**Optimizing application deployment for Lambda**

Resources

* <https://docs.aws.amazon.com/lambda/latest/dg/versioning-aliases.html>
* <https://docs.aws.amazon.com/lambda/latest/dg/lambda-traffic-shifting-using-aliases.html>
* Use versioning to manage your in-production function code in AWS Lambda better
* By using versioning in AWS Lambda, you can publish one or more versions of your Lambda function
* Work with different variations of your Lambda function in development workflow
* After you publish a version, it can't be changed.

*arn:aws:lambda:aws-region:acct-id:function:helloworld:$LATEST*

* Supports creating aliases for each of Lambda function versions
* Conceptually, an AWS Lambda alias is a pointer to a specific Lambda function version
* AWS recommends to use versioning and aliases to deploy Lambda functions when building applications with multiple dependencies and developers involved
* By default, an alias points to a single Lambda function version.
* When the alias is updated to point to a different function version, incoming request traffic in turn instantly points to the updated version.
* To minimize this impact:
* implement the routing-config parameter of the Lambda alias that allows you to point to two different versions of the Lambda function
* dictate what percentage of incoming traffic is sent to each version.

**Optimizing application usage of SQS**

Resources

* <https://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/sqs-delay-queues.html>
* <https://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/sqs-long-polling.html>
* <https://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/sqs-visibility-timeout.html>
* <https://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/sqs-dead-letter-queues.html>
* <https://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/reducing-costs.html>

Key Points

* You can hide messages when they enter a Queue using delay queues. Delay queues are similar to visibility timeouts because both features make messages unavailable to consumers for a specific period of time. The difference between the two is that, for delay queues, a message is hidden when it is first added to queue, whereas for visibility timeouts a message is hidden only after it is consumed from the queue.
* Long polling helps reduce the cost of using Amazon SQS by eliminating the number of empty responses (when there are no messages available for a *ReceiveMessage* request) and false empty responses (when messages are available but aren't included in a response).
* SQS supports dead-letter queues, which other queues (source queues) can target for messages that can't be processed (consumed) successfully.
* Message batching in SQS can help reduce overall costs

**Migrating Code**

Resources

* <https://docs.aws.amazon.com/codecommit/latest/userguide/how-to-migrate-repository.html>
* <https://aws.amazon.com/blogs/devops/setting-up-the-jenkins-plugin-for-aws-codedeploy/>
* <https://docs.aws.amazon.com/codepipeline/latest/userguide/tutorials-four-stage-pipeline.html>
* <https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/create_deploy_docker.html>
* <https://docs.aws.amazon.com/AmazonECS/latest/developerguide/docker-basics.html>
* <https://aws.amazon.com/cloud-migration/>

Key Points

* You can migrate a Git repository to a CodeCommit repository in a number of ways: by cloning it, mirroring it, migrating all or just some of the branches, and so on. You can also migrate local, unversioned content on your computer to CodeCommit.
* You can reuse you investment in Jenkins using the CodeDeploy or CodePipeline jenkin plugin
* If you want to containerize on premise, you can use AWS ECS and ElasticBeanStalk
* Amazon ECS has two modes: Fargate launch type and EC2 launch type. With Fargate launch type, all you have to do is package your application in containers, specify the CPU and memory requirements, define networking and IAM policies, and launch the application. EC2 launch type allows you to have server-level, more granular control over the infrastructure that runs your container applications
* AWS support a number of tools to support server and data migration

- Server Migration Service

- Database Migration Service

- S3 Transfer Acceleration

- Snowball/Snowmobile