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

Architecting on AWS

Designing for Cost

Designing for Cost | What we'll cover

1. Cost model
2. Services and feature costs
3. Billing options
4. Best practices

Designing for Cost | Cost Model

1

Cost model

Cost Model

- Amazon wants customers to pay for exactly what they use
- Do not pay for unutilized feature or services
- This model translates into a very granular cost structure
- Every Application has different component bounding (CPU, Memory, Disk I/O), pay for what you use
- Customers have control of how they utilize our products and service, which leads to control over cost expenditures

Designing for Cost | Services and feature costs

2

Services and feature costs

ELB, EIP, and CloudWatch Costs

- EIP
  - Free when associated with an EC2 instance
  - .005 per hour unassociated
  - .10 per 100 remaps
- CloudWatch
  - detailed monitoring $3.50 per instance per month
  - custom metrics .50 per metric per month
- ELB
  - .025 per hour
  - .008 per GB of bandwidth

EBS Service and Feature Costs

- Standard EBS Volume
  - $.10 per GB/month
  - $.10 per million I/O operations
- Provisioned IOPS EBS Volumes
  - $.125 per GB/month
  - $.10 per Provisioned IOPS/month
  - (IOPS x % of time use per month x $.10) = upcharge
- EBS Optimized Instances Surcharge
  - $.025 per hour for 500 MB/s Storage NIC
  - $.05 per hour for 1 GB/s Storage NIC

S3, S3 RRS, Glacier Costs

- Pay for capacity used:
  - S3 Standard Storage $.095
  - S3 Reduced Redundancy Storage $.076
  - Glacier $.01
- S3 PUT, COPY, POST, LIST .005 per 1000 requests
- Glacier Archive/Restore .05 per 1000 requests
- All GET .004 per 10,000 requests
- DELETE Free
- Data Transfer .12 per GB

RDS Service and Feature Costs

- Multiple instance types to choose from
- Provisioned IOPS (up to 30,000 per DB) optional
- Data Transfer Out of a Region: $0.12/GB
- Reserved billing model available

DynamoDB Service Costs

- Provision IOPS capacity .0065 per 10 units write, 50 units read capacity
- Indexed data storage $.25 GB/month
- Data Transfer $.12 per GB
- Reserved billing model available

R53 and CloudFront Service Costs

- Route 53
  - $0.50 per hosted zone
  - $0.50 per million for standard queries
  - $0.75 per million for latency based queries
  - $0.50 per health check per month inside AWS, $0.75 outside (S3 Endpoints are free)
- CloudFront
  - Bandwidth out .12 per GB
  - $.0075 per 10,000 requests

SQS, SNS, SES Service and Feature Costs

- SQS
  - $0.50 per million api requests
- SNS
  - $0.06 per 100,000 HTTP notifications
  - $2.00 per 100,000 Emails notifications
  - $0.75 per 100 SMS notifications
  - no charge for SQS notifications
- SES $0.10 per 1000 Emails out
- Data transfer outside a region $0.12 GB
- Data transfer inside a region is free

Designing for Cost | Billing Options

3

Billing options

Free Services and Features

- Free Tier Utilization
- VPC
- Auto Scaling
- Cloud Watch standard metrics
- CloudFormation
- IAM
- OpsWorks
- Elastic Beanstalk

EC2 Billing Options

- Prices vary by instance type (19+ Instance Types)
- On Demand prices should be considered retail rate
- Reserved Instance billing model available
- Unique Spot Market available

EC2 Billing Options

On-demand instances

Unix/Linux instances start at $0.02/hour

Pay as you go for compute power

Low cost and flexibility

Pay only for what you use, no up-front commitments or long-term contracts

Use Cases:

Applications with short term, spiky, or unpredictable workloads;

Application development or testing

Reserved instances

1 or 3 year terms

Pay low up-front fee, receive significant hourly discount

Low Cost / Predictability

Helps ensure compute capacity is available when needed

Use Cases:

Applications with steady state or predictable usage

Applications that require reserved capacity, including disaster recovery

Spot instances

Bid on unused EC2 capacity

Spot Price based on supply/demand, determined automatically

Cost / Large Scale, dynamic workload handling

Use Cases:

Applications with flexible start and end times

Applications only feasible at very low compute prices

EC2 Billing Options

On-demand instances

Unix/Linux instances start at $0.02/hour

Pay as you go for compute power

Low cost and flexibility

Pay only for what you use, no up-front commitments or long-term contracts

Use Cases:

Applications with short term, spiky, or unpredictable workloads;

Application development or testing

Reserved instances

1 or 3 year terms

Pay low up-front fee, receive significant hourly discount

Low Cost / Predictability

Helps ensure compute capacity is available when needed

Use Cases:

Applications with steady state or predictable usage

Applications that require reserved capacity, including disaster recovery

Spot instances

Bid on unused EC2 capacity

Spot Price based on supply/demand, determined automatically

Cost / Large Scale, dynamic workload handling

Use Cases:

Applications with flexible start and end times

Applications only feasible at very low compute prices

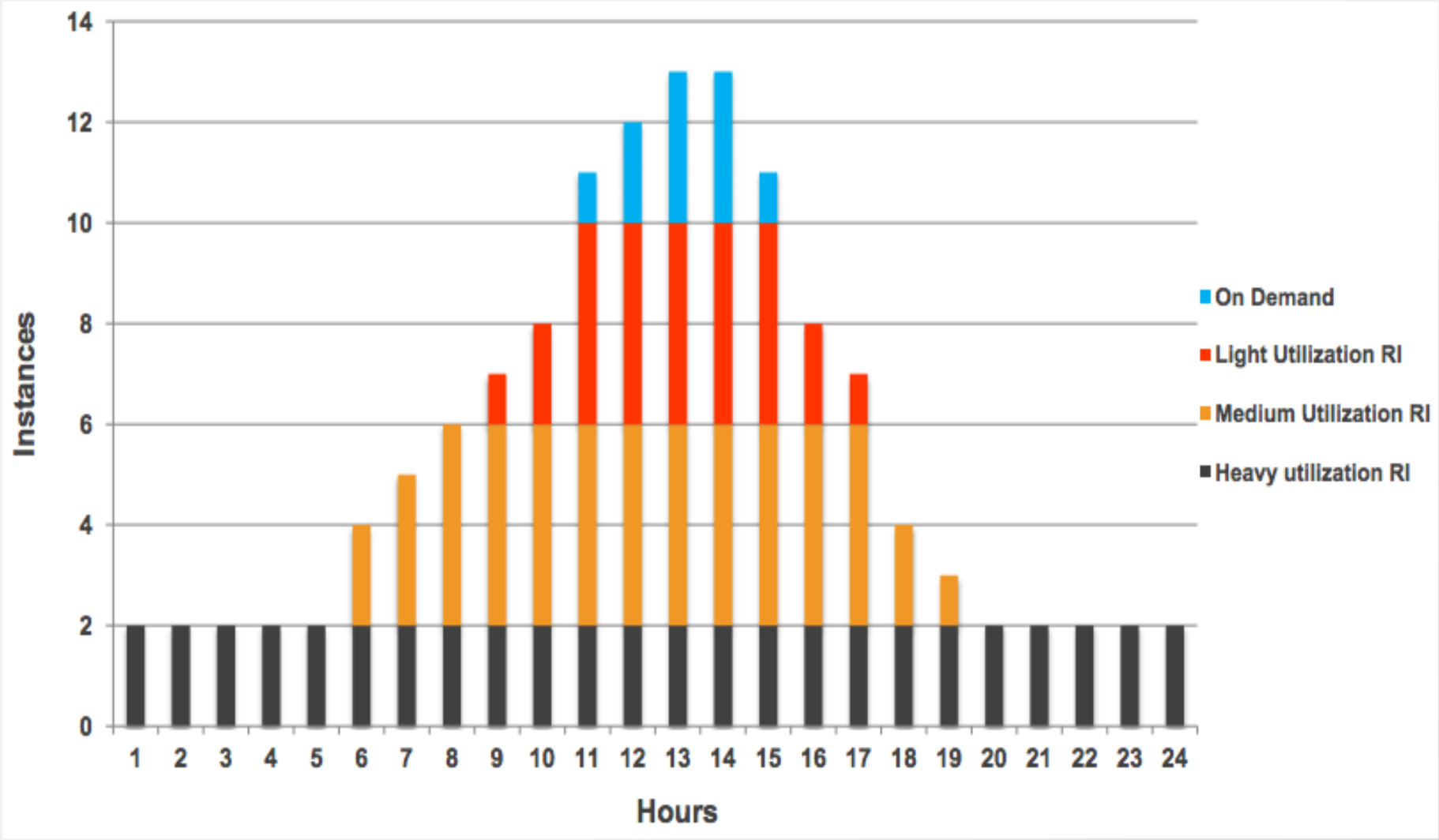
Reserved Instance Cost Savings Over On-Demand

(m1.large - Linux - One Year RI)

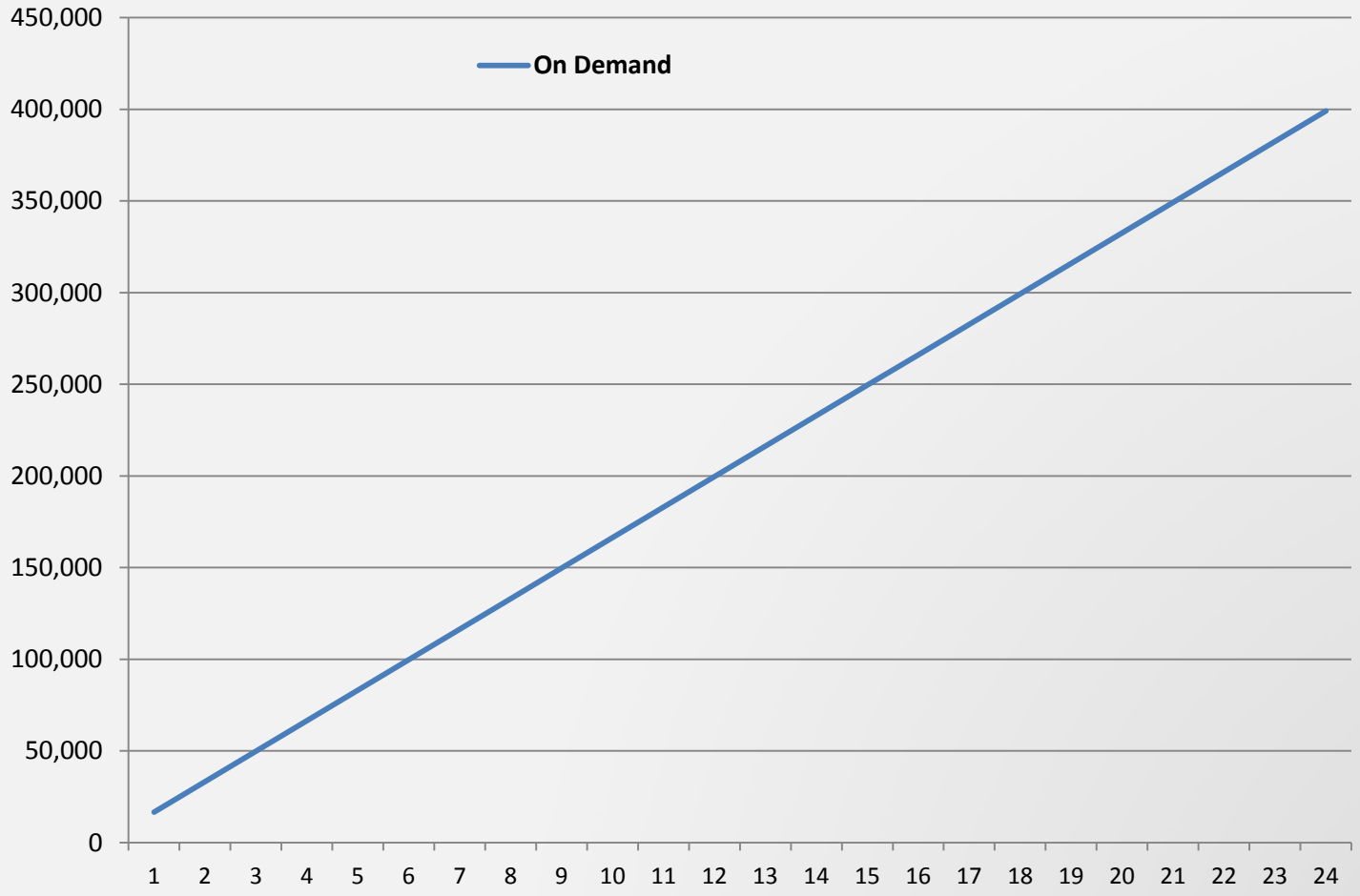
Annual Utilization	On Demand	Light Utilization RI	Medium Utilization RI	Heavy Utilization RI
10%	$234.00	-77.95%	-210.43%	-479.49%
20%	$468.00	-18.97%	-73.68%	-189.74%
30%	$702.00	0.68%	-28.09%	-93.16%
40%	$936.00	10.51%	-5.30%	-44.87%
50%	$1,170.00	16.41%	8.38%	-15.90%
60%	$1,404.00	20.34%	17.49%	3.42%
70%	$1,638.00	23.15%	24.00%	17.22%
80%	$1,872.00	25.26%	28.89%	27.56%
90%	$2,106.00	26.89%	32.69%	35.61%
100%	$2,340.00	28.21%	35.73%	42.05%

Optimal Savings Sub-Optimal Savings Least Savings

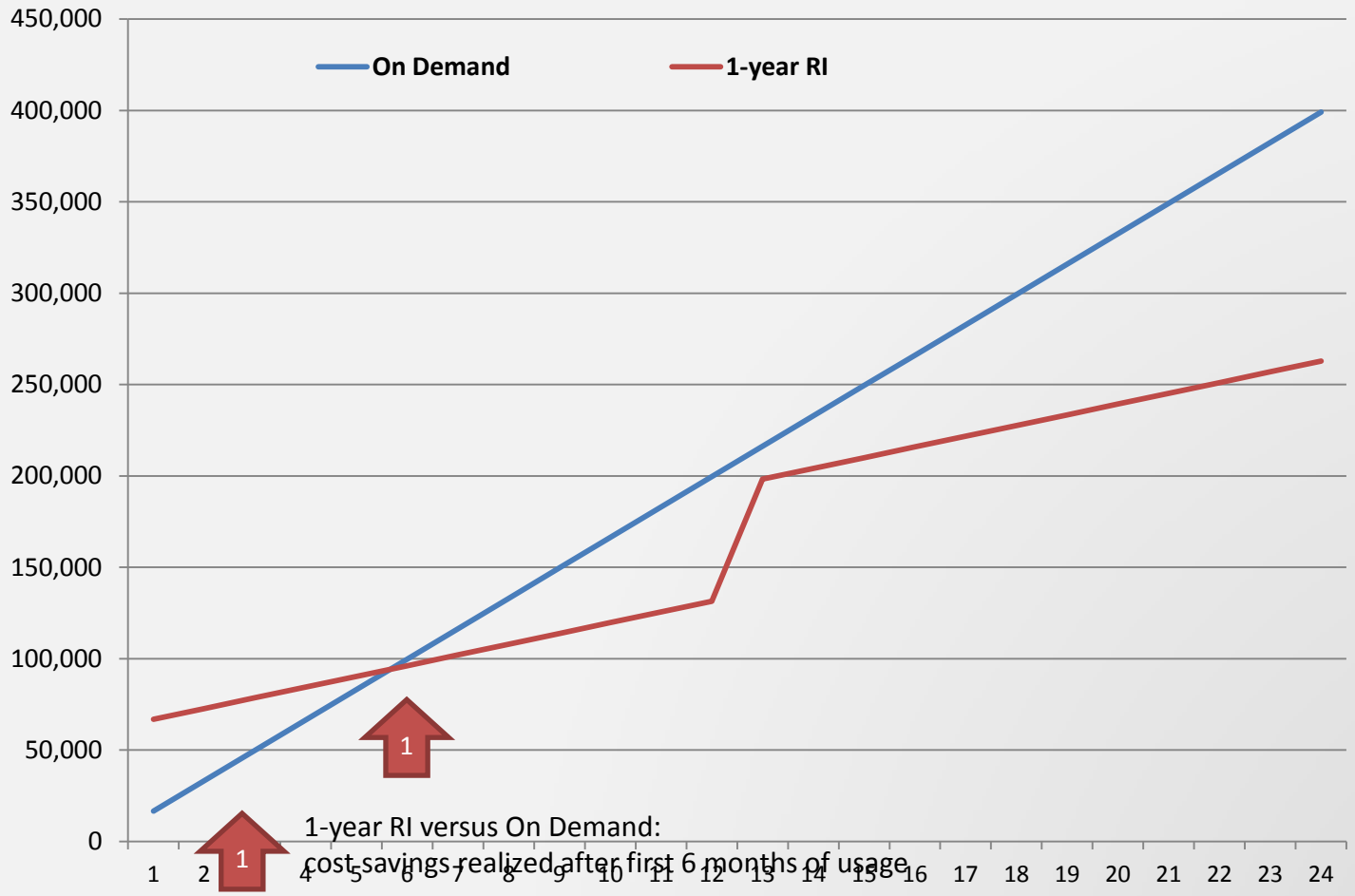
Reserved Instances



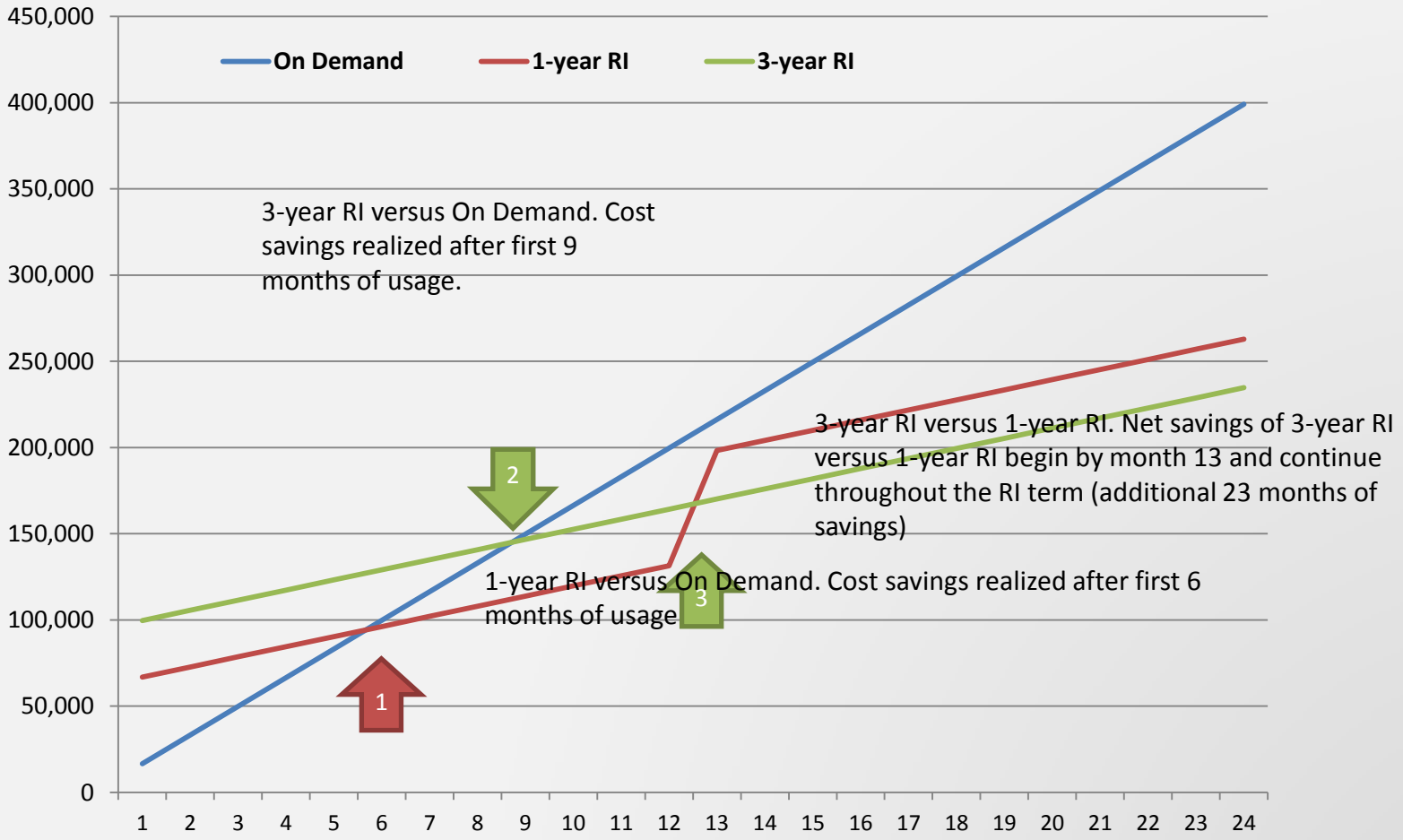
Reserved vs. On-Demand



Reserved vs. On-Demand



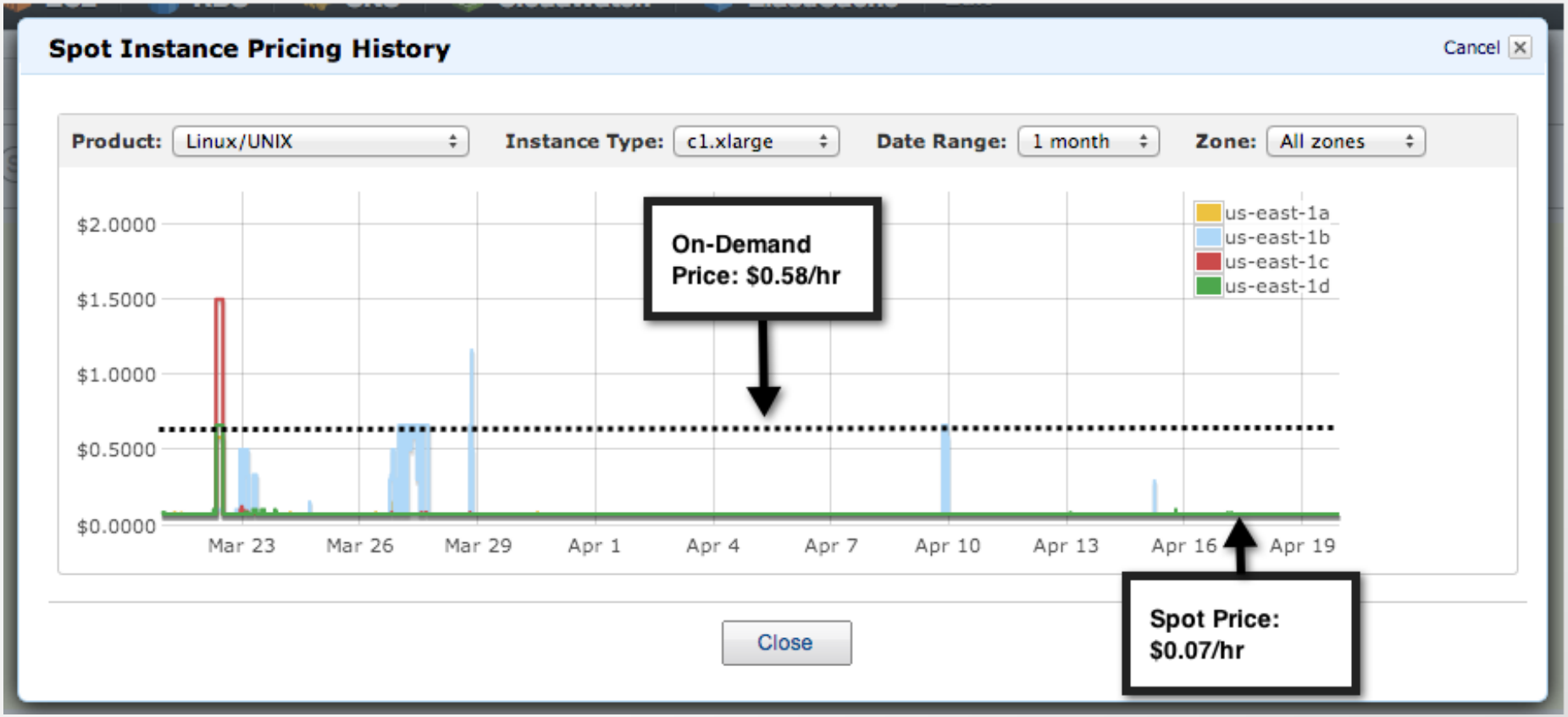
Reserved vs. On-Demand



Spot Market

- Spot instances often offer a significant savings over on-demand
- After an architecture is built for elasticity, leveraging spot instances can be a simple change

Spot Market



Spot Market

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

Spot Market

Best practices for using spot

- Save Your Work Frequently
- Add Checkpoints
- Split up Your Work
- Test Your Application

Spot Market

Best practices for using spot

- 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

4

Best practices

Minimize Always On instances

Elasticity is one of the fundamental properties of the cloud that drives many of its economic benefits

- Optimize your usage based on real-time demand
  - Reduce the number of web servers during off-peak periods
  - Shut down processing nodes unused at night
  - Purchase Reserved Instances for the Always On fleet

Scale-in automatically

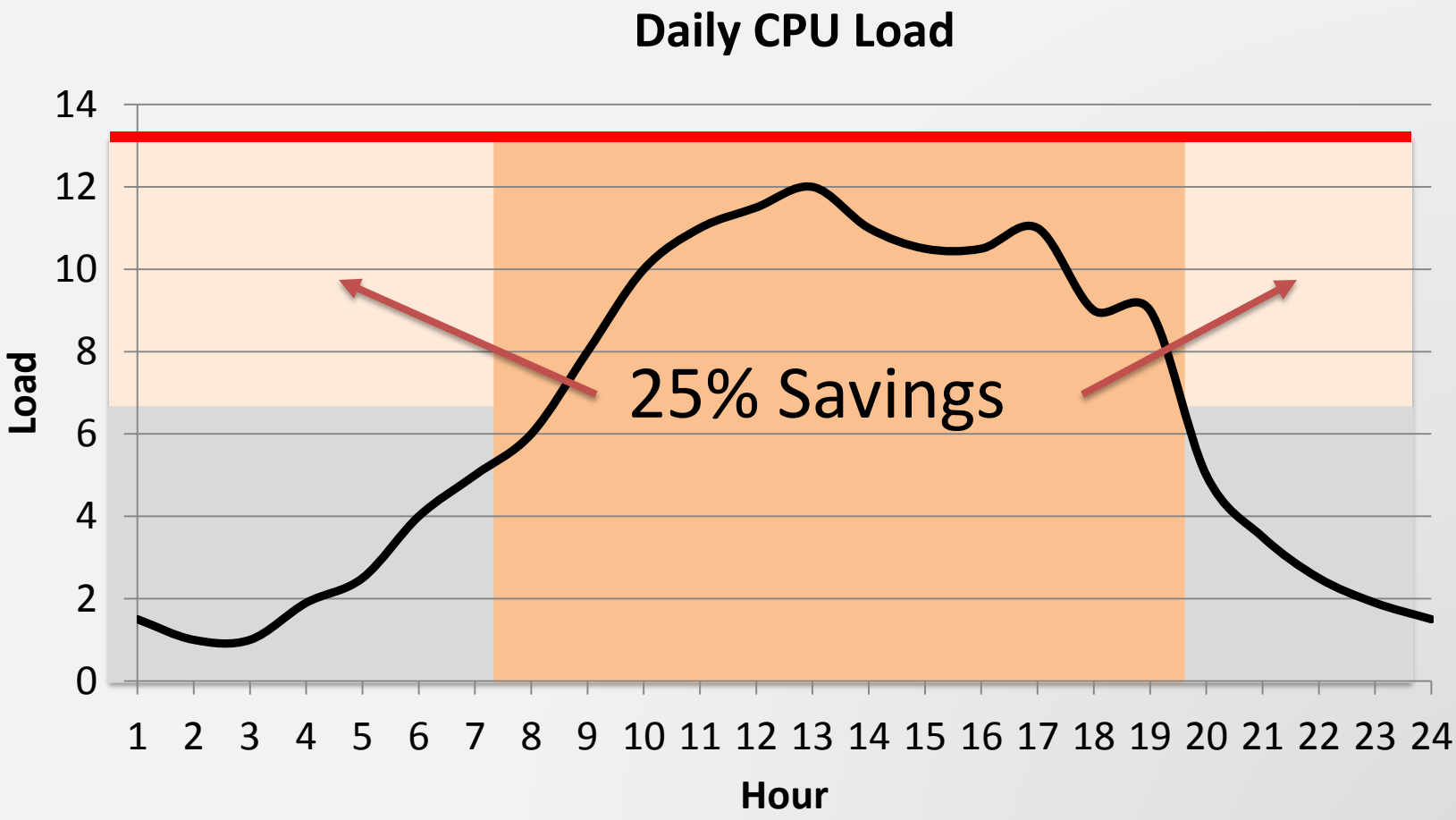
- Scaling out is key to serving customer needs but scaling in is where the savings occur
  - Auto-scaling works well for stateless components
  - Scripted scaling makes sense for stateful components
- Examples
  - Auto-scaling based on Network I/O for web servers
  - De-scaling RDS instances on the weekend
  - Shutting down all workers when batch processing queues are empty

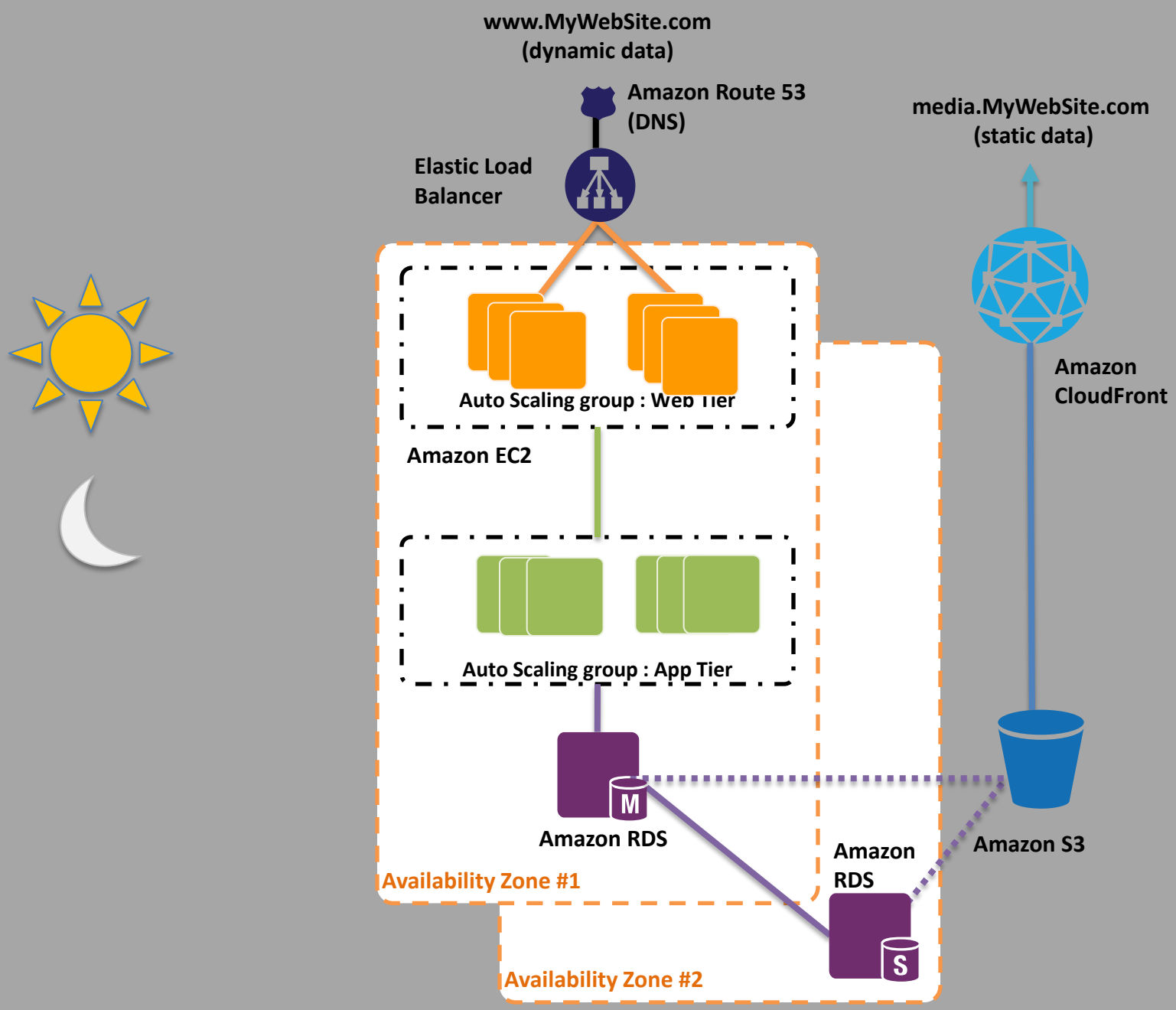
Scripted scaling

- Some scenarios do not lend themselves to automated de-scaling due to complex application logic
  - Shutting down worker nodes only when they are not currently working
  - Shutting down RDS read replicas during weekends

- Every AWS service has an API and command line tools for managing it
  - Scripting of scaling activities can be a significant cost savings for little effort

Optimize by time of day

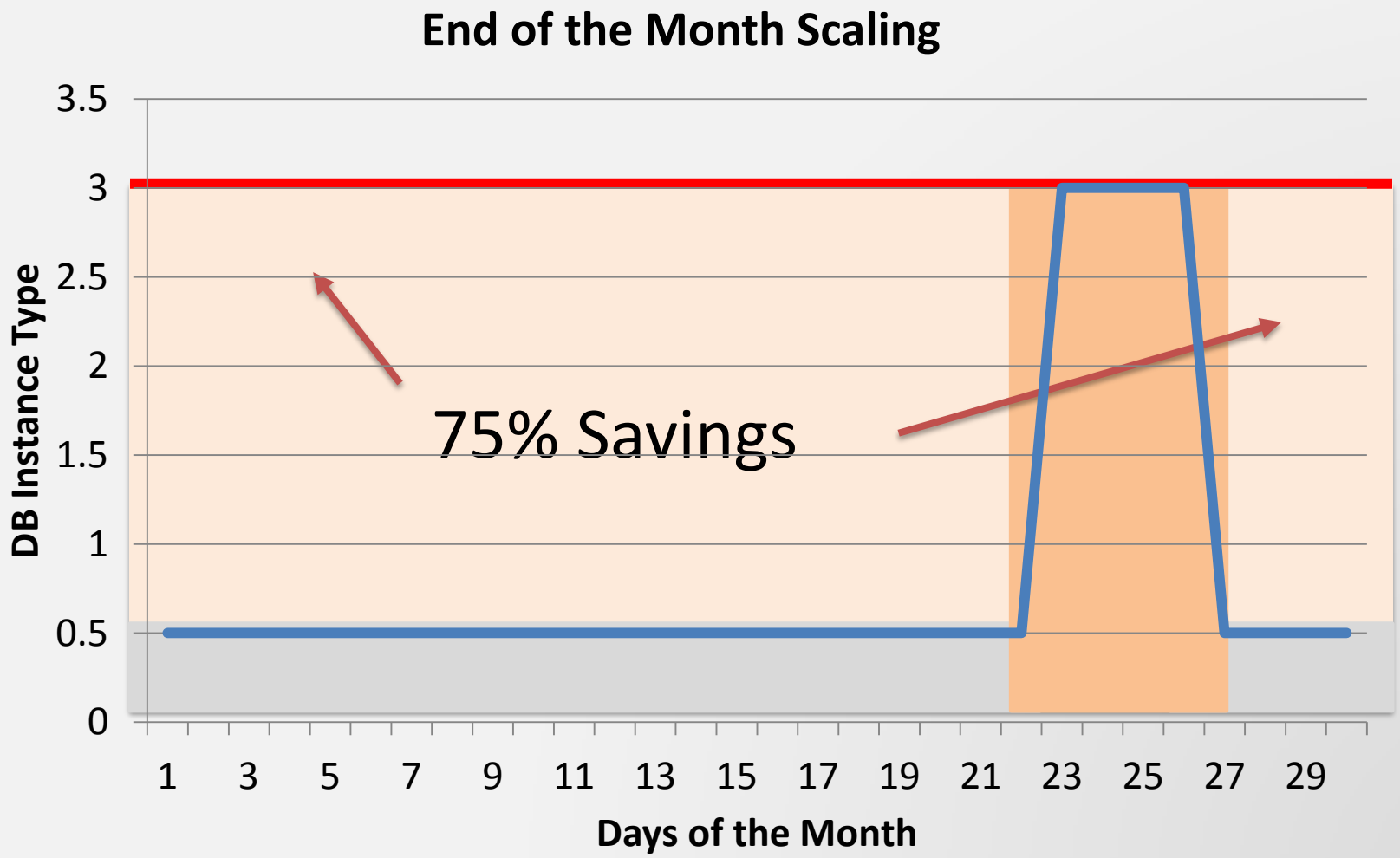



www.MyWebSite.com
(dynamic data)
Amazon Route 53
(DNS)
media.MyWebSite.com
(static data)
Elastic Load
Balancer
Amazon
CloudFront
Auto Scaling group : Web Tier
Amazon EC2
Auto Scaling group : App Tier
M
Amazon RDS
Amazon S3
Amazon
RDS
S
Availability Zone #1
Availability Zone #2

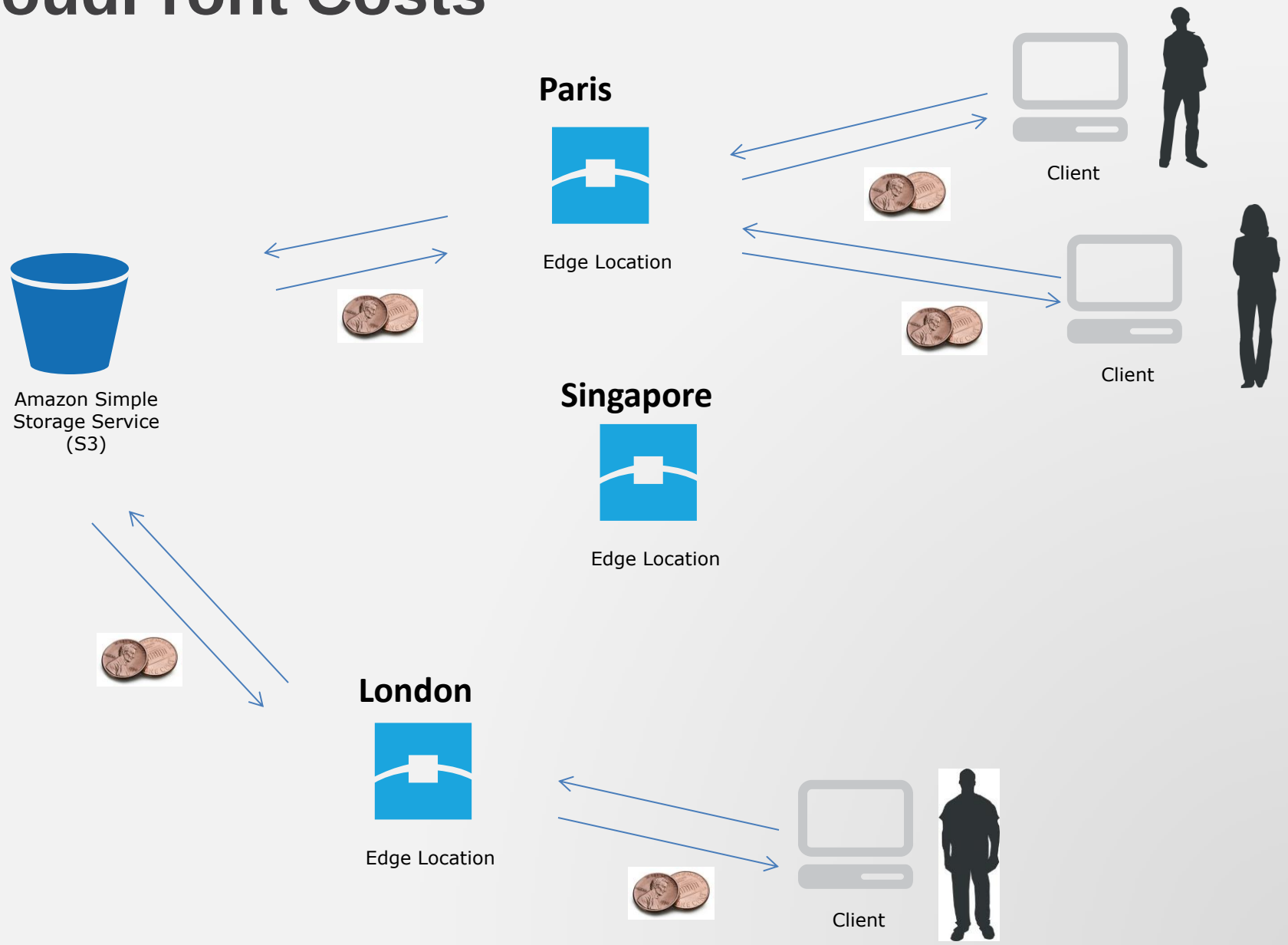
End of month processing

- 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)

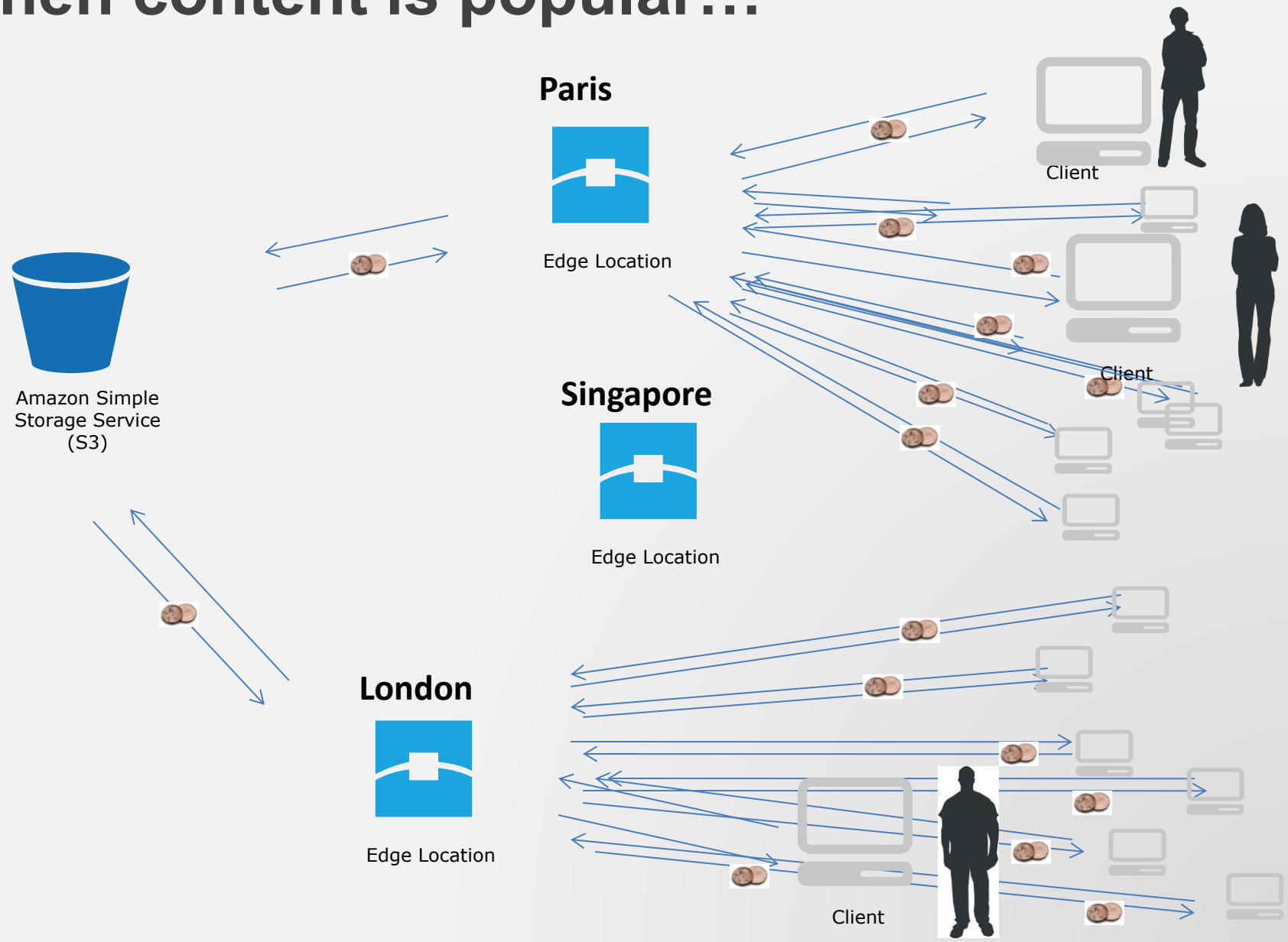
Optimize during the month



CloudFront Costs



When content is popular…



Leverage scalable, on-demand services

- EC2 can run almost anything but there are many cases where it is not cost effective
- AWS offers many scalable and cost-effective options for common application needs:
    - ELB instead of a software load balancer on EC2
    - SQS instead of a queue on EC2

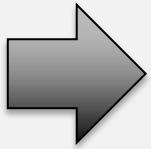
Software LB on EC2

Pros

- Application-tier load balancer

Cons

- SPOF
- Elasticity has to be implemented manually
- Not as cost-effective



ELB

Pros

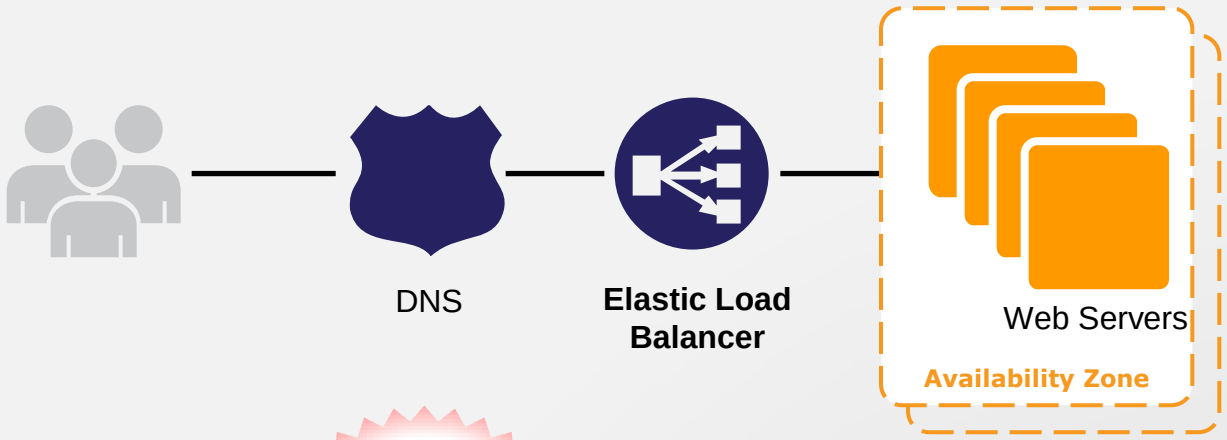
- Elastic and Fault-tolerant
- Auto scaling
- Monitoring included
- IPV6

Cons

- Internal load balancing only in VPC

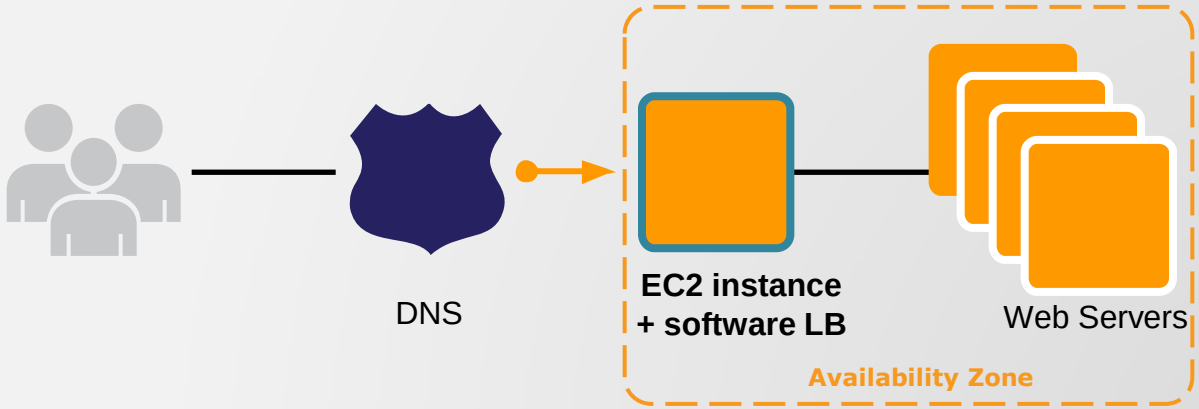
Designing for Cost | Best Practices

$0.025
per hour



VS.

$0.06
per hour
(m1.small)



Software versus Services

Software on EC2

Pros

- Custom features

Cons

- Requires an instance
- SPOF
- Limited to one AZ
- DIY administration

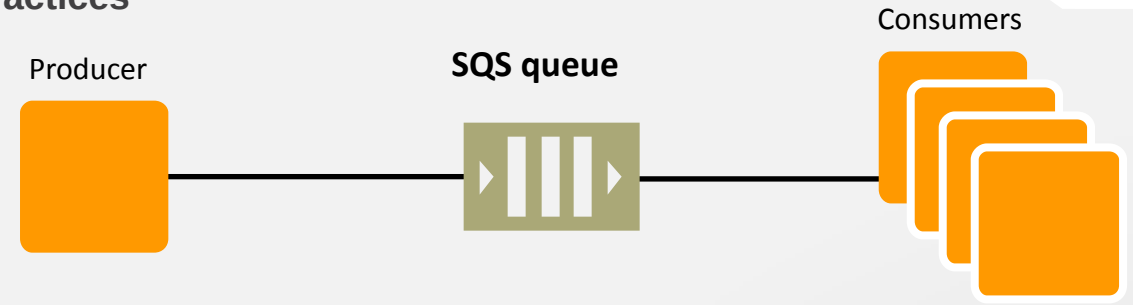
SNS, SQS, SES

Pros

- Pay as you go
- Scalability
- Availability
- High performance

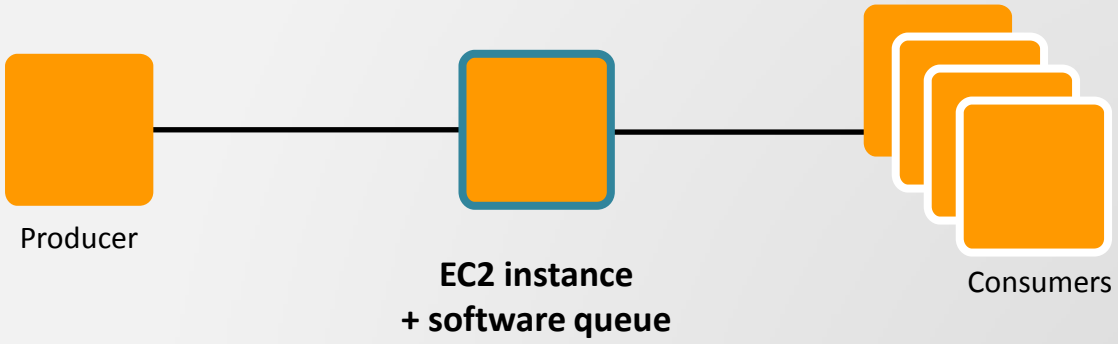
Designing for Cost | Best Practices

$0.01 per
10,000 Requests
($0.000001 per Request)



VS.

$0.095
per hour
(small instance)



Clean up after yourself

- When it is easy to create resources, it can be easy to forget about them
  - Use tagging to identify the purpose of resources
  - Use CloudWatch to identify underutilized resources
  - Keep track of objects in S3 and clean up unused content
  - Release unused Elastic IPs
- Examples
  - Daily report on utilization of resources
  - Clean up script to delete old S3 objects
- Make use of Trusted Advisor

Economics Center

http://aws.amazon.com/economics