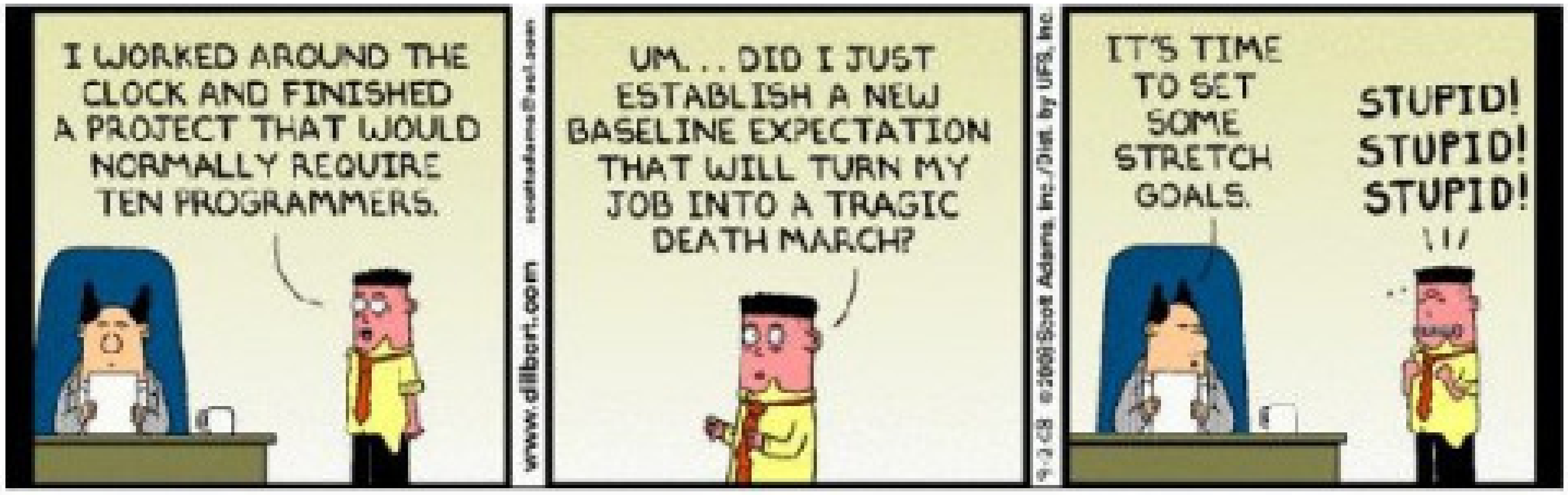
# Auto Scaling

CS4230 – Distributed and Cloud Computing

Jay Urbain

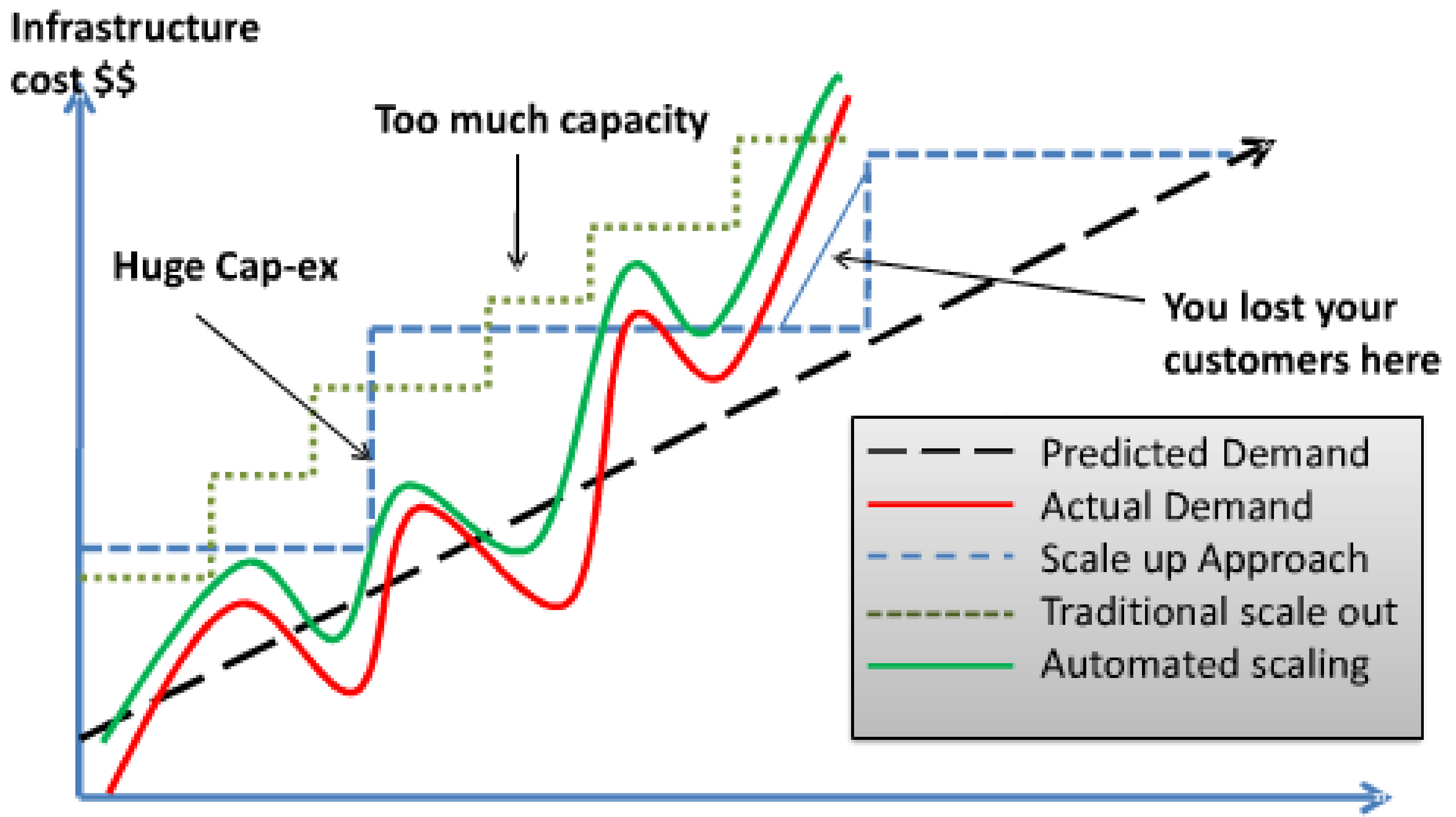
Credits: AWS, www.8kmiles.com


I WORKED AROUND THE CLOCK AND FINISHED A PROJECT THAT WOULD NORMALLY REQUIRE TEN PROGRAMMERS.
www.dilbert.com
scottadams@aol.com
UM... DID I JUST ESTABLISH A NEW BASELINE EXPECTATION THAT WILL TURN MY JOB INTO A TRAGIC DEATH MARCH?
9-3-08 © 2008 Scott Adams, Inc./Dist. by UFS, Inc.
IT'S TIME TO SET SOME STRETCH GOALS.
STUPID! STUPID! STUPID!

## Auto Scaling with AWS

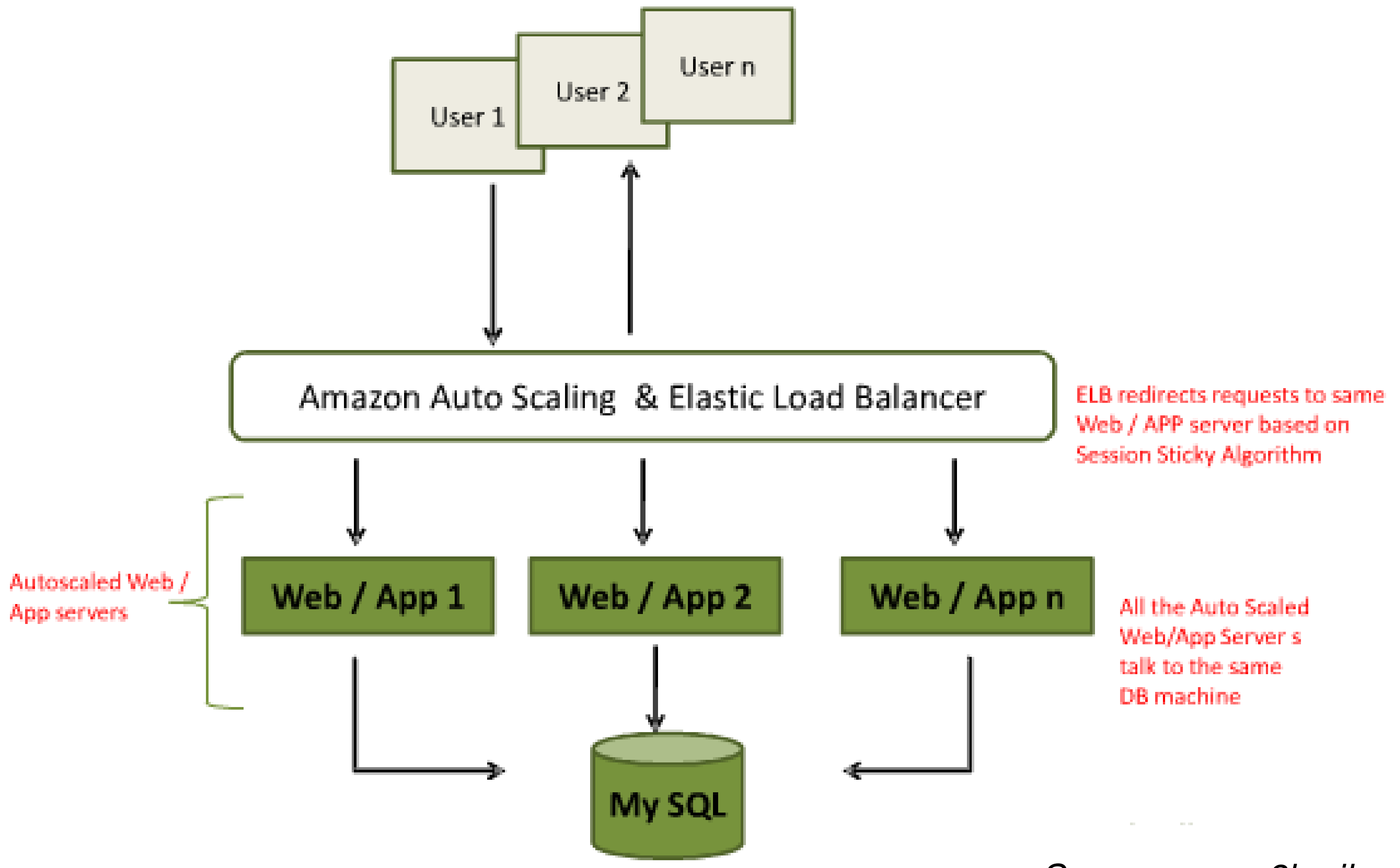
- Objective:
  - Ensure the number of Amazon EC2 instances increases during demand spikes to maintain performance, and
  - Decreases during demand lulls to minimize costs.
- *Auto Scaling* allows you to scale your EC2 capacity up or down automatically according to conditions you define.
- Well suited for applications that experience hourly, daily, or weekly variability in usage.

# Understanding Scaling



Source: Amazon AWS

# Auto Scaling Architcture


User 1
User 2
User n
Amazon Auto Scaling & Elastic Load Balancer
ELB redirects requests to same Web / APP server based on Session Sticky Algorithm
Autoscaled Web / App servers
Web / App 1
Web / App 2
Web / App n
All the Auto Scaled Web/App Server s talk to the same DB machine
My SQL

Source: *www.8kmiles.com*

## AWS Solution Components

Solution Components

- Elastic Block Storage (EBS) – server images
- Simple Storage Service (S3) – storing objects as key value pair
- Simple Queue Service (SQS) – queue
- Elastic Load Balancer (ELB) – load balancer
- AutoScale – for scaling servers up and down automatically
- SimpleDB – scalable database

## Solution Components - EBS

- Elastic Block Storage (EBS) – Provides block level storage volumes for use with Amazon EC2 instances.
- EBS is well suited for applications that require a database, file system, or access to raw block level storage.
- Sample Use case:
  - Data stores, application executables, configurations, and OS are installed in the EBS.

# Solution Components – S3

- Simple Storage Service (S3) – Provides a simple web services interface that can be used to store /retrieve any amount of data, at any time, from anywhere on the web.
- Sample Use case :
  - Uploaded data and files, generated reports and data are stored in S3.

## Solution Components – SQS

- Simple Queue Service (SQS) – Reliable, highly scalable, hosted queue for storing messages as they travel between computers, i.e., EC2 instances.
- Sample Use case:
  - Meta data about the files/data to be processed are put on the queue for processing.
  - Background application picks up the meta data from the SQS and accesses and processes the data from a data store, i.e.,S3, Simple DB, or relational database.

## Solution Components – Simple DB

- Simple DB – Highly available, scalable, and flexible *non-relational* key-value data store.
- Store and query data items via web services requests.
- Sample Use case :
  - Store data record by recordID.
  - Store inter-application information can be stored in Simple DB.

## Solution Components – ELB

- Elastic Load Balancer (ELB) - Automatically distributes incoming application traffic across multiple Amazon EC2 instances.
- Detects unhealthy instances within a pool and automatically reroutes traffic to healthy instances until the unhealthy instances have been restored.
- Sample Use case:
  - Dynamically distribute work load among Servers located in multiple zones.
  - Can use dynamically Auto Scaled EC2 instances.

## Solution Components – Auto Scaling

- Auto Scaling – Automatically scale EC2 capacity up or down according to conditions you define.
- Well suited for applications that experience hourly, daily, or weekly variability in usage.
- Sample Use case:
  - Dynamically scale EC2 instances up and down depending upon current workload.
  - Dynamically add new EC2 instances to replace “unhealthy” instances.

## Auto Scaling Setup

Download from EC2 API Tools (main page):

- http://aws.amazon.com/developertools?_encoding=UTF8&jiveRedirect=1

- Auto Scaling Tools

http://aws.amazon.com/developertools/2535

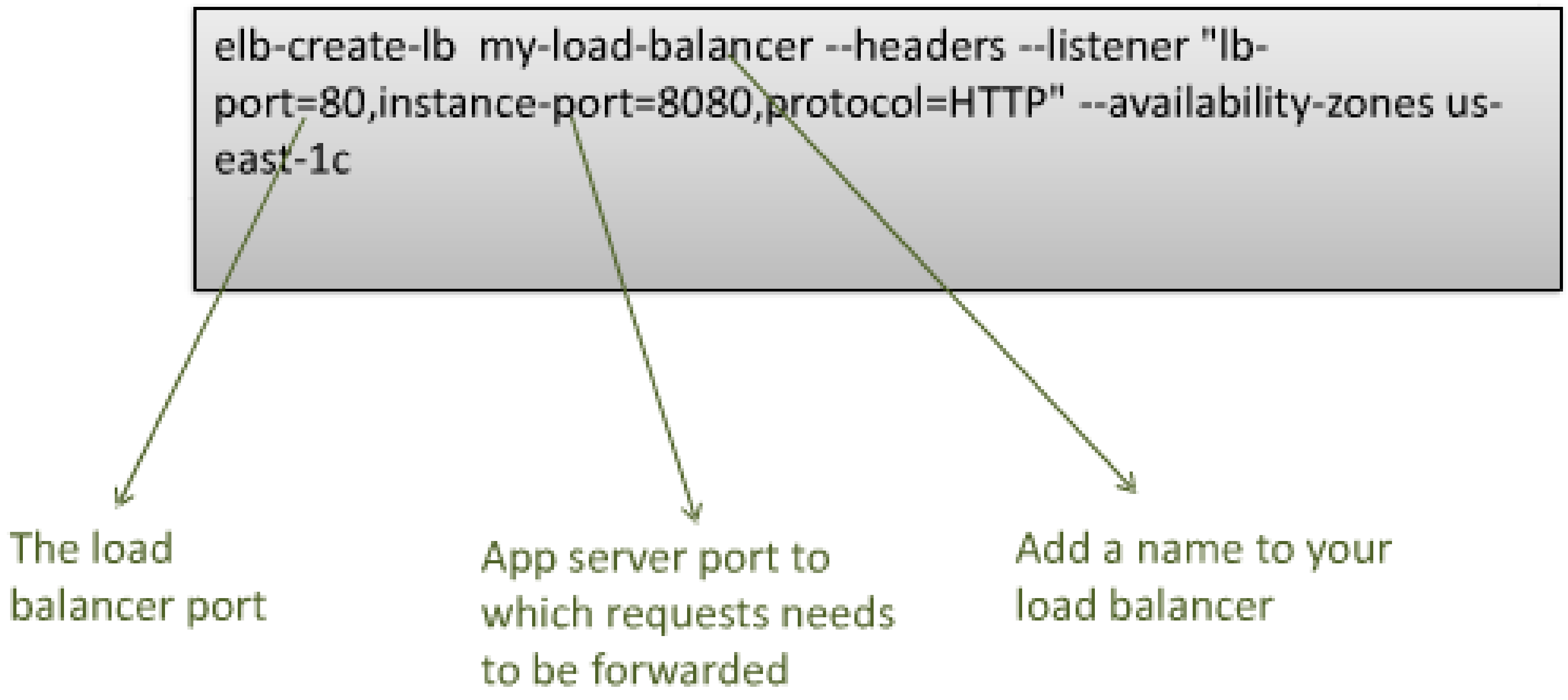
- CloudWatch Command Line Tools

http://aws.amazon.com/developertools/2534

- EC2 API Tools (while your there!)

http://aws.amazon.com/developertools/351

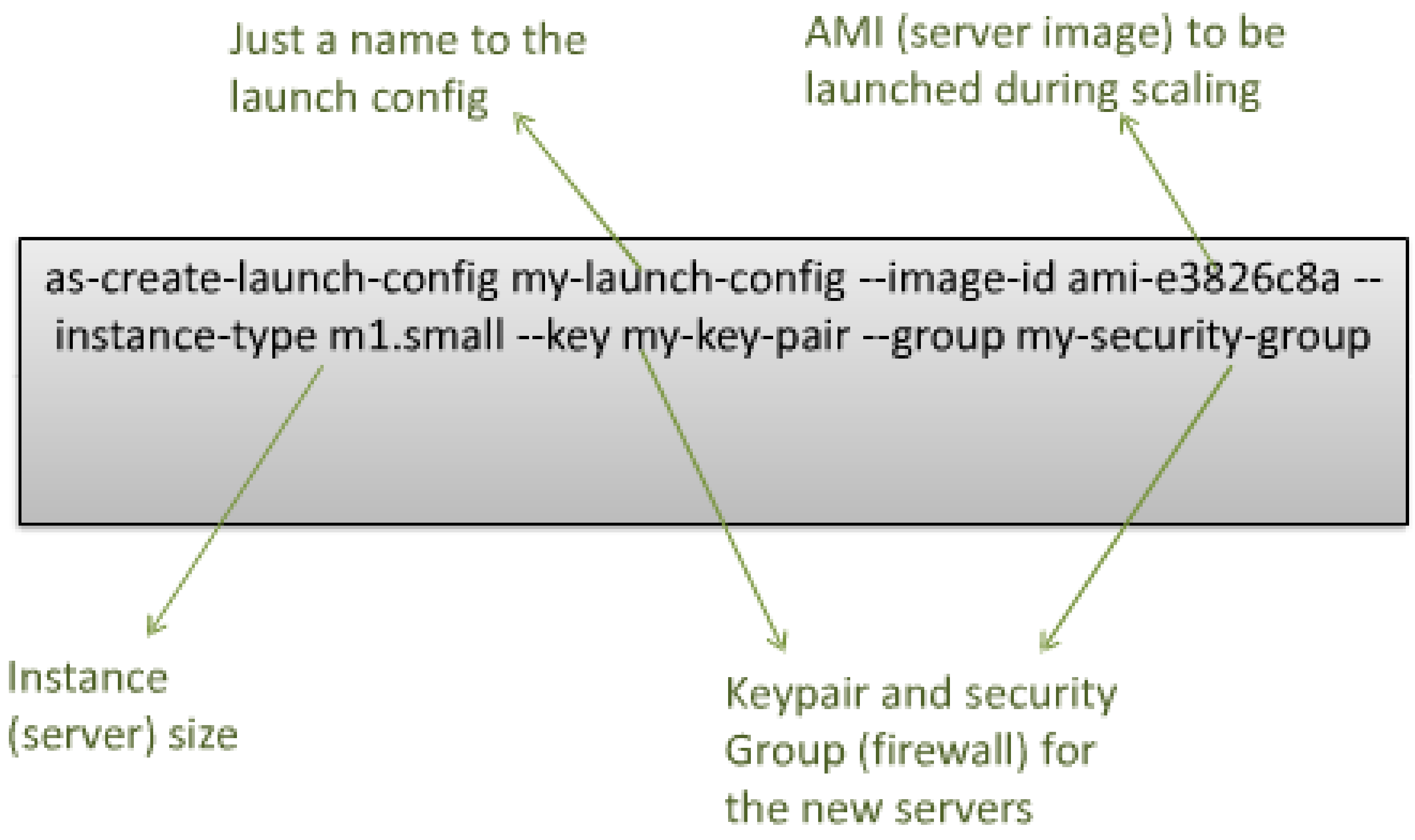
## Elastic Load Balancer



```
elb-create-lb  my-load-balancer --headers --listener "lb-port=80,instance-port=8080,protocol=HTTP" --availability-zones us-east-1c
```

The load balancer port
App server port to which requests needs to be forwarded
Add a name to your load balancer

## Launch Configuration



```
as-create-launch-config my-launch-config --image-id ami-e3826c8a --
instance-type m1.small --key my-key-pair --group my-security-group
```

Just a name to the launch config
AMI (server image) to be launched during scaling
Instance (server) size
Keypair and security Group (firewall) for the new servers

## Auto Scaling Group

Name your auto Scale group

Min and Max number of instances to be spawned

```
as-create-auto-scaling-group my-as-group --availability-zones us-east-
1c --launch-configuration my-launch-config --max-size  11  --min-size
3 --cooldown  180 --load-balancers my-load-balancer
```

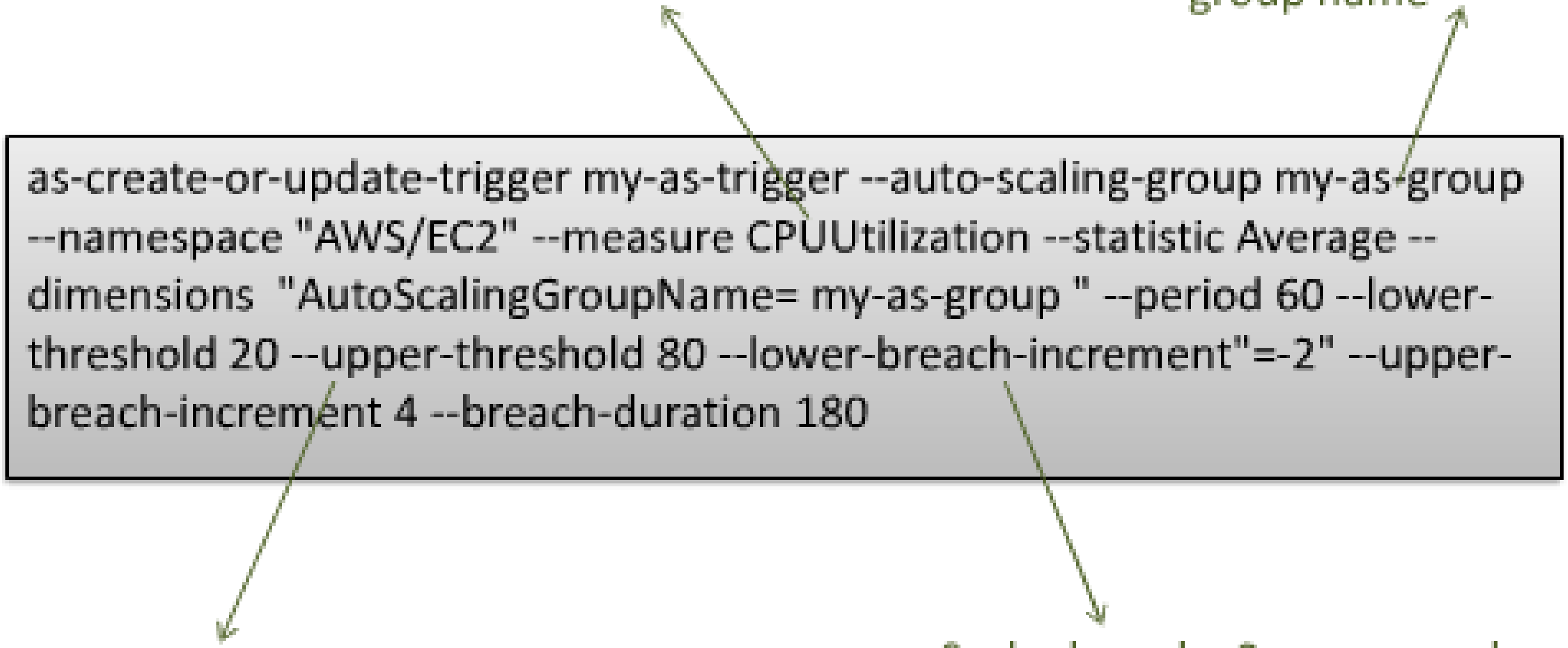
Mention the launch config (the one we created in last step)

Specify the load balancer to which the new servers needs to be attached

# Configure Triggers

Measure the avg CPU of the autoscale group

Specify the autoscale group name



```
as-create-or-update-trigger my-as-trigger --auto-scaling-group my-as-group
--namespace "AWS/EC2" --measure CPUUtilization --statistic Average --
dimensions  "AutoScalingGroupName= my-as-group " --period 60 --lower-
threshold 20 --upper-threshold 80 --lower-breach-increment"=-2" --upper-
breach-increment 4 --breach-duration 180
```

Lower CPU limit is 20% and upper CPU limit is 80%

Scale down by 2 servers and scale up by 4 servers

## CloudWatch Monitoring

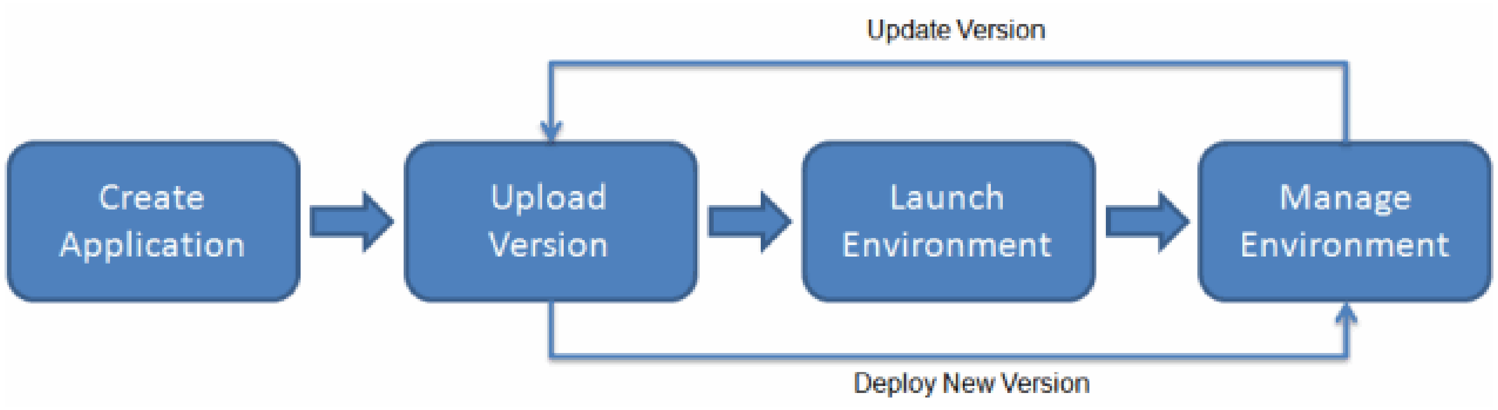
- Use the *mon-put-metric-alarm* CloudWatch command to create an alarm for each condition under which you want to add or remove Amazon EC2 instances (or use Management Console).
- Specify the Auto Scaling Policy that you want the alarm to execute when that condition is met.
- You can define alarms based on any metric that Amazon CloudWatch collects. E.g. of metrics on which you can set conditions include average CPU utilization, network activity or disk utilization.
- Auto Scaling tracks when your conditions have been met and automatically takes the corresponding scaling action on your behalf.

## AWS Elastic Beanstalk

- Elastic Beanstalk automatically handles the details of load balancing, scaling, and application monitoring.
- It’s basically how Amazon competes with Google AppEngine.
- Uses AWS technologies:
  - Amazon Elastic Compute Cloud (Amazon EC2)
  - Amazon Simple Storage Service (Amazon S3)
  - Amazon Simple Notification Service (Amazon SNS)
  - Amazon CloudWatch
  - Elastic Load Balancing
  - Auto Scaling

## Using AWS Elastic Beanstalk

- Create an application, upload an application version (for example, a Java WAR file) to AWS Elastic Beanstalk.
- Provide some information about the application.
- Elastic Beanstalk launches an environment and creates and configures the AWS resources needed to run your code.
- After your environment is launched, you can then manage your environment and deploy new application versions.



## Using AWS Elastic Beanstalk

- You can use Java with the AWS Toolkit for Eclipse (ADT plugin).
- Toolkit includes the AWS libraries, project templates, code samples, and documentation.
- Supports Java 5 or Java 6.
- AWS Elastic Beanstalk supports the following container types:
  - 32-bit Amazon Linux running Tomcat 6
  - 64-bit Amazon Linux running Tomcat 6
  - 32-bit Amazon Linux running Tomcat 7
  - 64-bit Amazon Linux running Tomcat 7
- There is no additional charge for AWS Elastic Beanstalk; you pay only for the underlying AWS resources that your application consumes