**ID Generation Techniques Tweaks**

1. Twitter Snowflake – Original id 64 Bits

41 bits timestamp

10 bits Instance id

12 bits Counter

**Modified Approach1**--: Since Pod id/ Instance id is not known and expecting Kubernetes deployment type is stateful set where every pod will have unique id in the ordinal order

For eg Container 1 name of app – **web**

Replica – 3

Possible id will be like web-01, web-02, web-03 (to be confirm)

Name of app Container2– translationapp

Replica 2

Possible id will be like translationapp-04, translationapp-05

If above conditions true we can modify the twitter snowflake approach in following

41 bits -🡪 timestamp

10 bits 🡪 Stateful set id (will extract 01 from id received through env (web 01)) 2^10=1024

12bits 🡪 sequence counter 2^12=4096

The above is calculated using formula

48bit timestamp<<22

Statefulsetid<<12

Timestamp| Statefulsetid |Sequence where | is OR operations in binary

Resultant id-🡪 **6981492114374791169-web01**

(Here Statefulsetid id is getting used in << and OR operations and also as identifier (web01))

**Modified Approach2**-: If the deployment type stateful set gives ordinal id for only the containers and its replica

Foreg -: Container 1 name of app – **web**

Replica – 3

Possible id will be like web-01, web-02, web-03 (to be confirm)

Name of app Container2– translationapp

Replica 2

Possible id will be like translationapp-01, translationapp-02

If above conditions true we can modify the twitter snowflake approach in following

41 bits -🡪 timestamp

22bits 🡪 sequence counter

The above is calculated using formula

48bit timestamp<<22 (left shift operation)

Timestamp |Sequence where | is OR operations in binary

Resultant id-🡪 **6981492114374787073-web01 or 69814921 web0114374787073**

(Here Statefulsetid id not getting used as << and OR operation, its is only getting used as identifier)

1. **UUID (Universal Unique Identifier) and Types**

UUID1 – Generate UUID using a Host MAC address, sequence number and the current time. This version uses the IEEE 802 MAC addresses.

UUID3 and UUID 5 uses cryptographic hashing and application-provided text strings to generate UUID. UUID 3 uses MD5 hashing, and UUID 5 uses SHA-1 hashing.

**Aproach3-** Using 128 bits UUID1 using RFC 4122 algorithm

generate a version 1, 3, 4, and 5 UUIDs as specified in RFC 4122

UUID 1 --: “**time\_low** “-” **time\_mid** “-“**time\_high\_and\_version** ” -“**clock\_seq\_hi\_and\_reserved\_And\_clock\_seq\_low**“-” **Node**.

Where Node /mac address can be replaced by a Statefulset id received from env and the id can be represented in hexa or alphanumeric fields

Foreg The generated uuid 1 is eeccda1e-4086-11ed-bdbe-010101010000 in hexadecimal

Where original fields are (4006402590, 16518, 4589, 189, 190, 1103823437824)

****

We can create a hashable form of id or we can have INT form of UUID like

0f634a04-4088-11ed-8cc3-010101010000

258165252-16520-4589-140195-1103823437824 (Non hexa)