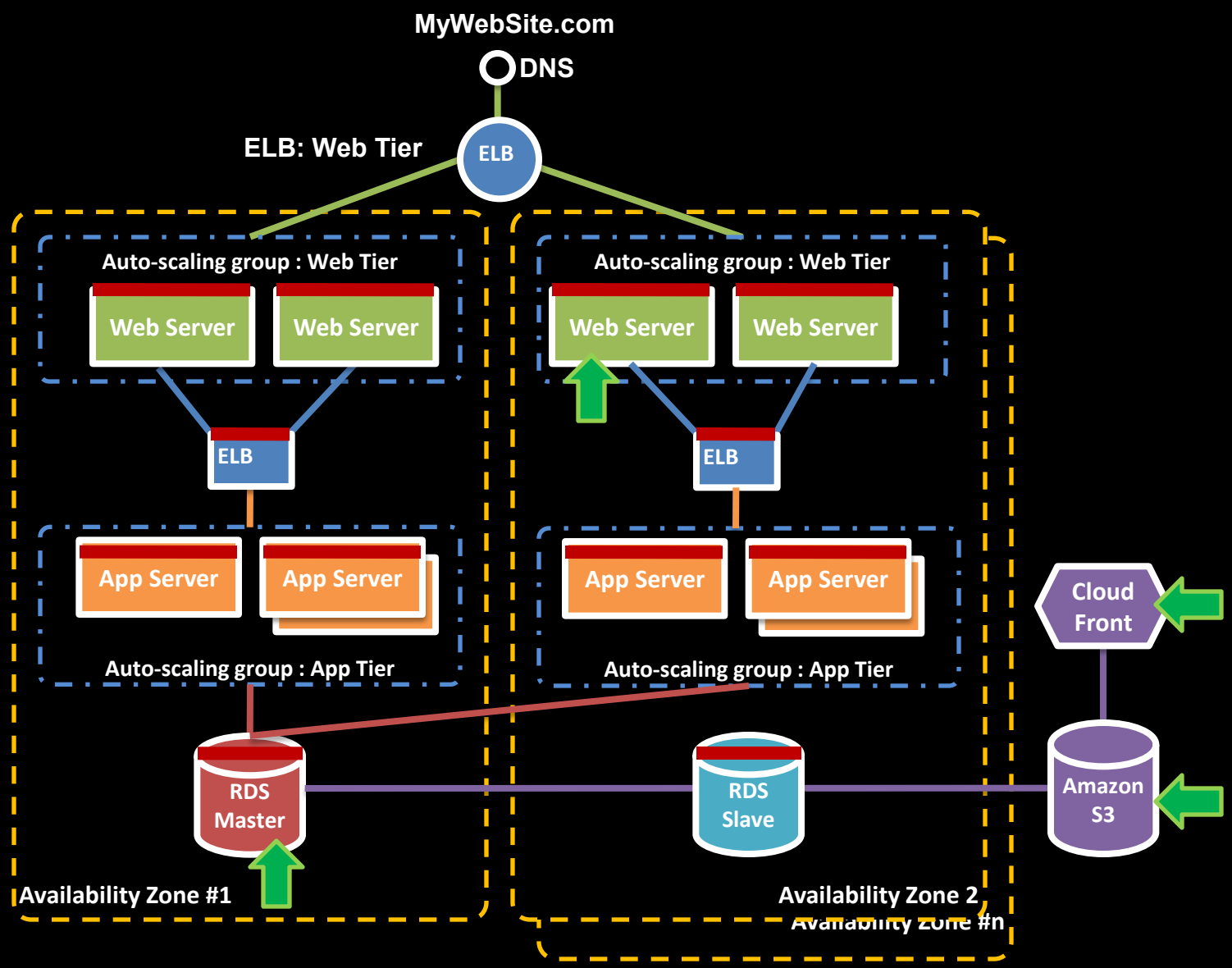
A Scalable Web Architecture on AWS

Build Security in every layer



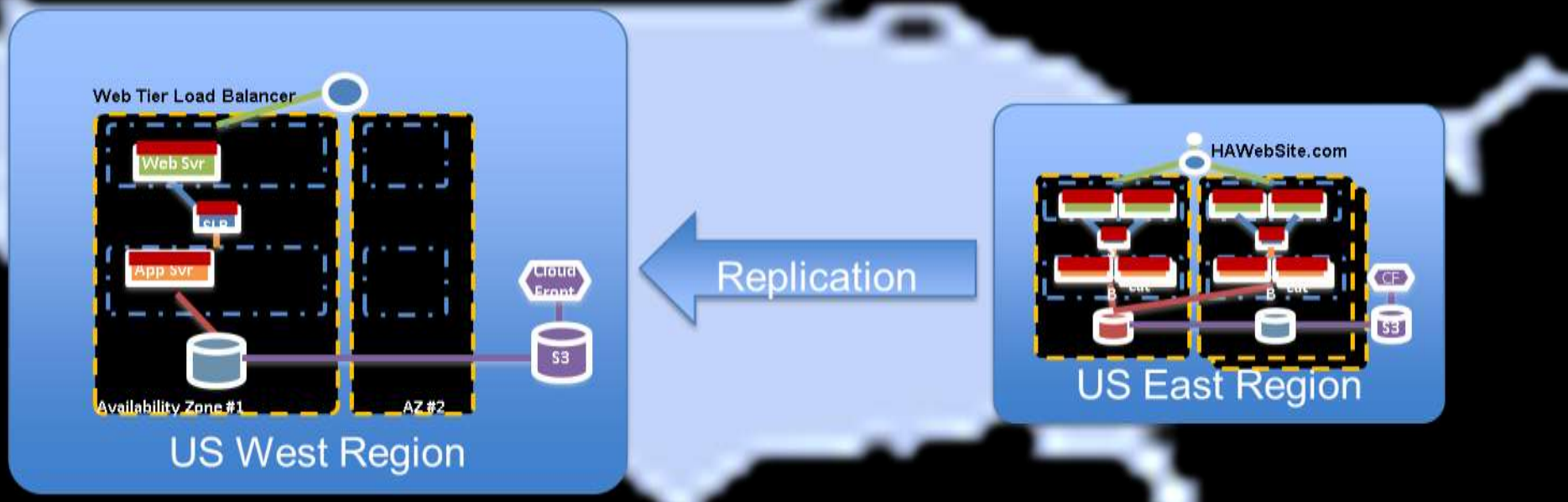
A Scalable Web Architecture on AWS

Leverage many storage options



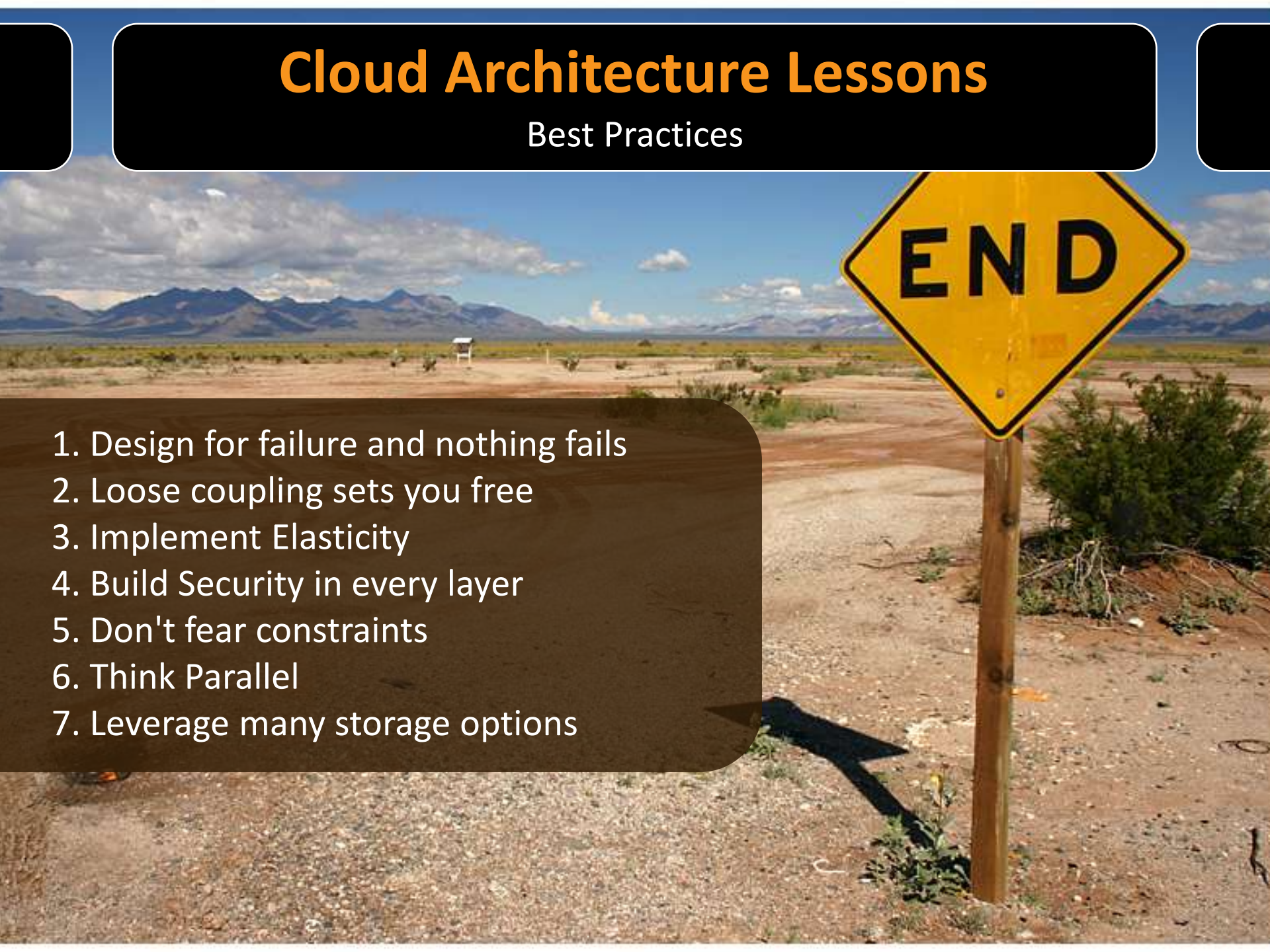
A Scalable Web Architecture on AWS

Properly Architected Application for DR



Cloud Architecture Lessons

Best Practices



1. Design for failure and nothing fails
2. Loose coupling sets you free
3. Implement Elasticity
4. Build Security in every layer
5. Don't fear constraints
6. Think Parallel
7. Leverage many storage options