Day - 5

30 May 2025 10:10

1. Alert.
2. AWS integration with DT
3. Database in DT
4. Access Management.

Dynatrace certification:-

1. Dynatrace Associate certification.
2. Dynatrace Professional Certification
3. Dynatrace Master Certification

1. Associate Certification:- USD $200
   60 min. — 70% Passing
   1. Foundational level
   2. Architecture, Core Components & Basic Monitoring capabilities.

2. Professional Certification:- $250 USD
   1. Deploying
   2. Configuration.
   3. Complex Monitoring Service

   75% → Passing Score
   Time → 90 minutes.

3. Master Certification:-
   1. Expert level Knowledge is required
   2. Implementation, Manage Comprehensive DT Solution.

   Merge → Practical Assessment

Anamoly detection and Alert Mechanism:-

Anamoly Detection:- limited Pre defined templates

Davis (Dynatrace AI Causation Engine) → Automatic Anamoly Detection.

Davis AI Working:-

1. Baselining:- RT, CPU, Throughput. → establish a baseline of Normal behaviour.
2. Detection:- Deviation from these baselines.
3. Causation Analysis:- Pinpoint root cause of Problems.
4. Noise Reduction:- Combine multiple Anamolies into a single problem, reduce alert fatigue.

Anamolie Detected Type:-

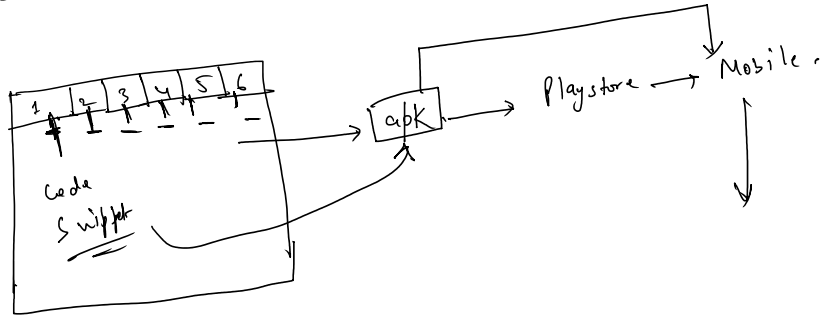
1. Slowdown in Service
2. Increased failure rate or error rate.

... or Container resource saturation.

... breaches.

① Slowdown.
② Increased failure rate.
③ Host or Container resource saturation.
④ Unusual deployment behavior.
⑤ Synthetic monitors failed or threshold breaches.

DT Server needs 7day of historical data, to create the baseline.



Alert Mechanism in Dynatrace:-

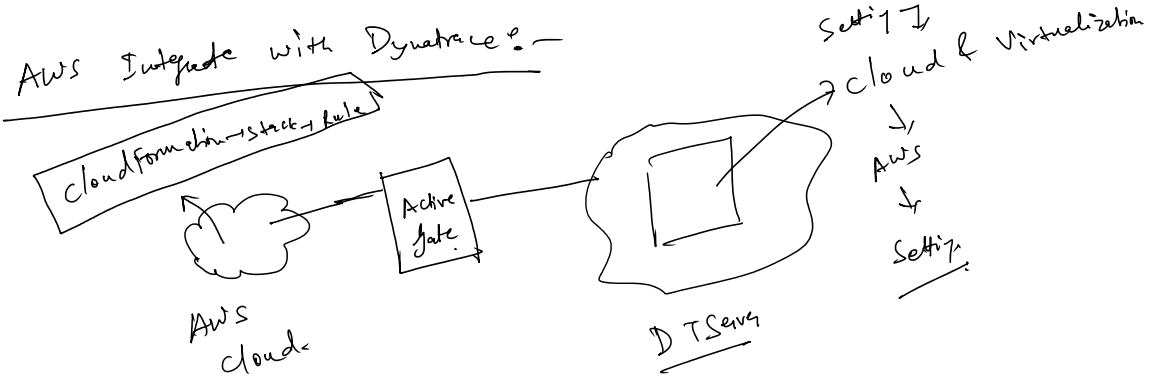
① Multiple Anamolies into a Single Problem Alert

Root cause
Start/end time
Anamoly type (degradation, failure)
Impacted service.

② Alert configuration options

① Custom Metric option.
② Log pattern event
③ Availability event (eg. one Agent, Synthetic failures)

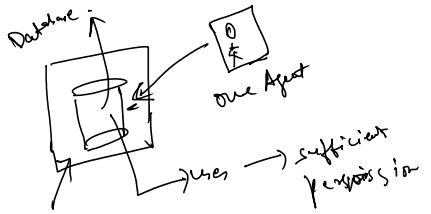
AWS Integrate with Dynatrace:-



Database

Managed service | SAAS environment

Managed service

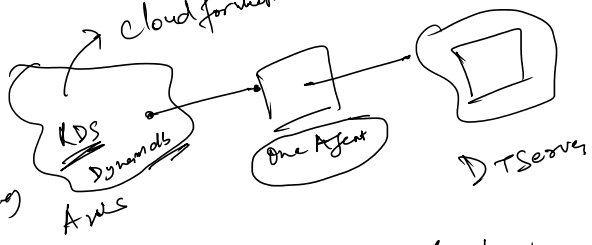


SAAS environment

→ RDS (AWS)
Cosmosdb (Azure)

No need to manage the VM.
Only need to access the database.

→ cloudformation → stack



RDS (AWS)
cosmosdb MySQL (Azure)

MySQL → tabular structured data. Proper Row & Column

NoSQL → No Defined schema.

DynamoDB (AWS)
cosmosdb-NoSQL (Azure) Json

Eden space →
Tenured gen →
Thread →
Heap Memory →
→ JVM memory areas

JVM Memory

Heap Memory (Object Allocation)

Non-Heap Memory (class Metadata, codecache)

Thread stack (one per thread)

Native memory (JVM internal processes its)

Pre-Java 8 (Hotspot cl

Java 8+ (Replaced

clas

Thread local Memory

1. Each thread has its o
2. Method call frames, l
3. Not GC → de

Heap memory — Store Java object & class instances.
GC — Garbage collector

young generation (young gen)
(New object is Allocated)

Old generation (Tenured gen)
(object long-lived stored here)
Several round in young GC
Full GC is triggered when this space fills.

1. Ca
2. St
3.

ass metaclass)
by metaspace)

s def.
tic method
Compiled code
Cache.

n stack
ocal variable, & return values.
troyed when thread
ermi nates.

ll stack (execution
traces)

lack frame
Reference to heap
object.

(New object is ...

Eden space
(All new object are created)
Minor GC

Survivor space

S0
(Object survived a few GCs in Eden are copied here)

S1

Full GC is triggered when space fills.
(Ex: Cached data, static objects, Abandoned Session.)

3

object is created → Eden
if it survive GC → moved to Survivor 0 (S0)
Survive Again → moved to Survivor1 (S1)
After a threshold → moved to old gen.

Summary :-

Eden space → Minor GC → Very fast, freq. col
Survivor → Minor GC → filters long-living obj
old generation → Full GC → Infrequent, expensive,
Metaspace → Manual/auto → collected with cla
Unloading

Reference to heap object.

lection

ect

pame JVM

ss