Sthitheprajna

* Invalidate an API Gateway cache entry
  - → The client must sends a request that contains max-age = 0;
  - → attach InvalidateCache policy to IAM role.

* If you have IAM resources with custom names you must specify CAPABILITY_NAMED_IAM

* To decrease connections made to database
  - → go through RDS proxy.

* disable rollback command.

* whenever there is a file upload in S3, immediately trigger λ function.
  - → A trigger is a resource where we you configure to allow another service to invoke your function when an event (or) conditions occurs.

* when calling API if any error related to time and date

→ change the format of λ fun response

* S3 Intelligent Tiering → Patterns

* Bulk retrival atleast requires 48 hrs.

* Standard retrival means within 24 hrs.

* Mem cached → multi thread & scalable caching solution that used to offload heavy traffic.

* Redis → Datasets as hashsets, sorted sets, lists. Supports replication.

* Sam local start api → to run λ functions locally and test locally

* Parameter store → used to store parameters. Sensitive data.

* Secrets manager → DB credentials encrypted rotated at regular interval.

* CDK Custom resource

→ integrate seamlessly

→ least amount of configuration

* Before deploying using SAM CLI

→ bundle the serverless application using Sam Package.

Synchronous → will wait

Asynchronous → immediately.

* Asynchronous invocation

→ λ for trier error twice.

→ queue may wait for hrs or days

→ configure DLQ to successful process.

* Signed URLs → lowest cost, secure control access to downloads.

* Global Secondary Index → Fastest response.

* Aws. Kms managed keys → audit trail available.

* To prevent deletion (or) updates to resources in cf stack

→ Deletion Retention Policy.

→ Prevent updates to stack.

→ IAM policy to restrict.

→ Turn on terminication Protection.

* env files under task definition but not service definition

* Time related → IntegrationLatency, Latency.

* Count related → CacheHitCount, CacheMissCount

* encrypt at rest → default → cost effective
  managed keys → cost will have.

* to cache POST requests to optimize the API resources
  → override the cache method in API GW. select Post method.

* ThrottlingException errors, error handling
  → exponential backoff.

* AWS Parameter store (Parameters).
  → securely encrypt connection string

* TTL (time to live)
  → automated way to delete old items from the table.
  → delete items in DynamoDB after certain time (48hrs like).

* Previous version must retired when deployement of new version completes
  → Blue/green deployments (more cost)
  → Immutable deployements. (less cost).

* lamda layers for dependencies.

* when API triggers lambda.
  Asynchronous → immediate process
  Synchronous → wait to process.
  X-Amz-Invocation-Type header. ("Event")

* Principle of least privileges → whatever they want give that permissions.

* update item → update, creat item → put.

* Initializing SDK outside λ function then
  Primary benefit is
  "Takes advantage of runtime env used".

* Event Bridge → designated time each day.

* SDK encrypts keys & returns Ciphertext

* Trust policies → from role A in account one to role B in account two.

* test locally (automated manner).
  → sam local generate-event command.

* existing versions → All at once, rollback
* new versions → blue/green, immutable.

* AWS X-ray → to identify & troubleshoot in distributed λ functions

* custom namespace metric is used for graphical representation of metrics

* session data is stored in ElasticCache

* EventBridge → used to schedule λ functions for a period of time.

Build - Test - Deploy

* KMS keys annual rotation is keys generated by AWS but not customer.

* sam sync watch → redeploy λ changes

(cloudfront + S3 → unanimous)

* whenever we make changes before deployed to left, use existing API and modify.

* To resolve merge conflicts, the developer ship PR's from main branch & Rebase.

* AWS X-ray SDK → for bottleneck errors and other errors.
(x-ray demon)

* API GW supports mock integrations.

* Anytime DB Credentials must securely go with Secrets manager.

* AWS Secrets Manager → rotates
manages
retrives DB Credentials
API GW's throughout
its life cycle.

* NAT GW → enables instances in Private subnet. to connect internet.

* dedicated left env means we will having multiple stages in API GW.
↓
for each env use one λ fun

* BatchGetItem operation returns attributes of one (more items from one/more tables.

* whenever external library comes go with λ layers. (3rd party) → Centralize libraries.

* lazy loading → Cache when every need
* write through → Cache when real-time

* Lambda@Edge functions can be created only in us-east-1 region. (N. virginia).

* DynamoDB - single-digit millisecond retrival time is possible only dynamodb

* cloudwatch Agents for minimum charges.
((EC2))

* WebSocket Connections

→ Callback URL

→ $connect & $disconnect.

* whenever DynamoDB, near-real time

→ go with DynamoDB Streams.

authenticated access → AUTH AWS IAM.

unauthenticated → NONE AUTH

* Pseudo Parameters → Predefined by AWS CF

* Security Groups → Inbound & Outbound traffics.

* All at Once → deploy new versions very fast ext. quick deploy method.

* Cross Region AMI Copy.

* already created account means go with Cognito identity pools.

Personal → name, Ph, id, address

Financial → Credit card.

* EFS → Same shared log file

* whenever throttling go with SQS

UR 2 functions

1 fun → to load , 2 fun → to transform

* when img is Uploaded S3 sends notification to SNS topic.

* VPC level logs → VPC Flow logs.

* retrive multiple items from DynamoDB table

BatchGetItem.

* Best option to store session data is

ElasticCache.

* Run cdk synth & sam local invoke for testing a specific λ fun locally.

* BeanStalk → without require knowledge of underlying infrastructure.

* cloudwatch Agent → tracks memory usage, gain insights, custom metrics also.

* CW logs only publish metric data only after filter is created.

* whenever you see encryption related, go through KMS only.

* To know EC2 IPv4 address or instance metadata.
  → Query instance metadata.

* for accurate rolling avg → must use previous values.

* λ env variables, can changed easily without modifying the code.

* manually instrument X-ray SDK to prevent PII goes out of EC2 instances.

* In deployments when performance degrades go with ~~add~~ rolling with additional batches.

* To simulate backend responses without invoking backend service
→ go with mock integration response

* keep /tmp directory → files which not be saved or retrive in future.

* x-amz-sse → to encrypt obj with S3 managed keys

* If you want to create custom domain from a 3rd party provider
→ it should be in us-east-1 Region only (i.e) N. virginia.)

* AWS SAM → serverless resources (YAML)

* Redis → sorted sets, hashes, lists
* memcache → multithread, session states.

* To enhance responsiveness of the APIs
→ enable API Caching in API G-W

run-order for inplace deployments

Application Stop > Before Install > After Install > Application Start

DynamoDB + S3 ⇒ unanimous

* To return old versions of functions quickly & seamlessly
  → use function alias with different versions.

* X-ray Demon → to enable x-ray tracing
  For. onpremisis servers
  (minimal configurations).

* API keys are also stored in secret manager.
  Access tokens ↗

* Step functions → manage & automate the orchestration of data flows

* GSI does not consume DynamoDB's existing table read capacity. so go with (GSI)

rolling → small batches one at time (size of 1)

all at once → all instances replaced simultaneously leading to downtime.

canary → gradually role out to subset of users

immutable → for all users quickly, new versions will be enabled & old versions are terminated.

* encrypted Snapshots for data at rest.

* To maintain session states across EC2 instances use ElasticCache [memcache].

* unnecessary API requests are caused by long polling.

* Test code in local system using CodeBuild agent before pushing to main branch.

* Event Driven → As soon as new data is added to DynamoDB (near real-time purpose)

* Stage → different environments
* alias → different versions

λ

* Sticky session → it ties the user with specific EC2 instance.

* To maintain session login use DynamoDB table to store session (login status)

* use Elasticache to reduce ReadLatency metric
→ more db reads than writes. so

* Before uploading files to AWS S3 to encrypt the files we use
→ create data key in KMS & use SDK

* If we want to use ALB with Cognito then
⇒ authenticate - cognito must

* To prevent throttling in λ
→ use functions reserved Concurrency
→ Req for service quota.

* whenever we see 10.1', 10 min like that go with canary [pre traffic & post traffic (12 blue traffic stops)]

* use CDK assertion module to integrate unit testing & creat custom rules.

invoking means → EventBridge
orchestration means → Step functions

* Immutable, Blue/Green } → deploy changes to new instances only.

* /tmp → to store temporary files in λ

* cloudwatch logs, insights for dashboards

* To launch new EC2 instance through AWS

ec2
→ aws ec2 run-instances.

GetItem → to retrive
UpdateItem → to update
DeleteItem → to delete.

PutItem → to Create.

* X-Forwarded-For header

→ to maintain the ability to scale horizontally for ALB's.

* ~~To filter expressions in X-ray~~

~~Add a custom annotation~~

* To filter expressions in X-ray.

→ Add a custom attribute as annotations

* Kinesis Firehose → for loading, streaming data into destinations such as S3, Redshift, ElasticSearch.

* KDS → for real time process million of events recived from API

* Websocket API → to provide changes to the user without refreshing the webpage, periodically

Connect API

* to connect 3rd party service endpoints → OK
  API Gateway stage variables.

* In Serverless applications to deploy incremental changes → sam sync command (SAM)

Plaintext → to encrypt
encrypted key → to decrypt

* ephemeral disk → data will lost if instance is terminated.

* To overcome processing msgs multiple times in SQS → use deduplication ID

* lamda layers size limit to 250 MB only.

* whenever they ask about threshold within certain time period
  → go with cloudwatch metric filter.

* SQS max visibility timeout as 12 hrs only

* HTTP 401 → unauthorized logins.
  404 → request resource not found
  503, 505 } service not aviable.

* STS means AssumeRole only

→ DynamoDB has its own default encryption options.

* In SQS Simultaneous means concurrency

* Glacier Instant Retrival → milliseconds retrival for rarely accessed data.

* Applanfig → quickly deploy configuration changes without any disruptions.

* don't want to overwhelm means, we can decouple so the SQS

* subset of users → weighted (or) canary deployments

* delete old data based on timestamp go with TTL (direct-attach)

* SQS event source mapping can limit the concurrent requests.

* creating indexes helps us to speed up the process
  they are GSI & LSI

→ whenever they say attribute is not a table partition key (or) sort key
  create GSI

* lambda authorizer used for authorization

* To mock 3rd party API services like Payment go with AppSync GraphQL

* encrypt at rest → KMS (and related)
  encrypt at transit → certificates SSL/TLS

* failed msgs in λ mess → dead letter queue

* Transaction capability → DynamoDB r/w Transact
  Aurora MySQL

ssm-secure → secureString
ssm-dynamic → plainText

* for rate limit errors → reserved concurrency

* Partition key must be unique.

* to capture the client IP address

x-forwarded-for-header : (ALB & HTTP HTTPS)

* NLB → Layer 4 → TCP & UDP
* ALB → Layer 7 → authentication & authorization
* CLB → traffic distribution.
* AWS AppConfig → configuring & feature flags.

Ref intrinsic function → within same CF template
ImportValue " " → other CF templates.

* associate-kms-key command & ARN

used for encrypt future data.

LSI

↓

Same Partition key
diff Sort key

GSI

Diff partition key
Diff Sort key.

eventual Consistent →
Strongly consistent → latest data with all.

* To change partition key use GSI

* temp → ephemeral → temporary purpose

* whenever they ask client sessions

↓

go with DynamoDB & Elasticache

NLB → TCP level
GLB → distribute traffic.

* drift detection → to modify changes to security groups

* Amazon KDS → to monitor error rates & anomalies (near realtime)

+

Opensearch.

* Preserve the order → FIFO dLQ topic

otherwise Standard DLQ.

* IAM DB authentication for MySQL

↓

for short lived credentials (not long).

* DAX → a caching service designed to speed up read op's.

* DynamoDB Streams → capture changes, deletion related TTL, expire items like that (optimal)

* To add credential caching & reduce repeated usage of secret keys

↓

1. token based ↳ authorizer
2. mapping template (map context)

* Git tag → mark specific versions.

↓

* appsec → to define deploy instructions → Code Deploy

buildspec → to define test reports → Code build

(yaml)

* App Config Agent → extensions → to retrive Config data dynamically from AppConfig without redeployment

* PurgeQueue → delete all msgs
* SendMsgBatch API → to reprocess failed msg

* CodeDeploy Agent → test & debug code locally.

* zip files only 250 MB
  in case of ECR → 10 GiB } dependencies

* cdk bootstrap → create resources needed by CDK in other accounts also

* 502 error → related to λ response
  403
  401 } λ authorization header.

* X-Ray SDK → monitor end to end view of requests b/w microservices & debug.

* to reduce cold start time use provisioned concurrency

* To externalize the session state

→ DynamoDB
→ ElasticCache

DynamoDB - DAX → microseconds retrieval
RDS - ElasticCache

To automate deployment process in SAM

↓

→ aws cloudformation package
→ sam package

* In SQS - Received ordering

* Create, Describe, Delete, Inbound to SQS

* to reduce cold start

→ AWS SDK

→ ↑ memory Allocate

* to prevent over-writing use conditional write

* for more recent value for each data item

→ write-through strategy created & update

* on KDS

→ ↑ no. of shards

→ ↑ memory of λ fn.

* to place analysis stage before deployment

→ create new codepipeline state after img is built.

* 403 forbidden

→ add a second cache behavior with default settings unchanged.

* end-end testing → amplify.yml file.

* application performance degrades many use → rolling with additional batches.

* fun can't connect to RDS DB

→ Redeploy λ fun in same subnet & allow traffic from λ func.

* for accurate rolling avg calculations

→ Set fun's reserved concurrency to 1
→ store in ElastiCache

* PullRequest Created
PullRequestSourceBranch Updated } CodeCommit

* to know public IPv4 address of an instance → use metadata

* Generally for access client IP address we use x-forwared-for-headers
If they use HTTP/HTTPS → ALB

* gradually means linear
otherwise canary only

* to prevent others to overwrite

→ update ↗ fun
→ modify ↗ fun.

} hotfix alias

* to test ↗ fun locally

→ cdk synth & sam local invoke

* to share & access files securly

→ S3 presigned URL's

* to set user credentials without hardcoding

→ Lambda@Edge, invoke viewer request, add permisson to func execution role.

* VPCExecutionRole policy
RDS SG to allow inbound.

} VPC Queries

* ~~FAST~~
* fastest deployment is → All at Once.

* cached data must populate realtime dashboards

→ write through cache.

Lazy loading → only cache when necessary

- SSM Put Parameter operation and Parameter type to SecureString.

- aws:SourceVpc → StringNotEquals Condition & Repeat for all 2 DB's

- Root Cert CA
  - → secrets Manager
  - → Refactor λ fun outside

- Step functions to monitor API failures & use wait state to delay

- sam local generate-event } SAM CLI tool
  sam local invoke.

- Initialising SDK outside λ
  - → Take advantage of runtime env reuse.

- for ProvisionedThroughputExceededException error
  - → exp backoff ✓
  - → AWS SDK ✓

* to optimize photo, to reduce load time & increase photo quality

→ use lambda@Edge with.
& subsequent req

fixed % → alias with weighted alias
10%, 90% → canary
gradually ↑% → linear

* to AWS SAM deploymts to multiple env
→ use TOML format

* to provision & update CF stacks using CLI
→ add disable rollback command to create stack, update stack

* to use filter expressions in X-RAY
→ add custom attributes as annotations

* millions of events in real time
Processing recieved through API
→ KDS

* to reference new S3 bucket from another CF template

→ add exports to output & ImportValue

* In SQS, 3rd party API returns HTTP 429 too many req

→ max concurrency on SQS

* Before uploading to S3 bucket we have to encrypt means

→ Create data key in KMS & use SDK

* Canary AutoPublish Alias only.

* to immediately update the validate report on UI dashboard without reloading

→ use websocket API, WebSocket connectID.

* JSP Question

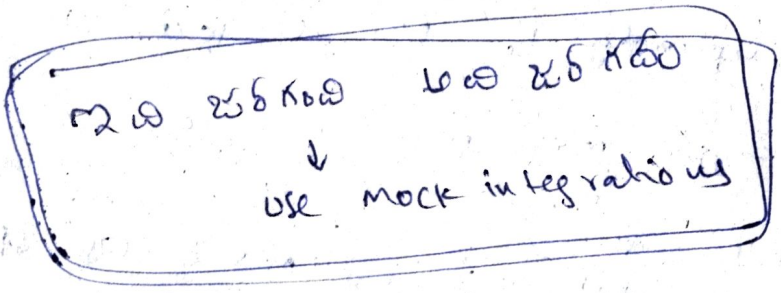
→ Authorizer Caching enabled in 1 authorizer.

+ to deploy incremental changes but not entire application

→ use AWS SAM → sam sync command

+ to avoid processing msgs multiple time in SQS

→ use SQS deduplication ID and change to SQS FIFO.



④ ⇒ layers only 250 KB
if more go with EFS

+ METHOD_NOT_ALLOWED → 401

+ to avoid resetting parameter values

→ set deletion policy to Retain.

+ ⇒ for destination with failure condition

* Put Cluster Capacity Providers

FARGATE as Provider 1 base val

FARGATE-SPOT as Provider 2 failover.

* event source mapping - maximum concurrency 10 for high priority queue 90 for low priority queue

* create new codepipeline steps after image is built

* for rate limiting errors - reversed Concurrency

* store video in S3 & set Prefixed URLs

* freeze changes to codecommit means

→ so with ~~Merge~~ merge

* To ensure workflow automatically retries

→ Add a retry field in step functions,

* automatic rotation ↔ alternating users.

* to conduct tests with subset of users before deployment

→ λ proxy integration & canary

* If some services are missing from X-ray Service map (also for identify potential bottlenecks)

→ Instrument the applicate by X-ray SDK

* to optimize S3 bucket for high request rates

→ use object key names across multiple prefixes

plaintext → ssm dynamic reference

secretstring → ssm secure dynamic reference

* In RDS, 2 AZ question

1. ALB
2. Port 443 - default authentication.

* ASL orderSource as partition key, mobile app as key

Same partkey & Diff. sortkey → LSI
Both different → GSI

* To ensure that no sessions are lost if EC2 instance fails
  → use DynamoDB.

* request → json [200], response is html.

* To access λ fn over HTTP
  1. function URLs
  2. API Gateway.

* aws λ invoke command to test in CI CD Process