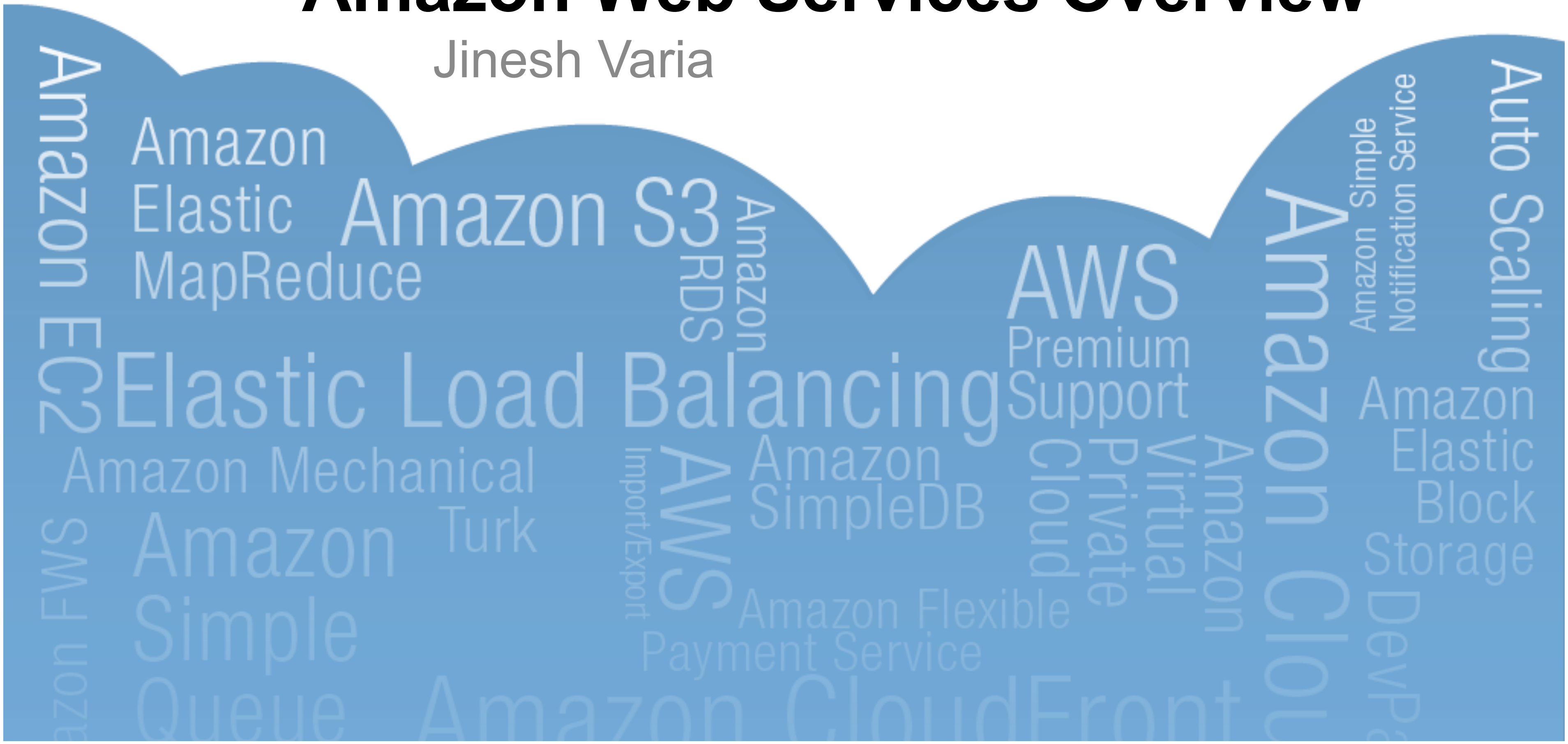
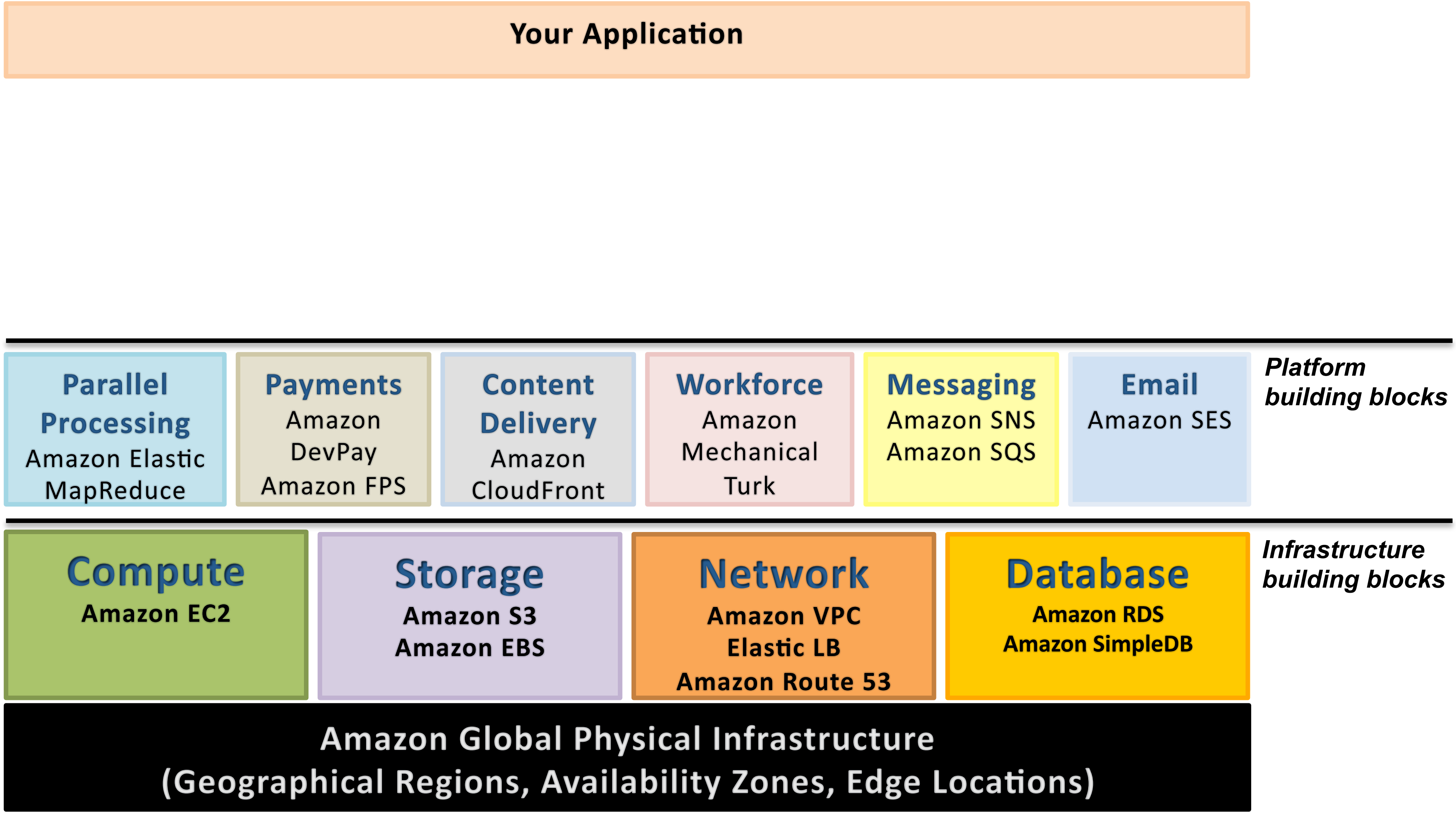
Amazon Web Services Overview

Jinesh Varia



The “Living and Evolving” AWS Cloud



Your Application

Platform building blocks

Parallel Processing
Amazon Elastic MapReduce

Payments
Amazon DevPay
Amazon FPS

Content Delivery
Amazon CloudFront

Workforce
Amazon Mechanical Turk

Messaging
Amazon SNS
Amazon SQS

Email
Amazon SES

Infrastructure building blocks

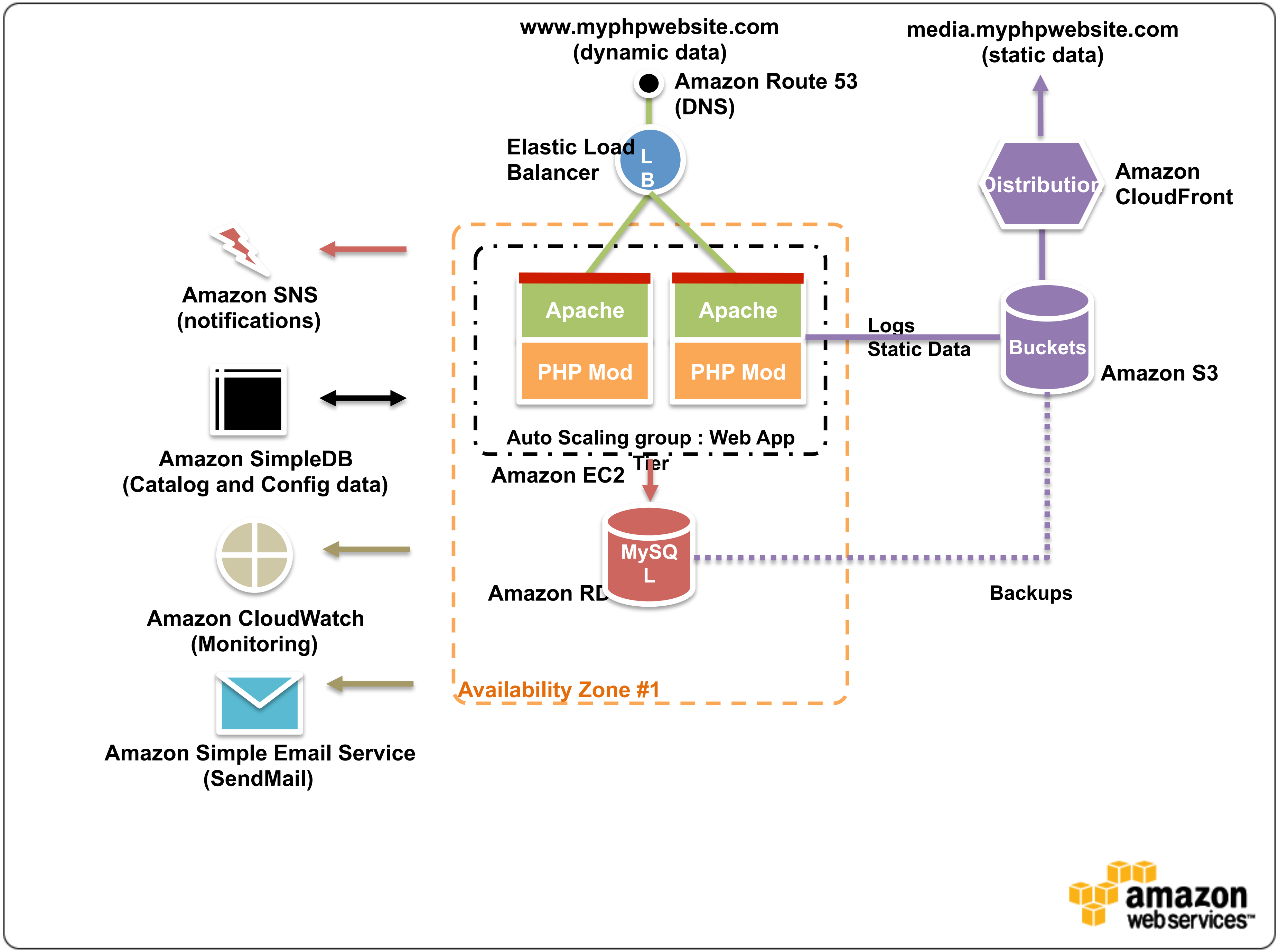
Compute
Amazon EC2

Storage
Amazon S3
Amazon EBS

Network
Amazon VPC
Elastic LB
Amazon Route 53

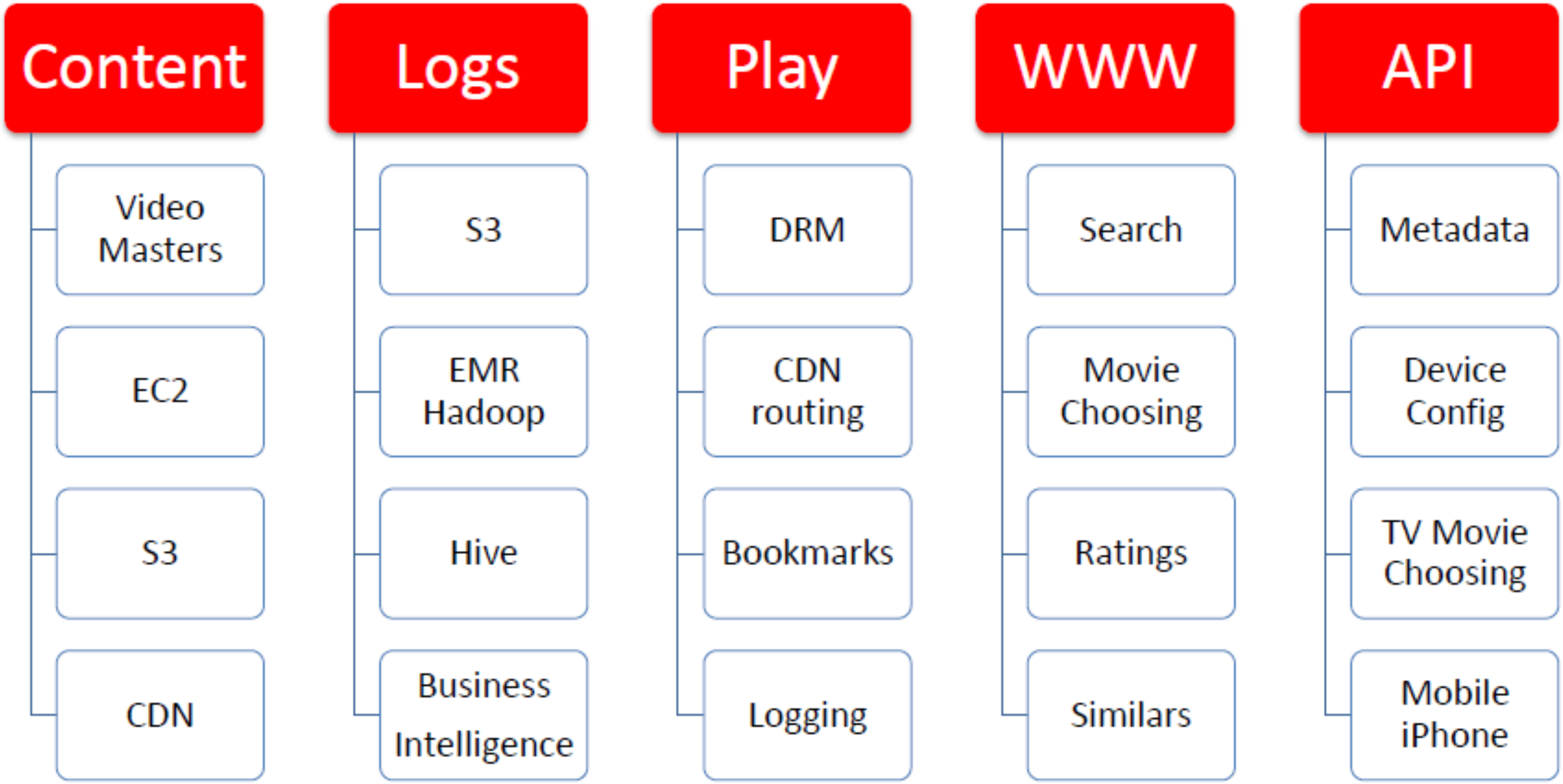
Database
Amazon RDS
Amazon SimpleDB

Amazon Global Physical Infrastructure
(Geographical Regions, Availability Zones, Edge Locations)


www.myphpwebsite.com
(dynamic data)
Amazon Route 53
(DNS)
media.myphpwebsite.com
(static data)
Elastic Load
Balancer
L
B
Distribution
Amazon
CloudFront
Amazon SNS
(notifications)
Apache
Apache
PHP Mod
PHP Mod
Logs
Static Data
Buckets
Amazon S3
Amazon SimpleDB
(Catalog and Config data)
Auto Scaling group : Web App
Tier
Amazon EC2
MySQ
L
Amazon RD
Backups
Amazon CloudWatch
(Monitoring)
Availability Zone #1
Amazon Simple Email Service
(SendMail)
amazon
webservices™

Netflix Deployed on AWS

amazon webservices™

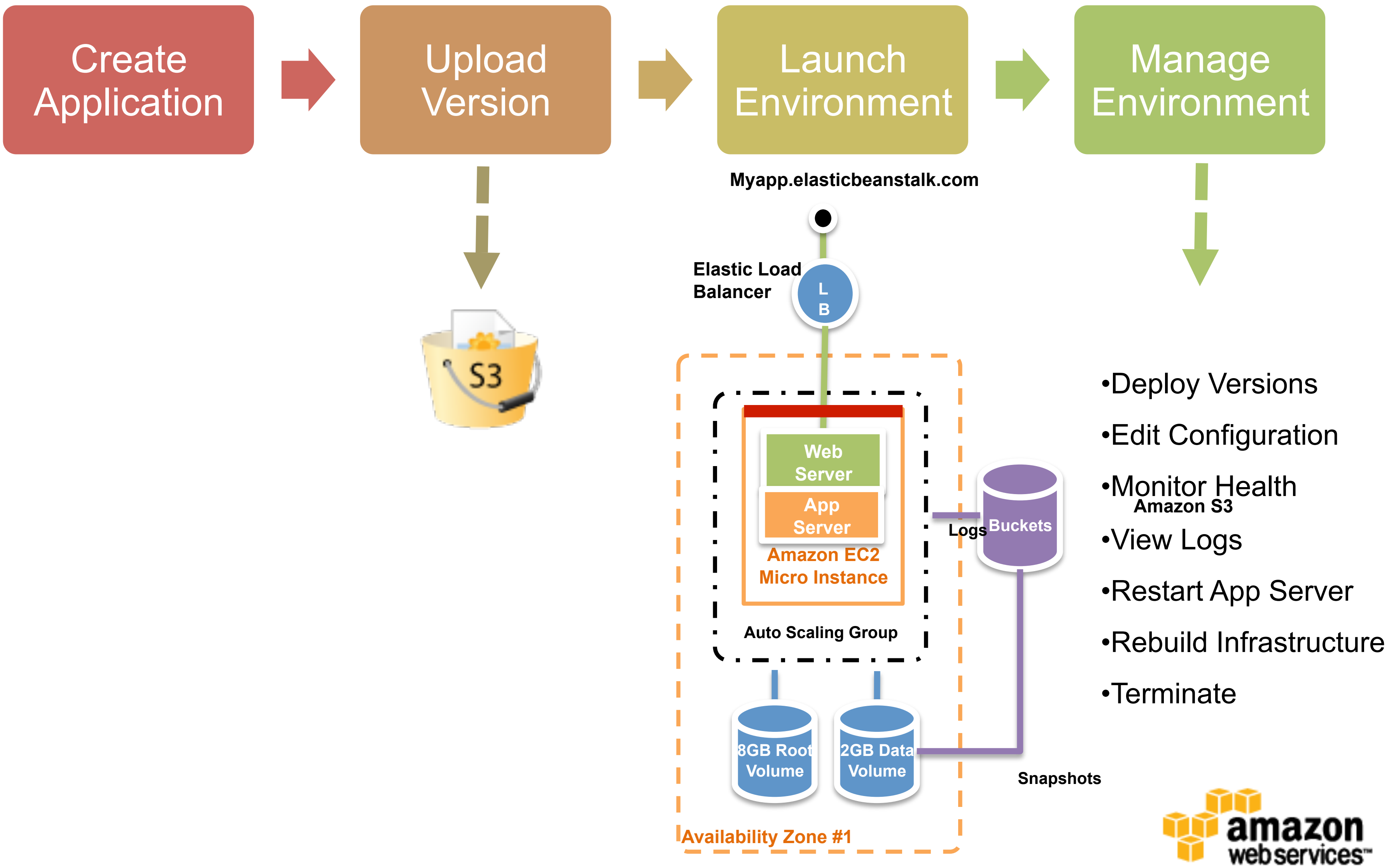


Content	Logs	Play	WWW	API
Video Masters	S3	DRM	Search	Metadata
EC2	EMR Hadoop	CDN routing	Movie Choosing	Device Config
S3	Hive	Bookmarks	Ratings	TV Movie Choosing
CDN	Business Intelligence	Logging	Similars	Mobile iPhone

NETFLIX

AWS Elastic Beanstalk

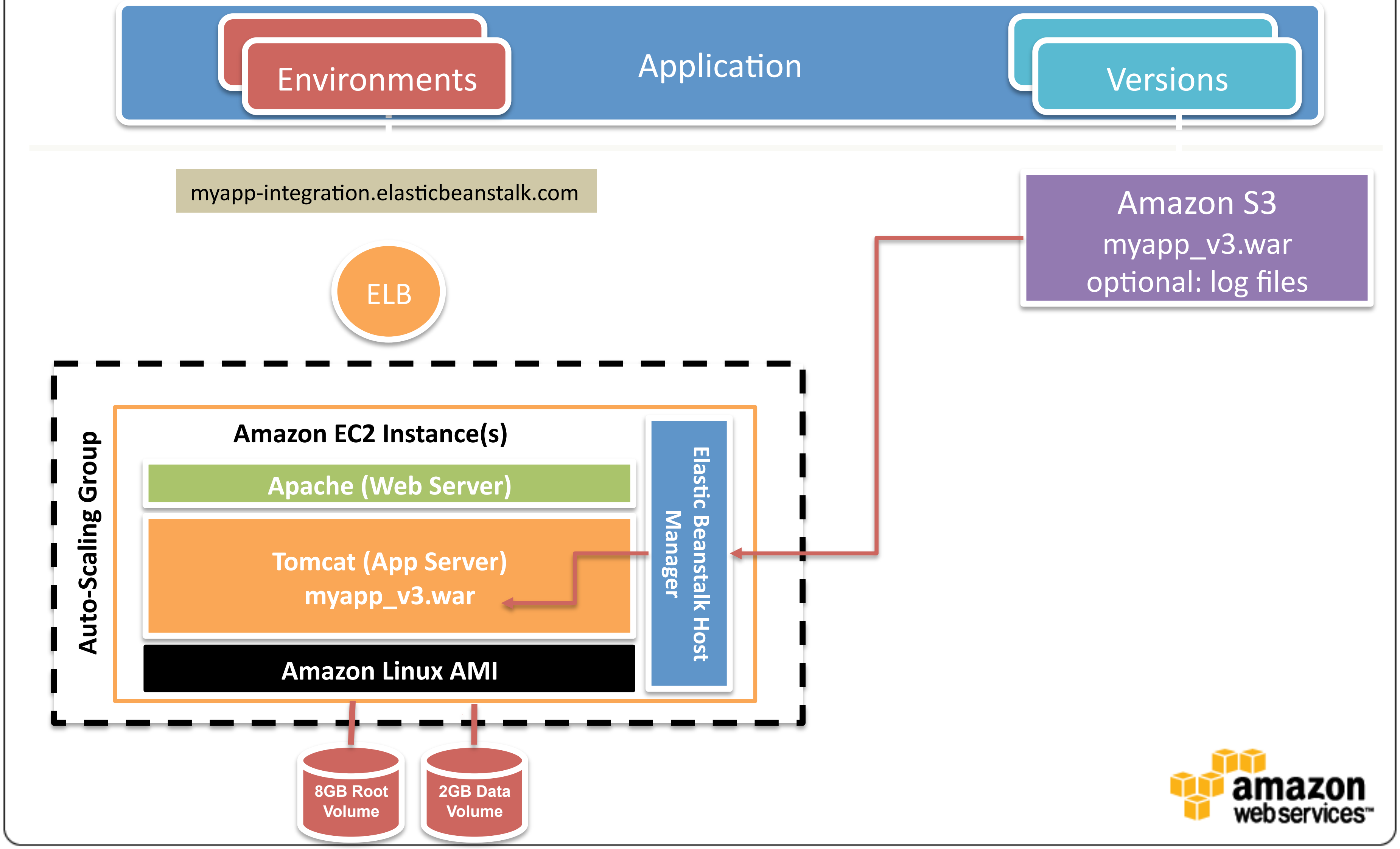
“Put your application on Auto Pilot”



Create Application → Upload Version → Launch Environment → Manage Environment

- Deploy Versions
- Edit Configuration
- Monitor Health
- View Logs
- Restart App Server
- Rebuild Infrastructure
- Terminate

Elastic Beanstalk "under the hood"



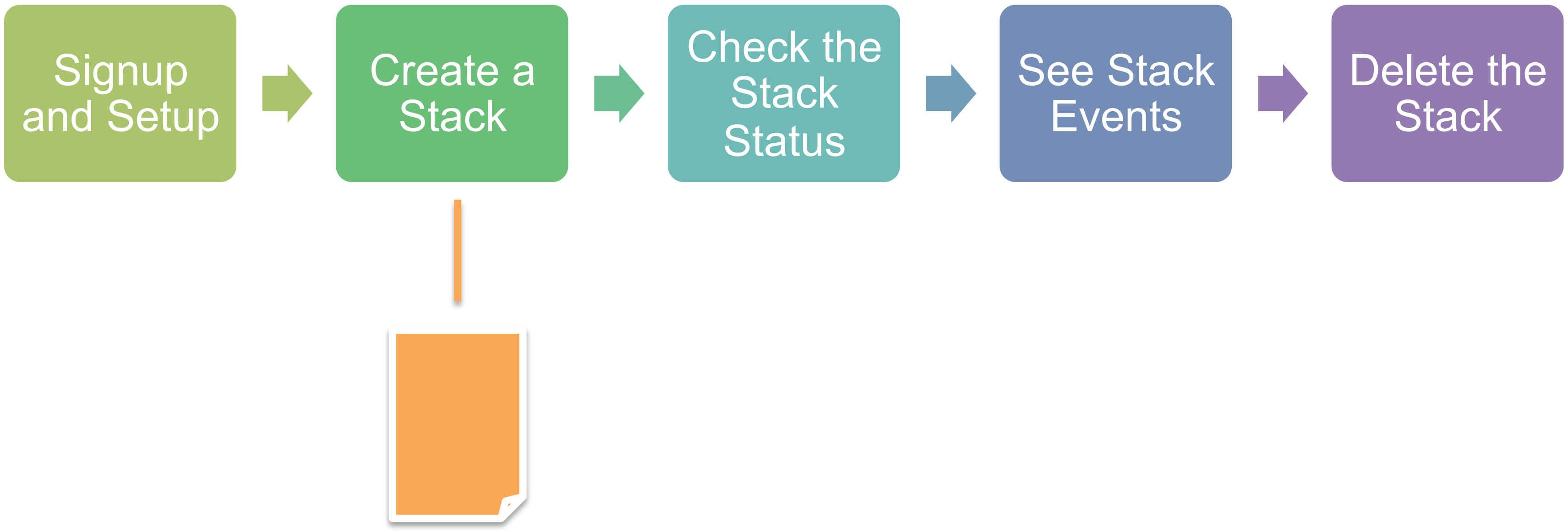
AWS CloudFormation

“Provision your infrastructure stack using one script”

AWS CloudFormation: Provisioning cloud resources made easy

- Fully **declarative** system
- **Document** based infrastructure specification
- **Logical** naming convention
- **Atomically** creates / destroys groups of AWS objects together
- Deploy **multi-tier** and **multi-AZ** stacks
- Handles the **bookkeeping** and **muck of provisioning** multiple related resources
- Focuses on **AWS resources**, while sys admins and developers focus on OS and application provisioning

AWS CloudFormation: Provisioning cloud resources made easy



Signup and Setup

Create a Stack

Check the Stack Status

See Stack Events

Delete the Stack

JSON Template

Load balanced webapp, fault tolerant in 3 AZ, RDS in 2 AZs

```
{
  "AWSTemplateFormatVersion" : "2010-06-08",
  "Parameters" : { "instanceType"    : { "Type" : "String", "Default" : "c1.medium" },
                   "capacity"        : { "Type" : "String", "Default" : "3" },
                   "zones"           : { "Type" : "CommaDelimitedList",
                                         "Default" : "us-east-1a,us-east-1b,us-east-1c"},
                   "dbPort"          : { "Type" : "String", "Default"  : "8443" },
                   "dbUser"          : { "Type" : "String", "NoEcho"  : "true" },
                   "dbPWD"           : { "Type" : "String", "NoEcho"  : "true" },
                   "accessKeyID"     : { "Type" : "String", "NoEcho"  : "true" },
                   "secretAccessKey" : { "Type" : "String", "NoEcho"  : "true" }
  },
  "Resources" : {
     "fw" : { "Type" : "AWS::SecurityGroup", "Properties" : {
              "GroupDescription" : "instance access for load balancer" }},
     "webRule" : { "Type" : "AWS::SecurityGroupIngress", "Properties" : {
                   "GroupName" : {"Ref":"fw"}, "IpProtocol" : "tcp",
                   "FromPort" : "4242", "ToPort" : "4242", "CidrIp" : "0.0.0.0/0" }},
     "sshRule" : { "Type" : "AWS::SecurityGroupIngress", "Properties" : {
                   "GroupName" : {"Ref":"fw"}, "IpProtocol" : "tcp",
                   "FromPort" : "22", "ToPort" : "22", "CidrIp" : "0.0.0.0/0" }},
     "dbfw" : { "Type" : "AWS::DBSecurityGroup", "Properties" : {
                "GroupDescription" : "backend database instance access" }},
     "dbRule" : { "Type" : "AWS::DBSecurityGroupIngress", "Properties" : {
                  "DBSecurityGroupName" : {"Ref":"dbfw"},
                  "EC2SecurityGroupName" : {"Ref":"fw"} }},
```

Legend:

Resources

Wiring

Fault tolerance

Application

Keys, etc.

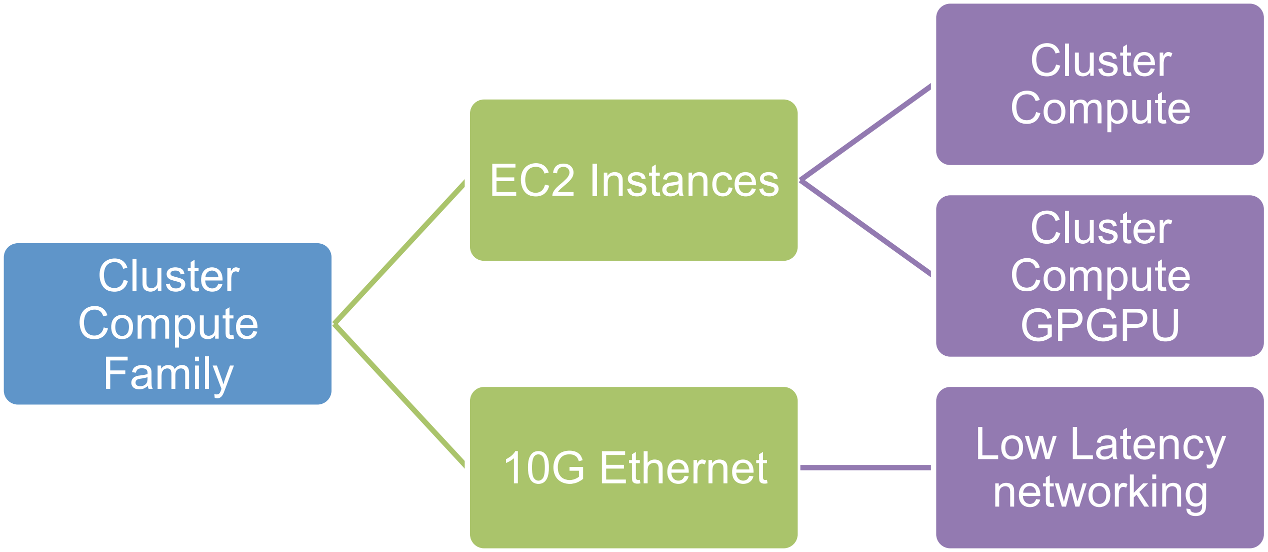
Cont on next slide ...

Innovative Business Models

On-demand Instances	Reserved Instances	Spot Instances	Dedicated Instances
• Pay as you go • Starts from 0.03/Hour	• Onetime upfront + Pay as you go • $56 for 1 year term and then $0.01/Hour	• Requested Bid Price and Pay as you go • $0.005 /Hour as of today at 9 AM	• Standard and Reserved • Multi-Tenant Single Customer • $10/Region + 0.105/Hour
For Spiky Workloads	**For Steady State Workloads**	**For Time-insensitive workloads**	**For Regulatory and Compliant Workloads**

Cloud HPC: Cluster Compute Instance

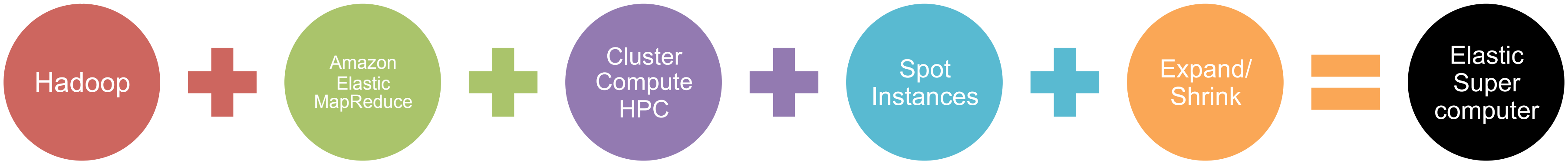
Cloud HPC: Cluster Compute Instance



2 * Xeon 5570
 (Intel “Nehalem”)
23 GB RAM
10 gbps Ethernet
1690 TB local disk
HVM-based virtualization
2 Tesla NVDIA GPGPUs

Cc1.4xl = $1.60 / hr
Cg1.4xl = $2.10 /hr

This is how customers leverage the “Big Data Cloud”



Distributed Processing Framework

On-demand Infrastructure (Cloud) + Automation

Each VM =
2 Xeon “Nehalem” Quad-core
10G Ethernet
2 GPGPUs

Cost savings due to lower “Spot price”
(Time-insensitive tasks)

Expand or Shrink a running cluster

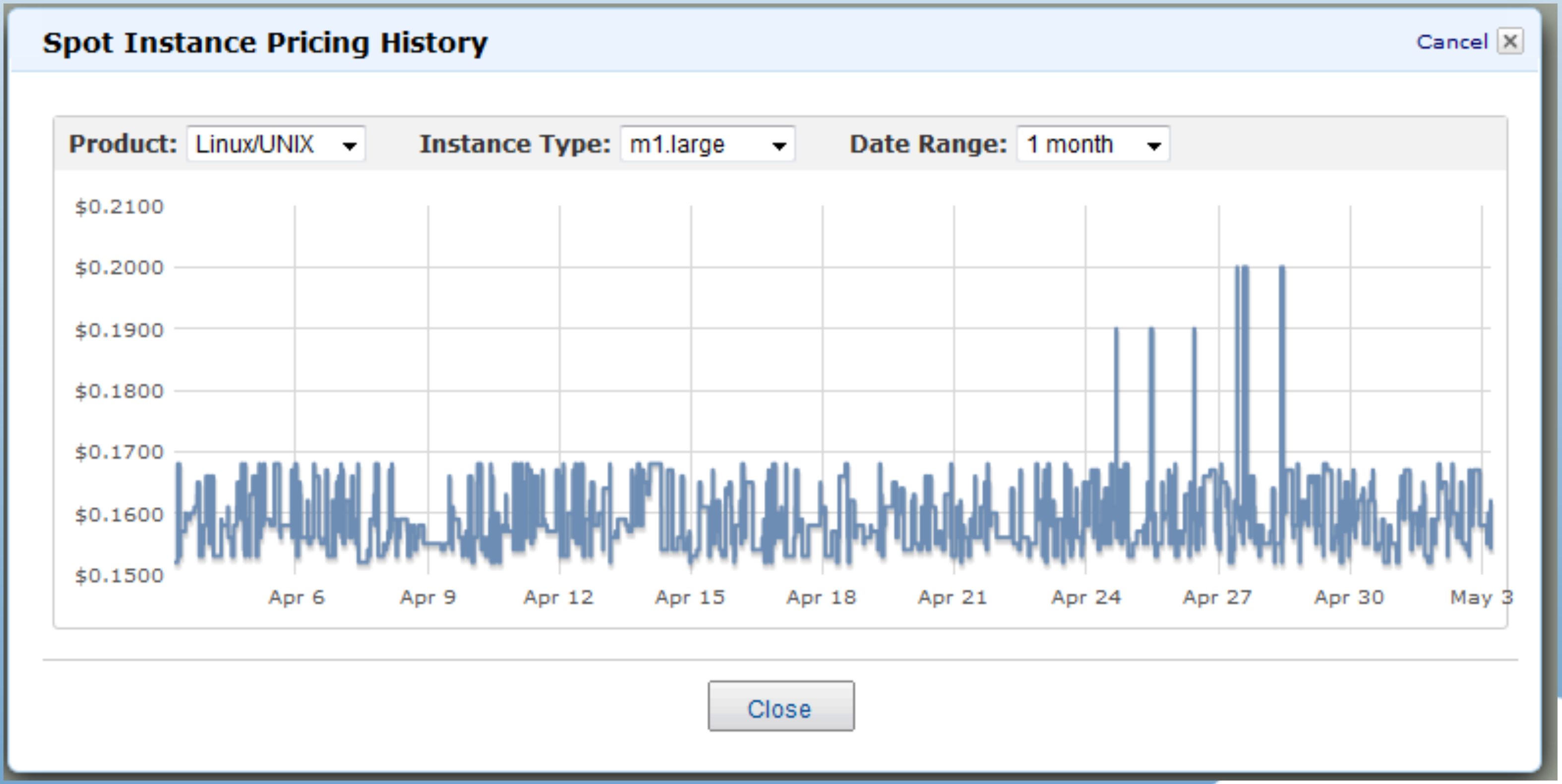
Big Data power house

amazon webservices™

Choosing the right pricing model

On-demand Instances vs. Spot Instances

Save more money by using Spot Instances



Spot Instance Pricing History

Product: Linux/UNIX

Instance Type: m1.large

Date Range: 1 month

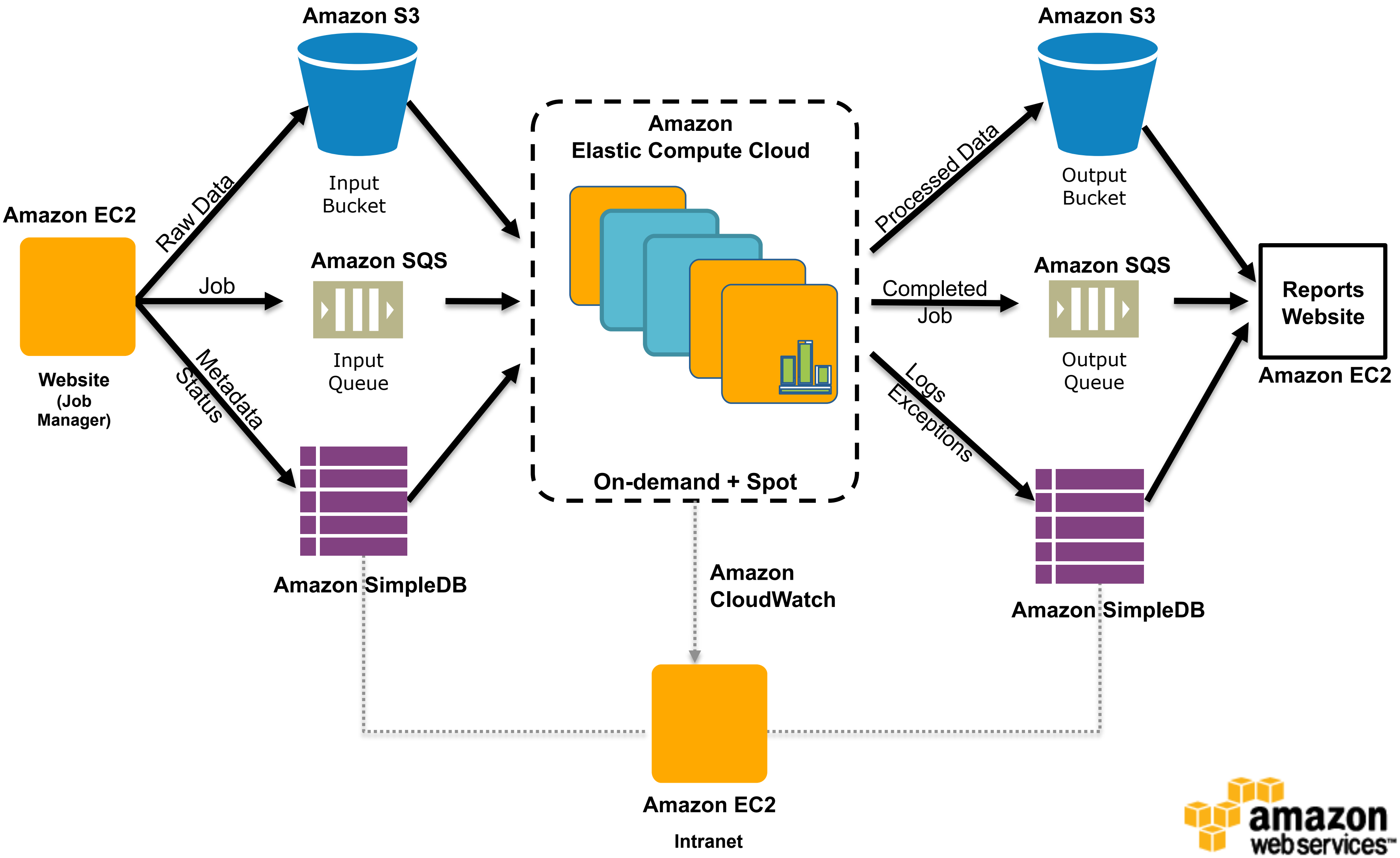
Spot Use cases

Use Case	Types of Applications
Batch Processing	Generic background processing (scale out computing)
Hadoop	Hadoop/MapReduce processing type jobs (e.g. Search, Big Data, etc.)
Scientific Computing	Scientific trials/simulations/analysis in chemistry, physics, and biology
Video and Image Processing/Rendering	Transform videos into specific formats
Testing	Provide testing of software, web sites, etc
Web/Data Crawling	Analyzing data and processing it
Financial	Hedgefund analytics, energy trading, etc
HPC	Utilize HPC servers to do embarrassingly parallel jobs
Cheap Compute	Backend servers for Facebook games

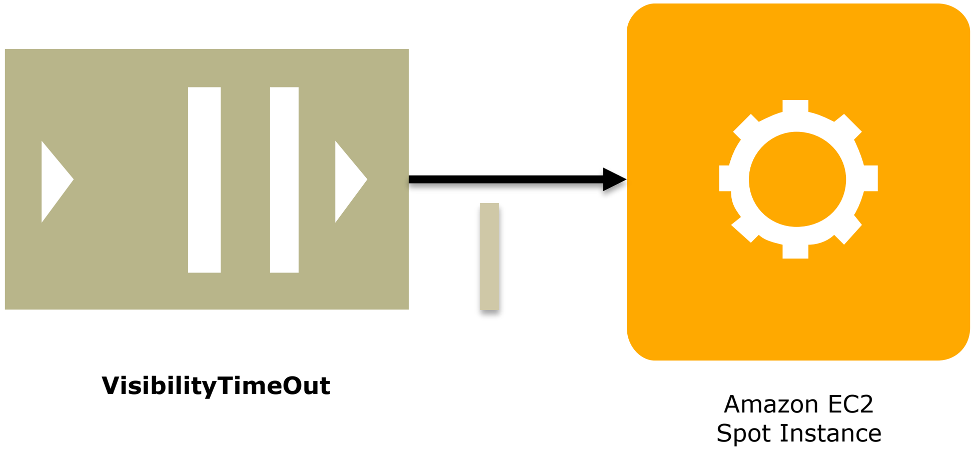
Basic tactics for Spot Instances

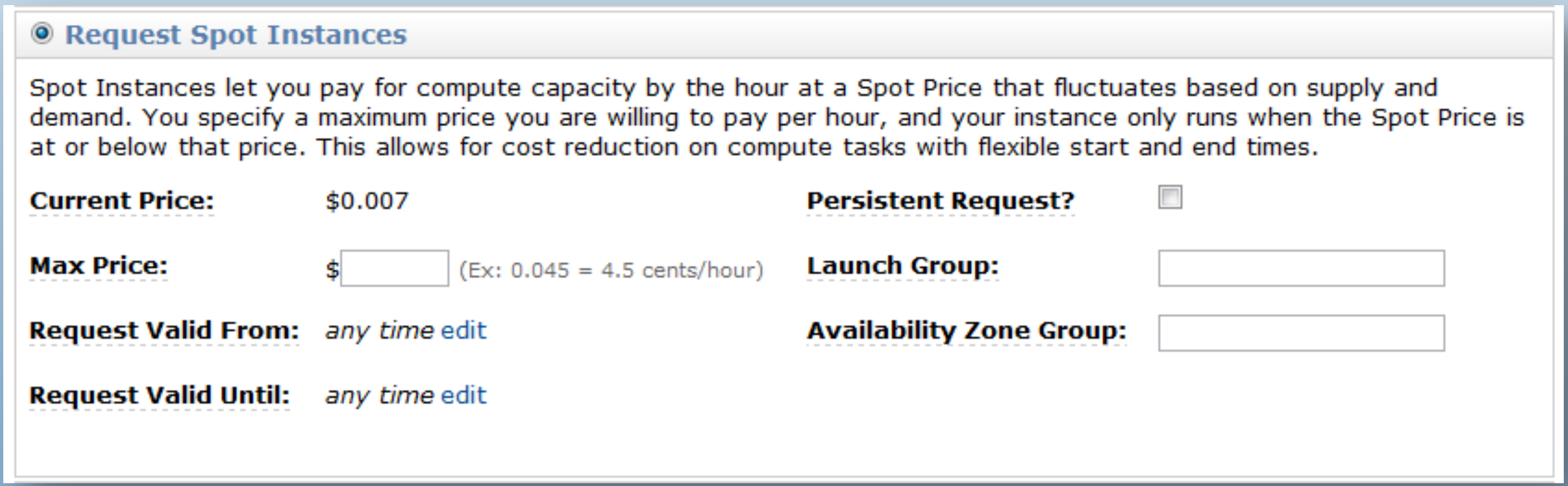
- Try to launch Spot instances first and then on-demand instances if you don’t get the spot instances in under 15 minutes
- Use Spot and On-demand in Hybrid Fashion. Master Node in Cluster is on-demand instance, worker nodes are spot instances

Video Transcoding Application Example



Use of Amazon SQS in Spot Architectures





◉ Request Spot Instances

Spot Instances let you pay for compute capacity by the hour at a Spot Price that fluctuates based on supply and demand. You specify a maximum price you are willing to pay per hour, and your instance only runs when the Spot Price is at or below that price. This allows for cost reduction on compute tasks with flexible start and end times.

Current Price:	$0.007	**Persistent Request?**	□
Max Price:	$ (Ex: 0.045 = 4.5 cents/hour)	**Launch Group:**	
Request Valid From:	*any time* edit	**Availability Zone Group:**	
Request Valid Until:	*any time* edit		

amazon
webservices™

Best Practices for using Spot Instances

- Save Your Work Frequently
- Add Checkpoints
- Split up Your Work
- Test Your Application
- Minimize Group Instance Launches
- Use Persistent Requests for Continuous Tasks
- Track when Spot Instances Start and Stop
- Access Large Pools of Compute Capacity

Case Study: Optimizing Video Transcoding Workloads (On-demand + Spot + Reserved)

Free Offering

- Optimize for reducing cost
- Acceptable Delay Limits

Implementation

- Set Persistent Requests
- Use on-demand Instances, if delay

Maximum Bid Price

< On-demand Rate

Get *your* set reduced price for your workload

Premium Offering

- Optimized for Faster response times
- No Delays

Implementation

- Invest in RIs
- Use on-demand for Elasticity

Maximum Bid Price

>= On-demand Rate

Get Instant Capacity for higher price

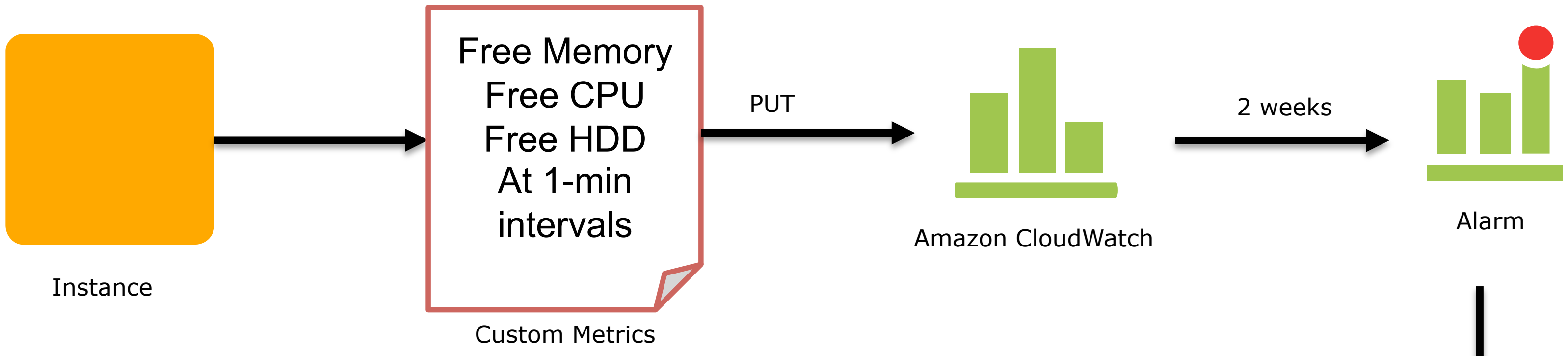
Optimize by choosing the Right Instance Type

M1.xlarge, c1.medium, multiple m1.small etc.

Basic recommendations on Instance Type

- Choose the EC2 instance type that best matches the resources required by the application
  - Start with memory requirements and architecture type (32bit or 64-bit)
  - Then choose the closest number of virtual cores required
- Scaling across AZs
  - Smaller sizes give more granularity for deploying to multiple AZs

Tip – Instance Optimizer



Instance

Free Memory
Free CPU
Free HDD
At 1-min
intervals

Custom Metrics

PUT

Amazon CloudWatch

2 weeks

Alarm

“You could save a bunch of money by switching to a **small instance**, Click on CloudFormation Script to Save”

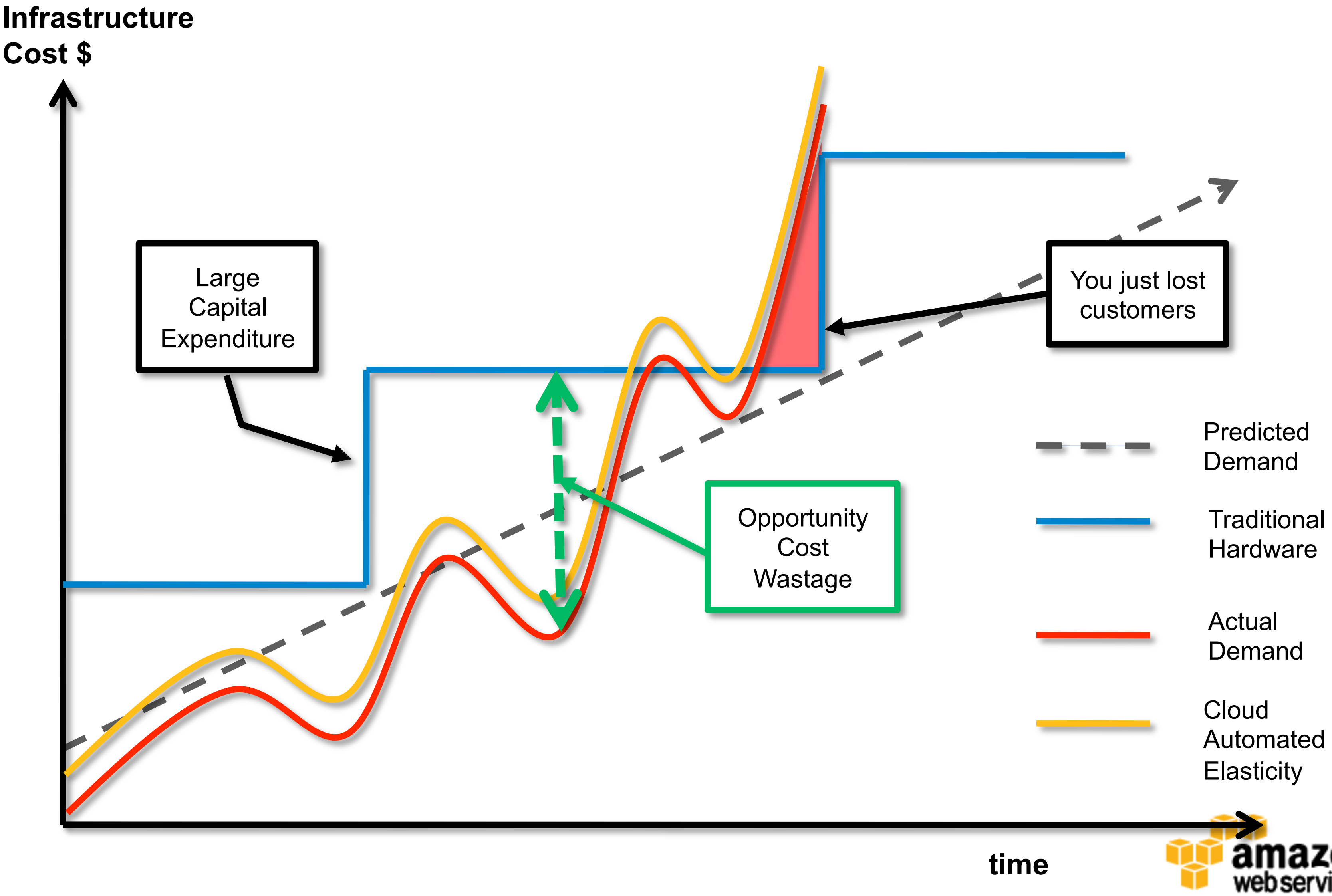
Thank you!

jvaria@amazon.com
Twitter: @jinman

amazon
web services™

http://aws.amazon.com

Optimize by Implementing Elasticity



Large Capital Expenditure

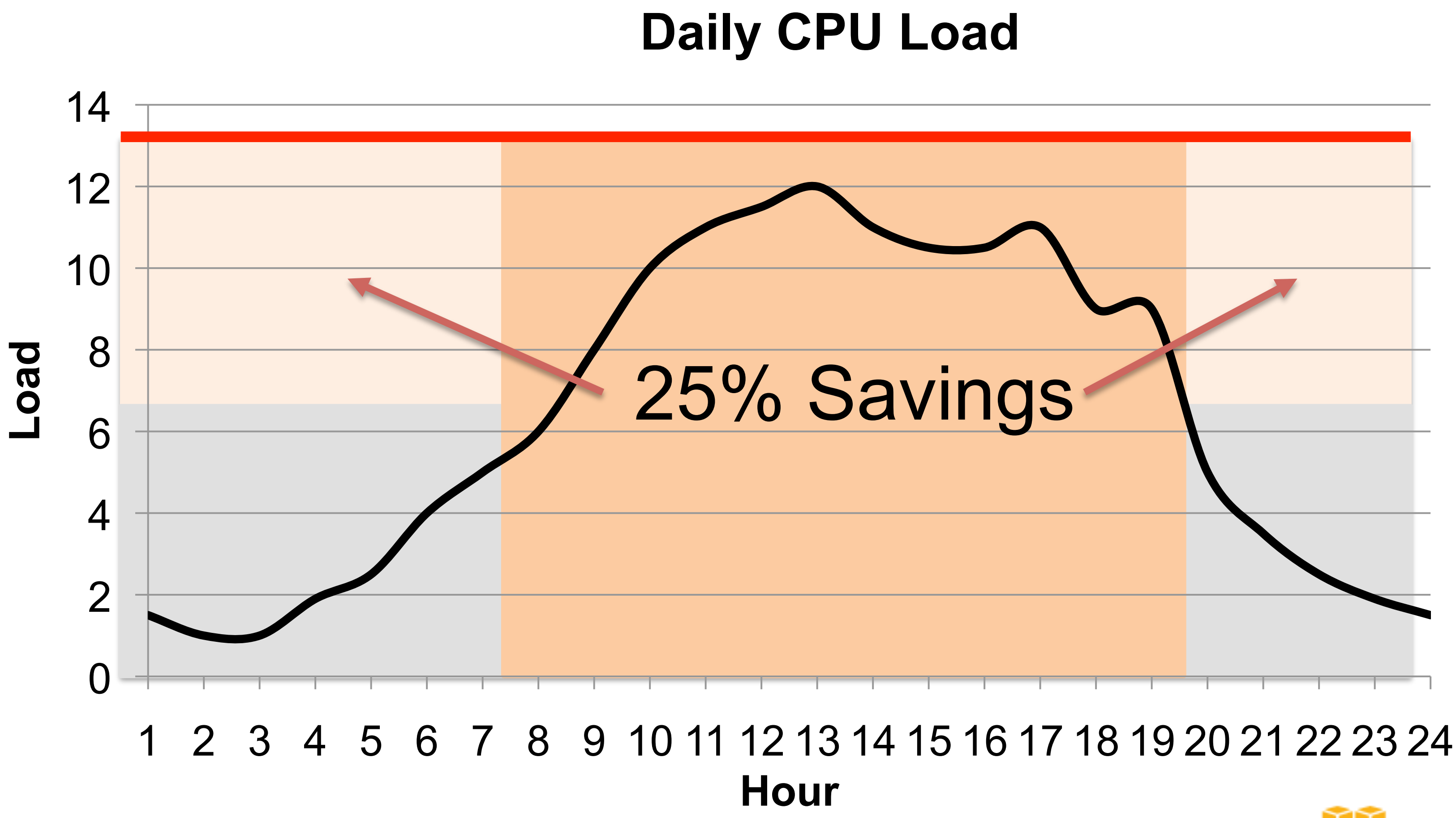
Opportunity Cost Wastage

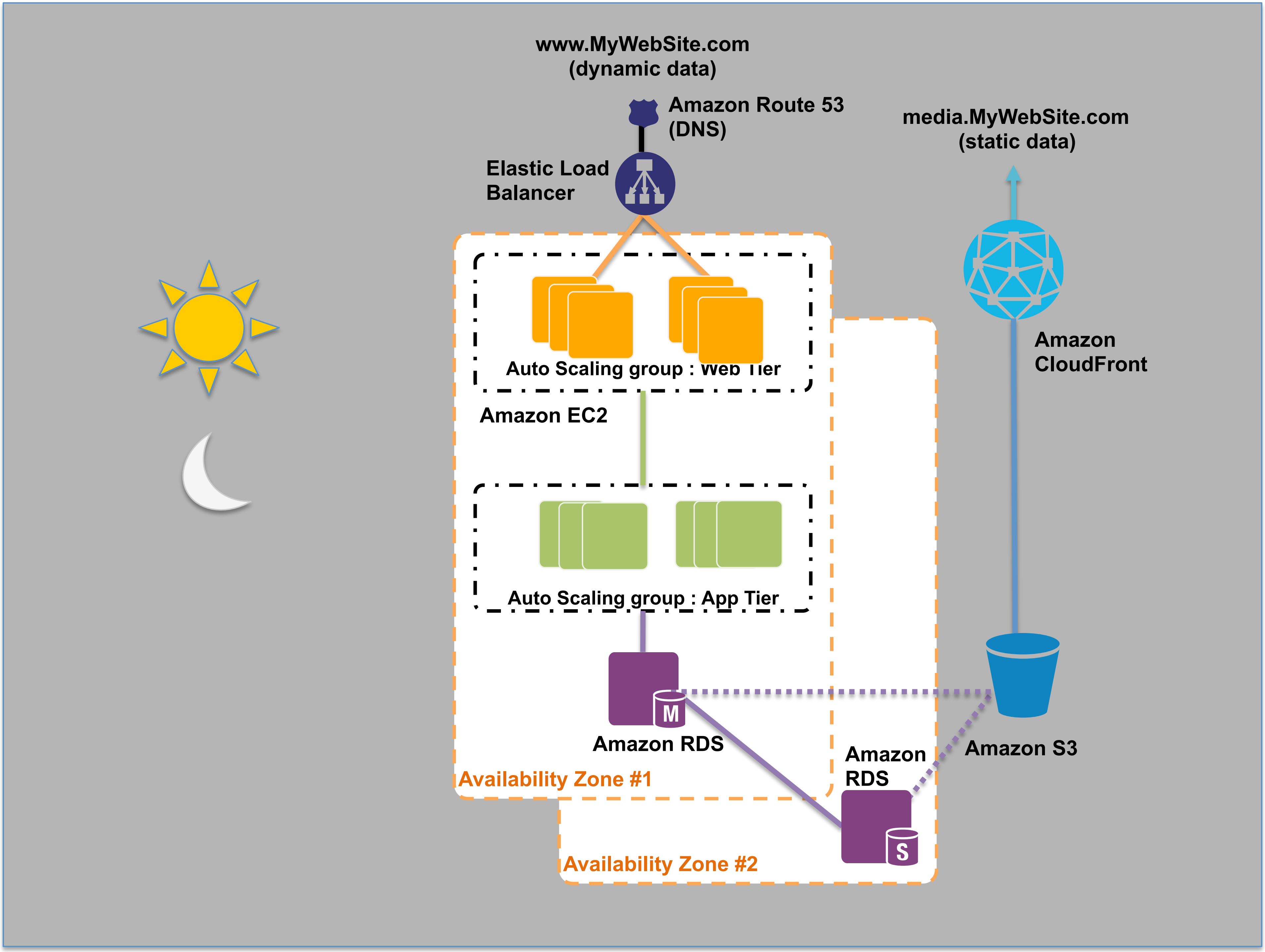
You just lost customers

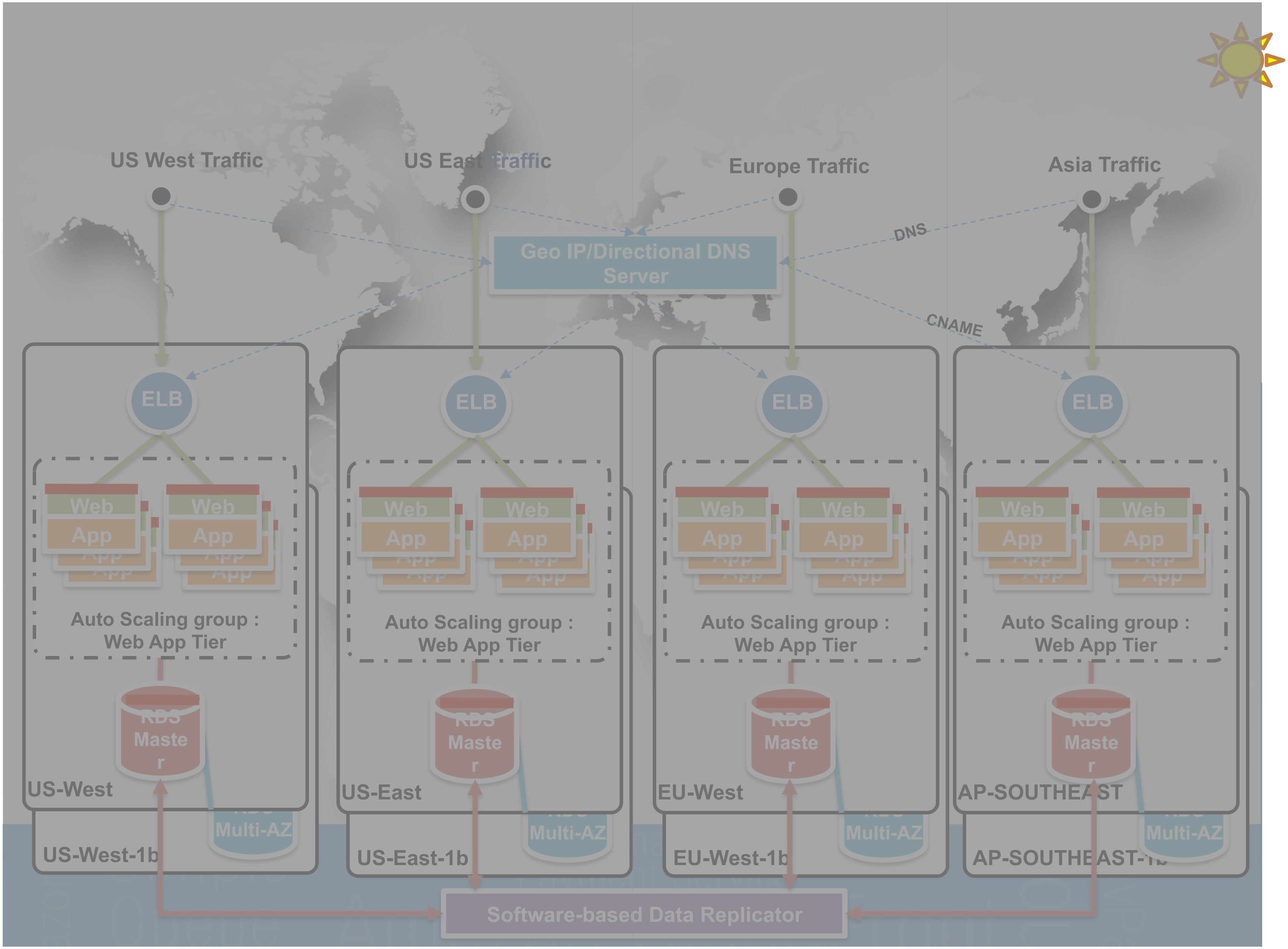
Optimize based on demand

Understand your usage patterns

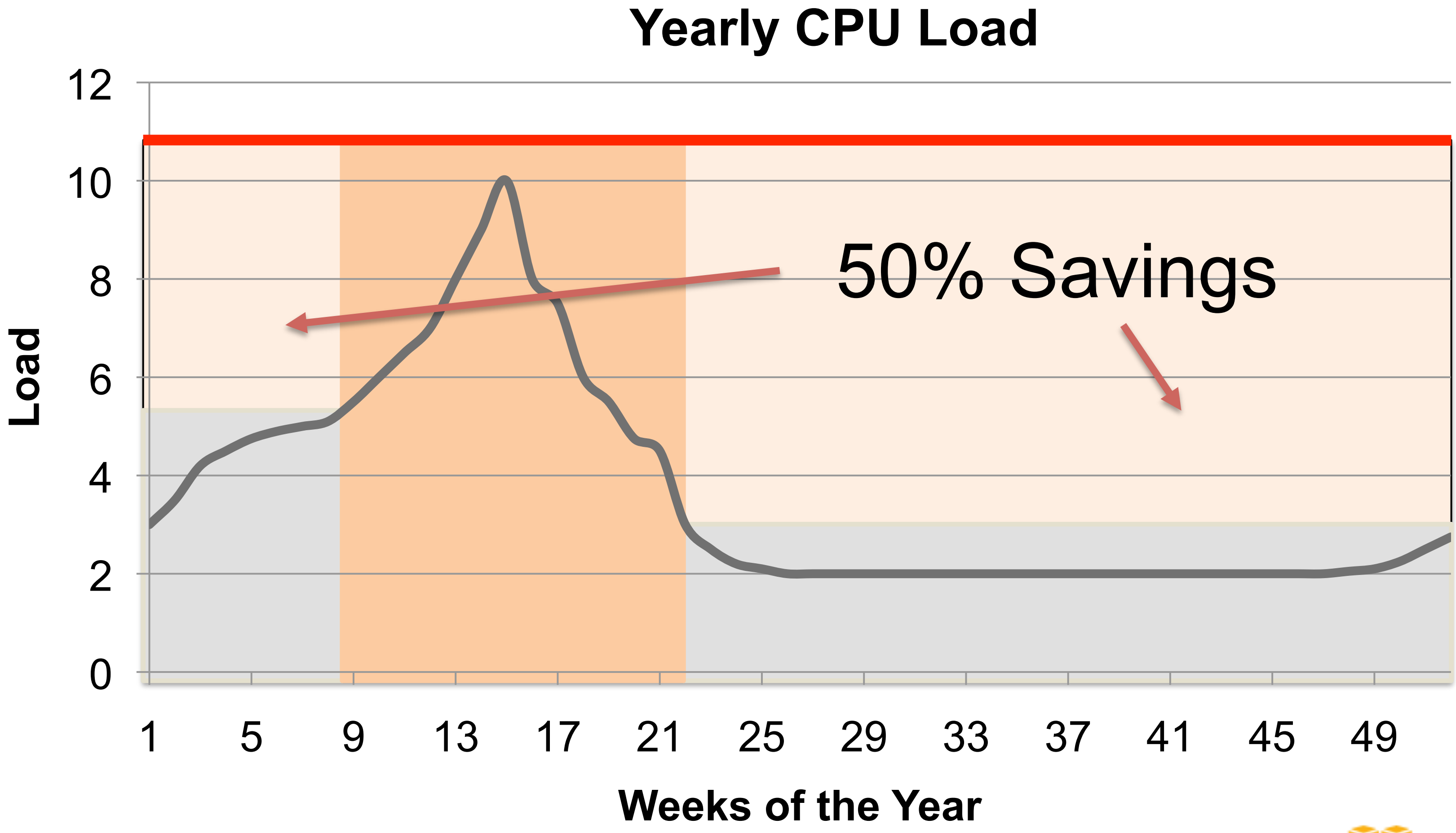
Optimize by time of the day




www.MyWebSite.com
(dynamic data)
Amazon Route 53
(DNS)
Elastic Load
Balancer
media.MyWebSite.com
(static data)
Amazon
CloudFront
Auto Scaling group : Web Tier
Amazon EC2
Auto Scaling group : App Tier
M
Amazon RDS
Amazon
RDS
S
Amazon S3
Availability Zone #1
Availability Zone #2


US West Traffic
US East Traffic
Europe Traffic
Asia Traffic
DNS
Geo IP/Directional DNS Server
CNAME
ELB
Web
App
App
App
Auto Scaling group : Web App Tier
RDS Maste r
US-West
RDS Multi-AZ
US-West-1b
ELB
Web
App
App
App
Auto Scaling group : Web App Tier
RDS Maste r
US-East
RDS Multi-AZ
US-East-1b
ELB
Web
App
App
App
Auto Scaling group : Web App Tier
RDS Maste r
EU-West
RDS Multi-AZ
EU-West-1b
ELB
Web
App
App
App
Auto Scaling group : Web App Tier
RDS Maste r
AP-SOUTHEAST
RDS Multi-AZ
AP-SOUTHEAST-1b
Software-based Data Replicator

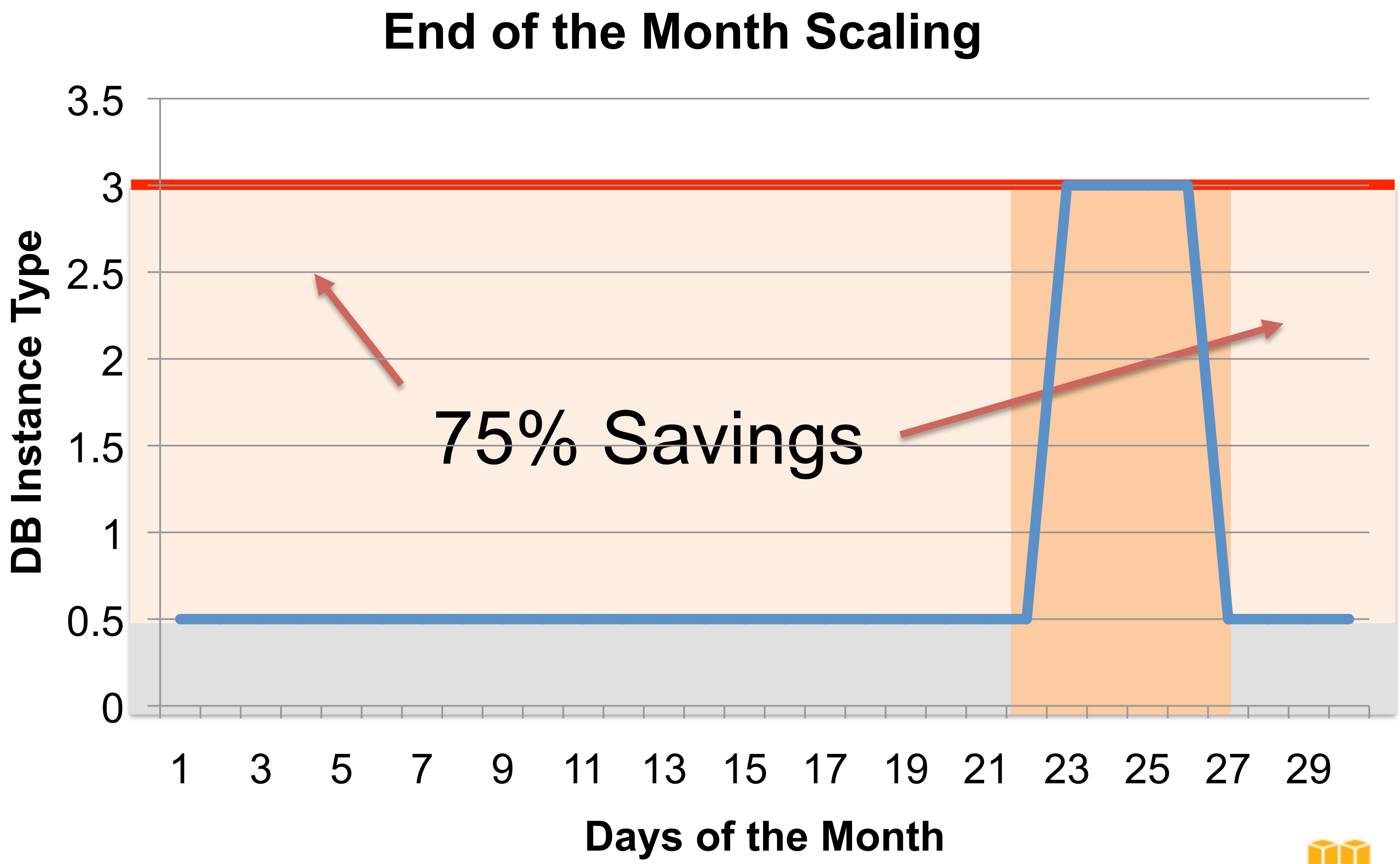
Optimize by seasonal cycles



Auto scaling : Types of Scaling

- Scaling by Schedule
  - Use Scheduled Actions in Auto Scaling Service
    - Date
    - Time
    - Min and Max of Auto Scaling Group Size
  - You can create up to 125 actions, scheduled up to 31 days into the future, for each of your auto scaling groups. This gives you the ability to scale up to four times a day for a month.
- Scaling by Policy
  - Scaling up Policy - Double the group size
  - Scaling down Policy - Decrement by 1

Optimize during the month



End of the month processing

- Expand the cluster at the end of the month
  - Expand/Shrink feature in Amazon Elastic MapReduce
- Vertically Scale up at the end of the month
  - Modify-DB-Instance (in Amazon RDS) (or a New RDS DB Instance)
  - CloudFormation Script (in Amazon EC2)

Choosing the right pricing model

On-demand Instances vs. Reserved Instances

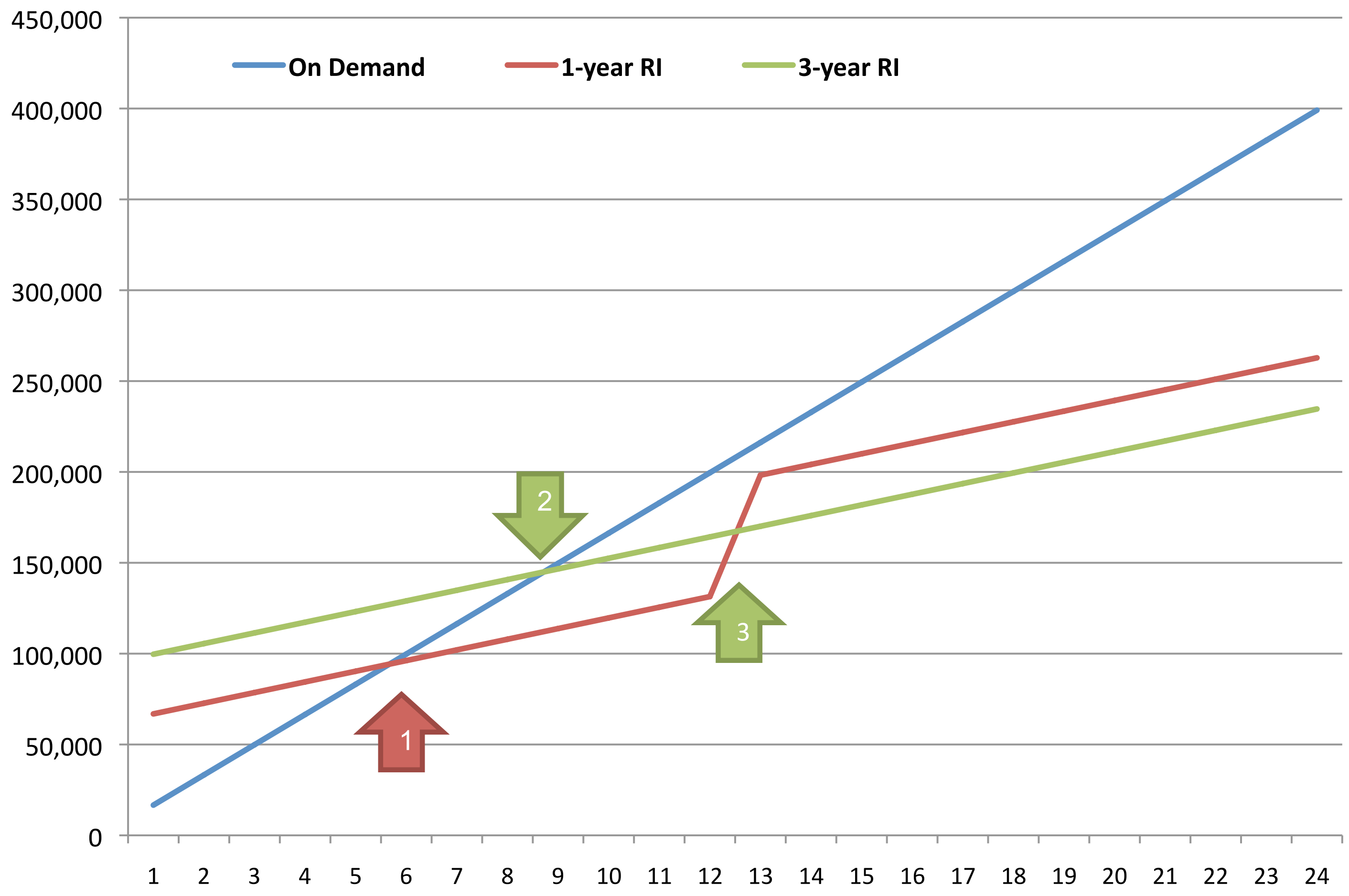
Steady State Usage

Total Cost for 1 Year-term of 2 application servers

	Usage Fee	One-time Fee	Total	Savings
Option 1 On-Demand only	$1493	-	$1493	**-**
Option 2 On-Demand + Reserved	$1008	$227	$1234	**~20%**
Option 3 All reserved	$528	$455	$983	**~35%**

Total Cost for 3 Year-term of the same 2 application servers

	Usage Fee	One-time Fee	Total	Savings
Option 1 On-Demand only	$4479	-	$4479	**-**
Option 2 On-Demand + Reserved	$3024	$350	$3374	**~30%**
Option 3 All reserved	$1584	$700	$2284	**~50%**



1 1-year RI versus On Demand:
cost savings realized after first 6 months of usage

2 3-year RI versus On Demand:
cost savings realized after first 9 months of usage.

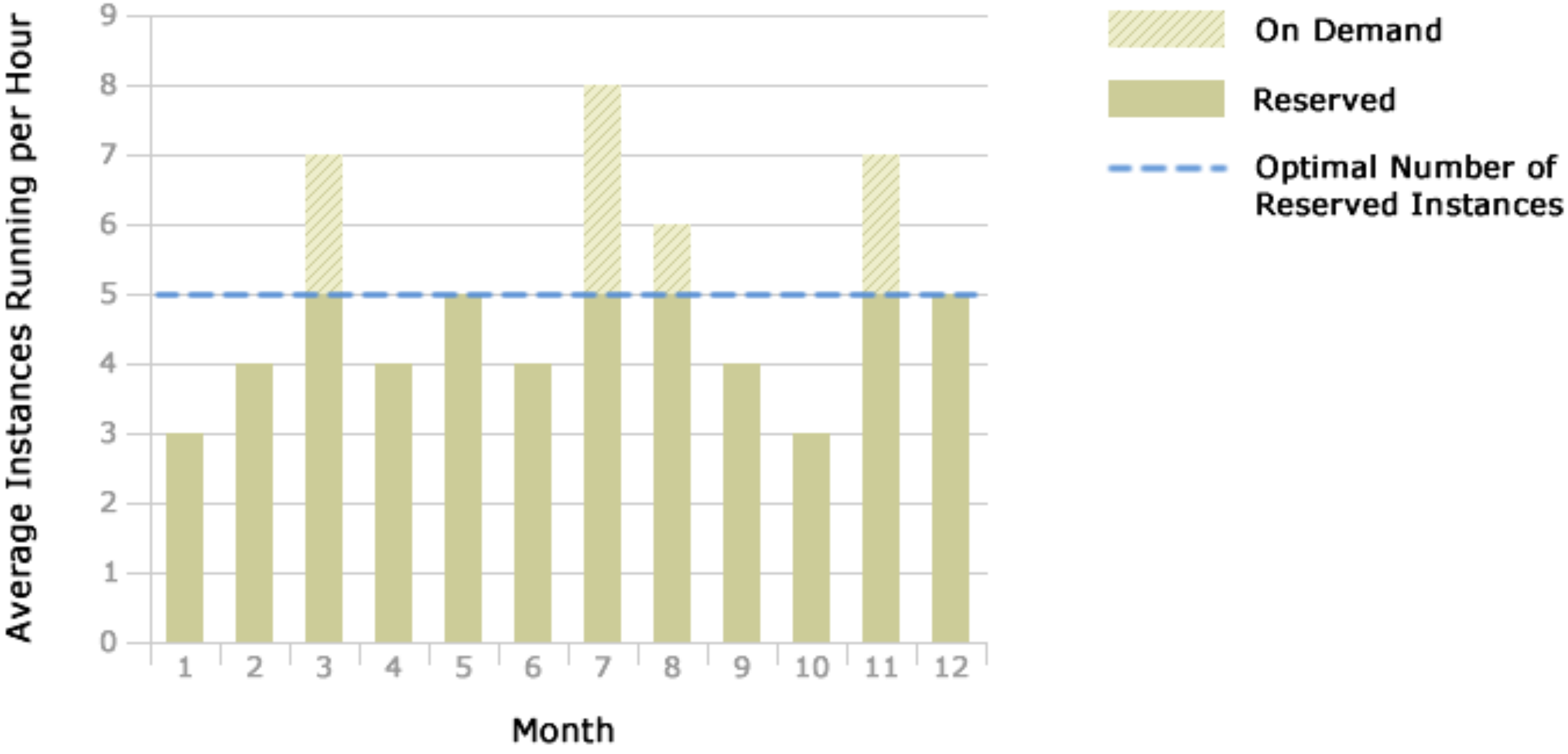
3 3-year RI versus 1-year RI:
Net savings of 3-year RI versus 1-year RI begin by month 13 and continue throughout the RI term (additional 23 months of savings)

amazon webservices™

Recommendations

- Steady State Usage : Use All Reserved
    - If you plan on running for at least 6 months, invest in RI for 1-year term
    - If you plan on running for at least 8.7 months, invest in RI for 3-year term

Common Pattern: Reserved + On-Demand



Recommendations

- Steady State Usage : Use All Reserved
  - If you plan on running for at least 6 months, invest in RI for 1-year term
  - If you plan on running for at least 8.7 months, invest in RI for 3-year term
- Unpredictable Demand : Use combination of Reserved + On-demand
  - With Reserved Instances, the costs average to an effective hourly rate 40% lower than the On-Demand rate.