**AMNMToken**

* **Purpose: -**AMNMToken is to be used for authenticating AM –NM communication. This token will be issued by RM but will be validated at NM side.
* **Contents: -**

1. appId (ApplicationId)
2. nodeHostAddr (String - "<ip-addr>:<http-port>”)
3. appSubmitter (String – Application submitter user name)
4. MastKeyId (int)

* **Validity of the token: -**RM will issue this token when first time AM receives a new container on underlying NM. This token will remain valid for AM to communicate with NM until

1. AM doesn’t receive any new container on the same NM
2. AM receives new container on same NM but underlying shared master key between RM and NM doesn’t change (rollover).

This token will be reissued by RM when underlying key has rolled over and AM receives new container on same NM. RM remembers that it has issued token to AM for specific NM until key rolls over.

NM when receives AMNMToken; remembers it until either application finishes (for which it will remove all the entries) or AM sends new AMNMToken (this may be because key rolled over – In this case NM will validate the key and will refresh its cache).

* **RM side changes: -** RM will now have a new secret token manager (in both secured and unsecured environment). Master key will be shared with all the NM during RM-NM heartbeat. Its master key will be rolled over after rolling interval specified in yarn configuration. Once the new key is generated it will be distributed to all NM via RM-NM heartbeat. After activation period (minimum activation period should be twice the RM-NM-expiry-interval) RM will replace older key with new key. All new AMNMTokens will be generated with this key.
* **AM side changes: -** AM will now receive AMNMToken as a part of AM-NM protocol (allocate) if it receives this token then it will have to remember this token for all future communication with underlying NM. It will have to replace this token (for long running jobs or jobs running across key rollover) with the new toke issued by RM. AM can now distribute this token to either other container or may be an external service who wants to communicate with NM for monitoring container status.
* **NM side changes: -**NM will now authenticate all the communication with AM using this token. NM will internally cache these tokens per AM for future authentication. NM will validate this token based on

1. If AMNMToken is using current MasterKeyId then it will update its cache if that contains none or an older key. (Either AMNMToken got updated or AM is communicating with NM for the first time).
2. If AMNMToken is using older MasterKeyId (after key rolled over on RM) then this call will still be authenticated for a time twice the container token validation period. After this period AMNMTokens will be rejected if they are using older key for AM-NM communication for the very first time (i.e. cache doesn’t contain keyId for this application).
3. If AMNMToken is using MasterKeyId as per the cache value (in which case it is a long running job) then it will be authenticated using this value.

* **Impact on ContainerToken usage: -**

1. ContainerToken was earlier used for validating startContainer calls on NM. Now AMNMToken will replace this. Additionally AMNMToken will authenticate stopContainer and getContainerStatus calls.
2. ContainerToken will be used only for authorization during startContainer call.

* **Advantages: -**

1. Single AMNMToken for AM to communicate with NM in both secured and unsecured environment (Earlier it was per container).
2. Now all AM-NM calls can be authenticated. AM can distribute this to may be other containers who wants to monitor this NM’s container progress.
3. RM doesn’t issue AM new token on every container request. So ideally for most of the non-long running jobs it will be issued once or twice (if rolled over).
4. Now we will be issuing the AM new AMNMToken after key rollover only if it gets new container on underlying NM thereby we don’t have spike at key roll over. (If we wanted to renew all the AMNMTokens then there could have been a spike there).