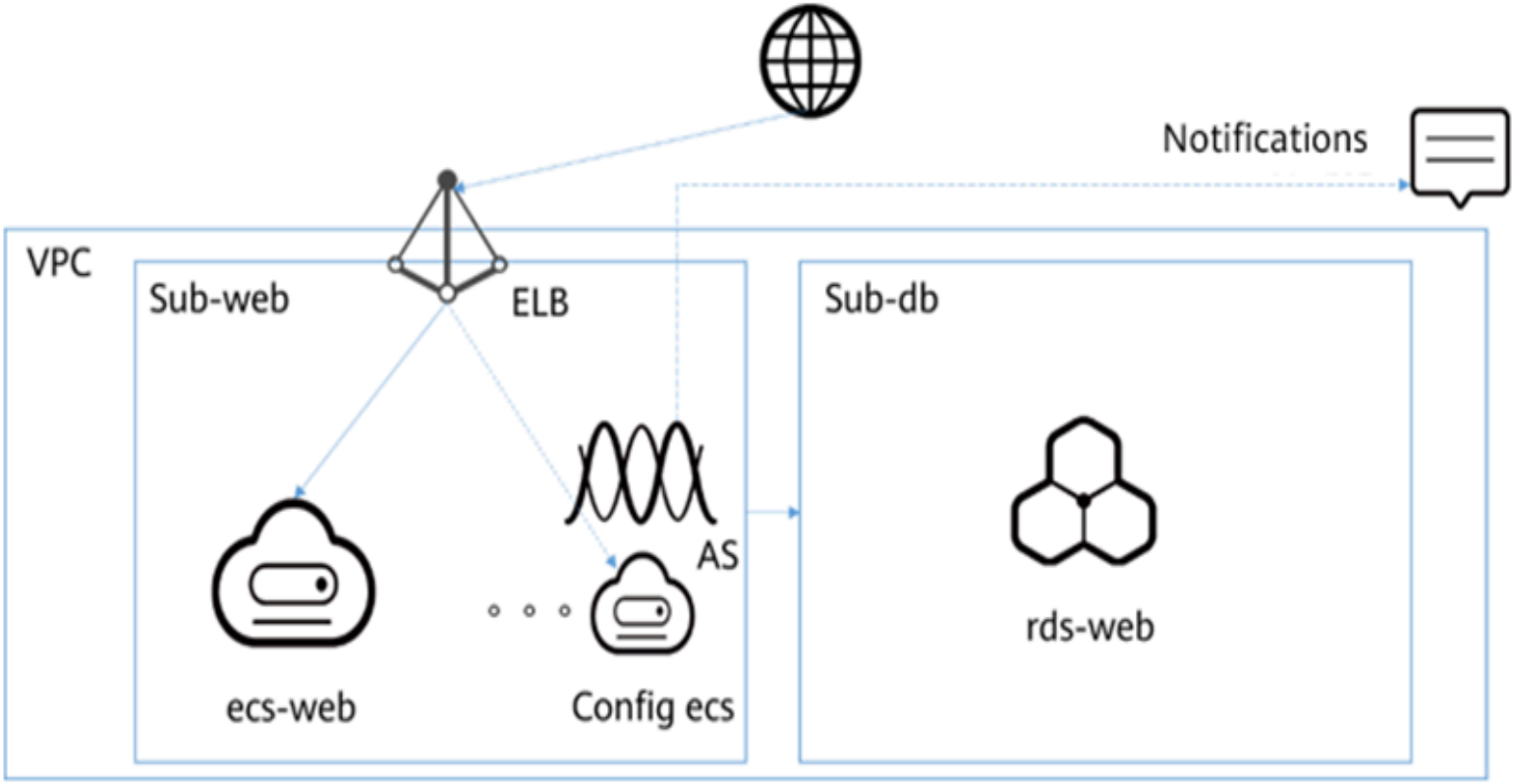
Building a Highly Elastic Service System on Huawei Cloud: A Discuz! Deployment Guide

🎯 Exercise Overview

Objective

Build a distributed, highly elastic service architecture for a Discuz! forum using Huawei Cloud services including Auto Scaling, Elastic Load Balance (ELB), and Relational Database Service (RDS).

Images



Key Technologies Used

- **Discuz!** - Popular open-source forum software
- **Auto Scaling (AS)** - For automatic scaling of compute and bandwidth resources
- **Elastic Load Balance (ELB)** - For traffic distribution
- **Relational Database Service (RDS)** - Managed MySQL database
- **Virtual Private Cloud (VPC)** - Network isolation
- **Image Management Service (IMS)** - For creating server templates

📋 Prerequisites & Setup

Important Notes Before Starting

1. **Use Preset Lab Environment** - Click the Preset Lab button to automatically provision infrastructure
2. **Follow Specifications Exactly** - Resources may be reclaimed if not purchased according to requirements
3. **Use Provided Credentials** - Do not use personal Huawei Cloud account credentials
4. **View Account Credentials** - Click the credentials icon as shown in the tutorial

Required Knowledge

- Basic understanding of Huawei Cloud infrastructure services
- Familiarity with Linux command line operations
- Fundamental knowledge of web application architecture

🏗️ Phase 1: Basic Environment Configuration

Step 1: Creating Subnet for Database Layer

Purpose: Isolate database traffic from web traffic for better security and performance.

Configuration Details:

- **VPC:** `vpc-primary` (pre-configured)
- **Subnet Name:** `subnet-db`
- **CIDR Block:** `192.168.2.0/24`
- **Location:** Same region as web subnet

Why This Matters:

- Separates database and web tiers for security
- Allows independent scaling of network resources
- Follows best practices for multi-tier architecture

Step 2: Creating Security Group for RDS

Security Group Configuration:

- **Name:** `sg-db`

- **Template:** Fast-add rule
- **Inbound Rule:** MySQL (3306)
- **Source Restriction:** Change from `0.0.0.0/0` to the **private IP of ecs-web**

Critical Security Consideration:

- Initially allows traffic from anywhere (0.0.0.0/0)
- **Must be restricted** to only the web server's private IP
- This prevents unauthorized database access from external sources

How to Find ECS Private IP:

1. Navigate to ECS Console
2. Locate `ecs-web` instance
3. Copy the private IP address from instance details

Step 3: Purchasing RDS Instance

Database Specifications:

Billing Mode Pay-per-use Region AP-Singapore DB Instance Name `rds-web`

DB Engine MySQL 5.7 Instance Type Primary/Standby Storage Type Cloud SSD

Primary AZ AZ5 Standby AZ AZ3 Time Zone UTC+08:00 Instance Class 2 vCPUs

Storage Space 40 GB VPC vpc-primary Subnet subnet-db Port 3306

Security Group sg-db Password Huawei@123#$

High Availability Design:

- Primary instance in AZ5
- Standby instance in AZ3
- Automatic failover capability
- Data redundancy across availability zones

Step 4: Installing and Configuring Discuz!

Installation Process:

1. **Access Discuz! Installer:** `http://<ECS-Public-IP>:80`
2. **System Check:** Verify directory permissions and dependencies

3. **Installation Type:** Select “A new installation Discuz! X (Contain UCenter Server)”
4. **Database Configuration:**
    - Database Server: RDS instance private IP
    - Database Password: Root password set during RDS creation
    - Admin Password: Set Discuz! administrator password

Verification:
After installation, access `http://<ECS-Public-IP>:80` again to confirm Discuz! is running properly.

🔄 Phase 2: Configuring Elastic Load Balance (ELB)

Step 1: Unbinding EIP from ECS

Why Unbind EIP First?

- ECS public IP cannot be directly transferred to ELB
- ELB needs its own public IP to distribute traffic
- Prevents IP address conflicts

Process:

1. Navigate to ECS Console

2. Find `ecs-web` instance
3. More → Manage Network → Unbind EIP
4. Confirm unbinding operation

Step 2: Creating Load Balancer

ELB Configuration:

Type Shared load balancer Billing Mode Pay-per-use Region AP-Singapore

Name `elb-web` Network Type Public IPv4 VPC vpc-primary Subnet subnet-web

EIP Use existing

Listener Configuration:

Listener Name `listener-web` Frontend Protocol TCP Frontend Port 80

Access Control All IP addresses Backend Server Group `server_group-web`

Backend Protocol TCP Algorithm Weighted round robin Backend Port 80

Why TCP Instead of HTTP?

- TCP layer load balancing provides better performance
- No need for HTTP header inspection for this use case
- Simpler configuration and lower overhead

Step 3: Service Verification

Testing ELB:

1. Use ELB's EIP instead of ECS's old EIP
2. Access `http://<ELB-EIP>:80`
3. Verify Discuz! loads correctly

Health Check Mechanism:

- ELB continuously checks backend server health
- Sends periodic requests to ensure servers are responsive
- Automatically removes unhealthy servers from rotation
- **Note:** High frequency of health checks is normal and ensures service reliability

---

📈 Phase 3: Auto Scaling for Compute Resources

Step 1: Creating System Image

Purpose: Create a template of the configured Discuz! server for cloning.

Image Creation Process:

1. Navigate to Image Management Service
2. Create Image from `ecs-web`
3. Name: `image-web`
4. Type: System disk image
5. Region: AP-Singapore

Why Create an Image?

- Standardizes server configuration
- Enables rapid deployment of identical servers
- Essential for auto-scaling operations

Step 2: Configuring Auto Scaling Group

AS Configuration (`as-config-web`):

Billing Mode Pay-per-use CPU Architecture x86 Specifications 2 vCPUs

Image `image-web` (private) Disk 40 GiB High I/O Security Group `sg-web`

Login Mode Password

Auto Scaling Group (`as-group-web`):

Name `as-group-web` Max Instances 5 Expected Instances 2 Min Instances 1

VPC vpc-primary Subnet subnet-web Load Balancing `elb-web`

Removal Policy Oldest instance first

Step 3: Creating Scaling Policies

Scale-Out Policy:

- **Policy Name:** `as-policy-web`
- **Type:** Alarm-triggered
- **Trigger:** CPU Usage > 70%

- **Monitoring Interval:** 5 minutes
- **Consecutive Occurrences:** 2 times
- **Action:** Add 1 instance
- **Cooldown:** 300 seconds

Scale-In Policy:

- **Trigger:** CPU Usage < 30%
- **Consecutive Occurrences:** 2 times
- **Action:** Remove 1 instance
- **Cooldown:** 300 seconds

Understanding Cooldown Period:

- Prevents rapid oscillation between scale-in and scale-out
- Allows system to stabilize after scaling action
- Configurable based on application needs

Step 4: Testing Auto Scaling

Stress Test Procedure:

1. Scale out an ECS instance manually
2. Bind EIP to the new instance
3. Install stress tool: `yum -y install stress`

4. Generate CPU load: `stress --cpu 2 --timeout 900`
5. Monitor CPU usage in Cloud Eye
6. Wait for scaling policy to trigger (approx. 10 minutes)

Monitoring Points:

- CPU usage graphs in AS group monitoring
- Instance count changes
- ELB backend server group updates

🌊 Phase 4: Bandwidth Auto Scaling

Creating Bandwidth Scaling Policy

Configuration:

Policy Name `as-policy-web` Resource Type EIP Target EIP ELB's public IP

Trigger Condition Inbound Bandwidth > 18 Mbps Monitoring Interval 5 minutes

Consecutive Occurrences 3 times Scaling Action Increase to 30 Mbps

Cooldown Period 300 seconds

When to Use Bandwidth Scaling:

- During traffic spikes (marketing campaigns, viral content)
- Scheduled events with predictable traffic patterns
- Seasonal variations in user activity

ELB and Cloud Eye Integration:

- **Must be used together** for effective scaling
- ELB provides traffic distribution
- Cloud Eye provides metrics for decision making
- Auto Scaling executes scaling actions based on metrics

🛠️ Phase 5: Advanced Auto Scaling Management

Step 1: Manual Instance Management

Adding Instances Manually:

1. Navigate to AS group instances tab
2. Click “Add”
3. Select existing ECS (`ecs-web`)
4. Add to load balancer backend group

Use Cases for Manual Addition:

- Planned maintenance
- Known traffic spikes (product launches)
- Testing scenarios

Step 2: Instance Protection

Enabling Protection:

- Prevents automatic removal during scale-in
- Useful for stateful instances or special configurations
- **Note:** Health check failures still cause removal

How to Enable:

1. Locate ECS in AS group instances
2. Enable “Instance Protection” option
3. Instance will be excluded from automatic removal

Step 3: Changing Instance Specifications

Process:

1. AS Group → More → Modify AS Configuration
2. Deselect current specification
3. Select new specification (e.g., `c7n.large.4`)
4. New instances will use updated specifications

Considerations:

- Existing instances remain unchanged
- Only new instances created after modification use new specs
- May require image compatibility verification

Step 4: Notification Configuration

Setting Up Notifications:

1. Navigate to AS Group → Notifications tab
2. Click “Add Notification”
3. Select all scaling events
4. Create notification topic
5. Configure email/SMS recipients

Notification Events:

- Scale-out success/failure
- Scale-in success/failure
- Instance addition/removal
- Health check failures

❓ Frequently Asked Questions

Q1: What is a cooldown period and why is it required?

Answer: A cooldown period is a waiting time after a scaling action during which no new alarm-triggered scaling actions can occur. This prevents:

- Rapid oscillation between scaling states
- Unnecessary scaling costs
- System instability from frequent changes

Key Points:

- Does not affect scheduled or periodic scaling
- Configurable per scaling policy
- Typical values: 300-600 seconds

Q2: What types of sticky sessions does ELB support?

Answer: ELB supports three types of session persistence:

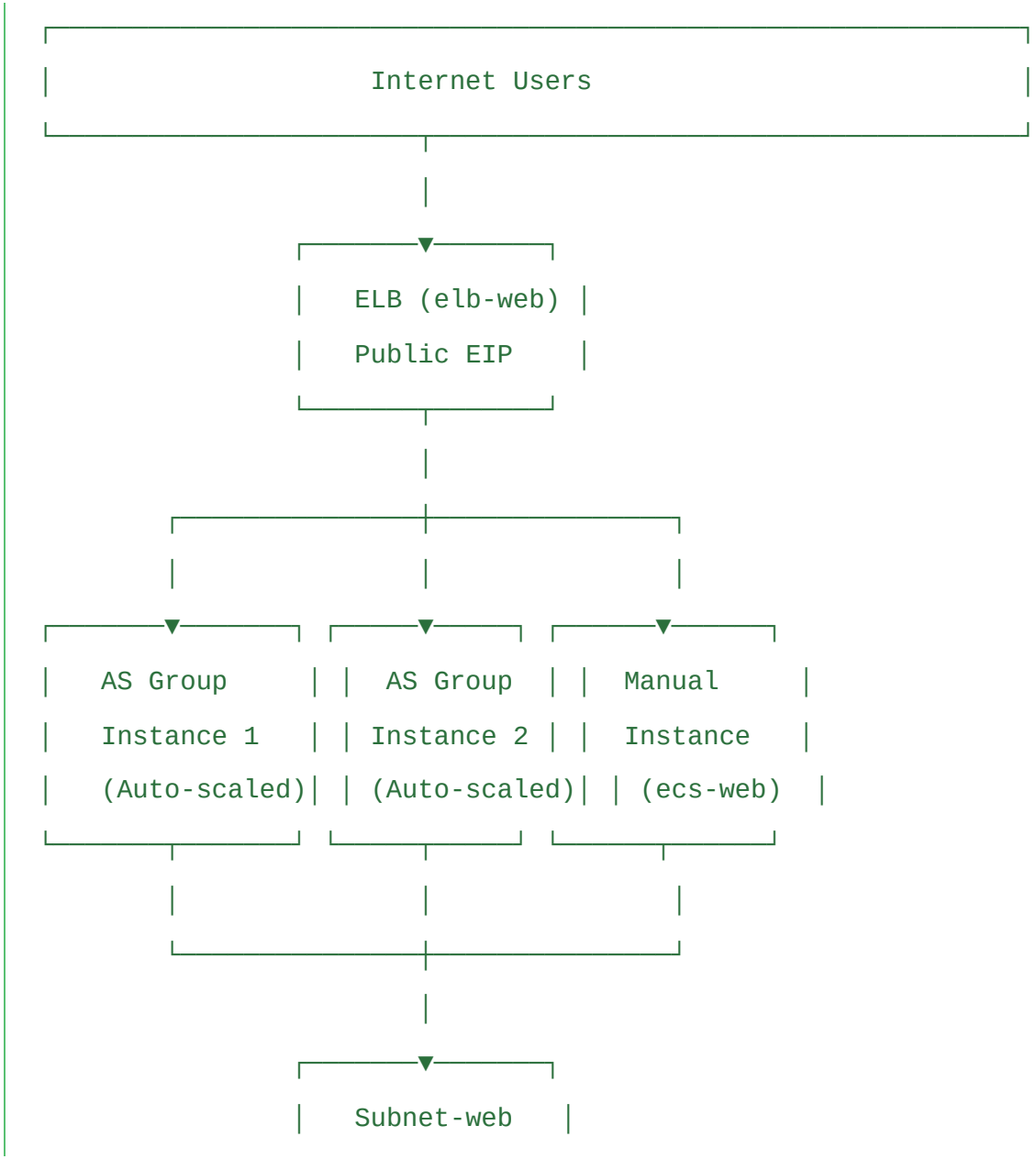
Dedicated ELB	1. Source IP address 2. Load balancer cookie	Shared ELB	1. Source IP address 2. Load balancer cookie 3. Application cookie

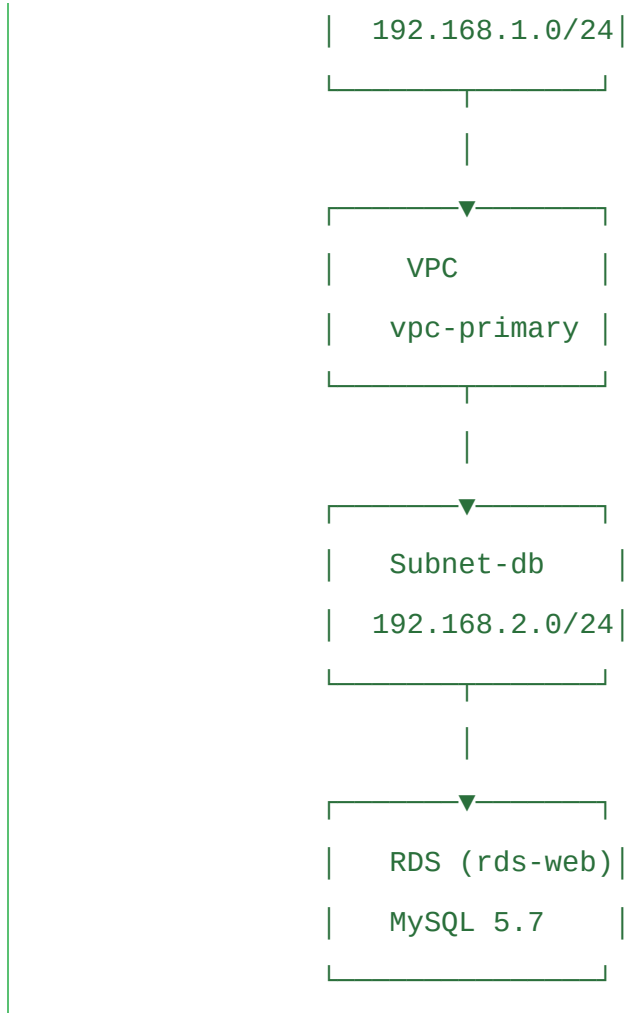
When to Use Sticky Sessions:

- Shopping carts in e-commerce
- Multi-step forms
- User authentication states
- Any application requiring session continuity

📊 Architecture Summary

Final Architecture Components:





Key Benefits Achieved:

1. **High Availability:** Multi-AZ deployment for RDS and potential ECS distribution
2. **Auto-scaling:** Both compute and bandwidth scale automatically
3. **Load Distribution:** ELB distributes traffic evenly
4. **Cost Optimization:** Pay-per-use model with automatic scale-in
5. **Maintainability:** Standardized images and automated processes

💡 Best Practices & Recommendations

Security Considerations:

1. Always restrict database access to specific IPs
2. Use strong passwords for all services
3. Regularly update system images with security patches
4. Monitor access logs for unusual patterns

Performance Optimization:

1. Set appropriate scaling thresholds based on actual usage patterns
2. Use connection pooling for database access
3. Implement caching where appropriate
4. Regularly review and adjust scaling policies

Cost Management:

1. Set appropriate minimum and maximum instance limits
2. Monitor unused resources
3. Use pay-per-use billing for unpredictable workloads
4. Consider reserved instances for baseline workloads

Monitoring and Maintenance:

1. Set up comprehensive Cloud Eye alerts
2. Regular backup of RDS data
3. Periodic testing of failover scenarios
4. Document all configurations and changes

🔮 Future Enhancements

Consider adding these components for a more robust architecture:

1. **Content Delivery Network (CDN)** for static assets
2. **Object Storage Service (OBS)** for user uploads
3. **Distributed Cache Service (DCS)** for session storage
4. **Web Application Firewall (WAF)** for security
5. **API Gateway** for microservices architecture

This guide provides comprehensive documentation of building a highly elastic service system on Huawei Cloud. Save this document for future reference, troubleshooting, and architecture planning.