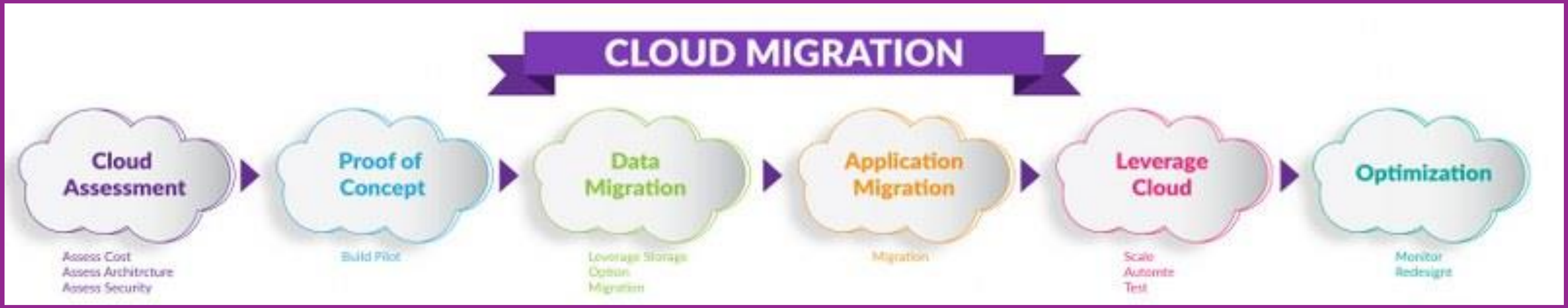
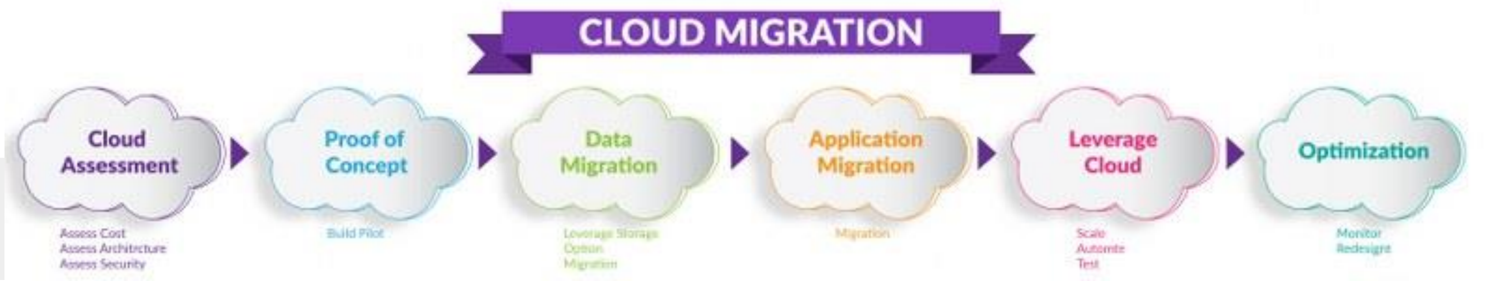
High-Availability in the Cloud Architectural Best Practices



Summary

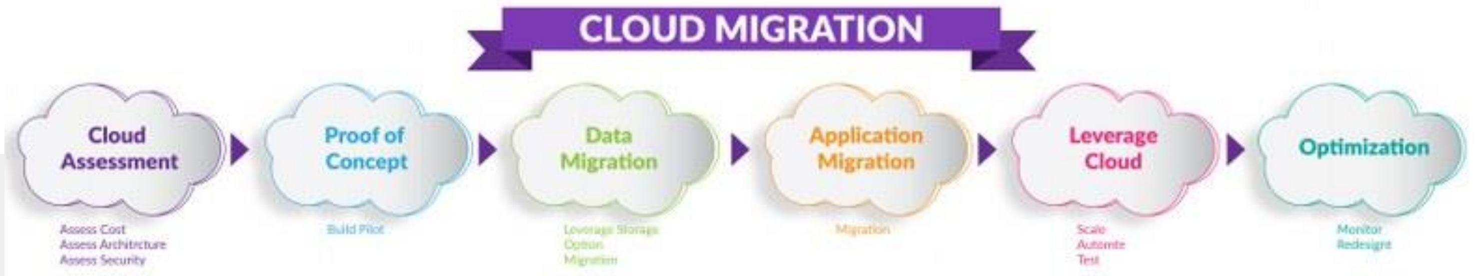
- Resources, Budget, Business Architecture, Process, Data Models
- Follow CAF, Togaf, or something

- Need to design for failure – Netflix eg.
- Use architectural best practices
- Back up and replication need to be taken seriously
- Unsurpassed availability of resources and options are now available



Designing for Failure

- Large scale failures in the cloud are rare but do happen
- Application owners are ultimately responsible for availability and recoverability
- Need to balance cost and complexity of HA efforts against risk(s) you are willing to bear
- Cloud infrastructure has made DR and HA remarkably affordable versus past options
  - Multi-server
  - Multi-AZ
  - Multi-Region
  - Multi-Cloud

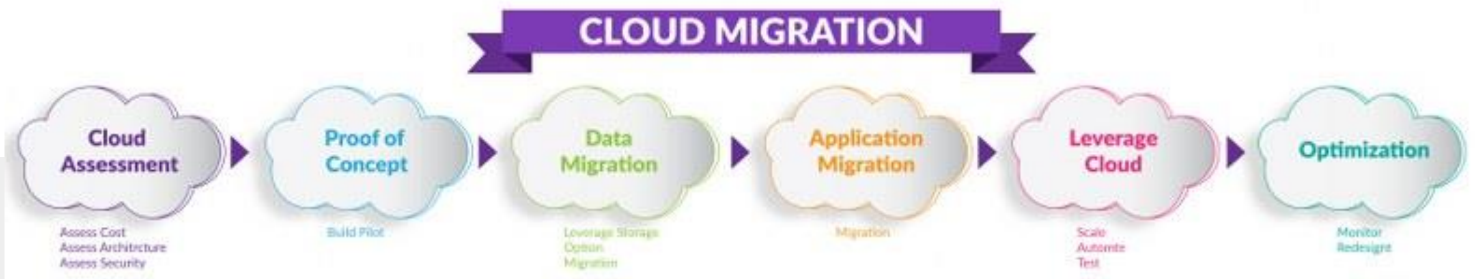


What do we mean by “Cloud”?

- A cloud is a physical datacenter entity behind an API endpoint

- What does that really mean?
  - Amazon Web Services is not a cloud
  - EC2 is not a cloud
  - Eucalyptus, Cloud.com, OpenStack are not clouds
  - EC2 US-East, "my private cloud"… these are clouds
  - An availability zone is not a cloud (but it is part of one)

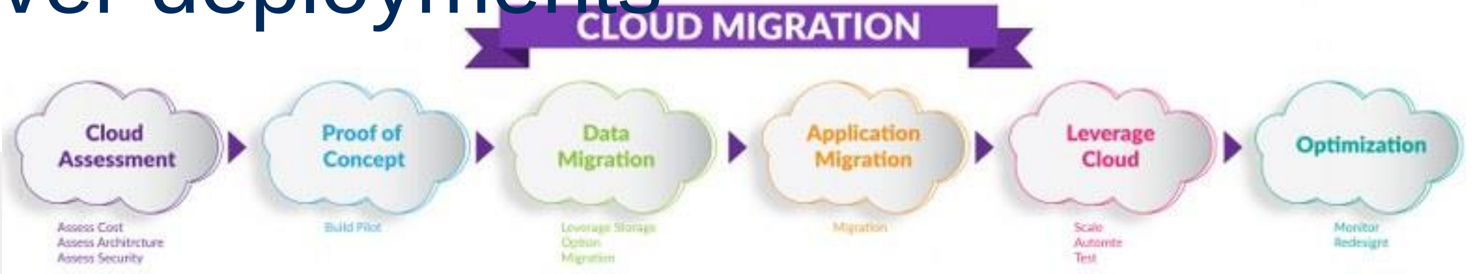
Think of a cloud as a “resource pool” accessed via an API



Multi-cloud Best Practices: ServerTemplates

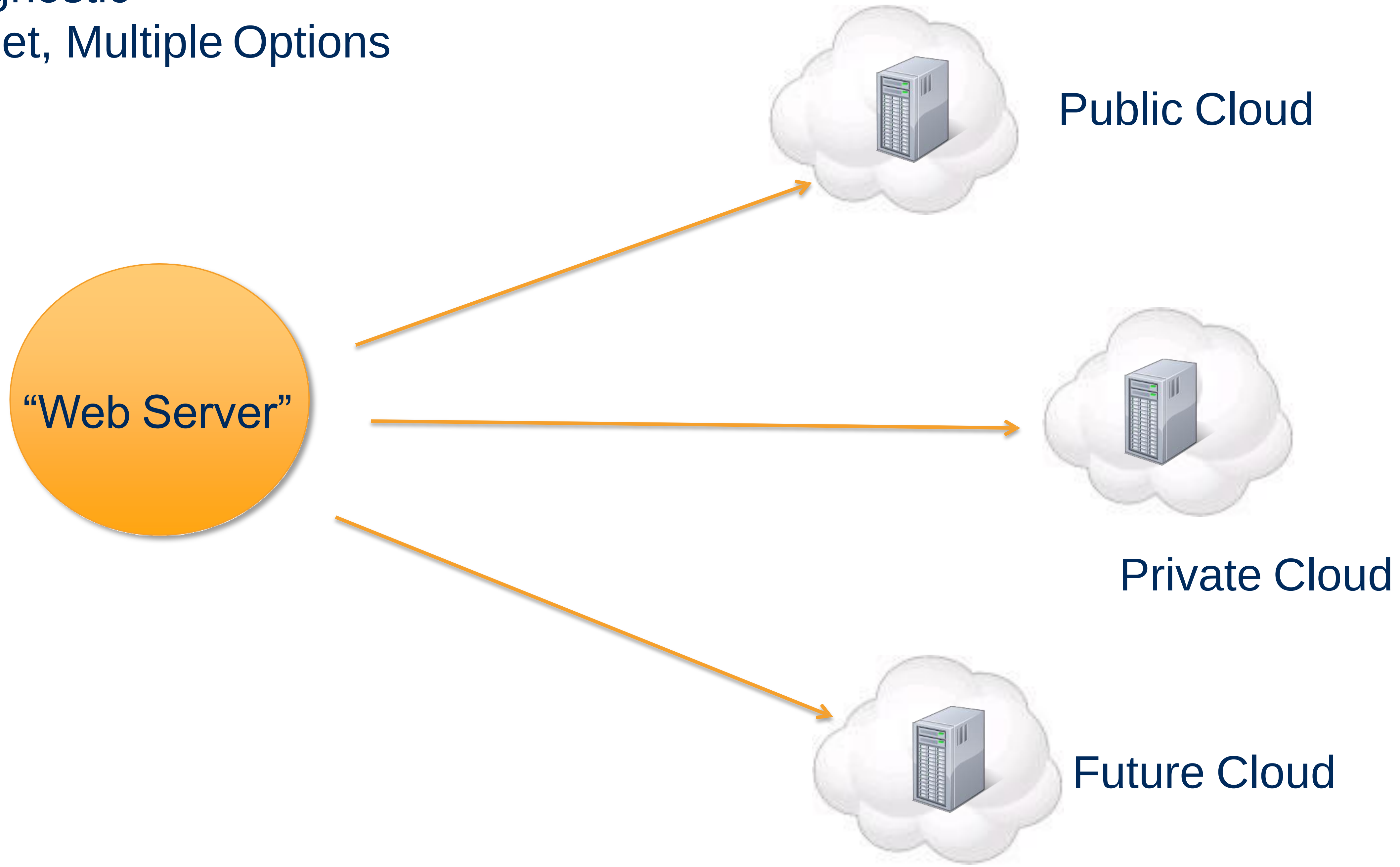
VS.

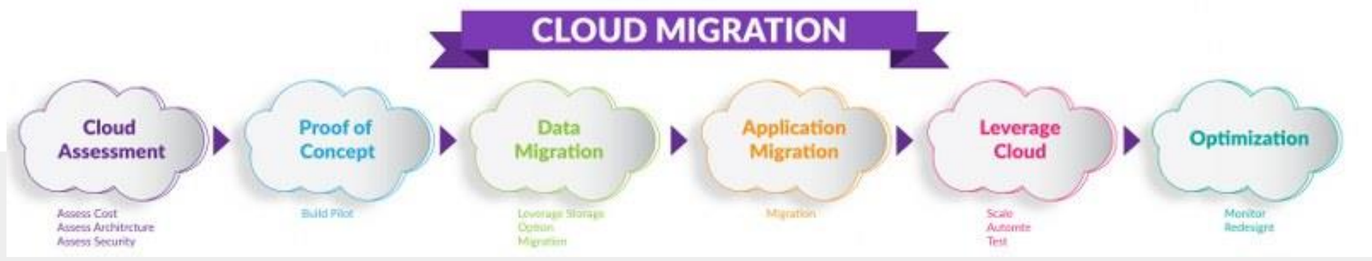
- Integrated approach that puts together all the parts needed to architect single & multi-server deployments



ServerTemplates are Server DNA

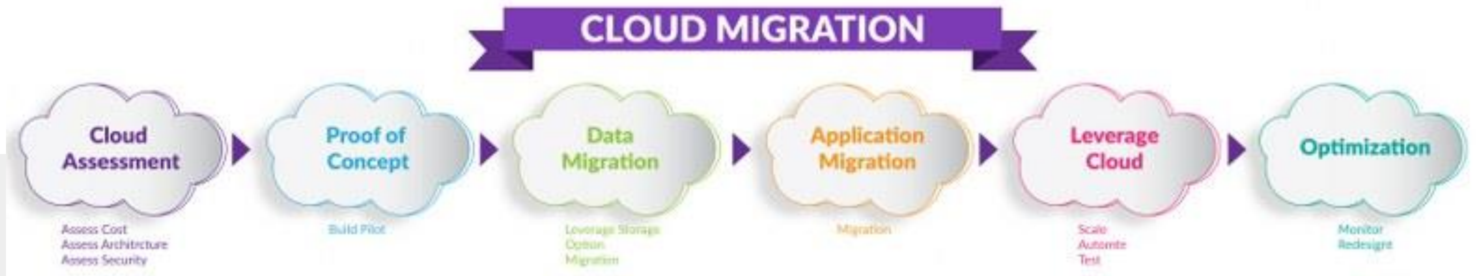
Cloud Agnostic
One Asset, Multiple Options





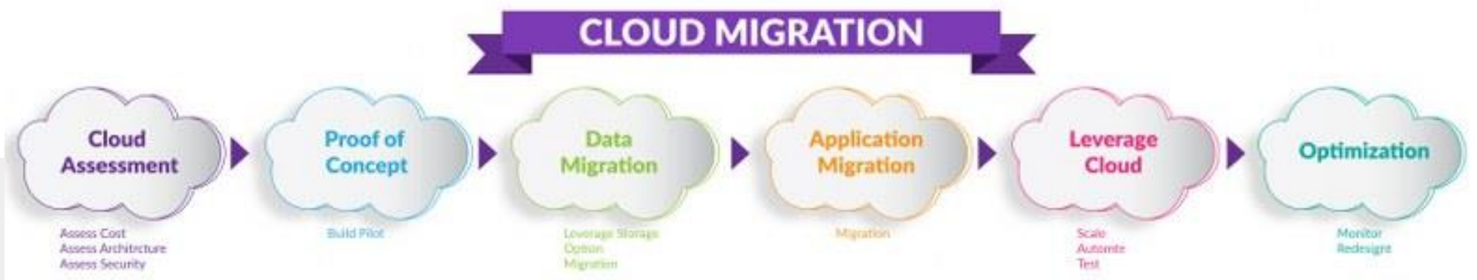
HA/DR Checklist for Risk Mitigation

- ✓ Determine who owns the architecture, DR process and testing.
- ✓ Develop expertise in-house and / or get outside help.
- ✓ Conduct a risk assessment for each application.
- ✓ Specify your target Recovery Time Objective and Recovery Point Objective.
- ✓ Design for failure starting with application architecture. This will help drive the infrastructure architecture.
- ✓ Implement HA best practices balancing cost, complexity and risk.
  - ✓ Automate infrastructure for consistency and reliability.
- ✓ Document operational processes and automations.
- ✓ Test the failover... then test it again.
- ✓ Release the Chaos Monkey.



General HA Best Practices

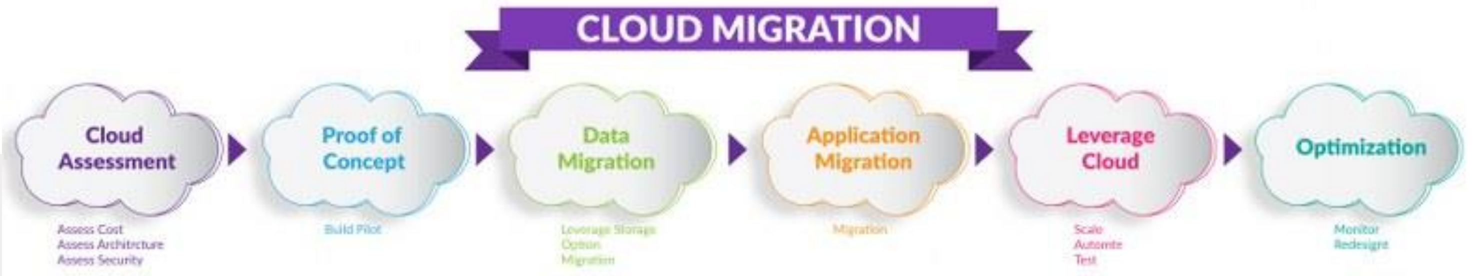
- ✓ Avoid single points of failure
- ✓ Always place (at least) one of each component (load balancers, app servers, databases) in at least two AZs
- ✓ Maintain sufficient capacity to absorb AZ / cloud failures
  - ✓ Reserved Instances – guarantee capacity is available in a separate region/cloud
- ✓ Replicate data across AZs and backup or replicate across clouds/regions for failover
- ✓ Setup monitoring, alerts and operations to identify and automate problem resolution or failover process
- ✓ Design stateless applications for resilience to reboot / relaunch



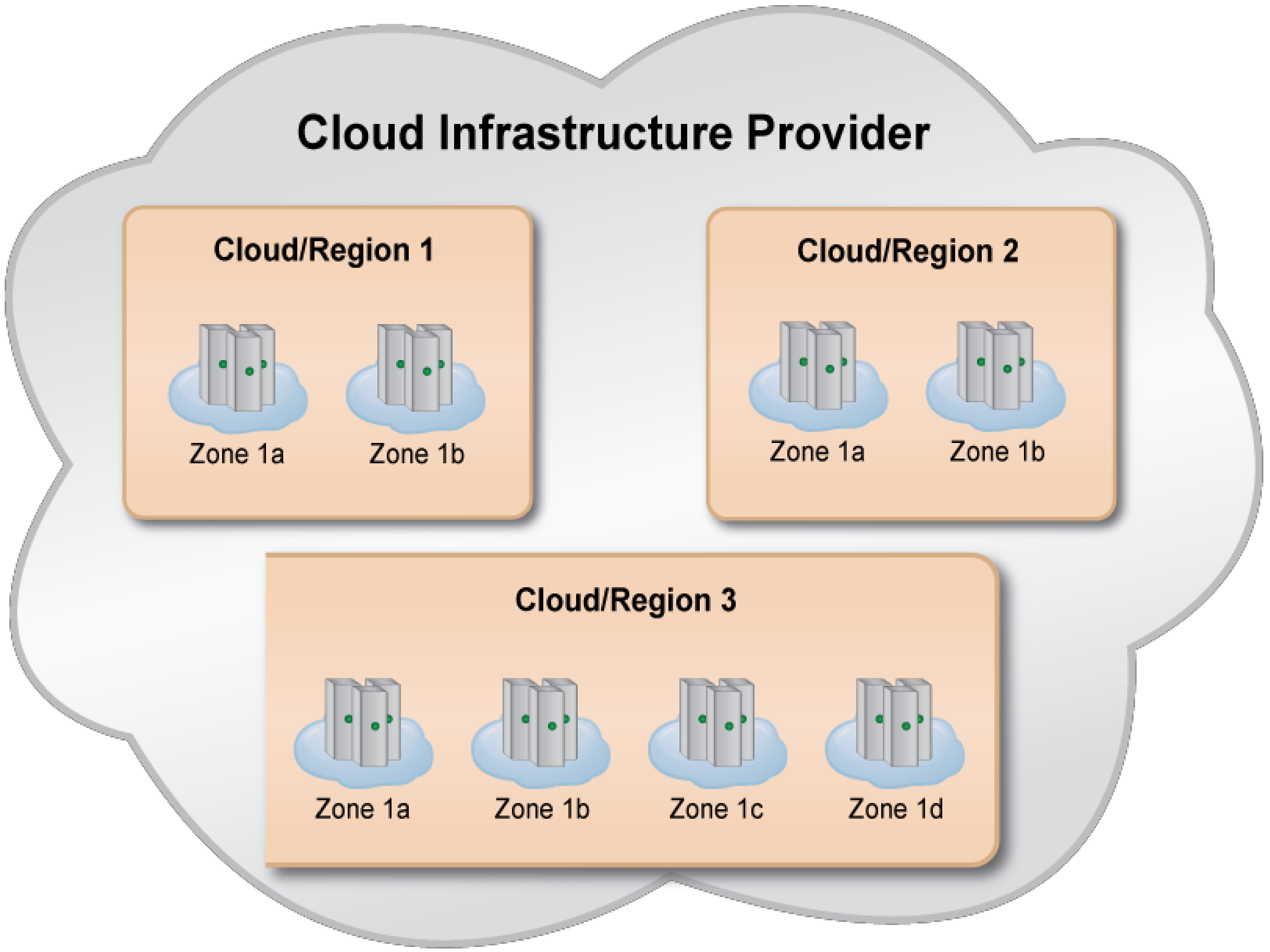
Other items:

Caching
Logging
Session State
Instance costs and Capacity
Proper 3 Tier Architecture
Replication, Read Only
Active-Passive relationships per node
AZ replication
Snapshots

AMIs basic vs ‘golden’ incl. dependencies for that particular environment



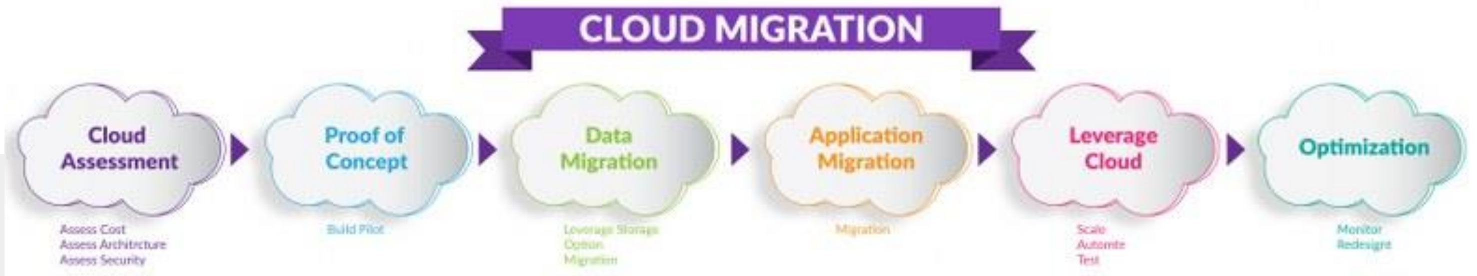
Regions & Availability Zones



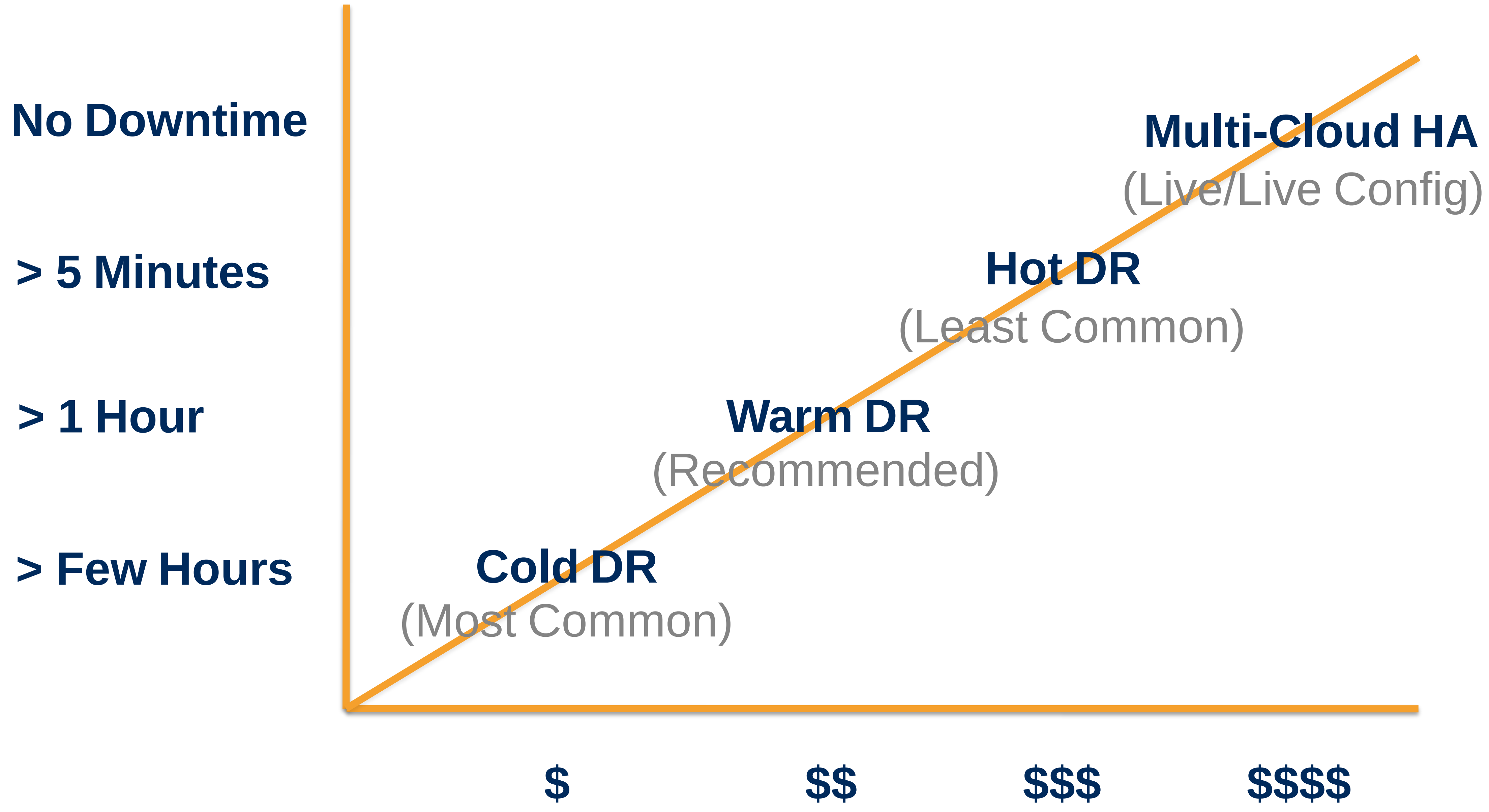
- Zones within a region share a LAN (high bandwidth, low latency, private IP access)
- Zones utilize separate power sources, are physically segregated
- Regions are "islands", and share no resources

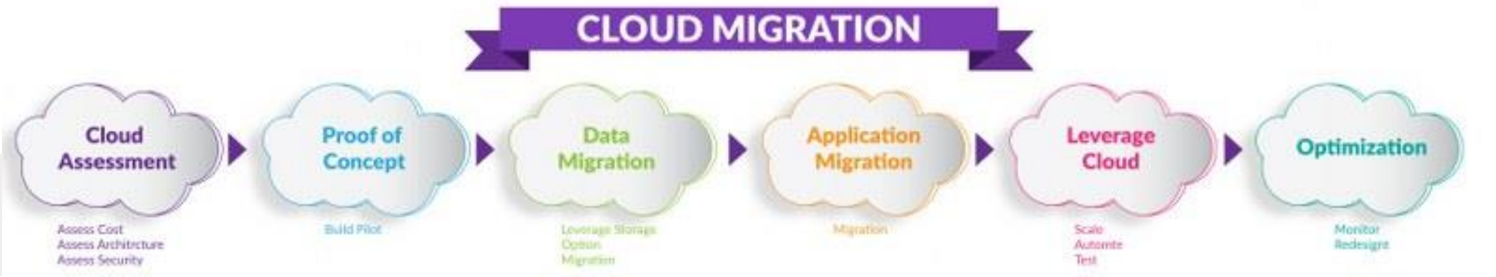
Application Architecture Deployment Options

Component	Options/Considerations
DNS	DNS APIs for dynamic configuration (DynDNS, Route 53, DNS Made Easy)
Load Balancing	HAProxy, Zeus, aiCache, ELB
Storage	Local storage, EBS, S3, CloudFiles, GlusterFS, etc.
Server Array	Scalable tiers; scale up and down conservatively
Database	MySQL, PostgreSQL, SQL Server, RDS, NoSQL



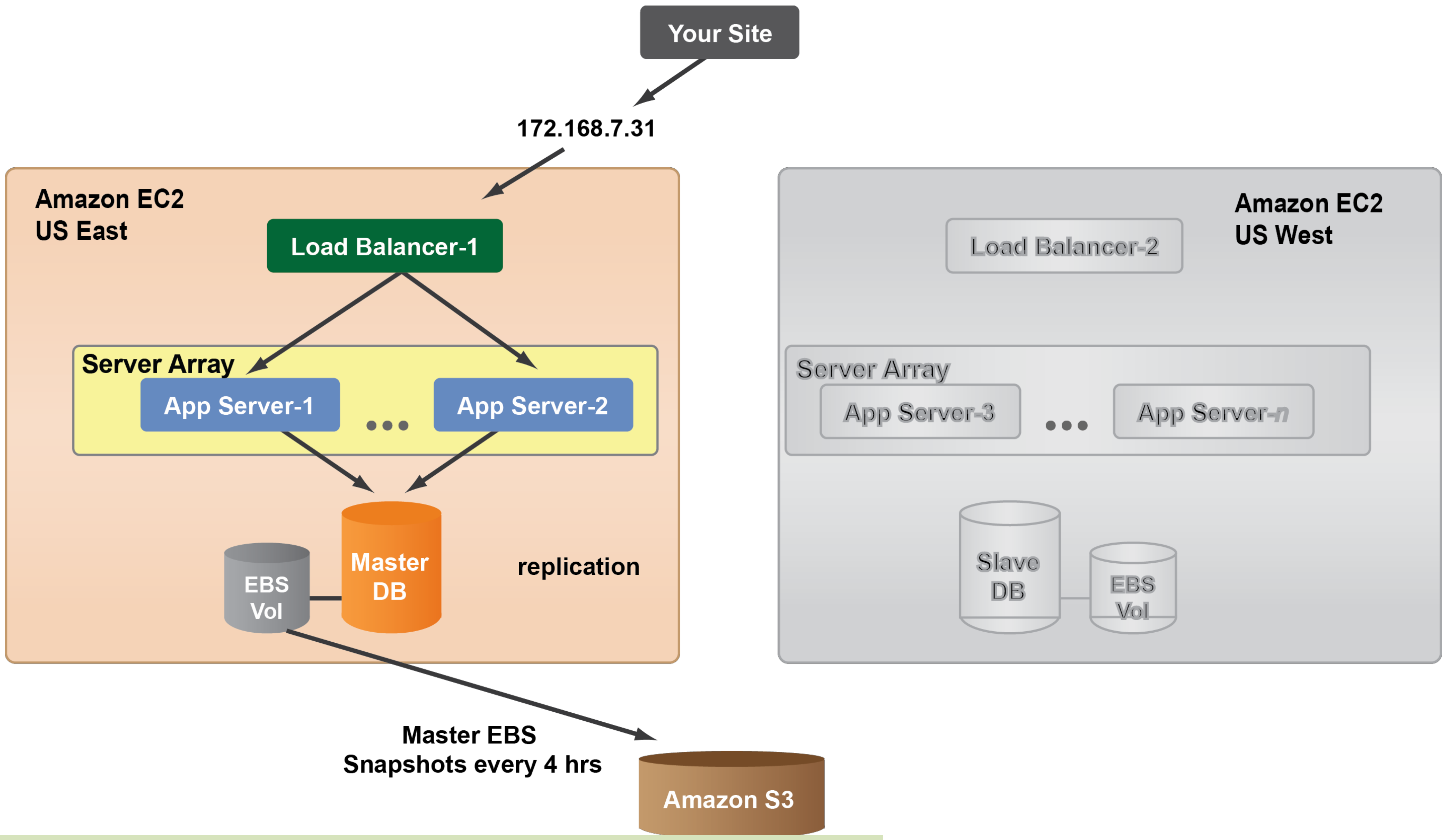
Multi-Cloud Cold / Warm / Hot DR Options



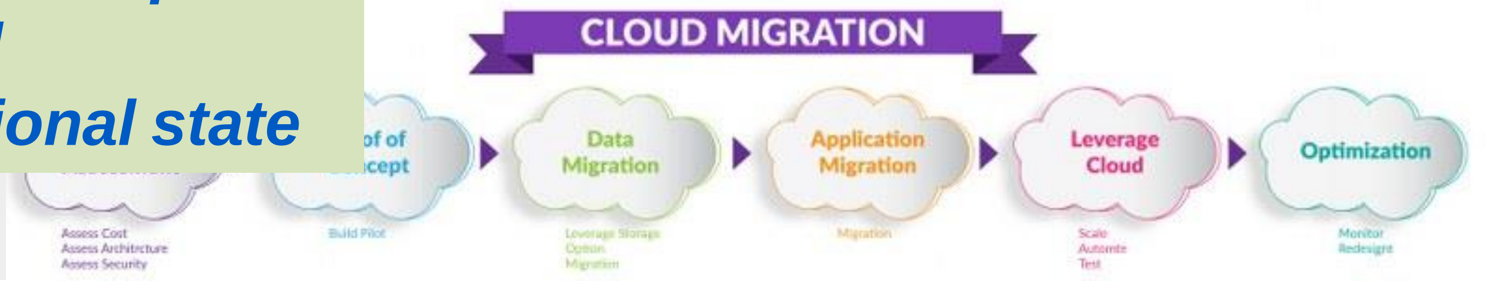


Multi-Cloud Cold DR Example

Staged Server Configuration and generally no staged data

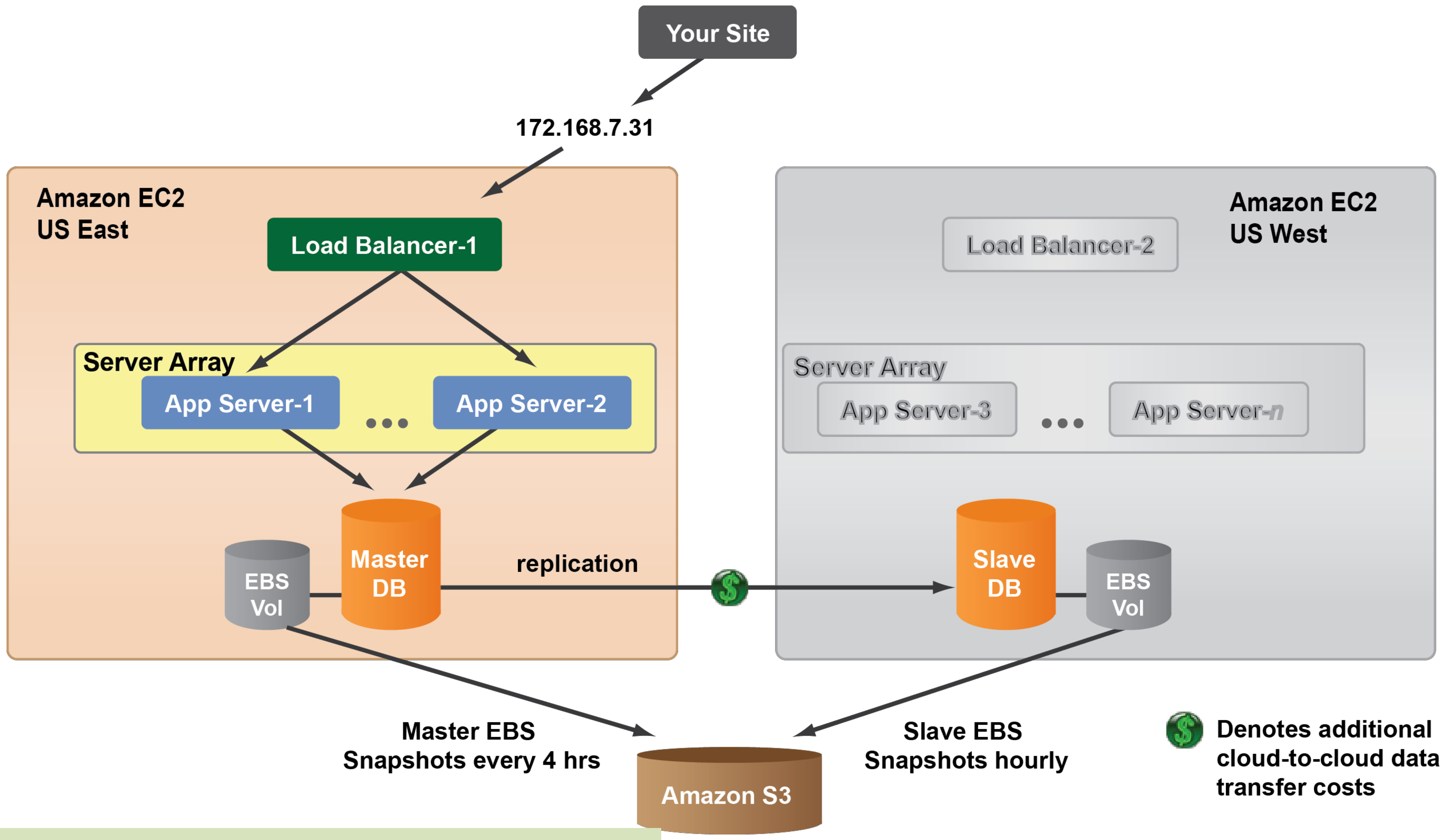


- *Not recommended if rapid recovery is required*
- *Slow to replicate data to other cloud*
- *Slow to bring database to an operational state*

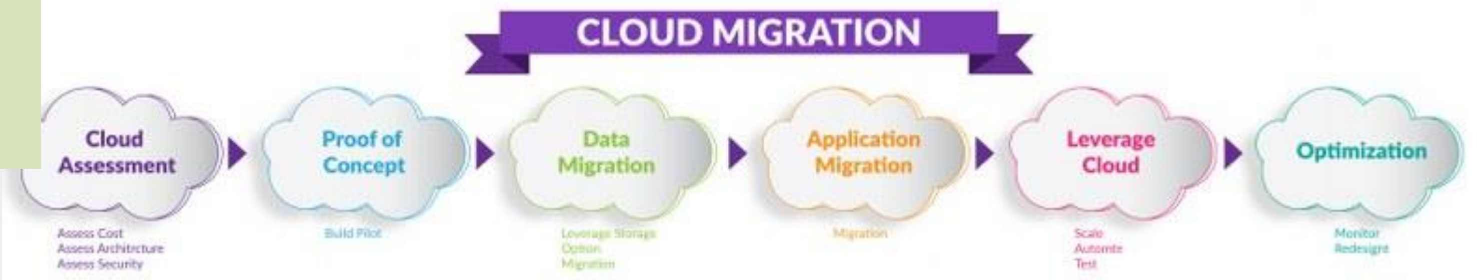


Multi-Cloud Warm DR Example

Staged Server Configuration, pre-staged data and running Slave Database Server

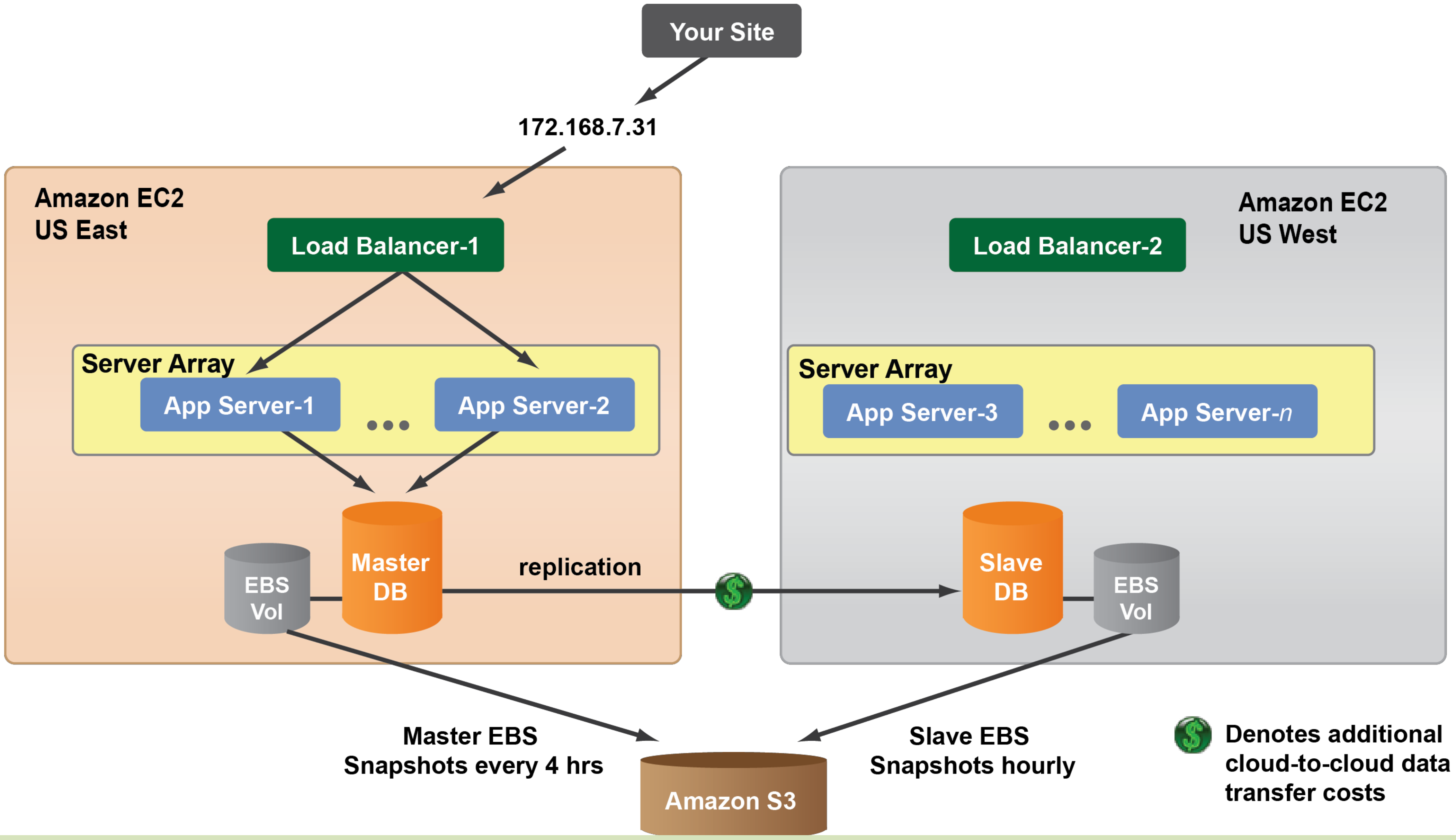


- *Generally recommended DR solution*
- *Minimal additional cost*



Multi-Cloud Hot DR Example

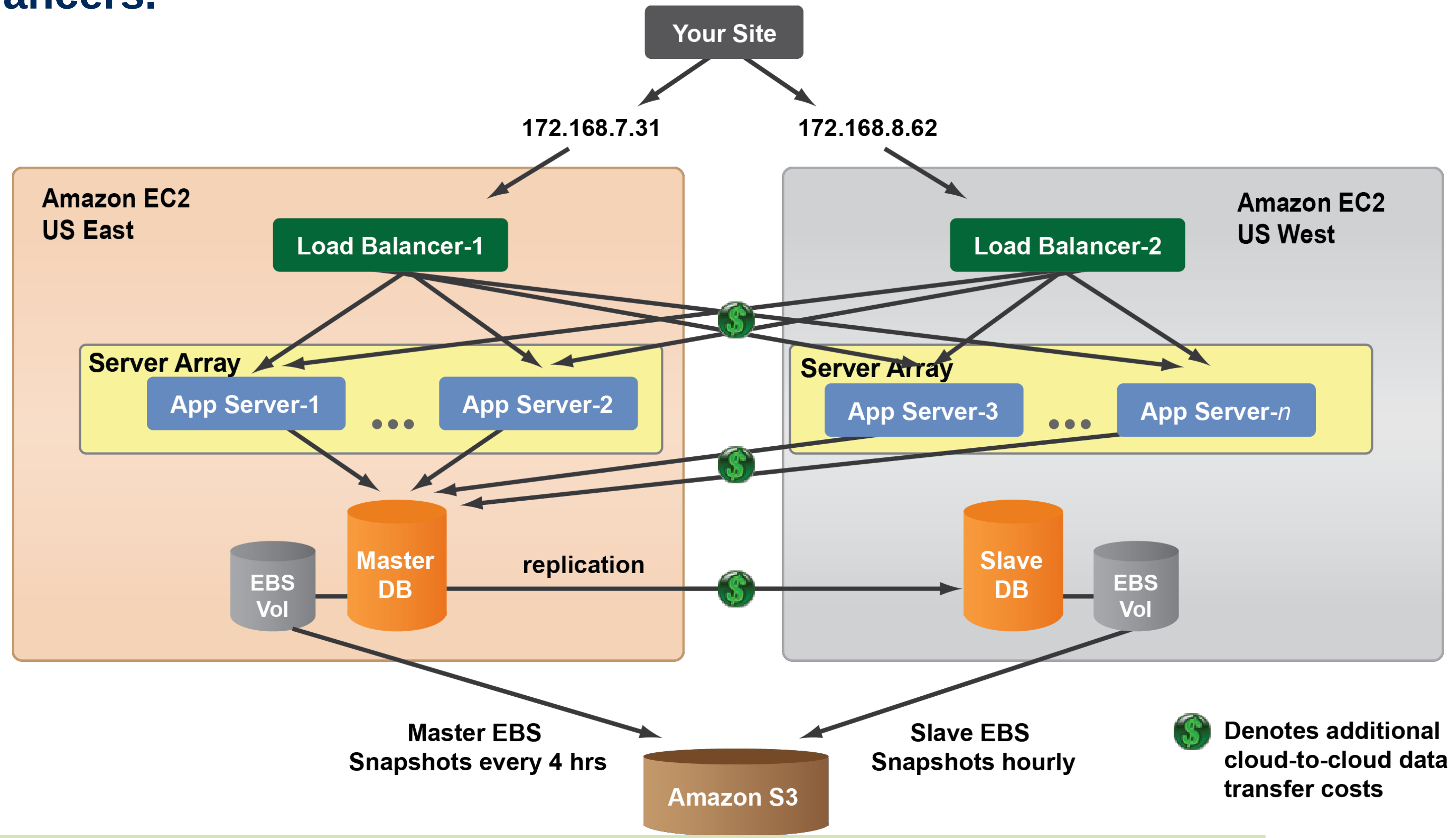
Parallel Deployment with all servers running but all traffic going to primary



- *Not recommended*
- *Very high additional cost*

Multi-Cloud Hot HA Example

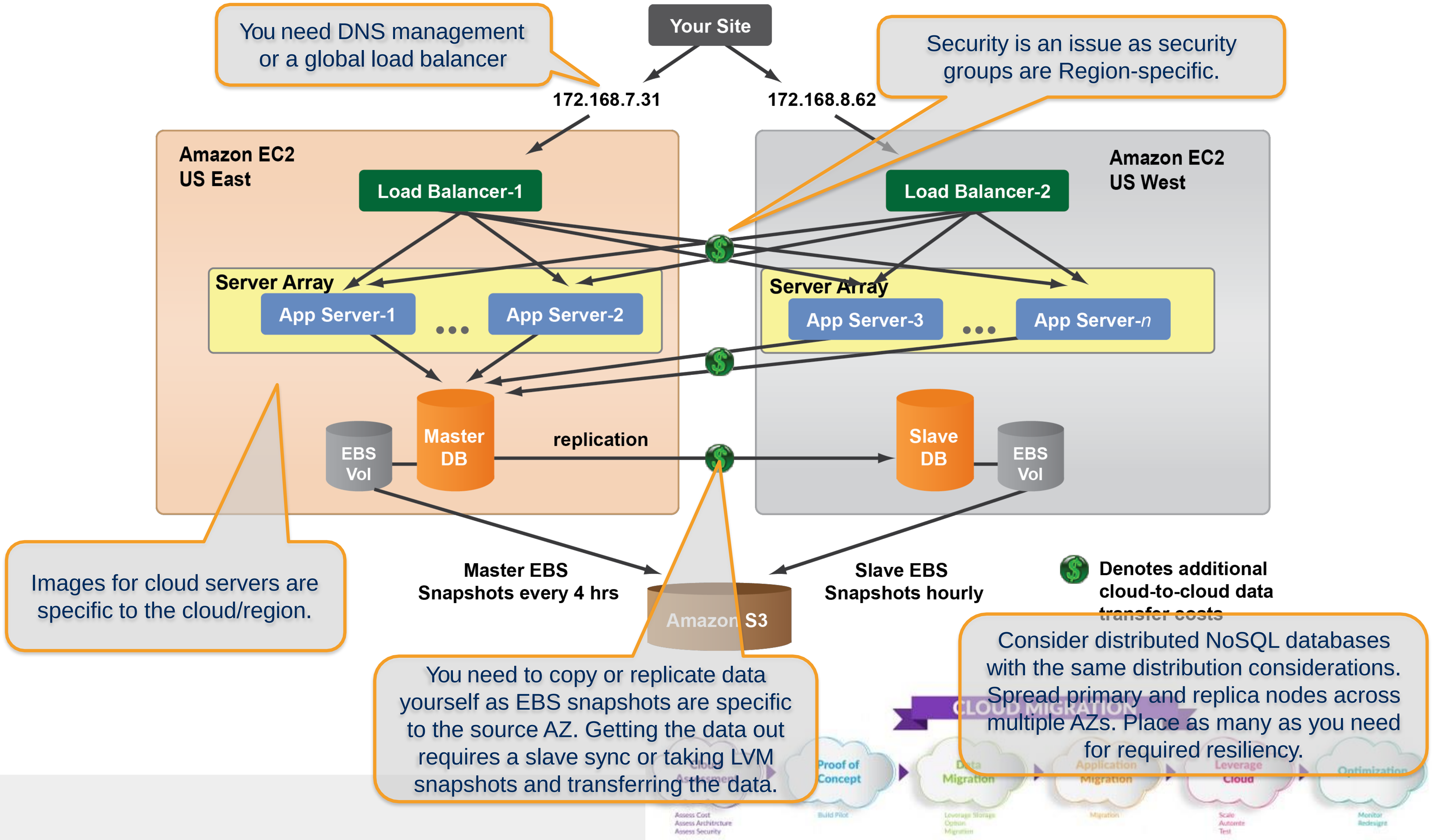
Live/Live configuration. May use Geo-target IP services to direct traffic to regional load balancers.



- *Possible, but not recommended (more to follow…)*
- *Maximum additional cost*
- *Provides high availability, but complex to implement and manage*

Multi-Cloud Hot HA Example

Multi-Cloud looks similar to Multi-AZ… but there are additional problems to solve as some resources are not shared across clouds



You need DNS management or a global load balancer

Security is an issue as security groups are Region-specific.

Images for cloud servers are specific to the cloud/region.

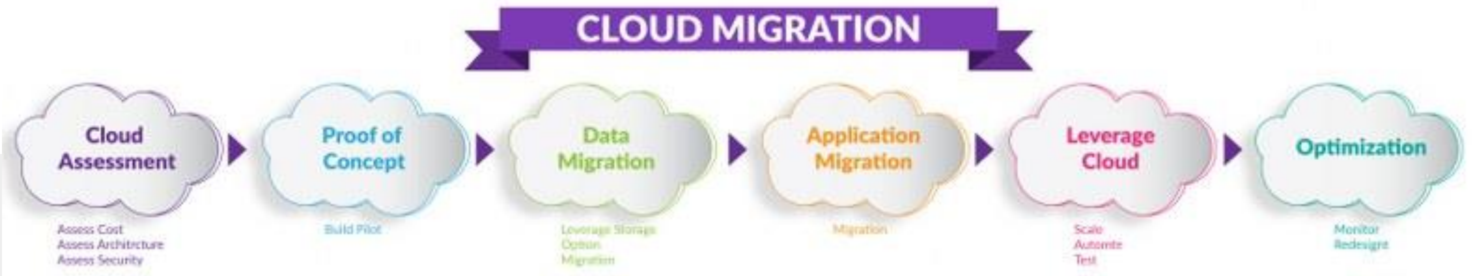
You need to copy or replicate data yourself as EBS snapshots are specific to the source AZ. Getting the data out requires a slave sync or taking LVM snapshots and transferring the data.

💲 Denotes additional cloud-to-cloud data transfer costs

Consider distributed NoSQL databases with the same distribution considerations. Spread primary and replica nodes across multiple AZs. Place as many as you need for required resiliency.

So What’s Best?

- Design for failure
  - Assume everything will fail, and architect a solution capable of handing each and every failure condition
- No one size fits all solution
  - Every application has its own architecture
    - Not all infrastructure building blocks fit well with all applications
  - Tradeoffs between levels of resiliency and cost
- The options available in the cloud today are unprecedented
  - Capabilities for global redundancy
  - Time to access
  - Investment required
- Follow High Availability Checklist (or create your own)
- Multi-AZ configurations with a solid DR plan are generally the most viable and cost-conscious solutions


CLOUD MIGRATION
Cloud Assessment
Assess Cost
Assess Architrcture
Assess Security
Proof of Concept
Build Pilot
Data Migration
Leverage Storage
Option
Migration
Application Migration
Migration
Leverage Cloud
Scale
Automte
Test
Optimization
Monitor
Redesignt