AWS Monitoring & Audit: CloudWatch,x-ray

Monitoring

why monitoring is important ?

- we know how to deploy applications
  - safely
  - automatically
  - using infrastructure as code
  - leveraging the best AWS conponents
- our application are deployed, and our users don't care how we did it
- our 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 [抱怨] is not a good outcome
  - troubleshooting and remediation [补救措施]
- 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 happens 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 [内部的] monitoring of API calls being made
  - audit [审核] changes to AWS Resources by your users

CloudWatch

Metrics

- CloudWatch provides metrics for every services in AWS
- Metrics is a variable to monitor (CPUUtilization ,NetworkIn..)
- Metrics belong to namespaces
- Dimension [维度] is an attribute of a metric (instance id,environment, etc,,,)

- up to 30 dimensions per metric
- Metrics have timestamps [时间戳]
- can create CloudWatch dashboards of metrics

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 scale faster for 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 for inside the the instance as a custom metric)

Custom Metrics

- possibility to define and send your own custom metrics to CloudWatch
- example:memory(RAM) usage, disk space, number of logged in users..
- use API call **PutMetricData**
- ability to use dimensions [维度](attributes) to segment [细分] metrics
  - instance.id
  - environment.name
- metric resolution (**StorageResolution** API parameter – two possible value):
  - standard :1 minute (60 seconds)
  - high resolution:1/5/10/30 seconds – higher cost
- **important** : accepts metric data points 2 weeks in the past and 2 hours in the future (make sure to configure your EC2 instance time correctly)

Logs

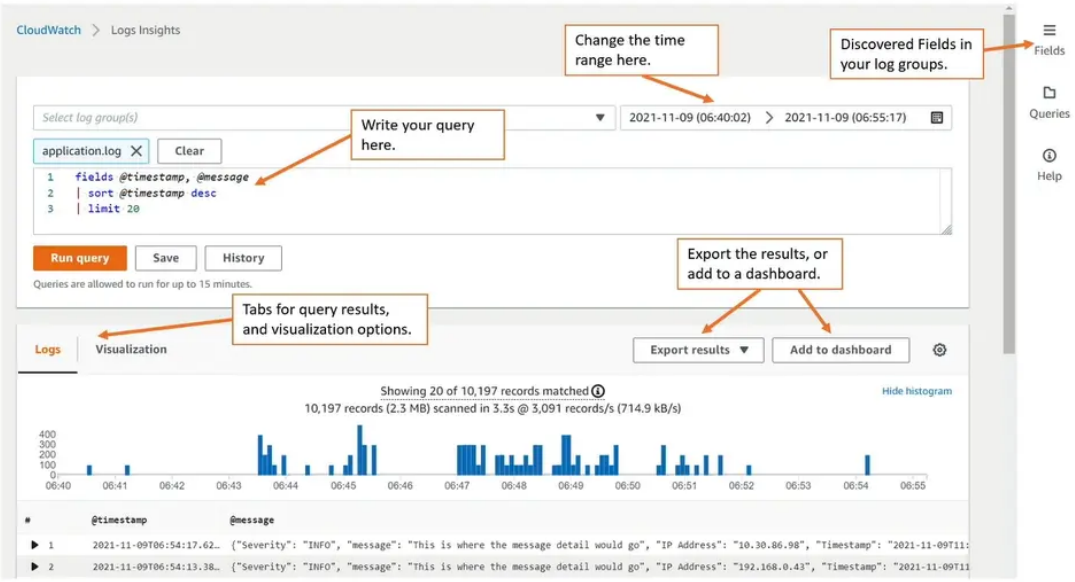
- **log groups** : arbitrary [随机] name, usually representing [代表] an application
- **log streams** : instances within application / log files / containers
- can define log expiration policies (never expire, 1day to 10 years)
- **CloudWatch logs can send logs to :**
  - amazon S3 (exports)
  - Kinesis Data Streams
  - Kinesis Data Firehose
  - AWS Lambda
  - OpenSearch
- logs are encrypted by default
- can setup KMS-based encryption with your own keys

Source

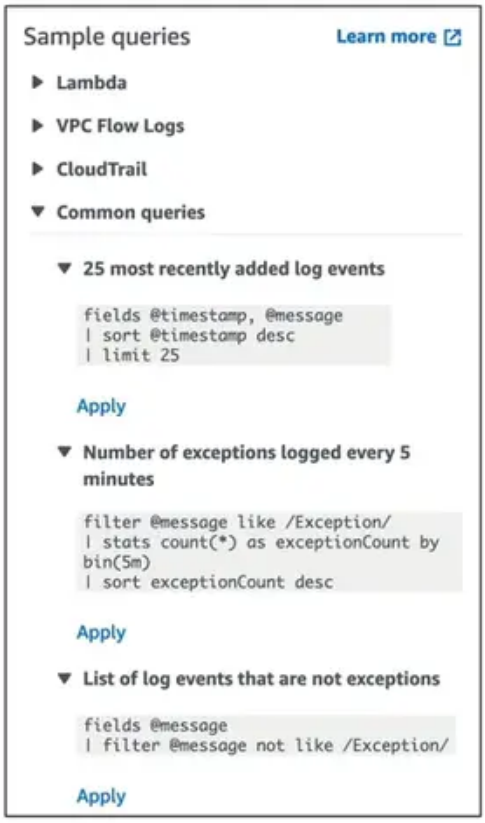
- SDK, CloudWatch Logs Agent, CloudWatch Unified [统一] Agent
- Elastic Beanstalk : collection of logs from application
- ECS： collection from contains
- AWS Lamabda: collection from function logs
- VPC Flow Logs : VPC specific logs

- API Gateway
- CloudTrail based on filter
- Route53 :Log DNS queries

logs insights



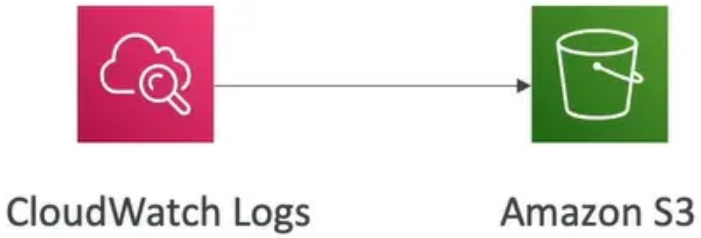
https://mng.workshop.aws/operations-2022/detect/cwlogs.html



- search and analyze log data stored in CloudWatch Logs
- example: find a specific IP inside a log, count occurrences [发生] of ”ERROR” in your logs
- provides a purpose-built [构建] query language
  - automatically discovers fields from AWS services and JSON log events
  - fetch desired [想要的] event fields, filter based on conditions, calculate aggregate [总计] statistics [统计], sort events,limit number of event..
  - can save queries and add them to CloudWatch Dashboards
- can query multiple Log Groups in different AWS accounts

- it's a query engine, not a real-time engine

S3 Export

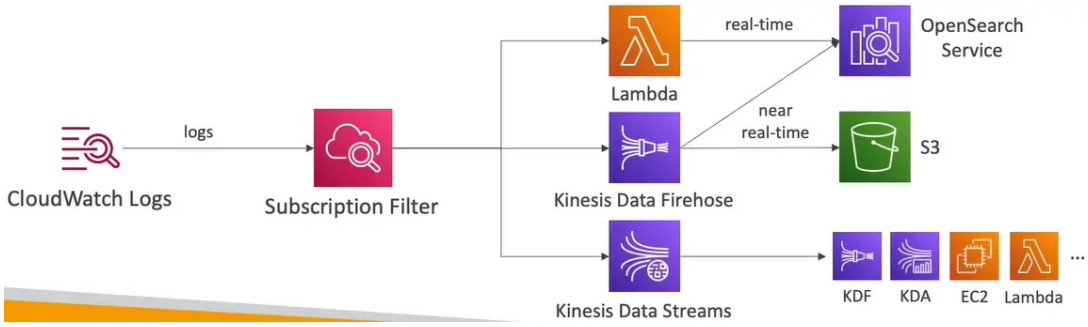


- log data can take **up to 12 hours** to become available for export
- the API call is **CreateExportTask**

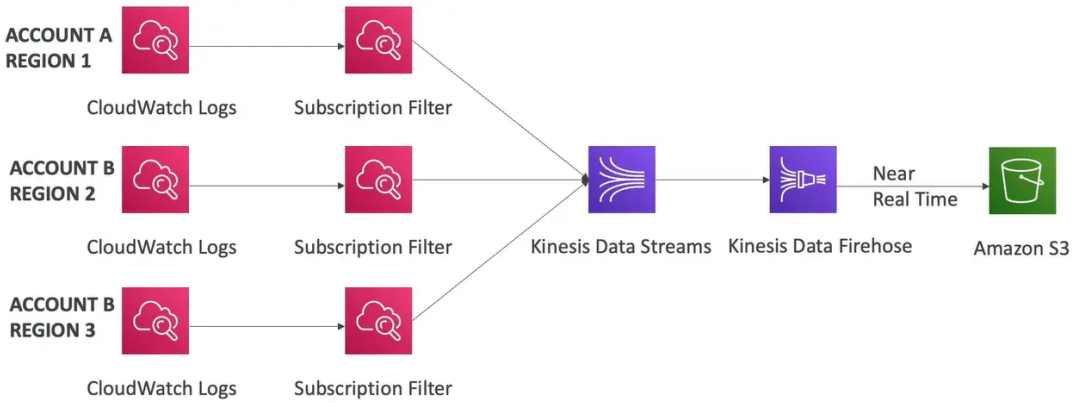
- not near-real time or real-time... use logs Subscriptions instead

Logs Subscriptions

- get a real-time log events from CloudWatch logs for processing and analysis
- send to Kinesis Data Streams, Kinesis Data Firehose, or Lambda
- **Subscription Filter** – filter which logs are events delivered to your destination

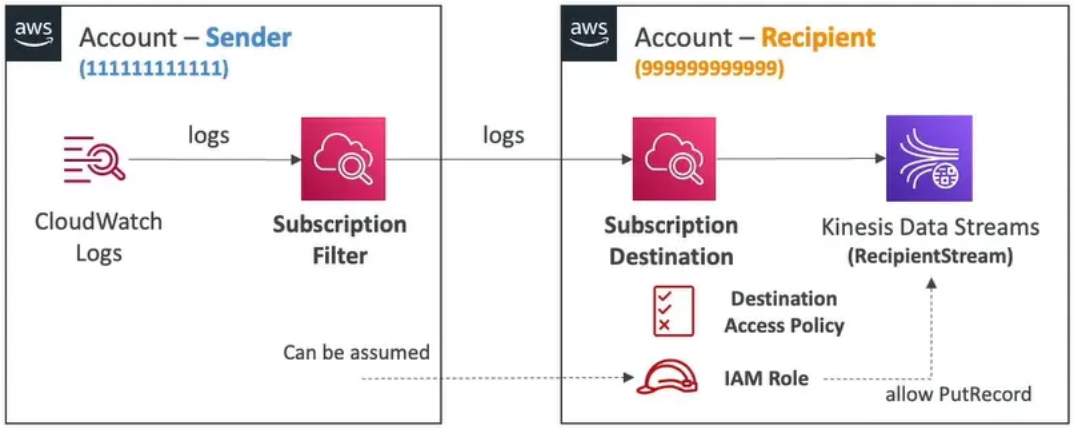


logs aggregation [聚合] multi-account & multi region



CloudWatch Logs Subscriptions

- **Cross-Account Subscription** – send log event to resource in a different AWS account(KDS,KDF)



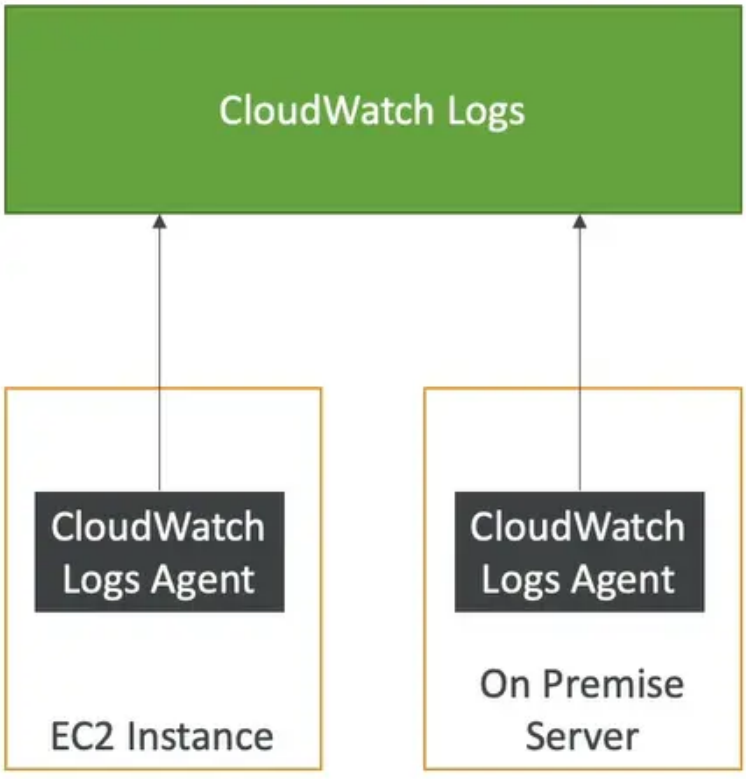
IAM Role (Cross-Account)

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "kinesis:PutRecord",
      "Resource": "arn:aws:kinesis:us-east-1:
999999999999:stream/RecipientStream"
    }
  ]
}
```

Destination Access Policy

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "AWS": "111111111111"
      },
      "Action": "logs:PutSubscriptionFilter",
      "Resource": "arn:aws:logs:us-east-1:999999999999:
destination:testDestination"
    }
  ]
}
```

Logs for EC2



- by default , no logs from your EC2 machine will go to CloudWatch
- you need to run a CloudWatch agent on EC2 to push the log files you want
- make sure IAM permissions are correct
- the CloudWatch log agent can be setup on-premises too

CloudWatch Logs Agent & Unified Agent

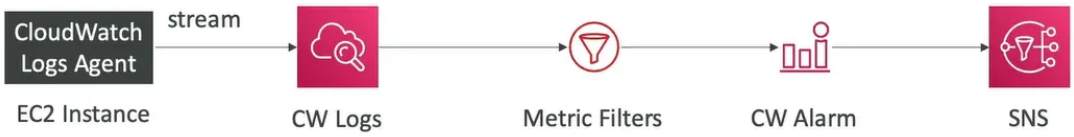
- for virtual servers (EC2 instance, on-permise servers..)
- CloudWatch Logs Agent
  - old version of the agent
  - can only send to CloudWatch Logs
- CloudWatch Unified Agent
  - Collect additional system-level metrics such as RAM,process,etc,,,
  - collect logs to send CloudWatch Logs
  - centralized [统一] configuration using SSM Parameter Store

CloudWatch Unified Agent -Metrics

- collected directly on your Linux server / EC2 instance
- CPU (active,guest,idle [闲置], sytem,user,steal)
- dish metrics (free, used,total,),dish IO (writes, reads,bytes,iops)
- RAM (free,inactive,used,total,cached)
- Netstat (number of TCP and UDP connections, net packets, bytes)
- Processes (total,dead,bloqued,idle,running ,sleep)
- Swap Space [交换空间](free,used,used %)

- Remember： out-of-the box metrics for EC2 -disk,CPU,network (high level)

CloudWatch Logs Metric Filter



- CloudWatch logs can use filter expressions
  - for example, find a specific IP inside of a log
  - or count occurrences of "ERROR" in your logs
  - metric filters can be used to trigger [触发] alarms
- filter do not retroactively [追溯] filter data, filters only publish the metric data points for events that happed after the filter was created
- ability to specify up to 3 Dimensions [维度] for the Metric Filter (optional)

Alarm

- alarms are used to trigger notification for any metric
- various options (sampling, %,max,min,etc..)
- alarm states:
  - ok
  - insufficient [不足]_data
  - alarm
- period:

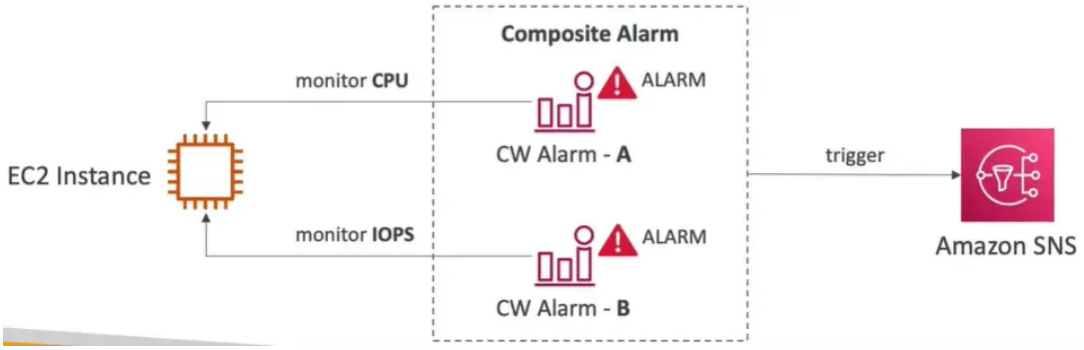
- length of time in seconds to evaluate the metric
- high resolution custom metrics: 10sec, 30sec or multiples of 60sec

targets

- stop,teminate,reboot, or recover an EC2 instance
- trigger auto scaling action
- send notication to SNS (from which you can do pretty much anything)

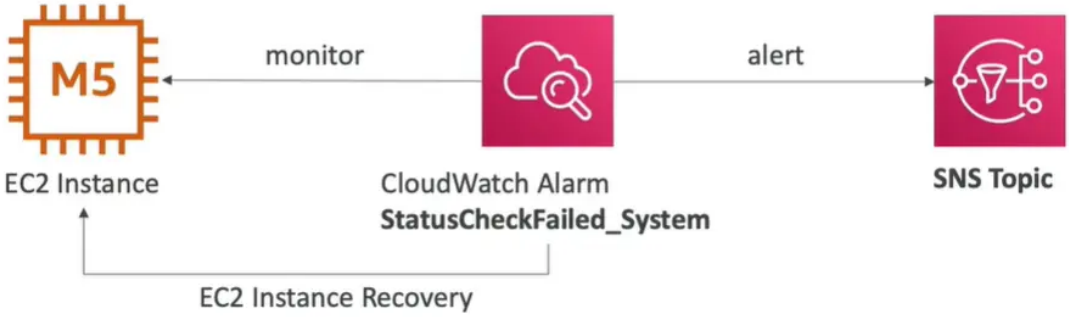
Composite [合成的] Alarms

- cloudwatch alarms are on a single metric
- composite alarms are monitoring the states of multiple other alarms
- AND and OR conditions
- helpful to reduce ”alarm noise” by creating complex composite alarms



EC2 instance recovery

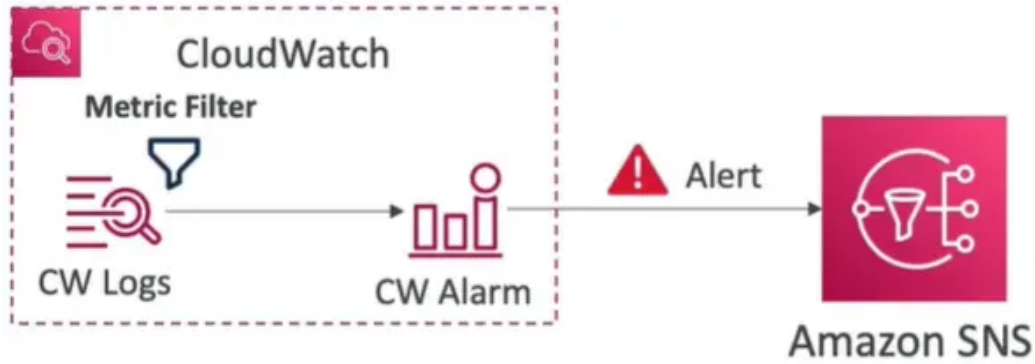
- status check:
  - instance status = check the EC2 VM
  - system status = check the underlying hardware



- Recovery : same private,public,elastic,IP,metadata,placement group

good to know

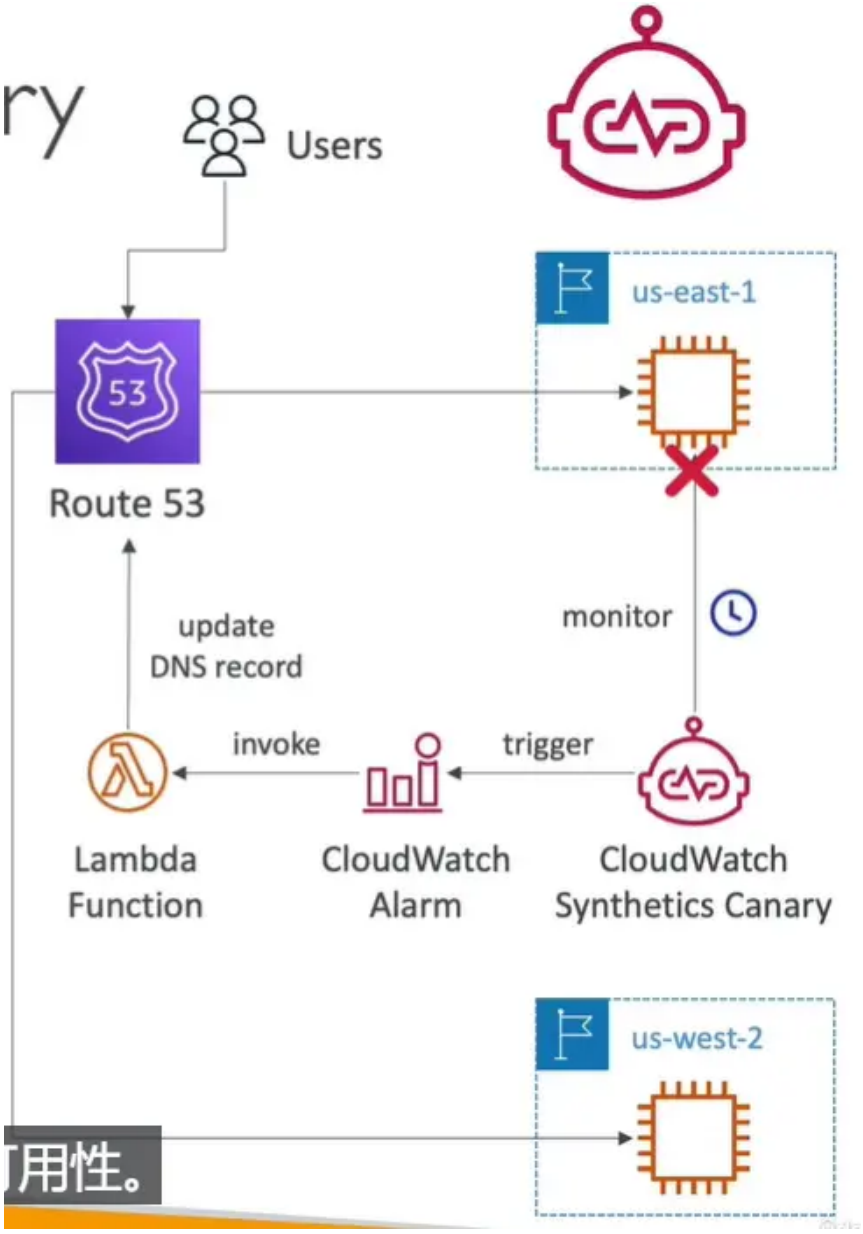
- alarms can be created based on CloudWatch logs metrics filters



- to test alarms and notifications,set the alarm state to Alarm using CLI

aws cloudwatch set–alarm–state ––alarm–name ”myalarm” ––state–value ALARM ––state–reason ”testing purposes”

Synthetics [合成的] Canary



- configurable script that monitor your APIs,URLs,Websites...
- reproduce what your customers do programmatically to find issues before customers are impacted [受影响的]

- checks the availability and latency of you endpoints and can store load time data and screenshots of the UI
- integration with cloudwatch alarms
- scripts written in Node.js or Python
- programmatic access to a headless google chrome browser
- can run once or on a regular schedule

Synthetics Canary Blueprints

- **heartbeat monitor** – load URL,store screenshot and an HTTP archive file
- **API Canary** – test basic read and write functions of REST APIs
- **broken link checker** – check all links inside the URL that you are testing
- **visual monitoring** – compare a screenshot taken during a canary run with a baseline screenshot
- **canary recorder** – used with CloudWatch Synthetics Recorder (record your actions on a website and automatically generates a script for that)
- **GUI Workflow Builder** – verifies that actions can be taken on your webpage (e.g, test a webpage with a login form)

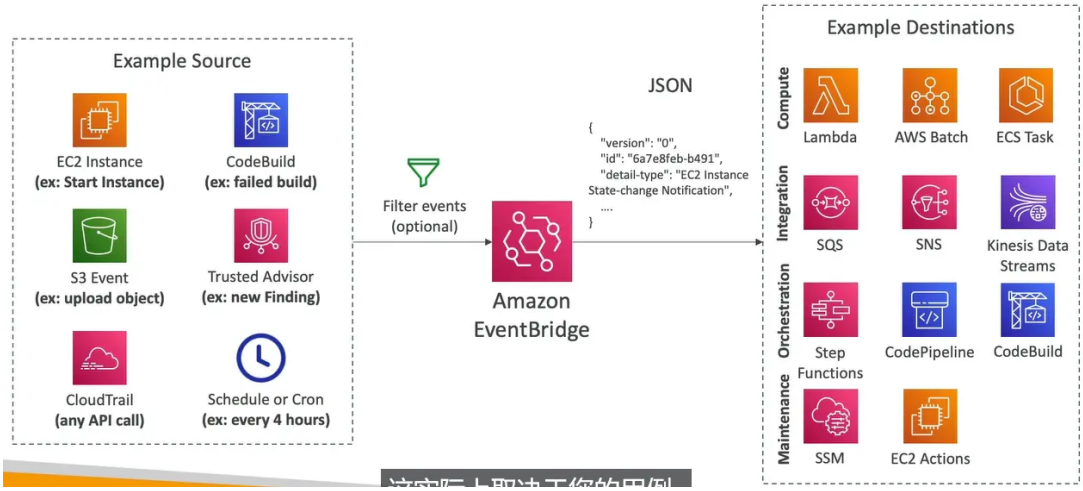
Amazon EventBridge (formerly CloudWatch Events)

- Schedule : Cron jobs (schedule scripts)

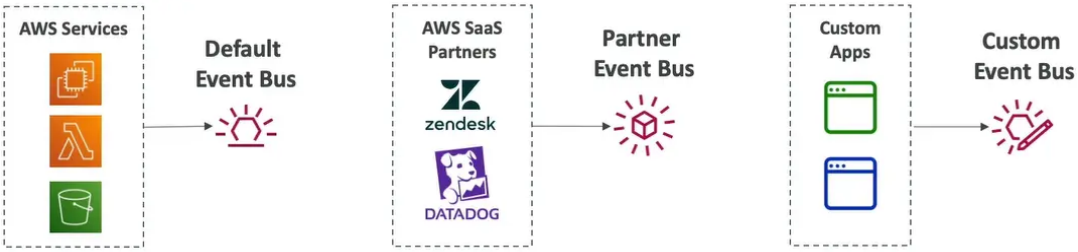
- Event pattern :event rules to react to a service doing something

- trigger Lambda functions, send SQS / SNS message...

rules



amazon eventbridge



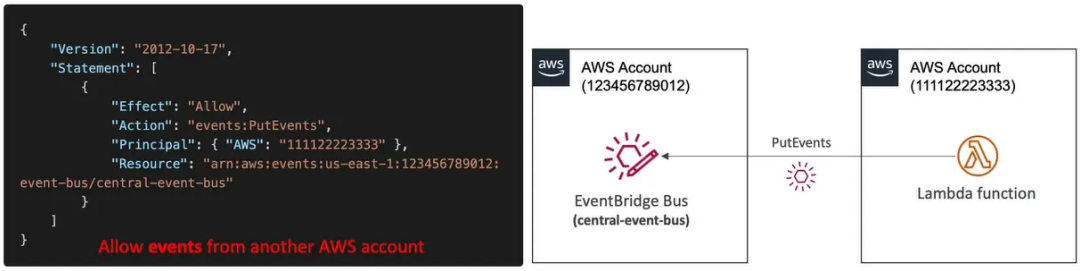
- event buses can be accessed by other AWS accounts using Resource-based Policies

- you can archive events (all / filter) sent to an event bus (idenfinitely or set period)
- ability to replay archived events

Schema Registry [模式注册表]

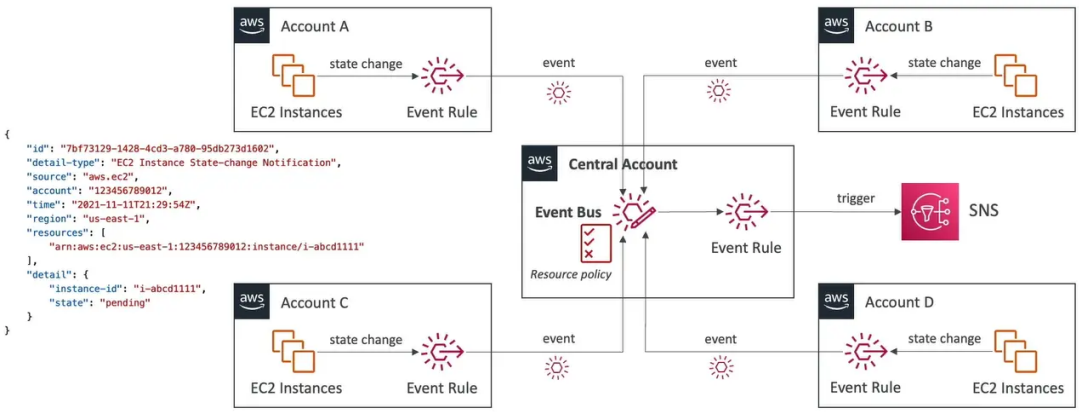
- EventBridge can analyze the events in your bus and infer [推断] the schema
- the schema registry allows you to generate code for your application, that will know in advance [提前] how data is structured in the event bus
- schema can be versioned

Resource- based Policy



- manage permissions for a specific Event Bus
- example: allow / deny events from another AWS account or AWS region
- use case : aggregate [聚合] all events from your AWS Organization in a single AWS account or AWS region

multi-account aggregation



AWS X-Ray

- debugging in production, the good old way:
  - test locally
  - add log statement everywhere
  - re-deploy in production
- log formats differ across applications using CloudWatch and analytics is hard
- debugging: monolith [单体] "easy",distributed services "hard"
- no common views of your entire architecture

visual analysis of our applications

advantages

- troubleshooting performance (bottlenecks [瓶颈])
- understanding dependencies in a microservice architecture
- pinpoint [查明] service issues
- review request behavior
- find errors and exceptions
- are we meeting time SLA ?
- where am I throttled [限制]?
- identify users that are impacted

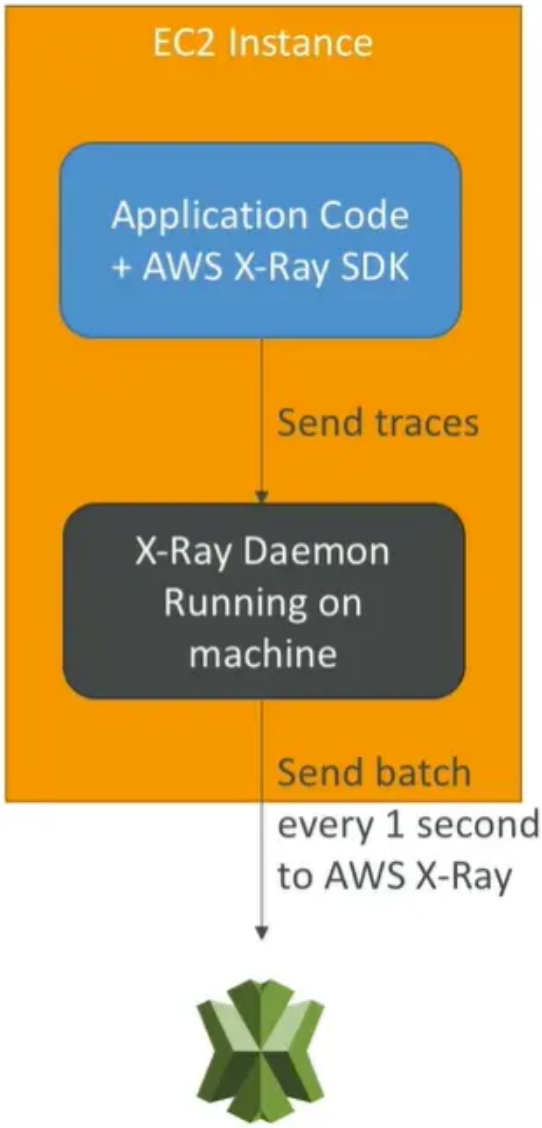
compatibility

- aws lambda
- elastic beanstalk
- ecs
- elb
- api gateway
- ec2 instance or any application server (even on premise [前提])

Leverages Tracing

- tracing is an end to end way to following a ”request”
- each component dealing with the request adds it’s own ”trace”
- tracing is made of segments [段] (+sub segments)
- annotations [注释] can be added to traces to provide extra-information
- ability to trace:
  - every request
  - sample request (as a % for example or a rate per minute)
- X-Ray Security
  - IAM for authorization
  - KMS for encryption at rest

how to enable it ?



1. your code (java,python,go,node.js,.net) must import the AWS X-Ray SDK

a. very little code modification needed
b. the application SDK will then capture
    i. calls to AWS services
    ii. HTTP / HTTPS requests
    iii. database calls (mySQL,PostgreSQL,DynamoDB)
    iv. queue calls (SQS)

2. install the X-Ray daemon [守护进程] or enable X-Ray AWS Integration
    a. X-Ray daemon works as a low level UDP packet interceptor [拦截器] (Linux /windows/mac)
    b. aws lambda /other AWS services already run the X-Ray daemon for you
    c. each application must have the IAM rights to write data to X-Ray

the X-Ray magic

- X-Ray service collects data from all the different services
- service map is computed from all the segments and traces
- X-Ray is graphical [图形化的], so even no technical people can help troubleshoot

Troubleshooting

- if X-Ray is not working on EC2
    - ensure the EC2 IAM Role has the proper [正确的] permissions
    - ensure the EC2 instance is running the X-Ray Daemon

- to enable on AWS Lambda
    - ensure it has an IAM execution role with proper policy (AWSX-RayWriteOnlyAccess)
    - ensure that X-Ray is imported in the code
    - enable Lambda X-Ray Active Tracing

X-Ray Instrumentation [检测] in your code

Example for Node.js & Express

```
var app = express();

var AWSXRay = require('aws-xray-sdk');
app.use(AWSXRay.express.openSegment('MyApp'));

app.get('/', function (req, res) {
  res.render('index');
});

app.use(AWSXRay.express.closeSegment());
```

- instrumentation means the measure of product's performance, diagnose [诊断] errors, and to write trace information
- to instrument your application code, you use the **X-Ray SDK**
- many SDK require only configuration changes
- you can modify your application code to custmize and annotation the data that the SDK sends to X-Ray, using **interceptors,filters, handlers, minddleware...**

Concepts [概念]

- segments : each application / service will send them
- subsegments : if you need more details in your segment
- trace : segments collected together to form an end-to-end trace
- sampling [采样]: decrease the amount of requests sent to X-Ray, reduce cose
- annotations : key value pairs used to **index** traces and use with **filters**
- metadata: kye value pairs, **not indexed**, not used for searching

- the X-Ray daemon / agent has a config to send traces cross account
  - make sure the IAM permissions are correct – the agent will assume the role
  - this allows to have a central account for all your application tracing

Sampling Rules

- with sampling rules, you control the amount of data that you record
- you can modify sampling rules without changing your code

- by default ,the X-Ray SDK records the first request **each second**, and **5 percent** of any additional requests
- **one request per second is the reservoir**, which ensures that at least one trace is recorded each second as long the service is serving requests
- **5 percent is the rate** at which addtional requests beyond the reservoir size are sampled

Custom Sampling Rules

Example Higher minimum rate for POSTs

- **Rule name** – `POST minimum`
- **Priority** – `100`
- **Reservoir** – `10`
- **Rate** – `0.10`
- **Service name** – `*`
- **Service type** – `*`
- **Host** – `*`
- **HTTP method** – `POST`
- **URL path** – `*`
- **Resource ARN** – `*`

Example Debugging rule to trace all requests for a problematic route

A high-priority rule applied temporarily for debugging.

- **Rule name** – `DEBUG - history updates`
- **Priority** – `1`
- **Reservoir** – `1`
- **Rate** – `1`
- **Service name** – `Scorekeep`
- **Service type** – `*`
- **Host** – `*`
- **HTTP method** – `PUT`
- **URL path** – `/history/*`
- **Resource ARN** – `*`

- you can create your own rules with the reservoir and rate

API

X-Ray Write APIs (used by the X-Ray daemon)

```
"Effect": "Allow",
"Action": [
    "xray:PutTraceSegments",
    "xray:PutTelemetryRecords",
    "xray:GetSamplingRules",
    "xray:GetSamplingTargets",
    "xray:GetSamplingStatisticSummaries"
],
"Resource": [
    "*"
]
```

arn:aws:iam::aws:policy/AWSXrayWriteOnlyAccess

- PutTraceSegments: uploads segment documents to AWS X-Ray
- PutTelemetryRecords : used by the AWS X-Ray daemon to upload telemetry [遥感数据].
  - SegmentsReceivedCount.
  - SegmentsRejectedCount,BackendConnectionErrors..
- GetSamplingRules: retrieve [检索] all sampling rules (to know what/when to send)
- GetSamplingTargets & GetSamplingStatisticSummaries: advanced
- the X-Ray daemon needs to have an IAM policy authorizing the correct API calls to function correctly

X-Ray Read APIs – continued

```
"Effect": "Allow",
"Action": [
    "xray:GetSamplingRules",
    "xray:GetSamplingTargets",
    "xray:GetSamplingStatisticSummaries",
    "xray:BatchGetTraces",
    "xray:GetServiceGraph",
    "xray:GetTraceGraph",
    "xray:GetTraceSummaries",
    "xray:GetGroups",
    "xray:GetGroup",
    "xray:GetTimeSeriesServiceStatistics"
],
"Resource": [
    "*"
]
```

- GetServiceGraph: main graph
- BatchGetTraces: retrirves a list of traces specified by ID, Each trace is a collection of segment documents that originates from a single request
- GetTraceSummaries:retrieves IDs and annotations for traces available for a specified time frame using an optional filter. to get the full traces, pass the trace IDs to BatchGetTraces.
- GetTraceGraph :retrieves a service graph for one or more specific trace IDs

X-Ray with Beanstalk

- AWS Elastic Beanstalk platforms include the X-Ray daemon
- you can run the daemon by setting an option in the Elastic Beanstalk console or with a configuration file (in. ebextensions /xray-daemon.config)

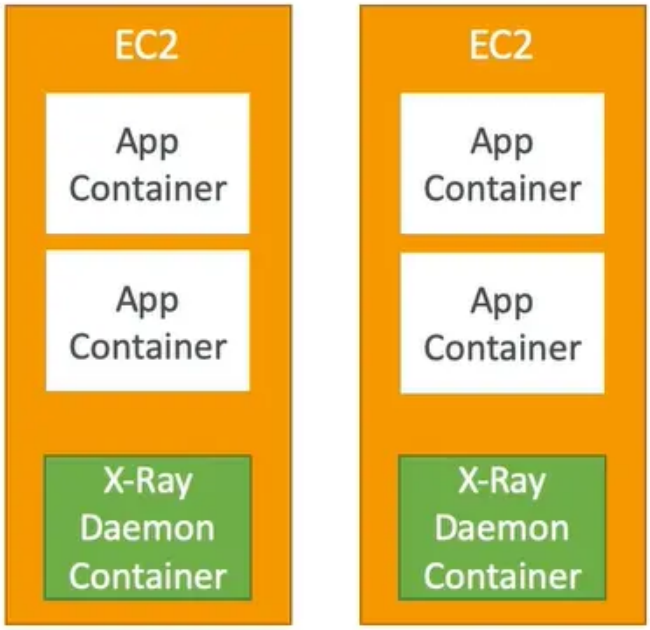
```
option_settings:
  aws:elasticbeanstalk:xray:
    XRayEnabled: true
```

- make sure to give your instance profile the correct IAM permissions so that the X-Ray daemon can function correctly
- then make sure your application code is instrumented with the X-Ray SDK
- Note : the X-Ray daemon is not provided for Multicontainer Docker

ECS + X-Ray integration options

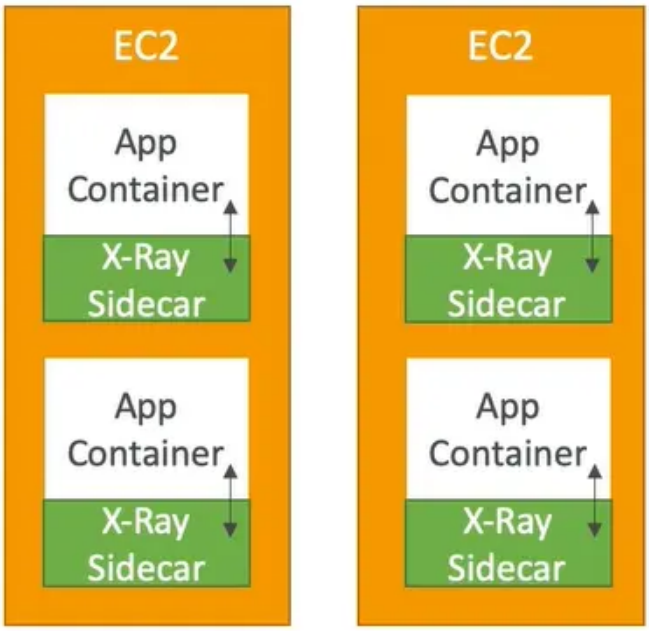
ECS Cluster

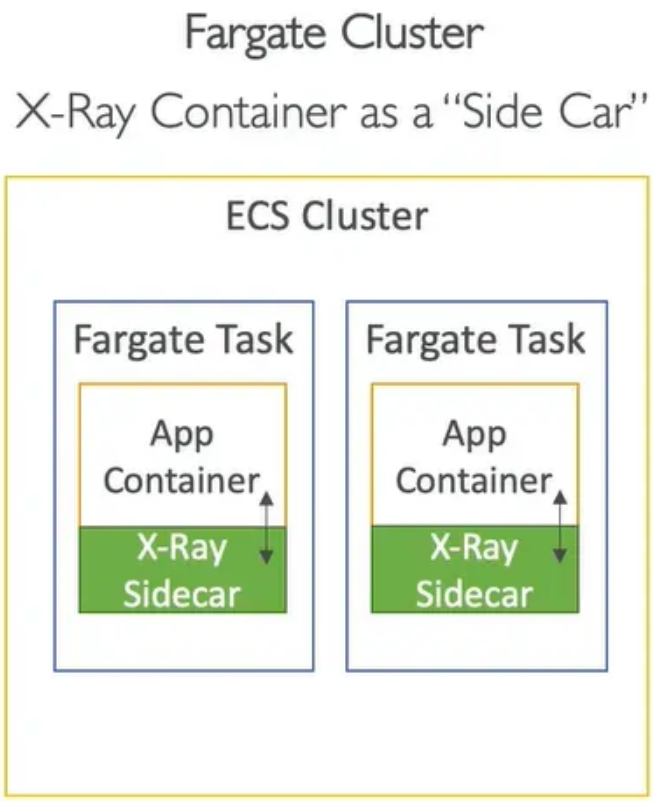
X-Ray Container as a Daemon



ECS Cluster

X-Ray Container as a "Side Car"





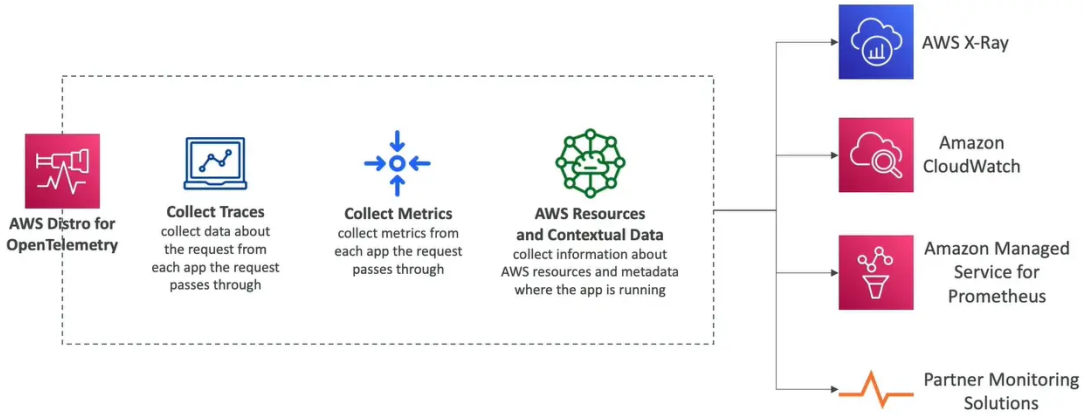
ECS +X-Ray : Example Task Definition

```
{
  "name": "xray-daemon",
  "image": "123456789012.dkr.ecr.us-east-2.amazonaws.com/xray-daemon",
  "cpu": 32,
  "memoryReservation": 256,
  "portMappings" : [
      {
          "hostPort": 0,
          "containerPort": 2000,
          "protocol": "udp"
      }
   ],
},
{
  "name": "scorekeep-api",
  "image": "123456789012.dkr.ecr.us-east-2.amazonaws.com/scorekeep-api",
  "cpu": 192,
  "memoryReservation": 512,
  "environment": [
      { "name" : "AWS_REGION", "value" : "us-east-2" },
      { "name" : "NOTIFICATION_TOPIC", "value" : "arn:aws:sns:us-east-2:123456789012:scorekeep-notifications" },
      { "name" : "AWS_XRAY_DAEMON_ADDRESS", "value" : "xray-daemon:2000" }
  ],
  "portMappings" : [
      {
          "hostPort": 5000,
          "containerPort": 5000
      }
  ],
  "links": [
    "xray-daemon"
  ]
}
```

AWS Distro for Open Telemetry

- secure, production-ready AWS-support distribution of the open-source project Open Telemetry project
- provides a single set of APIs, libraries. agents, and collectior services
- collects distributed traces and metrics from your apps
- collects metadata from your AWS resources and services
- Auto-instrbumentation Agents to collect traces without changing your code
- send traces and metrics to multiple AWS services and partner solutions
  - X-Ray,CloudWatch,Prometheus..

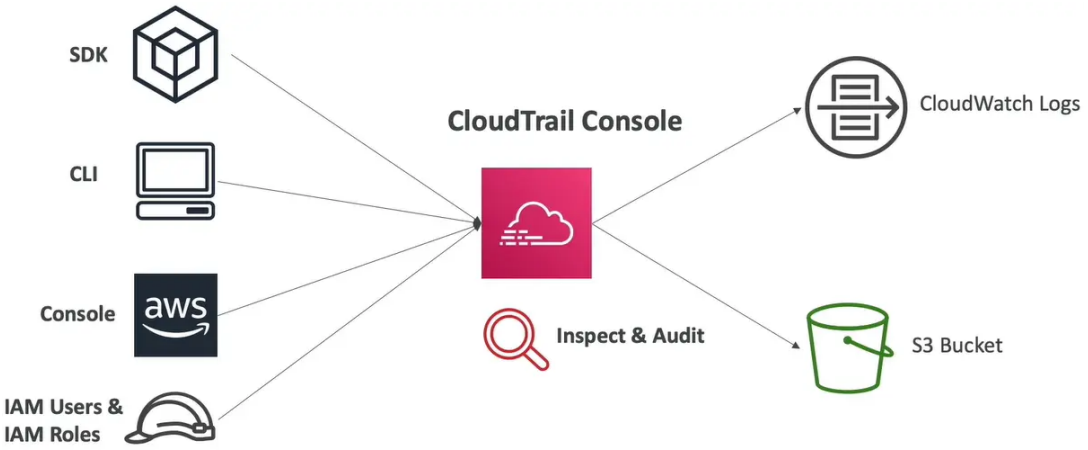
- instrument your apps running on AWS (eg.EC2,ECS,EKS,Fargate,Lambda)as well as on-premises
- migrate from X-Ray to AWS Distro for Temeletry if you want to standardize with open-source APIs from Telemetry or send traces to multiple destinations simultaneously



AWS CloudTrail

- provides governance [治理], compliance [遵守] and audit [审计] for your AWS Account
- CloudTrail is enabled by default
- get an history of events / API calls made within your AWS Account by
  - console
  - sdk
  - cli
  - aws service
- can put logs from CloudTrail into CloudWatch Logs or S3
- A trail can be applied to All Region (default) or a single Region
- if a resource is deleted in AWS, investigate [调查] CloudTrail first !

CloudTrail Diagram



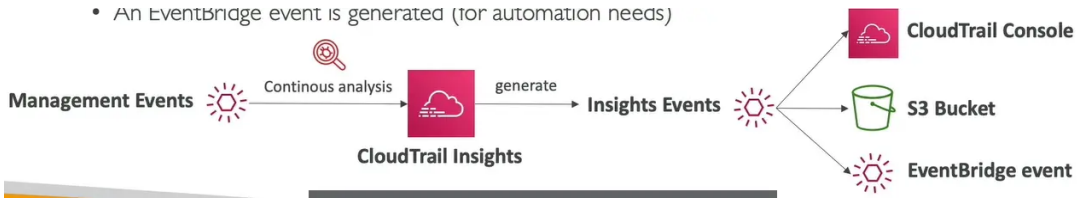
CloudTrail Events

- Management Events
  - operations that are performed on resource in your AWS account
  - examples:
    - configuring security (IAM AttachRolePolicy)
    - configuring rules for routing data (Amazon EC2 CreateSubnet)
    - setting up logging (AWS CloudTrail CreateTrail)
  - by default, trails are configured to log management events

    - can separate **Read Events** (that don't modify resource) from **Write Events** (that may modify resources)
- Data Event
    - **by default, data events are not logged (because high volume operations)**
    - Amazon S3 object-level activity (ex: **GetObject,DeleteObject,PutObject**): can separated Read and Write Eventss
    - AWS Lambda function excution activity (the **Invoke [调用]** API)
- CloudTrail Insights Events

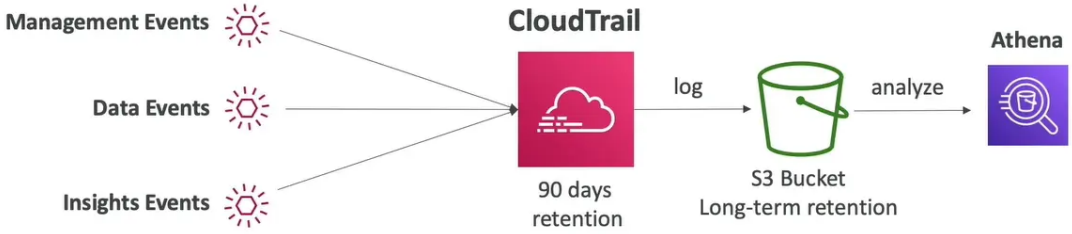
CloudTrail Insights

- enable **CloudTrail Insights to detect [探测] unusual activity** in your account
    - inaccurate [不准确] resource provisioning
    - hitting service limits
    - bursts of AWS IAM actions
    - gaps in periodic maintenance activity
- CloudTrail insights analyze normal management events to create a baseline
- and then **continuously analyze write events to detect unusual patterns**
    - anomalise appear in the CloudTrail console
    - event is sent to Amazon S3
    - an EventBridge event is generated (for automation needs)

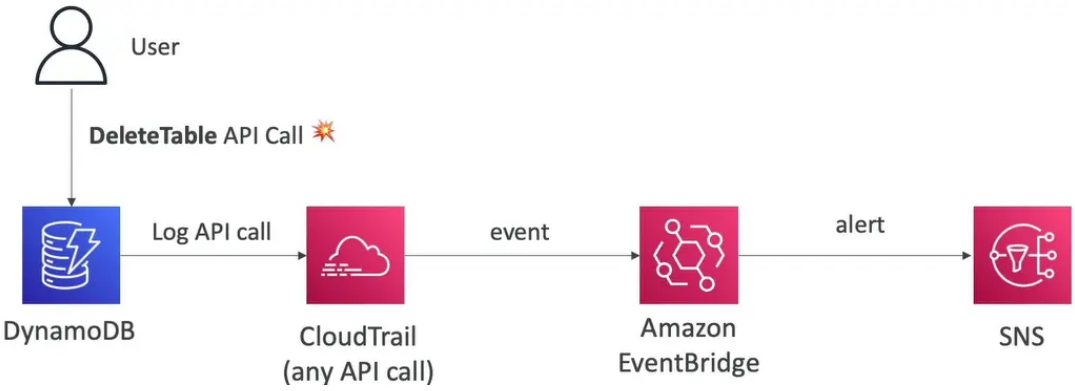


Event Retention

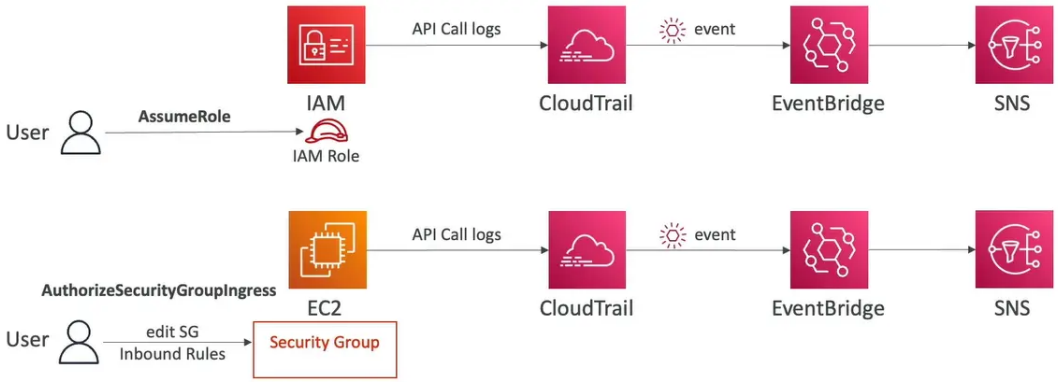
- events are stored for 90 days in CloudTrail
- to keep events beyond this period, log them to S3 and use Athena



Amazon EventBridge – Intercept API Calls



Amazon EventBridge – CloudTrail



CloudTrail vs CloudWatch vs X-Ray

- CloudTrail
  - audit API calls made by users/services /AWS console
  - useful to detect unauthorized calls or root cause of changes
- CloudWatch
  - CloudWatch Metrics over time for monitoring
  - CloudWatch logs for storing application log
  - CloudWatch Alarms to send notification in case of unexpected metrics
- X-Ray:
  - automated Trace Analysis & Central Service Map Visualization
  - latency,Errors and Fault analysis
  - request stracking across distributed system