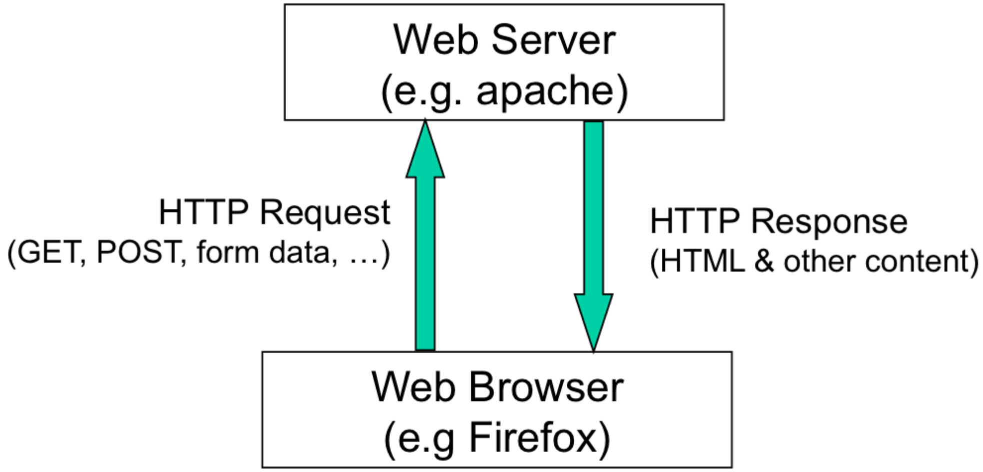
# Developer Operations

## Performance & Scaling

# HTTP dialogue

- Communication on the web means browsers talking to web servers over HTTP



- Within a browsing session the user is (ideally) given an illusion that they are having a one-to-one dialogue with the server
- Part of this illusion is management of **state**
- Another part is achieved by **performance** and **scalability** management

# Performance issues

- Response time
  - How quickly does a server respond to client requests?
  - Affected by
    - Server load
    - Network load
    - Reliance on third-party services (e.g. DNS)
    - Delays in all software components (server & client OS, process switching, server & client network stack)

# Performance issues

- Throughput
  - Rate at which computational work is done
  - Transactions per unit time
  - Affected by
    - Processor speed
    - Storage performance (memory/disk)
    - Some resources overloading badly (e.g. network bottlenecks)
    - Reliance on OS and network services
    - Reliance on third-party services

# Issues related to performance

- Reliability
  - Uptime, ideally 99.999...%
  - Consistency
- Fault tolerance
  - Achieved through redundancy of components, etc
- Adaptability
- Security
  - Can add to load on processor, network, …
- Quality of Service / Quality of Experience
  - Combination of performance, availability, adaptability, security

# How to improve performance

- Better network speeds
- Faster CPU
- More memory
- More disk space
- Less process switching
- Better design – less bottlenecks, better concurrency (e.g. less resource holding)

- All of the above can help performance of individual servers, but don’t **scale** indefinitely due to physical limits
  - Also decreasing cost effectiveness

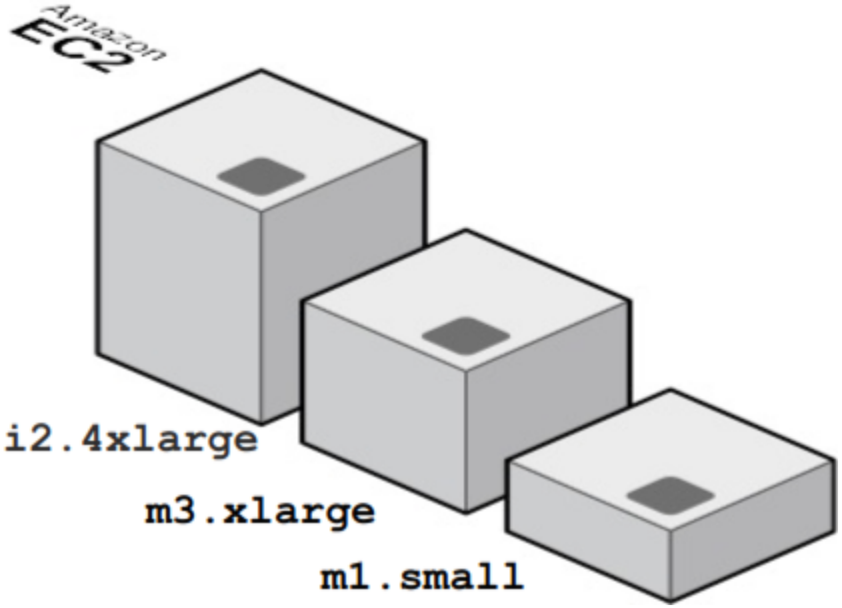
# Scalability

- Ability to grow in size according to need
  - While maintaining performance
- Removal of limits to performance
- Elasticity is one of the main benefits of cloud computing

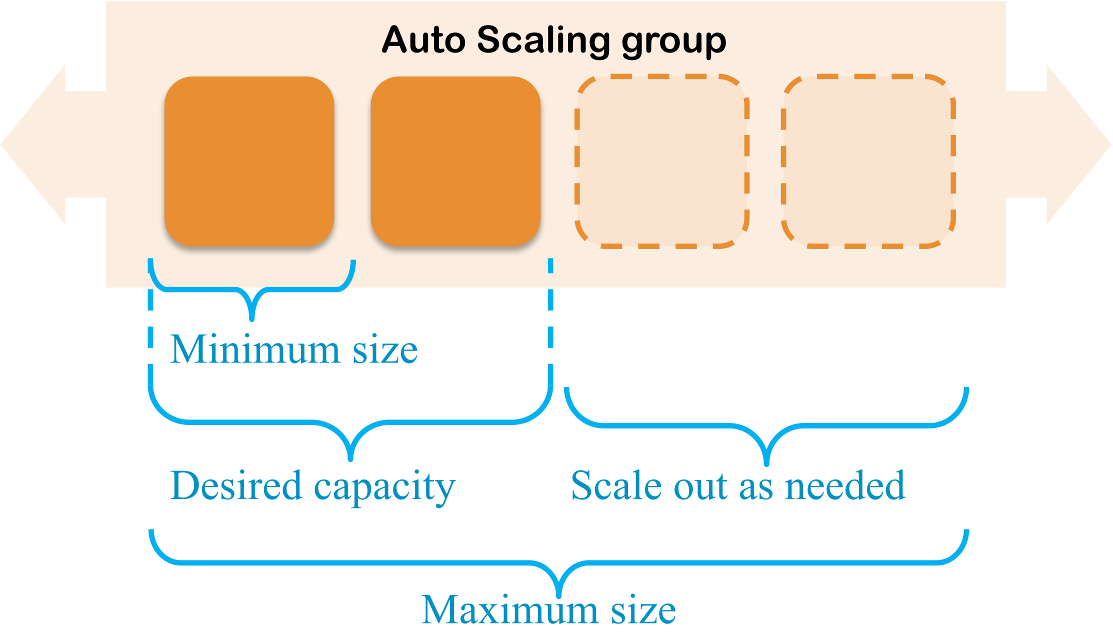
- One way to remove performance limits is to add resources to each element
  - **Vertical** scaling (scale **up**)
- Another way to remove performance limits is to replicate elements (and balance the load between them)
  - **Horizontal** scaling (scale **out**)

# Vertical Scaling: Scale up in AWS

- Simple approach.
- High memory/IO/CPU/ Storage.
- Easy to change instance size.
- Will ultimately hit limit.



# Horizontal Scaling: AWS Auto Scaling



Auto Scaling groups contain a collection of EC2 instances that share similar characteristics.

Instances in an Auto Scaling group are treated as a logical grouping for the purpose of instance scaling and management.

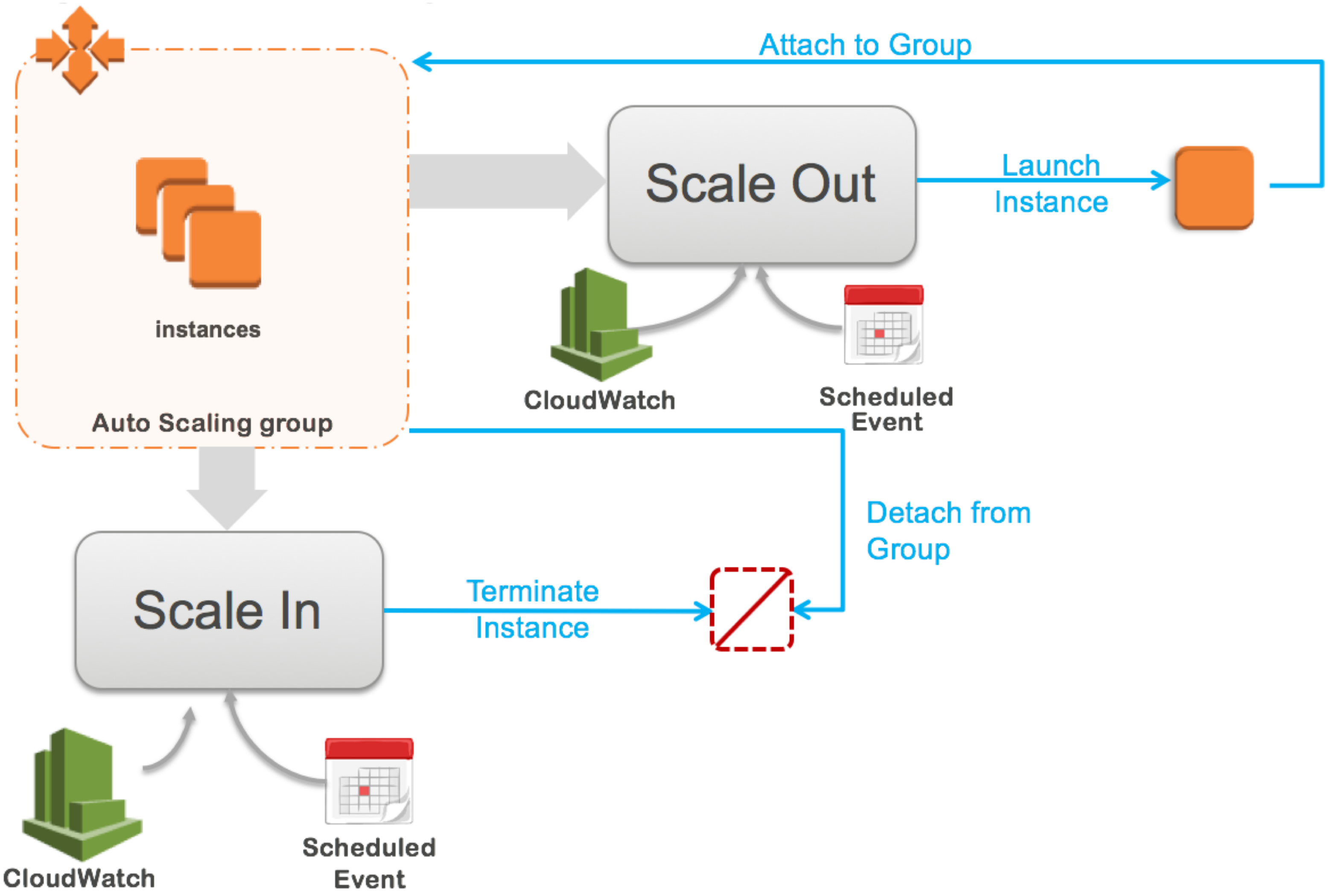
# Dynamic Scaling in AWS

You can create a scaling policy that uses **CloudWatch alarms** to determine when your **Auto Scaling group** should…

You can use alarms to monitor:

- Any of the metrics that AWS services send to CloudWatch
- Your own custom metrics

# Auto Scaling Basic Lifecycle



# AWS Launch Configurations

- A **launch configuration** is a template that an Auto Scaling group uses to launch EC2 instances
- When you create a launch configuration, you can specify:
  - AMI
  - Instance type
  - Key pair
  - Security groups
  - Block device mapping
  - User data

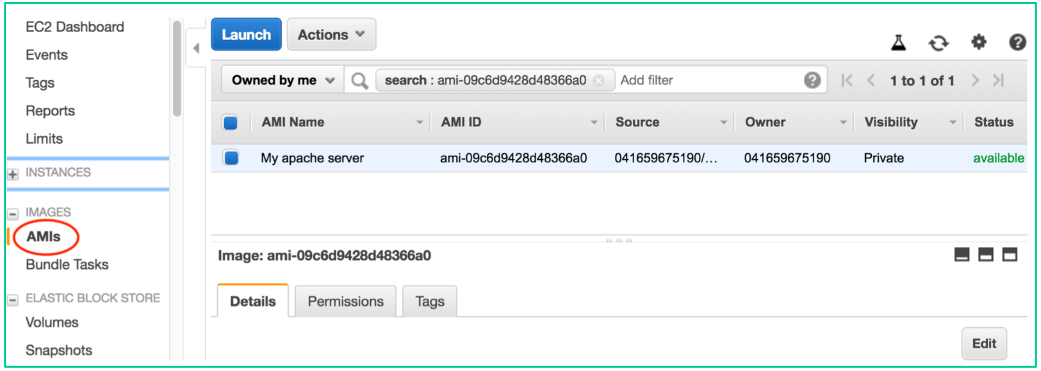
- This provides a **blueprint** for each instance that is automatically started by the auto scaler

# Next: Try out AWS auto-scaling

- Create AMI for launch config

- Create Auto Scaling Group

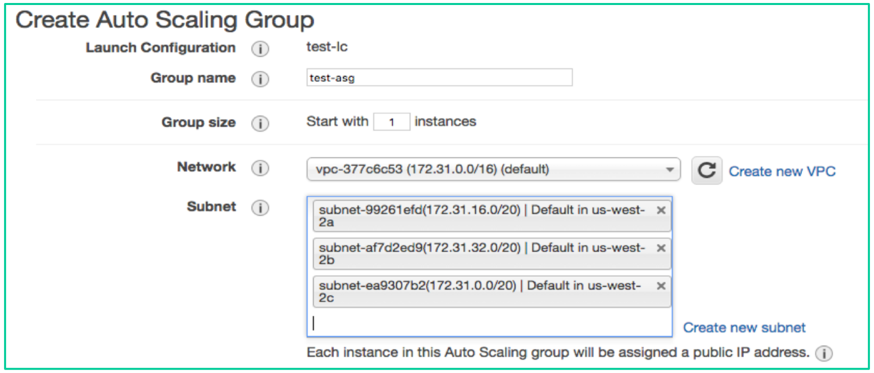
- Define Scaling Policies



search : ami-09c6d9428d48366a0

| AMI Name | AMI ID | Source | Owner | Visibility | Status |
|---|---|---|---|---|---|
| My apache server | ami-09c6d9428d48366a0 | 041659675190/… | 041659675190 | Private | available |

Image: ami-09c6d9428d48366a0



Create Auto Scaling Group

Launch Configuration: test-lc

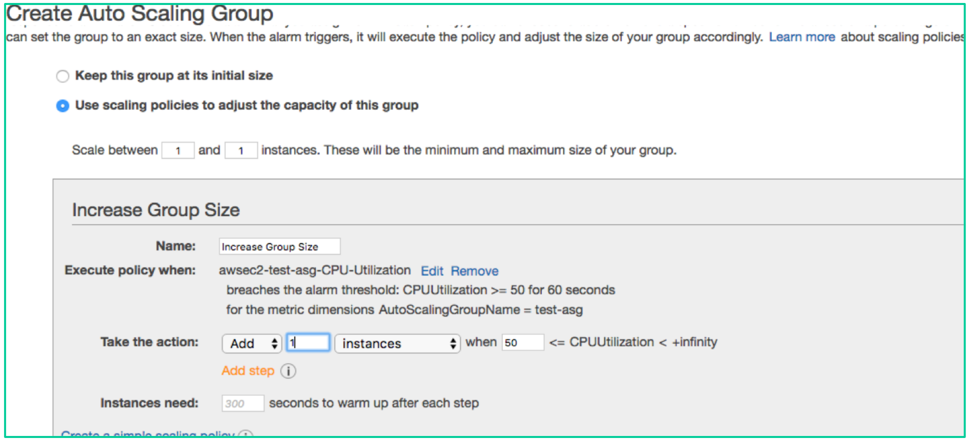
Group name: test-asg

Group size: Start with 1 instances

Network: vpc-377c6c53 (172.31.0.0/16) (default)

Subnet:
subnet-99261efd(172.31.16.0/20) | Default in us-west-2a
subnet-af7d2ed9(172.31.32.0/20) | Default in us-west-2b
subnet-ea9307b2(172.31.0.0/20) | Default in us-west-2c

Each instance in this Auto Scaling group will be assigned a public IP address.



Create Auto Scaling Group

can set the group to an exact size. When the alarm triggers, it will execute the policy and adjust the size of your group accordingly. Learn more about scaling policies

○ Keep this group at its initial size

● Use scaling policies to adjust the capacity of this group

Scale between 1 and 1 instances. These will be the minimum and maximum size of your group.

Increase Group Size

Name: Increase Group Size

Execute policy when: awsec2-test-asg-CPU-Utilization
breaches the alarm threshold: CPUUtilization >= 50 for 60 seconds
for the metric dimensions AutoScalingGroupName = test-asg

Take the action: Add 1 instances when 50 <= CPUUtilization < +infinity

Instances need: 300 seconds to warm up after each step