aws academy

CSCI 5902 Adv. Cloud Architecting
Fall 2023
Instructor: Lu Yang

Module 11 Microservices & serverless (Sections 6-7) &
Module 12 Disaster recovery (Sections 1-3)
Nov 27, 2023

Housekeeping items and feedback

1. Start recording
2. The last lab assignment due on Dec 1
3. The last architecture assignment due on Dec 4
4. SLEQ
5. PIER tour

Recap of our last lecture

AWS Academy Cloud Architecting

Module 11: Building Microservices and Serverless Architectures

Module overview

Sections

1. Architectural need
2. Introducing microservices
3. Building microservice applications with AWS container services
4. Introducing serverless architectures
5. Building serverless architectures with AWS Lambda
6. Extending serverless architectures with Amazon API Gateway
7. Orchestrating microservices with AWS Step Functions

We stopped here in the last lecture

Module 13: Building Microservices and Serverless Architectures

Section 6: Extending serverless architectures with Amazon API Gateway

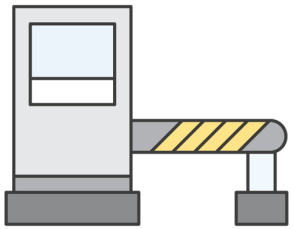
aws academy

Amazon API Gateway

Amazon API Gateway

- Enables you to create, publish, maintain, monitor, and secure APIs that act as entry points to backend resources for your applications
- Handles up to hundreds of thousands of concurrent API calls
- Can handle workloads that run on –
  - Amazon EC2
  - Lambda
  - Any web application
  - Real-time communication applications
- Can host and use multiple versions and stages of your APIs

Amazon API Gateway security



Require authorization

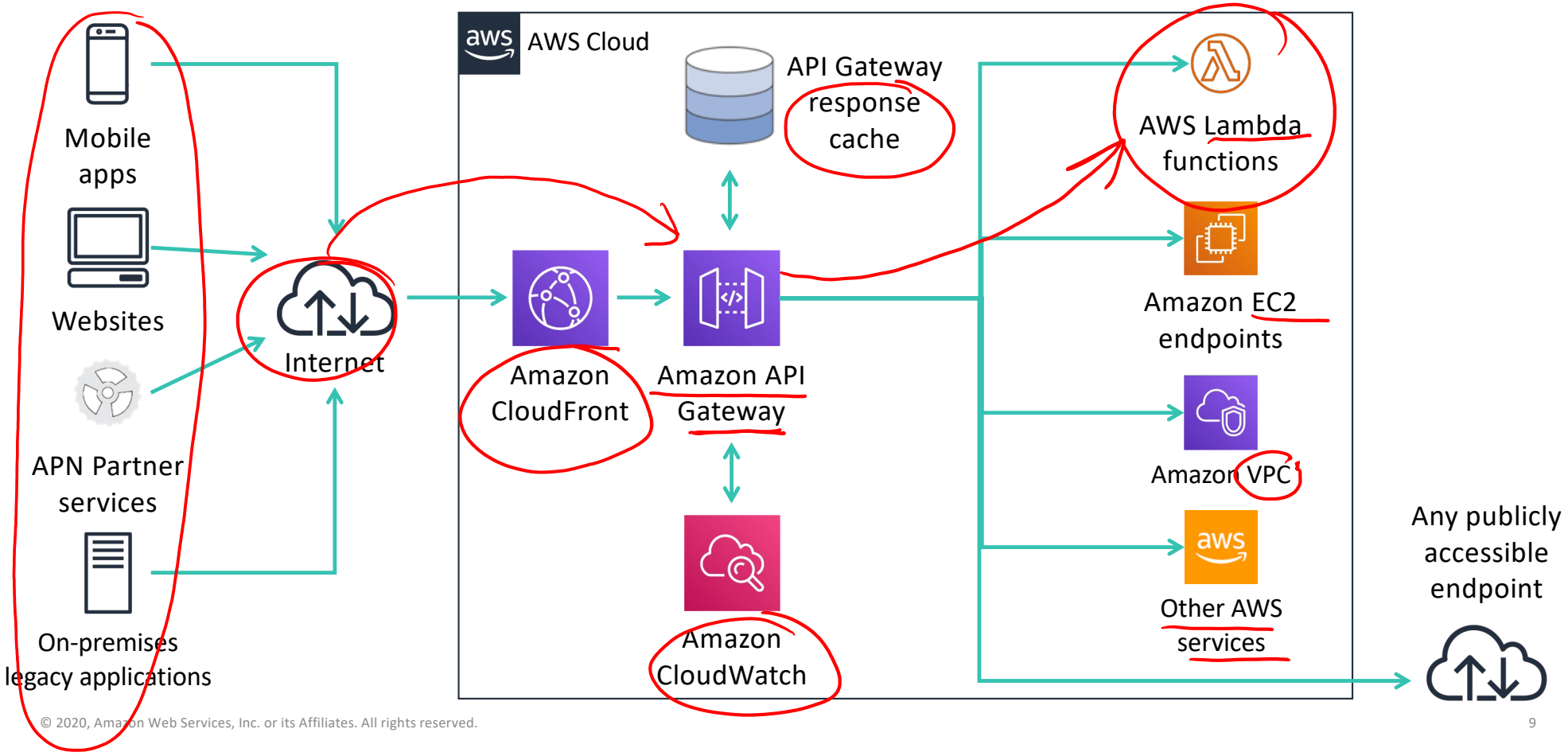
Apply resource policies

Throttling settings

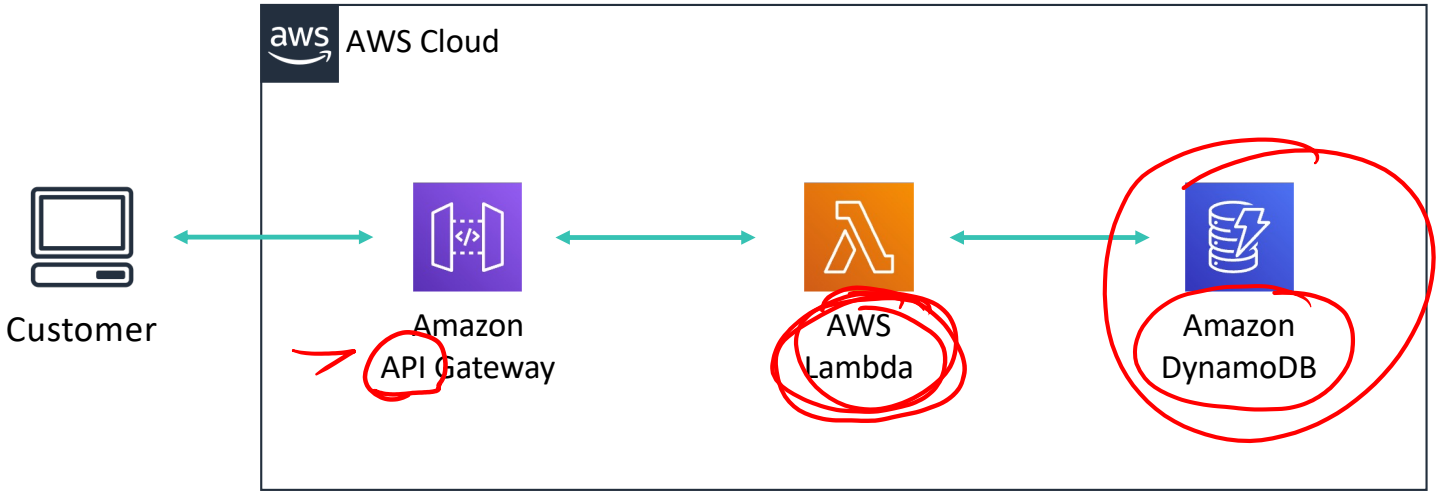
Protection from Distributed Denial of Service (DDoS) and injection attacks

AWS WAF

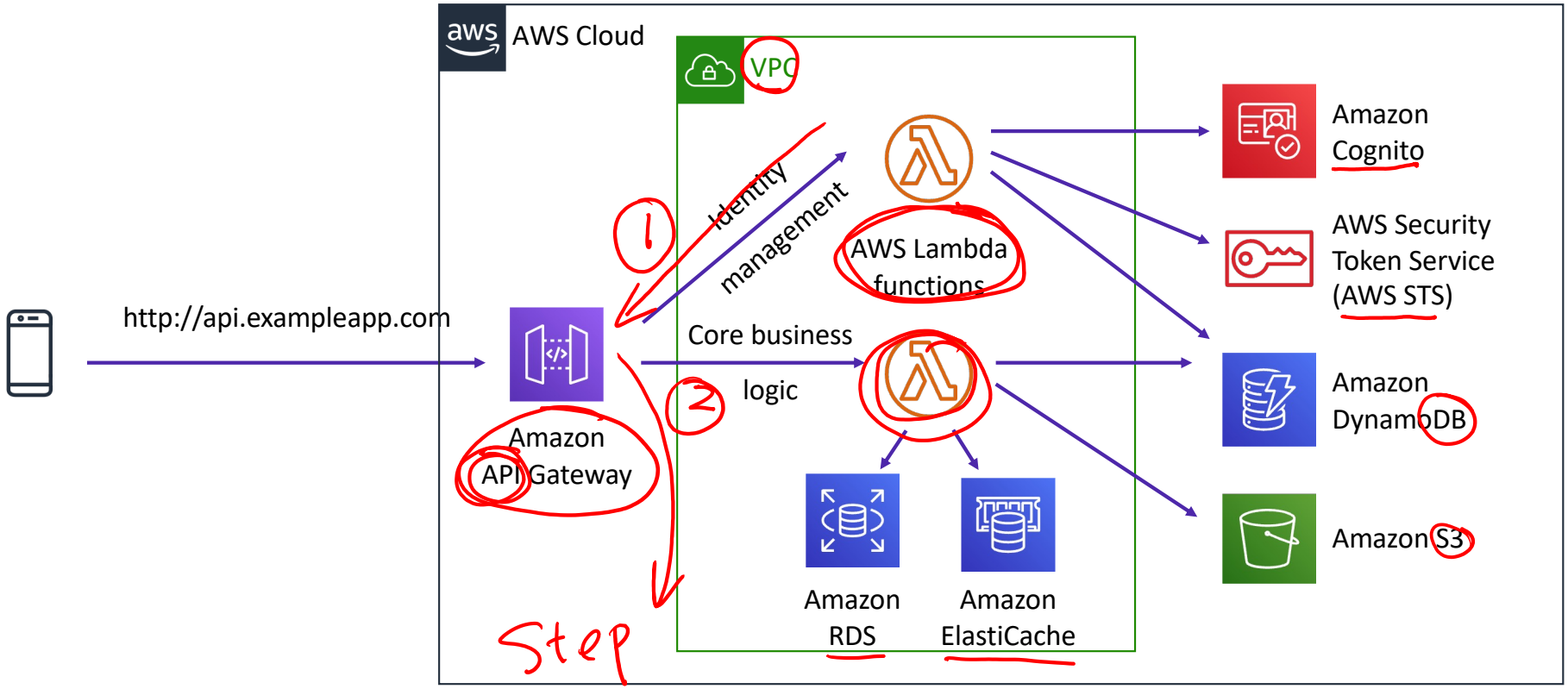
Amazon API Gateway: Common architecture example



Example: RESTful microservices



Example: Serverless mobile backend



Section 6 key takeaways

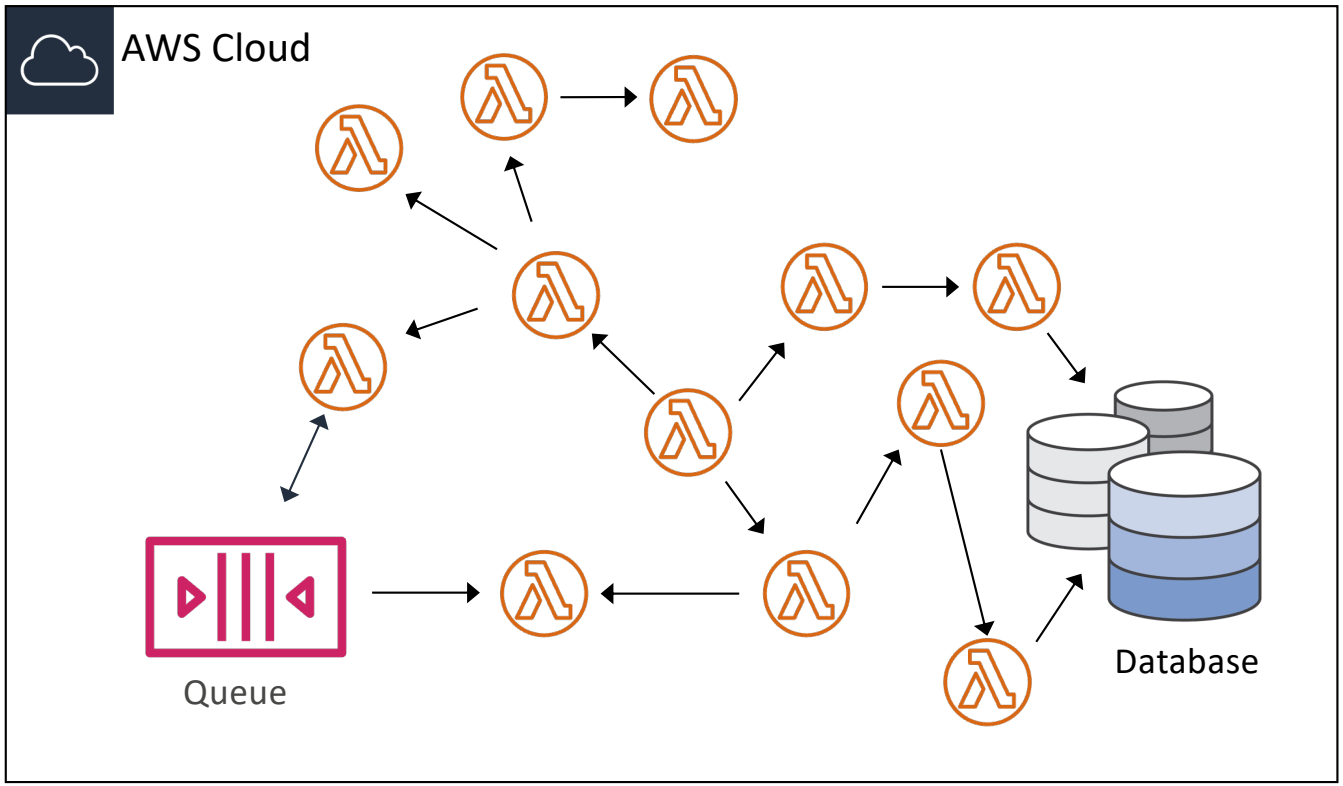
- Amazon API Gateway is a fully managed service that enables you to create, publish, maintain, monitor, and secure APIs at any scale.
- Amazon API Gateway acts as an entry point to backend resources for your applications. It abstracts and exposes APIs that can call various backend applications. These applications include Lambda functions, Docker containers that run on EC2 instances, VPCs, or any publicly accessible endpoint.
- Amazon API Gateway is deeply integrated with Lambda.

Module 13: Building Microservices and Serverless Architectures

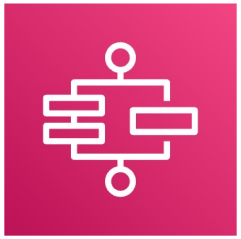
Section 7: Orchestrating microservices with AWS Step Functions

aws academy

Challenges with microservice applications



AWS Step Functions

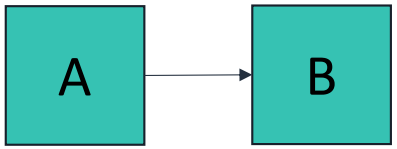


AWS Step Functions

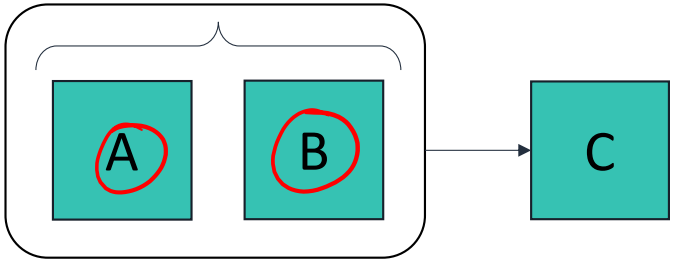
- Coordinates microservices by using visual workflows
- Enables you to step through the functions of your application
- Automatically triggers and tracks each step
- Provides simple error catching and logging if a step fails

- Allow you to create workflows that follow a fixed or dynamic sequences
- Built-in retry functionality – don't progress until success
- Native integration with AWS services like Lambda, SNS, SQS, Dynamodb, and so on
- Highly scalable and low cost ($0.25/1000 state transitions)

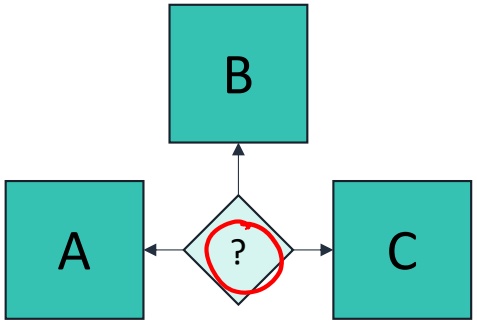
Workflow coordination



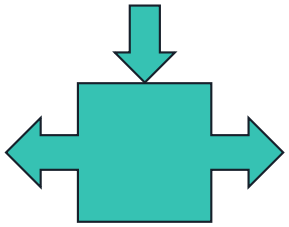
Run tasks in sequence



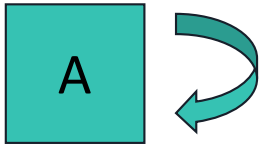
Run tasks in parallel



Select task based on data



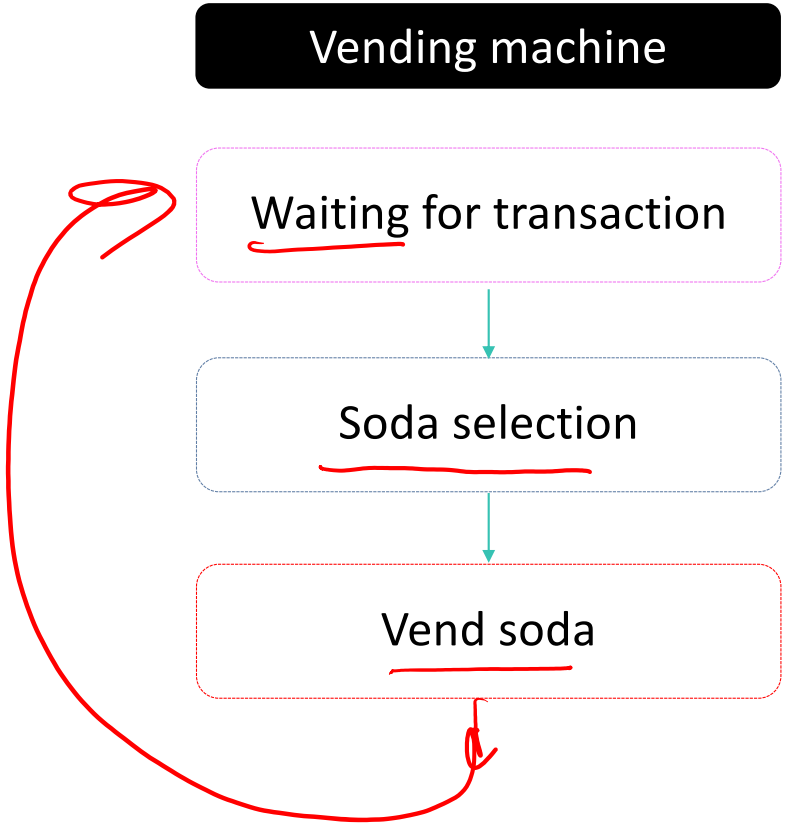
Manage try-catch-finally behavior



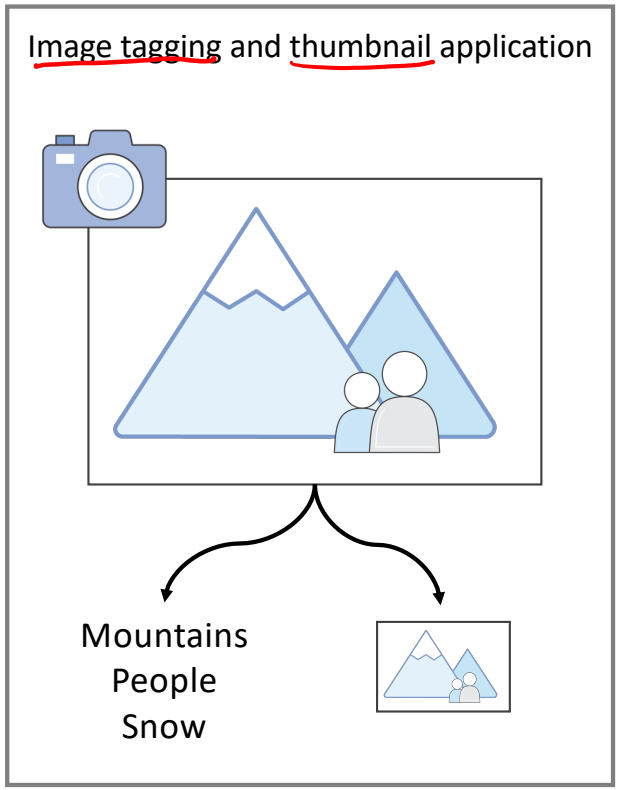
Retry failed tasks

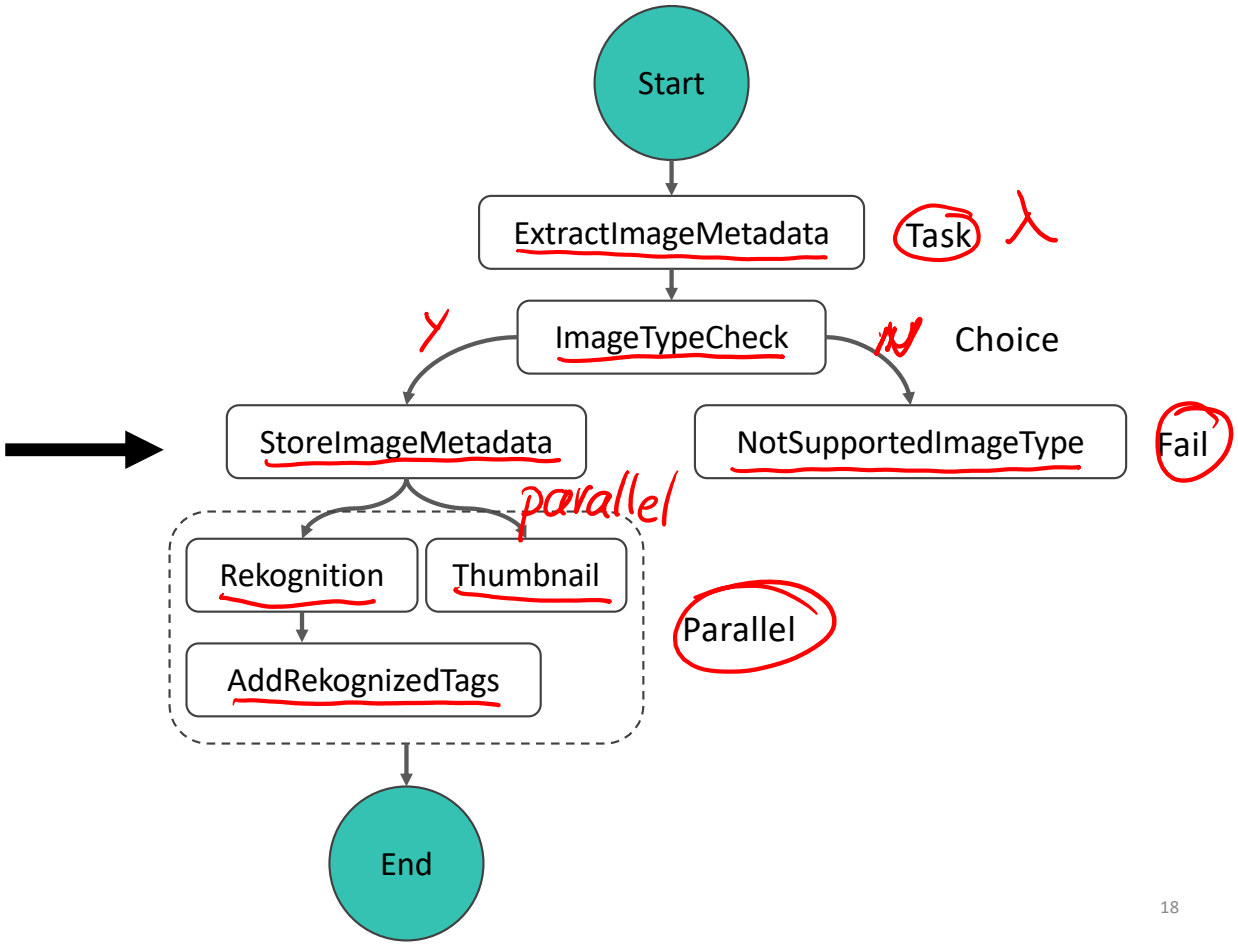
State machines

A state machine is a collection of states that can do work.



States





State types

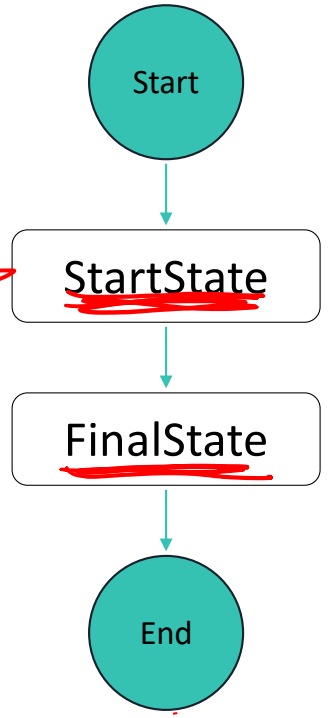
Task	A single unit of work performed by a state machine
Choice	Adds branching logic to a state machine
Fail	Stops a running state machine and marks it as a failure
Succeed	Stops a running state machine successfully
Pass	Passes its input to its output, without performing work
Wait	Delays from continuing for a specified time
Parallel	Creates parallel branches to run in your state machine
Map	Dynamically iterates steps

Amazon States Language

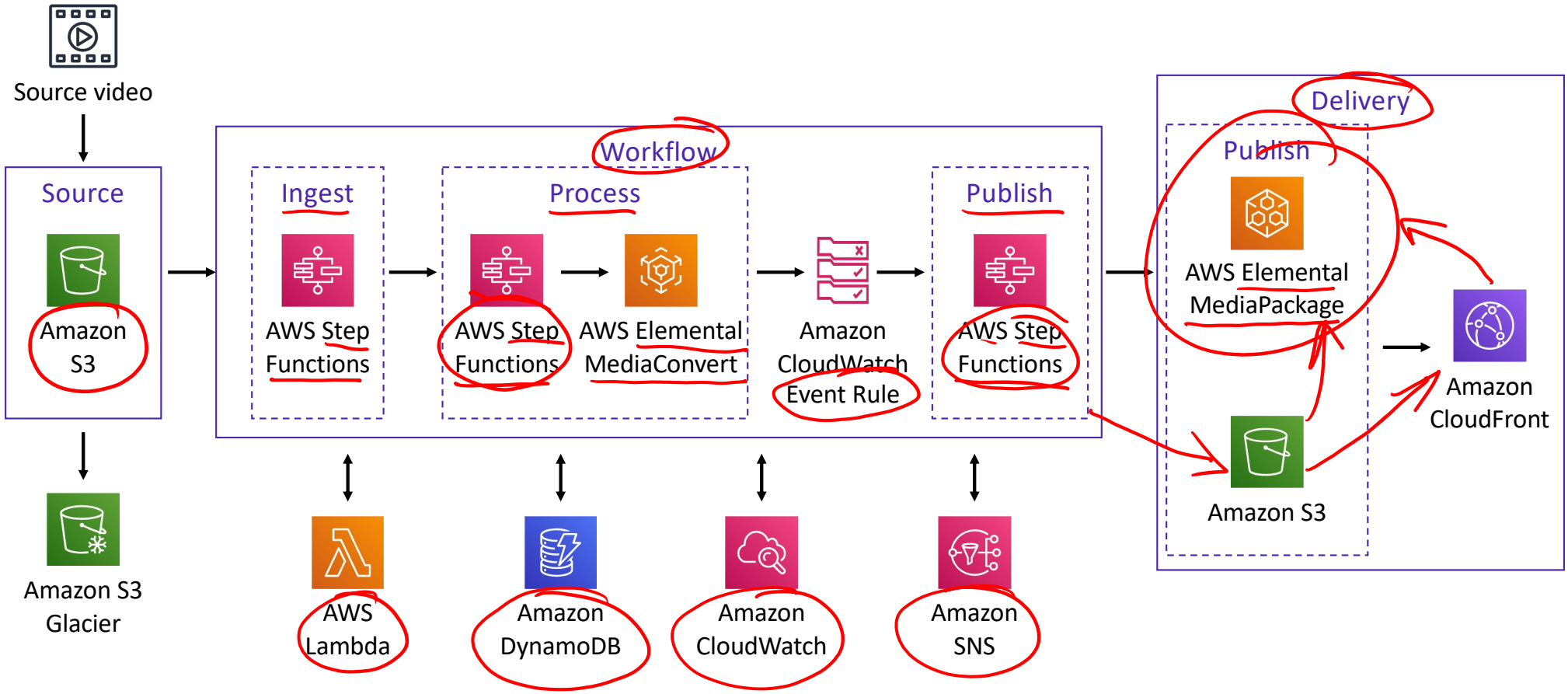
Define workflow in JSON
by using the Amazon States Language

```
{
 "Comment": "An example of the ASL.",
 "StartAt": "StartState",
 "States": {
 "StartState": {
 "Type": "Task",
 "Resource": "arn:aws:lambda:us-east…,
 "Next": "FinalState"
 }
 "FinalState": {
 "Type": "Task",
 "Resource": "arn:aws:lambda:us-east…,
 "End": true
 }
 }
}
```

Visualize workflow in the
Step Functions console



AWS Step Functions example: Video-on-demand (VOD) architecture



Demonstration: Creating an AWS Step function with Lambda: (https://www.youtube.com/watch?v=s0XFX3WHg0w)

Section 7 key takeaways

- AWS Step Functions is a web service that enables you to coordinate components of distributed applications and microservices by using visual workflows
- AWS Step Functions enables you to create and automate your own state machines within the AWS environment
- AWS Step Functions manages the logic of your application for you, and it implements basic primitives, such as sequential or parallel branches, and timeouts
- You define state machines by using the Amazon States Language

Prime Video audio/video monitoring service and reducing costs by 90%

https://www.primevideotech.com/video-streaming/scaling-up-the-prime-video-audio-video-monitoring-service-and-reducing-costs-by-90

https://www.youtube.com/watch?v=JTp0TY_2hXM

Module 13: Building Microservices and Serverless Architectures

Module wrap-up

aws academy

Module summary

In summary, in this module, you learned how to:

- Indicate the characteristics of microservices
- Refactor a monolithic application into microservices and use Amazon ECS to deploy the containerized microservices
- Explain serverless architecture
- Implement a serverless architecture with AWS Lambda
- Describe a common architecture for Amazon API Gateway
- Describe the types of workflows that AWS Step Functions supports

AWS Academy Cloud Architecting

Module 12: Planning for Disaster

Module overview

aws academy

Sections

1. Architectural need
2. Disaster planning strategies
3. Disaster recovery patterns

Module objectives

At the end of this module, you should be able to:

- Identify strategies for disaster planning
- Define recovery point objective (RPO) and recovery time objective (RTO)
- Describe four common patterns for backup and disaster recovery and how to implement them
- Use AWS Storage Gateway for on-premises-to-cloud backup solutions

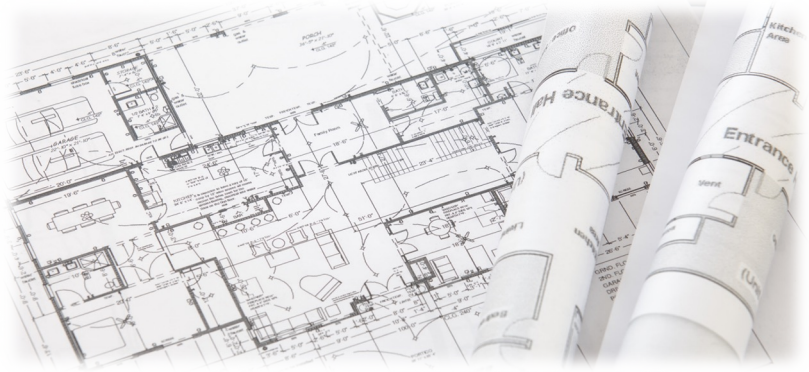
Module 14: Planning for Disaster

Section 1: Architectural need

aws academy

Café business requirement

If the café's infrastructure ever becomes unavailable, the staff must be able to get their applications running again within an amount of time that is acceptable to the business. They need an architecture that supports their disaster recovery plans while also optimizing for cost.



Module 14: Planning for Disaster

Section 2: Disaster planning strategies

aws academy

Planning for failures

"*Everything fails, all the time.*"
– Werner Vogels

Small-scale events

Large-scale events

Colossal events

How do you prepare for these events?

Avoiding and planning for disaster

High availability

- Minimize how often your applications and data become unavailable

Backup

- Make sure that your data is safe in case of disaster

Disaster recovery (DR)

- Recover your data and get your applications back online after a disaster

Selected AWS Well-Architected Framework design principles

Operational Excellence pillar

- Anticipate failure
- Refine operational procedures frequently

Reliability pillar

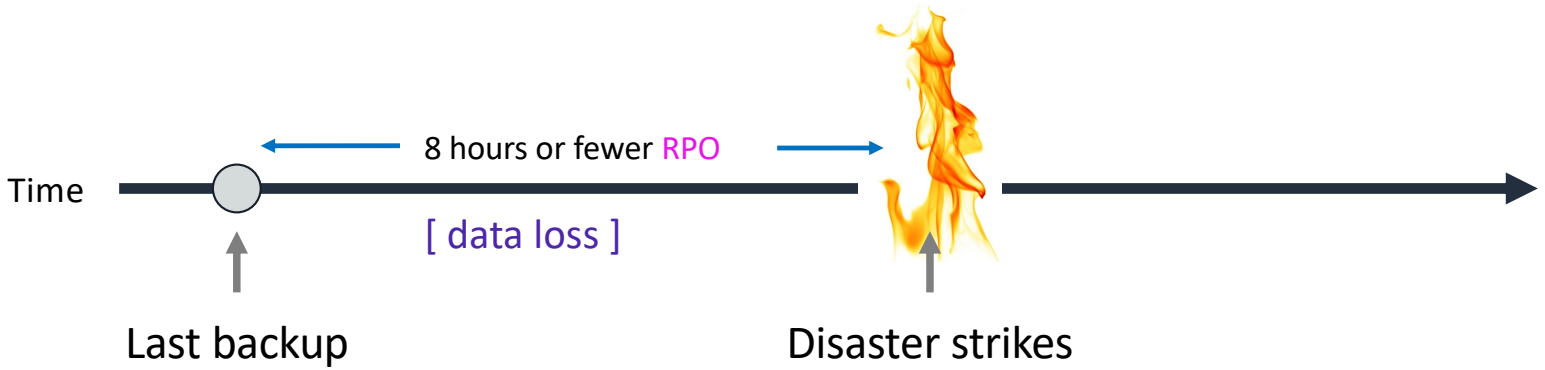
- Test recovery procedures
- Automatically recover from failure

Recovery point objective (RPO)

Recovery point objective (RPO) is the maximum acceptable amount of data loss, measured in time.

How often must your data be backed up?

Example RPO: *The business can recover from losing (at most) the last 8 hours of data.*

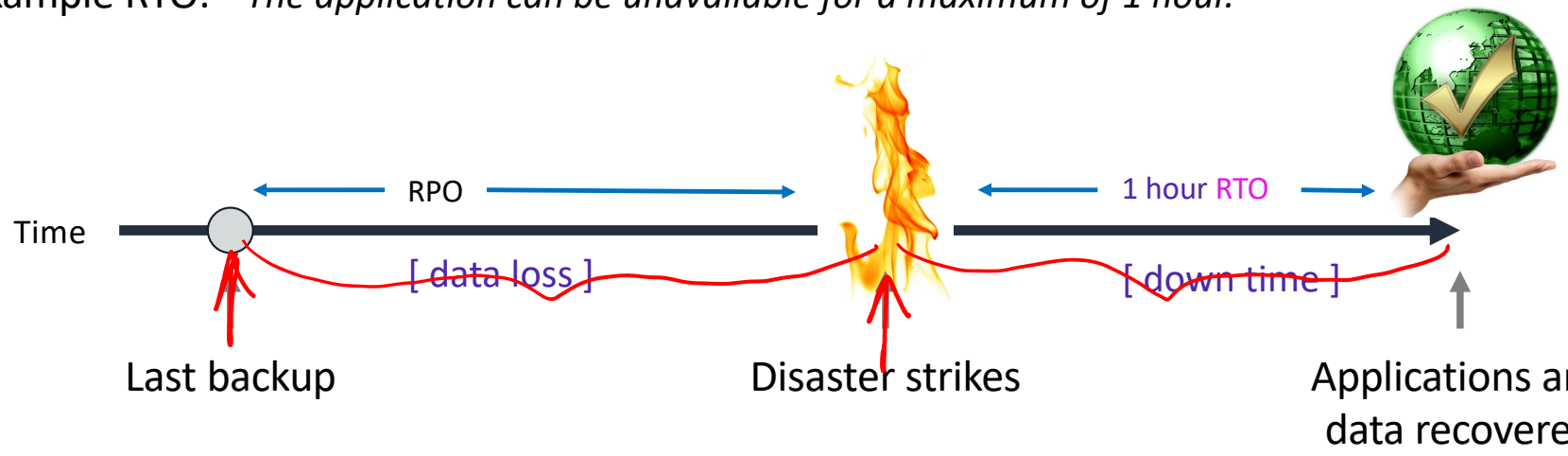


Recovery time objective (RTO)

Recovery time objective (RTO) is the maximum acceptable amount of time after disaster strikes that a business process can remain out of commission.

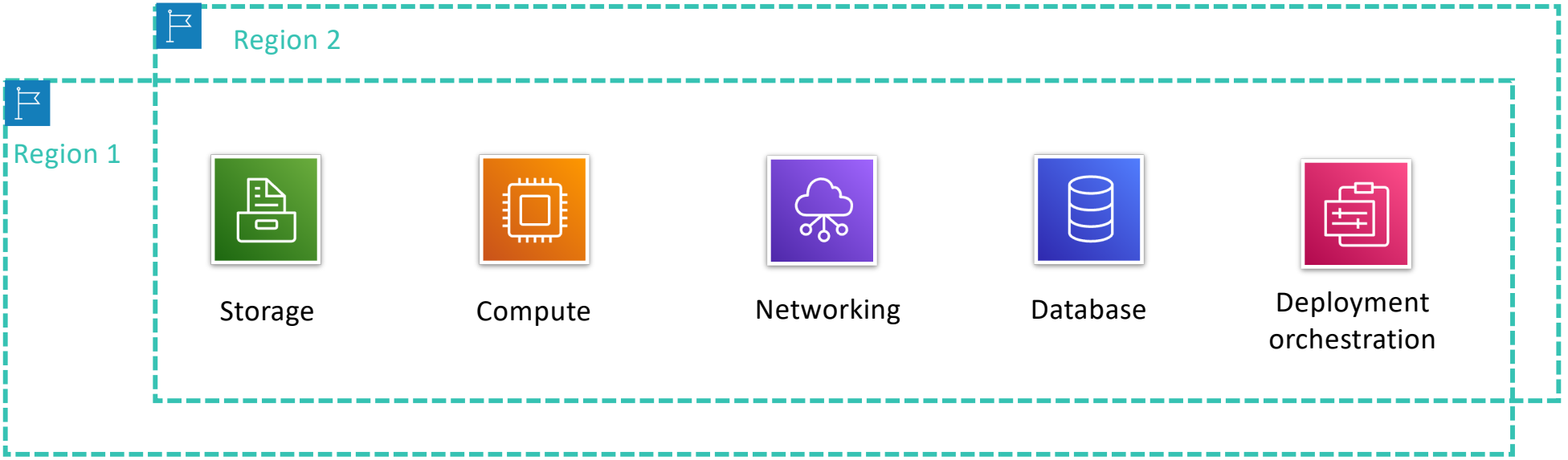
How quickly must your applications and data be recovered?

Example RTO: *The application can be unavailable for a maximum of 1 hour.*



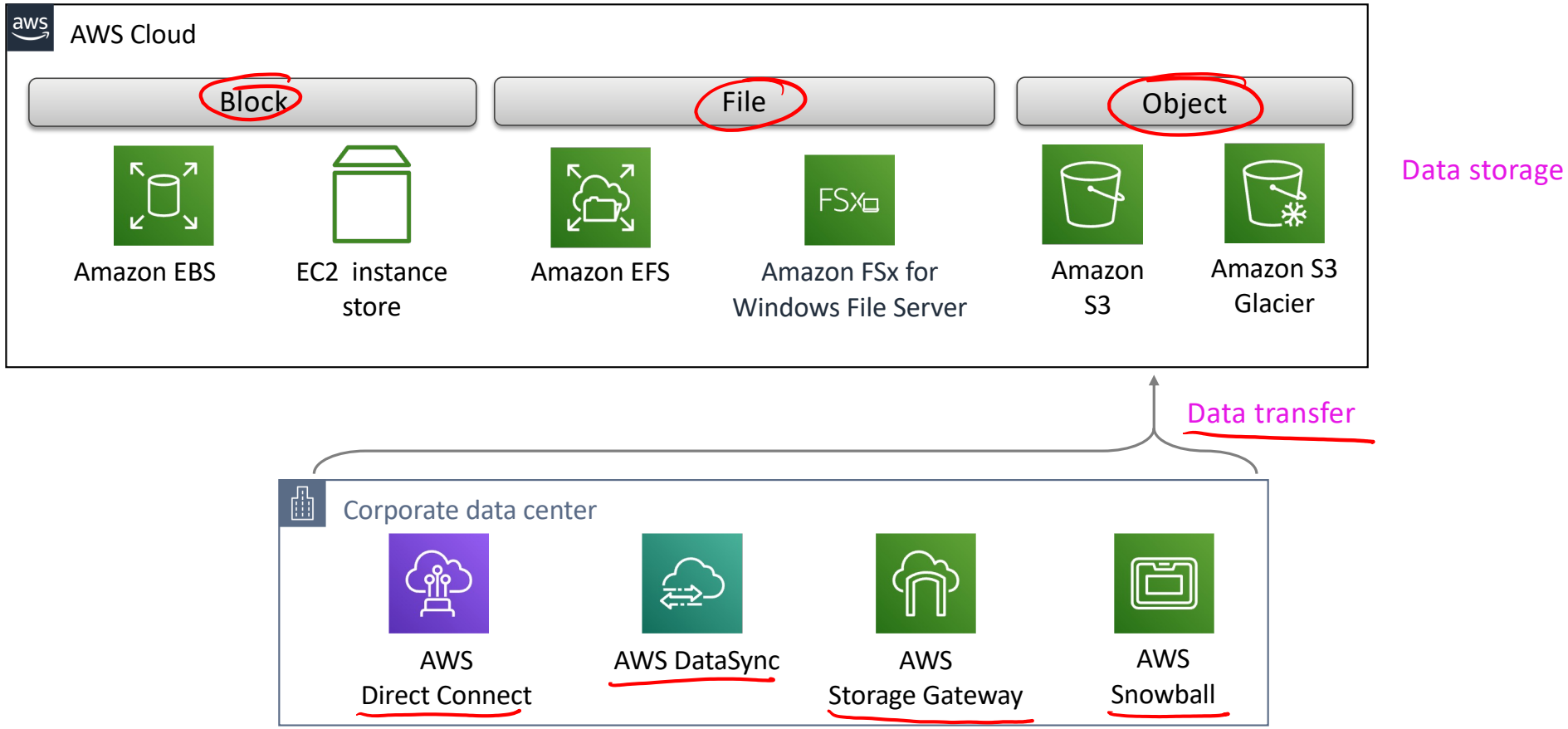
Plan for disaster recovery

Be intentional about where your data is stored and where your applications run.

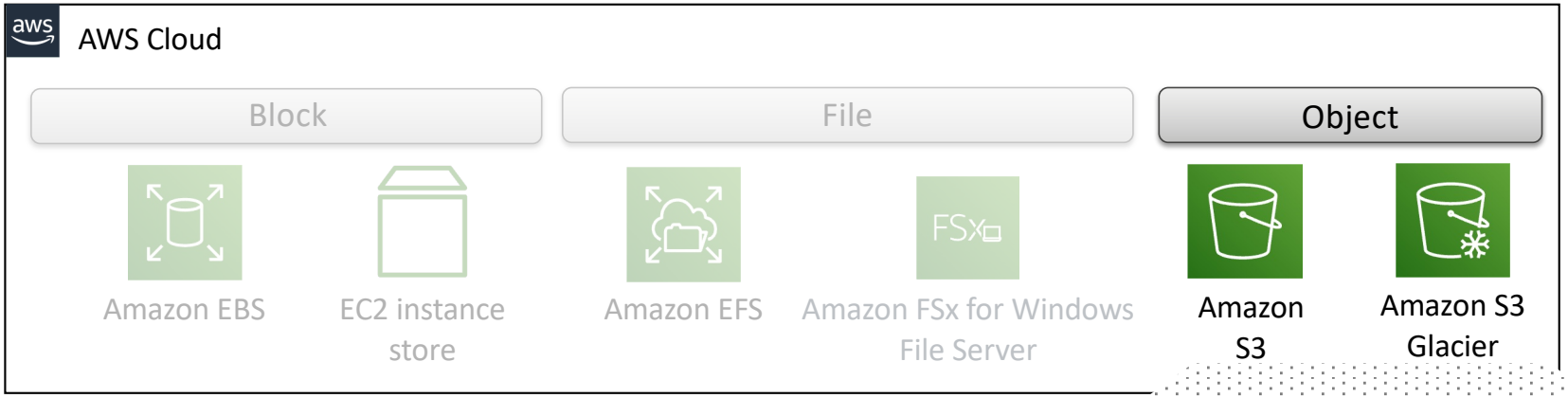


The most robust DR plans span more than one Region.

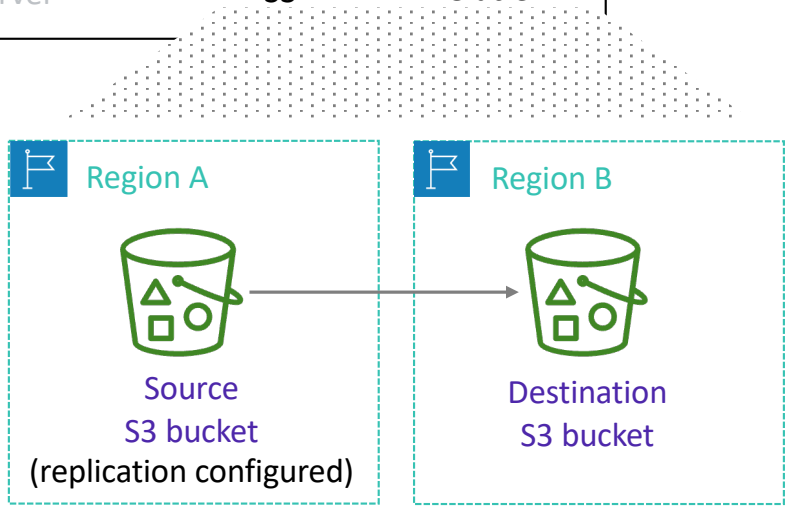
Storage and backup building blocks



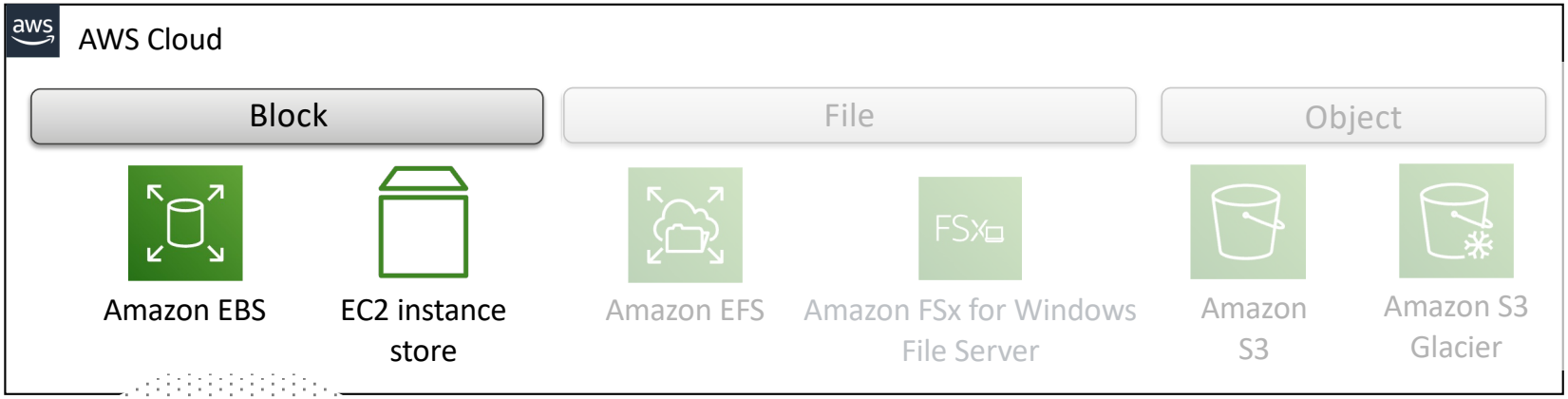
Best practice: S3 Cross-Region Replication

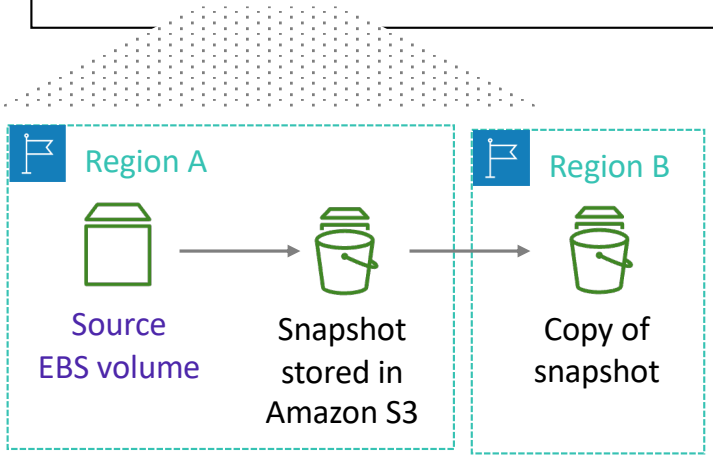


- Most S3 storage classes replicate data across Availability Zones within a single Region
- Configure S3 cross-Region replication for higher-level data security
  - Automatically, asynchronously replicates objects created after you add the replication configuration
  - Can also help meet compliance requirements and reduce latency for users who are accessing objects



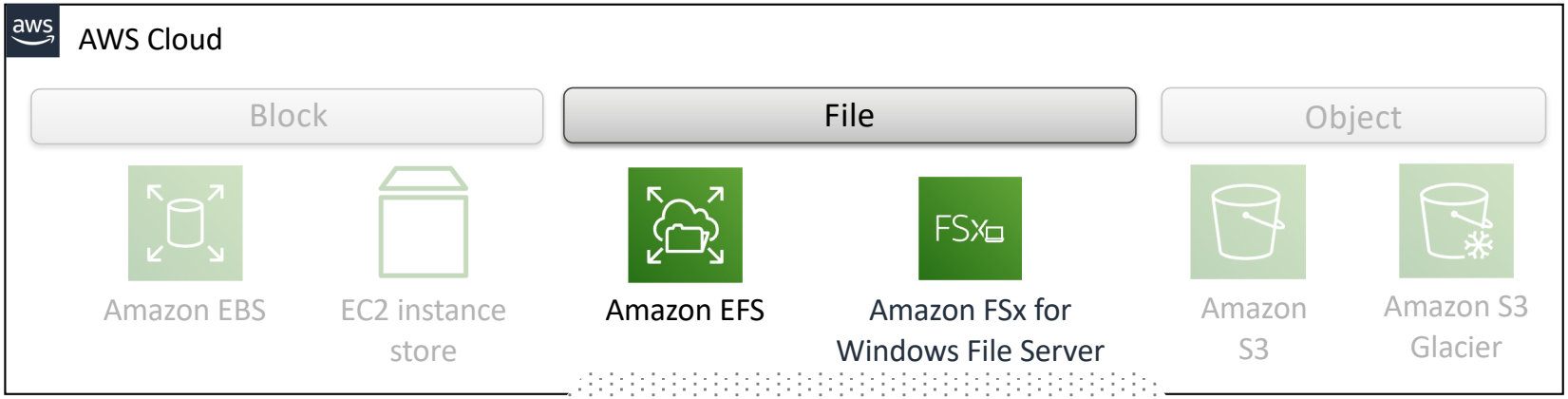
Best practice: EBS volume snapshots



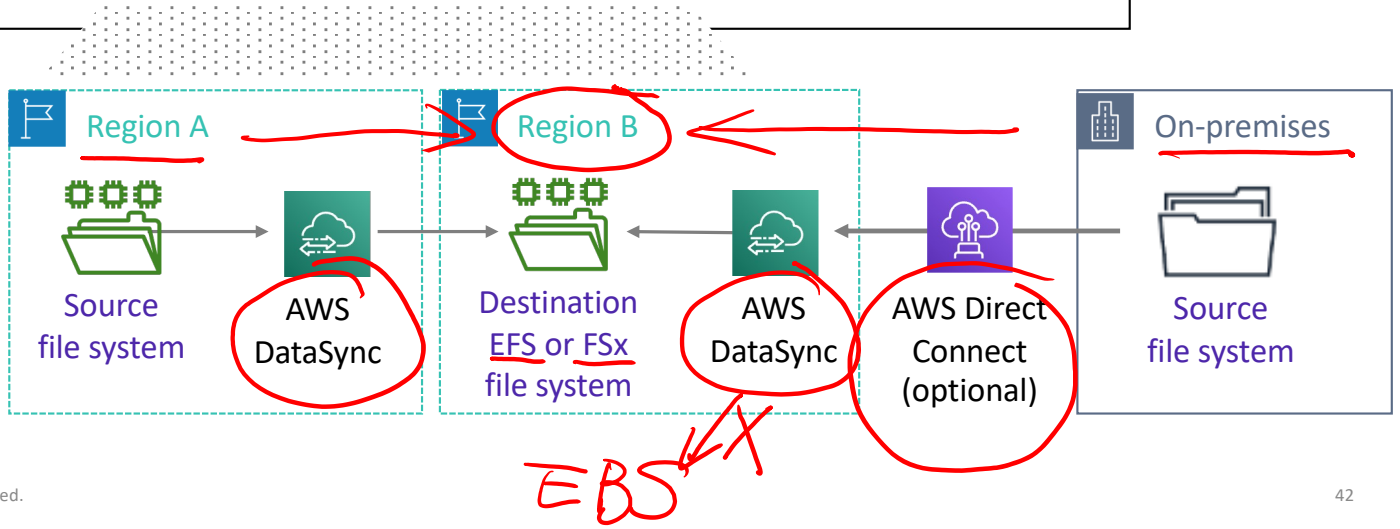


- Create point-in-time snapshots of EBS volumes
  - Snapshots provide incremental backups (they back up the blocks that changed since the previous snapshot)
- Snapshots enable you to restore data to a new EBS volume
- Use Amazon Data Lifecycle Manager to automate the creation, retention, and deletion of snapshots
- You cannot snapshot instance store volumes

Best practice: File system replication

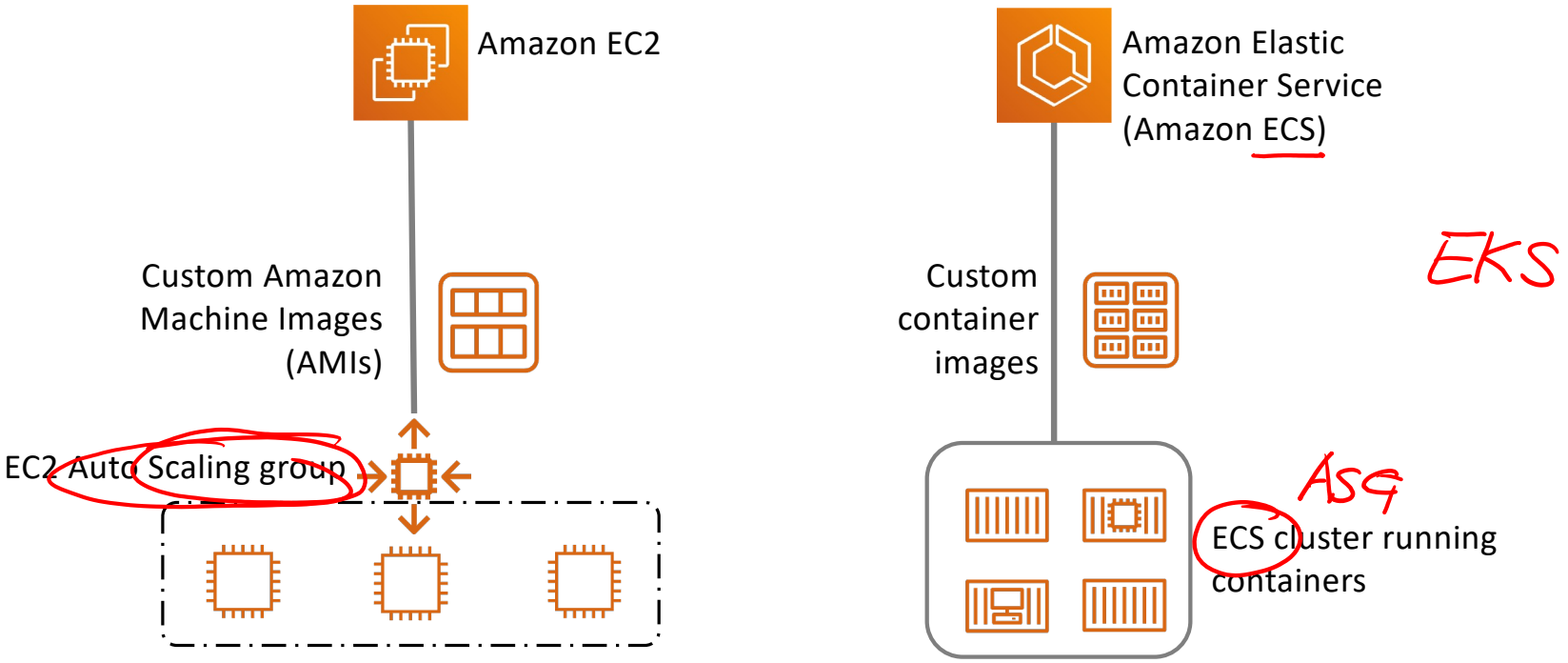


- Replicate EFS or FSx for Windows File Server file systems across Regions
- Replicate on-premises file systems to the cloud



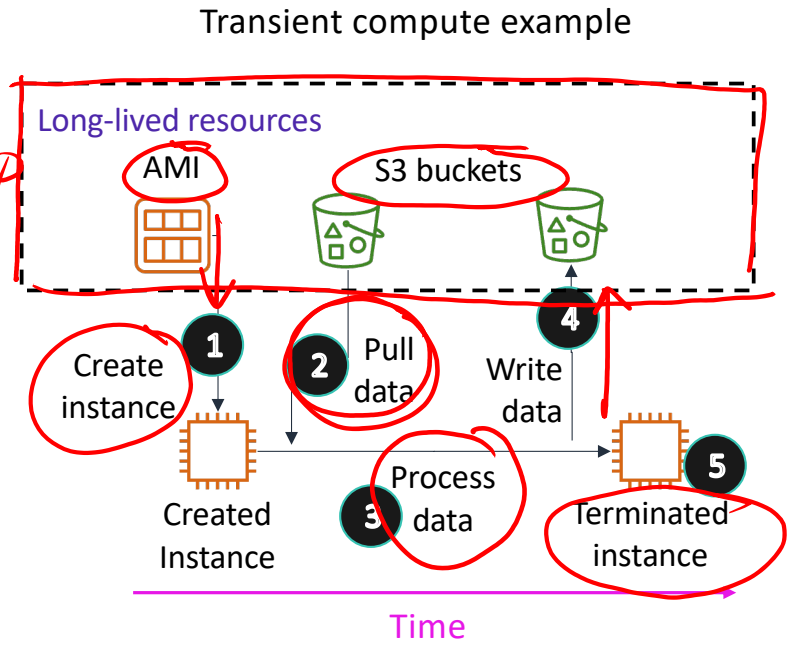
Compute capacity should be quickly recoverable

Obtain and boot new server instances or containers within minutes.



Strategies for compute disaster recovery

- Use the Amazon EC2 snapshot capability for backups
  - Snapshots can be performed manually, or scheduled (for example, by using AWS Lambda)
- Use system or instance level system backups infrequently and as a last resort
  - Drives up the cost of storage that is used quickly
  - Prefer automated rebuild from configuration or code repositories instead
- Cross-region AMI copies
- Cross-region snapshot copies
- Consider transient compute architectures
  - Store essential data off of the instance



Networking: Design for resilience, recovery

Amazon Route 53

- Traffic distribution
- Failover

Elastic Load Balancing

- Load balancing
- Health checks and failover

Amazon Virtual Private Cloud (Amazon VPC)

Extend your existing on-premises network topology to the cloud

AWS Direct Connect

Fast, consistent replication and backups of large on-premises environments to the cloud

Databases: Features that support recovery

Amazon Relational Database Service (Amazon RDS)

- Take snapshot data and save it in a separate Region
- Combine read replicas with Multi-AZ deployments to build a resilient disaster recovery strategy
- Retain automated backups

Amazon DynamoDB

- Back up entire tables in seconds
- Use point-in-time-recovery to continuously back up tables for up to 35 days
- Initiate backups with a single click in the console or a single application programming interface (API) call
- Use Global Tables to build a multi-region, multi-master database that provides fast local performance for massively scaled globally distributed applications

Automation services: Quickly replicate or redeploy environments

AWS CloudFormation

- Use templates to quickly deploy collections of resources as needed
- Duplicate production environments in new Region or VPC in minutes

ECS EKS

AWS Elastic Beanstalk

- Quickly redeploy your entire stack in only a few clicks

AWS OpsWorks

Chef or Puppet

- Automatic host replacement
- Combine it with AWS CloudFormation in the recovery phase
- Provision a new stack that supports the defined RTO

Section 2 key takeaways

- To choose the correct disaster recovery strategy, first identify your recovery point objective (RPO) and recovery time objective (RTO)
- Use features such as S3 Cross-Region Replication, EBS volume snapshots, and Amazon RDS snapshots to protect data
- Use networking features (such as Route 53 failover and Elastic Load Balancing) to improve application availability
- Use automation services (such as AWS CloudFormation) as part of your DR strategy to quickly deploy duplicate environments when needed

Thank you, and Kahoot!

aws academy