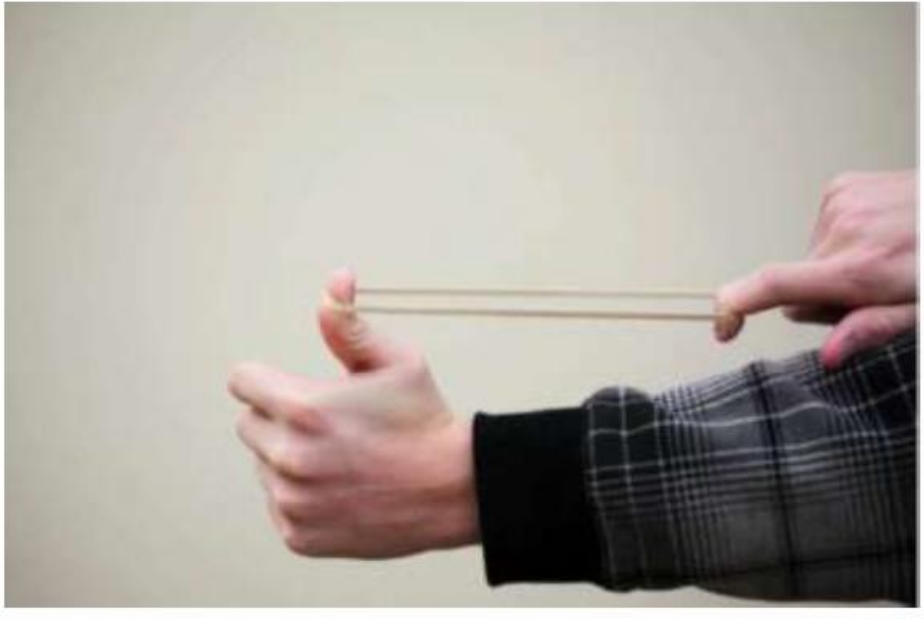
SCALING & ELASTICITY

Elasticity Fundamentals

What is elasticity?

- The ability to scale up for demand, then retract back when demand slows down
- Pay only for what you need, when you need it.



Scalability Fundamentals

- Scalability focuses more on building for growth
- Examples:
  - Increasing instance size
  - Increasing the number of available instances
  - Increasing volume capacity
- Various AWS services have different scalability and elasticity possibilities

DynamoDB – Scaling and elasticity

- Scalability:
  - We can keep storing more and more data without having to provision any hardware

- Elasticity:
  - We can increase or decrease read and write throughput capacity on demand
  - As read requests increase, we can increase read throughput capacity
  - As read requests slow down, we can decrease capacity

EC2 – Scaling and elasticity

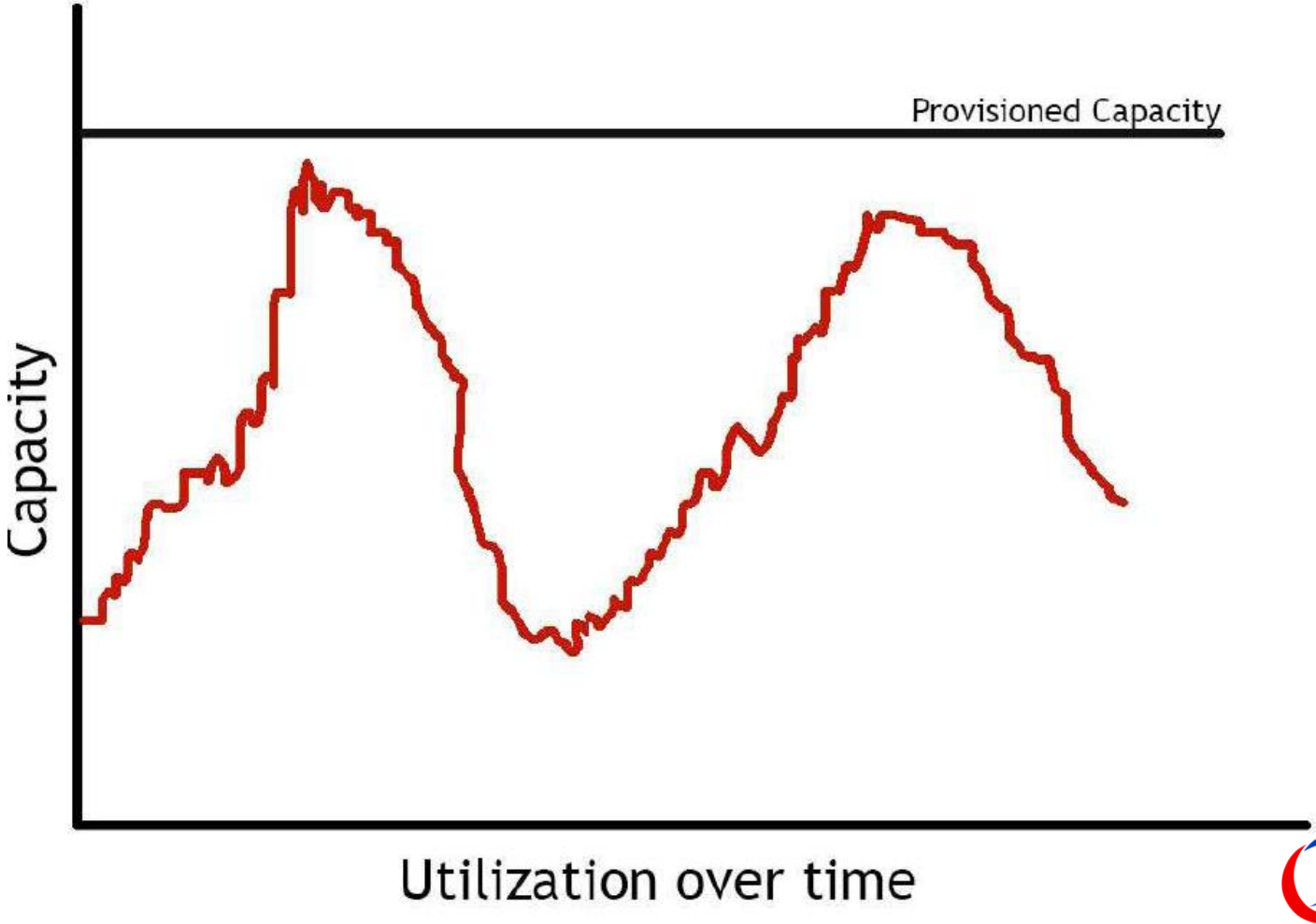
- Scalability:
  - We can increase the size of instances
  - There are different instance types we can choose from to grow
  - We can launch more instances

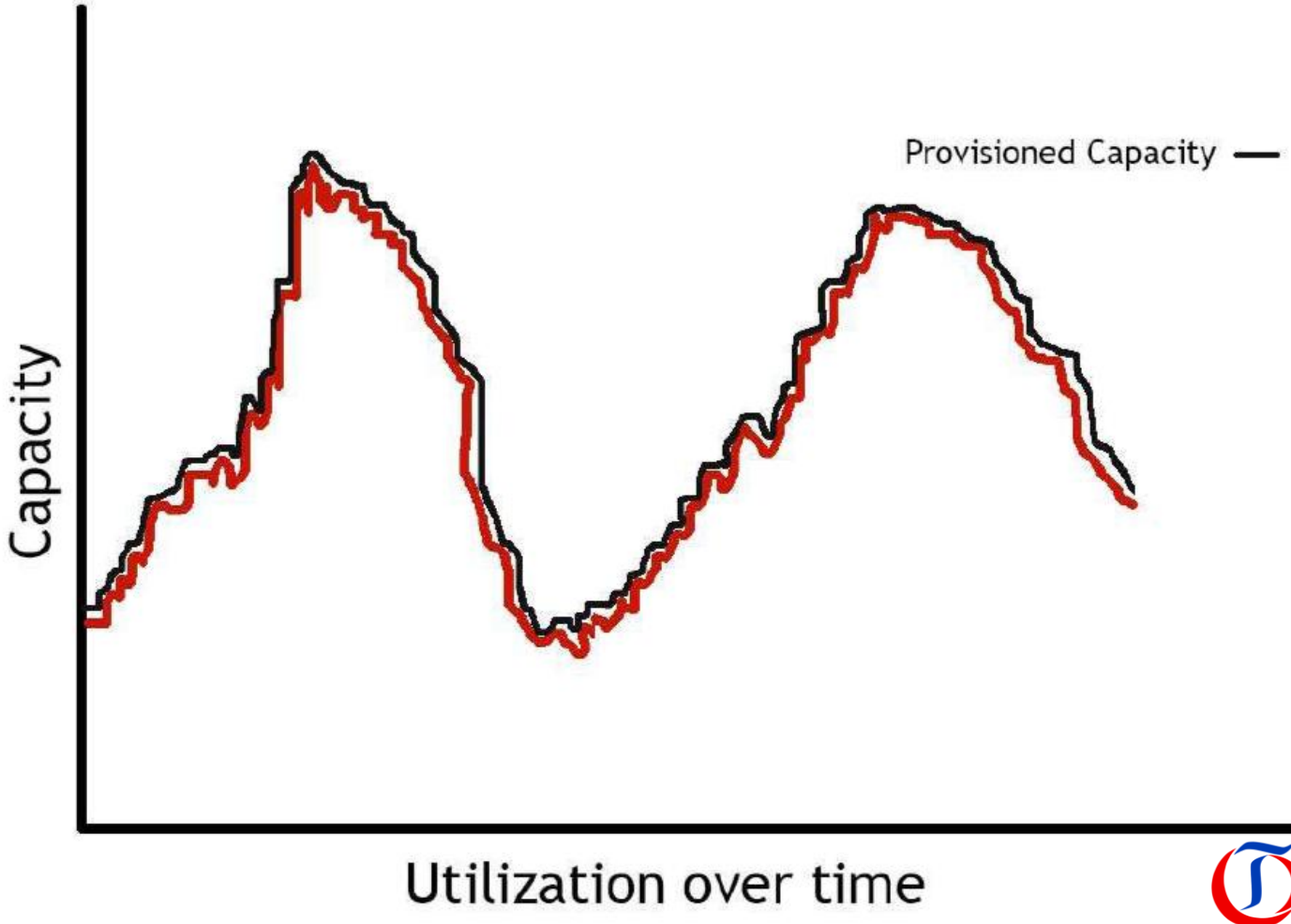
- Elasticity:
  - Auto Scaling gives the ability to grow with demand, and shrink back during slower periods

RDS – Scaling and elasticity

- Scalability:
  - We can increase the size of instances
  - Launch read replicas (for read-heavy databases)
  - There are different instance types we can choose from to grow

- Elasticity:
  - Limited





Reserved Instances

- Reserved instances give us the ability to purchase instance capacity for a specific period of time
- We can choose Standard Reserved Instances or Scheduled Reserved Instances
- Offers discounts
- Reserves capacity

Example #1

- Question: A company is using large T2 instances, but is expecting consistent growth and the need to upgrade to M4 instances towards the end of the year. M4 instances will put the company over budget, but they know that they'll be able to use that instance type for at least 3 years. What can they do?
- Solution: They could purchase reserved instances
- Explanation: Reserved M4 instances purchased under a 3 year term could offer significant discounts. Even if the company needs to change instance sizes, as long as they are still M4 instances running non-licensed Linux platforms, they can change their reserved instances at no extra cost.

Example #2

- Question: We work for an ecommerce platform that loses $x amount per minute of downtime. We setup Auto Scaling for elasticity. One day, during our peak hour of sales, AWS returns an error when Auto Scaling attempts to launch more instances: “InsuffientInstanceCapacity” which causes our instances to be overworked and miss requests. As a result, we lost a lot of sales. How can we avoid this in the future?

- Solution: We could purchase reserved instances

- Explanation: When we purchase reserved instances, we’re purchasing capacity. Even if we don’t need it 100% of the time, it’s there if we need it. That means we don’t have to rely on AWS having enough On-Demand capacity. We could also purchase Scheduled Reserved Instances for peak hours.

Reserved Instance Marketplace

- If requirements or needs change, we can sell reserved instances on the marketplace
- Sellers can avoid wasting capacity and money
- Buyers can get shorter terms

Amazon RDS and ElastiCache

- Reserved capacity is also available for Amazon RDS instances and ElastiCache nodes
- New generation of Reserved Cache Nodes only offer Heavy Utilization nodes, while older generations offer Heavy, Medium, and Light Utilization

Auto Scaling vs. Resizing instances

Auto Scaling

- Distributes the load across multiple instances
- Uses metrics and rules to automate spinning up/terminating instances

Changing instance sizes

- Increases/decreases resources available to our application

When to choose one over the other?

- They both have pros and cons

Auto Scaling vs. Resizing instances – Scenario #1

- Scenario: Our PHP application is growing in terms of demand and needs to be highly available. It should scale with demand and shrink back down during slower times. It should also be able to withstand an availability zone going down. For these reasons, we've implemented Auto Scaling.
- Should we also resize instances?
  - We may not want to launch a lot of smaller instances if we can launch fewer larger ones
  - We could launch specialized instances to meet the needs of our application if we need more of one type of resource (Compute Optimized, for example)

Auto Scaling vs. Resizing instances – Scenario #2

- Scenario: We have an application that processes customer orders with the help of SQS. Orders are added to a queue, which is then polled by backend instances that process the orders. To meet capacity, we launch a certain number of instances. The issue is that sales change depending on the season, time of day, and day of the week.
- Should we Auto Scale, resize instances, or both?
  - Upgrading instance sizes to meet peaks in sales would leave us overpaying during slow periods
  - We can use Auto Scaling to check the queue length, and adjust based off of that
  - Auto Scaling makes the most sense in this scenario

Scheduled Scaling

- Auto Scaling can scale or shrink on a schedule
  - One time occurrence or recurring schedule
  - Can define a new minimum, maximum, and scaling size
  - Lets you scale out before you actually need capacity in order to avoid delays

Challenges of Auto Scaling

- Auto Scaling is relatively complicated to setup
  - Instances can be started and stopped at any time
  - Applications need to be designed to handle distributed work
  - Important data (sessions, images, etc…) needs to be stored in a central location
  - If one server terminates, the application should still function

- Delays in scaling
  - Instances take time to initialize
  - Applications may require setup which could take even more time

Challenges of Resizing Instances

- Compatibility
  - Instances must have the same virtualization type to resize
  - Incompatible instances require migration
- EBS-backed instances need to be stopped before resizing
- Instance store-backed instances require migration by creating an image and launching a new instance from that image
- Resizing isn't very flexible compared to Auto Scaling
- There usually has to be downtime and careful planning
- Resizing instances in Auto Scaling groups may need "suspending"