Serverless Solutions on AWS

Nguyen Pham Luan Tien – NashTech Vietnam

May 2023

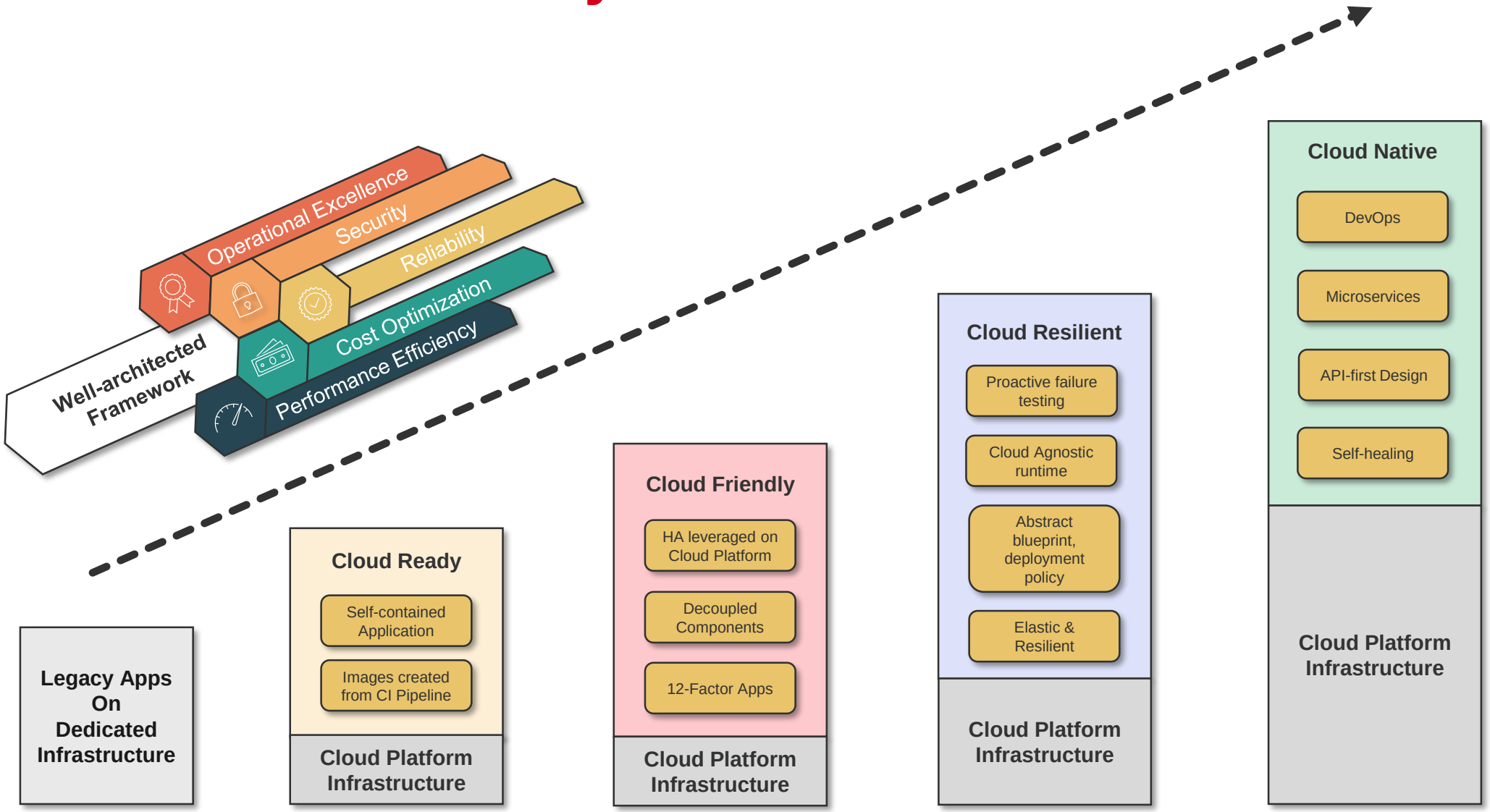
For NashTech Viet Nam internal circulation only

Agenda

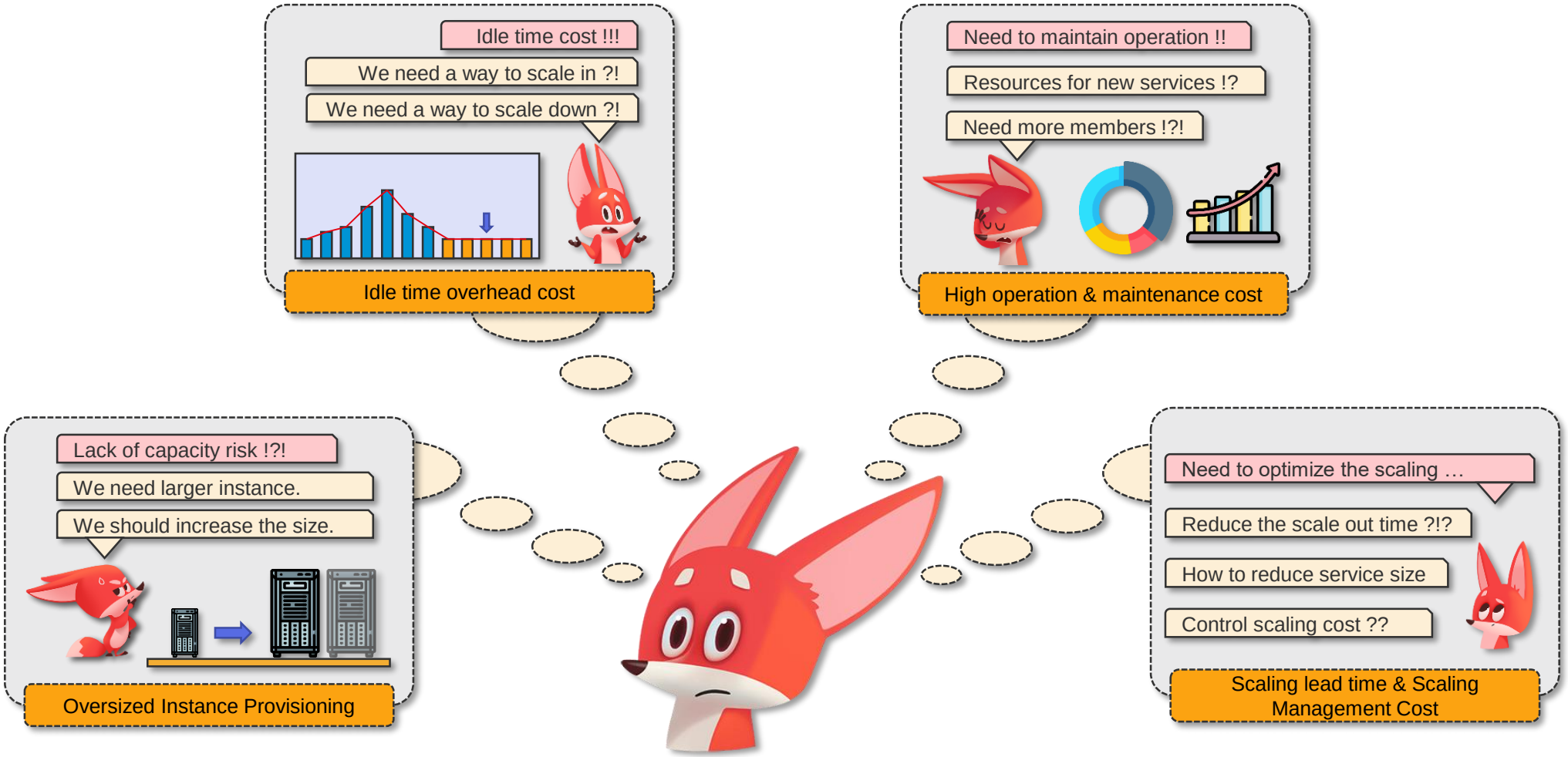
1. Why Serverless
2. What is Serverless
3. How to adapt Serverless in AWS
4. Pattern Samples

Why Serverless

Cloud-Native Maturity Model



Server hosting Challenges



What is Serverless

Serverless is …

- **Cloud-native** development **model approach**
- Build and run apps **without** having to **manage servers**

Serverless has …

Advantages

- **Scales** on demand **automatically**
- **No** more **Idle time cost**
- **Eliminates server maintenance** and **operation**
- Inherently **scalable** with **small** a **light-weight services**
- **Reduced** packaging and **deployment complexity**
- **Shorten delivery life-cycle**, focus right on business

There are …

Considerations

- **Vendor lock-in**, depends on **cloud provider ecosystem**
- Dealing with **cold starts**
- **Testing and debugging** become more **challenging**
- **Not** built for **long-running processes**
- **Source code management** strategy
- **Less server control capability**

How to adapt Serverless in AWS

Popular serverless services

Popular

- CloudFront
- API Gateway
- Lambda Function
- DynamoDB
- Amazon Aurora
- Amazon Cognito
- Simple Storage Service

Integration

- Simple Notification Service
- Simple Queue Service
- Step Function

Monitoring & Analytic

- CloudWatch
- AWS X-Ray
- Kinesis
- Athena

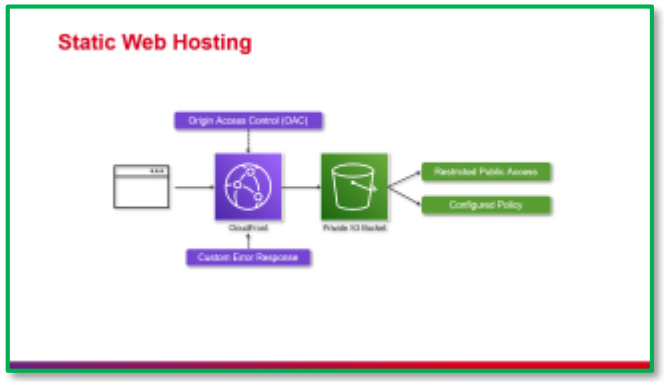
Others

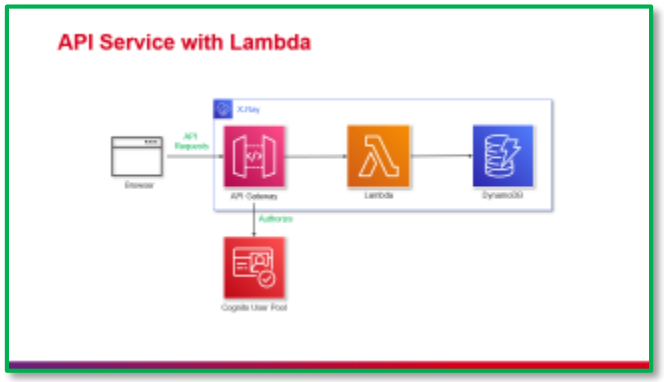
Containerization

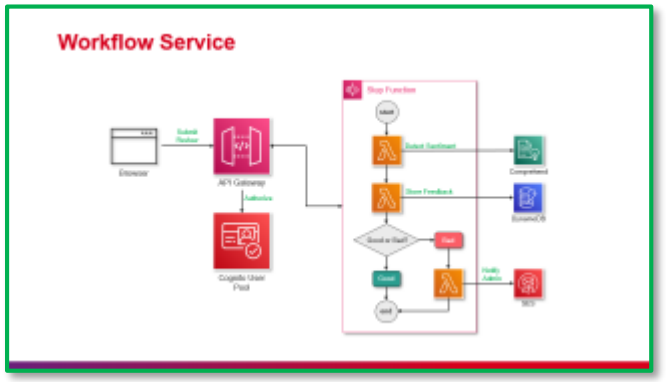
- Elastic Kubernetes Service
- Elastic Container Service
- \+
- Fargate

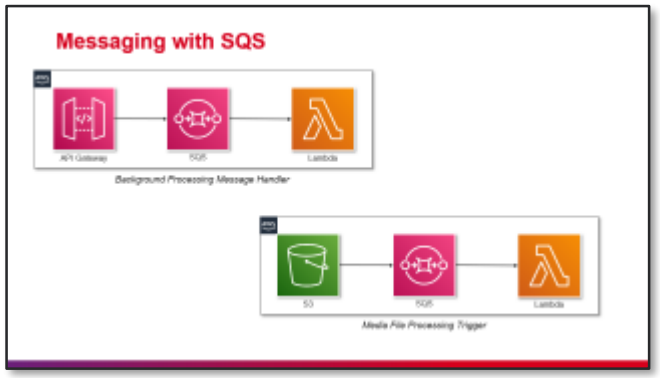
- CodeBuild
- CodeDeploy
- CodePipeline

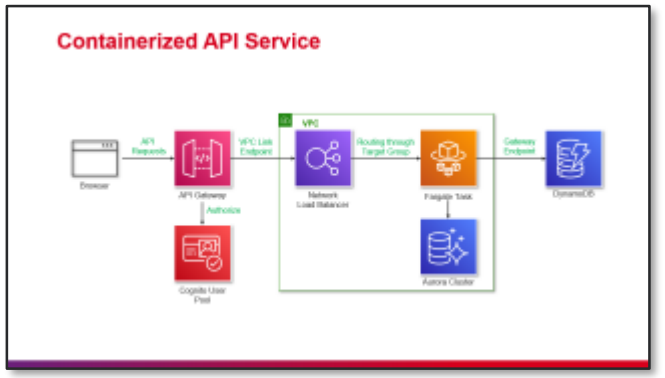
Common Patterns





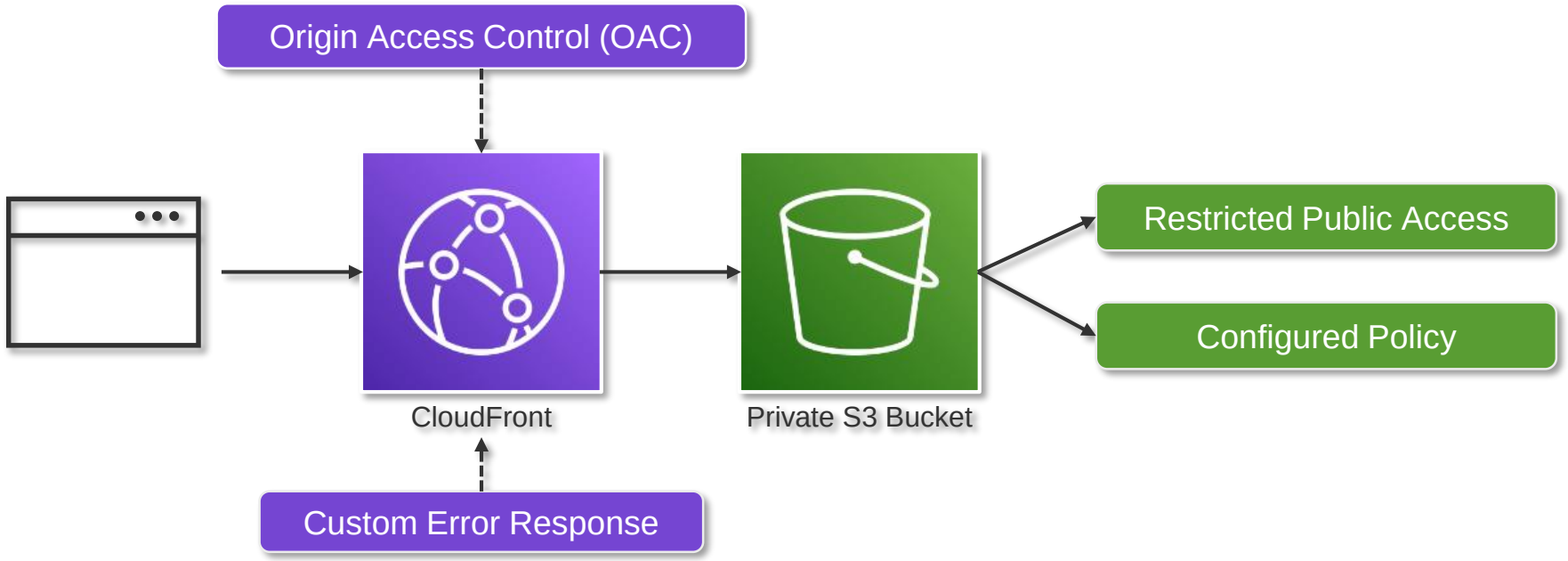




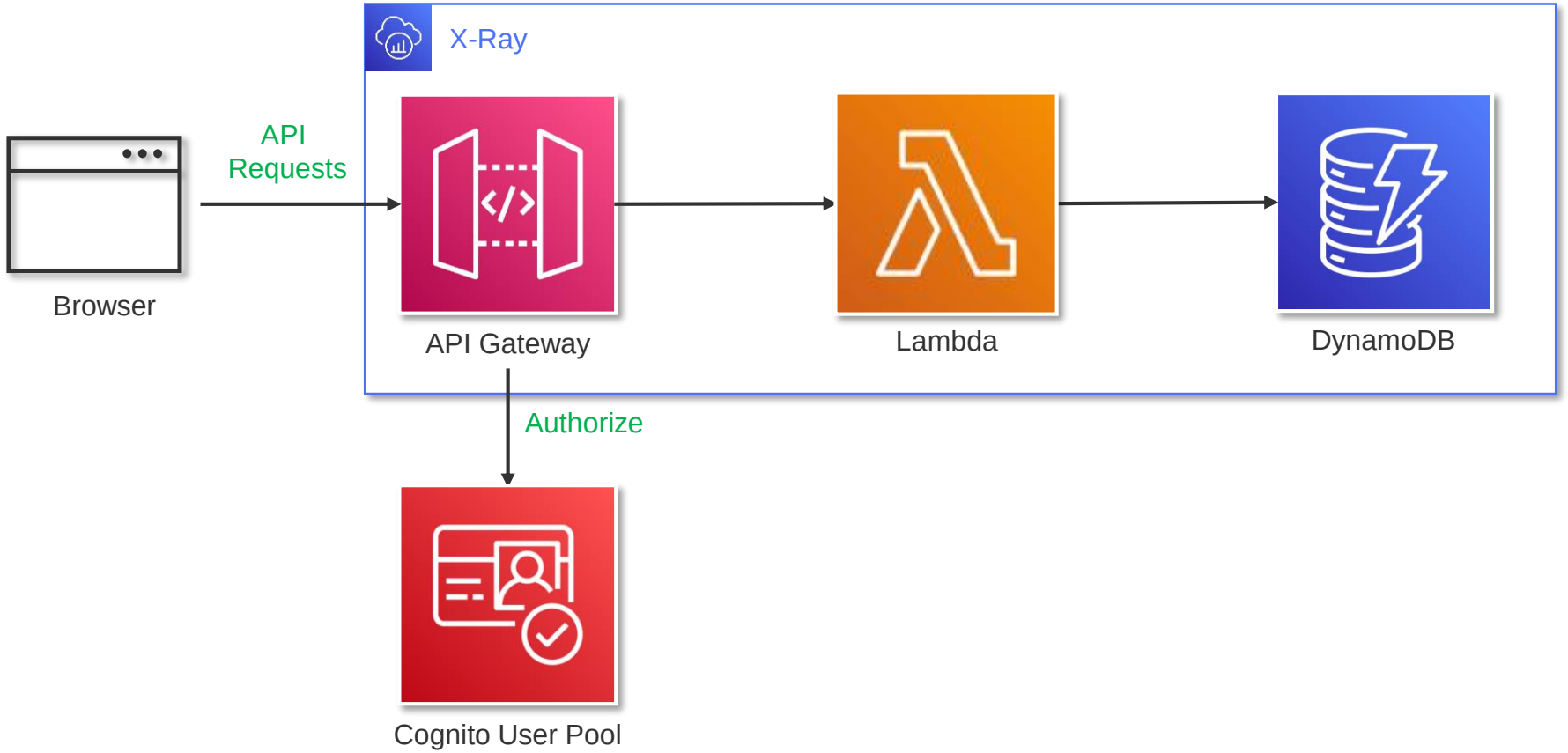


[Serverless patterns | Serverless Land]

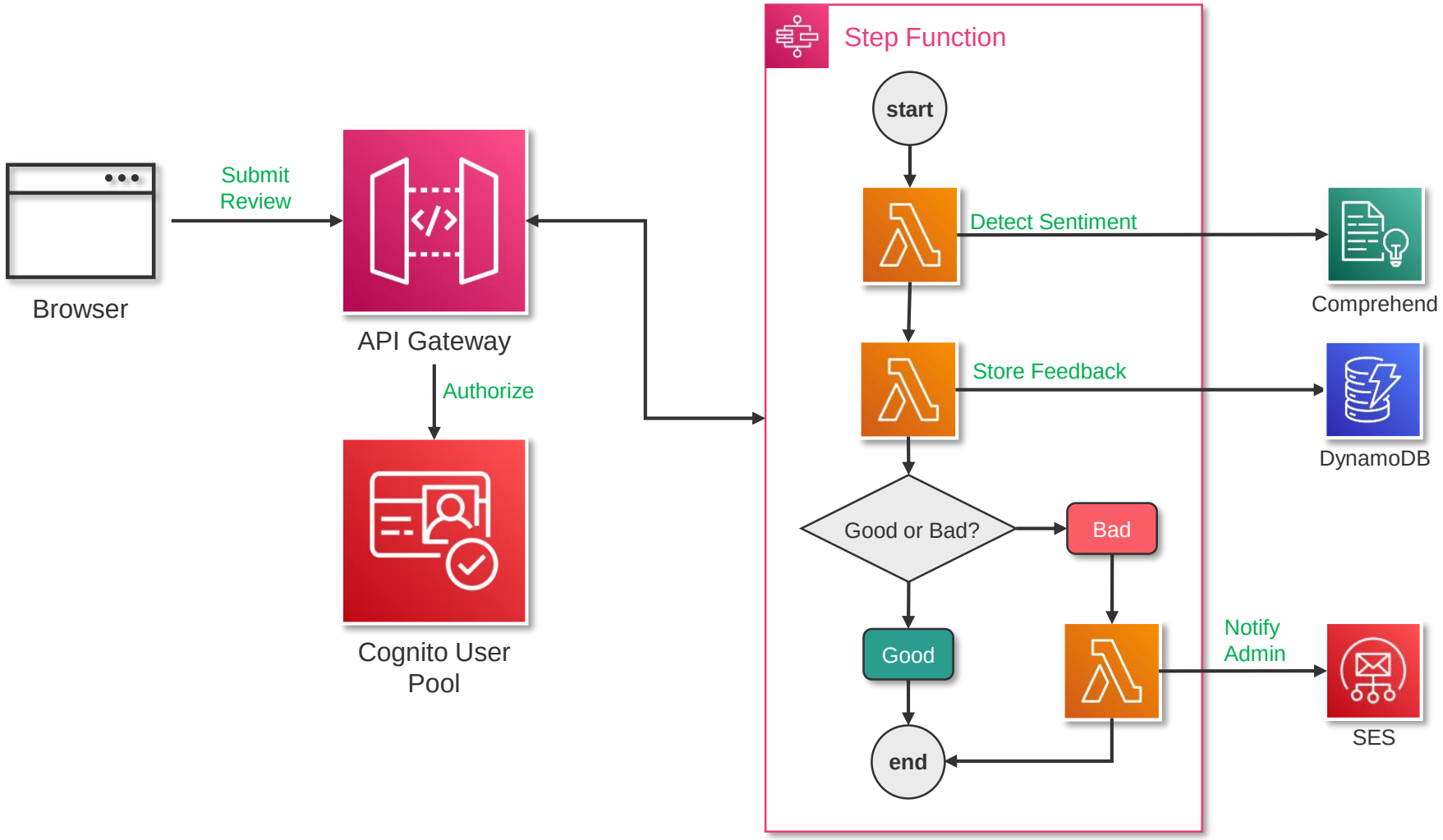
Static Web Hosting



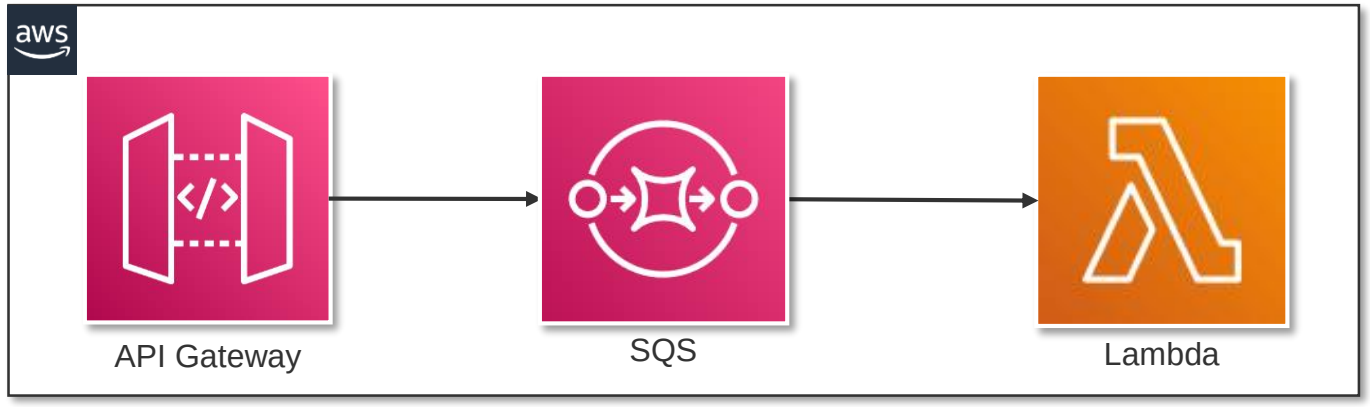
API Service with Lambda



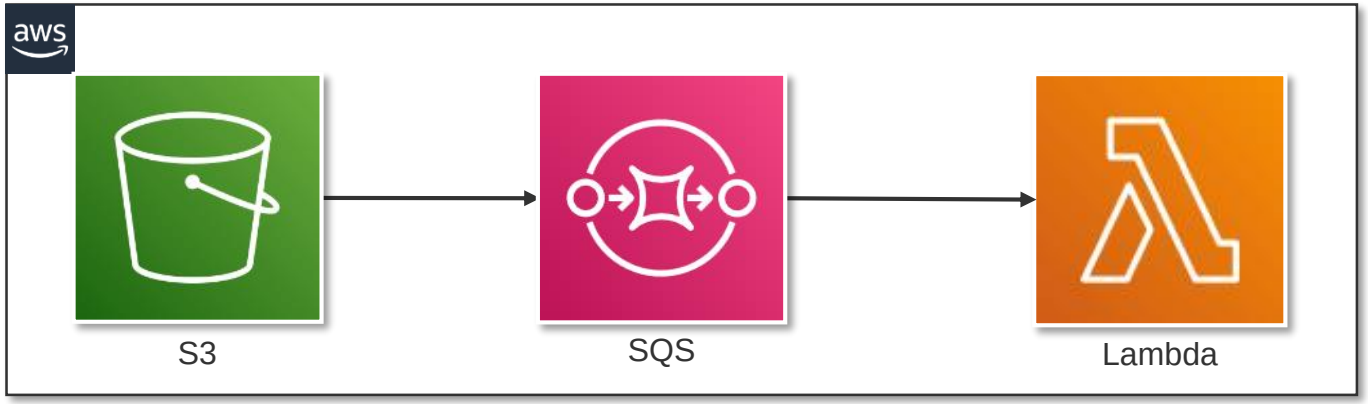
Workflow Service



Messaging with SQS

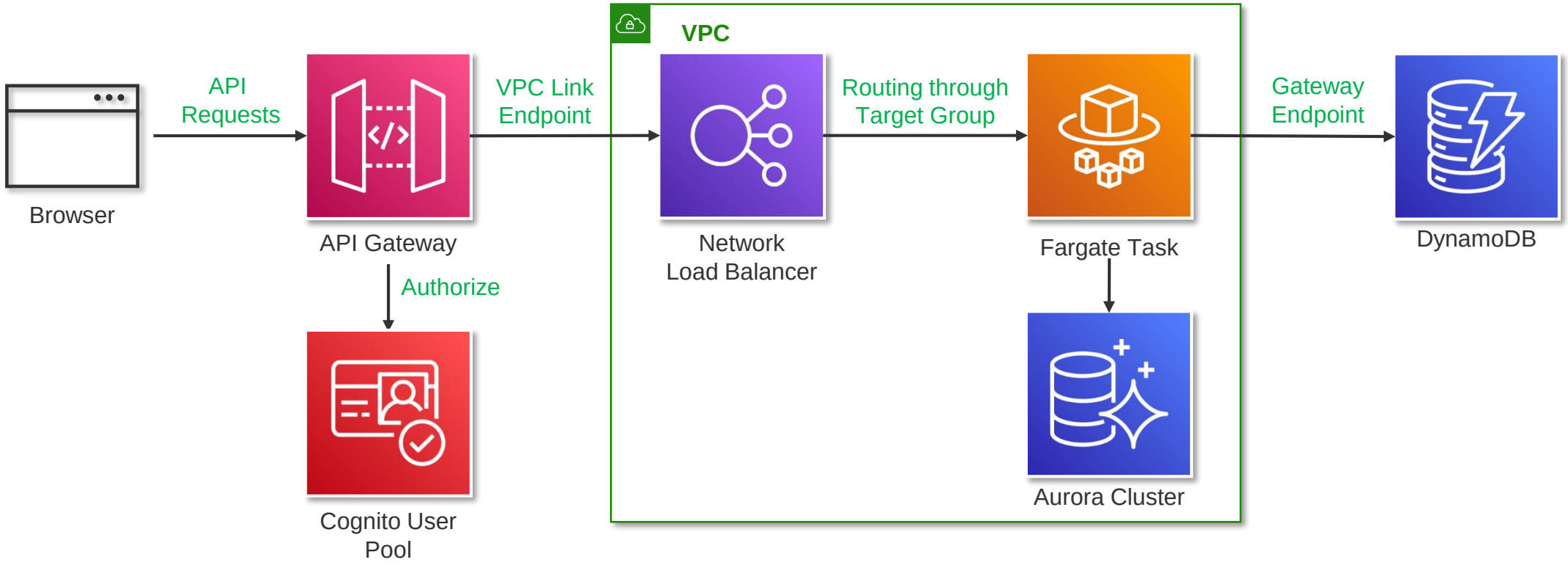


Background Processing Message Handler



Media File Processing Trigger

Containerized API Service



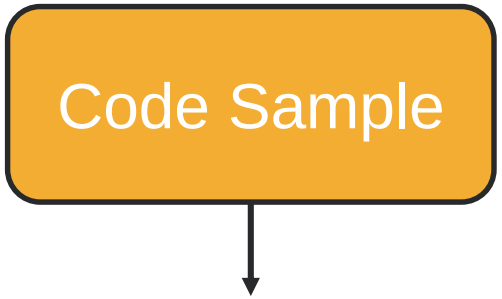
Pattern Samples

Prerequisites

Install or update the latest version of the AWS CLI - AWS Command Line Interface (amazon.com)

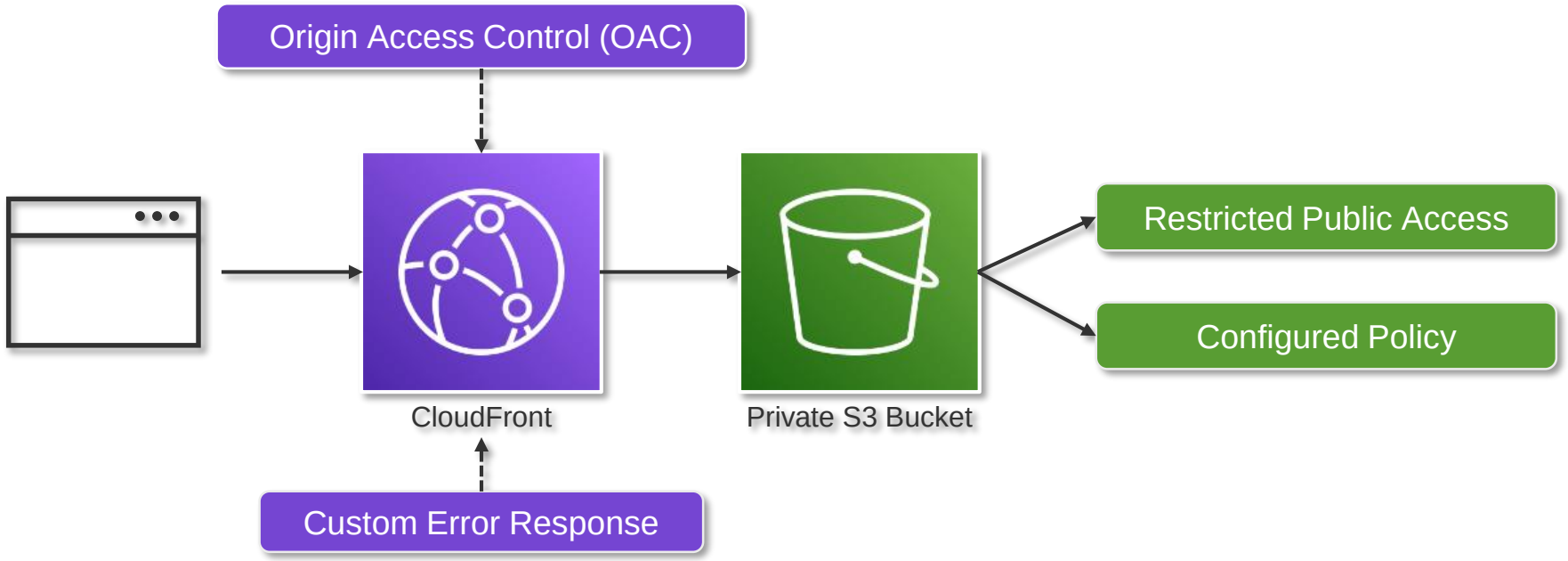
Installing the AWS SAM CLI - AWS Serverless Application Model (amazon.com)

Getting started with the AWS CDK - AWS Cloud Development Kit (AWS CDK) v2 (amazon.com)



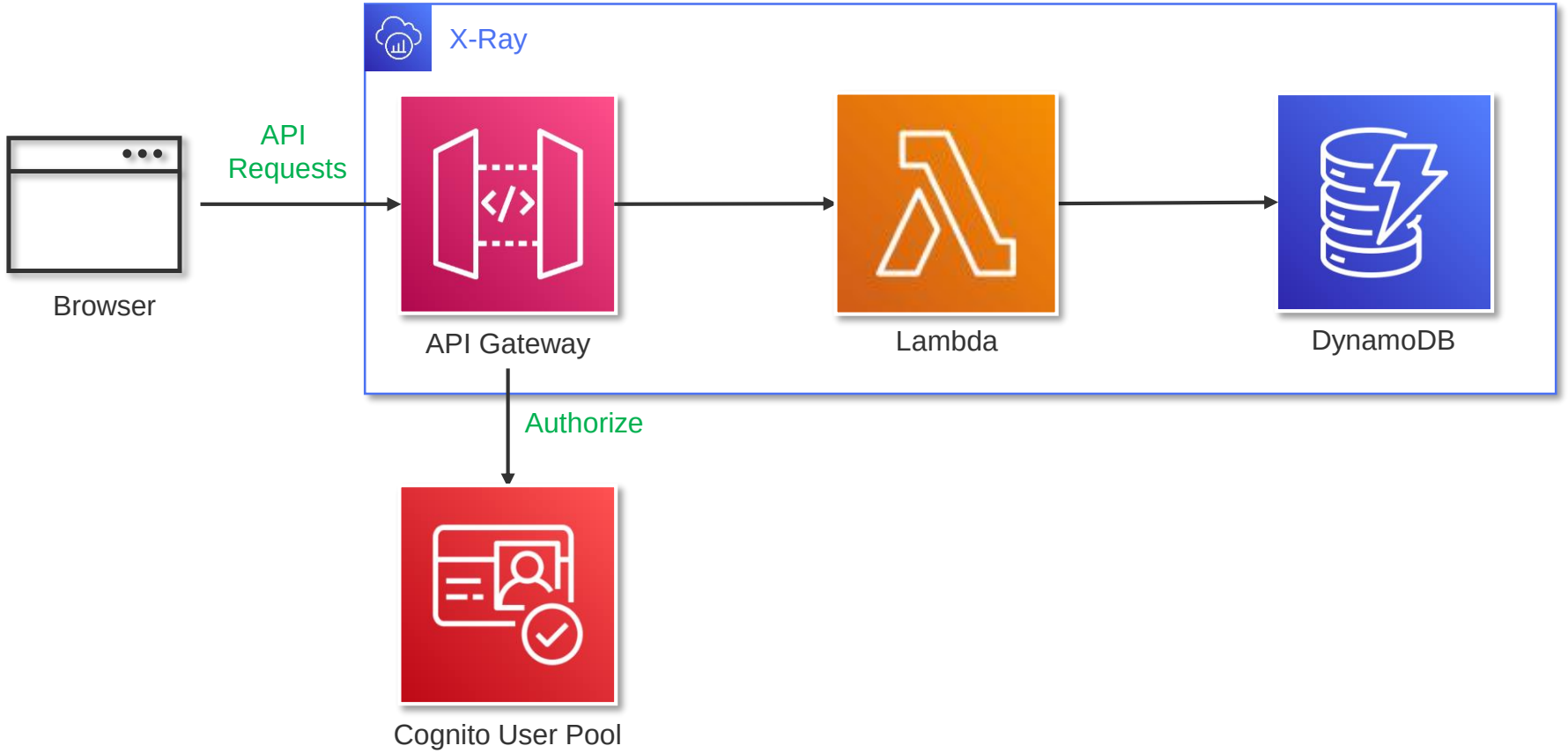
luantien/aws-serverless-samples (github.com)

UI Service


Origin Access Control (OAC)
CloudFront
Private S3 Bucket
Restricted Public Access
Configured Policy
Custom Error Response

Sample Time

API Service with Lambda



DynamoDB

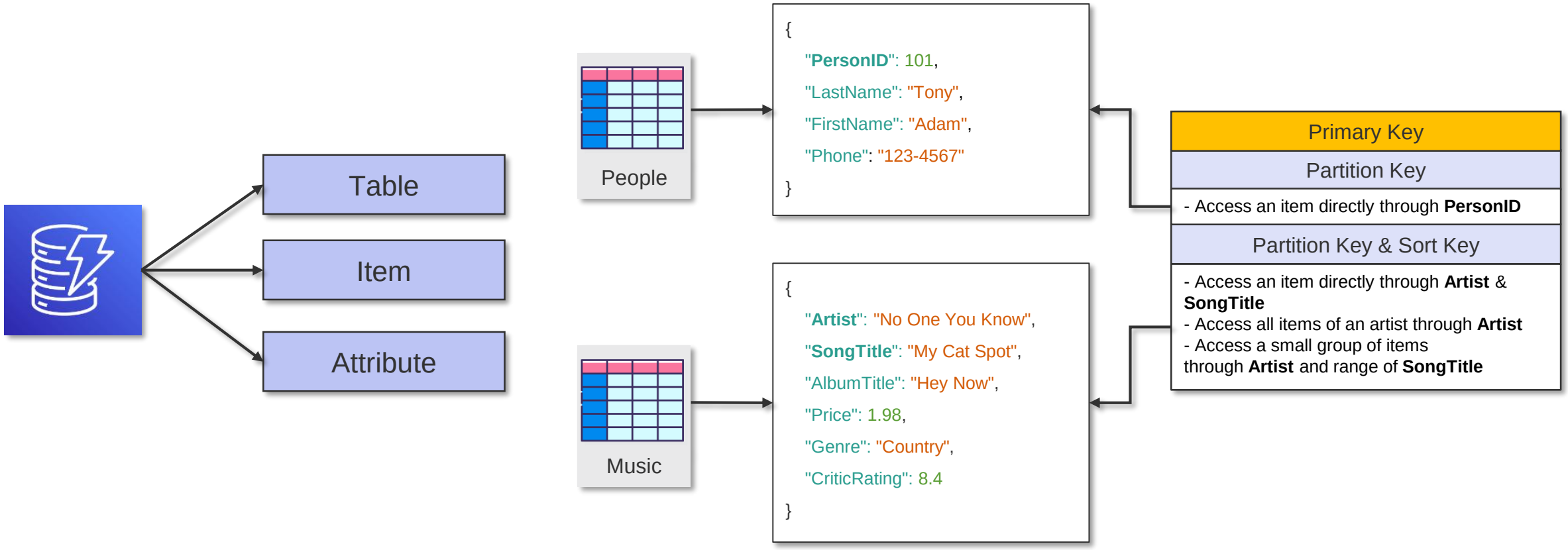
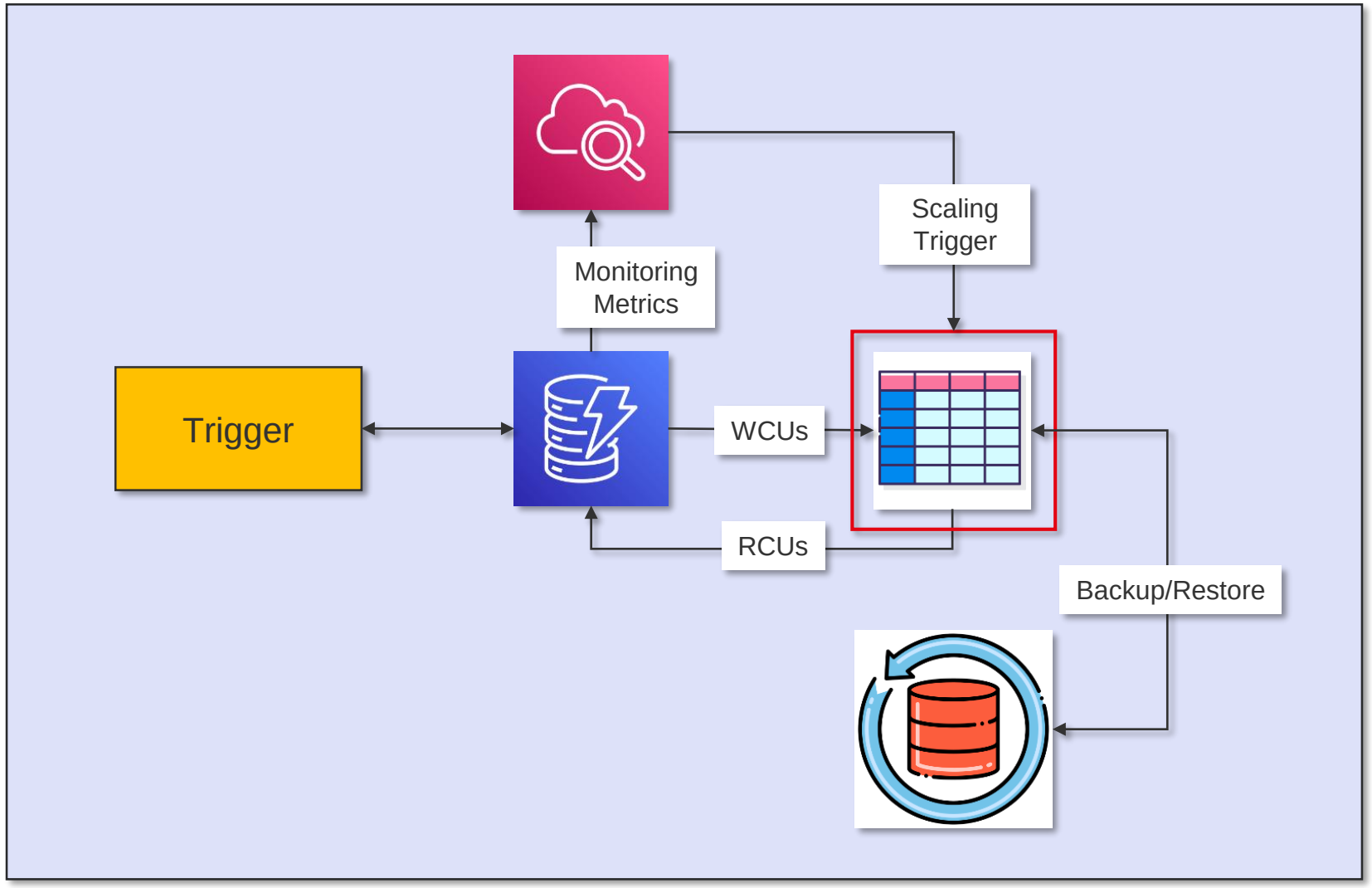


Table
Item
Attribute
People
Music
{
"PersonID": 101,
"LastName": "Tony",
"FirstName": "Adam",
"Phone": "123-4567"
}
{
"Artist": "No One You Know",
"SongTitle": "My Cat Spot",
"AlbumTitle": "Hey Now",
"Price": 1.98,
"Genre": "Country",
"CriticRating": 8.4
}
Primary Key
Partition Key
- Access an item directly through PersonID
Partition Key & Sort Key
- Access an item directly through Artist & SongTitle
- Access all items of an artist through Artist
- Access a small group of items through Artist and range of SongTitle

DynamoDB Operation

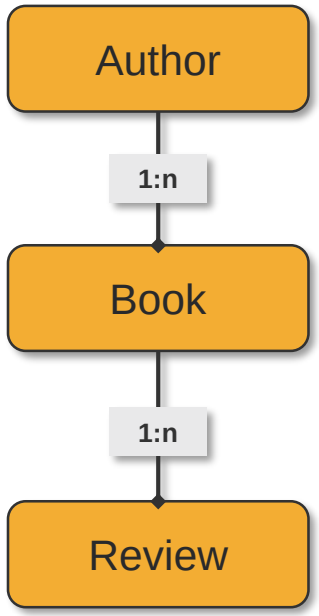


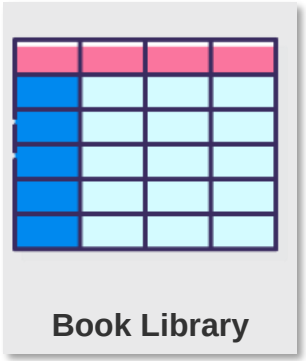
DynamoDB Pricing

Pricing	
On-demand	Provisioned
- Create new tables with u**nknown workloads.** Have **unpredictable** application **traffic** - Prefer the **ease of paying** for only **what you use**. - **Risk:** no threshold can lead to unexpected high payment amount.	- Have **predictable** application **traffic**. - Run applications whose traffic is consistent or ramps gradually. - Can **forecast** capacity requirements to **control costs.** - **Risk**: capacity limit should be monitoring properly otherwise it can impact application availability and user experience.

Billing		
Features	Billing Unit	Description
Write Request	Write request unit (**WCU**)	Write data to your table: - *Standard write* request unit can write an item up to **1 KB**. - Transactional write requires **double** more **WCU** than *standard write*. - Ex: Standard write for 3KB item is 3 WCUs. Transactional write for 3KB item is 6 WCUs.
Read Request	Read request unit (**RCU**)	Read data from your table: - *Strongly consistent* read of item up to 4 **KB** requires **1 RCU**. - *Eventually consistent* read of item up to **4 KB** requires **0.5 RCUs**. - *Transactional* read of item up to **4 KB** requires **2 RCUs**. Ex: Eventually consistent read of 8 KB item is 1 RCU. Transactional read of 8 KB item is 4 RCUs.
Data Storage	GB storage per month (**GB-month**)	- Raw byte size of your data + per-item overhead of 100 bytes for indexing in table. - Ex: your table occupies 25 GB at the beginning of the month, grows to 29 GB by the end of month (30 days). Average storage in 30 days is 27 GB. We will be charged 27-25=2GB.

DynamoDB Table Design





BookLibrary			
Access Patterns	Table/GSI	Key Conditions	Example
Get book for a given bookId	Table	PK=bookId and SK=bookId	PK="b#12345" and SK="b#12345"
Get author for a given authorId	Table	PK=authorId and SK=authorId	PK="a#12345" and SK="a#12345"
Get all reviews for a given bookId	Table	PK=bookId and SK begins_with "r#"	PK="b#12345" and SK begins_with "r#"
Get all books for a given authorId	GSI1	PK=authorId and SK begins_with "b#"	PK="a#12345" and SK begins_with "b#"

Primary Key		Atrributes								
PK	SK									
b#12345	b#12345	EntityType	AuthorId (GSI1-PK)	PublishedDate	Title	Description	Price	Positive	Negative	Neutral
		book	a#12345	7/10/2022	book title 1	description 1	100	0	0	0
	r#12345	EntityType	Reviewer	Title	Description					
		review	Tony Adam	Bravo book	This is a very nice book to read					
a#12345	a#12345	EntityType	Name							
		author	author 1							

DynamoDB Table Design

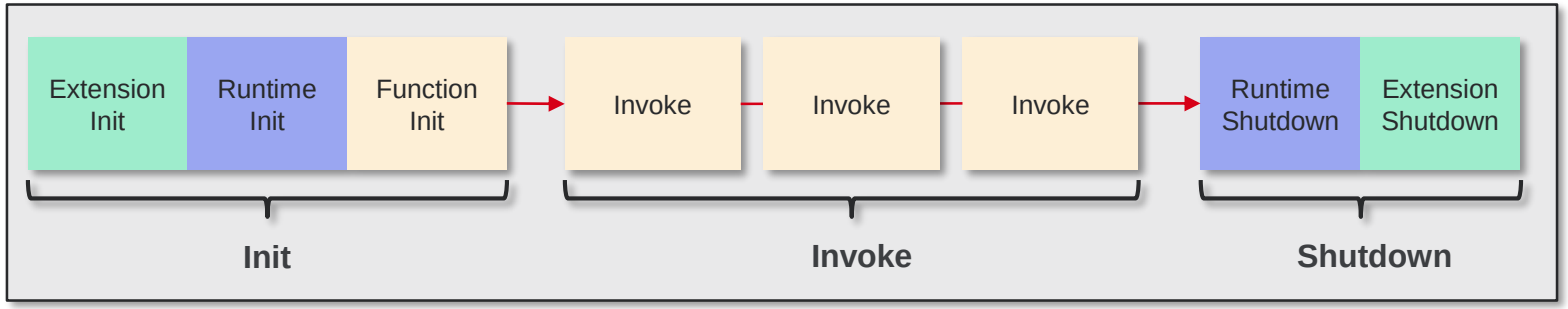
BookLibrary GSI: AuthorIndex

Primary key		Attributes				
Partition key: PK	Sort key: SK					
a#1	a#1	EntityType	Name			
		author	Author 1			
b#2	b#2	AuthorId	EntityType	Description	PublishedDate	Title
		a#1	book	Description 2	2019-08-01	Book 2
b#3	b#3	AuthorId	EntityType	Description	PublishedDate	Title
		a#2	book	Description 3	2018-05-22	Book 3
b#1	b#1	AuthorId	EntityType	Description	PublishedDate	Title
		a#1	book	Description 1	2018-01-01	Book 1
b#6	b#6	AuthorId	EntityType	Description	PublishedDate	Title
		a#3	book	Description 6	2020-09-03	Book 6
a#3	a#3	EntityType	Name			
		author	Author 3			
b#5	b#5	AuthorId	EntityType	Description	PublishedDate	Title
		a#3	book	Description 5	2018-11-04	Book 5
a#2	a#2	EntityType	Name			
		author	Author 2			
b#4	b#4	AuthorId	EntityType	Description	PublishedDate	Title
		a#2	book	Description 4	2021-07-20	Book 4

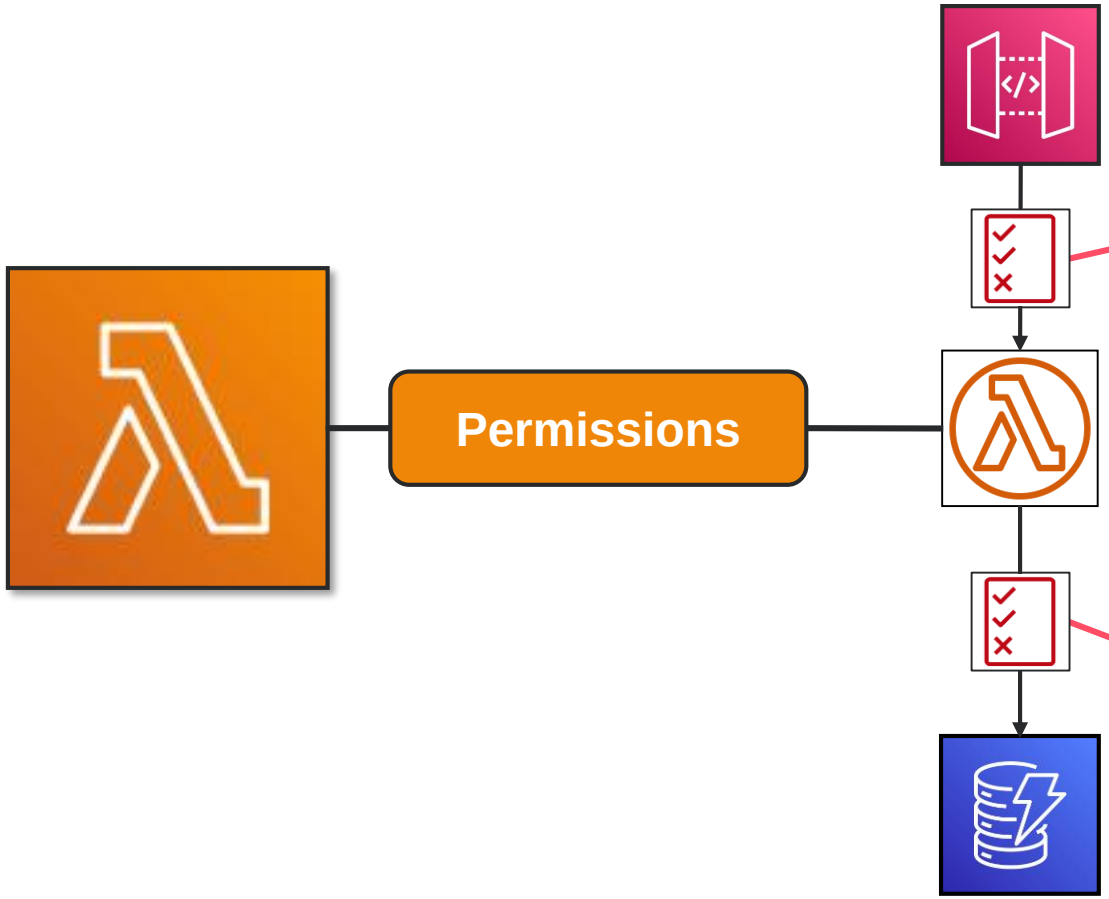
Lambda Function

Invocations

- **Synchronous**
  - Run the function and waits for response
  - No built-in retries
  - API Gateway, CloudFront, Cloud Formation
- **Asynchronous**
  - The events are queued, no need to wait for completion
  - Use *destinations* to send *invocation records* to other services
  - Built in – retries twice
  - SNS, SQS, EventBridge
- **Pooling**
  - Designed for integration with AWS streaming or queue-based services
  - Retries – Depends on event source
  - DynamoDB Stream, SQS, Kinesis



Lambda Function Permission



IAM Resource Policy

- Tells Lambda which *principals have permission* to *invoke function*.
- Principal can be a user, role, other AWS service, another AWS account.

IAM Execution Role

- *Gives* the function *permission* to *interact* with *other services*. (*E.g., permission to read/write on a specific DynamoDB table*).
- *Lambda* will *assume the role* when the *function* is *invoked*.
- The role must include a policy to allows Lambda service to "AssumeRole".

Lambda Configuration

Configurations

Memory

- Can allocate up to ***10 GB of memory***.
- Lambda ***allocates CPU*** and ***other resources linearly***.
- Additional costs for using more memory may be offset by lower duration.

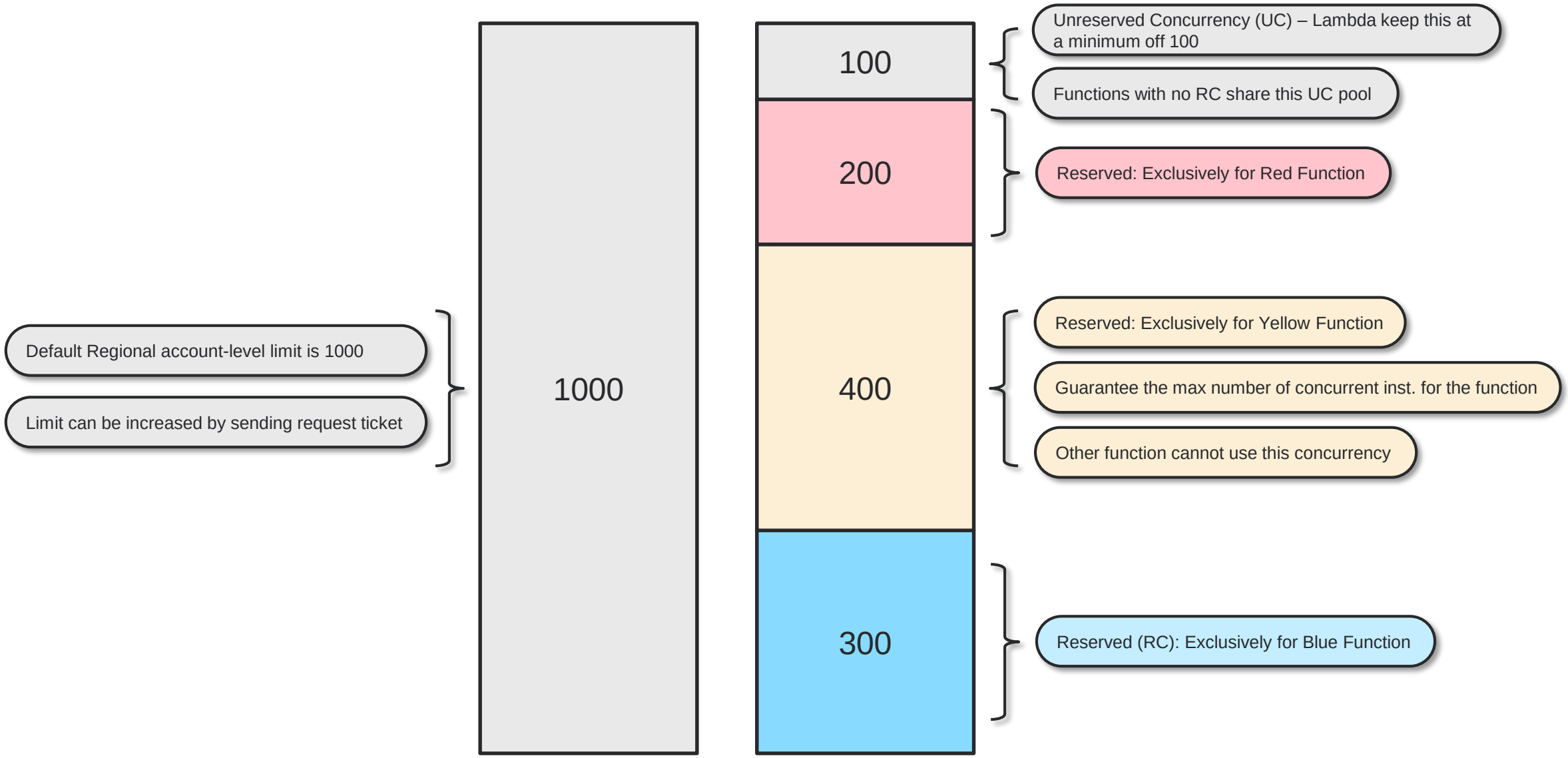
Timeout

- ***Dictates how long*** a ***function can run*** before terminated.
- ***Maximum timeout*** for a function is ***900 seconds*** (15 minutes).
- ***Avoiding lengthy timeouts*** for functions can prevent you from being billed.

Concurrency

- The ***number of invocations*** that ***function runs*** at any ***given moment***.
- When a function is invoked, Lambda launches an instance of the function to process the event.
- ***Types***: Unreserved Concurrency, Reserved Concurrency, Provisioned Concurrency

Lambda Concurrency Samples



Lambda Deployment

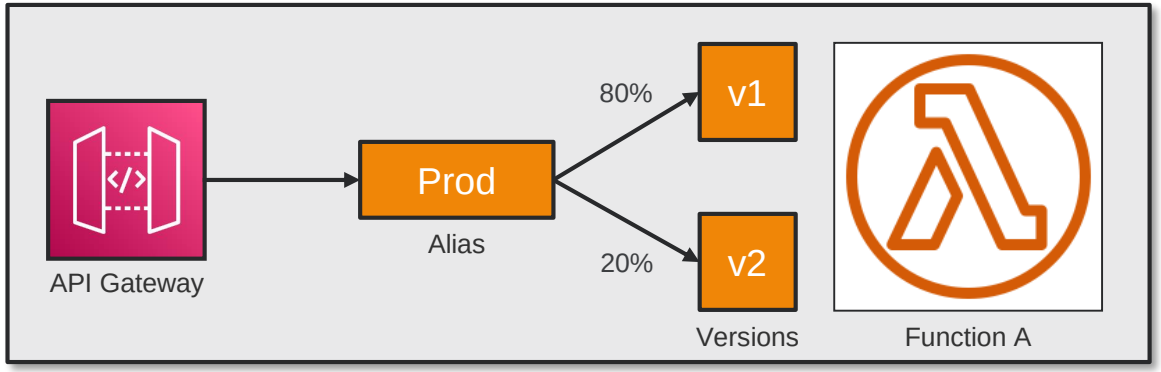
Deployments

Alias

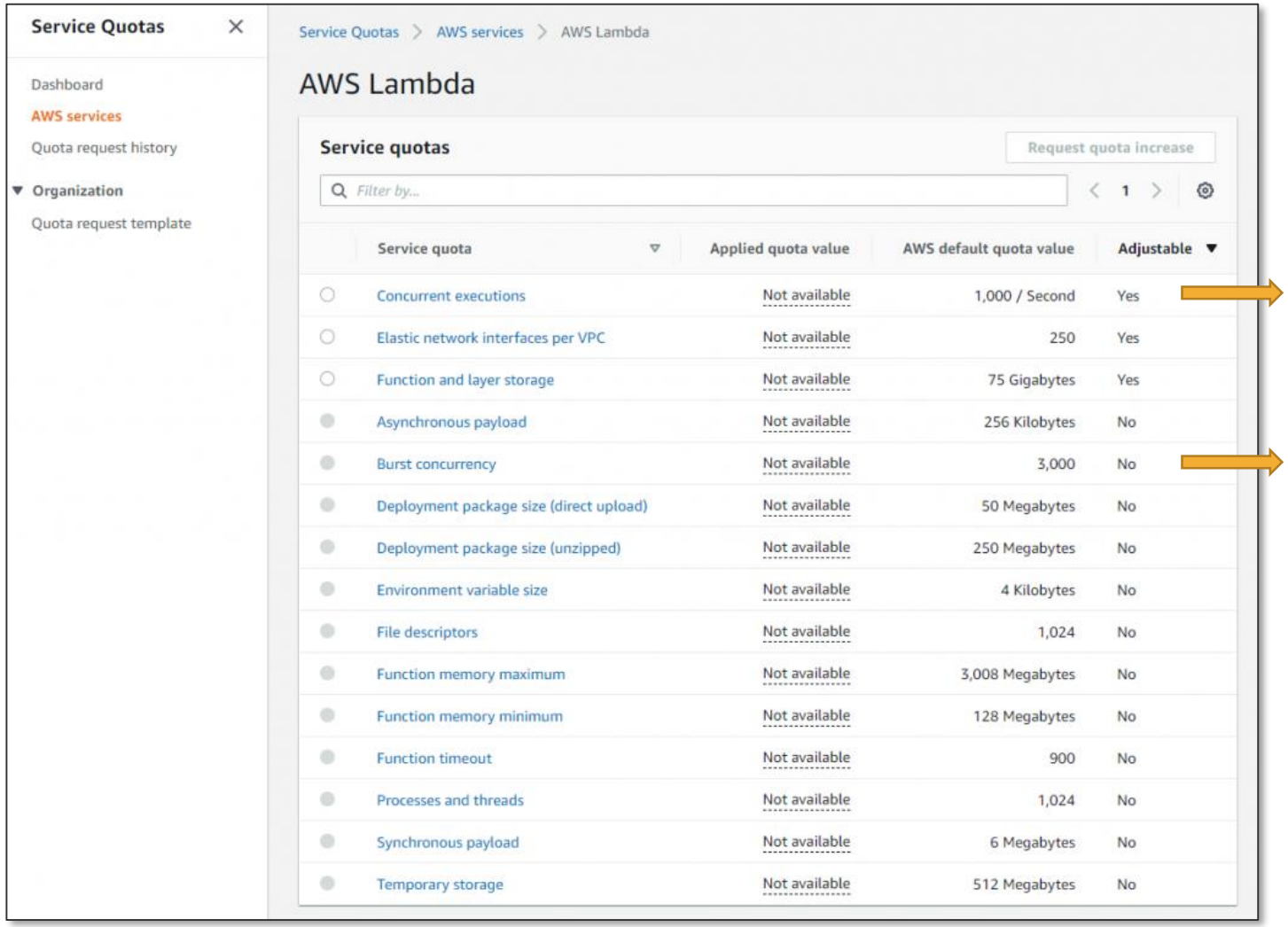
- Act like ***a pointer*** to a specific ***function version***.
- Each alias has ***a unique ARN***.
- Alias can point only to a function version, not another alias.
- Specify alias ARN in resource policy, then we do not need to update the policy when function version changes.

Version

- Can be used to manage function's deployment.
- A function version includes:
  - Function code, associated dependencies, settings (inc. env variables).
  - Lambda runtime identifier and runtime version.
  - Function settings, including environment variables.
  - A ***unique ARN*** to identify the specific function version



Lambda Scaling and Quota

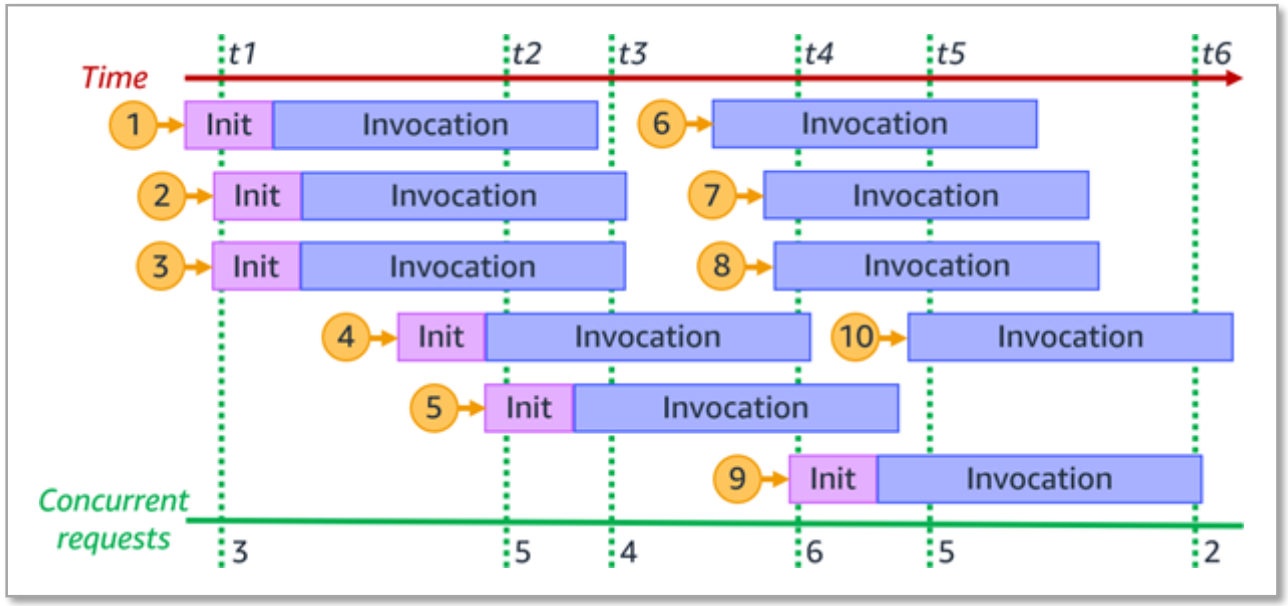


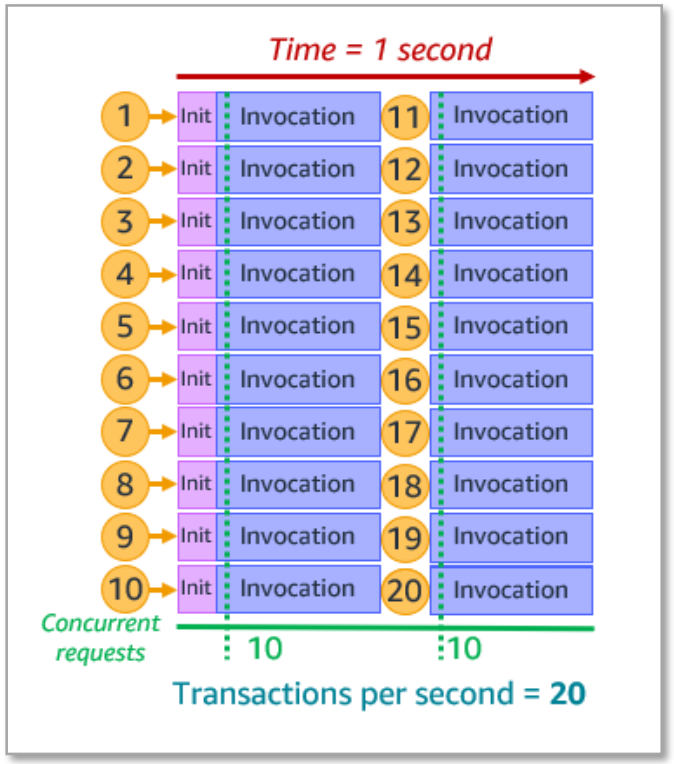
Up to 10,000 per region

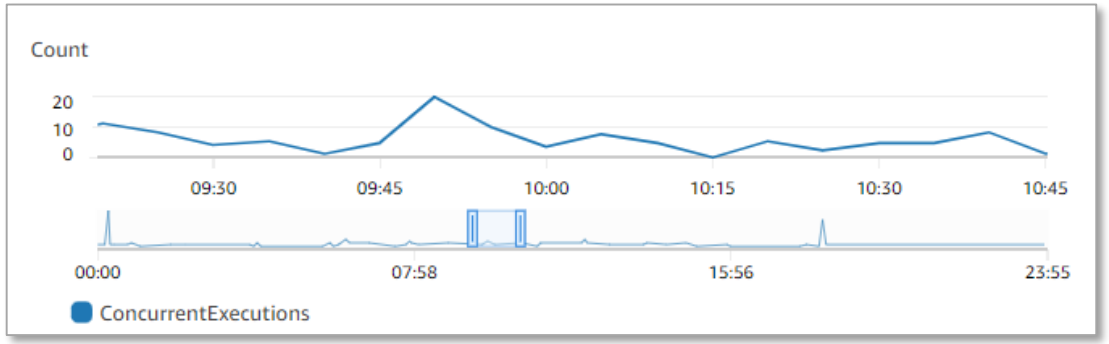
- **3000** – US West (Oregon), US East (N. Virginia), Europe (Ireland)
- **1000** – Asia Pacific (Tokyo), Europe (Frankfurt), US East (Ohio)
- **500** – Other Regions

[Lambda quotas - AWS Lambda (amazon.com)]

Lambda Scaling and Quota



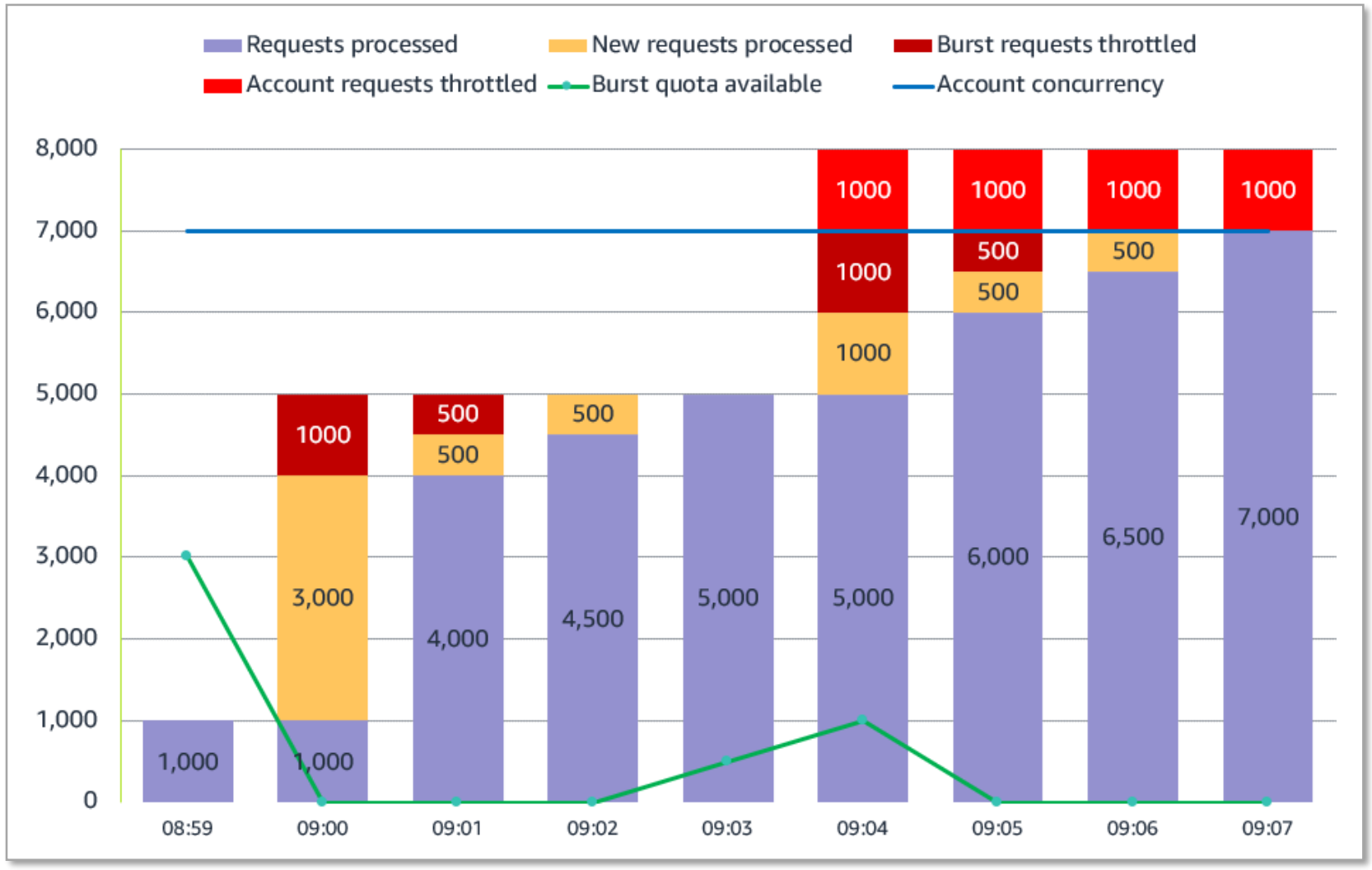




RequestsPerSecond x *AvgDurationInSeconds* = *concurrent requests*

100 requests/second x *0.5 sec* = *50 concurrent requests*

Lambda Scaling and Quota



[Understanding AWS Lambda scaling and throughput | AWS Compute Blog (amazon.com)]

08:59
- Using **1,000** concurrent **runtime environments**.
- Lambda invocation takes **250ms.**
- **Transactions per second (TPS)** are **4,000**.

09:00
- Large traffic with **5,000 sustained requests**.
- **1000 requests executed** by existing runtime environments.
- Lambda uses the **3,000 available burst concurrency** to create new environments to handle the additional load.
- **1,000 requests** are **throttled** as there is not enough burst concurrency to handle all 5,000 requests.
- **TPS** are **16,000**.

09:01
- Lambda scales by another **500 concurrent invocations per minute**.
- **500 requests** are still throttled.
- App can now handle **4,500 concurrent requests**.

09:02
- Lambda scales by another **500 concurrent invocations per minute**.
- No requests are throttled.
- App can now handle all **5,000 concurrent requests**

09:03
- The application continues to handle the **sustained 5000 requests**.
- The **burst concurrency quota** rises to **500**.

09:04
- The application continues to handle the **sustained 5000 requests**.
- The **burst concurrency quota** rises to **500**.

09:05
- Lambda scales by another **500 concurrent requests**.
- The application can now handle **6,500 requests**.
- **500 requests** are **throttled** as there is **not enough burst concurrency**.
- **1,000 requests** are still **throttled** as the **account concurrency quota** has been **reached**.

Lambda Pricing

Allocated memory: 512 MB

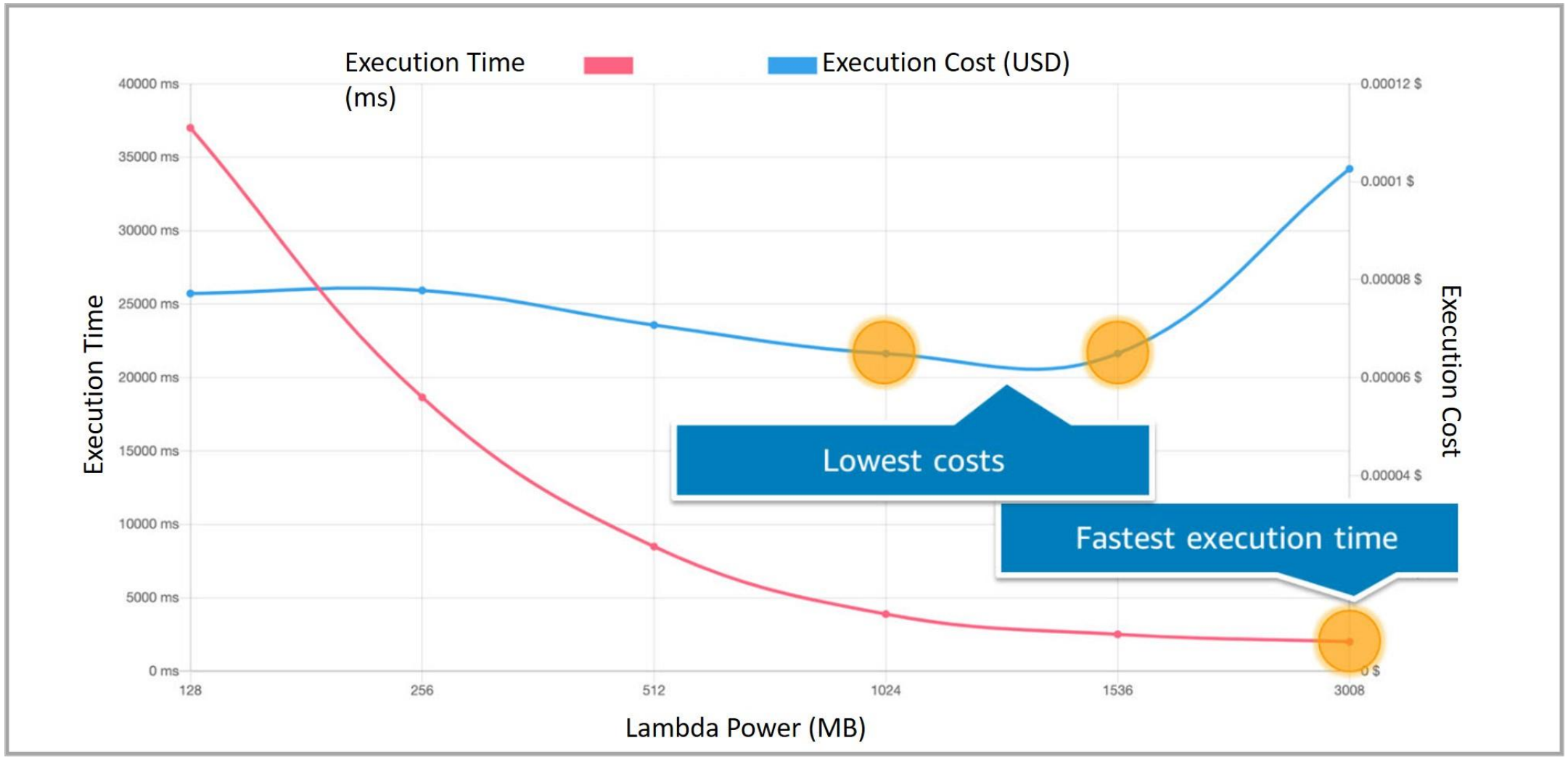
No. Invocations: 20,000 times/month

Execution duration: 1sec

AWS LAMBDA	
GB-sec 20,000 * 512/1024	10,000 GB-sec
Compute Charged 10,000 * 0.00001667	$0.1667
Requests Charged (20,000/1 Mil) * 0.2 per Mil	$0.004
TOTAL	**$0.1707**

EC2 (on-demand t2.nano)	
$0.0081*24*30	$5.832
TOTAL	**$5.832**

Lambda Pricing

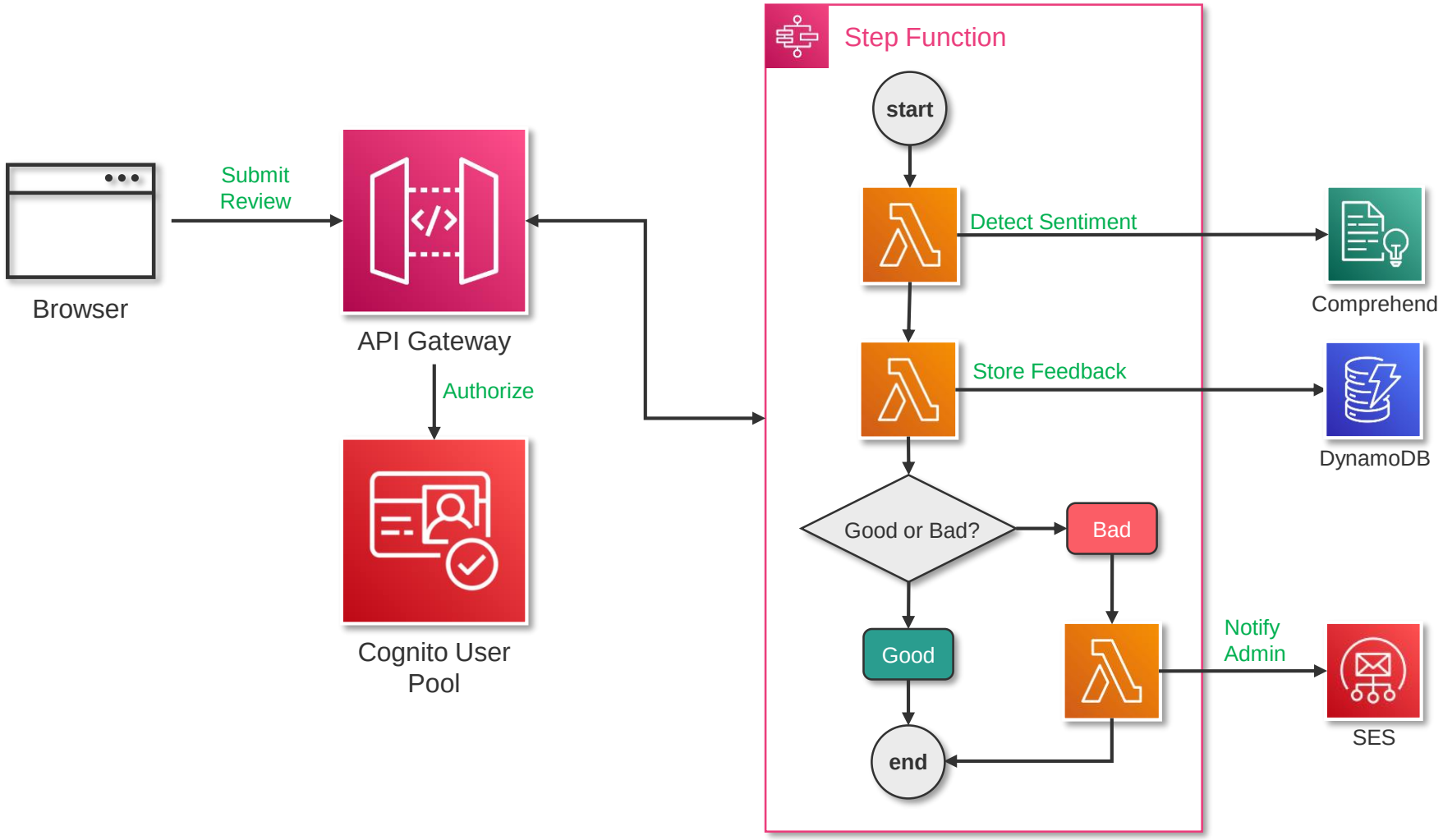


Lambda Power Tuning provides a comparison of cost vs. speed of your Lambda function

Source: AWS Skill Builder

Sample Time

Workflow Service (Sync)



State Machine Type

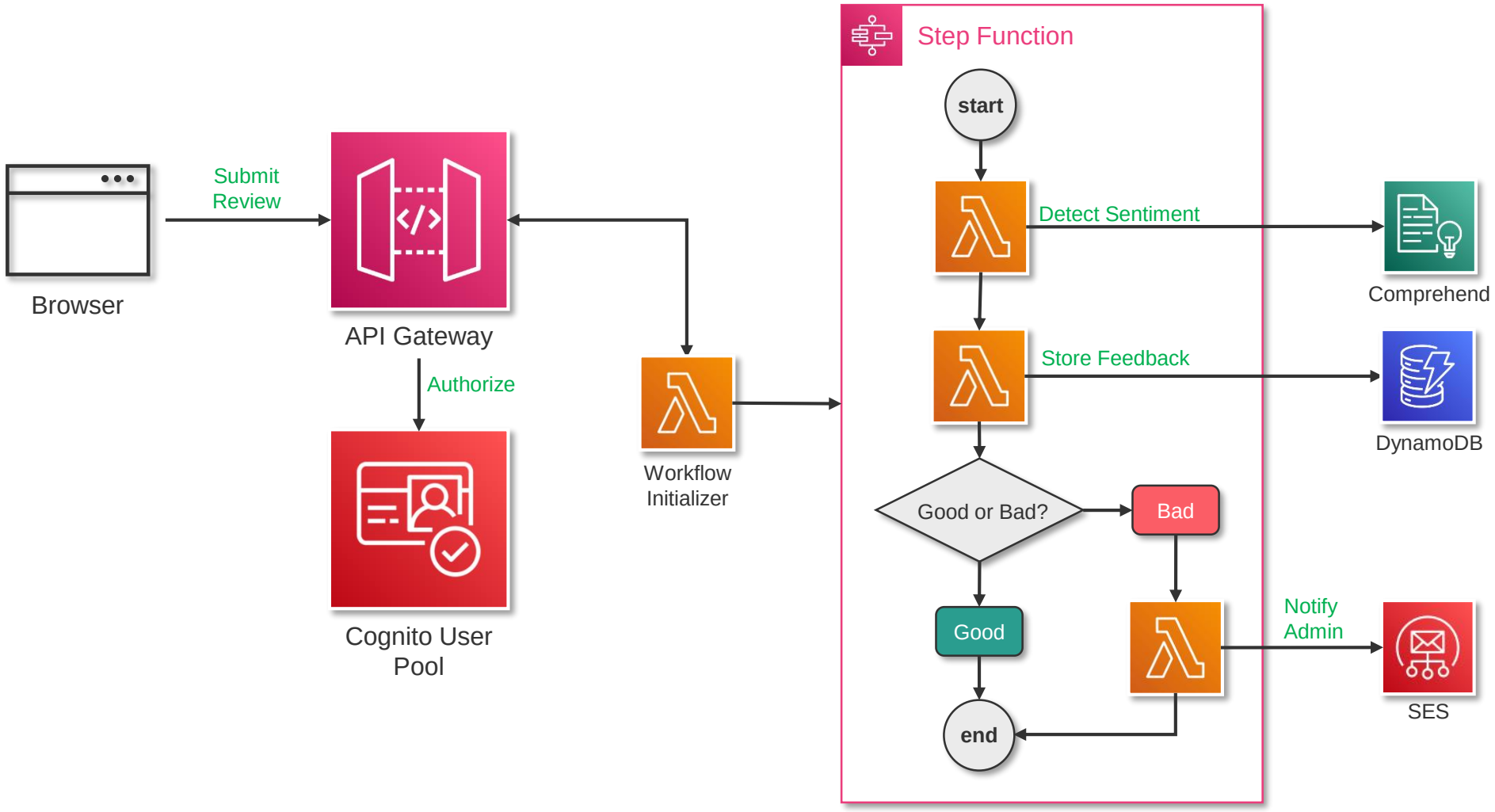
Standard vs Express Workflows

	Standard Workflows	Express Workflows: Synchronous and Asynchronous
Maximum duration	1 year.	5 minutes.
Supported execution start rate	Over 2,000 per second	Over 100,000 per second
Supported state transition rate	Over 4,000 per second per account	Nearly unlimited
Pricing	Priced per state transition. A state transition is counted each time a step in your execution is completed.	Priced by the number of executions you run, their duration, and memory consumption.
Execution history	Executions can be listed and described with Step Functions APIs, and visually debugged through the console. They can also be inspected in CloudWatch Logs by enabling logging on your state machine.	Unlimited execution history, that is, as many execution history entries are maintained as you can generate within a 5-minute period. Further, executions can be inspected in CloudWatch Logs by enabling logging on your state machine.
Execution semantics	Exactly-once workflow execution.	*Asynchronous Express Workflows*: At-least-once workflow execution. *Synchronous Express Workflows*: At-most-once workflow execution.
Service integrations	Supports all service integrations and patterns.	Supports all service integrations. **Note** Express Workflows do not support Job-run (.sync) or Callback (.waitForTaskToken) service integration patterns.
Step Functions activities	Supports Step Functions activities.	Does not support Step Functions activities.

State Machine Type

Standard Workflows	Asynchronous Express Workflows	Synchronous Express Workflows
Exactly-once workflow execution	At-least-once workflow execution	At-most-once workflow execution
The execution state is internally persisted on every state transition. To guarantee that only one workflow with the same name can run, Step Functions will return an idempotent response when you start a Standard Workflow with the same name as an already running workflow. In this case, Step Functions will not start a new workflow. When the workflow completes, Step Functions will respond with an exception. After 90 days, the workflow data will be removed, and the name can then be reused.	No internally persisted state for workflow progress. If you attempt to start an Express Workflow with the same name more than once, each attempt causes a workflow to start concurrently. In rare cases, the internal state of a workflow can be lost, and the workflow will be automatically restarted from beginning. You should ensure your state machine logic is idempotent and should not be affected adversely by multiple concurrent executions of the same input.	After a workflow starts, Step Functions will wait and returns the result as part of the API response. If service exceptions occur, Step Functions will not restart from the beginning. You should ensure your state machine logic is idempotent and should not be affected adversely by multiple concurrent executions of the same input.

Workflow Service (Async)



Sample Time

Thank you

Nash Tech.