AWS Lambda

It's a serverless world

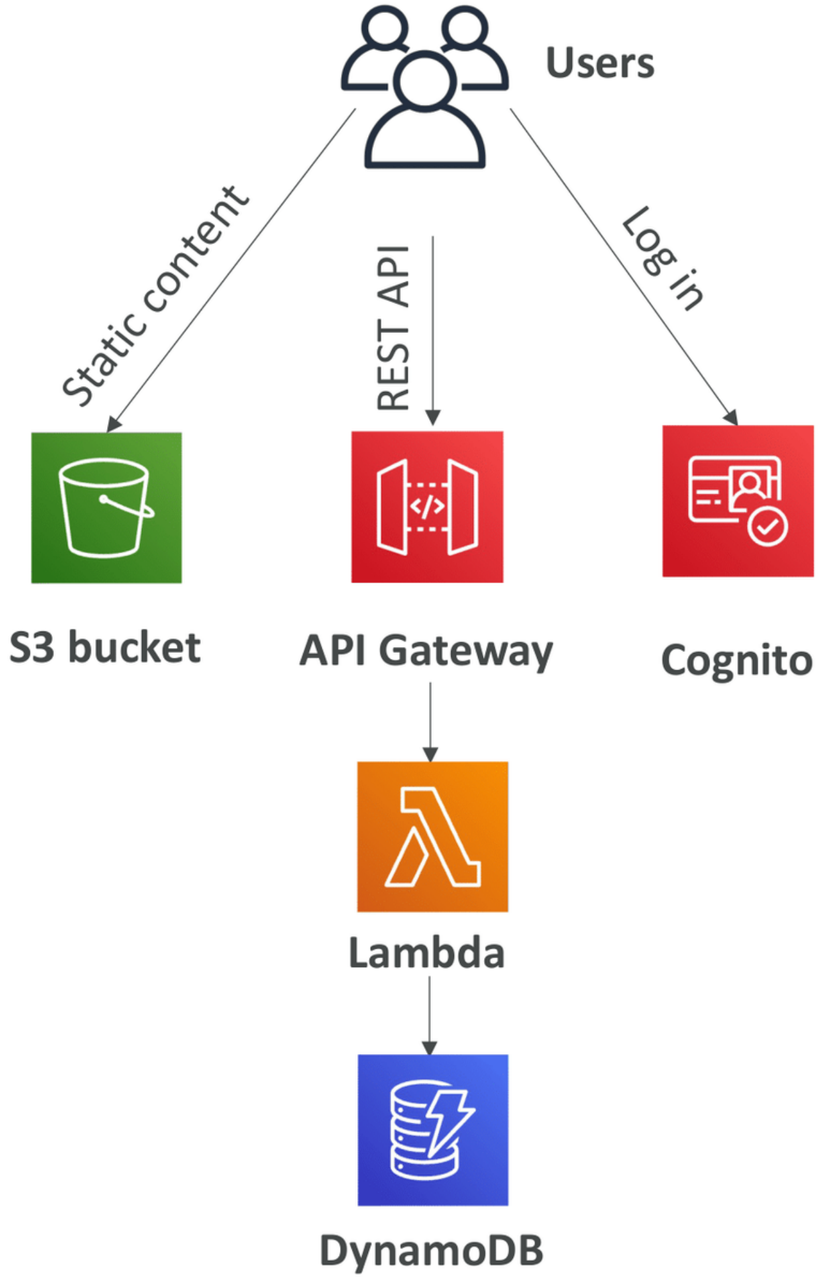
What's serverless?

- Serverless is a new paradigm in which the developers don't have to manage servers anymore…
- They just deploy code
- They just deploy… functions !
- Initially... Serverless == FaaS (Function as a Service)
- Serverless was pioneered by AWS Lambda but now also includes anything that's managed: "databases, messaging, storage, etc."
- **Serverless does not mean there are no servers…**
  it means you just don't manage / provision / see them

VINSYS®

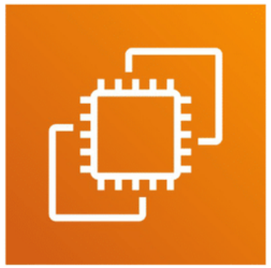
Serverless in AWS

- AWS Lambda
- DynamoDB
- AWS Cognito
- AWS API Gateway
- Amazon S3
- AWS SNS & SQS
- AWS Kinesis Data Firehose
- Aurora Serverless
- Step Functions
- Fargate



VINSYS

Why AWS Lambda



Amazon EC2

- Virtual Servers in the Cloud
- Limited by RAM and CPU
- Continuously running
- Scaling means intervention to add / remove servers

Amazon Lambda

- Virtual **functions** – no servers to manage!
- Limited by time - **short executions**
- Run **on-demand**
- **Scaling is automated**!

VINSYS®

Benefits of AWS Lambda

- Easy Pricing:
  - Pay per request and compute time
  - Free tier of 1,000,000 AWS Lambda requests and 400,000 GBs of compute time

- Integrated with the whole AWS suite of services
- Integrated with many programming languages
- Easy monitoring through AWS CloudWatch
- Easy to get more resources per functions (up to 10GB of RAM!)
- Increasing RAM will also improve CPU and network!

AWS Lambda Integrations
Main ones

API Gateway

Kinesis

DynamoDB

S3

CloudFront

CloudWatch Events
EventBridge

CloudWatch Logs

SNS

SQS

Cognito

VINSYS®

AWS Lambda language support

- Node.js (JavaScript)
- Python
- Java (Java 8 compatible)
- C# (.NET Core)
- Golang
- C# / Powershell
- Ruby
- Custom Runtime API (community supported, example Rust)

- Lambda Container Image
  - The container image must implement the Lambda Runtime API
  - ECS / Fargate is preferred for running arbitrary Docker images

VINSYS®

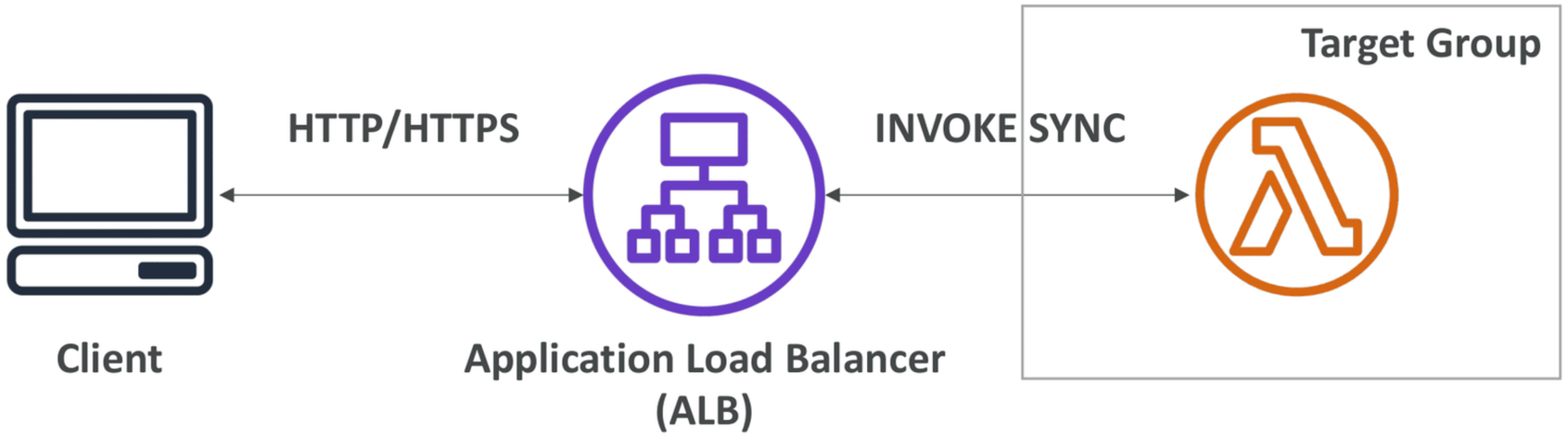
Some Use Cases of Lambda Functions in Banking

- Mobile Banking or Online Banking App Backend
- Real-time Transaction Processing
- Fraud Detection
- Customer Notifications
- Data Transformation
- ATM and POS Transaction Processing
- Loan Approval and Underwriting
- Customer Support Chatbots
- Risk Assessment

VINSYS®

Lambda Integration with ALB

- To expose a Lambda function as an HTTP(S) endpoint…
- You can use the **Application Load Balancer** (or an API Gateway)
- The Lambda function must be registered in a **target group**



VINSYS

Example: Serverless Thumbnail creation

trigger

push

New thumbnail in S3

push

Metadata in DynamoDB

Image name
Image size
Creation date
etc…

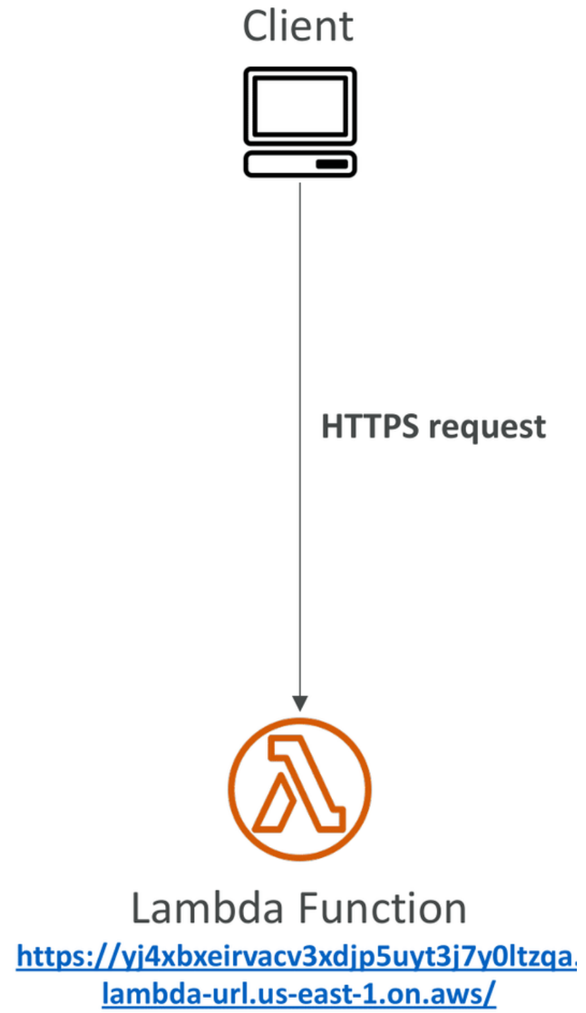
New image in S3

AWS Lambda Function Creates a Thumbnail

VINSYS

Lambda – Function URL

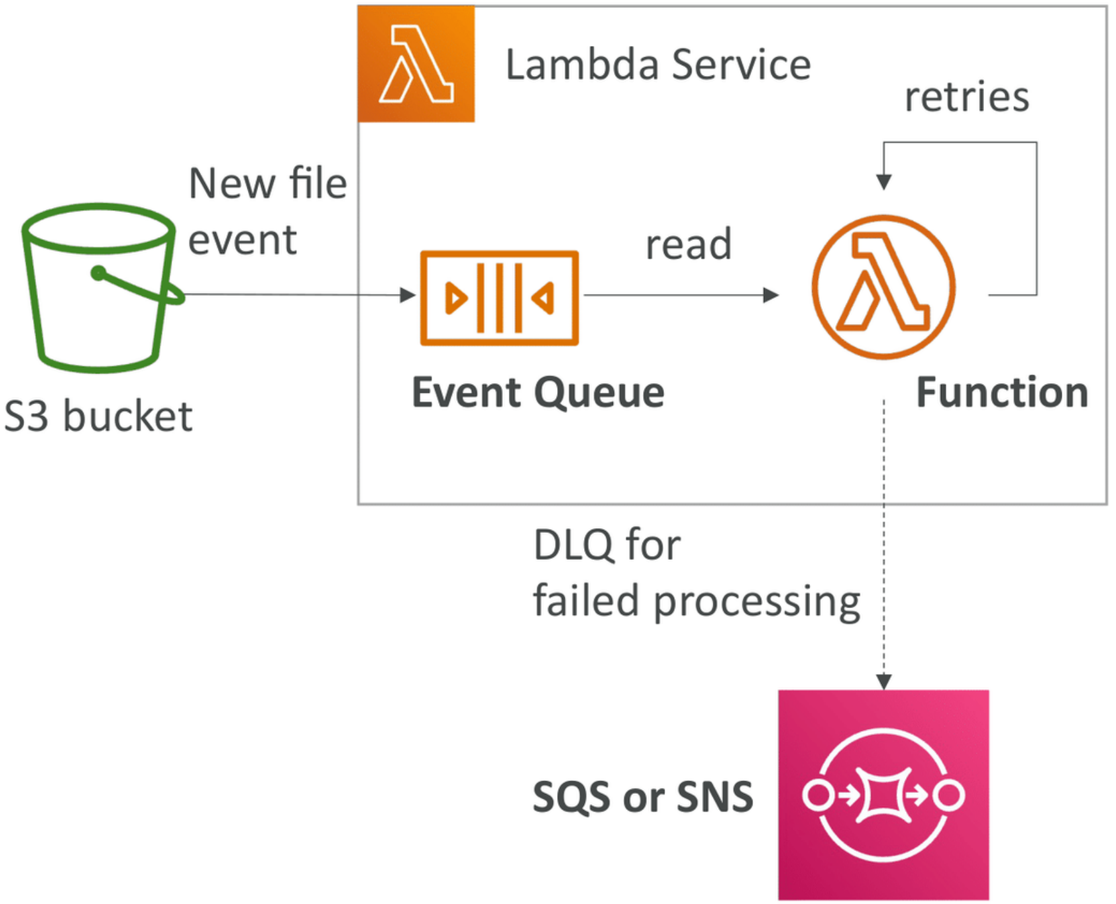
- Dedicated HTTP(S) endpoint for your Lambda function
- A unique URL endpoint is generated for you (never changes)
  - https://<*url-id*>.lambda-url.<*region*>.on.aws (dual-stack IPv4 & IPv6)
- Invoke via a web browser, curl, Postman, or any HTTP client
- Access your function URL through the public Internet only
  - Doesn't support PrivateLink (Lambda functions do support)
- Supports Resource-based Policies & CORS configurations
- Can be applied to any function alias or to $LATEST (can't be applied to other function versions)
- Create and configure using AWS Console or AWS API
- Throttle your function by using Reserved Concurrency



VINSYS

Lambda – Asynchronous Invocations

- S3, SNS, CloudWatch Events…
- The events are placed in an **Event Queue**
- Lambda attempts to retry on errors
  - 3 tries total
  - 1 minute wait after 1st , then 2 minutes wait
- Make sure the processing is **idempotent** (in case of retries)
- If the function is retried, you will see **duplicate logs entries in CloudWatch Logs**
- Can define a DLQ (dead-letter queue) – **SNS or SQS** – for failed processing (need correct IAM permissions)
- Asynchronous invocations allow you to speed up the processing if you don't need to wait for the result (ex: you need 1000 files processed)

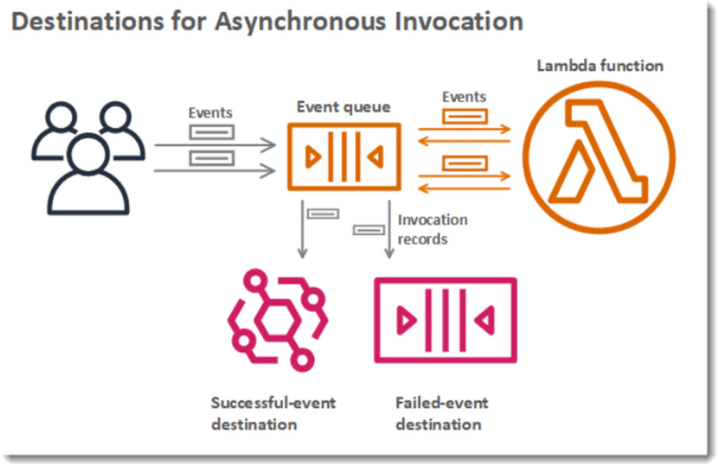


VINSYS

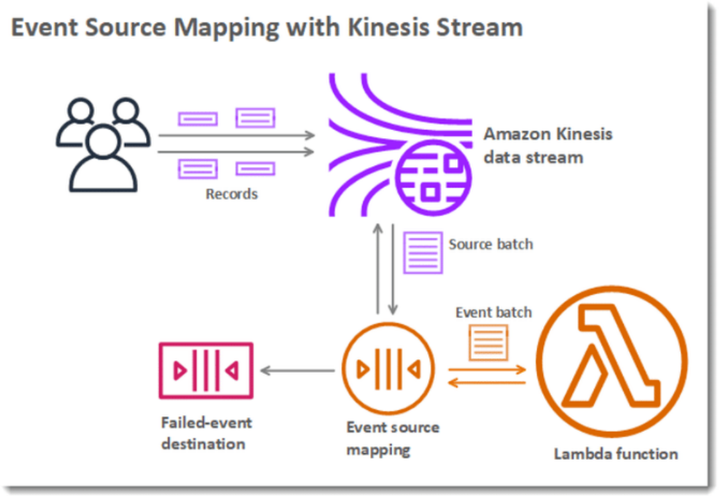
Lambda – Destinations

- Nov 2019: Can configure to send result to a **destination**
- **Asynchronous invocations** - can define destinations for successful and failed event:
  - Amazon SQS
  - Amazon SNS
  - AWS Lambda
  - Amazon EventBridge bus
- *Note: AWS recommends you use destinations instead of DLQ now (but both can be used at the same time)*

- **Event Source mapping:** for discarded event batches
  - Amazon SQS
  - Amazon SNS
- *Note: you can send events to a DLQ directly from SQS*



https://docs.aws.amazon.com/lambda/latest/dg/invocation-async.html



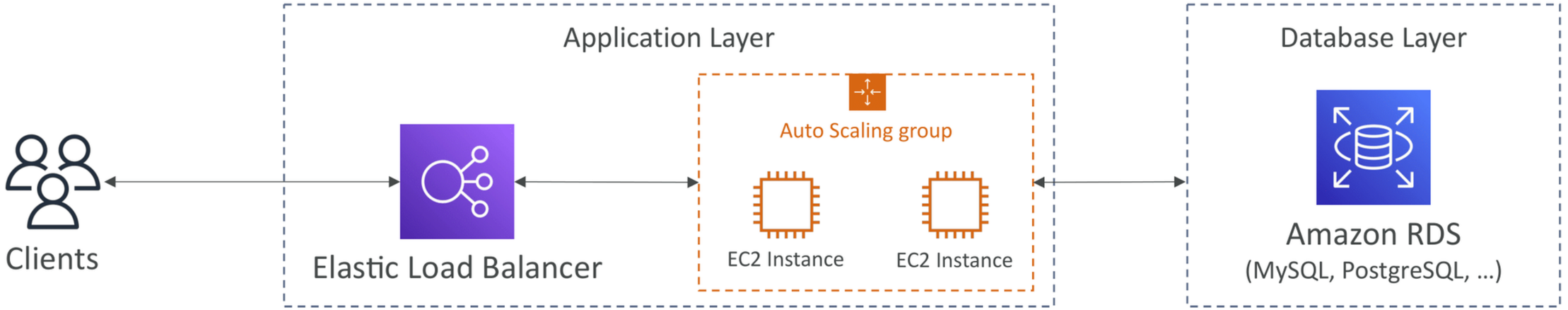
https://docs.aws.amazon.com/lambda/latest/dg/invocation-eventsourcemapping.html

VINSYS

Amazon DynamoDB

NoSQL Serverless Database

Traditional Architecture



- Traditional applications leverage RDBMS databases
- These databases have the SQL query language
- Strong requirements about how the data should be modeled
- Ability to do query joins, aggregations, complex computations
- Vertical scaling (getting a more powerful CPU / RAM / IO)
- Horizontal scaling (increasing reading capability by adding EC2 / RDS Read Replicas)

VINSYS

NoSQL databases

- NoSQL databases are non-relational databases and are **distributed**
- NoSQL databases include MongoDB, DynamoDB, …
- NoSQL databases do not support query joins (or just limited support)
- All the data that is needed for a query is present in one row
- NoSQL databases don't perform aggregations such as "SUM", "AVG", …
- **NoSQL databases scale horizontally**

- There's no "right or wrong" for NoSQL vs SQL, they just require to model the data differently and think about user queries differently

VINSYS®

Amazon DynamoDB

- Fully managed, highly available with replication across multiple AZs
- NoSQL database - not a relational database
- Scales to massive workloads, distributed database
- Millions of requests per seconds, trillions of row, 100s of TB of storage
- Fast and consistent in performance (low latency on retrieval)
- Integrated with IAM for security, authorization and administration
- Enables event driven programming with DynamoDB Streams
- Low cost and auto-scaling capabilities
- Standard & Infrequent Access (IA) Table Class

VINSYS®

Some Use Cases of Dynamo DB in Banking

- Product and Service Catalogs
- Customer Support and Case Management
- User Preferences and Personalization
- ATM and Branch Locator Services
- KYC Compliance

VINSYS®

DynamoDB – PartiQL

- SQL-compatible query language for DynamoDB
- Allows you to select, insert, update, and delete data in DynamoDB using SQL
- Run queries across multiple DynamoDB tables
- Run PartiQL queries from:
  - AWS Management Console
  - NoSQL Workbench for DynamoDB
  - DynamoDB APIs
  - AWS CLI
  - AWS SDK

```
SELECT OrderID, Total
FROM Orders
WHERE OrderID IN [1, 2, 3]
ORDER BY OrderID DESC
```

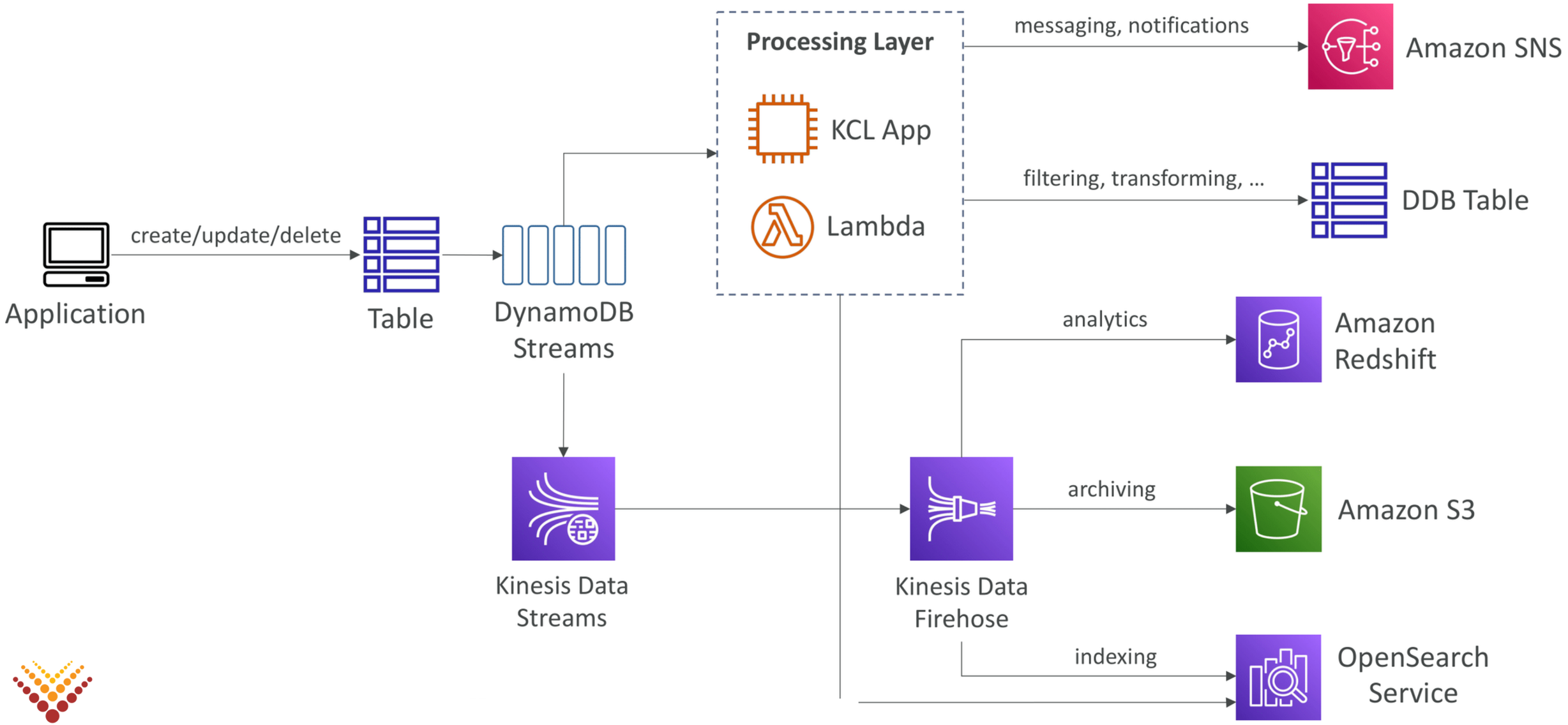
VINSYS

DynamoDB Streams

- Ordered stream of item-level modifications (create/update/delete) in a table
- Stream records can be:
  - Sent to **Kinesis Data Streams**
  - Read by **AWS Lambda**
  - Read by **Kinesis Client Library applications**
- Data Retention for up to 24 hours
- Use cases:
  - react to changes in real-time (welcome email to users)
  - Analytics
  - Insert into derivative tables
  - Insert into OpenSearch Service
  - Implement cross-region replication

VINSYS®

DynamoDB Streams

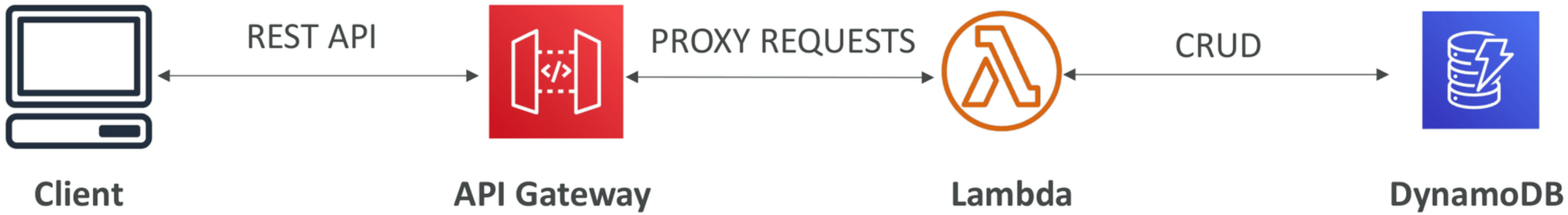


VINSYS

Amazon API Gateway

Build, Deploy and Manage APIs

Example: Building a Serverless API



VINSYS

AWS API Gateway

- AWS Lambda + API Gateway: No infrastructure to manage
- Support for the WebSocket Protocol
- Handle API versioning (v1, v2…)
- Handle different environments (dev, test, prod…)
- Handle security (Authentication and Authorization)
- Create API keys, handle request throttling
- Swagger / Open API import to quickly define APIs
- Transform and validate requests and responses
- Generate SDK and API specifications
- Cache API responses

VINSYS®

API Gateway – Integrations High Level

- Lambda Function
  - Invoke Lambda function
  - Easy way to expose REST API backed by AWS Lambda
- HTTP
  - Expose HTTP endpoints in the backend
  - Example: internal HTTP API on premise, Application Load Balancer…
  - Why? Add rate limiting, caching, user authentications, API keys, etc…
- AWS Service
  - Expose any AWS API through the API Gateway
  - Example: start an AWS Step Function workflow, post a message to SQS
  - Why? Add authentication, deploy publicly, rate control…

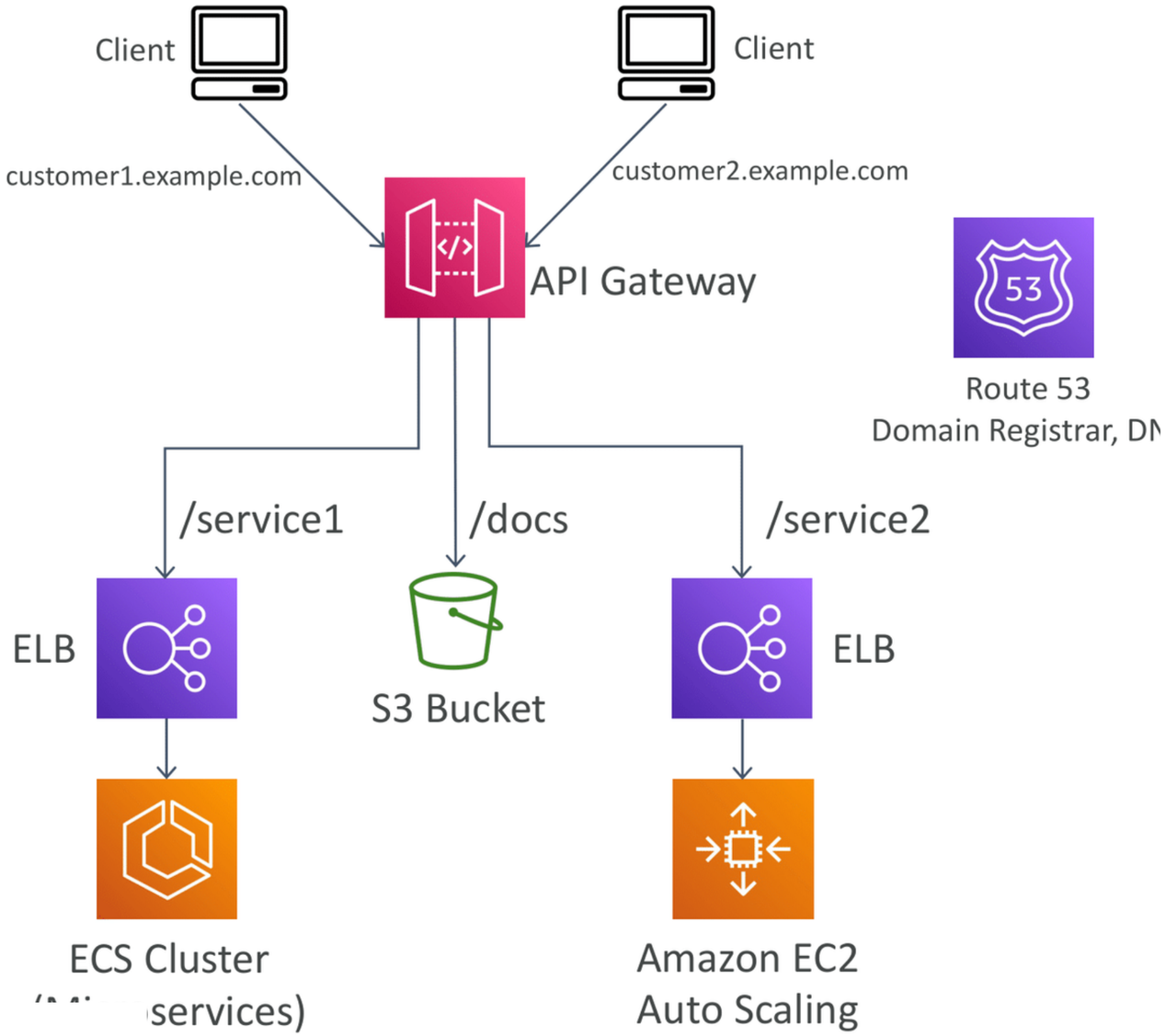
Some Use Cases of API Gateways in Banking

- Mobile Banking Apps
- Open Banking – Third Party
- Customer Authentication
- Payment Processing
- Investment Portfolio Management
- Market Data Access

VINSYS®

API Gateway - Architecture

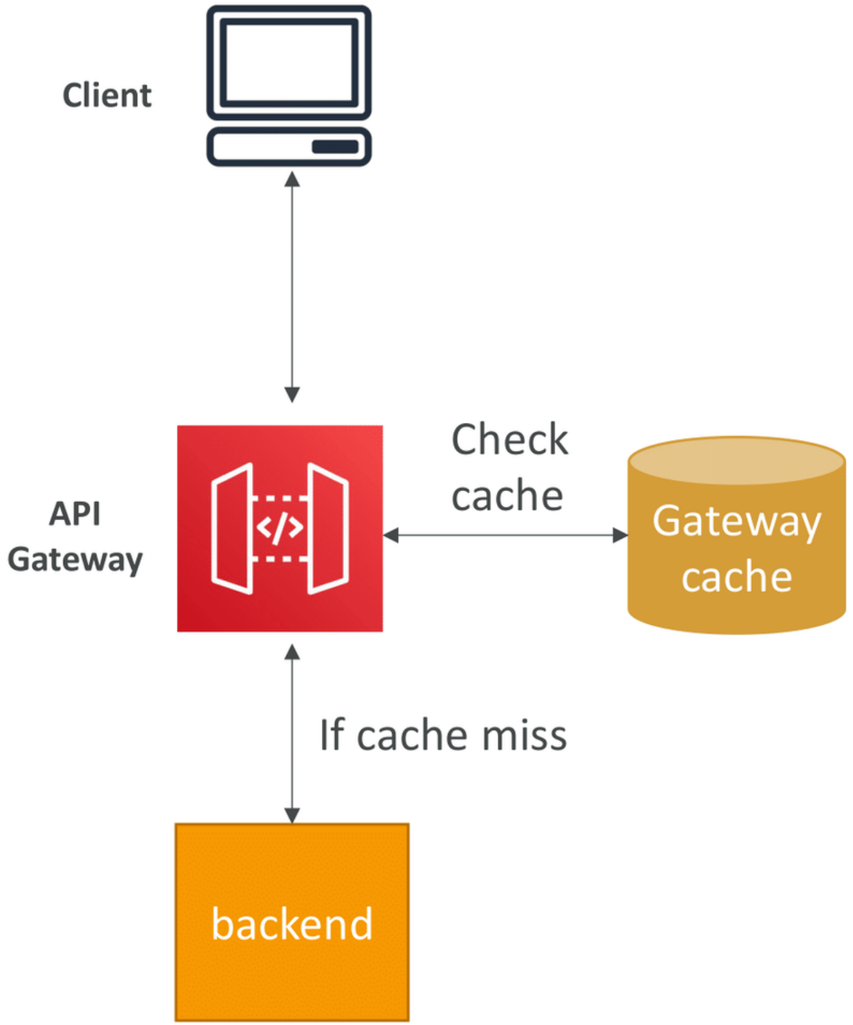
- Create a single interface for all the microservices in your company
- Use API endpoints with various resources
- Apply a simple domain name and SSL certificates
- Can apply forwarding and transformation rules at the API Gateway level



VINSYS

Caching API responses

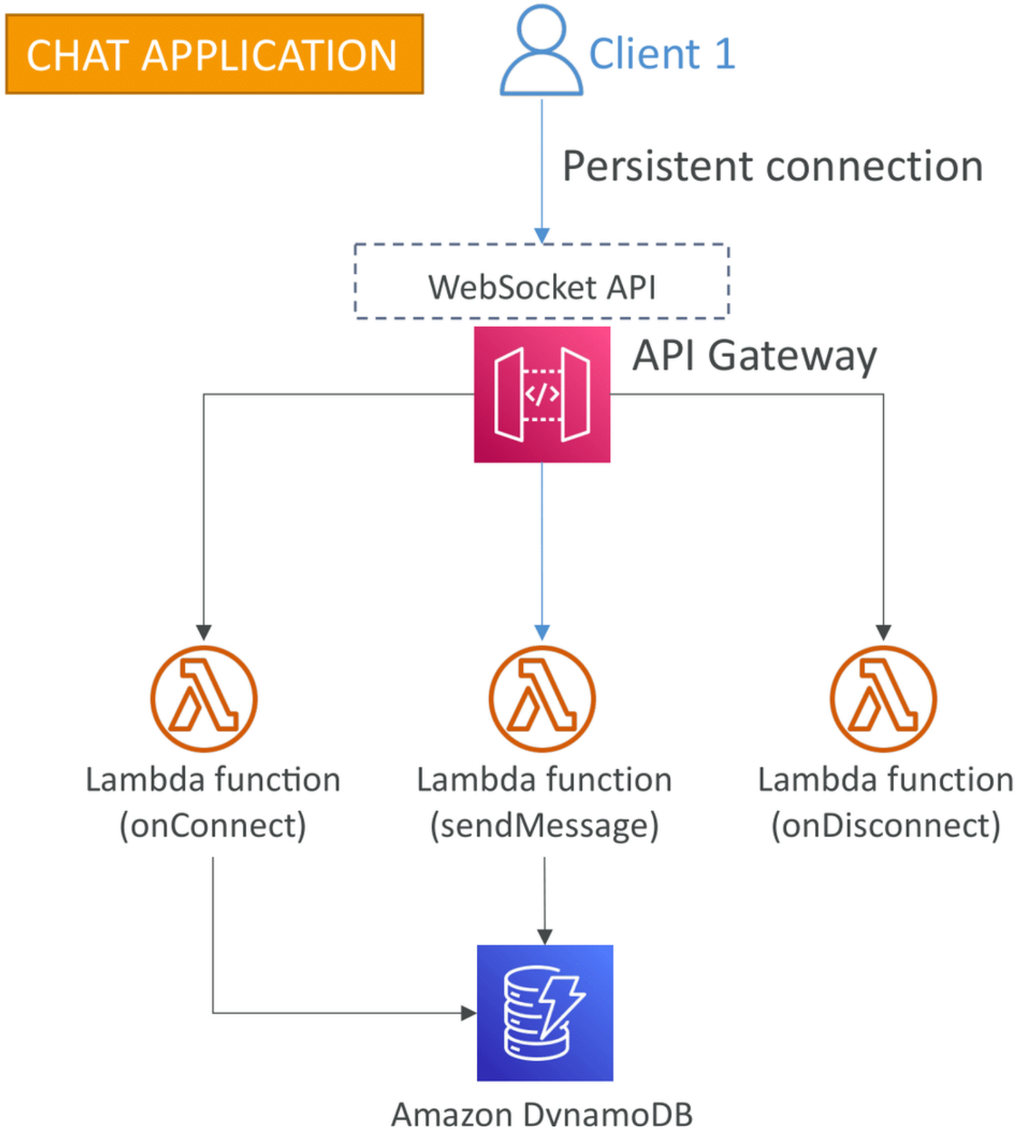
- Caching reduces the number of calls made to the backend
- Default TTL (time to live) is 300 seconds (min: 0s, max: 3600s)
- **Caches are defined per stage**
- Possible to override cache settings **per method**
- Cache encryption option
- Cache capacity between 0.5GB to 237GB
- Cache is expensive, makes sense in production, may not make sense in dev / test



VINSYS

API Gateway – WebSocket API – Overview

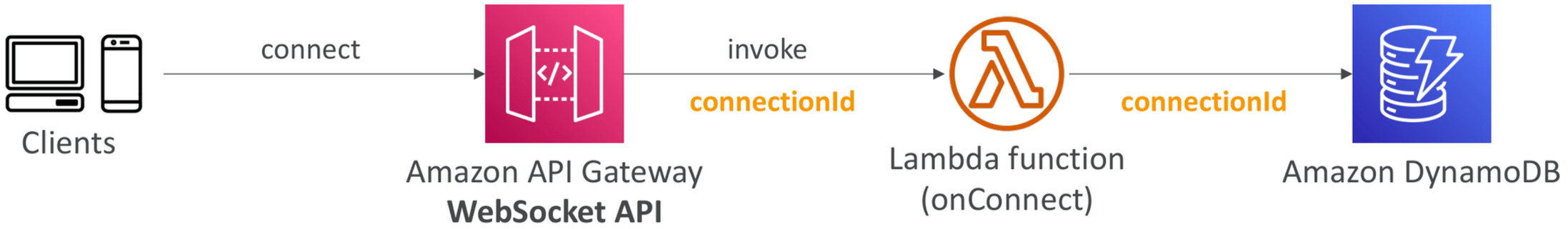
- What's WebSocket?
  - Two-way interactive communication between a user's browser and a server
  - Server can push information to the client
  - This enables **stateful** application use cases
- WebSocket APIs are often used in **real-time applications** such as chat applications, collaboration platforms, multiplayer games, and financial trading platforms.
- Works with AWS Services (Lambda, DynamoDB) or HTTP endpoints



VINSYS

Connecting to the API

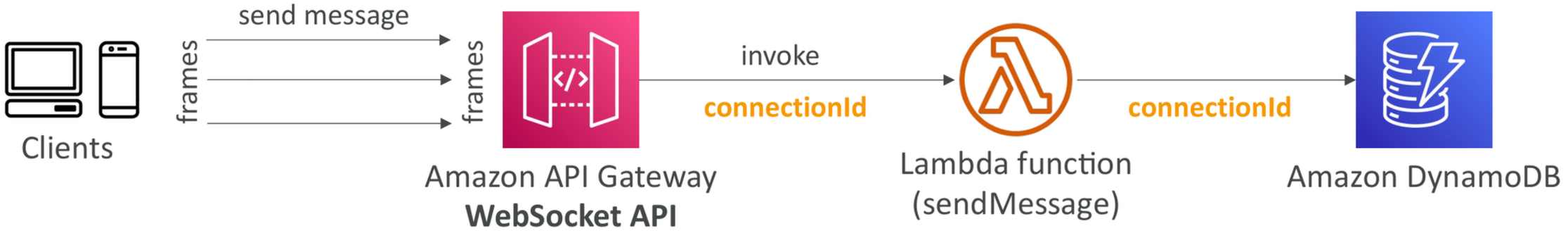
WebSocket URL
wss://[some-uniqueid].execute-api.[region].amazonaws.com/[stage-name]



VINSYS

Client to Server Messaging
ConnectionID is re-used

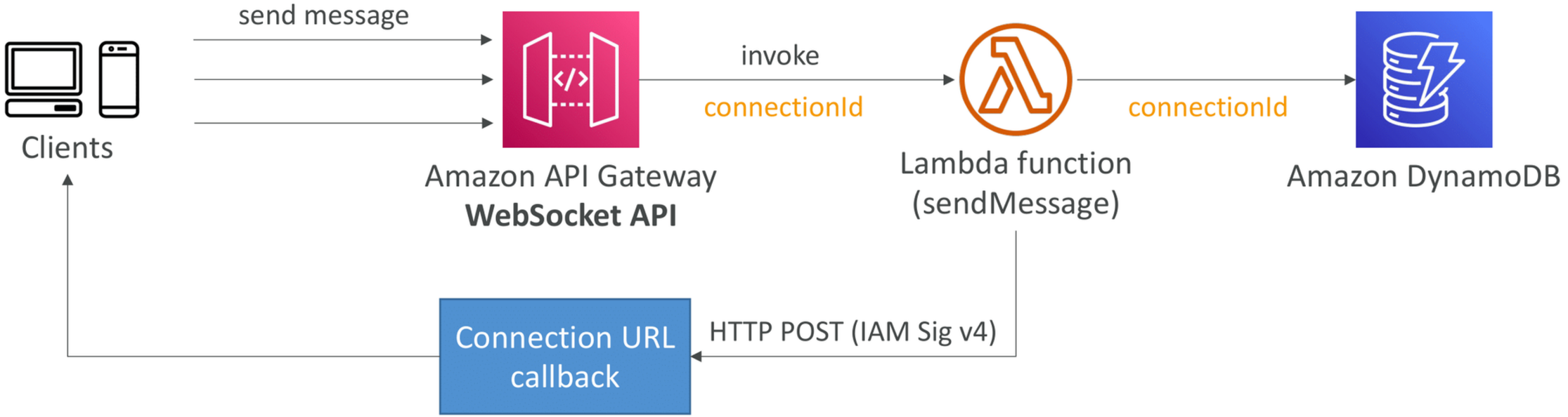
WebSocket URL
wss://abcdef.execute-api.us-west-1.amazonaws.com/dev



VINSYS

Server to Client Messaging

WebSocket URL
wss://abcdef.execute-api.us-west-1.amazonaws.com/dev



Connection URL
wss://abcdef.execute-api.us-west-1.amazonaws.com/dev/@connections/connectionId

VINSYS

AWS CICD

CodeCommit, CodePipeline, CodeBuild, CodeDeploy, …

CICD – Introduction

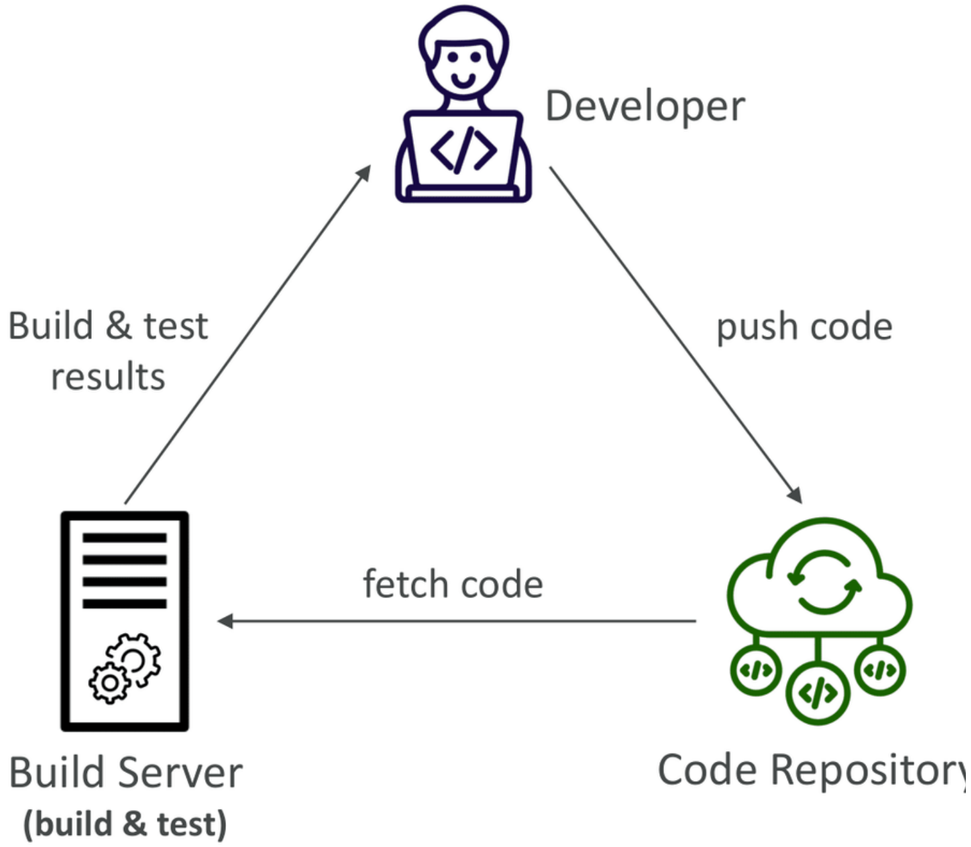
- This section is all about automating the deployment we've done so far while adding increased safety

- We'll learn about:
  - **AWS CodeCommit** – storing our code
  - **AWS CodePipeline** – automating our pipeline from code to Elastic Beanstalk
  - **AWS CodeBuild** – building and testing our code
  - **AWS CodeDeploy** – deploying the code to EC2 instances (not Elastic Beanstalk)
  - **AWS CodeStar** – manage software development activities in one place
  - **AWS CodeArtifact** – store, publish, and share software packages
  - **AWS CodeGuru** – automated code reviews using Machine Learning

Continuous Integration (CI)

- Developers push the code to a code repository often (e.g., GitHub, CodeCommit, Bitbucket…)
- A testing / build server checks the code as soon as it's pushed (CodeBuild, Jenkins CI, …)
- The developer gets feedback about the tests and checks that have passed / failed

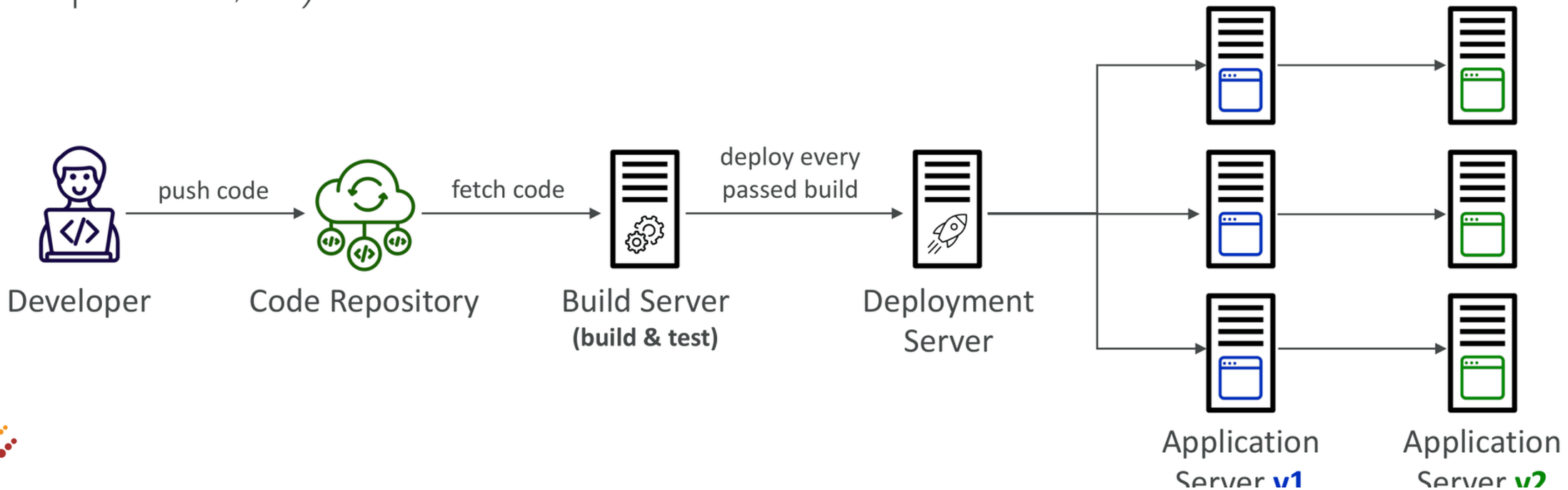
- Find bugs early, then fix bugs
- Deliver faster as the code is tested
- Deploy often
- Happier developers, as they're unblocked



VINSYS

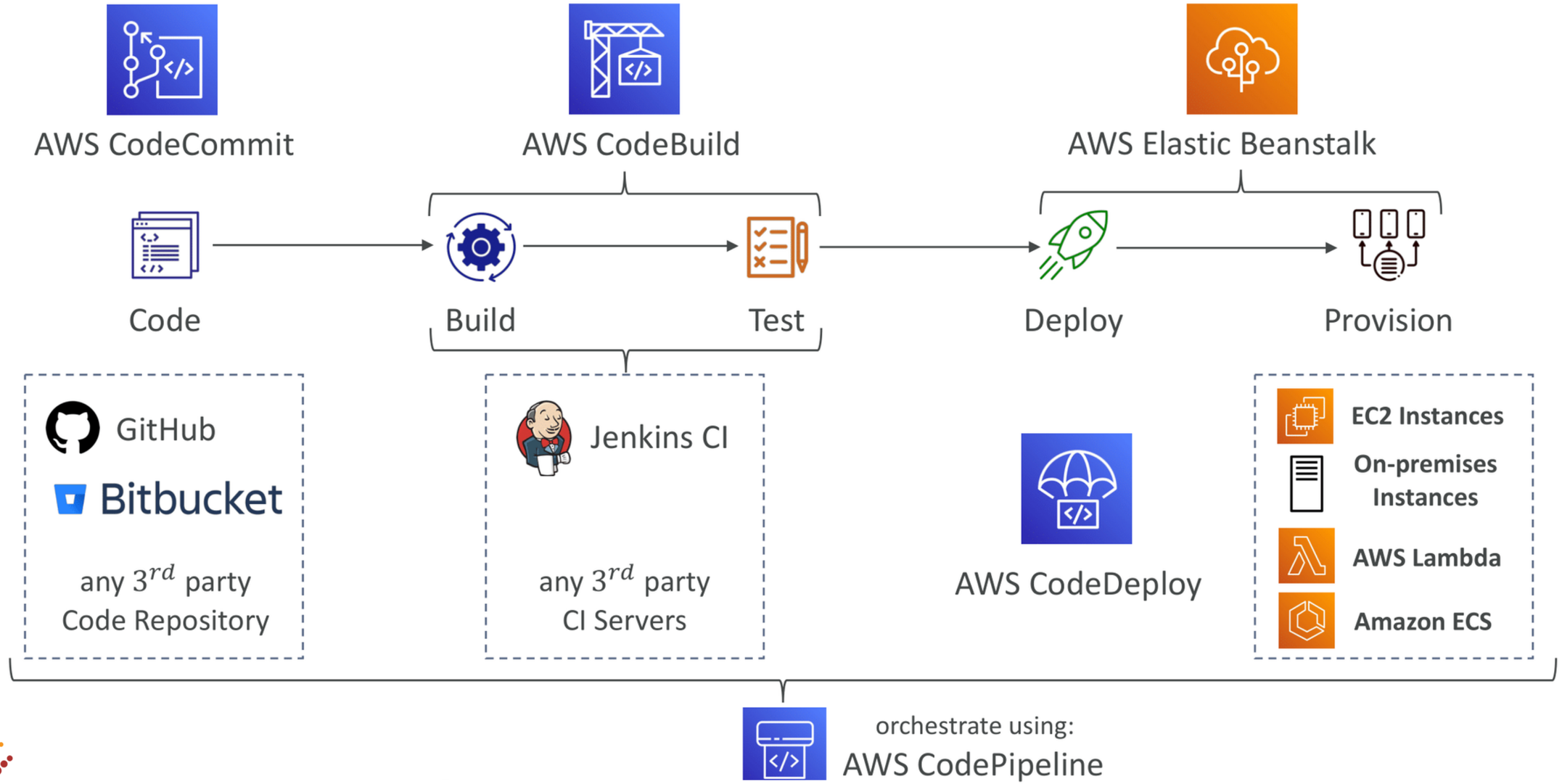
Continuous Delivery (CD)

- Ensures that the software can be released reliably whenever needed
- Ensures deployments happen often and are quick
- Shift away from "one release every 3 months" to "5 releases a day"
- That usually means automated deployment (e.g., CodeDeploy, Jenkins CD, Spinnaker, …)



VINSYS

Technology Stack for CICD



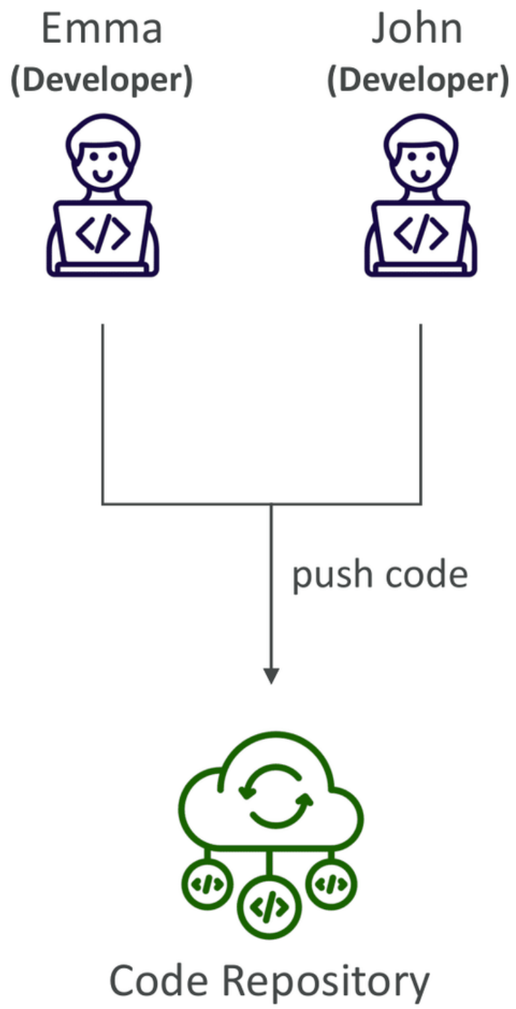
VINSYS

AWS CodeCommit

- **Version control** is the ability to understand the various changes that happened to the code over time (and possibly roll back)
- All these are enabled by using a version control system such as **Git**
- A Git repository can be synchronized on your computer, but it usually is uploaded on a central online repository
- Benefits are:
  - Collaborate with other developers
  - Make sure the code is backed-up somewhere
  - Make sure it's fully viewable and auditable

AWS CodeCommit

- Git repositories can be expensive
- The industry includes GitHub, GitLab, Bitbucket, …
- And **AWS CodeCommit:**
  - Private Git repositories
  - No size limit on repositories (scale seamlessly)
  - Fully managed, highly available
  - Code only in AWS Cloud account => increased security and compliance
  - Security (encrypted, access control, …)
  - Integrated with Jenkins, AWS CodeBuild, and other CI tools



VINSYS

CodeCommit vs. GitHub

	CodeCommit	GitHub
Support Code Review (Pull Requests)	✔	✔
Integration with AWS CodeBuild	✔	✔
Authentication (SSH & HTTPS)	✔	✔
Security	IAM Users & Roles	GitHub Users
Hosting	Managed & hosted by AWS	- Hosted by GitHub - GitHub Enterprise: self hosted on your servers
UI	Minimal	Fully Featured

VINSYS®

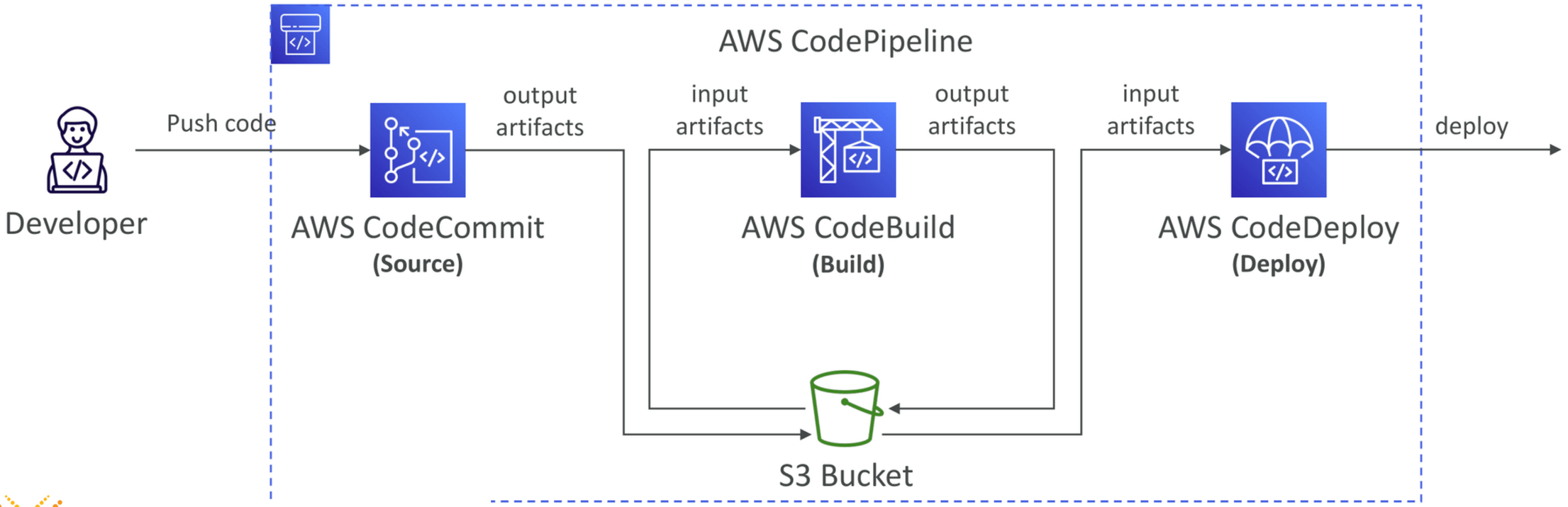
AWS CodePipeline

- Visual Workflow to orchestrate your CICD
- **Source** – CodeCommit, ECR, S3, Bitbucket, GitHub
- **Build** – CodeBuild, Jenkins, CloudBees, TeamCity
- **Test** – CodeBuild, AWS Device Farm, 3rd party tools, …
- **Deploy** – CodeDeploy, Elastic Beanstalk, CloudFormation, ECS, S3, …
- **Invoke** – Lambda, Step Functions
- Consists of stages:
  - Each stage can have sequential actions and/or parallel actions
  - Example: Build ➔ Test ➔ Deploy ➔ Load Testing ➔ …
  - Manual approval can be defined at any stage

VINSYS

CodePipeline – Artifacts

- Each pipeline stage can create **artifacts**
- Artifacts stored in an S3 bucket and passed on to the next stage



VINSYS

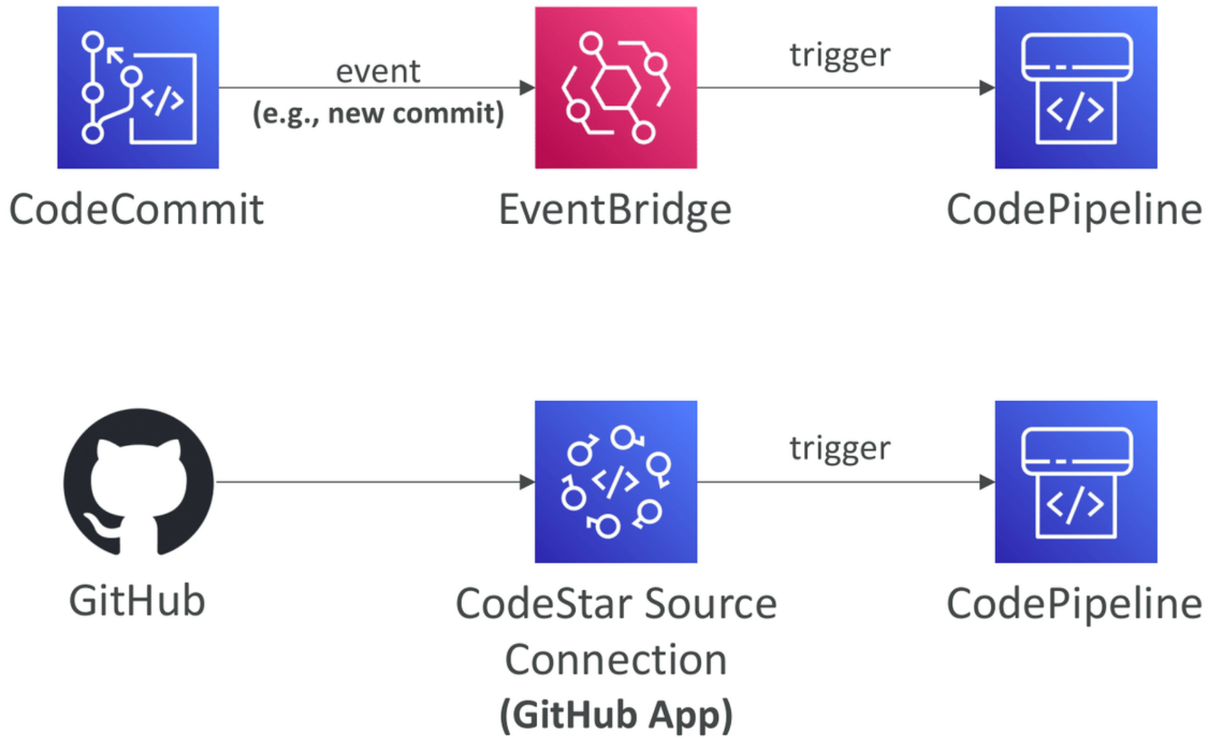
CodePipeline – Troubleshooting

- For CodePipeline Pipeline/Action/Stage Execution State Changes
- Use **CloudWatch Events (Amazon EventBridge).** Example:
  - You can create events for failed pipelines
  - You can create events for cancelled stages

- If CodePipeline fails a stage, your pipeline stops, and you can get information in the console
- If pipeline can't perform an action, make sure the "IAM Service Role" attached does have enough IAM permissions (IAM Policy)
- AWS CloudTrail can be used to audit AWS API calls

CodePipeline – Events

Events



VINSYS

CodePipeline – Action Types Constraints for Artifacts

- Owner
  - **AWS** – for AWS services
  - **3rd Party** – GitHub or Alexa Skills Kit
  - **Custom** – Jenkins
- Action Type
  - **Source** – S3, ECR, GitHub, …
  - **Build** – CodeBuild, Jenkins
  - **Test** – CodeBuild, Device Farm, Jenkins
  - **Approval** – **Manual**
  - **Invoke** – Lambda, Step Functions
  - **Deploy** – S3, CloudFormation, CodeDeploy, Elastic Beanstalk, OpsWorks, ECS, Service Catalog, …

Owner	Action Type	Provider
AWS	Source	S3
AWS	Source	CodeCommit
AWS	Source	ECR
3rd Party	Source	GitHub
AWS	Build	Codebuild
AWS	Test	CodeBuild
AWS	Test	Device Farm
AWS	Approval	Manual
AWS	Deploy	S3
AWS	Deploy	CloudFormation
AWS	Deploy	CodeDeploy
AWS	Deploy	Elastic Beanstalk
AWS	Deploy	OpsWorks Stacks
AWS	Deploy	ECS
AWS	Deploy	Service Catalog
AWS	Invoke	Lambda
AWS	Invoke	Step Functions
3rd Party	Deploy	Alexa Skills Kit
Custom	Build	Jenkins
Custom	Test	Jenkins
Custom	Any Suggested Category	Specified in Custom Action

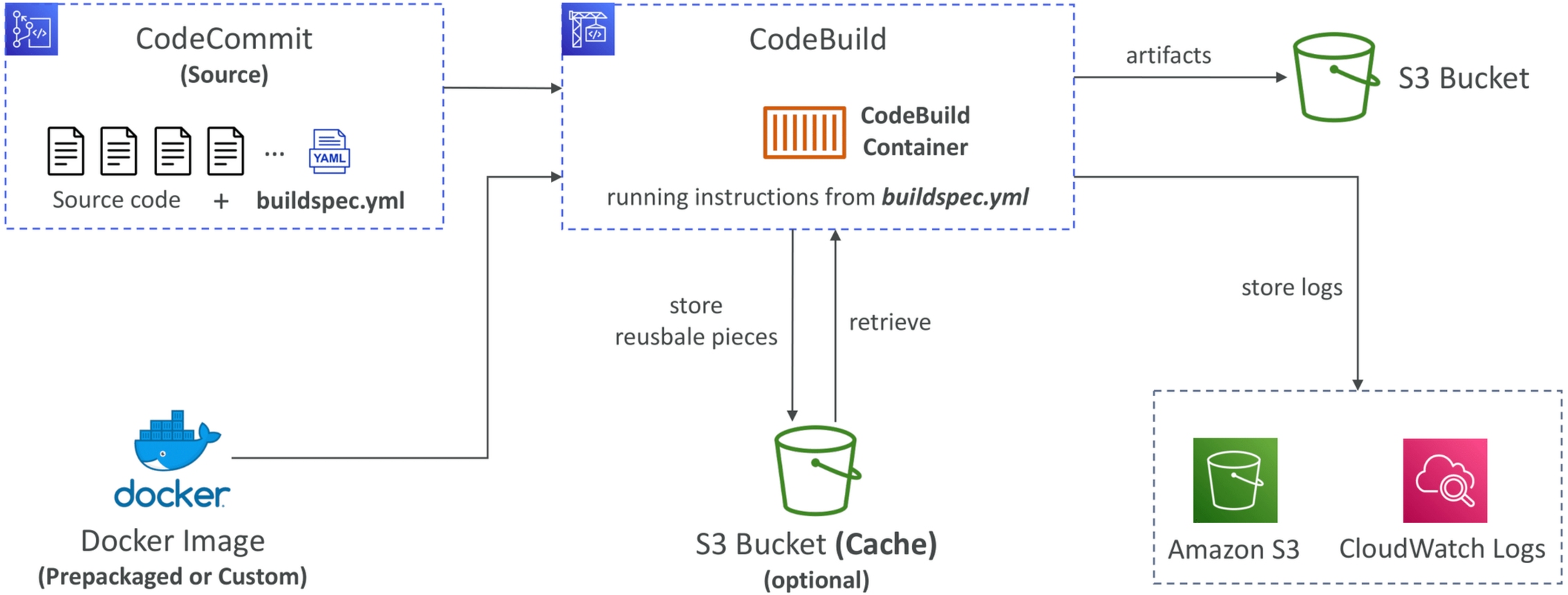
AWS CodeBuild

- **Source** – CodeCommit, S3, Bitbucket, GitHub
- **Build instructions**: Code file **buildspec.yml** or insert manually in Console
- **Output logs** can be stored in Amazon S3 & CloudWatch Logs
- Use CloudWatch Metrics to monitor build statistics
- Use EventBridge to detect failed builds and trigger notifications
- Use CloudWatch Alarms to notify if you need "thresholds" for failures

- Build Projects can be defined within CodePipeline or CodeBuild

VINSYS

CodeBuild – How it Works



VINSYS

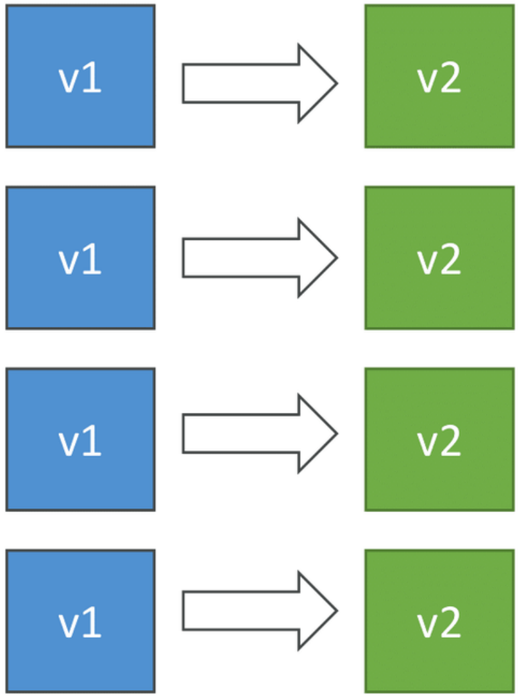
CodeBuild – Local Build

- In case of need of deep troubleshooting beyond logs…
- You can run CodeBuild locally on your desktop (after installing Docker)
- For this, leverage the CodeBuild Agent

- https://docs.aws.amazon.com/codebuild/latest/userguide/use-codebuild-agent.html

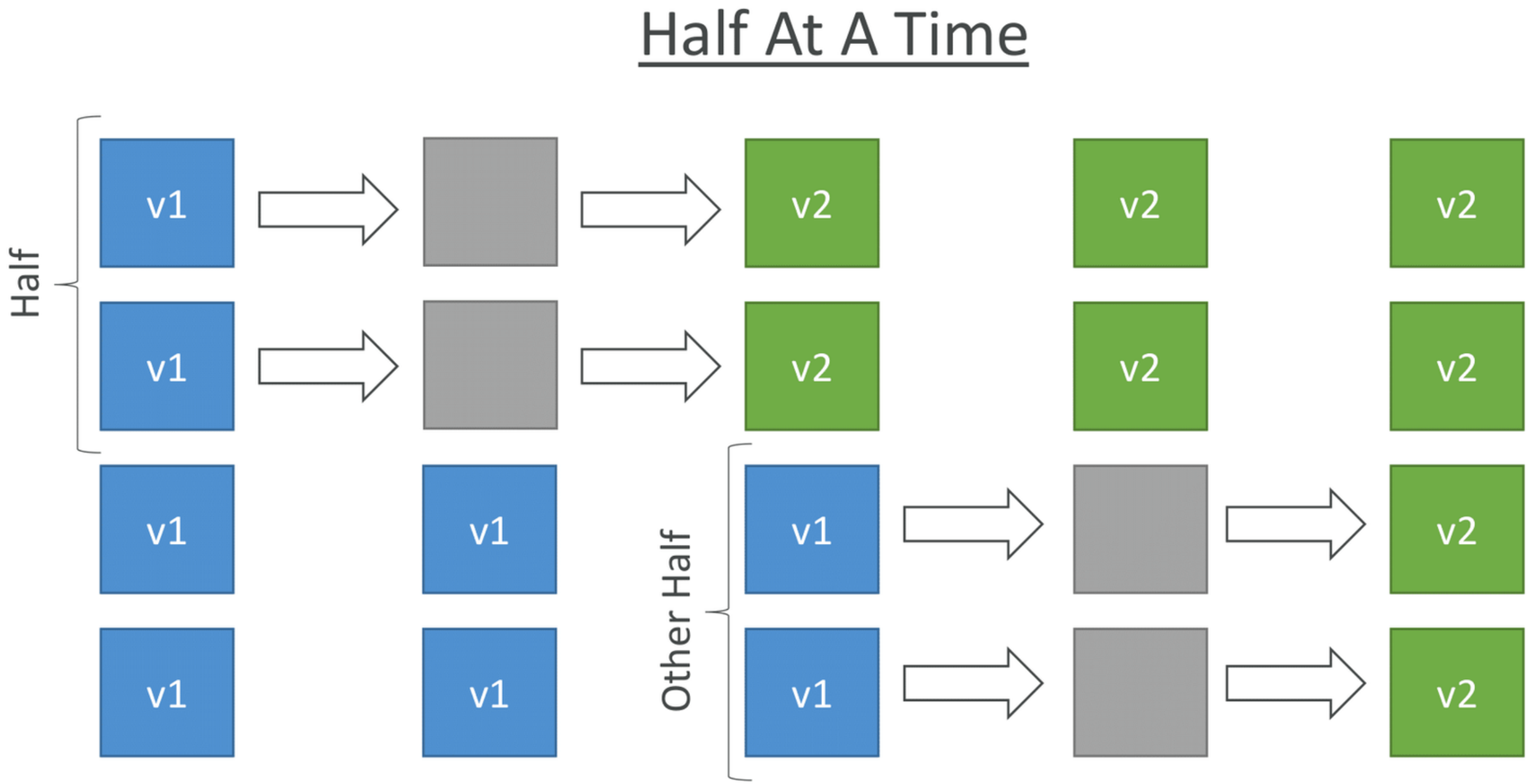
AWS CodeDeploy

- Deployment service that automates application deployment
- Deploy new applications versions to EC2 Instances, On-premises servers, Lambda functions, ECS Services
- Automated Rollback capability in case of failed deployments, or trigger CloudWatch Alarm
- Gradual deployment control
- A file named appspec.yml defines how the deployment happens



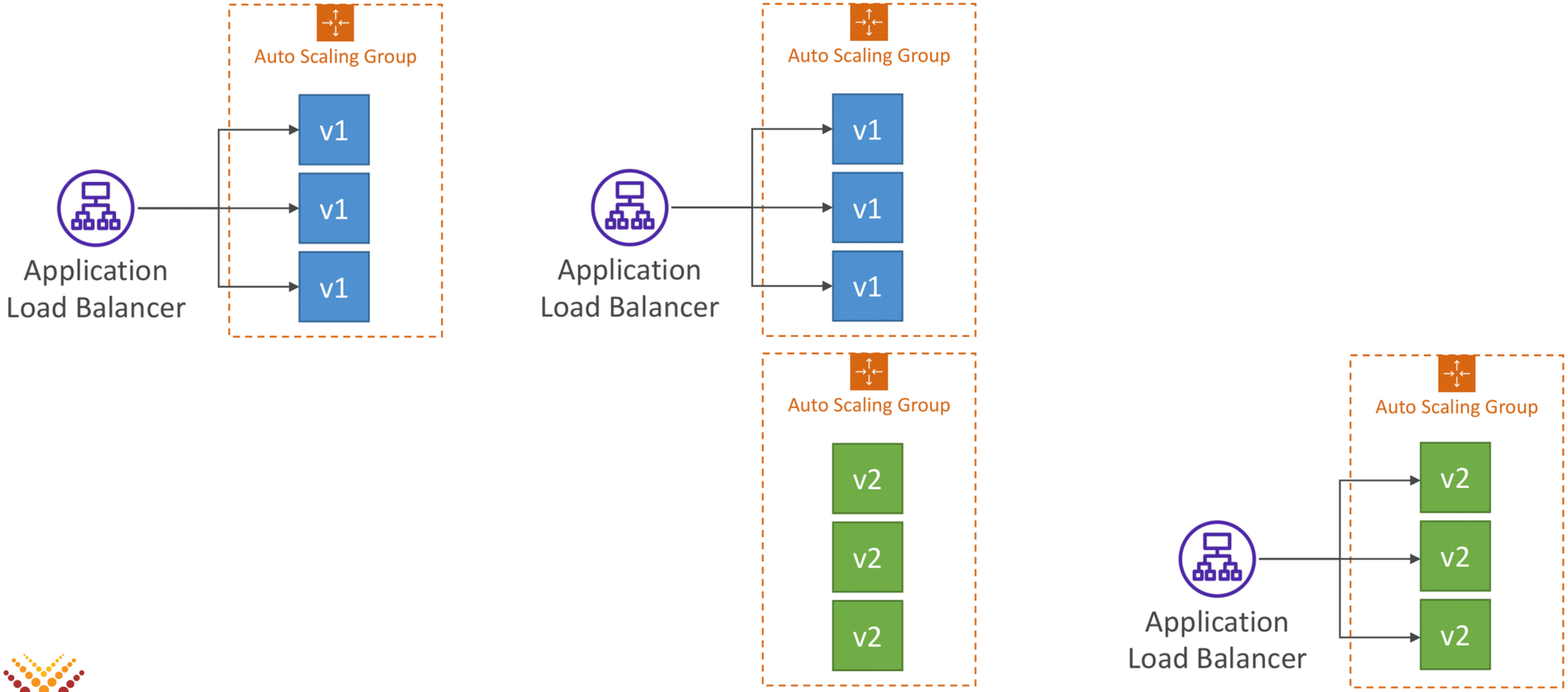
VINSYS

CodeDeploy – In-Place Deployment



VINSYS

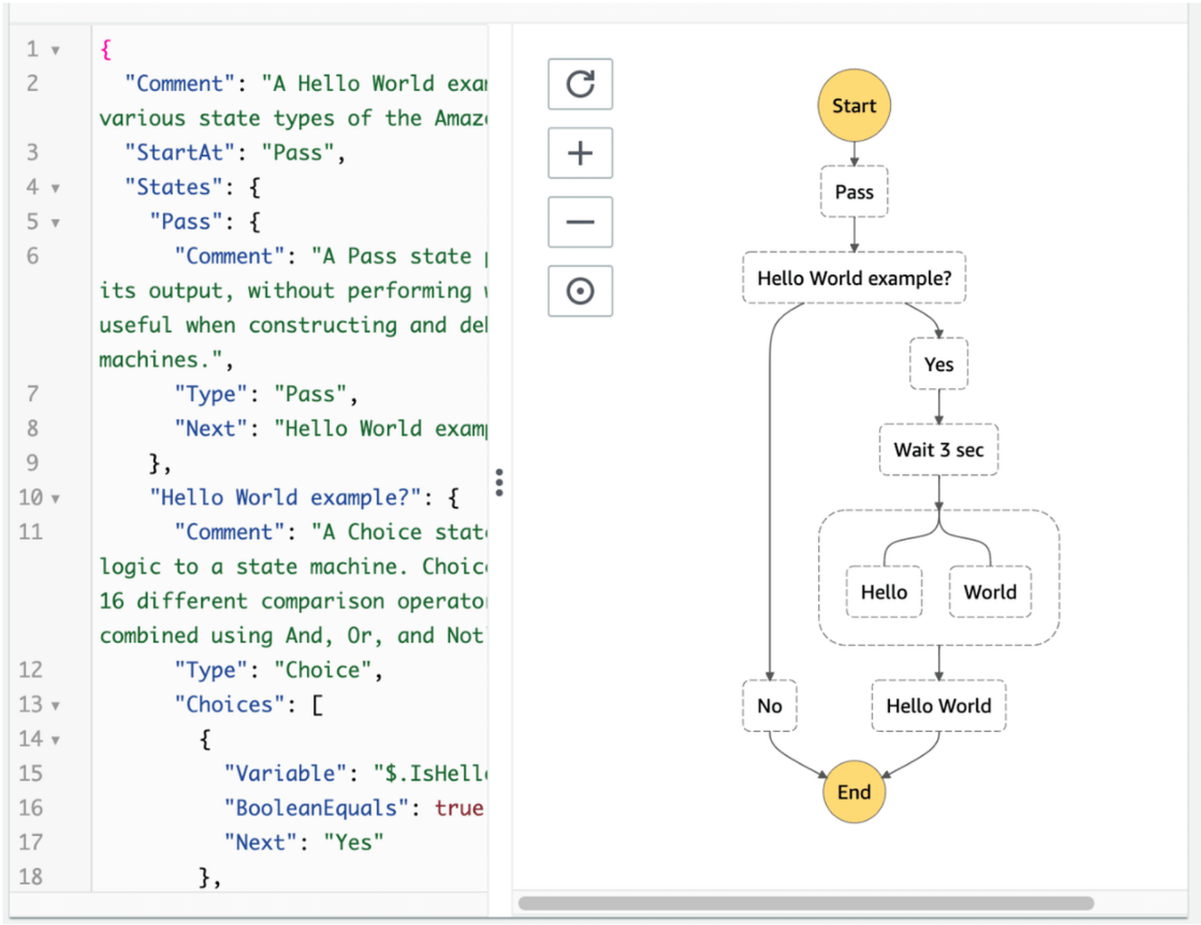
CodeDeploy – Blue-Green Deployment



VINSYS

AWS Step Functions

- Model your **workflows** as **state machines (one per workflow)**
  - Order fulfillment, Data processing
  - Web applications, Any workflow
- Written in JSON
- Visualization of the workflow and the execution of the workflow, as well as history
- Start workflow with SDK call, API Gateway, Event Bridge (CloudWatch Event)



Charges are based on requests per workflow

VINSYS

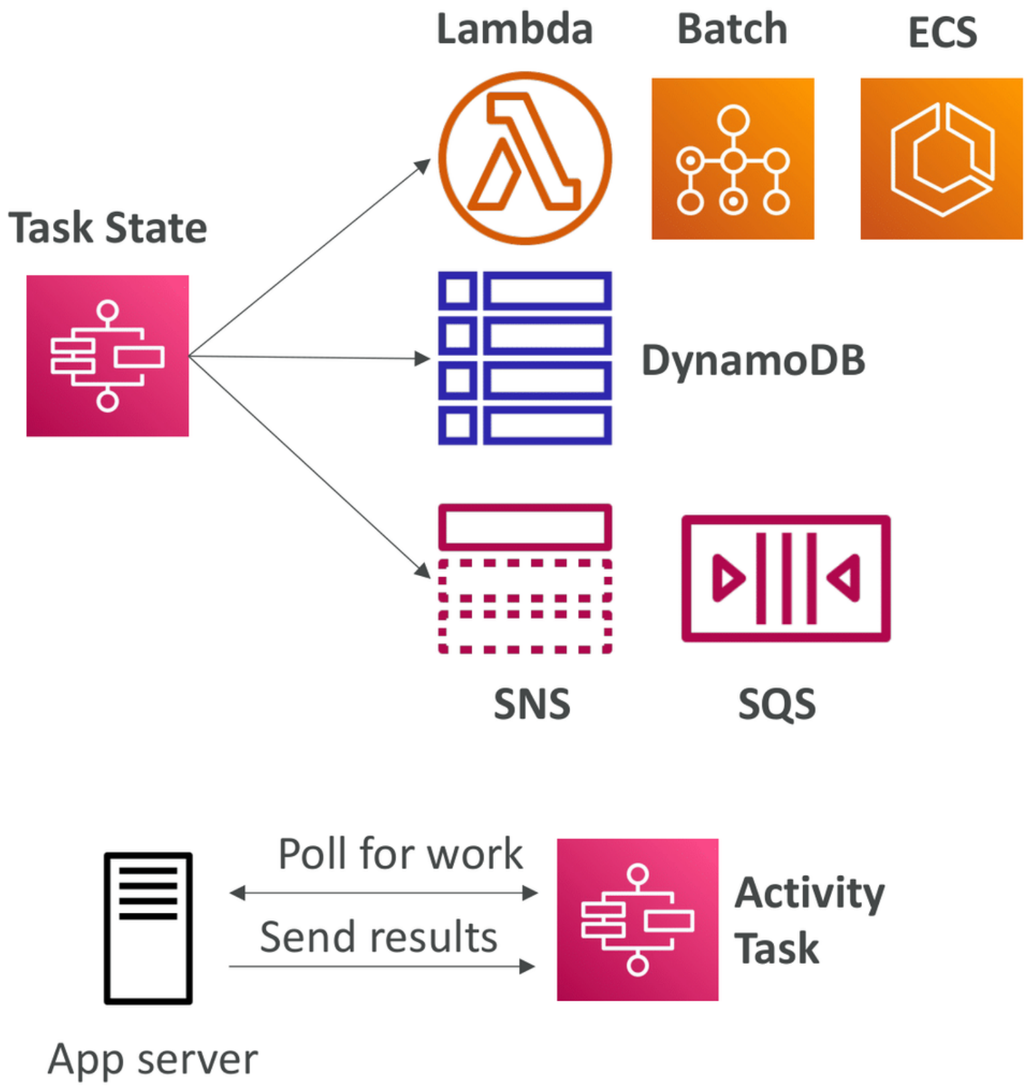
Some Use Cases of Step Functions in Banking

- Loan Application Processing
- Fraud Detection and Investigation
- Transaction Monitoring
- Customer Service Requests
- Account Reconciliation
- Multi-Channel Customer Communication

VINSYS®

Step Function – Task States

- Do some work in your state machine
- Invoke one AWS service
  - Can invoke a **Lambda function**
  - Run an AWS Batch job
  - Run an ECS task and wait for it to complete
  - Insert an item from DynamoDB
  - Publish message to SNS, SQS
  - Launch another Step Function workflow…
- Run an one Activity
  - EC2, Amazon ECS, on-premises
  - Activities poll the Step functions for work
  - Activities send results back to Step Functions



VINSYS

Example – Invoke Lambda Function

```
"Invoke Lambda function": {
    "Type": "Task",
    "Resource": "arn:aws:states:::lambda:invoke",
    "Parameters": {
        "FunctionName":
        "arn:aws:lambda:REGION:ACCOUNT_ID:function:FUNCTION_NAME",
        "Payload": {
            "Input.$": "$"
        }
    },
    "Next": "NEXT_STATE",
    "TimeoutSeconds": 300
}
```

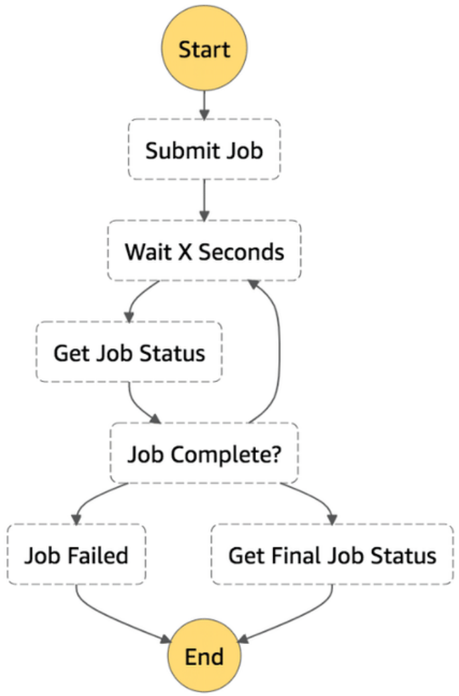
VINSYS

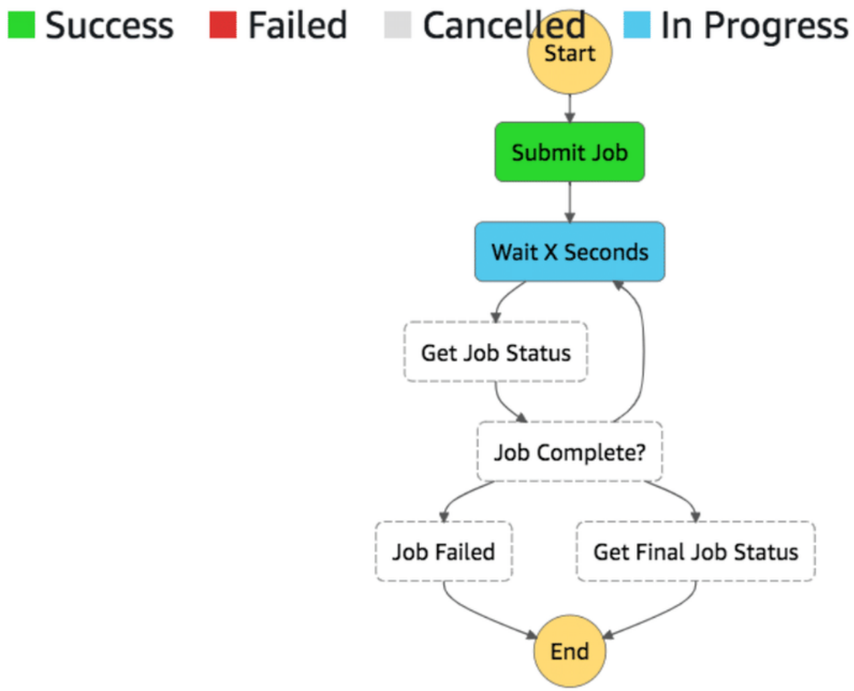
Step Function - States

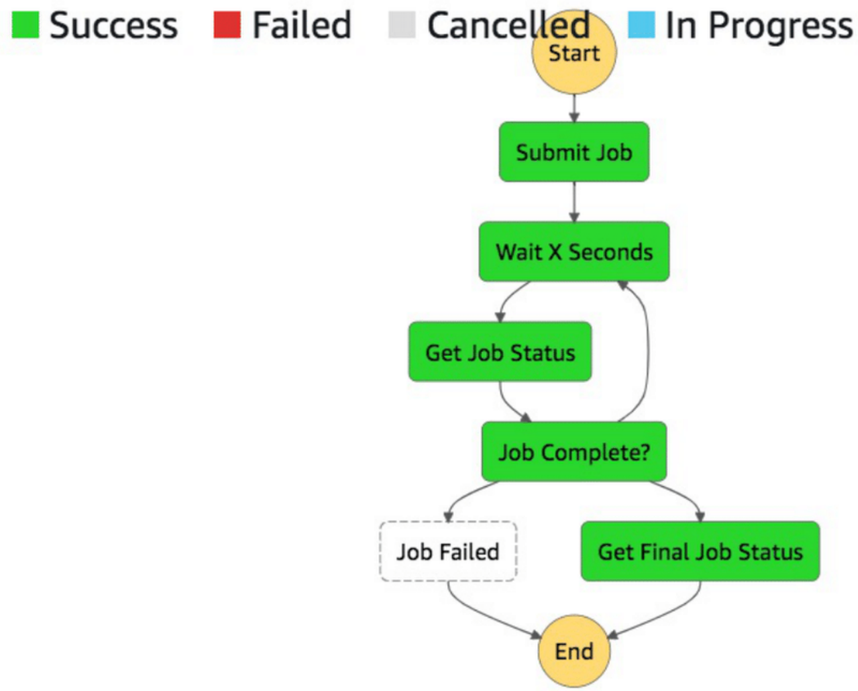
- **Choice State** - Test for a condition to send to a branch (or default branch)
- **Fail or Succeed State** - Stop execution with failure or success
- **Pass State** - Simply pass its input to its output or inject some fixed data, without performing work.
- **Wait State** - Provide a delay for a certain amount of time or until a specified time/date.
- **Map State** - Dynamically iterate steps.'
- ***Parallel State*** *- Begin parallel branches of execution.*

VINSYS®

Visual workflow in Step Functions







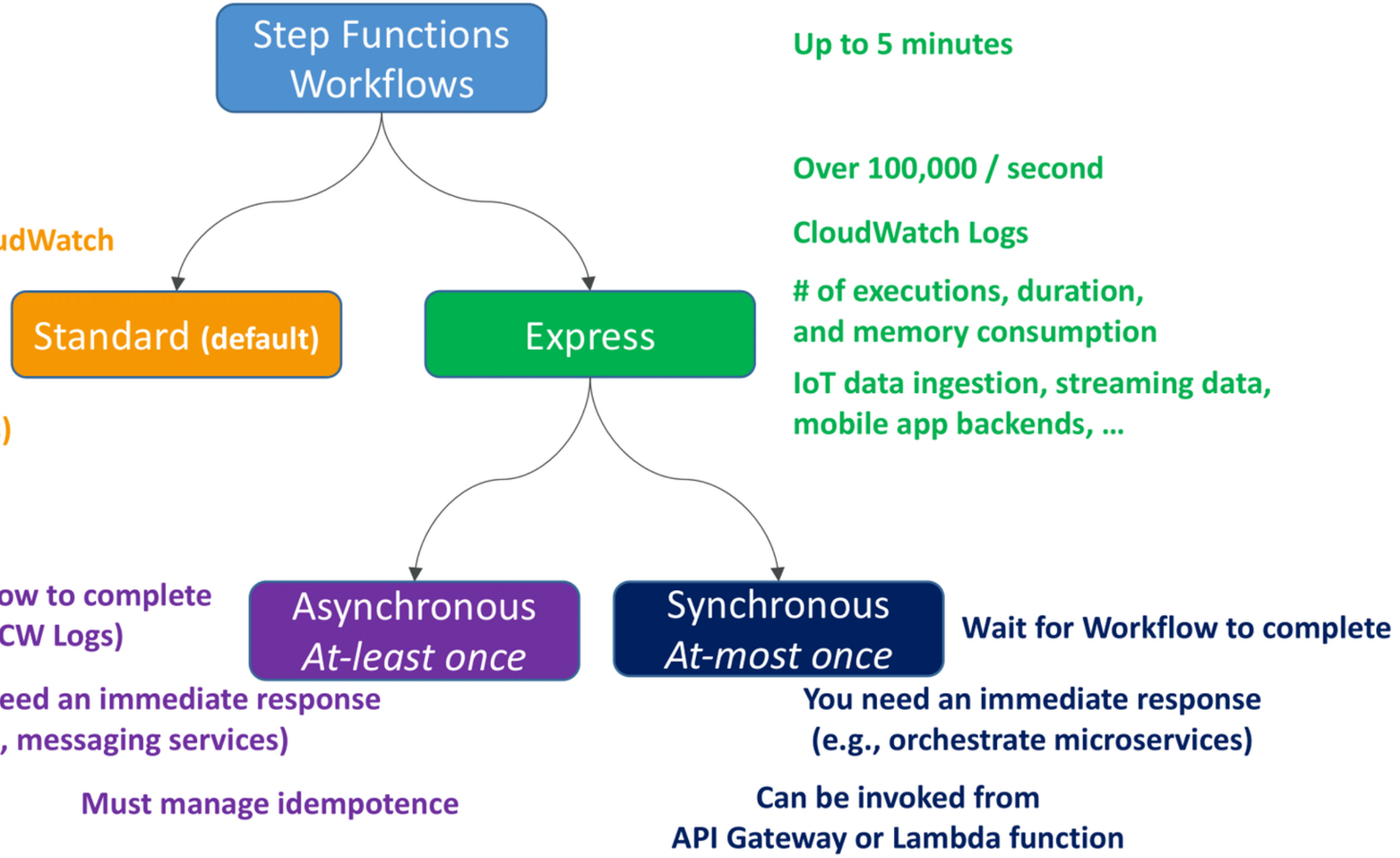
VINSYS®

Error Handling in Step Functions

- Any state can encounter runtime errors for various reasons:
  - State machine definition issues (for example, no matching rule in a Choice state)
  - Task failures (for example, an exception in a Lambda function)
  - Transient issues (for example, network partition events)
- Use **Retry** (to retry failed state) and **Catch** (transition to failure path) in the State Machine to handle the errors instead of inside the Application Code
- Predefined error codes:
  - States.ALL : matches any error name
  - States.Timeout: Task ran longer than TimeoutSeconds or no heartbeat received
  - States.TaskFailed: execution failure
  - States.Permissions: insufficient privileges to execute code
- The state may report is own errors

VINSYS®

Step Functions – Standard vs. Express



	Standard (default)	Express
Max. Duration	Up to 1 year	Up to 5 minutes
Execution Model	Exactly-once Execution	
Execution Rate	Over 2000 / second	Over 100,000 / second
Execution History	Up to 90 days or using CloudWatch	CloudWatch Logs
Pricing	# of State Transitions	# of executions, duration, and memory consumption
Use cases	Non-idempotent actions (e.g., processing Payments)	IoT data ingestion, streaming data, mobile app backends, …

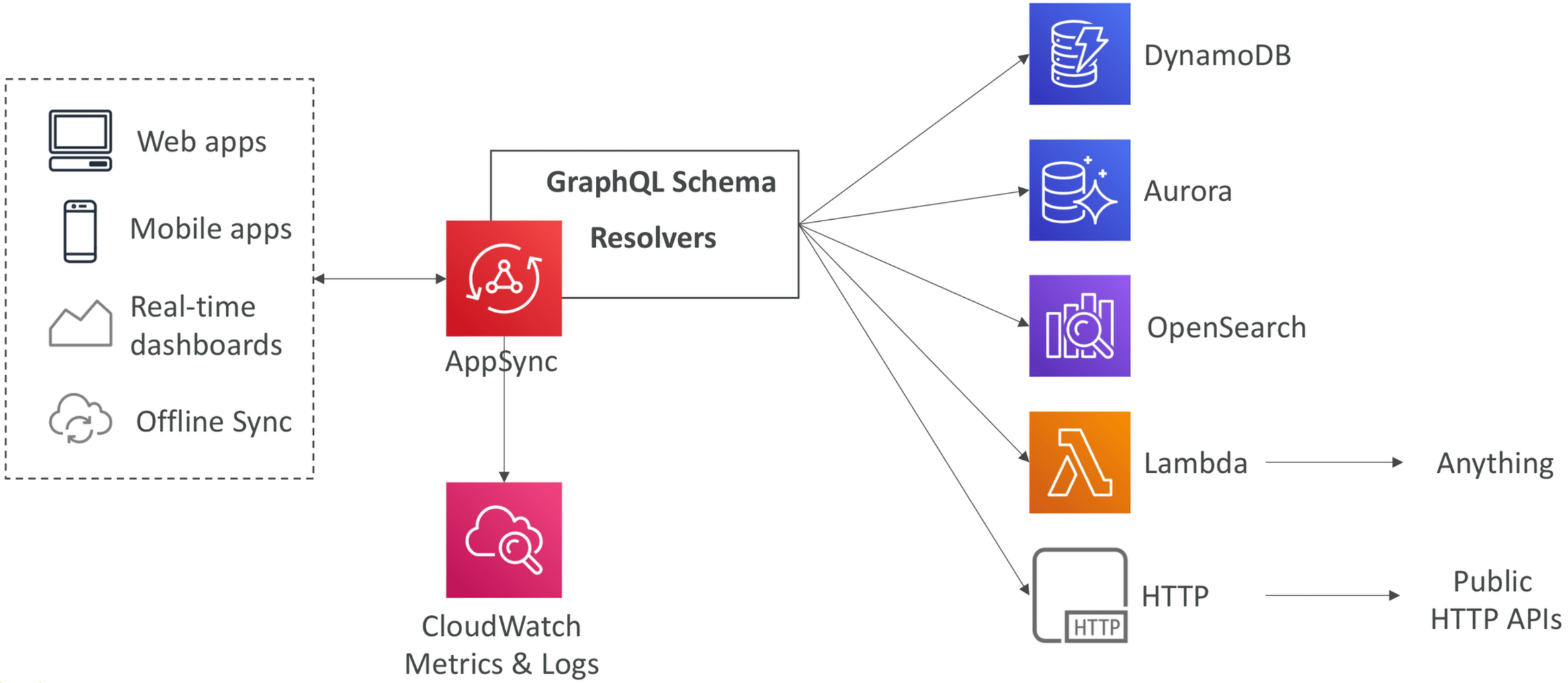
	Asynchronous *At-least once*	Synchronous *At-most once*
	Doesn't wait for Workflow to complete (get results from CW Logs)	Wait for Workflow to complete
	You don't need an immediate response (e.g., messaging services)	You need an immediate response (e.g., orchestrate microservices)
	Must manage idempotence	Can be invoked from API Gateway or Lambda function

VINSYS®

AWS AppSync - Overview

- **AppSync** is a managed service that uses **GraphQL**
- **GraphQL** makes it easy for applications to get exactly the data they need.
- This includes combining data from **one or more sources**
  - NoSQL data stores, Relational databases, HTTP APIs…
  - Integrates with DynamoDB, Aurora, OpenSearch & others
  - Custom sources with AWS Lambda
- Retrieve data in **real-time with WebSocket or MQTT on WebSocket**
- For mobile apps: local data access & data synchronization
- It all starts with uploading one **GraphQL schema**

AppSync Diagram



VINSYS

Use Cases of AppSync in Banking

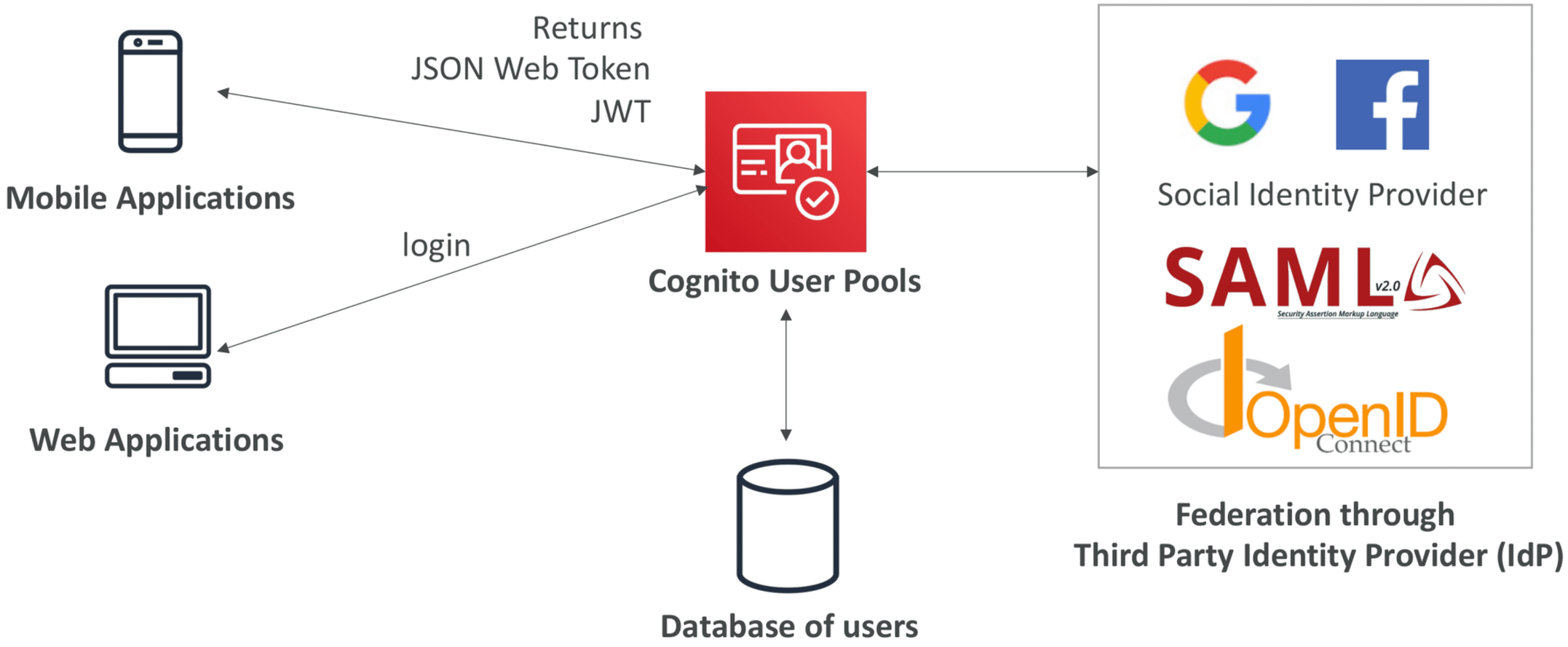
A bank wants to provide its customers with a secure and user-friendly way to retrieve their account information, including balances, transaction history, and recent activity, using a mobile banking application.

- Authentication and Authorization
- Data Sources Integration
- Schema Definition
- Offline Access
- Real-Time Updates / Notifications

VINSYS®

Amazon Cognito

Cognito User Pools (CUP) – Diagram



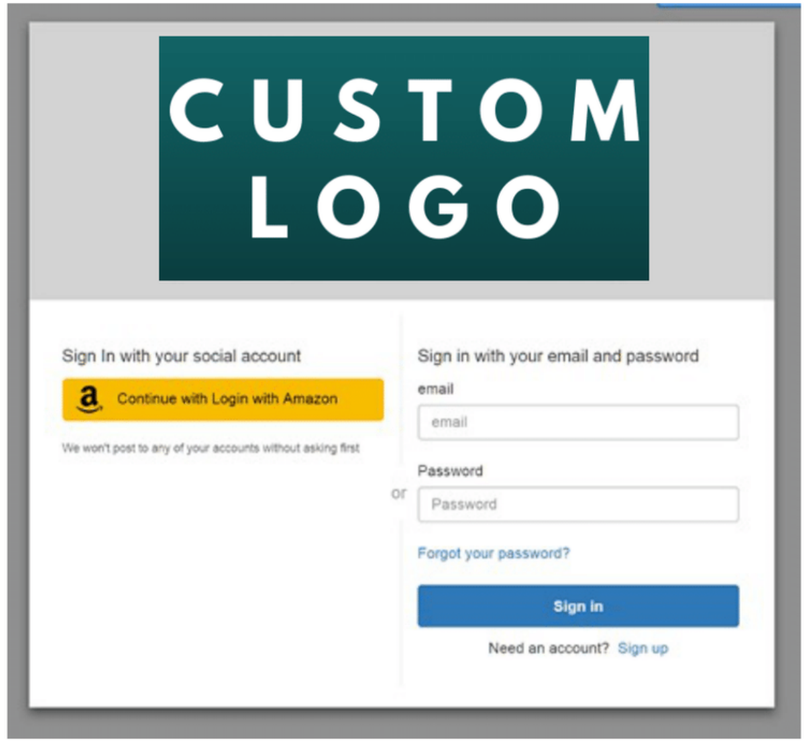
VINSYS

Cognito User Pools (CUP) – User Features

- Create a serverless database of user for your web & mobile apps
- Simple login: Username (or email) / password combination
- Password reset
- Email & Phone Number Verification
- Multi-factor authentication (MFA)
- Federated Identities: users from Facebook, Google, SAML…
- Feature: block users if their credentials are compromised elsewhere
- Login sends back a JSON Web Token (JWT)

Cognito User Pools – Hosted Authentication UI

- Cognito has a **hosted authentication UI** that you can add to your app to handle sign-up and sign-in workflows
- Using the hosted UI, you have a foundation for integration with social logins, OIDC or SAML
- Can customize with a **custom logo** and **custom CSS**



https://aws.amazon.com/blogs/aws/launch-amazon-cognito-user-pools-general-availability-app-integration-and-federation/

VINSYS

AWS Serverless Application Model (SAM)

Taking your Serverless Development to the next level

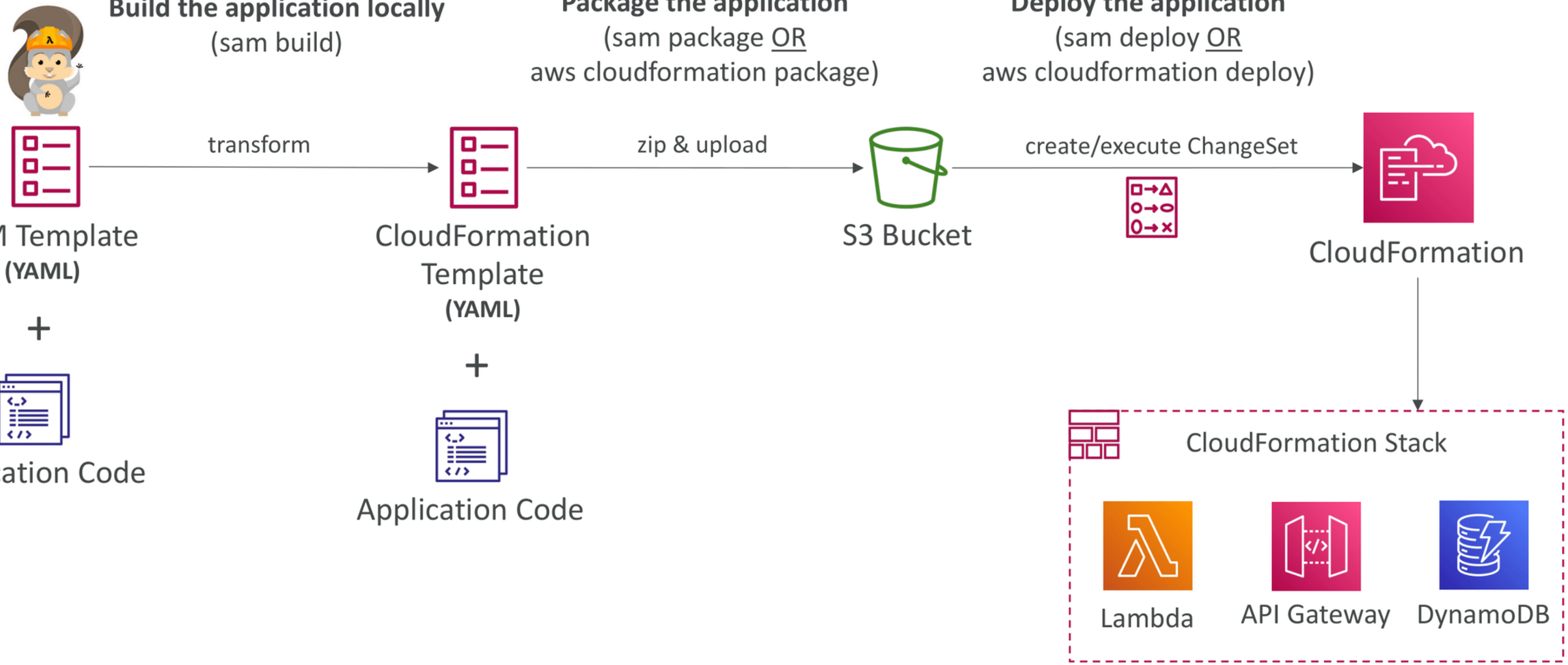
VINSYS

AWS SAM

- SAM = Serverless Application Model
- Framework for developing and deploying serverless applications
- All the configuration is YAML code
- Generate complex CloudFormation from simple SAM YAML file
- Supports anything from CloudFormation: Outputs, Mappings, Parameters, Resources…
- Only two commands to deploy to AWS
- SAM can use CodeDeploy to deploy Lambda functions
- SAM can help you to run Lambda, API Gateway, DynamoDB locally

VINSYS

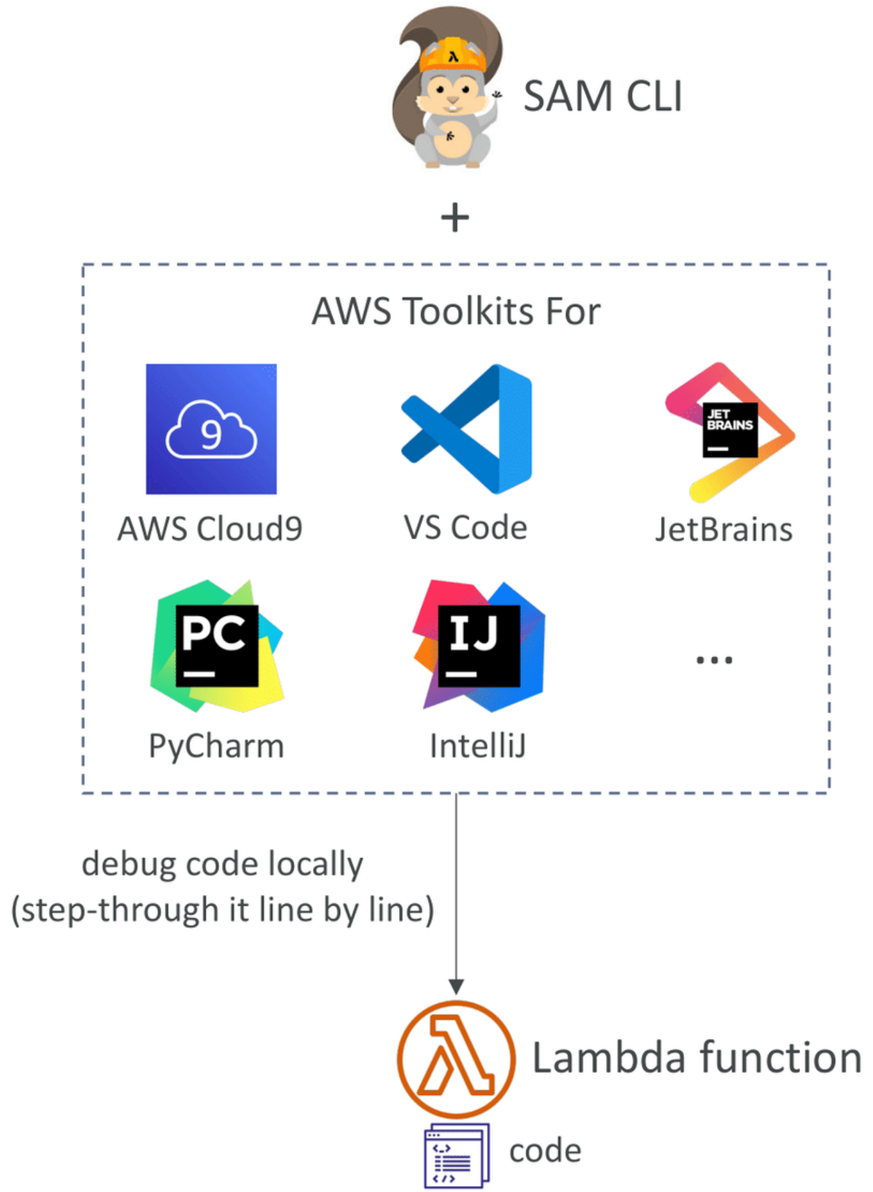
Deep Dive into SAM Deployment



VINSYS

SAM – CLI Debugging

- Locally build, test, and debug your serverless applications that are defined using AWS SAM templates
- Provides a lambda-like execution environment locally
- SAM CLI + AWS Toolkits => step-through and debug your code
- Supported IDEs: AWS Cloud9, Visual Studio Code, JetBrains, PyCharm, IntelliJ, …
- **AWS Toolkits:** IDE plugins which allows you to build, test, debug, deploy, and invoke Lambda functions built using AWS SAM



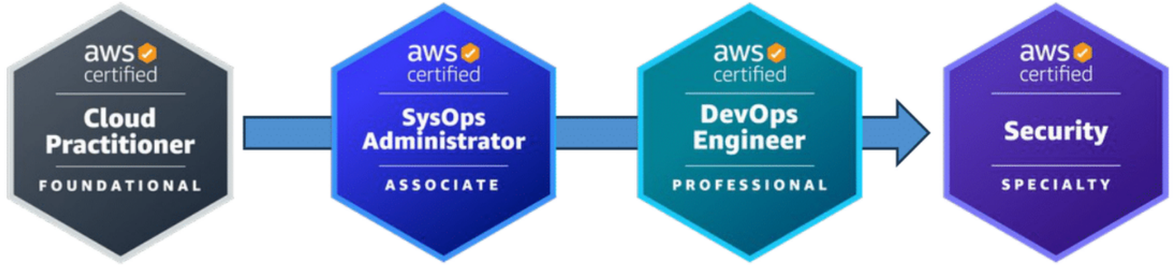
VINSYS

AWS Certification Paths – DevOps

DevOps
Test Engineer
Embed testing and quality best practices for software development from design to release, throughout the product life cycle

DevOps
Cloud DevOps Engineer
Design, deployment, and operations of large-scale global hybrid cloud computing environment, advocating for end-to-end automated CI/CD DevOps pipelines

DevOps
DevSecOps Engineer
Accelerate enterprise cloud adoption while enabling rapid and stable delivery of capabilities using CI/CD principles, methodologies, and technologies



VINSYS®