Part 3. Storing data in the cloud

There is one guy named Singleton in your office who knows all about your file server. If Singleton is out of office, no one else can maintain the file server. As you can imagine, while Singleton is on vacation, the file server crashes. No one else knows where the backup is located, but the boss needs a document now or the company will lose a lot of money. If Singleton had stored his knowledge in a database, coworkers could look up the information. But because the knowledge and Singleton are tidily coupled, the information is unavailable.

Imagine a virtual machine where important files are located on hard disk. As long as the virtual machine is up and running, everything is fine. But everything fails all the time, including virtual machines. If a user uploads a document on your website, where is it stored? Chances are high that the document is persisted to hard disk on the virtual machine. Let's imagine that the document was uploaded to your website but persisted as an object in an independent object store. If the virtual machine fails, the document will still be available. If you need two virtual machines to handle the load on your website, they both have access to that document because it is not tightly coupled with a single virtual machine. If you separate your state from your virtual machine, you will be able to become fault tolerant and elastic. Let highly specialized solutions like object stores and databases persist your state.

AWS offers many ways to store your data. The following table can help you decide which service to use for your data at a high level. The comparison is only a rough overview. We recommend that you choose two or three services that best fit your use case and then jump into the details by reading the chapters to make your decision.

Table 1. Overview of data storage services

Service	Access	Maximum storage volume	Latency	Storage cost
S3	AWS API (SDKs, CLI), third-party tools	Unlimited	High	Very low
EBS (SSD)	Attached to an EC2 instance via network	16 TiB	Low	Low
EC2 Instance Store (SSD)	Attached to an EC2 instance directly	305 TB	Very low	Very low
EFS	NFSv4.1, for example, from an EC2 instance or on-premises	Unlimited	Medium	Medium
RDS (MySQL, SSD)	SQL	64 TiB	Medium	Low
ElastiCache	Redis/Memcached protocol	635 GiB	Low	High
DynamoDB	AWS API (SDKs, CLI)	Unlimited	Medium	Medium

Chapter 7 will introduce S3, a service offering object storage. You will learn how to integrate the object storage into your applications to implement a stateless server.

Chapter 8 is about block-level storage for virtual machines offered by AWS. You will learn how to operate legacy software on block-level

storage.

Chapter 9 covers highly available block-level storage that can be shared across multiple virtual machines offered by AWS.

Chapter 10 introduces RDS, a service that offers managed relational database systems like PostgreSQL, MySQL, Oracle, or Microsoft SQL Server. If your applications use such a relational database system, this is an easy way to implement a stateless server architecture.

Chapter 11 introduces ElastiCache, a service that offers managed in-memory database systems like Redis or Memcached. If your applications need to cache data, you can use an in-memory database to externalize ephemeral state.

Chapter 12 will introduce DynamoDB, a service that offers a NoSQL database. You can integrate this NoSQL database into your applications to implement a stateless server.