MQ on AWS: PoC of high availability using EFS

Arthur Barr |Mar 14 2017 Updated

Amazon [recently declared](https://aws.amazon.com/blogs/aws/amazon-elastic-file-system-production-ready-in-three-regions/) its Elastic File System (EFS) as ready for production.  This enables a shared, networked file system, which (importantly) is replicated between multiple physical data centers (availability zones).  On paper, this makes EFS a good candidate for running MQ in a highly available way.  In this blog entry, I'll take you through our proof of concept (PoC) of running a single IBM MQ queue manager which can be automatically moved between availability zones in the case of a failure.



An EFS file system is scoped to a particular AWS region.  You can create "mount targets" for VPC subnets in different availability zones within that region.  Once the mount target has been created, EC2 instances in those subnets can successfully mount the file system using NFS v4.  You can read more about EFS in the [AWS EFS documentation](https://aws.amazon.com/documentation/efs/).

In our PoC, we used CloudFormation to run a single EC2 instance running MQ, as part of an Auto Scaling Group of one server.  This ensures that if the MQ instance is determined to be unhealthy, then AWS will destroy the instance and replace it with a new one, connected back to the same file system.  You can span multiple availability zones with an Auto Scaling Group.  The Auto Scaling Group has a policy applied to ensure that there are only ever 0 or 1 instances available: during an update to the CloudFormation stack, the existing instance is always terminated before starting a new one.

When the MQ EC2 instance first boots, it mounts the file system as /var/mqm, and adds a rule to /etc/fstab to ensure that it gets mounted again if the instance were re-booted.  If there's already data for a queue manager in the file system, then it sets up a systemd service to run the queue manager, and creates a dependency on the mount point being available.  This systemd service will also ensure that the queue manager is restarted upon re-boot.

We also used an Elastic Load Balancer (ELB) to provide a single TCP/IP endpoint for MQ client applications to connect to.  In some ways, an ELB is overkill here - alternatives include using an Elastic IP address which can be re-bound to a different EC2 instance, or using Route 53 to handle it via DNS.  With the ELB, we can also add a health check, to ensure that MQ is listening on port 1414, and mark the instance as unhealthy if not.  In addition, we added a health check to the instance which periodically runs dspmq to check that the queue manager is running.  If it is ever found to be down, then the AWS command line interface is used to mark the instance as unhealthy.  Any unhealthy instances will be terminated and replaced by the Auto Scaling Group.

**Reproducing our PoC**

If you'd like to try this out for yourself, then you can find all the code on [GitHub](https://github.com/ibm-messaging/mq-aws/tree/master/efs).  The PoC requires [Packer](https://packer.io/) to be installed on your local laptop or workstation.

1. Download the files from the Gist
2. Run packer build packer-mq-aws.json to build an AMI in the us-west-2 (Oregon) region.  If you'd like to use a different region, you can edit the JSON file, making sure to also replace the source\_ami with the equivalent [RHEL 7.2 AMI in your chosen region](https://aws.amazon.com/marketplace/pp/B019NS7T5I).  Note that, at the time of writing, EFS is not available in all regions.
3. Create a stack using the CloudFormation template [cloudformation-mq-efs.template](https://github.com/ibm-messaging/mq-aws/blob/master/efs/cloudformation-mq-efs.template).  This can be done through the AWS web console, or via the command line if you have the AWS CLI tools installed. For example, the following command line runs the CloudFormation stack in us-west-2 (Oregon).  Be sure to replace or set the variables MY\_KEY (to the name of an [EC2 key pair](http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-key-pairs.html#having-ec2-create-your-key-pair) for SSH) and MY\_AMI as well:

aws cloudformation create-stack --stack-name mqdev-efs --template-body file://./cloudformation-mq-efs.template --parameters ParameterKey=KeyName,ParameterValue=${MY\_KEY} ParameterKey=QueueManagerName,ParameterValue=mqdev ParameterKey=AMI,ParameterValue=${MY\_AMI} ParameterKey=AvailabilityZone1,ParameterValue=us-west-2a ParameterKey=AvailabilityZone2,ParameterValue=us-west-2b --capabilities CAPABILITY\_IAM --region us-west-2

The CloudFormation template includes many resources, including a VPC network, subnets, an Internet Gateway, the Auto Scaling Group and Launch Configuration, and an IAM role to enable the EC2 instances to report their health.

If you inspect the created resources (Edit: aws ec2 describe-images --owners self), you will see an Auto Scaling Group with a single instance.  You have several options to test out the fail-over:

1. SSH into the instance and stop/kill the MQ queue manager (with user ec2-user).  This will cause the local health-checking script to invoke the AWS CLI to mark the instance as unhealthy.
2. Terminate the instance entirely.
3. Mark the instance as unhealthy, either in the web console or on the command line.

Once the instance is marked as unhealthy, the AWS Auto Scaling Group will create a new one.  Note that as the instance is in an otherwise-healthy availability zone, the instance may be re-created in the same zone.  If you keep trying though, eventually, AWS should randomly assign the instance to the secondary zone.

Note that if you want to connect to the queue manager using an MQ client, the supplied scripts set up a PASSWORD.SVRCONN channel, with a user of johndoe, and a password of passw0rd.  It is, of course, recommended that you (at the very least) change this password, which can be found in the [configure-mq-aws.sh](https://github.com/ibm-messaging/mq-aws/blob/master/efs/configure-mq-aws.sh) script.

**Next steps and conclusion**

This is just a PoC, but so far, EFS seems to provide the right characteristics for running MQ.  There is clearly more to do here, including comprehensive testing of fail-over under load, and performance testing.  With this particular set up, the fail-over between zones seems to take a between one and three minutes, but that's nothing to do with EFS, and everything to do with the fact that we're creating a brand new EC2 instance when the old one fails - alternative solutions might use multi-instance queue managers, or an otherwise pre-created EC2 instance.  There's also some scope for better tuning the health check grace periods, to ensure things return to "healthy" status as quickly as possible.

A fail-over time for a single-instance queue manager measuring in a small number of minutes may well be enough for many people.  Either way, with EFS it's relatively easy to set up high availability across multiple availability zones without having to run your own replicated storage subsystem, which is definitely a positive thing.