**AWS Certified Solution Architect – Important Blogs**

Contents

[**Compute** 1](#_Toc43704188)

[**New Amazon EC2 Feature: Bring Your Own Keypair** 1](#_Toc43704189)

[**Automatically updating security groups with lambda function** 1](#_Toc43704190)

[**Best Practice organizing large serverless applications** 1](#_Toc43704191)

[**CDN** 2](#_Toc43704192)

[**Amazon CloudFront Support for Custom Origins** 2](#_Toc43704193)

[**Cloud migration** 2](#_Toc43704194)

[**6 Strategies for Migrating Applications to the Cloud** 2](#_Toc43704195)

# **Compute**

## **New Amazon EC2 Feature: Bring Your Own Keypair**

<https://aws.amazon.com/blogs/aws/new-amazon-ec2-feature-bring-your-own-keypair/>

One can import the public key for use with EC2 instance, this is possible to facilitate

1. Trust – By importing your own keypair you can ensure that you have complete control over your keys.
2. Security -You can be confident that your private key has never been transmitted over the wire.
3. Management of Multiple Regions – You can use the same public key across multiple AWS Regions.

## **Automatically updating security groups with lambda function**

<https://aws.amazon.com/blogs/compute/automating-security-group-updates-with-aws-lambda/>

When autoscaling in triggered, it transmits “lifecycle hook” event, which can be used for triggering a lambda function which can list out all the public IP address of that EC2 instance and then update the security group to add the new Ips to allow it passthrough the security group.

## **Best Practice organizing large serverless applications**

<https://aws.amazon.com/it/blogs/compute/best-practices-for-organizing-larger-serverless-applications/>

* Organized code into multiple repositories – not too small codebase that are difficult to share and not too large codebase that kills reusability. Use SAM to create a deployment pipeline for each of the organized code repositories.
* User AWS service instead of code libraries – for example instead of routing code written in Flask or node.js use API gateway.
* Use separate organization/AWS account for each developers or group of developers, separated from the production environment. Implement CI/CD for deployment of code from/to multiple account.
* Implement CI/CD to manage future releases.

# **CDN**

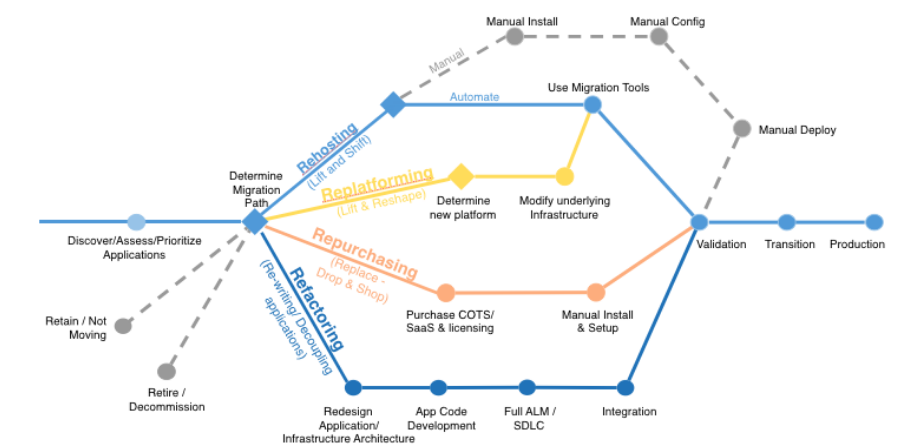
## **Amazon CloudFront Support for Custom Origins**

<https://aws.amazon.com/blogs/aws/amazon-cloudfront-support-for-custom-origins/>

# **Cloud migration**

## **6 Strategies for Migrating Applications to the Cloud**

<https://aws.amazon.com/blogs/enterprise-strategy/6-strategies-for-migrating-applications-to-the-cloud/>



**6 R’s of Cloud Migration are**

1. Rehost – lift and shift
2. Replatforming  – make few changes in the architecture for tangible benefits
3. Repurchasing – moving to a different software
4. Refactoring – architecting changing to the architecture to fit cloud land scape.
5. Retire – get rid of the legacy application.
6. Retain – identify and retain certain software that would make more sense for being on-premises.