Hello, Cloud Gurus, and welcome to this lesson

which is part three of the EC2 section summary,

beginning with Secrets Manager and Parameter Store.

So remember that Secrets Manager

is a secure way of storing secrets

so that they can be accessed

by applications and AWS resources.

It's a great place to store database credentials

as well as API keys.

And the real differentiator is that it supports

automatic rotation of your database passwords and API keys,

whereas Parameter Store is for wider use cases.

For instance, configuration variables

that are not necessarily secrets,

parameters that you define, or for license keys.

And if you did use it

to store database connection information

or database credentials,

then just remember it does not support

database password rotation.

So you cannot use Parameter Store

to rotate your database credentials.

MemoryDB for Redis is an in-memory database.

With ultra fast performance,

you get microsecond read

and single-digit millisecond write capability

and it is massively scalable to over 100 terabytes.

Use cases include high-performance,

large-scale microservices applications,

or online gaming with millions of users

sharing digital assets.

And if you have to choose between MemoryDB and ElastiCache,

then just be aware that MemoryDB can be used

to store your whole data set in memory,

no database required,

so it's a replacement for a traditional database,

whereas ElastiCache is an in-memory cache

for databases like RDS databases,

so it's an add-on to your existing database.

Moving on to RDS Proxy,

remember that RDS Proxy is used to scale your applications

by pooling and sharing database connections

to assist with application scalability

and database efficiency.

RDS Proxy manages the connection,

so your application is pointed towards RDS Proxy

and the RDS database receives information

from the application through RDS Proxy.

And the great thing about RDS Proxy

is that it is serverless and scales automatically

so it scales automatically to your workload

through pooling and sharing database connections.

It preserves application connections during failover,

detect failover, and route requests

to the standby database quickly.

It's deployable over multi-AZ

for protection against infrastructure failure,

and it enables up to 66% faster failover times.

EC2 Image Builder automates the process of creating

and maintaining AMI and container images.

It's a four-step process

whereby you select a base operating system image,

you customize by adding software,

then test the image

and distribute it to your chosen regions.

So the process looks like this.

First of all, you provide the base OS image,

for instance, the Amazon Linux 2 AMI.

Next, you define the software that you want to install,

for instance, .NET, Node.js, Python, et cetera.

And you can also add the latest security updates

and the latest kernel patches, et cetera.

The third step is to run tests on the new image.

For instance, you can test

whether the image boots correctly.

And then finally, distribute the image

to the regions of your choice.

And by default, it's going to be distributed

to the region that you are operating in.

AMI is a regional,

and if you would like to use an existing AMI

in a different region, then you'll need to create a copy

and specify a destination region for the copy.

You can apply encryption during the copying process.

However, you cannot remove encryption

during the copying process.

So if you're making a copy of an encrypted AMI,

then the copy is going to be encrypted.

Well, that is it for this lesson.

Any questions, let me know.

Otherwise, please join me in the next one, thank you.