An Honest Review of AWS Managed Apache Kafka: Amazon MSK

Stéphane Maarek Follow
Nov 30, 2018 · 7 min read

AWS just released a service for a Managed Apache Kafka at Re-Invent 2018. I jumped on it to get my hands dirty with the product. Is it worth the hype? Should you use it? Should other vendors get worried? Here is my review!

. . .

TLDR: While I recognize Amazon MSK is a public preview and it demonstrates good performance, as of today I find it lacks major features, including proper security, and that makes MSK something I cannot currently recommend.

. . .

I should be happy… but I'm not

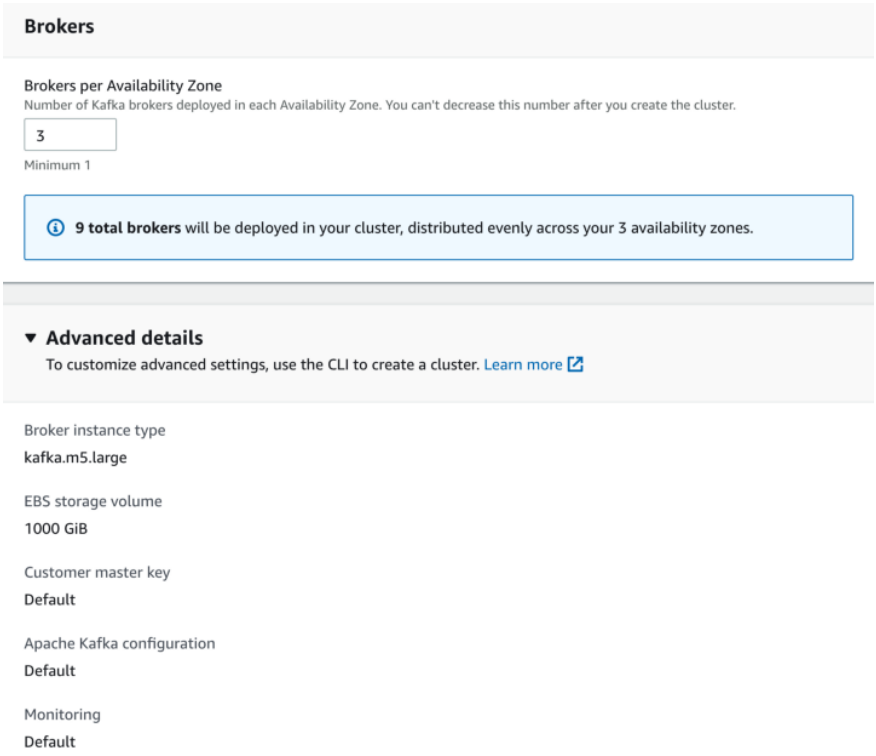
For me, the more people use Apache Kafka, the more business I get. As I teach Apache Kafka online on Udemy (links at https://kafka-tutorials.com/), the prospect of having an entire user base from AWS wanting to learn Apache Kafka is exciting! And as an Apache Kafka

consultant, it's always more fun to spend time deploying data pipelines than deploying infrastructure.

Unfortunately what AWS released today misses the mark. I think it's reminiscent of managed services of open source software in AWS overall: they're released early and lack features that I think should be MVP. In my opinion this will deter future users.

What you get

Let's start talking about the positive aspects. In a few clicks and 15 to 30 minutes of wait time, you get a fully replicated & multi-AZ Kafka cluster. It comes with Zookeeper managed as well, and data storage is backed by EBS volumes. You pay double the price for your EC2 instances, but that's the price to pay for a managed service. The EBS costs are the same as if you used an EBS volume yourself.



can't set the broker type and security group through the UI... seriously?

NB: you can only have a brokers that is a multiple of the number of AZ you're deploying to… which is a good thing!

From there, you get a bootstrap server URL and a Zookeeper DNS name you can hit and get started with Kafka. To obtain the URLs, you have to use the CLI…

```
$ aws kafka list-clusters
{
```

```
    "ClusterInfoList": [
        {
            "BrokerNodeGroupInfo": {
                "BrokerAZDistribution": "DEFAULT",
                "ClientSubnets": [
                    "subnet-c69251b1",
                    "subnet-07816c5e",
                    "subnet-fc512699"
                ],
                "InstanceType": "kafka.m5.large",
                "StorageInfo": {
                    "EbsStorageInfo": {
                        "VolumeSize": 1000
                    }
                }
            },
            "ClusterArn": "arn:aws:kafka:us-east-
1:160803060715:cluster/test/9d2be72e-0d3b-4a2a-abbd-
d8708ec4dca8-3",
            "ClusterName": "test",
            "CreationTime": "2018-11-30T08:21:36.824Z",
            "CurrentBrokerSoftwareInfo": {
                "KafkaVersion": "1.1.1"
            },
            "CurrentVersion": "K13V1IB3VIYZZH",
            "EncryptionInfo": {
                "EncryptionAtRest": {
                    "DataVolumeKMSKeyId": "arn:aws:kms:us-
east-1:160803060715:key/c8a78efb-4daa-48ee-9fe8-
26d40c274aaf"
                }
            },
            "EnhancedMonitoring": "DEFAULT",
            "NumberOfBrokerNodes": 3,
            "State": "ACTIVE",
            "ZookeeperConnectString":
"172.31.31.89:2181,172.31.9.136:2181,172.31.73.166:2181"
        }
    ]
}
```

```
$ aws kafka get-bootstrap-brokers --cluster-arn
arn:aws:kafka:us-east-1:160803060715:cluster/test/9d2be72e-
0d3b-4a2a-abbd-d8708ec4dca8-3
{
    "BootstrapBrokerString":
"172.31.72.155:9092,172.31.10.218:9092,172.31.18.106:9092"
}
```

In terms of Kafka version you get… 1.1.1. It's really old, we have 2.1 and 2.0.1 available nowadays… oh well. All the Kafka clients you already have should be working, that includes Kafka Connect, Kafka Streams, monitoring solutions that connect to Kafka, etc.

Monitoring wise, we can view the metrics in CloudWatch, which is I guess a welcome addition. The "default" monitoring level is free, and you have to pay extra for "per broker" and "per topics per broker". It's kind of ridiculous, because everyone needs "per topics per broker". Just factor that into the cost of the broker and make it free for everyone… ugh. The list of metrics is good though and I'm happy they're exposing some of the most important ones: https://docs.aws.amazon.com/msk/latest/developerguide/monitoring.html#metrics-details

Finally, because you procure EC2 instances, you're free to use any network bandwidth you like. It's I think a big differentiator in AWS' offering versus other managed Kafka Providers.

I've created a cluster and I did a quick performance test with 3 brokers m5.large and one machine m5a.large as a client. Performance wise the

cluster seems solid, I was able to obtain a peak throughput at 70MB/sec.

```
$ kafka-topics.sh — zookeeper
172.31.31.89:2181,172.31.9.136:2181,172.31.73.166:2181 —
create — topic test — replication-factor 3 — partitions 12
Created topic “test”.
```

```
$ kafka-producer-perf-test.sh — topic test — throughput
10000 — num-records 10000000000 — producer-props acks=all
linger.ms=10 batch.size=65536
bootstrap.servers=172.31.72.155:9092,172.31.10.218:9092,172.
31.18.106:9092 — record-size 1000
49912 records sent, 9968.4 records/sec (9.51 MB/sec), 17.4
ms avg latency, 236.0 max latency.
50076 records sent, 10011.2 records/sec (9.55 MB/sec), 10.6
ms avg latency, 35.0 max latency.
49994 records sent, 9994.8 records/sec (9.53 MB/sec), 10.0
ms avg latency, 29.0 max latency.
50074 records sent, 10014.8 records/sec (9.55 MB/sec), 9.9
ms avg latency, 33.0 max latency.
```

```
$ kafka-producer-perf-test.sh — topic test — throughput
50000 — num-records 10000000000 — producer-props acks=all
linger.ms=10 batch.size=65536
bootstrap.servers=172.31.72.155:9092,172.31.10.218:9092,172.
31.18.106:9092 — record-size 1000
249251 records sent, 49820.3 records/sec (47.51 MB/sec),
26.0 ms avg latency, 226.0 max latency.
250275 records sent, 50055.0 records/sec (47.74 MB/sec),
13.7 ms avg latency, 34.0 max latency.
250079 records sent, 49985.8 records/sec (47.67 MB/sec),
13.6 ms avg latency, 35.0 max latency.
```

```
$ kafka-producer-perf-test.sh — topic test — throughput
100000 — num-records 10000000000 — producer-props acks=all
linger.ms=10 batch.size=65536
```

```
bootstrap.servers=172.31.72.155:9092,172.31.10.218:9092,172.
31.18.106:9092 — record-size 1000
369975 records sent, 73906.3 records/sec (70.48 MB/sec),
343.9 ms avg latency, 1112.0 max latency.
351592 records sent, 70290.3 records/sec (67.03 MB/sec),
444.9 ms avg latency, 1402.0 max latency.
351236 records sent, 70233.2 records/sec (66.98 MB/sec),
453.6 ms avg latency, 1359.0 max latency.
354300 records sent, 70845.8 records/sec (67.56 MB/sec),
447.4 ms avg latency, 1365.0 max latency.
```

. . .

What we don't get

First, I want address the elephant in the room: security.

To me, MSK is not secure

I think the main reason why anyone would use MSK is for security. Okay great, I have an Apache Kafka cluster in own cloud, so I'm the only one to have access to it and that's a big plus. What did AWS do? Well, it's backed by KMS for encrypting EBS volumes, and you don't get SSH access to your data which is good. That gives you a twist on "at-rest encryption" for your data.

But since Kafka v0.9, which is over 3 years old, we've had proper Kafka security. It's still being improved at each Kafka release with new security mechanism. We have two components today in Kafka Security:

- SSL encryption in flight: you don't get that with MSK (for now). https://aws.amazon.com/msk/faqs/. That means all the data flows around your networks fully un-encrypted. Route53 and ACM integration will hopefully come later… but right now you're left exposed.

- Authentication and Authorization: so here is my biggest problem: you have created a MSK cluster, and you think it's secure because the marketing landing page told you it was secure. Well, not at all. It uses absolutely NO security mechanism, so anyone can create / read / write / delete data on any topics. Oh, and anyone can alter Zookeeper too. I mean come on, every single managed Kafka provider offers security. It comes in various form: SASL/GSSAPI, SASL/OAUTH, SASL/PLAIN, SASL/SCRAM, and AWS didn't implement any of those. Actually, when you think AWS and security, you think IAM. SASL/IAM would be the killer feature. Authentication and Authorization from within AWS IAM, the dream. Well… not going to happen anytime soon.

So while you get all the fancy network and service control security, your actual Kafka cluster is as opened as it could be. That means that any security control of the data you've been working so hard to do in your current systems is completely lost in MSK. To me, that's a deal breaker for all my consulting clients.

Let me nuance my points: this is a preview of MSK, and I'm sure that somewhere in their backlog, AWS does have a big card to implement and release some sort of real security. But you only get one chance to make an impression when you release a service, and you better something release somewhat complete. AWS missed the mark here,

and for a long time, I will think as MSK as non-secure. Even if one day it becomes “secure”, it shows the lack of commitment of AWS to support a secure-first Kafka, and that reflects on the mindset of the product: marketing over capabilities.

The other things we don’t get

Besides security, here’s a laundry list of the things I miss:

- managed Schema Registry
- ability to drop in jars in Kafka (for metrics reporter, specific classes, java agents)
- point and click mirror maker / replication
- point and click backup to S3
- point and click linking of Kinesis to Kafka
- managed Kafka Connect
- control over Kafka configs (although you could hack this using the kafka-configs command… you’re welcome AWS)
- scaling of cluster
- topic management from the console
- management of pretty much everything through the console
- support on your cluster—this is a big unknown for me as whether AWS support will be able to support your Kafka cluster. In my

> experience supporting Kafka clusters is not easy, and this is something other managed Kafka vendors may actually be good at. Another differentiator

. . .

AWS and the open source world

AWS loves to take products from the open source world and making them paid as a service. Elastic Search and Envoy are two examples, and now we have Apache Kafka. The problem is that historically, AWS doesn't give much back to the projects it makes a pile of cash from. And that's a big problem for these projects, because they live from community support.

Please prove me wrong AWS:

To be fair, AWS has started to open source something really big this year, FireCracker, which may indicate a shift in mindset… time will tell.

Closing thoughts

I think increased competition in the managed Kafka space is great. It's filled with great actors such as Confluent, Aiven, CloudKarafka, Instaclustr, Landoop, Heroku and probably others I'm missing. AWS surely is adding a lot of pressure, but right now Amazon MSK is so bare-minimum that I cannot recommend it. You still end up managing a lot, which wasn't something you signed up for in the first place, was it?

—

If you're still in with MSK, well you should know how Kafka works. My tutorials at https://kafka-tutorials.com should help! Happy learning :)