Part 4. Architecting on AWS

Werner Vogels, CTO of Amazon.com, is quoted as saying “Everything fails all the time.” Instead of trying to reach the unreachable goal of an unbreakable system, AWS plans for failure in the following ways:

- Hard drives can fail, so S3 stores data on multiple hard drives to prevent the loss of data.
- Computing hardware can fail, so virtual machines can be automatically restarted on another machine if necessary.
- Data centers can fail, so there are multiple data centers per region that can be used in parallel or on demand.

Outages of IT infrastructure and applications can cause a loss of trust and money and are a major risk for businesses. You will learn how to prevent an outage of your AWS applications by using the right tools and architecture.

Some AWS services handle failure by default in the background. For some services, responding to failure scenarios is available on demand. And some services don’t handle failure by themselves but offer the possibility to plan and react to failure. The following table shows an overview of the most important services and their failure handling.

Designing for failure is a fundamental principle of AWS. Another is to make use of the elasticity of the cloud. You will learn about how to increase your number of virtual machines based on the current workload. This will allow you to architect reliable systems for AWS.

Table 1. Overview of services and their failure-handling possibilities

	Description	Examples
Fault tolerant	Services can recover from failure automatically without any downtime.	S3 (object storage), DynamoDB (NoSQL database), Route 53 (DNS)
Highly available	Services can recover from some failures automatically with little downtime.	RDS (relational database), EBS (network attached storage)
Manual failure handling	Services do not recover from failure by default but offer tools to build a highly available infrastructure on top of them.	EC2 (virtual machine)

Chapter 13 lays the foundation for becoming independent of the risk of losing a single server or a complete data center. You will learn how to recover a single EC2 instance either in the same data center or in another data center.

Chapter 14 introduces the concept of decoupling your system to increase reliability. You will learn how to use synchronous decoupling with the help of load balancers on AWS. You'll also see asynchronous decoupling using Amazon SQS, a distributed queuing service, to build a fault-tolerant system.

Chapter 15 is about automating the deployment of an application with little or no downtime. It is also a prerequisite for building highly available and fault-tolerant systems.

Chapter 16 uses a lot of the services you've discovered so far to build a fault-tolerant application. You'll learn everything you need to design a fault-tolerant web application based on EC2 instances (which aren't fault tolerant by default).

Chapter 17 is all about elasticity. You will learn how to scale your capacity based on a schedule or based on the current load of your system.

Chapter 18 presents options for running new and existing applications on AWS using containers.