AWS re:Invent

CMP201

All You Need to Know About Auto Scaling

Michael Hanisch, AWS Solutions Architecture

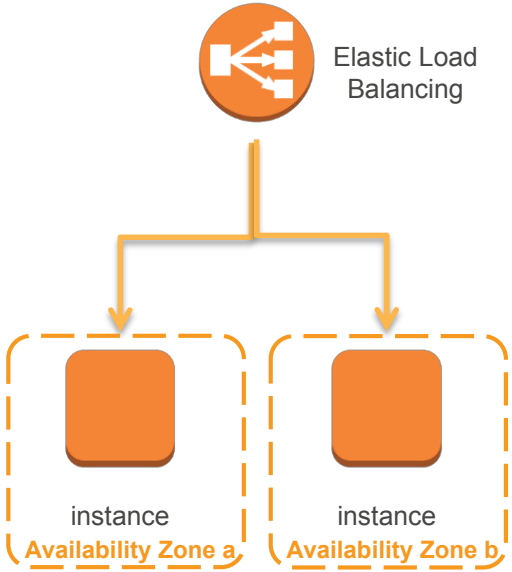
October 2015

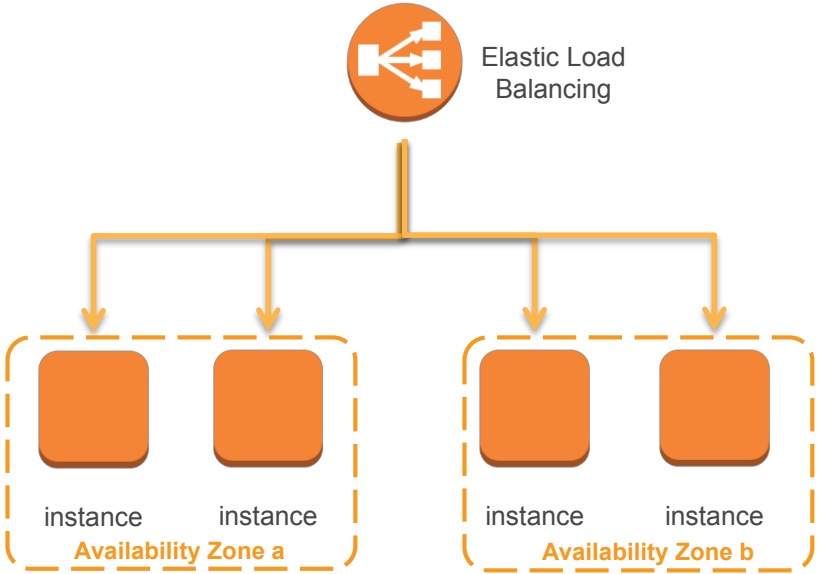
amazon web services

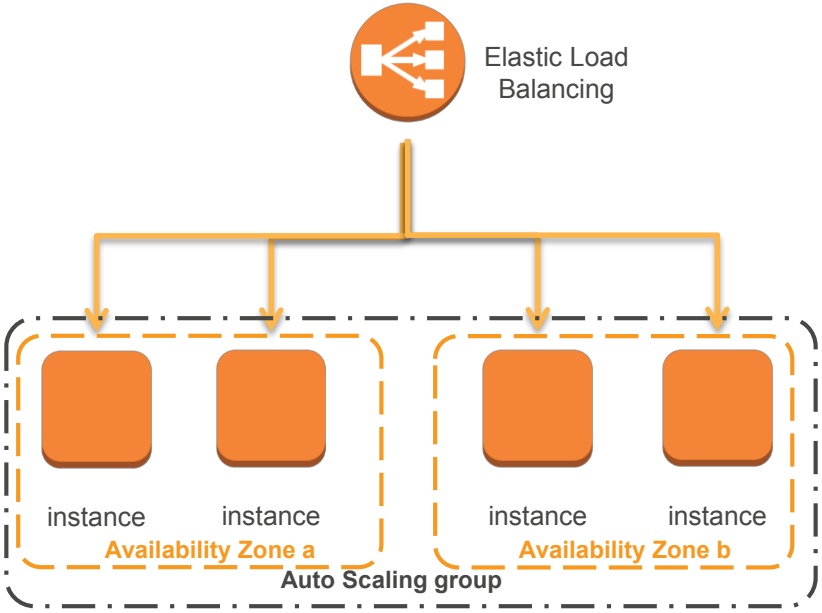
What to Expect from the Session

- Basic introduction to Auto Scaling
- Using Scaling Policies
- Controlling launch and termination of EC2 instances

Auto Scaling – The Basics


Elastic Load Balancing
instance
Availability Zone a
instance
Availability Zone b


Elastic Load Balancing
instance
instance
Availability Zone a
instance
instance
Availability Zone b



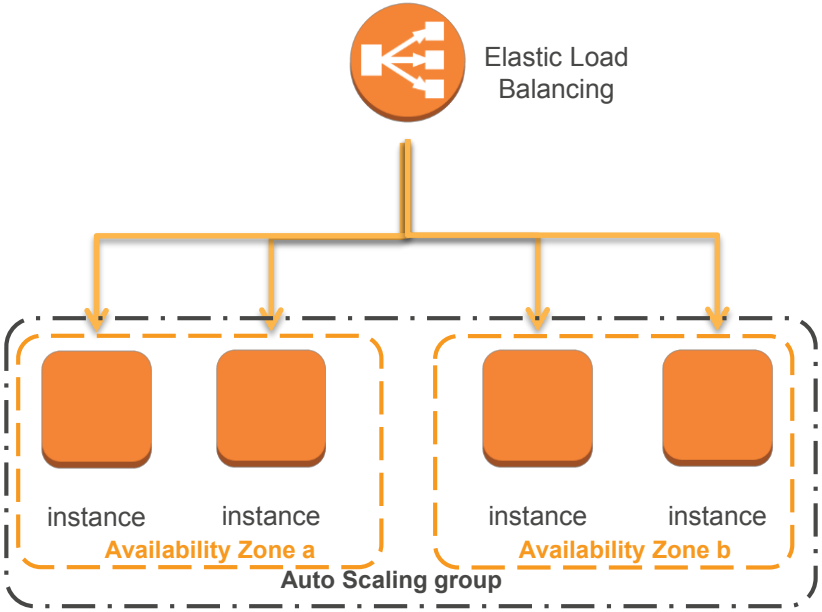
Minimum = 2

Maximum = 10

Desired # of instances = 4

Auto Scaling Groups

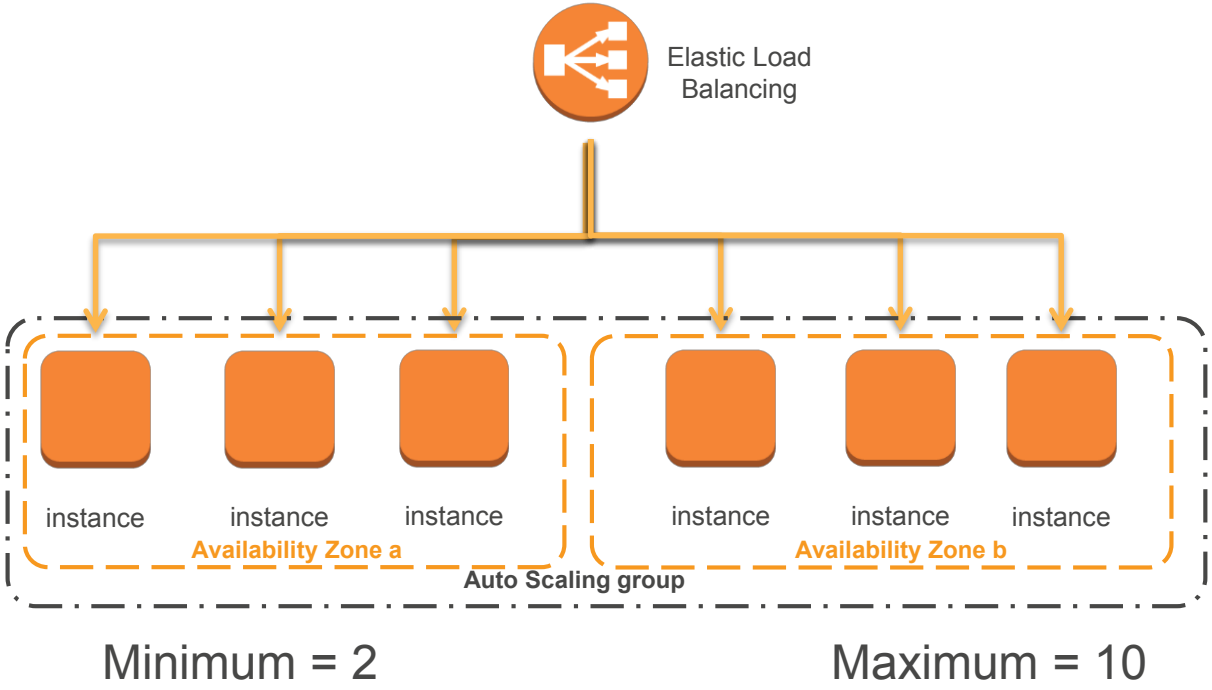
- Always keep **minimum** number of instances running
- Launch or terminate instances to meet **desired capacity**
- Never start more than **maximum** number of instances
- Keeps capacity **balanced** across AZs



Minimum = 2

Maximum = 10

Desired # of instances = 6


Elastic Load Balancing
instance
instance
instance
Availability Zone a
instance
instance
instance
Availability Zone b
Auto Scaling group
Minimum = 2
Maximum = 10
Desired # of instances = 6

Launch Configurations

Determine **what** is going to be launched:

- EC2 instance type & size
- Amazon Machine Image (AMI)
- Security groups, SSH keys, IAM instance profile
- User data

…

Bootstrapping

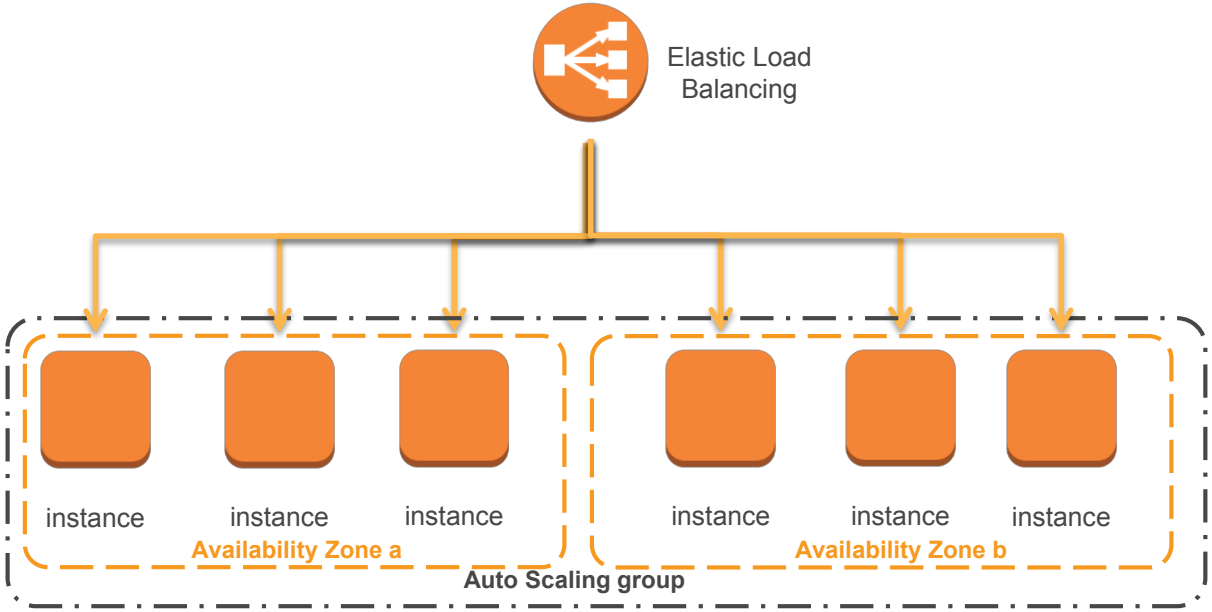
Installation & setup needs to be fully automated:

- Use Amazon Machine Image (AMI) with all required configuration & software (“golden image”)
- Base AMI + install code & configuration as needed
    - Via Userdata
    - Via Chef/Puppet/Ansible/…
    - Using AWS CodeDeploy

…

Bootstrapping

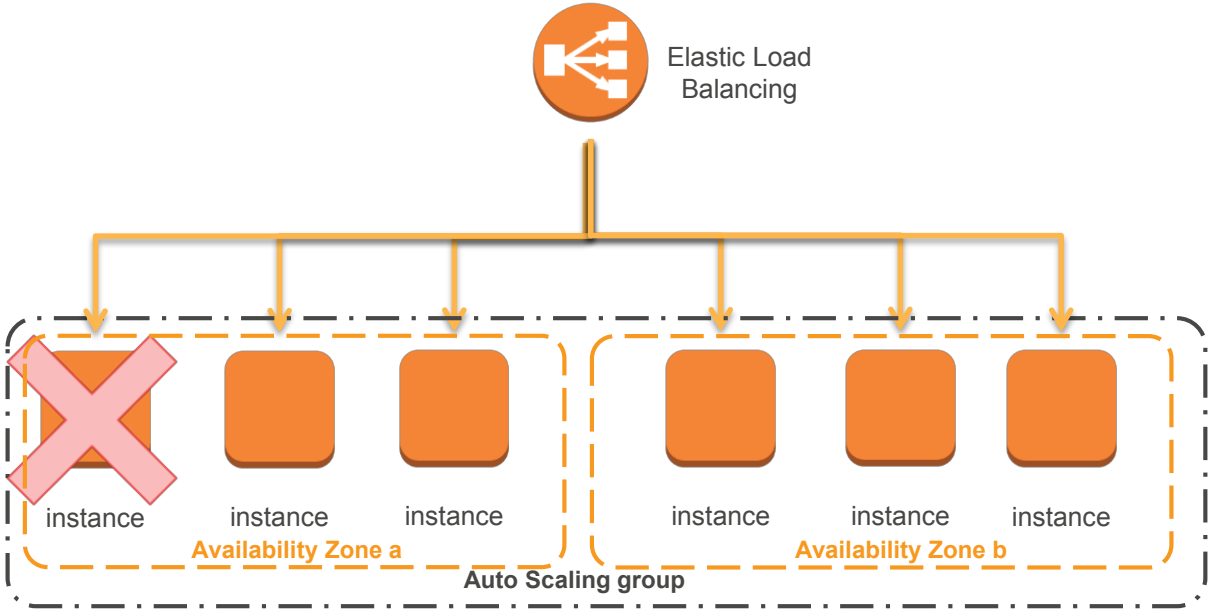
```
#!/bin/bash
yum -y update;
yum install -y ruby; yum install -y aws-cli;
cd /home/ec2-user;
aws s3 cp s3://aws-codedeploy-us-east-1/latest/ \
install . --region us-east-1;
chmod +x ./install;
./install auto;
```



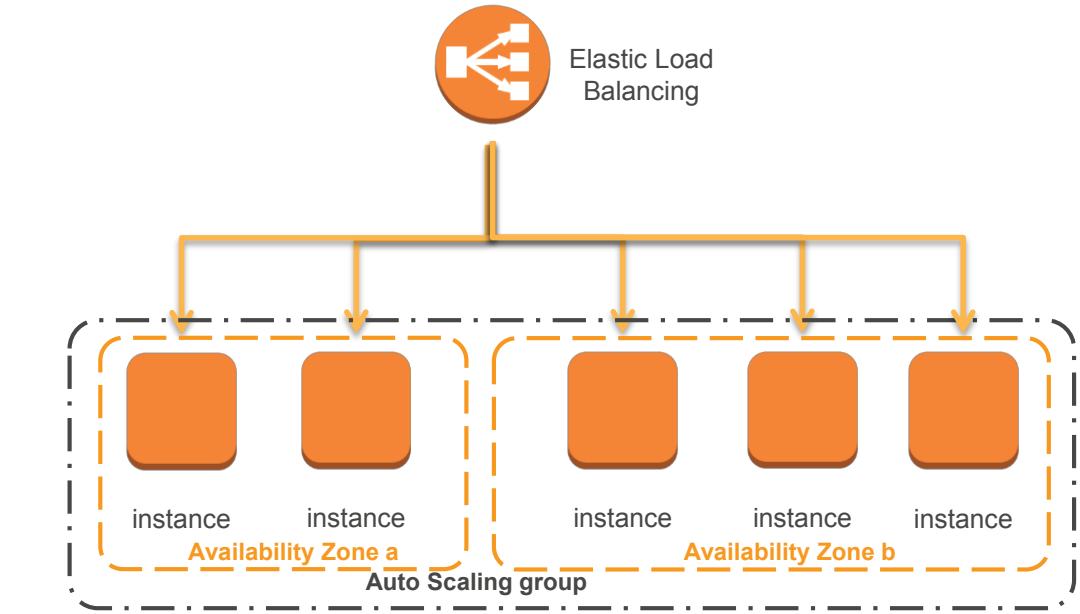
Minimum = 2

Maximum = 10

Desired # of instances = 6



Desired # of instances = 5



Minimum = 2

Maximum = 10

Desired # of instances **= 5**

Termination Policies

Determine which instances are terminated first:

- Longest running
- Oldest launch configuration
- Closest to full billing hour

But: rebalancing of capacity across AZs takes precedence!

Scaling Plans

Determine when the Auto Scaling group will scale in or out:

desired capacity > current capacity: launch instances

desired capacity < current capacity: terminate instances

Scaling Plans

- Default: ensure current capacity of **healthy** instances remains within boundaries (never less than minimum)
- ‘Manual scaling’: modify desired capacity (via API, console, CLI) to trigger a scaling event
- Scheduled: scale in / out based on timed events
- Dynamic scaling: scale on Amazon CloudWatch metrics

Auto Scaling Concepts

Auto Scaling Groups

- EC2 instances are managed by Auto Scaling *groups*.
- Create Auto Scaling groups by defining the minimum, maximum, and, optionally, the desired number of running EC2 instances.

Launch Configuration

- Auto Scaling groups use a *launch configuration* to launch EC2 instances.
- Provides information about the AMI and EC2 instance types/size

Scaling Plan

- A scaling plan tells Auto Scaling when and how to scale.
- Create a scaling plan based on the occurrence of specified conditions (dynamic scaling) or create a plan based on a specific schedule.

Integration with Amazon Elastic Load Balancing

Load Balance your Auto Scaling Group

Elastic Load Balancing

- Distribute incoming web traffic automatically.
- Single point of entry for your application.
- Sends data about your load balancers and EC2 instances to Amazon CloudWatch.
- Use Elastic Load Balancing metrics to scale your application.
- Use connection draining to wait for the in-flight requests to complete.

Load Balance your Auto Scaling Group

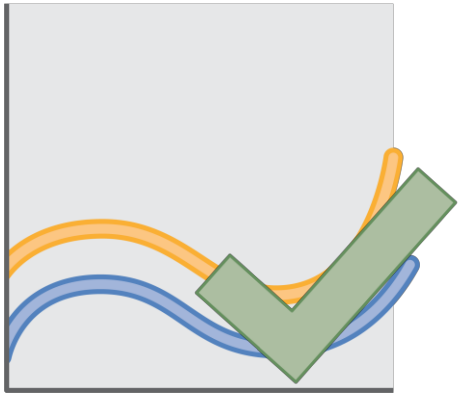
Elastic Load Balancing

- Configure your Auto Scaling Group to work with one or more Elastic Load Balancers
- Automatically registers new instances, deregisters on termination
- Use ELB health checks in your Auto Scaling Group
- Use Elastic Load Balancing metrics in scaling policies

Benefits of Auto Scaling

Elasticity:

Automatically adapt capacity to demand



Availability & Reliability

Health Checks

- Performed periodically
- Instances are marked as unhealthy or healthy
- Unhealthy instances are terminated and replaced

  (if new number of instances < minimum or < desired capacity)

Health Checks

- EC2 instance status:

  Instance is unhealthy when instance state `!=` ‘`running`’

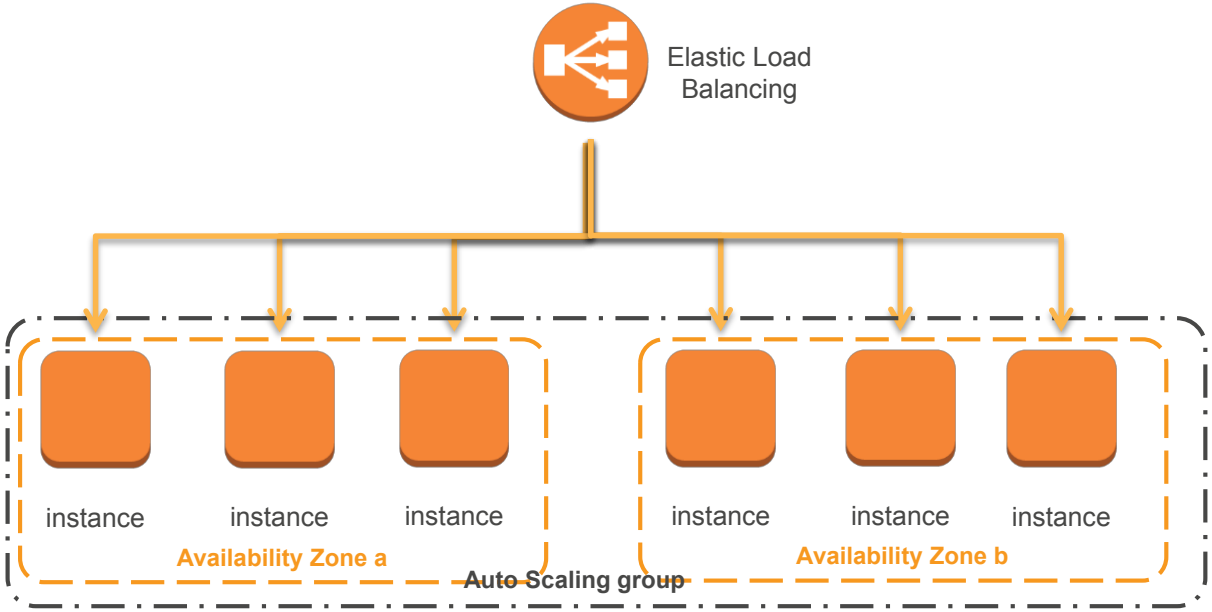
  or system health check == ‘`impaired`’

- ELB health checks:

  instance is unhealthy when ELB health check results in “`OutOfService`” (**or** EC2 health check failed)

- Manual: mark individual instances as ‘unhealthy’

  Instance unhealthy when marked as such **or** EC2 health check failed. Use to integrate with external monitoring systems.

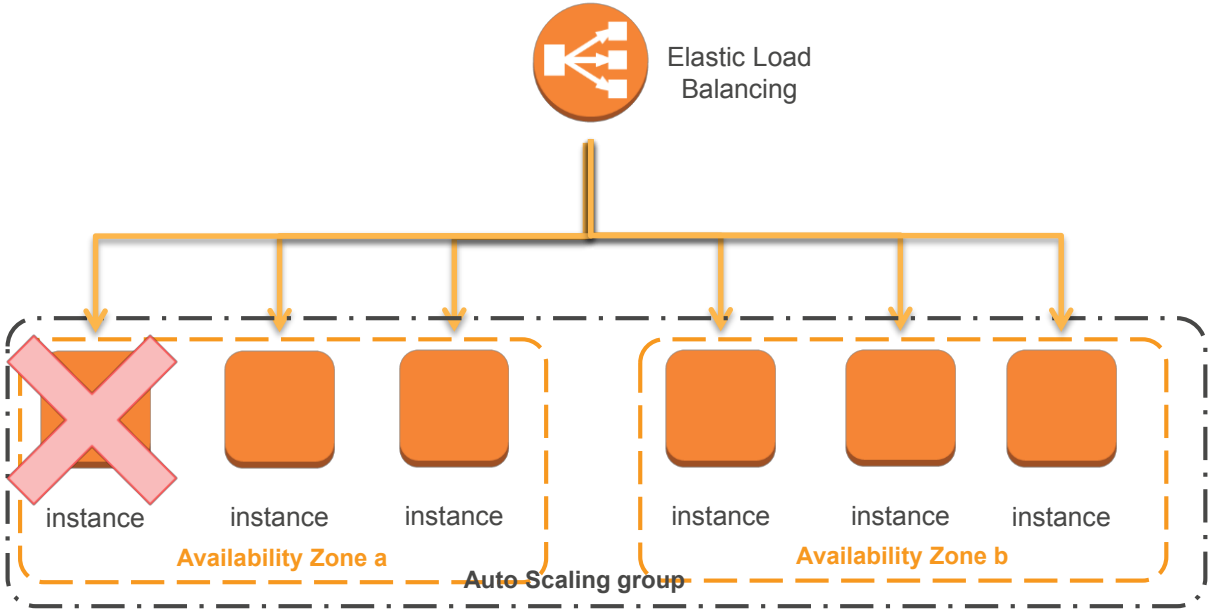


Minimum = 2

Maximum = 10

Desired # of instances = 6

Unhealthy Instances Get Replaced…

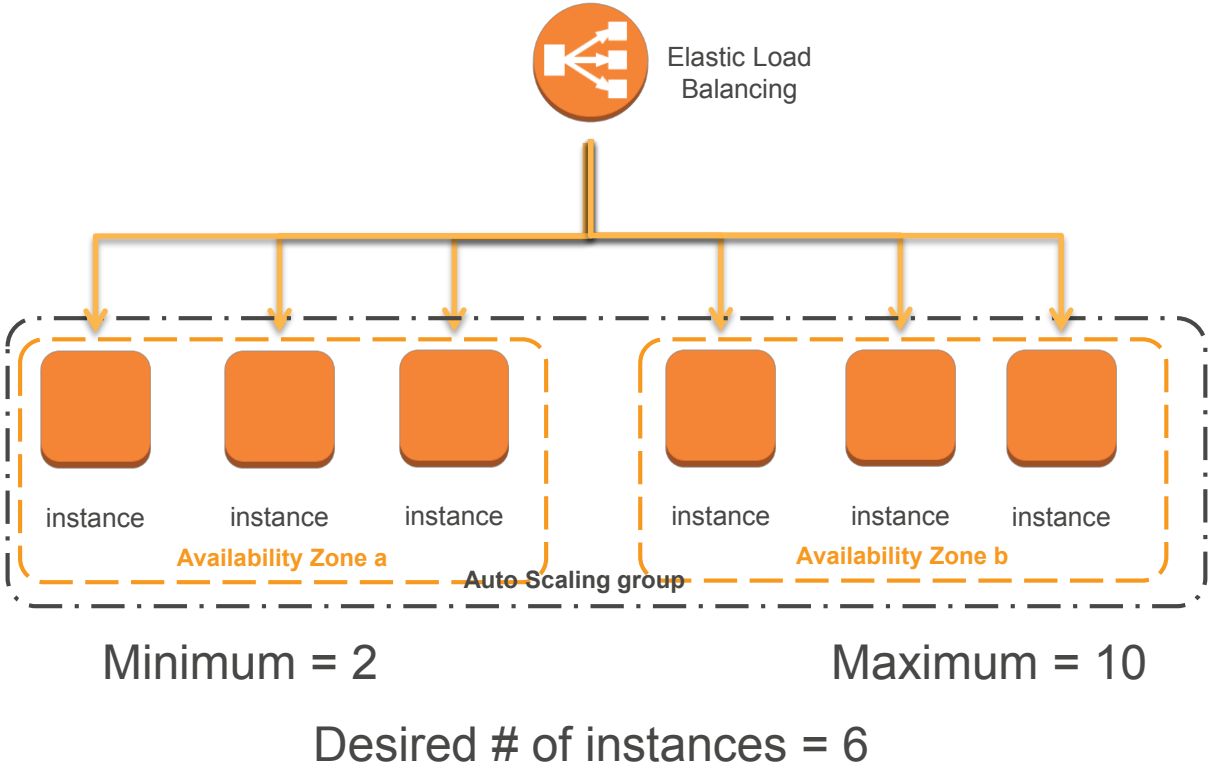


Minimum = 2

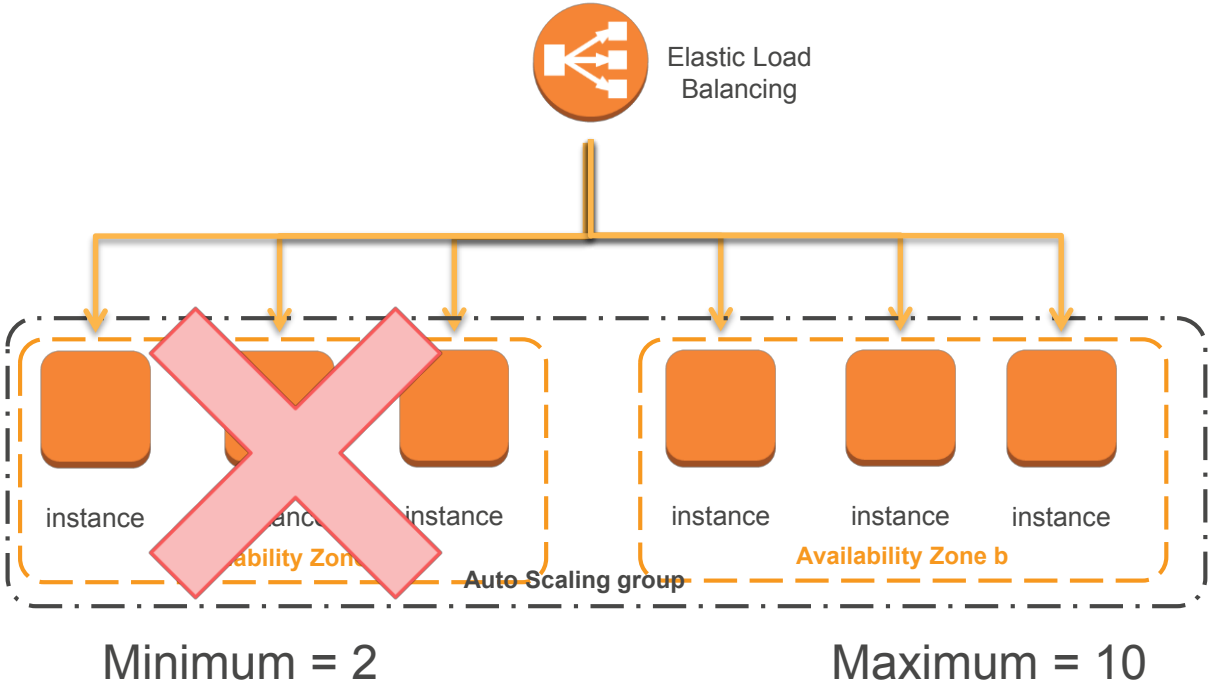
Maximum = 10

Desired # of instances = 6

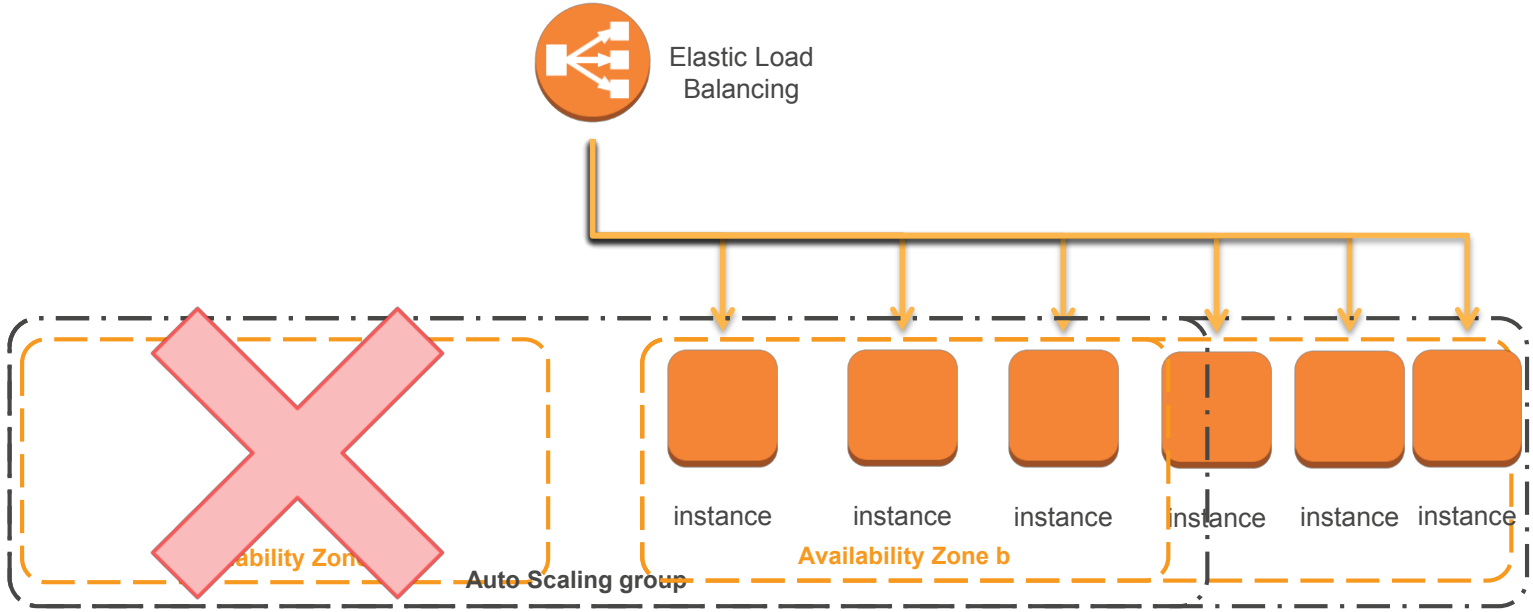
Unhealthy Instances Get Replaced…



Unhealthy Instances Get Replaced…



…In a Different AZ if Necessary



Minimum = 2

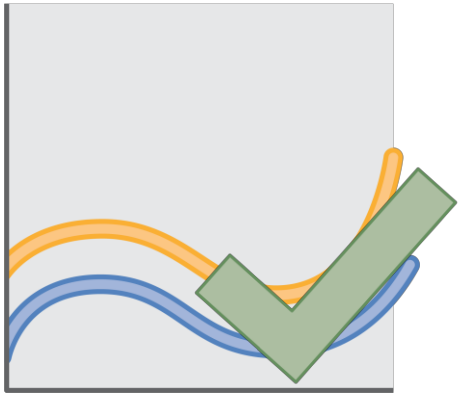
Maximum = 10

Desired # of instances = 6

Benefits of Auto Scaling

Elastic:

Automatically adapt capacity to demand



Reliable:

Counteract failures of instances or AZs

Scaling Policies

Scaling Policies

Can change capacity of the group in 3 different ways:

- Set fixed capacity, e.g., desired capacity = 4 instances
- Add / remove fixed number of instances, e.g., + 2
- Add / remove percentage of existing capacity, e.g. +20%

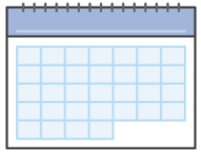
Scaling on a Schedule

Scaling on a Schedule

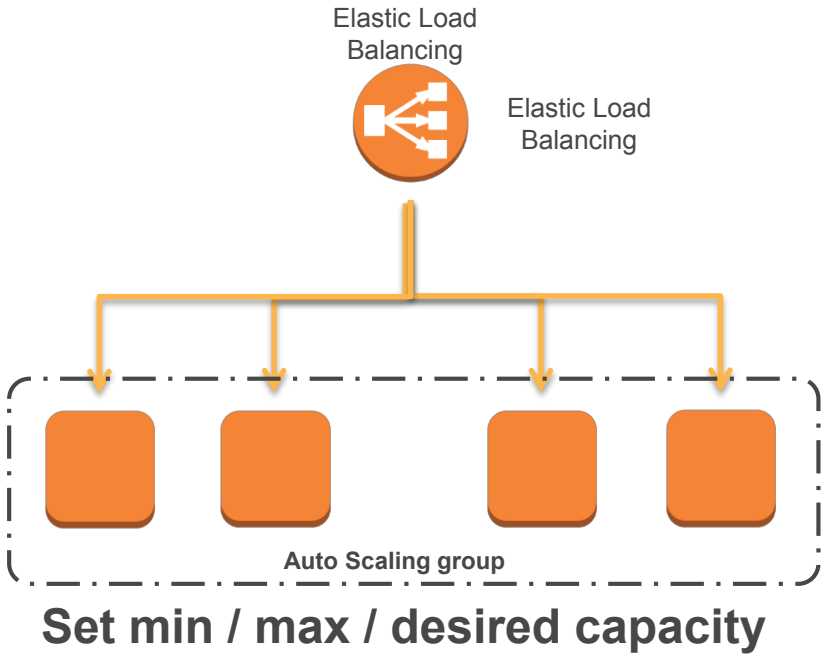
cron-like syntax for recurring scaling events

Scaling event

Schedule individual events (up to 135 events per group)



Scaling event



Auto Scaling CLI & SDK only!
Not available in AWS management console

Dynamic Scaling Policies

Dynamic Scaling Policies

Trigger scaling events based on demand:

- Demand is measured based on metrics
- Changes in metrics can be mapped to scaling policies

Amazon CloudWatch

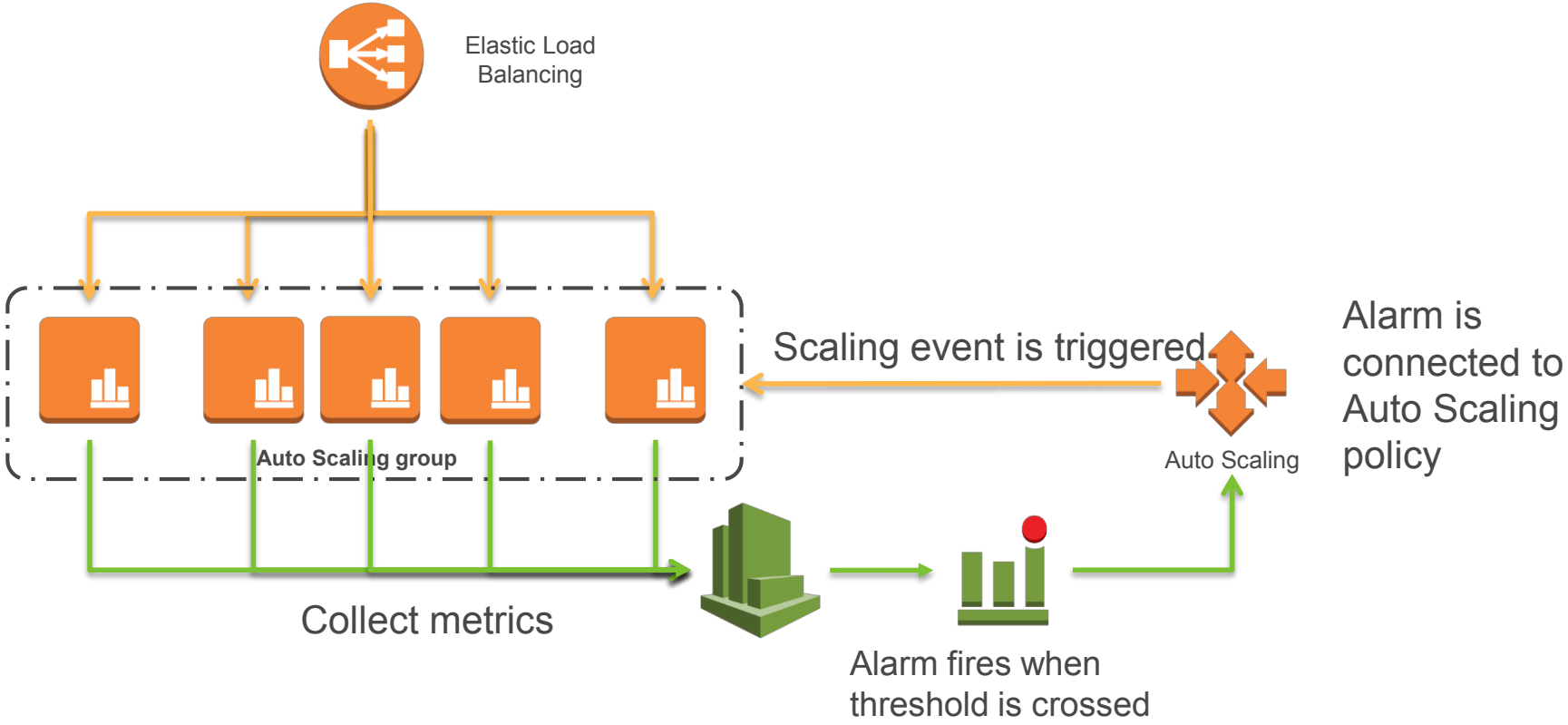
Amazon **CloudWatch**: A web service that enables you to monitor and manage various metrics, and configure alarm actions based on data from those metrics.

A **metric** is a variable that you want to monitor, e.g., CPU usage or incoming network traffic.

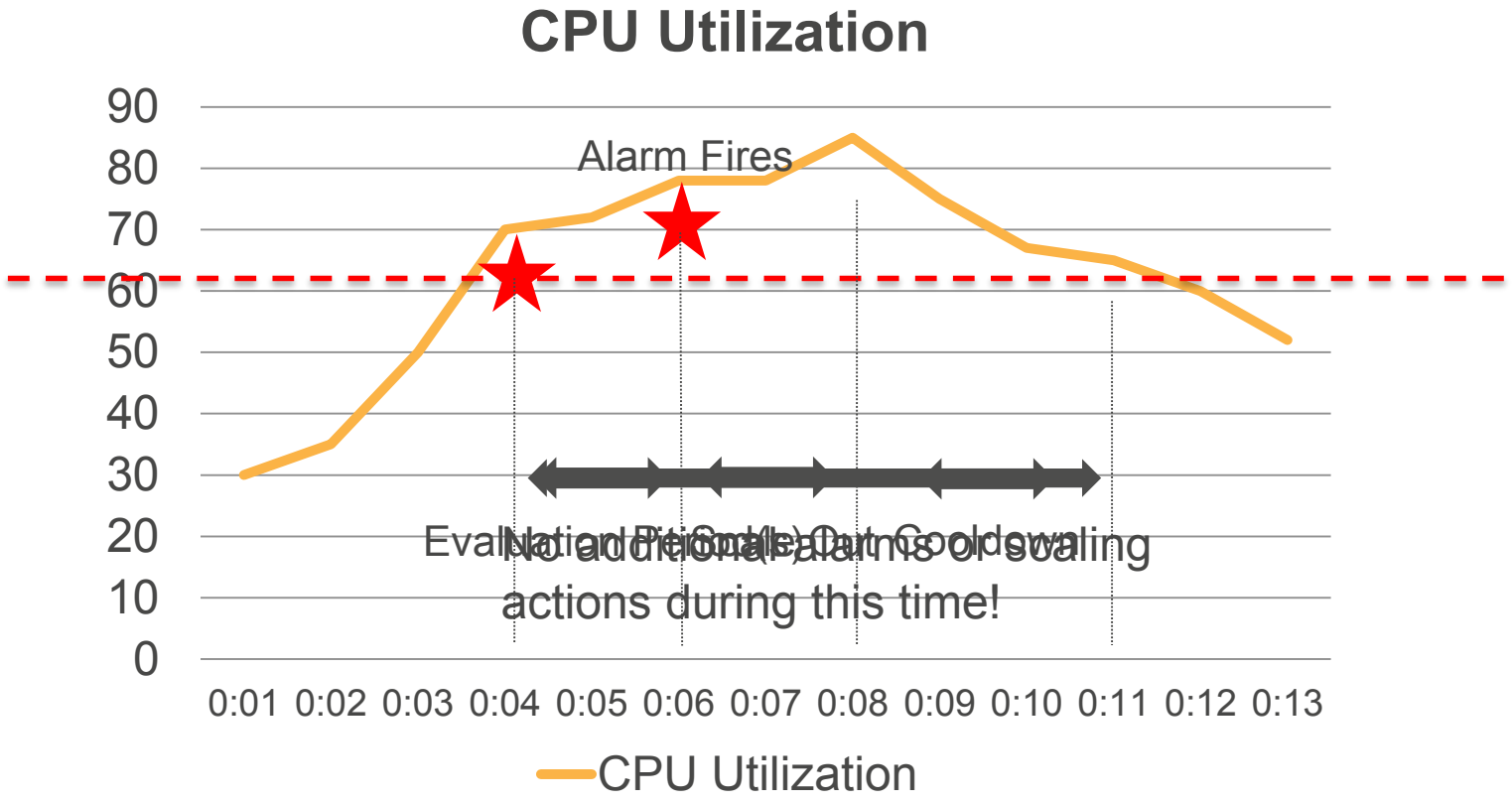
A **CloudWatch *alarm*** is an object that monitors a single metric over a specific period.

The alarm changes its **state** when the value of the metric breaches a defined range and maintains the change for a specified number of periods.

How Alarms Work



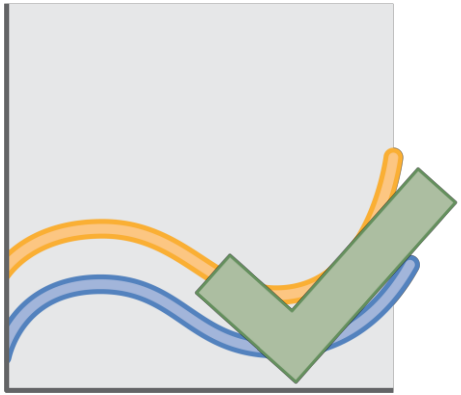
Amazon CloudWatch can aggregate metrics across pre-defined dimensions, e.g., aggregate average CPU utilization of all EC2 instances in an Auto Scaling group.


CPU Utilization
90
80
70
60
50
40
30
20
10
0
Alarm Fires
Evaluation Period
No additional alarms or scaling actions during this time!
0:01 0:02 0:03 0:04 0:05 0:06 0:07 0:08 0:09 0:10 0:11 0:12 0:13
CPU Utilization

Benefits of Auto Scaling

Elastic:

Automatically adapt capacity to demand

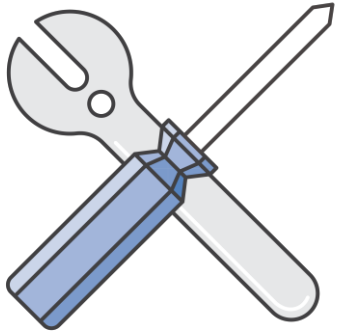


Reliable:

Counteract failures of instances or AZs

Customizable:

With bootstrapping & lifecycle hooks



Thank you!