That’s a fun challenge! Here is your lab guide, but with all the technical explanations turned into simple, child-friendly analogies, keeping the structure the same.

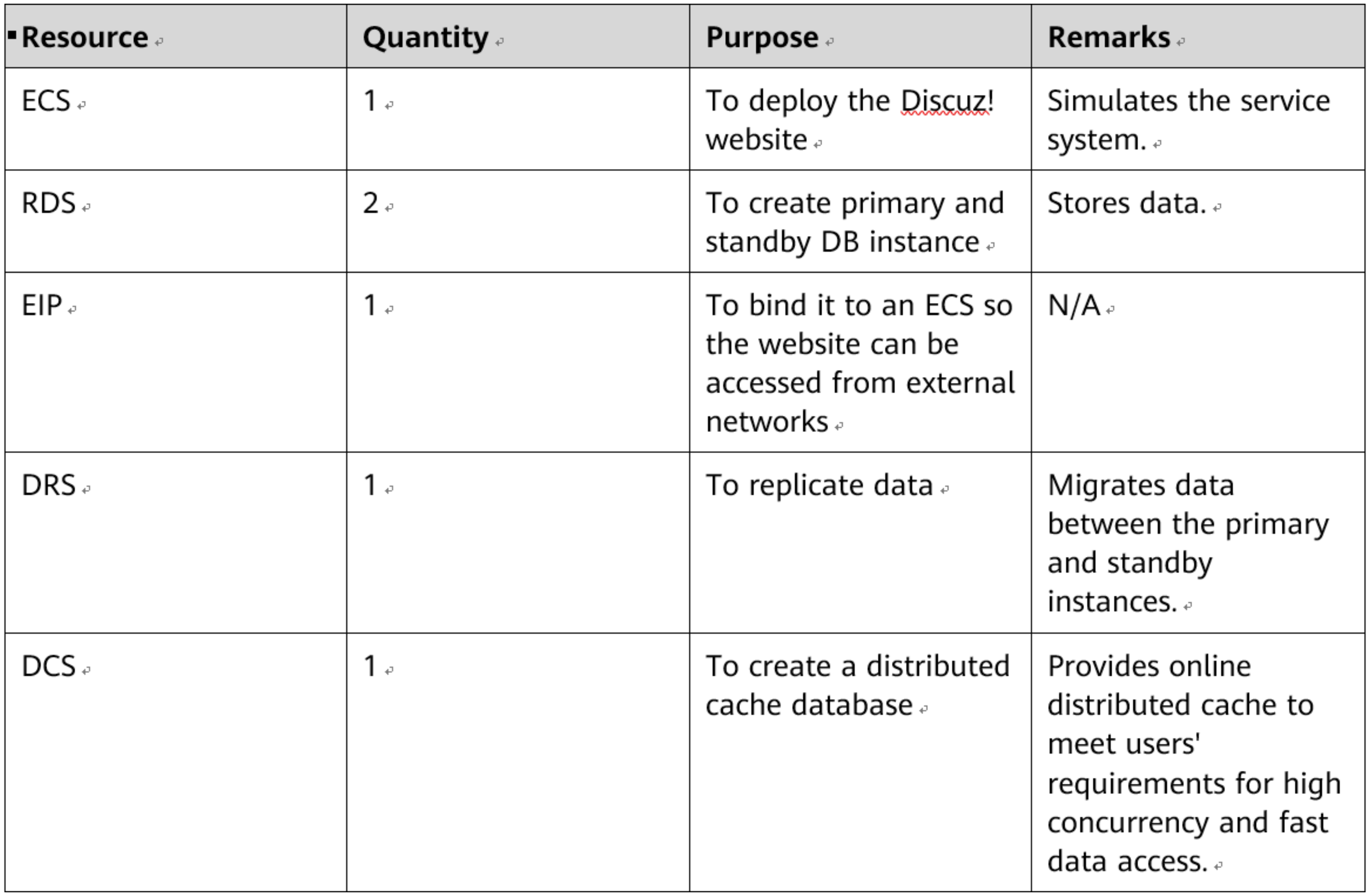
☁️ Highly Reliable Database Solution Lab Guide (Huawei Cloud)

📌 Overall Goal

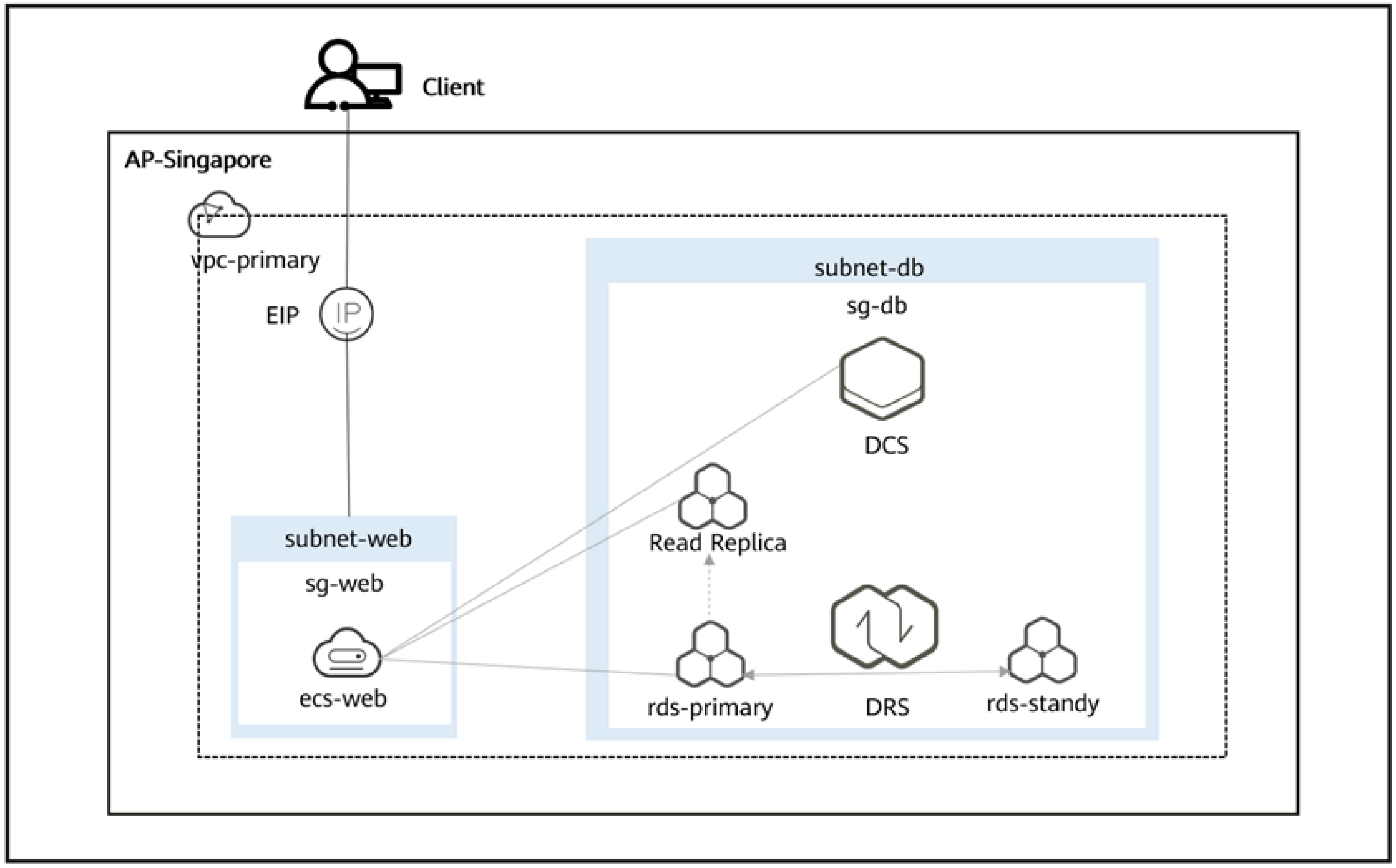
This whole exercise is like building a **super-strong, super-fast toy city** for your Discuz! toys to play in!

1. **RDS (Relational Database Service):** Making sure your toys are kept safe and sound in their main toy box.
2. **DRS (Data Replication Service):** Making an emergency backup toy box just in case!
3. **DCS (Distributed Cache Service - Redis):** Building a special, fast ticket booth to get toys quickly.

Images



Resource	Quantity	Purpose	Remarks
ECS	1	To deploy the Discuz! website	Simulates the service system.
RDS	2	To create primary and standby DB instance	Stores data.
EIP	1	To bind it to an ECS so the website can be accessed from external networks	N/A
DRS	1	To replicate data	Migrates data between the primary and standby instances.
DCS	1	To create a distributed cache database	Provides online distributed cache to meet users' requirements for high concurrency and fast data access.



I. Basic Environment Configuration (The Foundation)

This part is like drawing the map and building the first house for your toy city.

🛠️ Step 1: Creating Network Resources (Special Roads and Guards)

Create Subnet (`subnet-db`) A dedicated area on the map.

We are building a special, quiet road just for the important **Toy Boxes (Databases)**. This keeps them away from the busy highway where the **Web Servers** drive.

Create Security Group (`sg-db`) A fence with a gate.

We are putting a **security fence** around the database road. This fence has a special guard who checks everyone who wants to enter.

Modify Inbound Rule (Source: `sg-web`) Giving the Web Server car a special pass.

The fence guard is told: “Only let the cars coming from the **Web Server’s driveway (** **`sg-web`** **)** pass through this gate to talk to the toy boxes!” This keeps the important data safe from strangers.

💾 Step 2: Buying the Primary RDS Instance (The Main Toy Box)

Name `rds-primary` The first and most important toy box.

Type **Primary/Standby**

This is like buying **two** toy boxes and putting them in **two different rooms** right from the start. They will always match!

AZs Primary AZ1, Standby AZ5

Key for HA: We put them in two different rooms (AZ1 and AZ5). If a meteor hits room 1, we still have the toy box in room 5!

Network `vpc-primary` ** ** `subnet-db` **

🌐 Step 3: Installing Discuz! and Connecting to RDS (Moving the Toys In)

Access Website Use the web car’s address.

We drive to the new website (the toy store) to set it up.

Configure Database Address of `rds-primary`

We tell the website: “Every single time you need to save a toy or remember something, you MUST use the address of this **Main Toy Box**!”

II. Highly Reliable Database Design

This part is about making sure the toy city never has a problem and can serve lots of friends.

💾 Step 1: Creating a Standby RDS Instance (The Emergency Toy Box)

Create	`rds-standby`	Creating the second toy box.

We are building the **Backup Toy Box**. It’s there just in case the main one breaks, but for now, it’s empty.

Floating IP Address	**Record this IP!**	This is the address of the empty backup box.

🔄 Step 2: Creating a DRS Task for Synchronization (The Magic Copy Machine)

What is DRS? This is the **Magic Copy Machine** that moves things between the main box and the emergency box.

Migration Type Full + Incremental

Best for Toys: First, copy all the toys we have now (`Full`). Then, keep watching the main box and instantly copy any **NEW** toys put in (`Incremental`). This makes sure the backup box is always perfect!

Source/Destination `rds-primary` to `rds-standby`

The flow of the magic copy: **Main Box** $\rightarrow$ **Emergency Box**.

Configure Step Confirm permissions.

We tell the Magic Copy Machine: “Yes, you are allowed to look inside both boxes and move things!”

📈 Step 3: Creating a Read Replica (The Photocopies for Friends)

Why a Read Replica? To stop friends from fighting over the main toy box.

Creation Method Create Read Replica from `rds-primary` .

We tell the Primary Toy Box to make a special **Photocopy Station**!

Result New instance `replica-discuz` .

Now, when friends want to **look** at the toys (read), they can use the photocopy station. The Main Toy Box is only used for putting toys away (writes). This makes everything faster!

III. Cache Database Design

This part is about adding a super-speed machine to the toy city.

🚀 Step 1: Creating a DCS Instance (The Fast-Pass Ticket Booth)

What is DCS? This is Huawei Cloud's **Fast-Pass Line** for getting information. It stores data in super-fast memory (RAM), not on a slow hard drive.

Cache Engine	**Redis**	The name of the special Fast-Pass technology.
Instance Type	**Master/Standby**	Even the Fast-Pass Booth has a backup! If the main booth breaks, the backup one instantly opens.
IP Address	**Record this IP!**	This is the address of the Fast-Pass Booth.

⚙️ Step 2: Configuring the Cache Service for Discuz! (Building the Fast-Pass Gate)

This step is where we tell the website how to use the new Fast-Pass Booth!

1. **Log in to `ecs-web`:** We remotely control the Web Server car.
2. **Install PHP Redis Plugin:**
   ```
   yum install php-redis -y
   ```
   This is like installing a special walkie-talkie so the Web Server car can talk to the Fast-Pass Booth.
3. **Restart Web Service:**
   ```
   systemctl restart httpd
   ```

We turn the car off and on so it can start using the new walkie-talkie.

4. **Edit Configuration File:**

   ```
   vi /var/www/html/config/config_global.php
   ```

   We open the website's instruction book.

5. **Configure `CONFIG MEMORY`:**
   - We write the Fast-Pass Booth's address into the instruction book.
     *Now, before the website goes to the slow Toy Box, it checks the Fast-Pass Booth first. If the information is there, it's **super fast**!*