



# Elastic Cloud Services for Distributed Deployment



[www.huaweicloud.com](http://www.huaweicloud.com)



# Foreword

- In the course of business development, service systems can become overloaded, slow down, or even face interruptions due to demand fluctuations and market changes. Building scalable systems helps enhance service stability, improve response speed, and optimize resource utilization.
- This chapter describes what scalability is and how to build scalable systems on Huawei Cloud.



# Objectives

- Upon completion of this lesson, you will:
  - Learn the advantages of load balancing on the cloud.
  - Learn how to schedule distributed applications.
  - Learn how to adjust resources for distributed applications.

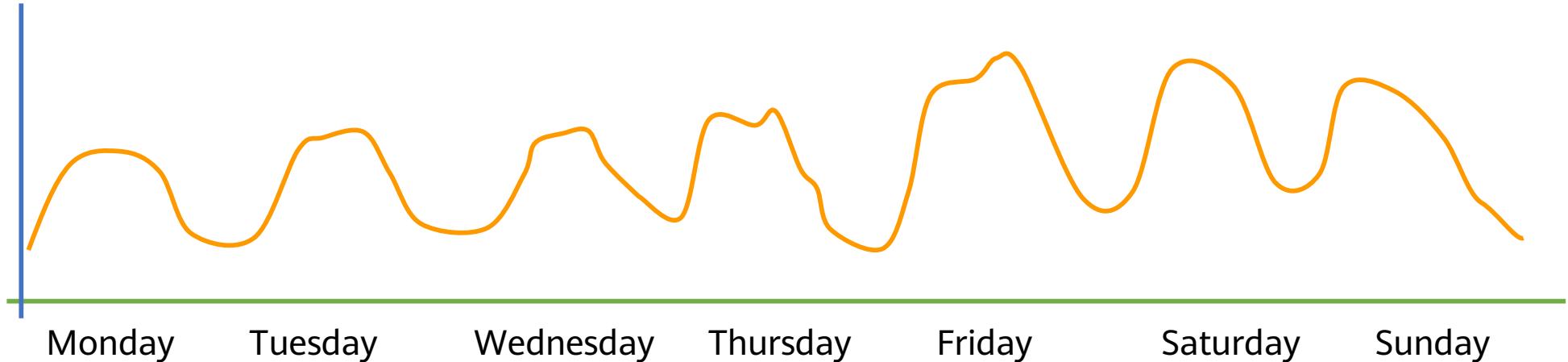


# Contents

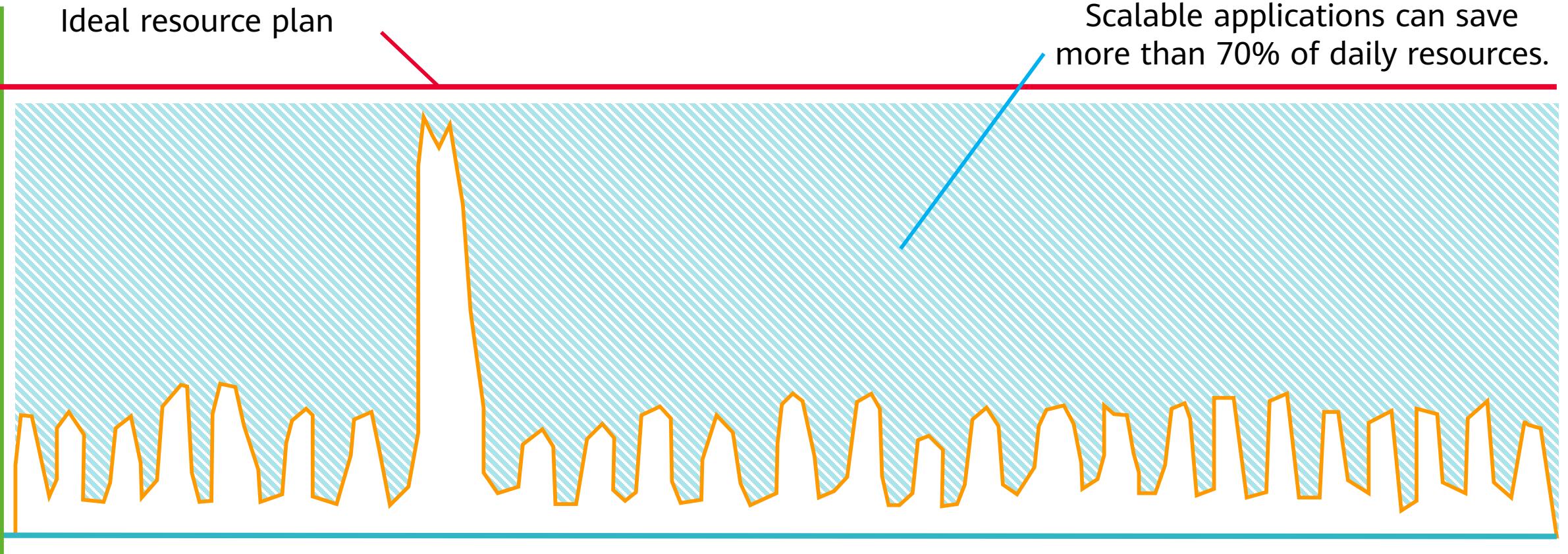
- 1. Scalability Implementation**
2. How to Build Scalable Applications
3. Typical Scalable Website Architecture

# Why Is Scalability Important?

The amount of traffic an application has to handle may fluctuate.  
Resources should be adjusted as traffic changes.

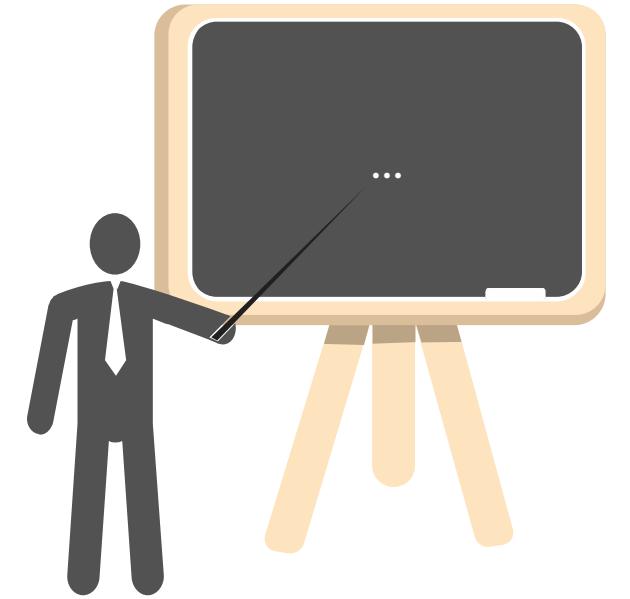


# Traffic Surges During Double 11 or Other Big Promotions

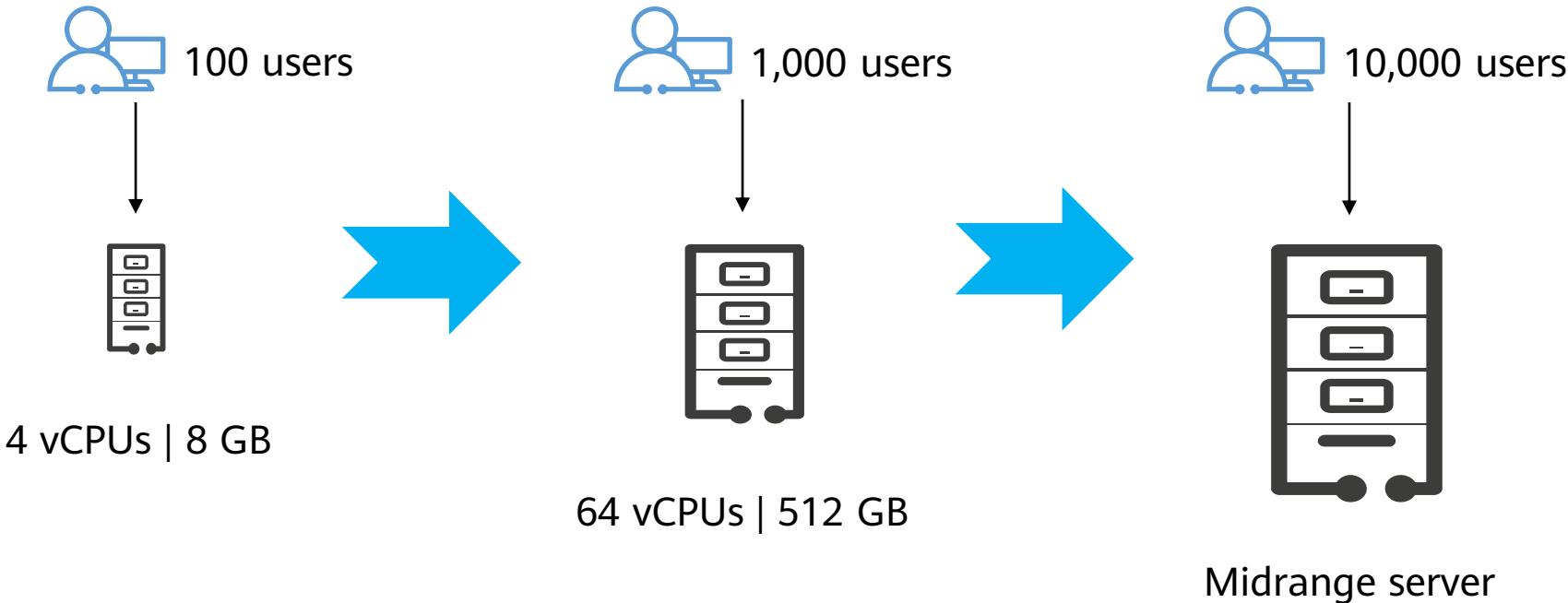


**It is not just a waste of resources.**

# How Can We Manage Sudden Traffic Changes?



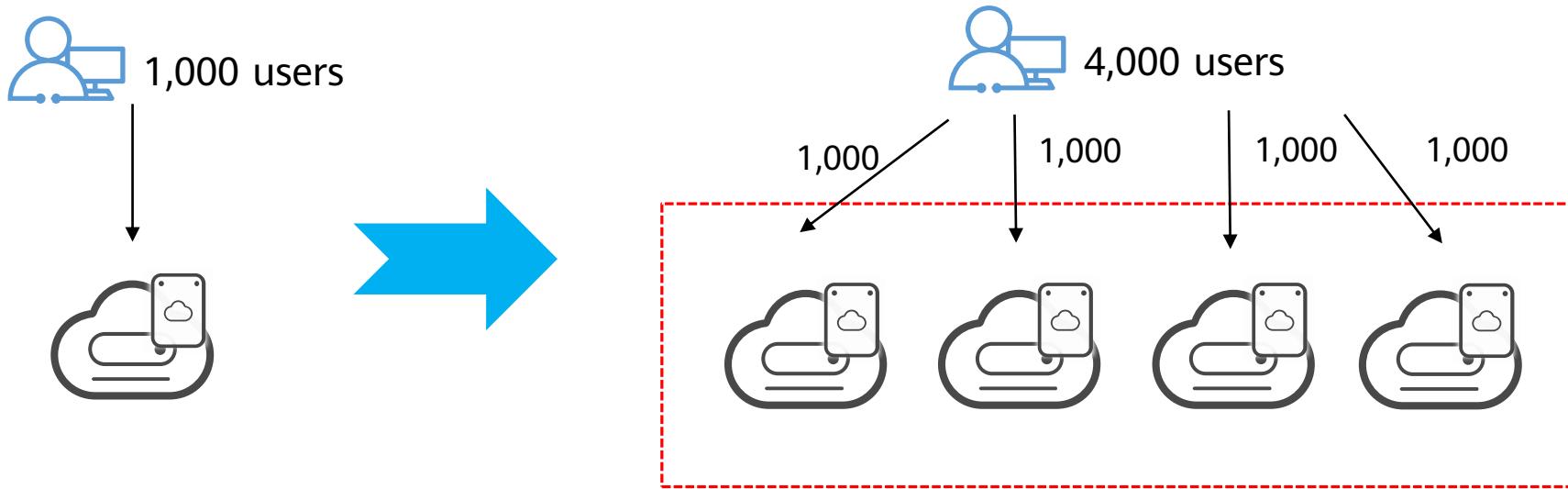
# Method 1: Scaling Up



Advantages: simple and independent

Disadvantages: **more bottlenecks**

# Method 2: Scaling Out



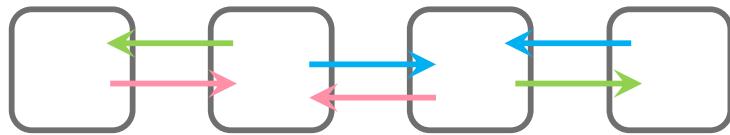
Advantages: You can use simple devices for unlimited scalability and higher availability.

Disadvantages: Technology coordination is required, and this method only works for stateless applications

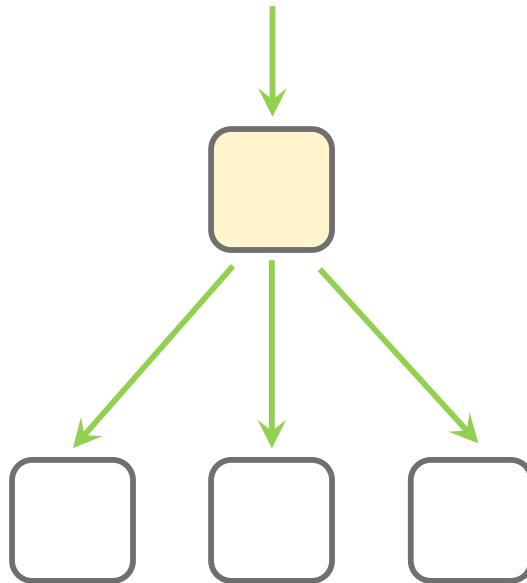


Distributed deployment seems better.

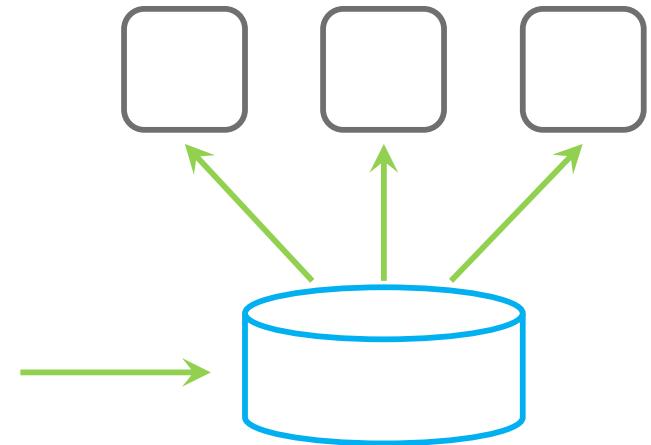
# Typical Distributed Models



Synchronous communication  
and coordination

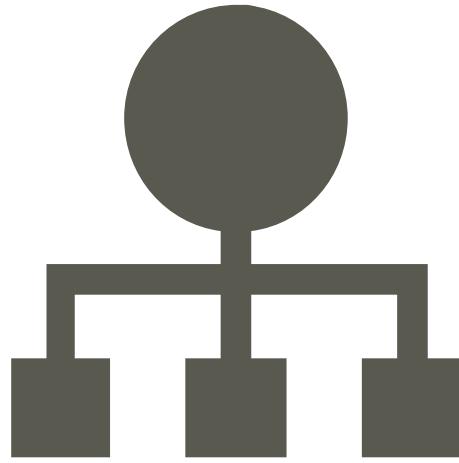


Request distribution

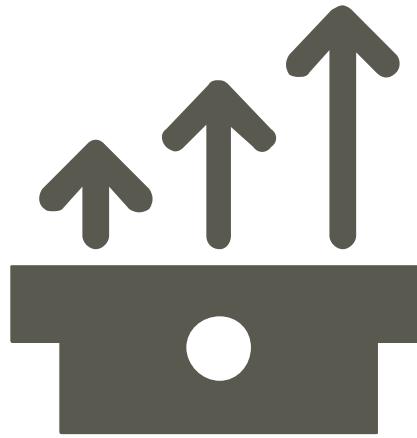


Task polling

# Benefits of Distributed Systems



**Huge resource pool for required performance**

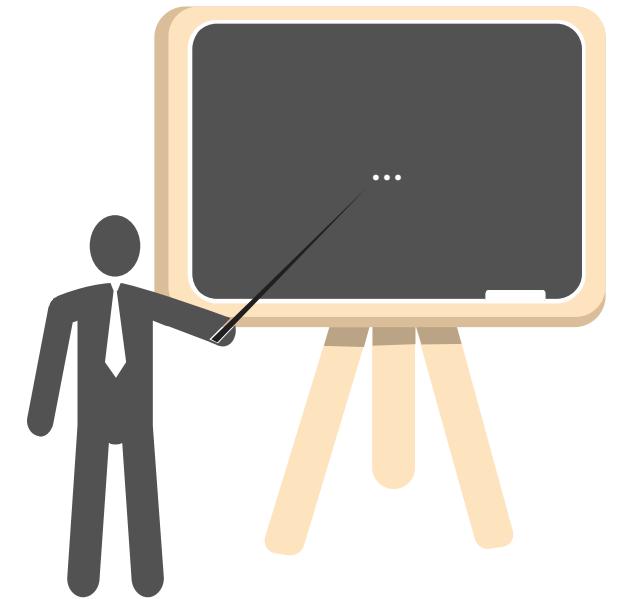


**Robust load balancing for stable performance**

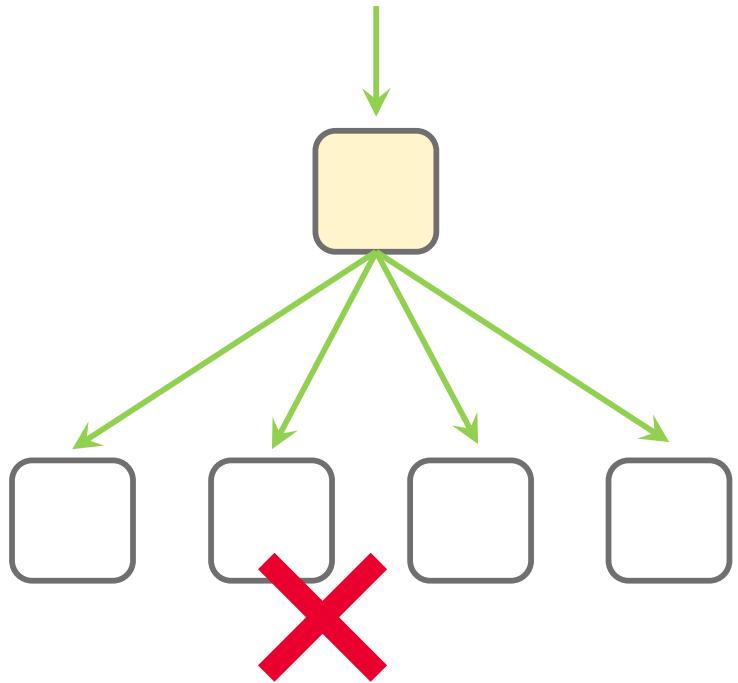


**Better fault recovery**

# What Else Can We Get from Scalable Systems?

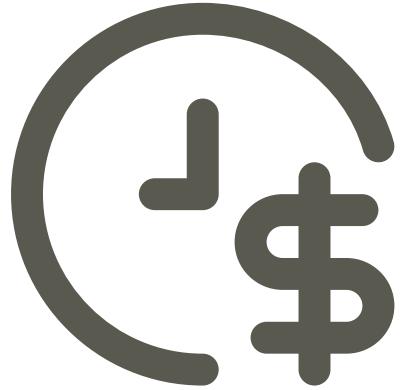


# A Good Scalability Design Means Improved Reliability



- If a node is unhealthy, the overall processing capacity is reduced.
- If this happens, the system will automatically add a healthy backend server to ensure stable performance.

# The Value of Scalability



**Adjust resources as demand changes to reduce costs.**



**Replace faulty nodes with healthy ones for higher reliability.**



# Contents

1. Scalability Implementation
2. **How to Build Scalable Applications**
3. Typical Scalable Website Architecture

# What Are Needed to Enable Scalability?

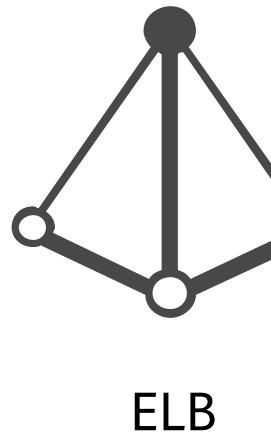
**Step 1** Distribute traffic to multiple servers.

**Step 2** Get to know when it is the time to scale resources.

**Step 3** Scale resources immediately as requirements change.

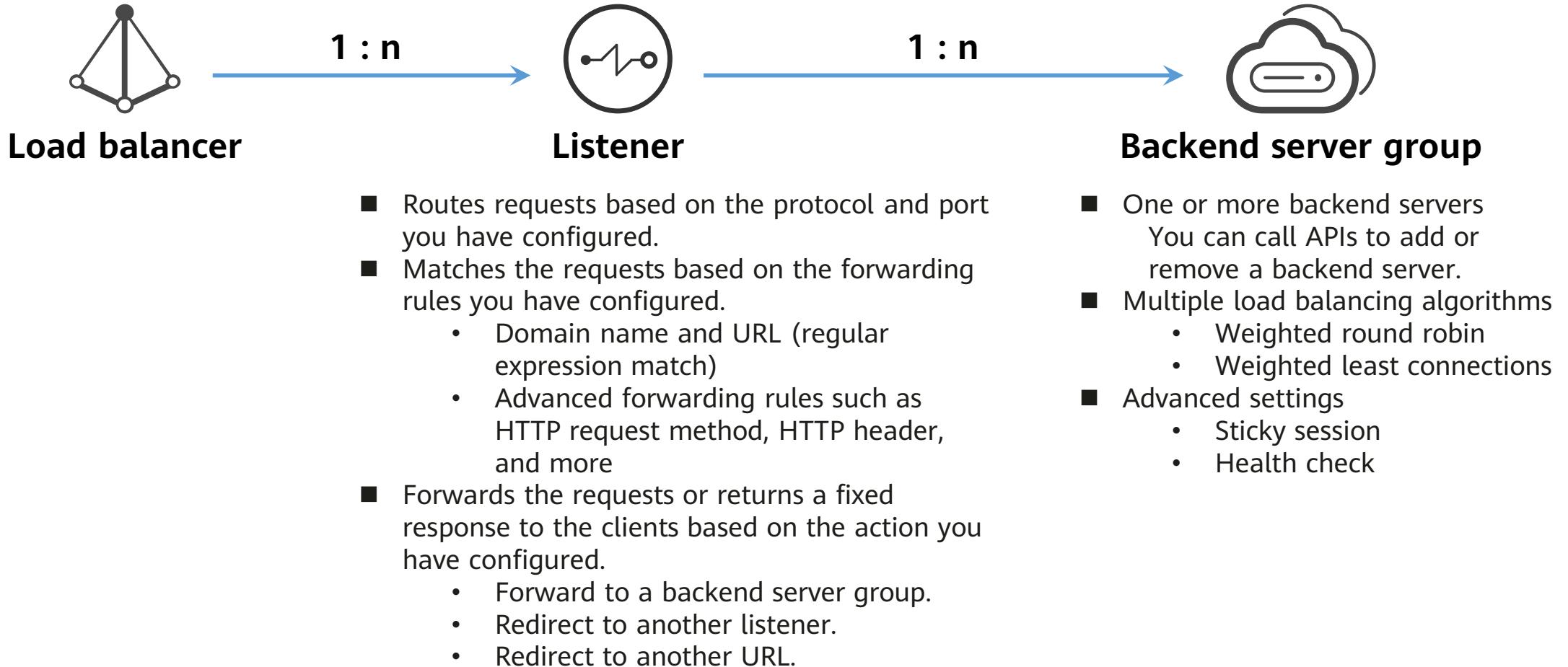
# Elastic Load Balance (ELB)

- ELB automatically distributes incoming traffic across multiple backend servers based on the listening rules you configure. It expands the service capabilities of your applications and improves their availability by eliminating single points of failure (SPOFs).

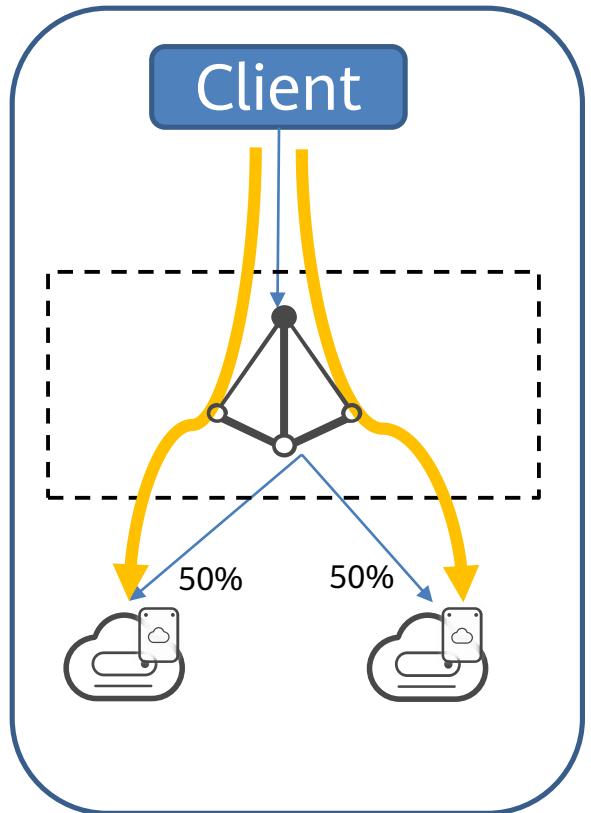


- Distributes requests across different backend servers.
- Supports built-in HA.
- Works with Auto Scaling to process a massive number of concurrent requests.
- Routes traffic across backend server groups based on the forwarding policies you have configured.
- Checks the health of backend servers to ensure that requests are routed to healthy servers.
- Supports load balancing at both Layer 4 and Layer 7.

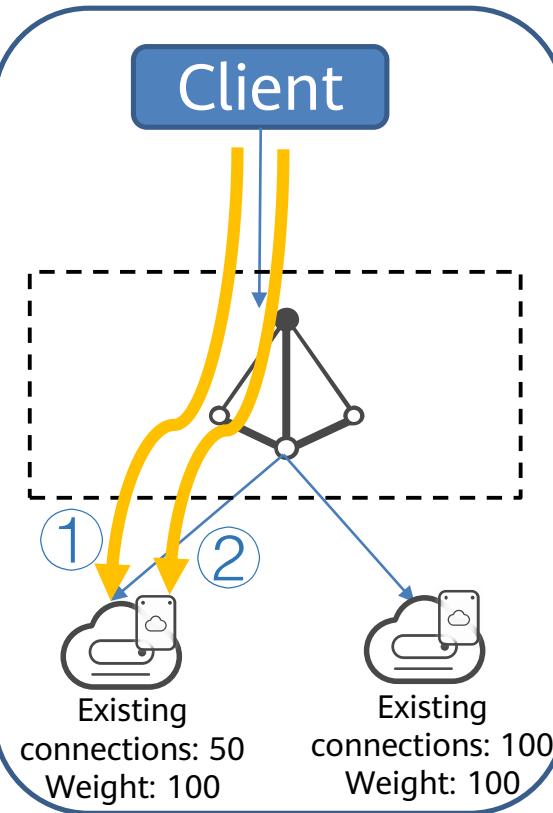
# ELB Components



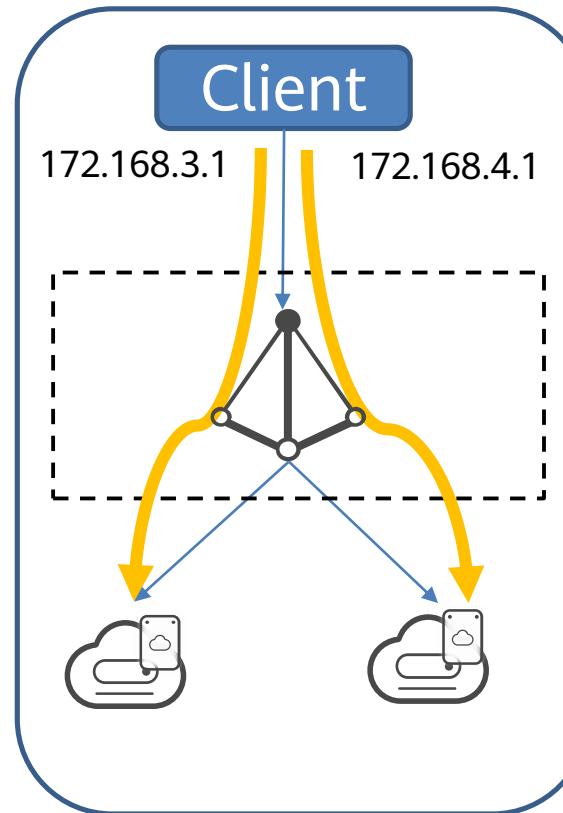
# Load Balancing Algorithms



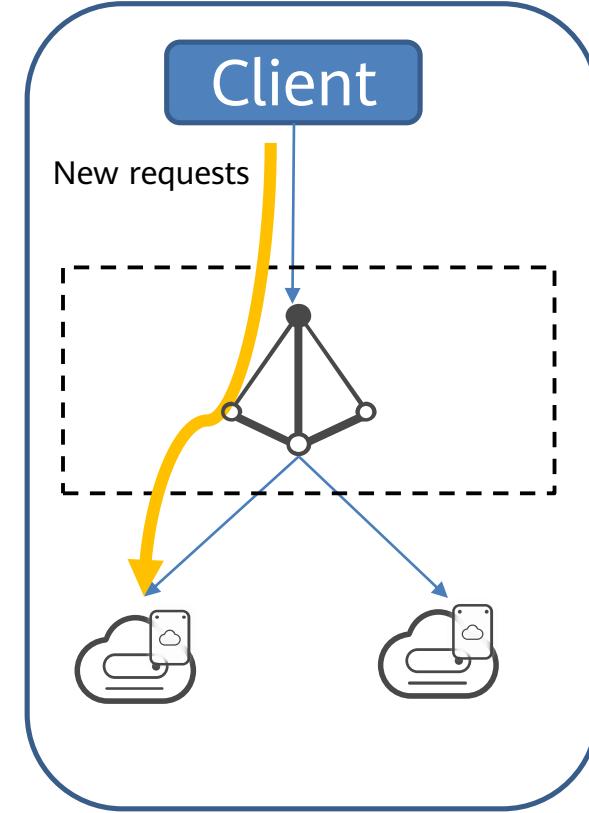
Weighted round robin



Weighted least connections

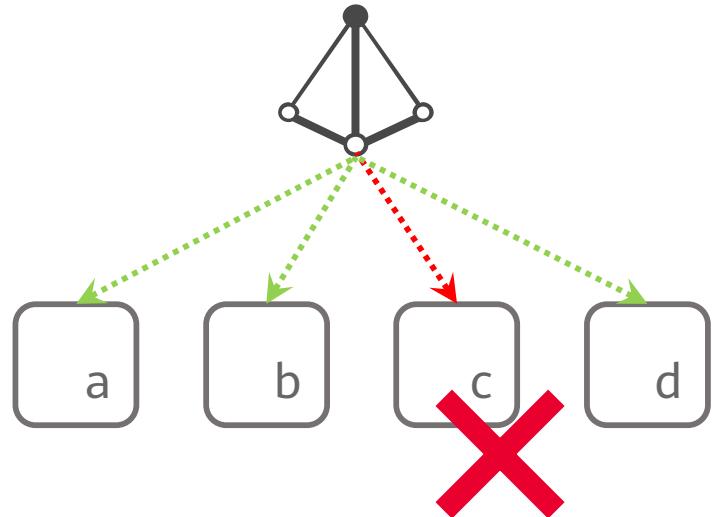


Source IP hash



Connection ID

# Health Check



- Configure a health check to avoid routing requests to unhealthy backend servers.
- How does a health check work?
  - ELB connects to a specific TCP port used by the backend server.  
A connection is established with the port.
  - ELB accesses a specific HTTP page, generally the root directory.  
A 2xx code is returned.
- You should select a dedicated page for health checks.

# ELB Configuration Process

## 1. Creating a Load Balancer

- Create a load balancer.
- Select the load balancer type.
- Configure the network parameters.

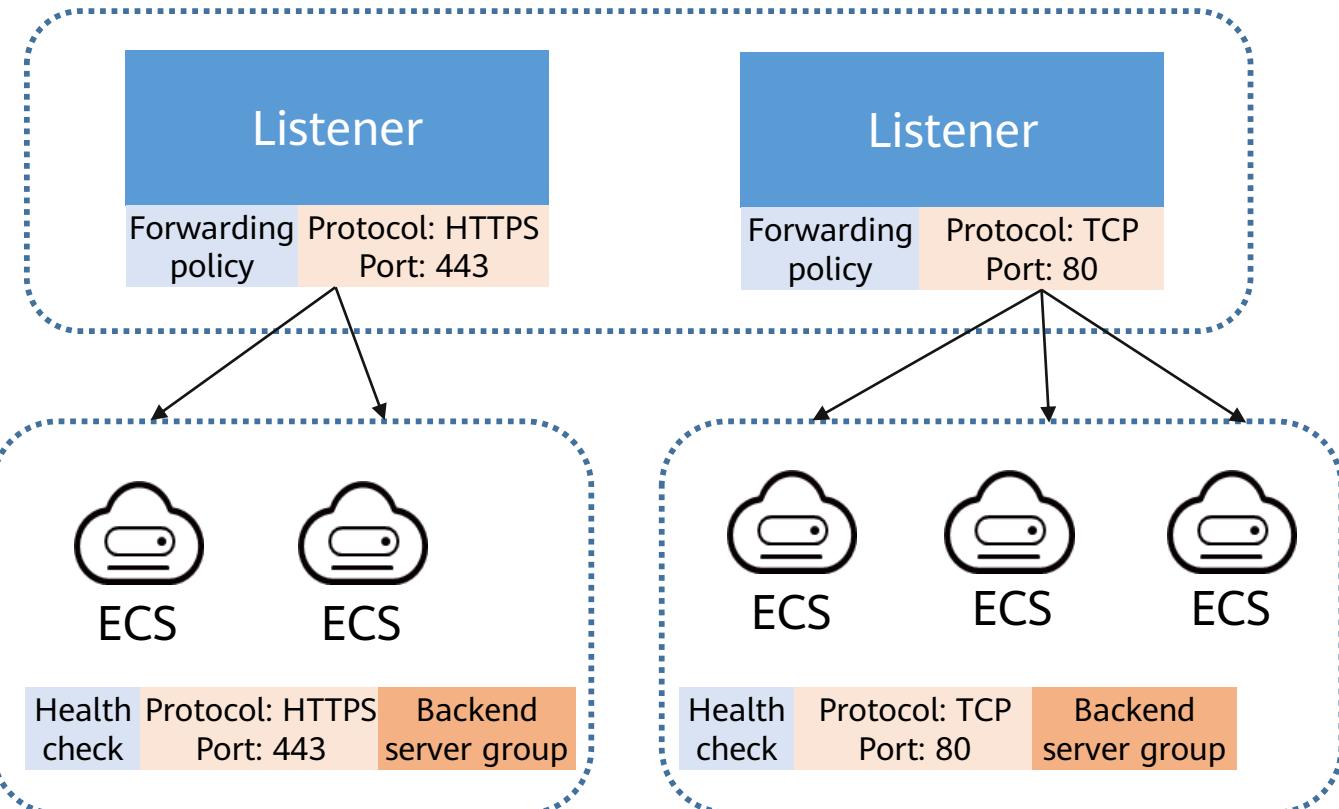
1

Load balancer

## 2. Adding a Listener

- Locate the created load balancer.
- Configure the protocol and port.

2



## 3. Adding a Backend Server Group

- Select a load balancing algorithm.
- Configure a health check.

3

# Creating a Load Balancer

**Basic Information**

Type

 **Dedicated load balancer**  
They work well for heavy-traffic and highly concurrent services, such as large websites, cloud-native applications, IoT, and multi-AZ disaster recovery applications.

 **Shared load balancer**  
They are good for services with low traffic, such as small websites and common HA applications.

The load balancer type cannot be changed after it is selected. View [Differences Between Dedicated and Shared Load Balancers](#) before selecting a type.

Billing Mode

Yearly/Monthly  Pay-per-use

Region

 CN East-Shanghai2

AZ

A...  A...

You can choose to deploy the load balancer in multiple AZs for higher availability.

Name

elb-2d2a

**Specification**

Specifications

Fixed  Elastic

For stable traffic

Application load balancing(HTTP/HTTPS) [?](#)

Small I | 2,000 new HTTP connections / 200 new HTTPS connections | 200,000 concurrent connections | 4,000 queries per second | [Compare specifications](#)

Network load balancing(TCP/UDP) [?](#)

Small I | 10,000 TCP / 10,000 UDP new connections | 500,000 TCP / 500,000 UDP concurrent connections | 50 Mbit/s of bandwidth | [Compare specifications](#)

**Network Configuration**

Network Type

Private IPv4 network

VPC

vpc-default [View VPCs](#) [Create VPC](#)

Once the load balancer is created, the VPC cannot be changed.

Frontend Subnet [?](#)

--Select-- [View Subnet](#) [Create Subnet](#)

Backend Subnet [?](#)

Subnet of the load balancer [View Subnet](#) [Create Subnet](#)

The load balancer requires a minimum of 12 IP addresses in the subnet.

Make sure that the security group and network ACL rules allow traffic from the backend subnet where the load balancer works to the backend servers.

[Learn how to configure a security group](#) [Configure Security Group Rule](#) [Configure Network ACL Rule](#)

# Adding a Listener

1 Configure Listener —— 2 Configure Routing Policy —— 3 Add Backend Server —— 4 Confirm

\* Name

Frontend Protocol The protocol used by the load balancer to receive requests from the clients. Select TCP, UDP for listeners at Layer 4, and select HTTP, HTTPS for listeners at Layer 7.

TCP or UDP listeners do not support access logging.

\* Frontend Port  Value range: 1 to 65535

Access Control

Transfer Client IP Address   If you enable this option, servers cannot be backend servers and clients at the same time.

Advanced Settings  Idle Timeout (s) 300 Description --

# Adding a Backend Server Group and Enabling Health Check

The screenshot shows a step-by-step configuration wizard:

- Configure Listener
- Configure Routing Policy
- Add Backend Server (highlighted)
- Confirm

**Backend Servers** (Step 3):

- Add Backend Server** button.
- Table headers: Backend Servers, Private IP Address, Backend Port.
- No data available.

**Health Check** (Step 3):

- Health Check** toggle switch is turned on.
- Description: "Health check detects the running of backend servers and ensures that requests are routed only to healthy backend servers."
- Advanced Settings** table:

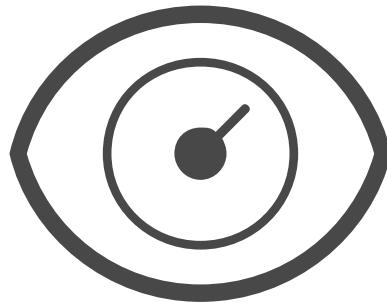
Setting	Value
Health Check Protocol	TCP
Health Check Port	Default backend server port
Interval (s)	5
Timeout (s)	3
Healthy Threshold	3
Unhealthy Threshold	3

**Configure Health Check** dialog (Step 3):

- Health Check** toggle switch is turned on.
- Health Check Protocol**: HTTP (selected).
- Domain Name**: Specified domain name (selected), value: www.example.com.
- Health Check Port**: Default backend server port (selected).
- Path**: /.
- Description: "Start the path with a slash (/). The path can contain 1 to 80 characters, including letters, digits, and the following characters: / % ? &."
- OK** and **Cancel** buttons.

# Cloud Eye

- Cloud Eye is a multi-dimensional resource monitoring service. Users can use Cloud Eye to monitor resources, configure alarm rules, identify resource exceptions, and respond to resource changes.

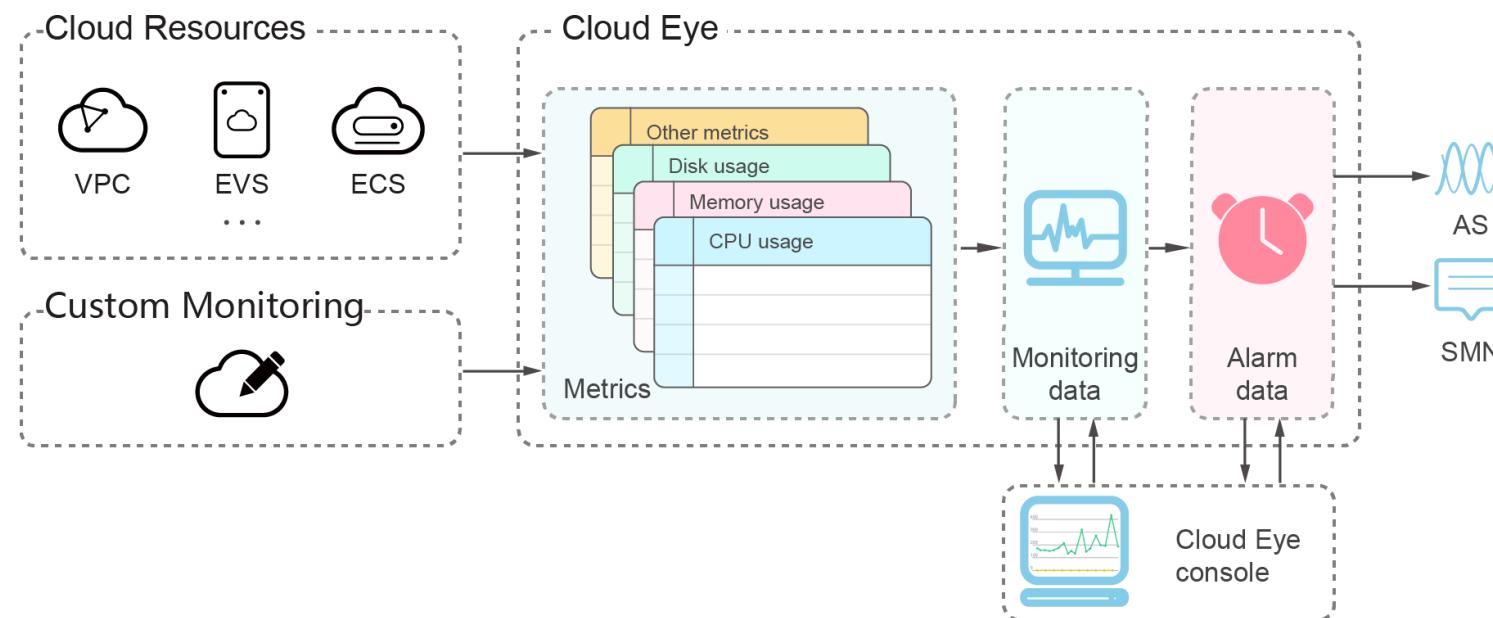


Cloud Eye

- O&M personnel cannot operate and maintain a system just based on their feelings.
- Cloud Eye is not a console where users can control different cloud services.
  - It is essentially a data collection system.
  - It is not tightly coupled with other services.
- Cloud Eye APIs can be integrated into third-party applications.
- Cloud Eye provides dashboards to visualize metrics and supports comprehensive alarm management.

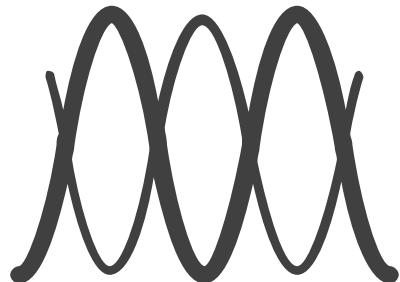
# Cloud Eye Architecture

- Cloud Eye collects cloud resource or custom metrics in real time. It allows users to flexibly configure alarm rules based on the collected data. When alarms are triggered, Cloud Eye notifies users using methods such as email or SMS message, and some service systems can respond to alarms automatically to ensure that services run smoothly.



# Auto Scaling

- Auto Scaling helps you automatically adjust Elastic Cloud Server (ECS) and bandwidth resources to keep up with changes in demand based on pre-configured scaling policies. It allows you to add ECS instances or bandwidth resources to handle increases in load and also save money by removing resources that are sitting idle.



Auto Scaling

- Can create or delete ECS instances across AZs.
- Supports a broad range of conditions and policies to create or delete instances.
- Automatically identifies and replaces unhealthy instances.
- Works together with ELB to distribute incoming traffic across healthy instances.

# Scaling Policies and Scenarios



- Real-time monitoring of performance
- Automatic, dynamic scaling
- Reactive scaling

**Dynamic scaling based on performance**



- Regular workload changes
- Scaling by schedule
- Proactive scaling

**Scheduled scaling based on predictable workload changes**

# Using Auto Scaling - Creating a Scaling Template

Create a scaling template.

Create a scaling group.

Create a scaling policy.

Billing Mode: Pay-per-use

\* Region: CN South-Guangzhou

\* Name: as-config-2987

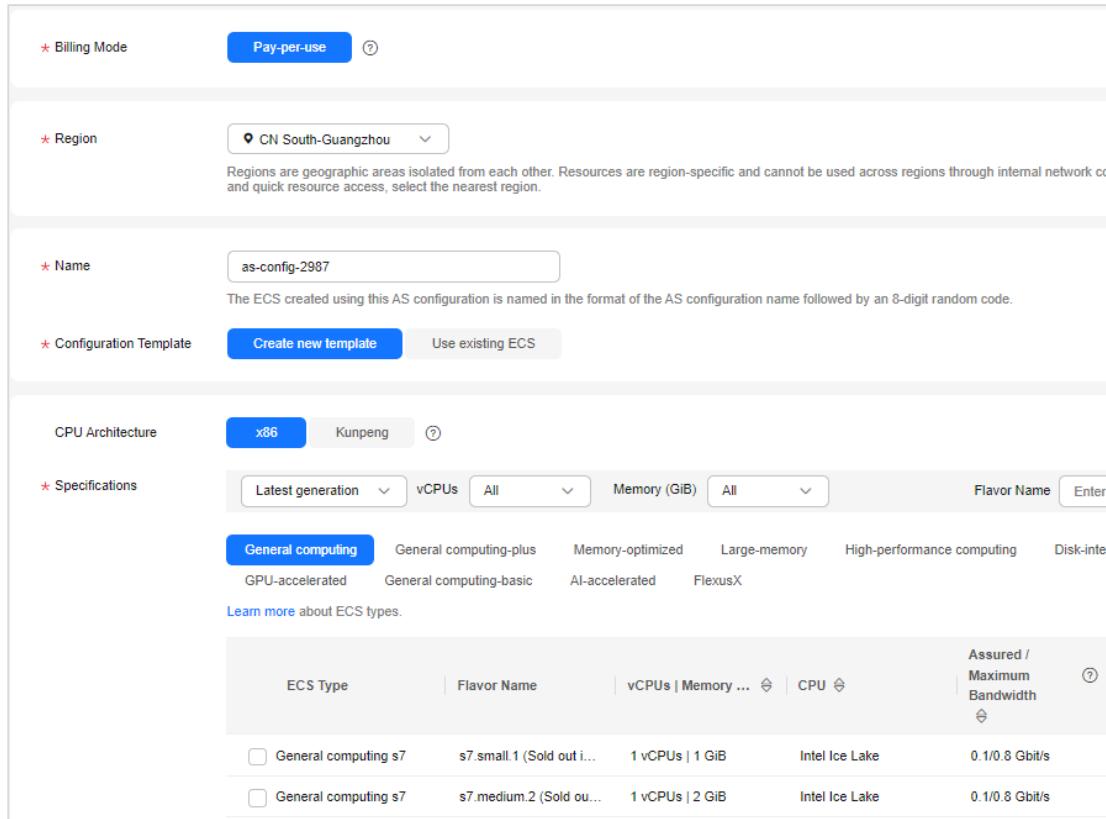
\* Configuration Template: Create new template

CPU Architecture: x86

\* Specifications: Latest generation, vCPUs: All, Memory (GiB): All, Flavor Name: Enter flavor name

General computing, GPU-accelerated, General computing-plus, General computing-basic, Memory-optimized, AI-accelerated, Large-memory, FlexusX, High-performance computing, Disk-inten...

ECS Type: General computing s7, s7.small.1 (Sold out i...), s7.medium.2 (Sold ou...), Flavor Name: s7.small.1, vCPUs: 1 vCPUs | 1 GiB, CPU: Intel Ice Lake, Assured / Maximum Bandwidth: 0.1/0.8 Gbit/s



Creating a template from scratch

\* Region: CN South-Guangzhou

\* Name: as-config-2987

\* Configuration Template: Create new template, Use existing ECS

Select ECS

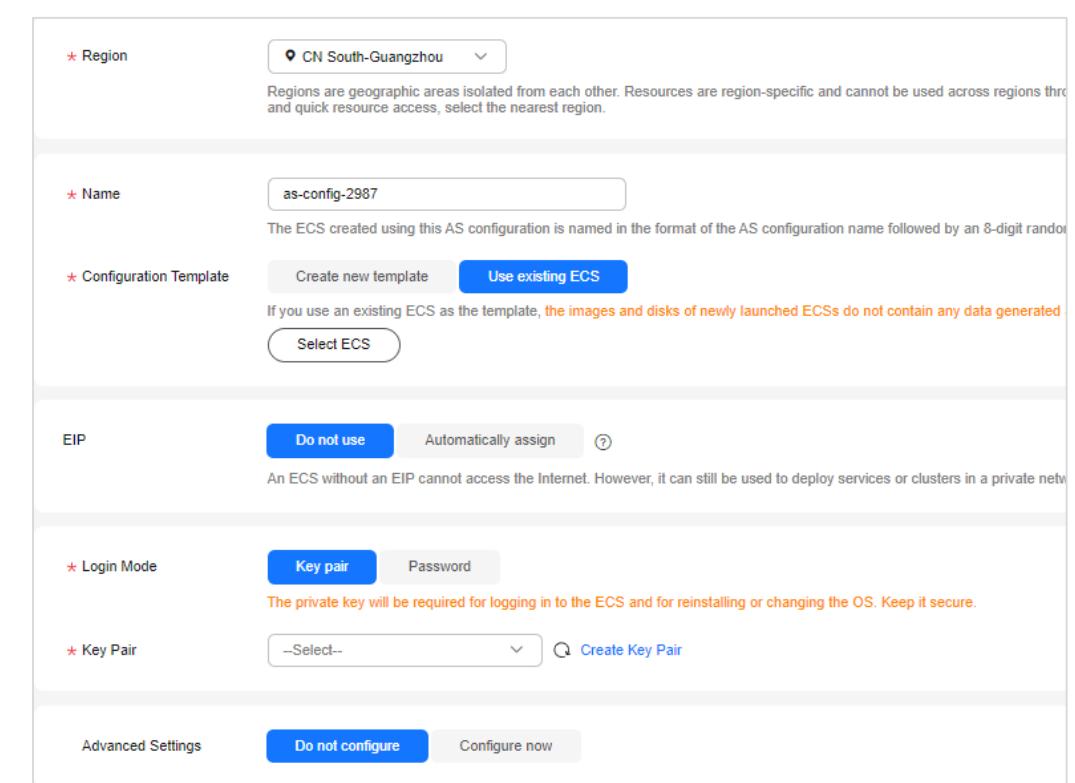
EIP: Do not use, Automatically assign

\* Login Mode: Key pair, Password

The private key will be required for logging in to the ECS and for reinstalling or changing the OS. Keep it secure.

\* Key Pair: Select, Create Key Pair

Advanced Settings: Do not configure, Configure now



Creating a template from an existing ECS

# Using Auto Scaling - Creating a Scaling Group

Create a scaling template.

Create a scaling group.

Create a scaling policy.

This screenshot shows the configuration for creating a scaling template. It includes fields for Region (CN South-Guangzhou), AZ (AZ7, AZ6, AZ5, AZ3, AZ2, AZ1), Multi-AZ Scaling Policy (Balanced selected), Name (as-group-20d9), Max. Instances (1), Expected Instances (0), and Min. Instances (0). The AS Configuration section is expanded, showing VPC (vpc-f298) and Subnet (subnet-f328). A note states: "The selected AS configuration serves as a specifications template for the instances in your AS group. After a subnet is selected, an IP address will be automatically assigned." At the bottom, there are Load Balancing options: "Do not use" and "Elastic load balancer".

This screenshot shows the configuration for creating a scaling group. It includes fields for Load Balancing (Do not use selected), Instance Removal Policy (Oldest instance created from oldest AS config...), EIP (Release selected), Data Disk (Delete selected), Health Check Method (ECS health check), Health Check Interval (5 minutes), and Health Check Grace Period (600 seconds). The Tag section allows adding tags with key and value fields. The Agency section has a dropdown for selecting an agency and a "Create Agency" button. An "Agreement" checkbox is present at the bottom.

# Using Auto Scaling - Creating a Scaling Policy

Create a scaling template.

Create a scaling group.

Create a scaling policy.

Add AS Policy

Policy Name: as-policy-6814

Policy Type: **Alarm** Scheduled Periodic  
Policies of this type are applied only when their associated alarm rules are enabled. [View alarm rules](#)

Alarm Rule: **Create** Use existing

Rule Name: as-alarm-6830

Monitoring Type: **System monitoring** Custom monitoring

Trigger Condition: CPU Usage Max. > %  
If you select a metric starting with (Agent), the Agent must be installed on all instances in the AS group. [Learn more](#)  
For more information about AS monitoring metrics, see [Monitoring Metrics](#).  
The metrics that can be monitored vary somewhat by OS. [Learn more](#)

Monitoring Interval: 5 minutes

Consecutive Occurrences:

Alarm Policy Type: **Simplified scaling** Refined scaling

Scaling Action: Add 1 instances

Cooldown Period (s): 300

**Alarm-based scaling policy**

Add AS Policy

Policy Name: as-policy-6814

Policy Type: Alarm **Scheduled** Periodic

Time Zone: GMT+08:00

Triggered On: Aug 15, 2024 14:48:46

The specified time must be later than the default time and the current system time.

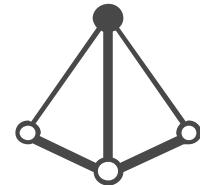
Scaling Action: Add 1 instances

Cooldown Period (s): 300

**Scheduled scaling policy**

# Cloud Services for Creating Scalable Systems

**Step 1** Distribute traffic to multiple servers.



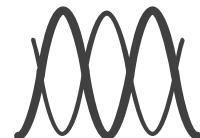
ELB

**Step 2** Get to know when it is the time to scale resources.



Cloud Eye

**Step 3** Quickly scale in or out resources.



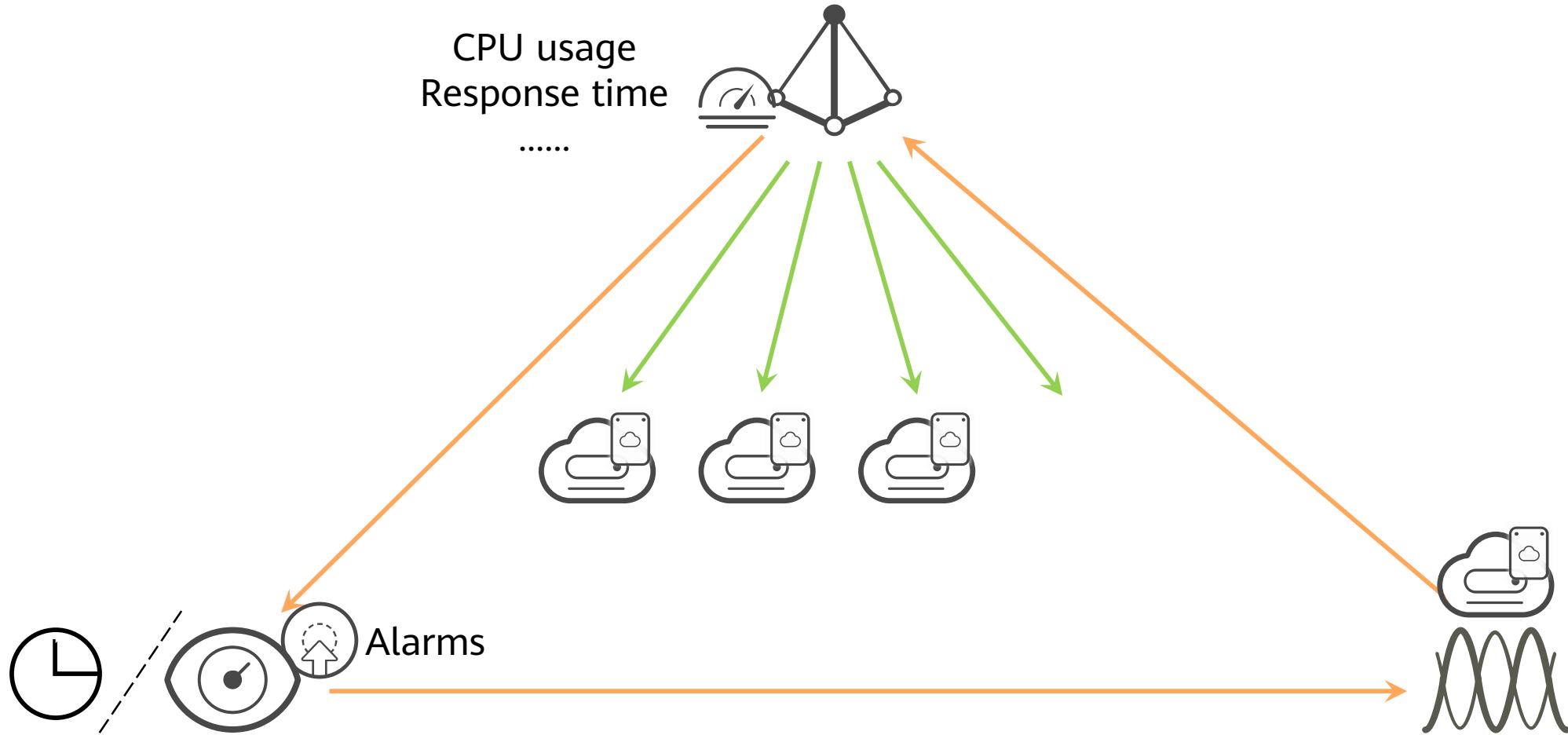
Auto Scaling



# Contents

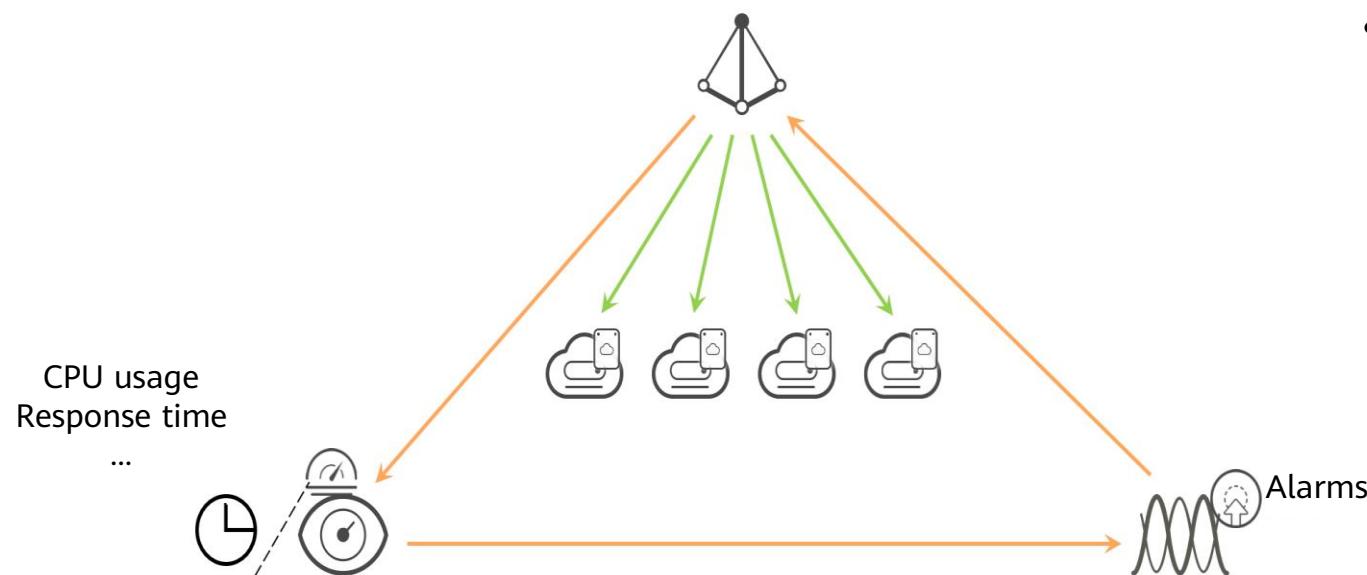
1. Scalability Implementation
2. How to Build Scalable Applications
- 3. Typical Scalable Website Architecture**

# Typical Scalable Website Architecture



# Application Scenarios

- **Heavy-traffic forums:** Service load changes are difficult to predict for heavy-traffic forum websites. AS dynamically adjusts the number of ECSs based on monitored ECS metrics, such as CPU usage and memory usage.
  - **Livestreaming:** A livestreaming website may broadcast popular programs from 14:00 to 16:00 every day. Auto Scaling automatically scales out ECS and bandwidth resources during this period to ensure a smooth viewer experience.



- **E-commerce:** During big promotions, E-commerce websites need more resources. Auto Scaling automatically adds resources within minutes to ensure that promotions go smoothly.



# Summary

- Learned the advantages of load balancing on the cloud.
- Learned how to schedule distributed applications.
- Learned how to adjust resources for distributed applications.



# Quiz

For a scalable system on Huawei Cloud, which of the following can be used to create or delete servers? ()

- A. Elastic Load Balance (ELB)
- B. Cloud Eye
- C. Auto Scaling
- D. Elastic Cloud Server (ECS)



# Acronyms and Abbreviations

- ECS: Elastic Cloud Server
- EVS: Elastic Volume Service



# Recommendations

- Huawei Cloud websites
  - Huawei Cloud: <https://www.huaweicloud.com/intl/en-us/>
  - Huawei Cloud Developer Institute: <https://edu.huaweicloud.com/intl/en-us/>



Huawei Cloud  
Developer Institute

# Thank You.

**Copyright © 2024 Huawei Technologies Co., Ltd. All Rights Reserved.**

The information in this document may contain predictive statements including, without limitation, statements regarding the future financial and operating results, future product portfolio, new technology, etc. There are a number of factors that could cause actual results and developments to differ materially from those expressed or implied in the predictive statements. Therefore, such information is provided for reference purpose only and constitutes neither an offer nor an acceptance. Huawei may change the information at any time without notice.

