Errors or corrections? Email us at aws-course-feedback@amazon.com.
Other questions? Email us at aws-training-info@amazon.com.

Architecting on AWS

Disaster Recovery and High Availability

High Availability and Disaster Recovery | What we'll cover

1. How High Availability and Disaster Recovery work together
2. Building highly available systems on AWS
3. Best practices for high availability and disaster recovery
4. Common patterns of disaster recovery on AWS

High Availability and Disaster Recovery | How Availability and Disaster Recovery Work Together

1

How High Availability and Disaster Recovery work together

HA and DR

- Think of it as a spectrum
- It's part of a business continuity plan
- It's not an all or nothing proposition
- In the face of internal or external events, how do you…
  - Keep your applications running 24x7
  - Make sure you data is safe
  - Get an application back up after a major disaster

High Availability

- HA is the other end of the spectrum:
  - Rather than recovery with defined RTO/RPO, design for continuous availability
- Goal: application never goes down
  - Eliminate single points of failure
  - “Self Healing” app recovers automatically from component failures
  - Uses graceful degradation if necessary
- Traditional IT model: HA is very expensive, suitable only for absolutely mission-critical apps

High Availability Terms

- Uptime: period when system is available for use
- Downtime (or Outage): period when system is unavailable for normal function or offline
- Graceful Degradation: system continues to be available, but at a reduced level of service or function
- Availability: percentage uptime in a given period (nines)
  - 99.999% (“five nines”) availability = no more than about 5 minutes downtime per year

DR on the spectrum

- Recover from any event
- Recovery Time Objective (RTO)
  - Acceptable time period within which normal operation (or degraded operation) needs to be restored after event
- Recovery Point Objective (RPO)
  - Acceptable data loss measured in time
- Traditional IT model has DR in a second physical site
  - Low end DR: off-site backups
  - High end DR: hot site active-active architecture

How Does AWS Change Traditional DR / HA?

- AWS is useful for low-end traditional DR to high-end HA, but…

- AWS encourages a rethinking of traditional DR / HA practices
  - Everything in the cloud is “off-site” and (potentially) “multi-site”
  - Using multiple sites (multiple AZs) comes largely for free
  - Using multiple geographically-distributed sites (multiple Regions) is significantly cheaper and easier

How Does AWS Change Traditional DR / HA?

- Tends to move the default design point away from “cold” Disaster Recovery toward “hot” High Availability

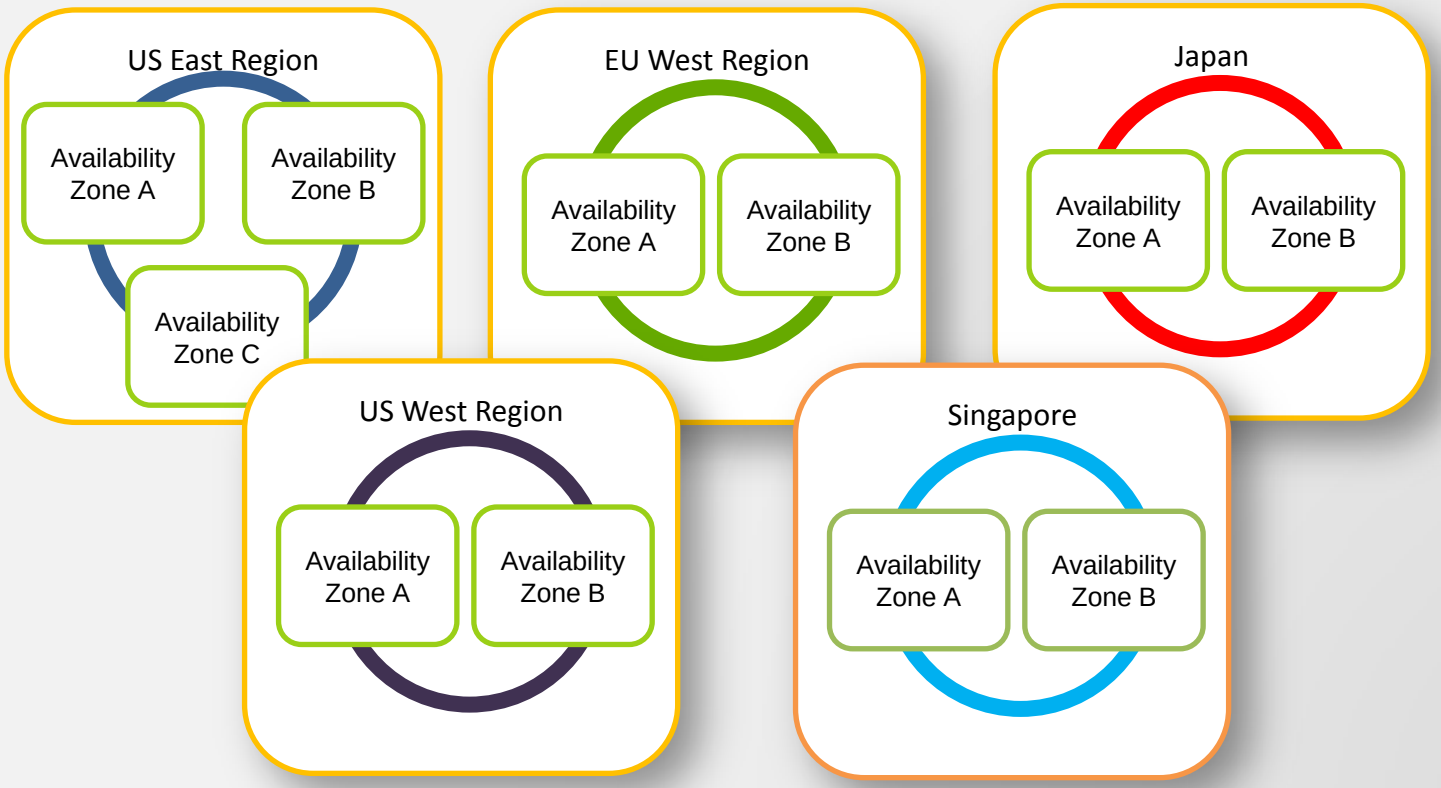
- Makes it easier to stack multiple mechanisms
  - e.g., Basic HA within one Region, DR site in second Region

2

Building highly available systems on AWS

AWS: Regions and Availability Zones

- Regions are completely separate clouds
- Multiple network-connected Availability Zones in each Region



Take advantage of multiple availability zones

- No cost difference between servers running in a single AZ versus multiple AZs
- Most services are designed with multiple AZ use in mind
  (We'll get into details in a moment)

Next step: Building across regions

- Advantages
  - reduce latency
  - increase throughput
- Challenges
  - Most services operate within a single region
    - S3, Auto Scaling, etc.
    - Several tools can help:
      - AMI copy, EBS Snapshot, Route 53, CloudFormation
    - Some AWS services are region-independent
      - Management Console, IAM, CloudWatch

Fault-Tolerant Building Blocks

- Many services are inherently fault-tolerant and highly available:
  - S3, SQS, RDS, DynamoDB, RDS, ELB
- Services for building highly available apps:
  - Availability Zones, Elastic IPs, EBS/Snapshots, ELB, Auto Scaling, Route 53, etc.
  - EC2 (On-Demand and Reserved), AMIs, etc…

AWS Features: Compute

- Amazon Elastic Compute Cloud (EC2)
  - Virtual hosts that can be provisioned momentarily
- AMI – Amazon Machine Image
  - Packaged, reusable functionality: OS + software stack
  - Basic unit of EC2 deployment
- Reserved Instances
  - Reserved capacity when it is needed
  - For further cost savings, can be used for non-critical workloads when not used for DR

AWS Features: Storage

- Amazon Simple Storage Service (S3)
  - Highly-durable file storage for archival and backup

- Elastic Block Store (EBS) and EBS Snapshots
  - Persistent Data volumes for EC2 instances
  - Redundant within a single Availability Zone
  - Snapshot backups provide long-term durability, and volume sharing / cloning capability within a Region

AWS Features: Data Migration

- AWS Storage Gateway
- Amazon VM Import
- Amazon Import/Export
- Other Mechanisms
  - Backup systems
  - Replication (mirroring, db replication, log shipping, etc.)
  - Managed File Transfer products
  - Scripted rsync, tsunami, etc.

AWS Features: Networking

- Amazon Virtual Private Cloud (VPC) & Direct Connect
  - VPC provisions an isolated area within EC2
  - DX provides dedicated fiber cross-connect

- Elastic IP Addresses (EIP)
  - Public Static IP addresses allocated to AWS account
  - EIP can be mapped / remapped to any running instance within a Region

- Amazon Route 53
  - Highly available DNS Service
  - Can quickly map URLs or hostnames between Regions
  - Supports Weighted Round Robin (even across Regions)

AWS Features: Monitoring / Scaling / Load Balancing

- CloudWatch Monitoring
  - Metrics and Alarms for AWS resources
  - Use in conjunction with open source and third-party monitoring tools
- Auto Scaling
  - Respond to changing conditions by adding or terminating EC2 instances (triggered by CloudWatch alarm)
  - Maintain a fixed number of instances running, replacing them if they fail or become unhealthy
- Elastic Load Balancing (ELB)
  - Distributes incoming traffic across multiple instances, in single AZ or across AZs
  - Sends traffic only to healthy instances

AWS Features: Automation

- "Everything is an API" philosophy enables automation of AWS resources

- AWS CloudFormation

Best Practices for HA

High Availability and Disaster Recovery | Best Practices

3

Best practices for high availability and disaster recovery

Best Practices for HA

- Build loosely coupled systems
  - Queues, load-balance tiers
- Implement elasticity
  - Bootstrapping, load balancing, Auto Scaling, etc…
  - Instance asks: “Who am I and what is my role?”
- Use abstract machine and system representations
  - Build images from recipes, stacks from CloudFormation

Design For Failure – Basic Principles

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

Test Your HA Solution – Chaos Monkey

http://techblog.netflix.com/2010/12/5-lessons-weve-learned-using-aws.html

Test Your HA Solution – Chaos Monkey

- From the Netflix blog:

> One of the first systems our engineers built in AWS is called the Chaos Monkey. The Chaos Monkey's job is to randomly kill instances and services within our architecture. If we aren't constantly testing our ability to succeed despite failure, then it isn't likely to work when it matters most – in the event of an unexpected outage.

- Simple monkey:
  - Kill any instance in the account
- Complex monkey:
  - Kill instances with specific tags
  - Introduce other faults (e.g. connectivity via Security Group)
- Human monkey:
  - Kill instances from the AWS Management Console

http://techblog.netflix.com/2010/12/5-lessons-weve-learned-using-aws.html

Architectural Patterns for Disaster Recovery

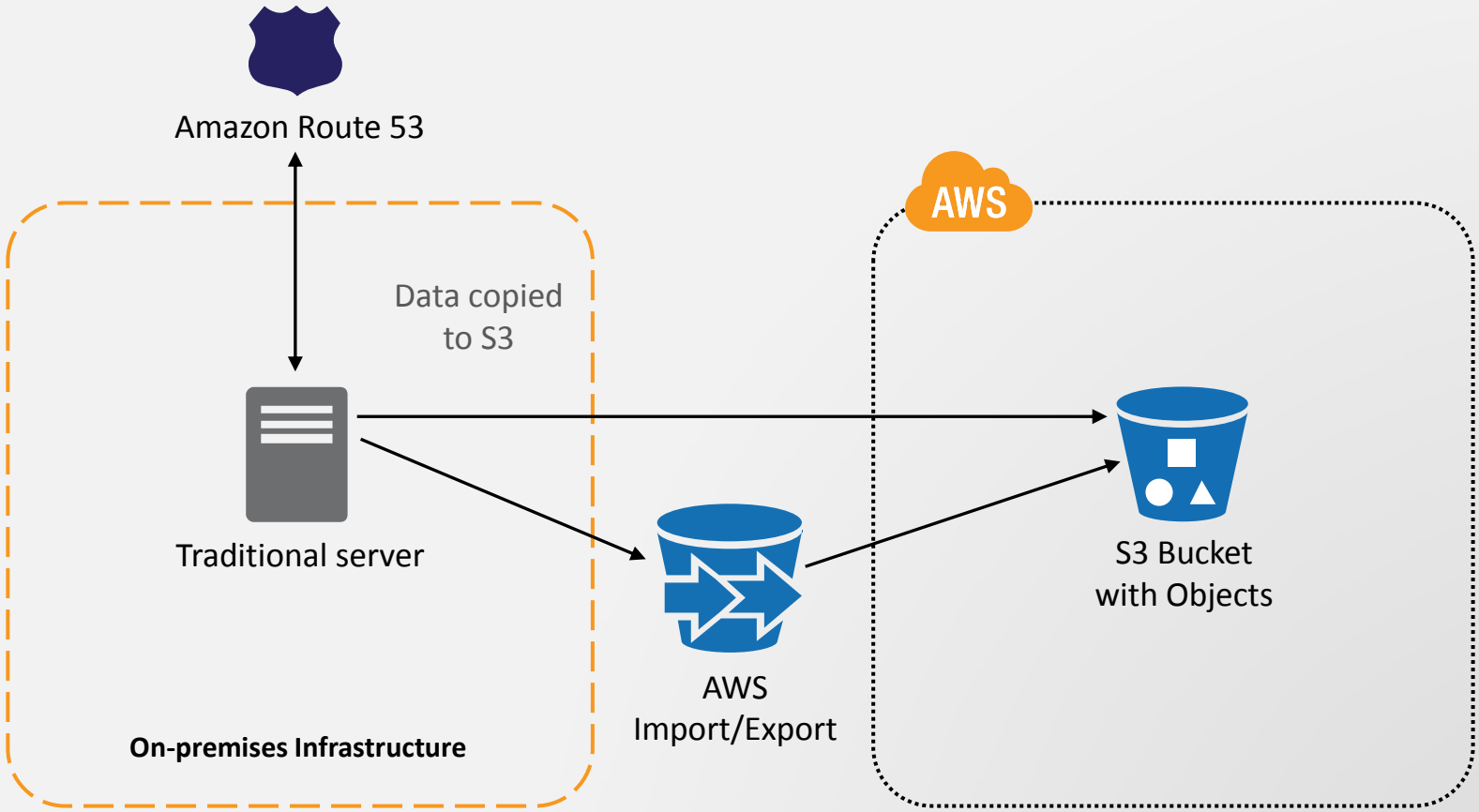
4

Common patterns of disaster recovery on AWS

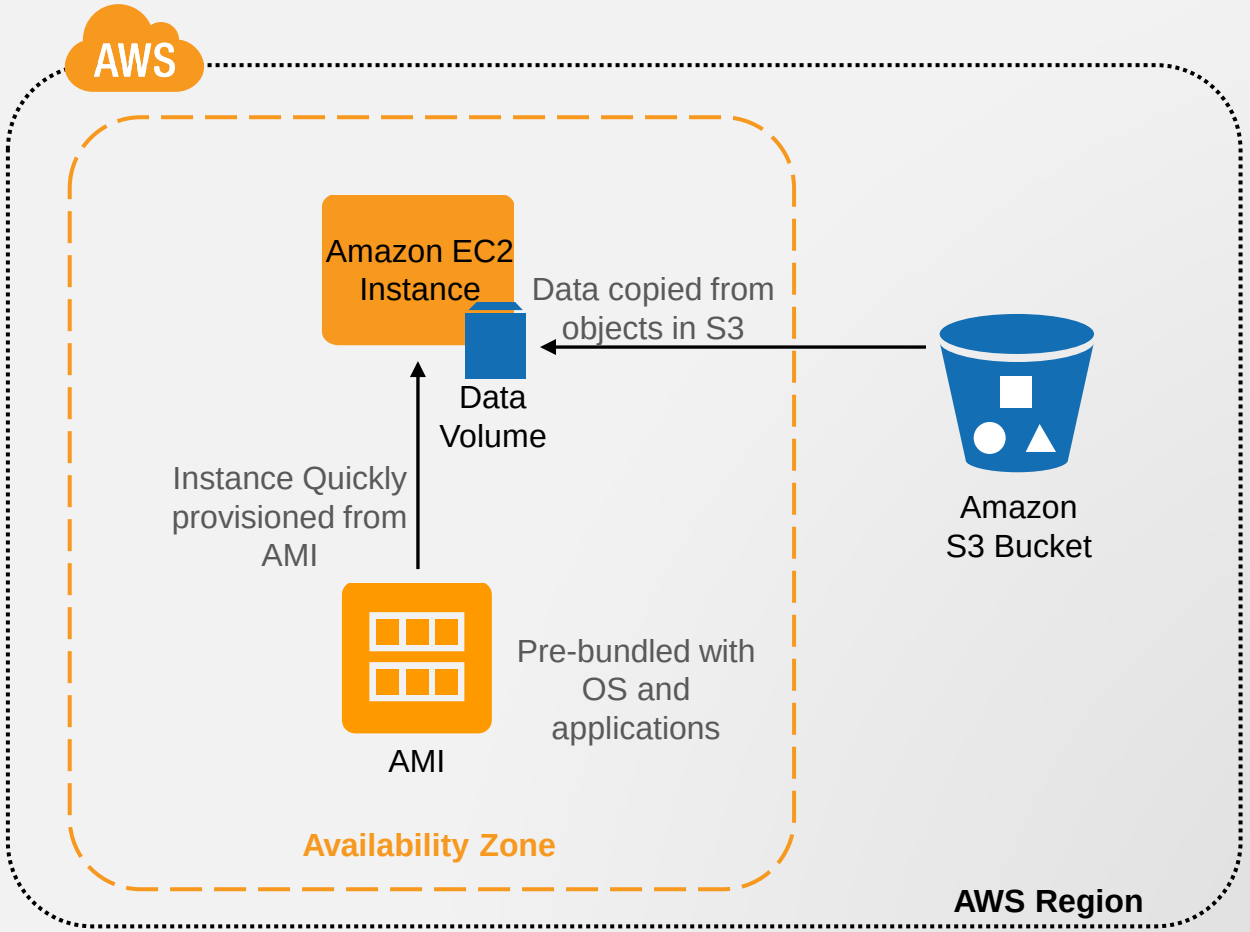
Architectural Patterns Overview

- Example Architectural Patterns (sorted by increasingly optimal RTO/RPO)
  - Backup and Restore
  - Pilot Light
  - Fully Working Low Capacity Standby
  - Multi-Site Hot Standby

Backup and Restore



Backup and Restore



Backup and Restore

- Advantages
  - Simple to get started
  - Extremely cost effective (mostly backup storage)

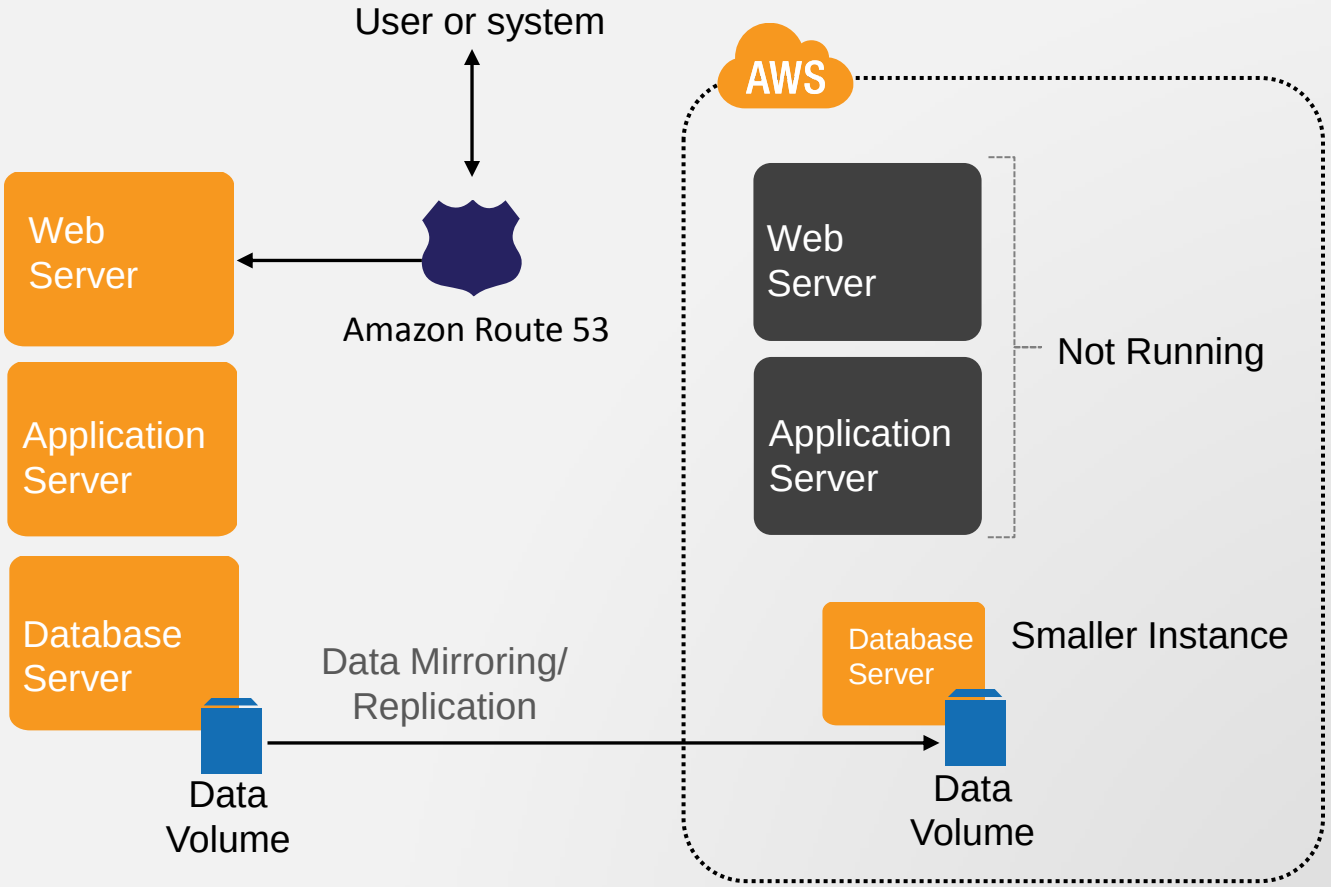
- Preparation Phase
  - Take backups of current systems
  - Store backups in S3
  - Describe procedure to restore from backup on AWS
    - Know which AMI to use, build your own as needed
    - Know how to restore system from backups
    - Know how to switch to new system
    - Know how to configure the deployment

Backup and Restore

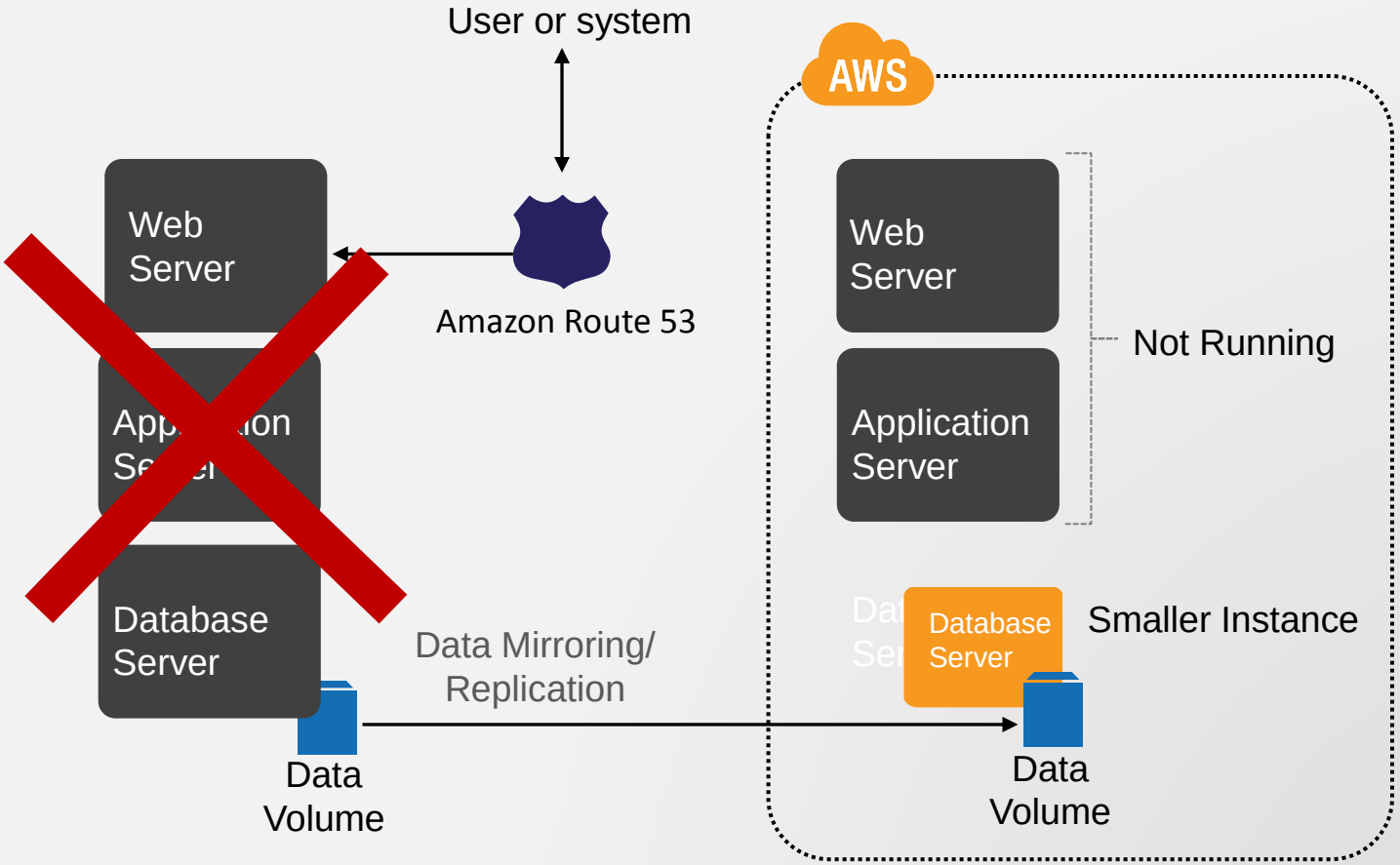
- In Case of Disaster
  - Retrieve backups from S3
  - Bring up required infrastructure
    - EC2 instances with prepared AMIs, Load Balancing, etc.
  - Restore system from backup
  - Switch over to the new system
    - Adjust DNS records to point to AWS

- Objectives
  - RTO: as long as it takes to bring up infrastructure and restore system from backups
  - RPO: time since last backup

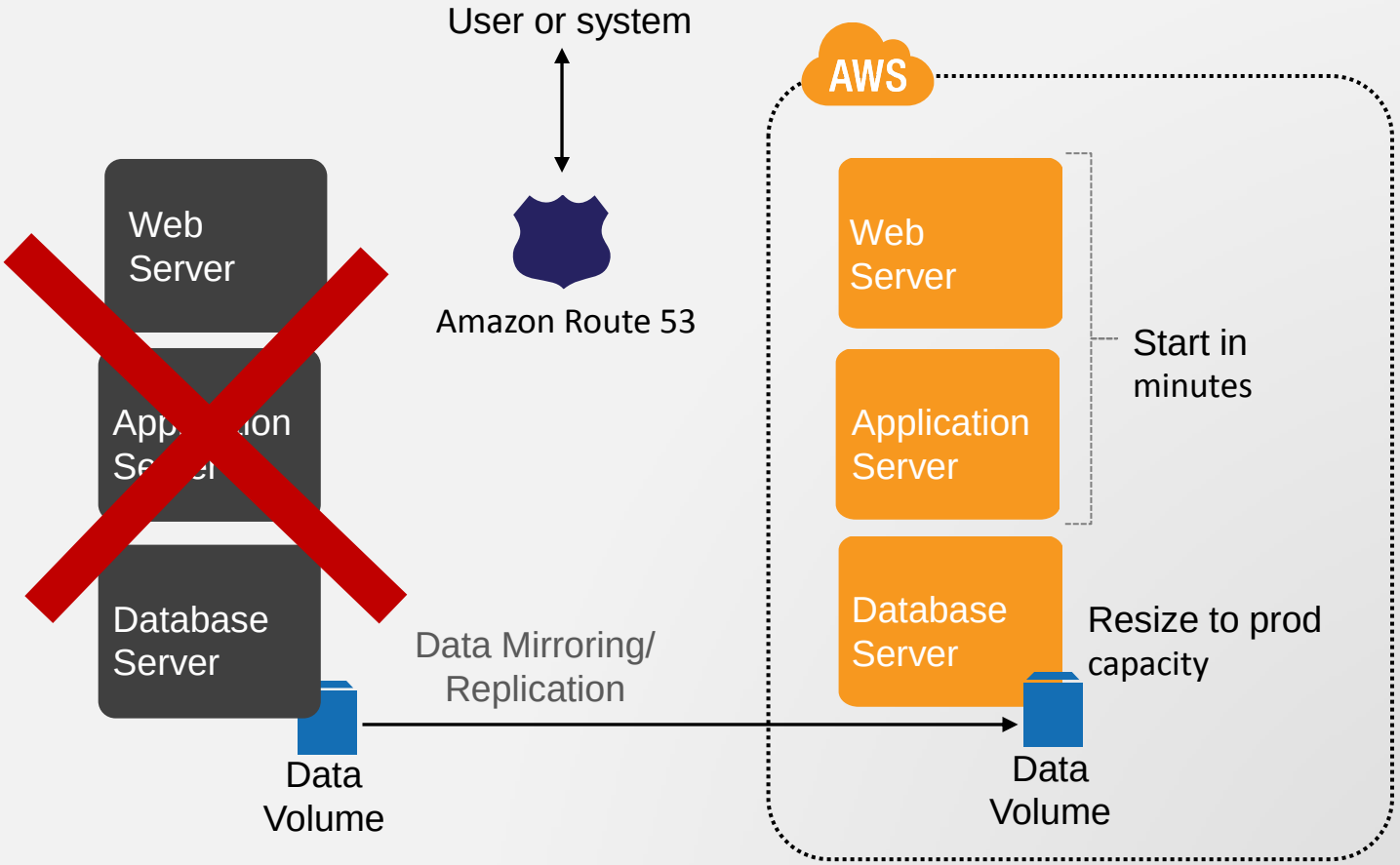
Pilot Light



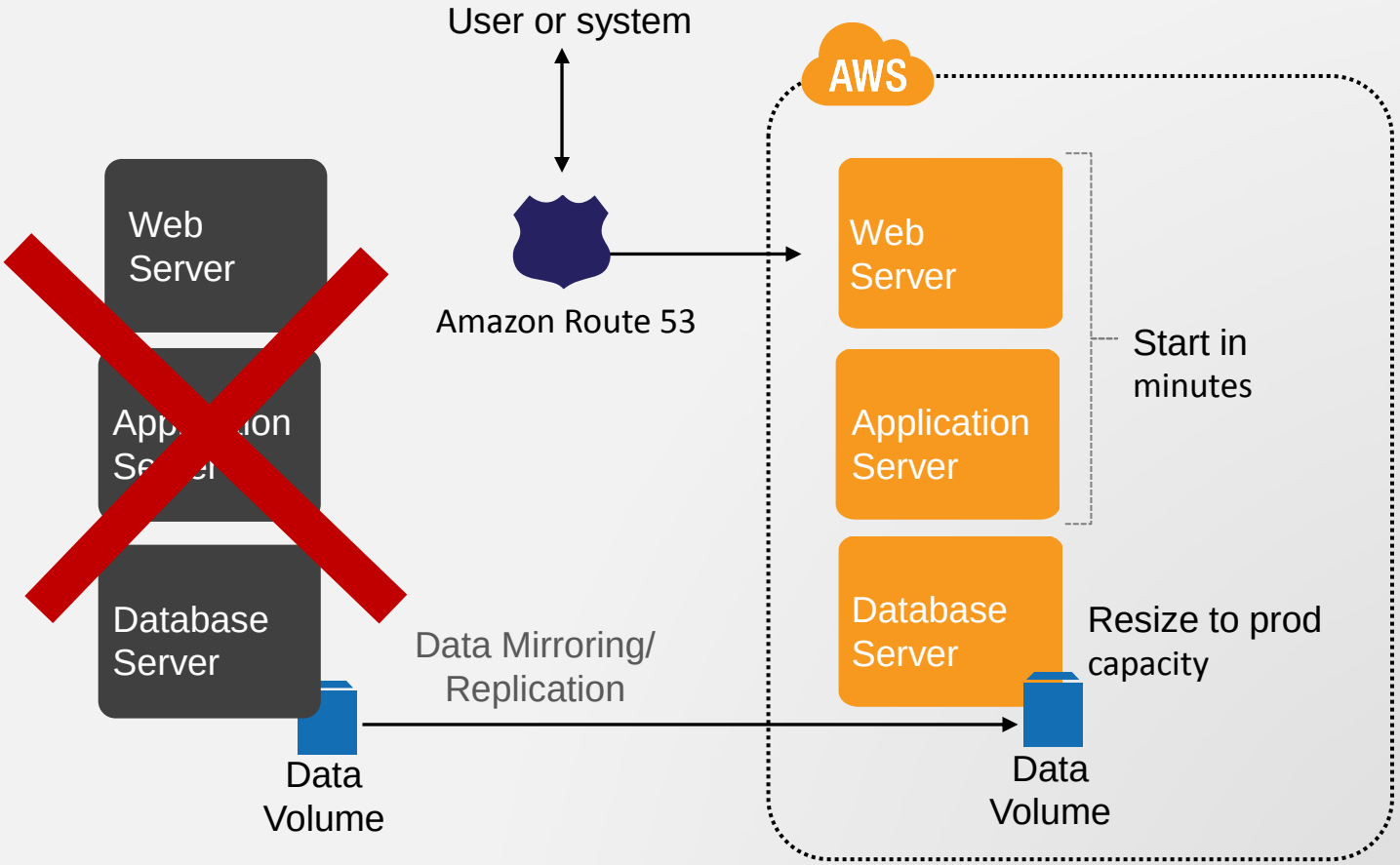
Pilot Light



Pilot Light



Pilot Light



Pilot Light

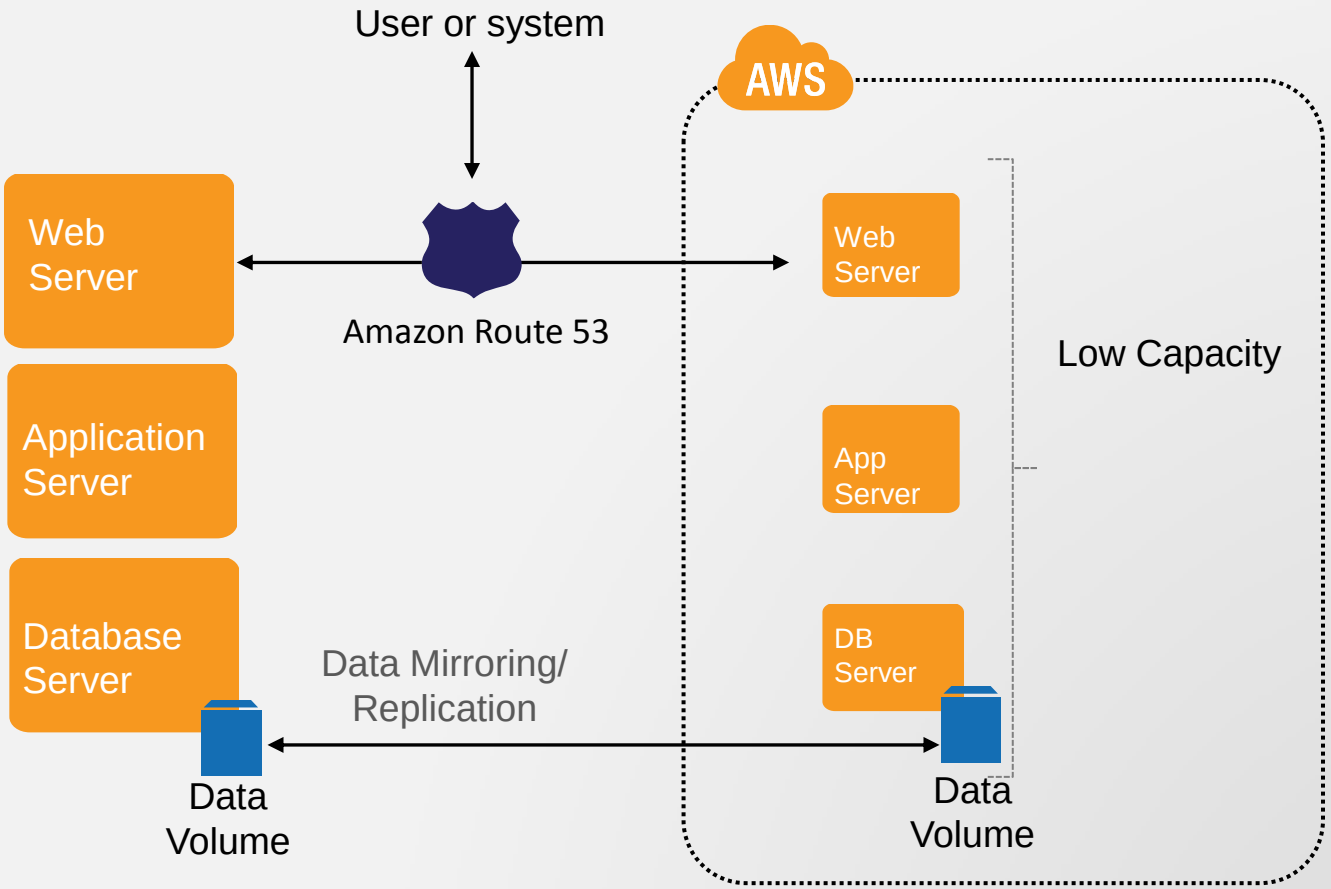
- Advantages
  - Very cost effective (fewer 24/7 resources)

- Preparation Phase
  - Enable replication of all critical data to AWS
  - Prepare all required resources for automatic start
    - AMIs, Network Settings, Load Balancing, etc.
  - Reserved Instances

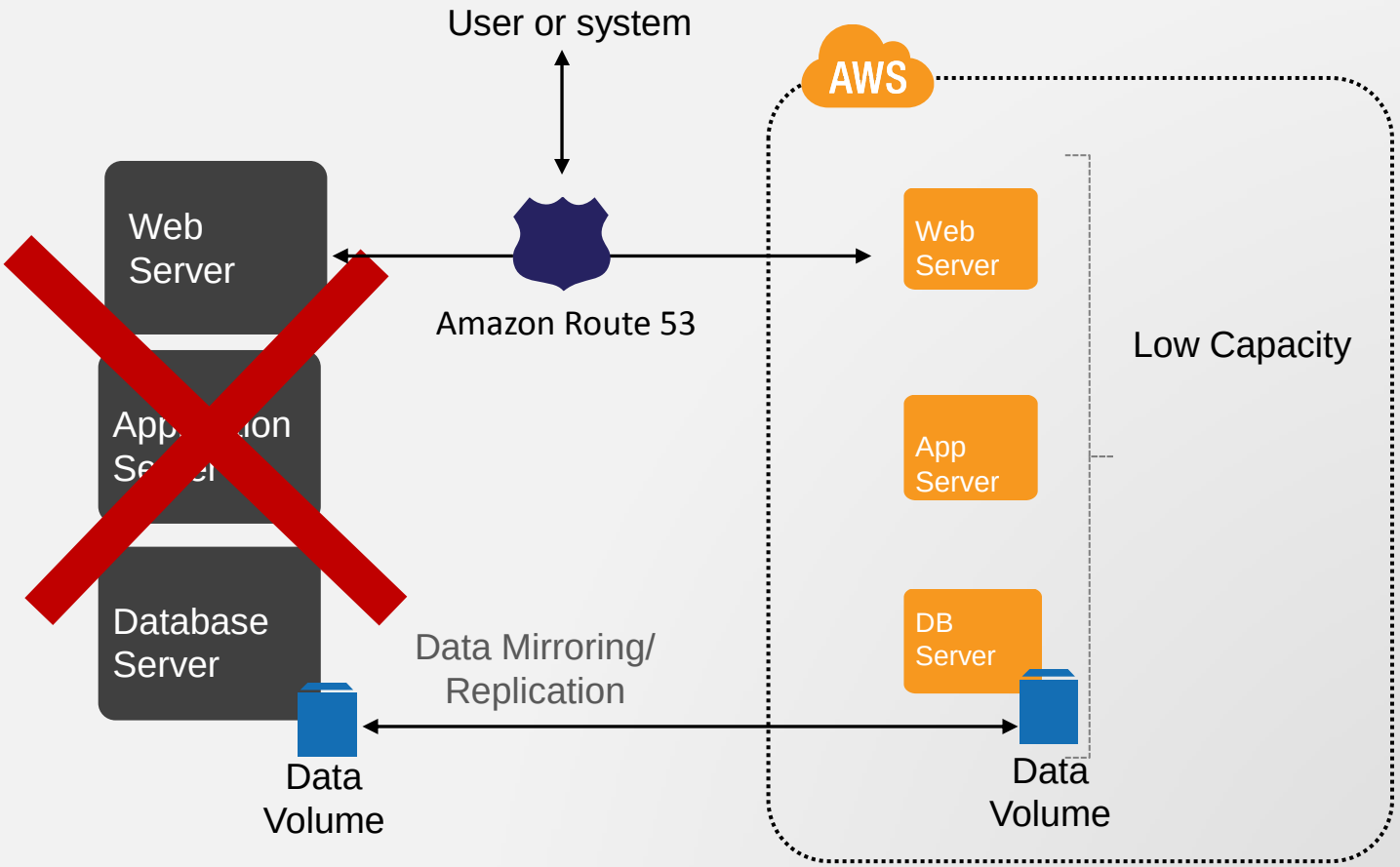
Pilot Light

- In Case of Disaster
  - Automatically bring up resources around the replicated core data set
  - Scale the system as needed to handle current production traffic
  - Switch over to the new system
    - Adjust DNS records to point to AWS
- Objectives
  - RTO: as long as it takes to detect need for DR and automatically scale up replacement system
  - RPO: depends on replication type

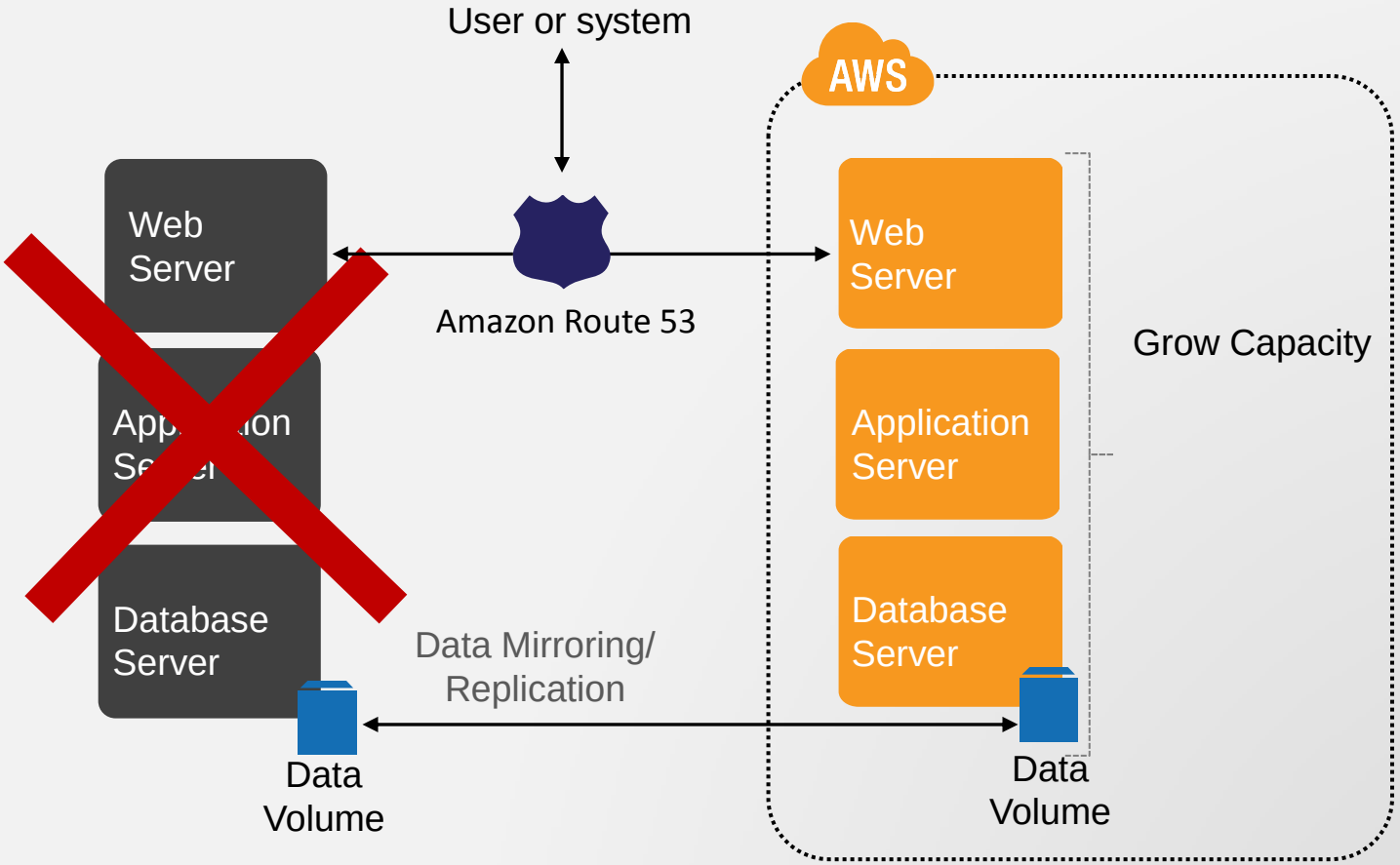
Fully-Working Low Capacity Standby



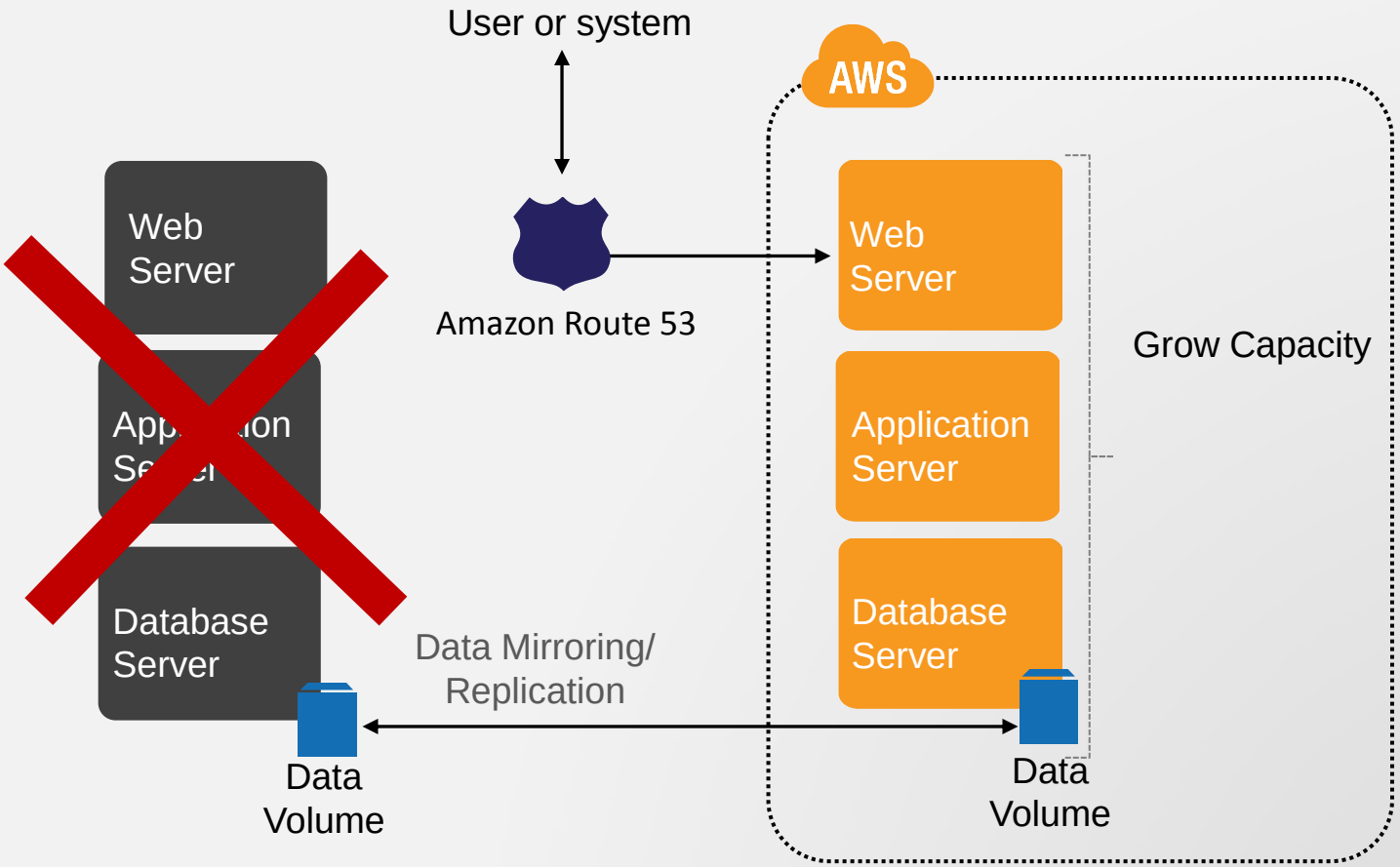
Fully-Working Low Capacity Standby



Fully-Working Low Capacity Standby



Fully-Working Low Capacity Standby



Fully-Working Low-Capacity Standby

- Advantages
  - Can take some production traffic at any time
  - Cost savings (IT footprint smaller than full DR)

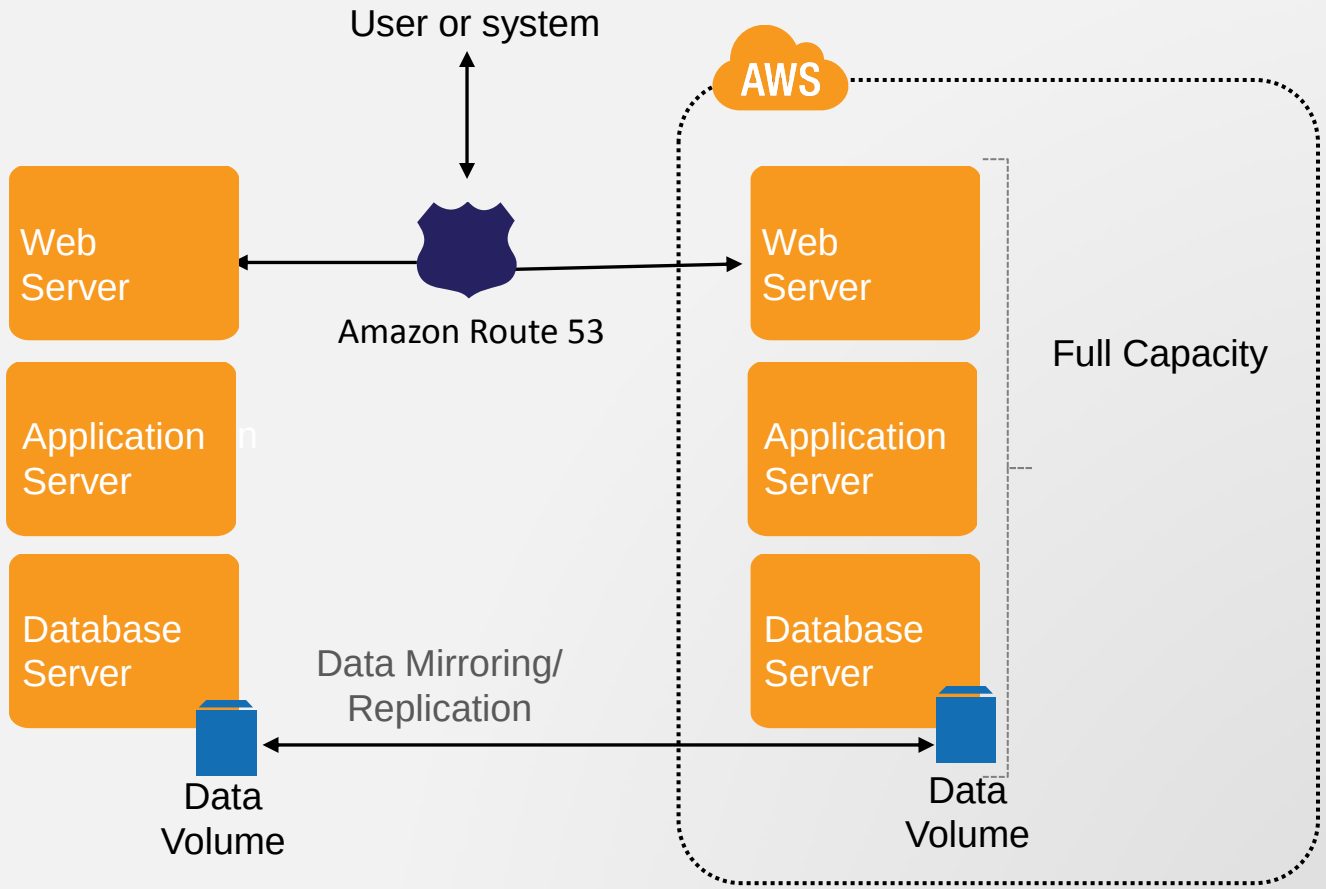
- Preparation
  - Similar to Pilot Light
  - All necessary components running 24/7, but not scaled for production traffic
  - Best practice – continuous testing
    - “Trickle” a statistical subset of production traffic to DR site

Fully-Working Low-Capacity Standby

- In Case of Disaster
  - Immediately fail over most critical production load
    - Adjust DNS records to point to AWS
  - (Auto) Scale the system further to handle all production load

- Objectives
  - RTO: for critical load: as long as it takes to fail over; for all other load, as long as it takes to scale further
  - RPO: depends on replication type

Multi-Site Active-Active



Multi-Site Hot Standby

- Advantages
  - At any moment can take all production load
- Preparation
  - Similar to Low-Capacity Standby
  - Fully scaling in/out with production load
- In Case of Disaster
  - Immediately fail over all production load
    - Adjust DNS records to point to AWS
- Objectives
  - RTO: as long as it takes fail over
  - RPO: depends on replication type

Hosted Desktops

- Advantages
  - Replacement of workstations in case of disaster
  - Pay only when used for DR
- Preparation
  - Set up AMIs with appropriate working environment
- In Case of Disaster
  - Launch desktop AMI and resume work
- Objectives
  - RTO: as long as it takes to launch AMI and restore work environment on virtual desktop
  - RPO: depends on state of AMI

Best Practices for Being Prepared

- Start simple and work your way up
  - Backups in AWS as a first step
  - Incrementally improve RTO/RPO as a continuous effort
- Check for any software licensing issues
- Exercise your DR Solution
  - Game Day
  - Ensure backups, snapshots, AMIs, etc. are working
  - Monitor your monitoring system

Conclusion – Advantages of DR with AWS

- Various building blocks available
- Fine control over cost vs. RTO/RPO tradeoffs
- Ability to scale up when needed
- Pay for what you use, and only when you use it (when an event happens)
- Ability to easily and effectively test your DR plan
- Availability of multiple locations world wide
- Hosted desktops available
- Variety of Solution Providers

Putting it Together: Incremental Improvement

- Start with existing on-premise app with traditional DR
- Gradually move DR (and thus the app) to the cloud:
  - Backup (to S3) and Restore
  - “Pilot Light” on AWS for Quick Recovery
  - Fully Working Low Capacity Standby on AWS
- Migrate to primary on AWS, DR on-premise
- Add hot standby in second AWS AZ or Region
- Incrementally add HA features to primary app
- End State: Use HA techniques in-Region, use DR techniques for Region-to-Region resiliency

- http://aws.amazon.com/solutions/solution-providers/
- http://aws.amazon.com/solutions/case-studies/

Questions?