CloudWatch

If you are studying for AWS Developer Associate Exam, this guide will help you with quick revision before the exam. it can use as study notes for your preparation.

Dashboard Other Certification Notes

CloudWatch

CloudWatch is used for monitoring.

Why is monitoring important?

- To deploy applications
  - Safely
  - Automatically
  - Using Infrastructure as Code
  - Leveraging AWS components
- Because applications are deployed, and users don't care what services we've used
- Users only care that the application is working!
  - Application latency: will it increase over time?
  - Application outages: customer experience should not be degraded
  - Users contacting the IT department or complaining is not a good outcome
  - Troubleshooting and remediation
- Internal monitoring:
  - Can we prevent issues before they happen?
  - Performance and Cost
  - Trends (scaling patterns)
  - Learning and Improvement

Monitoring in AWS

- AWS CloudWatch:
  - Metrics: Collect and track key metrics
  - Logs: Collect, monitor, analyze and store log files
  - Events: Send notifications when certain events happen in your AWS • Alarms: React in real-time to metrics / events
- AWS X-Ray:
  - Troubleshooting application performance and errors
  - Distributed tracing of microservices
- AWS CloudTrail:
  - Internal monitoring of API calls being made
  - Audit changes to AWS Resources by your users

CloudWatch Metrics

- CloudWatch provides metrics for every services in AWS
- **Metric** is a variable to monitor (CPUUtilization, NetworkIn...)
- Metrics belong to **namespaces**
- **Dimension** is an attribute of a metric (instance id, environment, etc...).
- Up to 10 dimensions per metric
- Metrics have **timestamps**
- Can create CloudWatch dashboards of metrics

CloudWatch EC2 Detailed monitoring

- EC2 instance metrics have metrics "every 5 minutes"
- With detailed monitoring (for a cost), you get data "every 1 minute"
- Use detailed monitoring if you want to more prompt scale your ASG!
- The AWS Free Tier allows us to have 10 detailed monitoring metrics
- **Note:** EC2 Memory usage is by default not pushed (must be pushed from inside the instance as a custom metric)

AWS CloudWatch Custom Metrics

- Possibility to define and send your own custom metrics to CloudWatch
- Ability to use dimensions (attributes) to segment metrics
  - Instance.id
  - Environment.name
- Metric resolution:
  - Standard: 1 minute
  - High Resolution: up to 1 second (StorageResolution API parameter - Higher cost)
- Use API call **PutMetricData**
- Use exponential back off in case of throttle errors

Alarms are used to trigger notifications for any metric

- Alarms can go to Auto Scaling, EC2 Actions, SNS notifications
- Various options (sampling, %, max, min, etc...)
- Alarm States:
  - OK
  - INSUFFICIENT_DATA
  - ALARM
- Period:
  - Length of time in seconds to evaluate the metric
  - High resolution custom metrics: can only choose 10 sec or 30 sec

AWS CloudWatch Logs

- Applications can send logs to CloudWatch using the SDK
- CloudWatch can collect log from:
  - Elastic Beanstalk: collection of logs from application
  - ECS: collection from containers
  - AWS Lambda: collection from function logs
  - VPC Flow Logs:VPC specific logs
  - API Gateway
  - CloudTrail based on filter
  - CloudWatch log agents: for example on EC2 machines
  - Route53: Log DNS queries
- CloudWatch Logs can go to:
  - Batch exporter to S3 for archival
  - Stream to ElasticSearch cluster for further analytics
- CloudWatch Logs can use filter expressions
- Logs storage architecture:
  - Log groups: arbitrary name, usually representing an application. Log expiration policy should be defineda at this level.
  - Log stream: instances within application / log files / containers
- Can define log expiration policies (never expire, 30 days, etc..)
- Using the AWS CLI we can tail CloudWatch logs
- To send logs to CloudWatch, make sure IAM permissions are correct!
- Security: encryption of logs using KMS at the Group Level

AWS CloudWatch Events

- Schedule: Cron jobs
- Event Pattern: Event rules to react to a service doing something
  - Ex: CodePipeline state changes
- Triggers to Lambda functions, SQS/SNS/Kinesis Messages
- CloudWatch Event creates a small JSON document to give information about the change