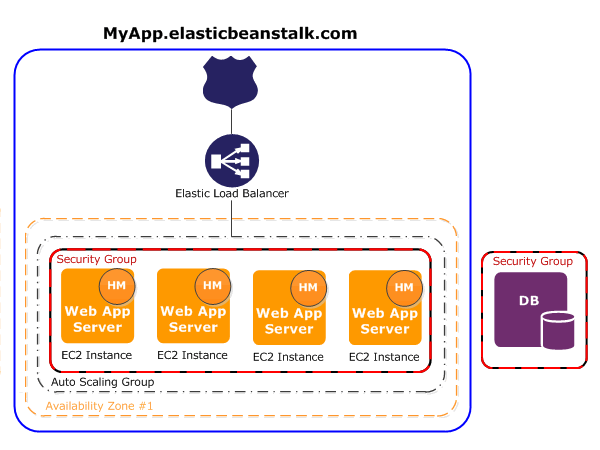
**Amazon Elastic Beanstalk**

* Amazon Elastic Beanstalk is an environment within which we can run our web application, handle their HTTP(S) requests or handle background-processing tasks.
* We can also run EC2 instances in an Elastic Beanstalk.
* Elastic Beanstalk provides platforms:
  + Supports programming languages (Java, PHP, Python, Ruby, Go)
  + Web containers (Tomcat, Passenger, Puma)
  + Docker containers with multiple configurations of each.



* Elastic Beanstalk will setup an "environment" for you that can contain:
  + A number of EC2 instances
  + An optional database
  + Elastic Load Balancer
  + Auto-Scaling Group
  + Security Group.

**Amazon EC2**

* Amazon EC2 provides fully scalable ,lightweight, fast, secure and integrated web services with instances (server).
* We can deploy our desired Operating systems as EC2 instances and scale them whenever we need.
* Amazon EC2 is integrated with most AWS services such as :
  + Amazon Simple Storage Service (Amazon S3)
  + Amazon Relational Database Service (Amazon RDS)
  + Amazon Virtual Private Cloud (Amazon VPC)

**AWS S3**

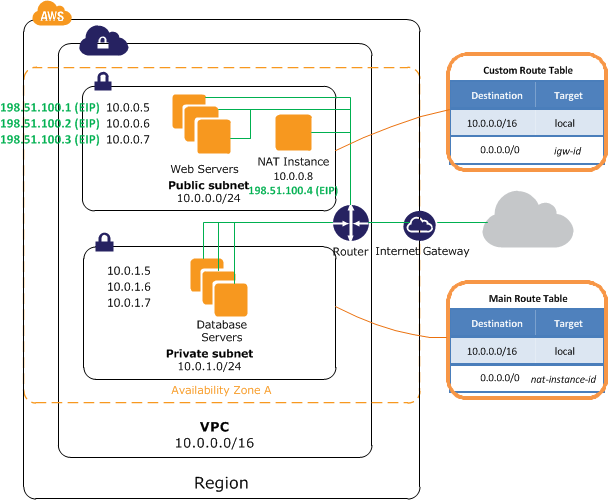
* Amazon S3 is an object storage capable of storing very large objects, upto 5TB in size.
* S3 is typically used for storing images, videos, logs and other types of files. There is no limit on the number of objects that can be stored in and S3 bucket.
* Each object in S3 has a url which can be used to download the object.
* Objects in S3 can also be delivered via Amazon Cloudfront CDN(Content Delivery network).
* Objects in S3 can be archived to Amazon Glacier which is a very cheap archival service.

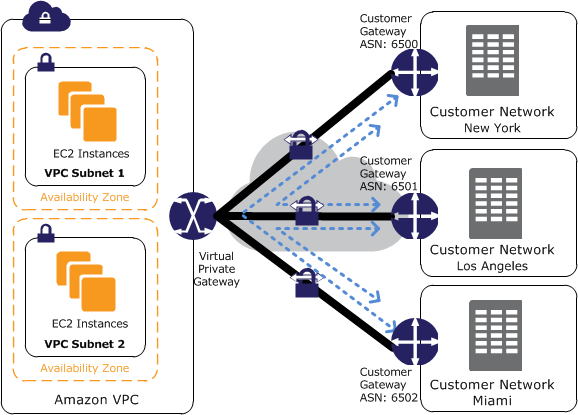
**Amazon RDS (Relational Database Service)**

* Amazon Relational Database Service (Amazon RDS) makes it easy to set up, operate, and scale a [relational database](https://aws.amazon.com/relational-database/) in the cloud.
* It provides cost-efficient and resizable capacity while automating time-consuming administration tasks such as hardware provisioning, database setup, patching and backups.
* It frees you to focus on your applications so you can give them the fast performance, high availability, security and compatibility they need.
* Amazon RDS is available on several database instance types and provides you with six familiar database engines to choose from:
  + [Amazon Aurora](https://aws.amazon.com/rds/aurora/)
  + [PostgreSQL](https://aws.amazon.com/rds/postgresql/)
  + [MySQL](https://aws.amazon.com/rds/mysql/)
  + [MariaDB](https://aws.amazon.com/rds/mariadb/)
  + [Oracle](https://aws.amazon.com/rds/oracle/)
  + [Microsoft SQL Server](https://aws.amazon.com/rds/sqlserver/)
* You can use the [AWS Database Migration Service](https://aws.amazon.com/dms/) to easily migrate or replicate your existing databases to Amazon RDS.

**AWS Virtual Private Cloud (VPC)**

* Amazon Virtual Private Cloud (Amazon VPC) lets you provision a logically isolated section of the Amazon Web Services (AWS) cloud where you can launch AWS resources in a virtual network that you define.
* You have complete control over your virtual networking environment, including selection of your own IP address range, creation of subnets, and configuration of route tables and network gateways.
* You can use both IPv4 and IPv6 in your VPC for secure and easy access to resources and applications.
* Using VPC, you can create a public-facing subnet for your web servers that has access to the Internet, and place your backend systems such as databases or application servers in a private-facing subnet with no Internet access.
* You can leverage multiple layers of security, including security groups and network access control lists, to help control access to Amazon EC2 instances in each subnet.
* You can use VPC to:
  + Host a simple, public-facing website
  + Host multi-tier web applications
  + Host scalable web applications in AWS Cloud that are connected to your datacenter
  + Disaster Recovery
    - You can periodically backup your mission critical data from your datacenter to a small number of Amazon EC2 instances with Amazon Elastic Block Store (EBS) volumes, or import your virtual machine images to Amazon EC2.
    - In the event of a disaster in your own datacenter, you can quickly launch replacement compute capacity in AWS to ensure business continuity.
    - When the disaster is over, you can send your mission critical data back to your datacenter and terminate the Amazon EC2 instances that you no longer need.



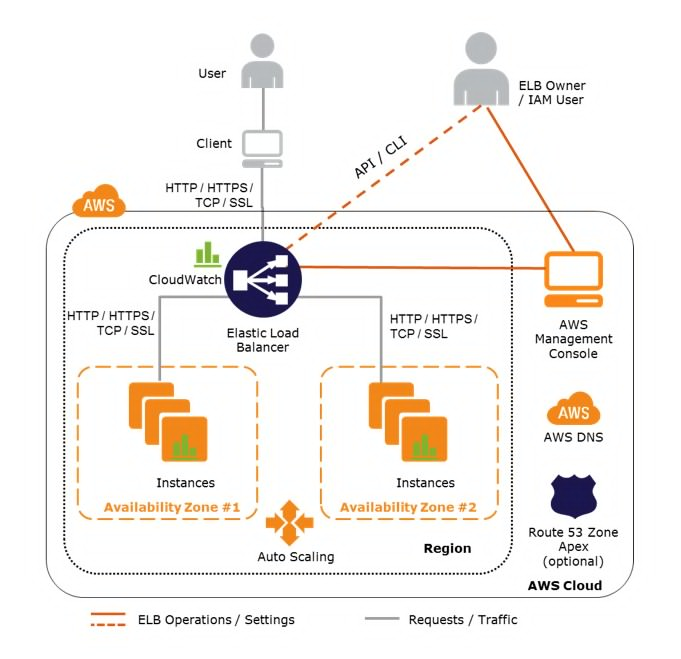
****

**Amazon Elastic Block Store (EBS)**

* Amazon Elastic Block Store (Amazon EBS) provides persistent block storage volumes for use with [Amazon EC2](https://aws.amazon.com/ec2-sla/) instances in the AWS Cloud.
* Each Amazon EBS volume is automatically replicated within its Availability Zone to protect you from component failure, offering high availability and durability.
* Amazon EBS volumes offer the consistent and low-latency performance needed to run your workloads.
* Typical use cases include:
  + Big Data analytics engines (like the Hadoop/HDFS ecosystem and [Amazon EMR](https://aws.amazon.com/emr/) clusters)
  + Relational and NoSQL databases (like Microsoft SQL Server, MySQL or Cassandra, MongoDB)
  + Stream and log processing applications (like Kafka and Splunk)
  + Data warehousing applications (like Vertica and Teradata).

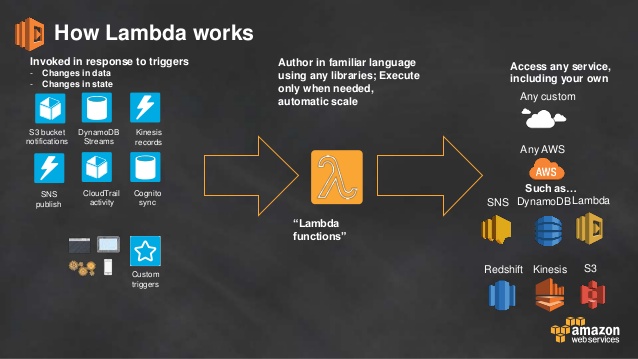
**Elastic Load Balancer**

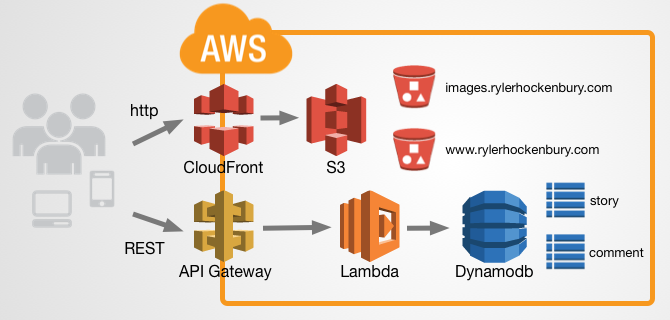
* Elastic Load Balancing automatically distributes incoming application traffic across multiple Amazon EC2 instances.
* It enables you to achieve fault tolerance in your applications, seamlessly providing the required amount of load balancing capacity needed to route application traffic.
* Elastic Load Balancing offers two types of load balancers that both feature high availability, automatic scaling, and robust security.These include
  + [Classic Load Balancer](https://aws.amazon.com/elasticloadbalancing/classicloadbalancer/) that routes traffic based on either application or network level information
  + [Application Load Balancer](https://aws.amazon.com/elasticloadbalancing/applicationloadbalancer/) that routes traffic based on advanced application level information that includes the content of the request.

****

**AWS Lambda**

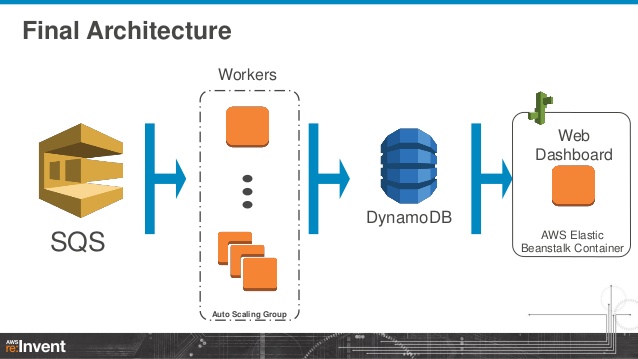
* AWS Lambda lets you run code without provisioning or managing servers.
* With Lambda, you can run code for virtually any type of application or backend service - all with zero administration.
* Use AWS Lambda to execute code in response to triggers such as changes in data, shifts in system state, or actions by users.
* Lambda can be directly triggered by AWS services such as
  + S3
  + DynamoDB
  + Kinesis
  + SNS
  + CloudWatch
* You can set up your code to automatically trigger from other AWS services or call it directly from any web or mobile app.
* AWS Lambda can be connected to any RDS or NoSQL Database which is installed within the Elastic Beanstalk environment.

****

****

**DynamoDB**

* DynamoDB on the other hand is a NoSQL database that can be used as a key value or a document(schema less record) store.
* DynamoDB is typically useful for storing a large number of small records with single digit millisecond latency.
* DynamoDB record size is limited to 64KB.
* Amazon DynamoDB Accelerator (DAX) is a fully managed, highly available, in-memory cache that can reduce Amazon DynamoDB response times from milliseconds to microseconds, even at millions of requests per second.

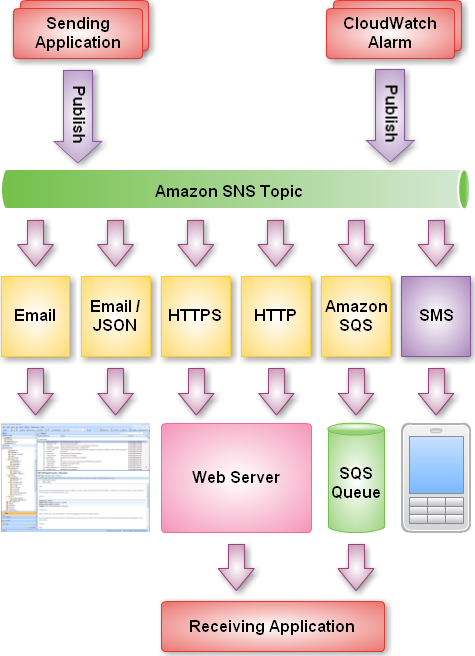


**Amazon SNS (Simple Notification Service)**

* Amazon Simple Notification Service (SNS) is a flexible, fully managed [pub/sub messaging](https://aws.amazon.com/pub-sub-messaging/) and mobile notifications service for coordinating the delivery of messages to subscribing endpoints and clients.
* Amazon SNS is a fully managed pub/sub messaging service that makes it easy to decouple and scale microservices, distributed systems, and serverless applications.
* With SNS, you can use topics to decouple message publishers from subscribers, fan-out messages to multiple recipients at once, and eliminate polling in your applications.
* AWS services, such as
  + Amazon EC2
  + Amazon S3
  + Amazon CloudWatch

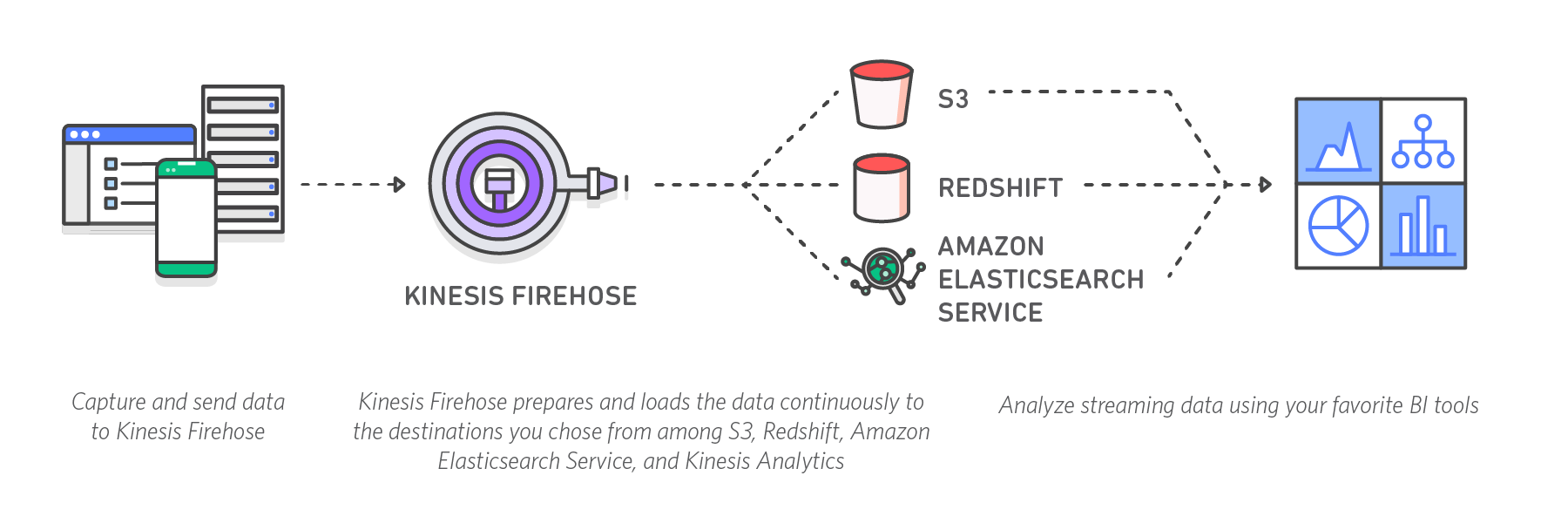
can publish messages to your SNS topics to trigger event-driven computing and workflows.

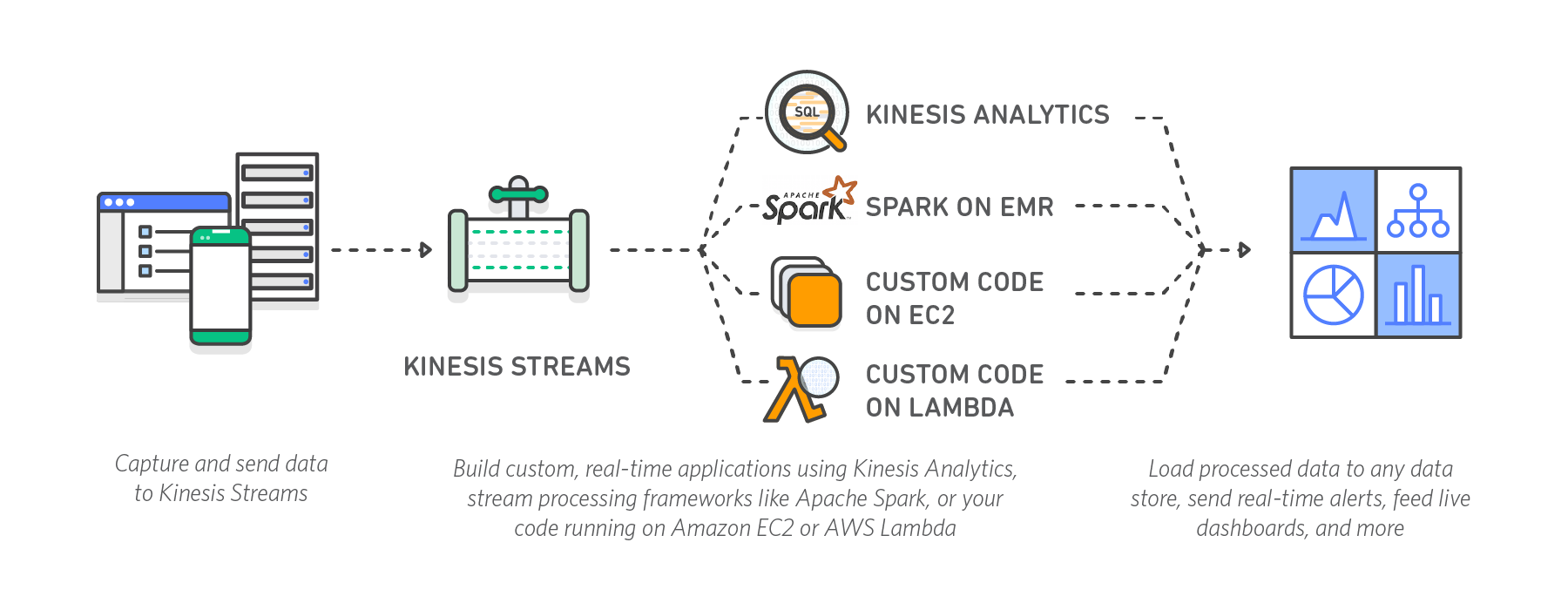
* SNS supports a variety of subscription types, allowing you to push messages directly to [Amazon Simple Queue Service (SQS)](https://aws.amazon.com/sqs/) queues, AWS Lambda functions, and HTTP endpoints.
* Amazon SNS Mobile Notifications makes it simple and cost effective to send push notifications to iOS, Android, Fire OS, Windows and Baidu-based devices.

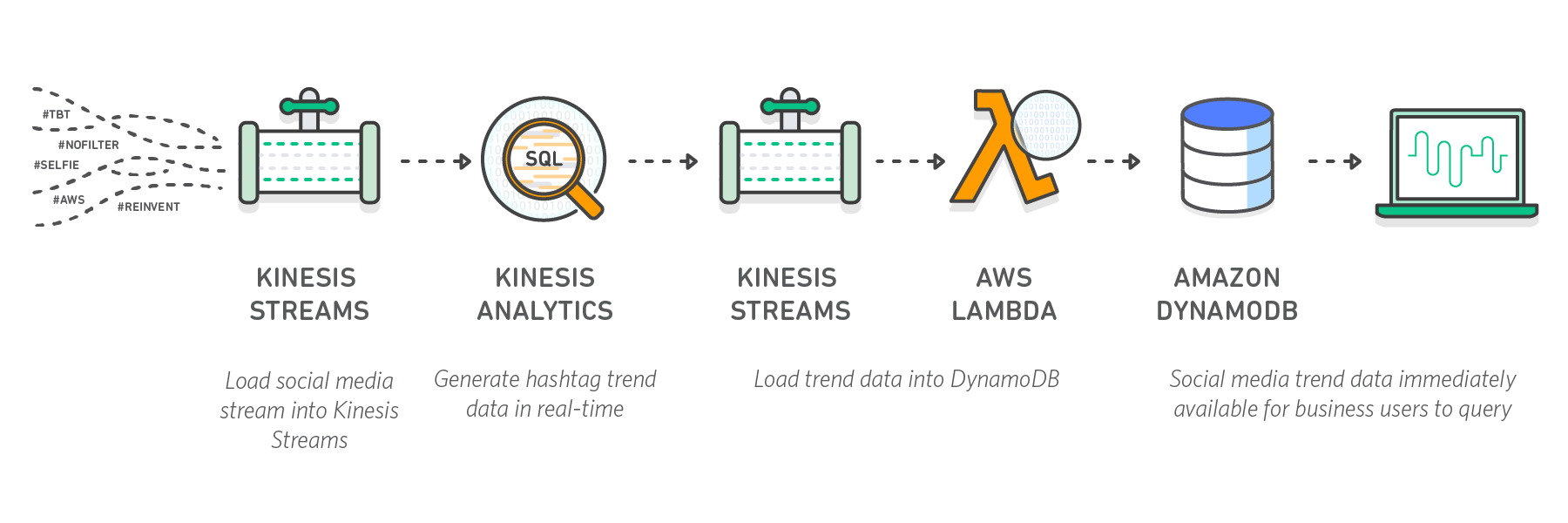


**Amazon Kinesis**

* Amazon Kinesis makes it easy to collect, process, and analyze [real-time, streaming data](https://aws.amazon.com/streaming-data/) so you can get timely insights and react quickly to new information.
* Amazon Kinesis offers [key capabilities](https://aws.amazon.com/kinesis/?sc_channel=PS&sc_campaign=acquisition_IN&sc_publisher=google&sc_medium=kinesis_b&sc_content=kinesis_e&sc_detail=aws%20kinesis&sc_category=kinesis&sc_segment=159808191781&sc_matchtype=e&sc_country=IN&s_kwcid=AL!4422!3!159808191781!e!!g!!aws%20kinesis&ef_id=WYrhowAAAbhbODmz:20170809122526:s#kinesis-capabilities) to cost effectively process streaming data at any scale, along with the flexibility to choose the tools that best suit the requirements of your application.
* With Amazon Kinesis, you can ingest real-time data such as application logs, website clickstreams, IoT telemetry data, and more into your databases, data lakes and data warehouses, or build your own real-time applications using this data.
* Amazon Kinesis enables you to process and analyze data as it arrives and respond in real-time instead of having to wait until all your data is collected before the processing can begin.

****

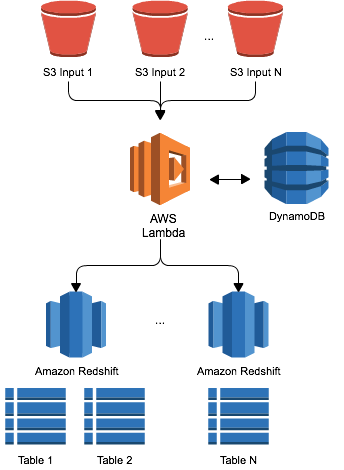
****

****

* With Amazon Kinesis, you can perform real-time analytics on data that has been traditionally analyzed using batch processing in data warehouses or using Hadoop frameworks.
* The most common use cases include:
  + Data lakes
  + Data science
  + Machine learning
  + You can use Kinesis Firehose to continuously load streaming data into your S3 data lakes. You can also update machine learning models more frequently as new data becomes available, ensuring accuracy and reliability of the outputs.

**Amazon Redshift**

* Amazon Redshift is a fast, fully managed [data warehouse](https://aws.amazon.com/data-warehouse/) that makes it simple and cost-effective to analyze all your data using standard SQL and your existing Business Intelligence (BI) tools.
* It allows you to run complex analytic queries against petabytes of structured data, using sophisticated query optimization, columnar storage on high-performance local disks, and massively parallel query execution.
* Amazon Redshift also includes [Redshift Spectrum](https://aws.amazon.com/redshift/spectrum/), allowing you to directly run SQL queries against exabytes of unstructured data in [Amazon S3](https://aws.amazon.com/s3/).

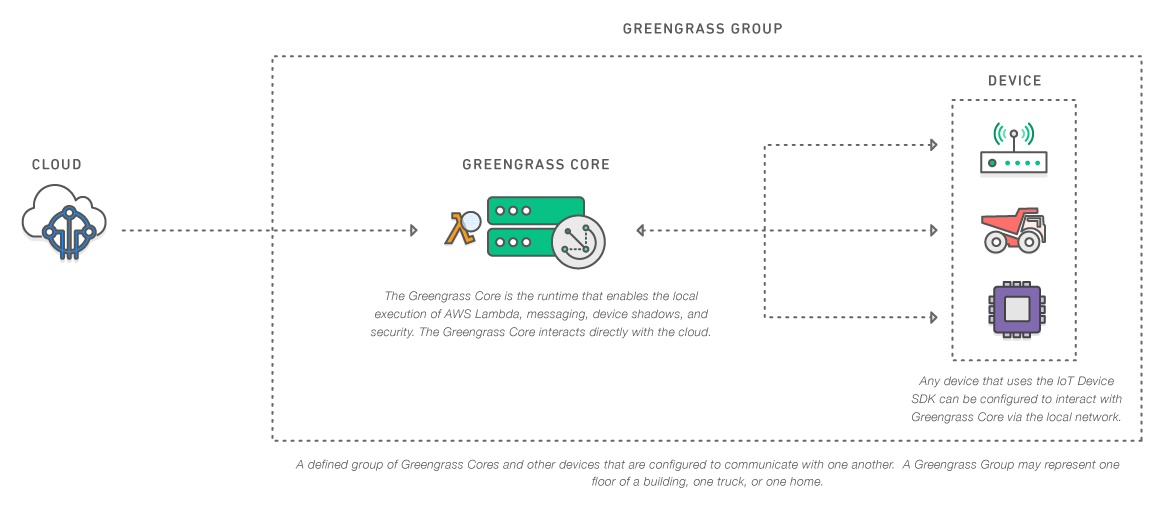
****

**Amazon ElastiCache**

* The service improves the performance of web applications by allowing you to retrieve information from fast, managed, in-memory data stores, instead of relying entirely on slower disk-based databases.
* Amazon ElastiCache supports two open-source in-memory engines:
  + [Redis](https://aws.amazon.com/elasticache/what-is-redis/) - a fast, open source, in-memory data store and cache. [Amazon ElastiCache for Redis](https://aws.amazon.com/elasticache/redis/) is a Redis-compatible in-memory service that delivers the ease-of-use and power of Redis along with the availability, reliability and performance suitable for the most demanding applications. Both single-node and up to 15-shard clusters are available, enabling scalability to up to 3.55 TiB of in-memory data.
  + Memcached - a widely adopted memory object caching system. ElastiCache is protocol compliant with Memcached, so popular tools that you use today with existing Memcached environments will work seamlessly with the service.

**AWS GreenGrass**

* AWS Greengrass is software that lets you run local compute, messaging, data caching, and sync capabilities for connected devices in a secure way.
* With AWS Greengrass, connected devices can run [AWS Lambda](https://aws.amazon.com/lambda/) functions, keep device data in sync, and communicate with other devices securely – even when not connected to the Internet.
* Using AWS Lambda, Greengrass ensures your IoT devices can respond quickly to local events, operate with intermittent connections, and minimize the cost of transmitting IoT data to the cloud.
* AWS Greengrass seamlessly extends AWS to devices so they can act locally on the data they generate, while still using the cloud for management, analytics, and durable storage.
* AWS Greengrass can be programmed to filter device data and only transmit necessary information back to the cloud.
* AWS Greengrass authenticates and encrypts device data at all points of connection using [AWS IoT’s](https://aws.amazon.com/iot-platform/) security and access management capabilities.
* Devices that run [Linux](https://aws.amazon.com/greengrass/faqs/) and support ARM or x86 architectures can host the Greengrass Core. The Greengrass Core enables the local execution of [AWS Lambda](https://aws.amazon.com/lambda/) code, messaging, data caching, and security.
* AWS Greengrass Core devices and the AWS IoT Device SDK-enabled devices can be configured to communicate with one another in a Greengrass Group.
* If the Greengrass Core device loses connection to the cloud, devices in the Greengrass Group can continue to communicate with each other over the local network. A Greengrass Group may represent one floor of a building, one truck, or one home.



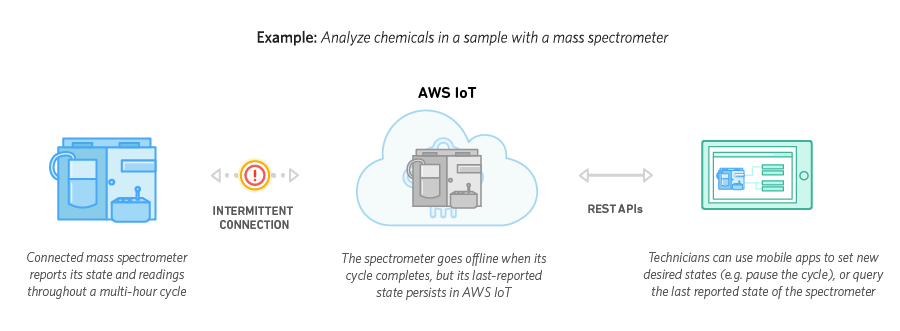
**AWS IoT**

* AWS IoT is a managed cloud platform that lets connected devices easily and securely interact with cloud applications and other devices.
* AWS IoT can support billions of devices and trillions of messages, and can process and route those messages to AWS endpoints and to other devices reliably and securely.
* With AWS IoT, your applications can keep track of and communicate with all your devices, all the time, even when they aren’t connected.
* AWS IoT makes it easy to use AWS services like
  + AWS Lambda
  + Amazon Kinesis
  + Amazon S3
  + Amazon Machine Learning
  + Amazon DynamoDB
  + Amazon CloudWatch
  + AWS CloudTrail
  + Amazon Elasticsearch Service with built-in Kibana integration

to build IoT applications that gather, process, analyze and act on data generated by connected devices, without having to manage any infrastructure.

### AWS IoT supports HTTP, WebSockets, and MQTT, a lightweight communication protocol specifically designed to tolerate intermittent connections, minimize the code footprint on devices, and reduce network bandwidth requirements.

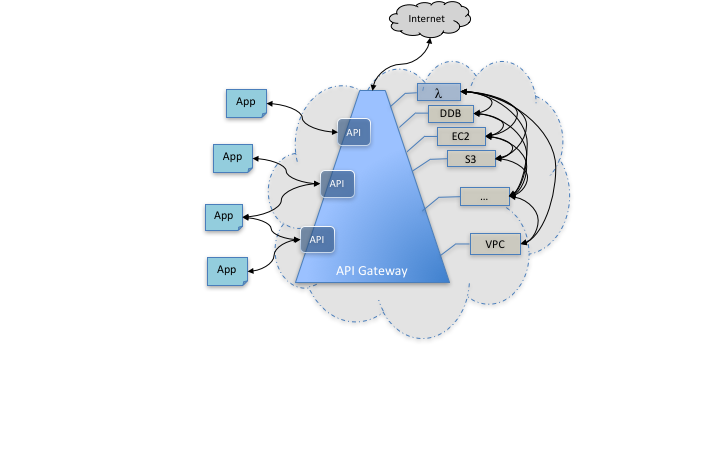
### With AWS IoT, you can filter, transform, and act upon device data on the fly, based on business rules you define.



**AWS API Gateway**

Amazon API Gateway is an AWS service that enables developers to create, publish, maintain, monitor, and secure APIs at any scale. You can create APIs that access AWS or other web services, as well as data stored in the AWS Cloud.

API Gateway can be considered a [backplane](https://en.wikipedia.org/wiki/Backplane) in the cloud to connect [AWS services](https://aws.amazon.com/) and other public or private web sites. It provides consistent RESTful application programming interfaces (APIs) for mobile and web applications to access AWS services.



API Gateway lets you create, configure, and host a [RESTful API](http://docs.aws.amazon.com/apigateway/api-reference/resource/rest-api/" \t "/home/abhay/Documents\\x/_blank) to enable applications to access the AWS Cloud.

For example, an application using API Gateway can upload a user's annual income and expense data to Amazon S3 or Amazon DynamoDB, process the data in AWS Lambda to compute tax owed, and file a tax return via the IRS website.

AWS API Gateway with AWS Lambda:

Together with AWS Lambda, API Gateway forms the app-facing part of the AWS serverless infrastructure. For an app to call publicly available AWS services, you can use Lambda to interact with the required services and expose the Lambda functions through API methods in API Gateway.

A method corresponds to a REST API request submitted by the user of your API and the corresponding response returned to the user. The app does not need to know where the requested data is stored and fetched from. The API interfaces with the backend using the integration request or integration response.

For example, with DynamoDB as the backend, the API developer sets up the integration request to forward the incoming method request to the chosen backend. The setup includes specifications of an appropriate DynamoDB action, required IAM role and policies, and required input data transformation. The backend returns the result to API Gateway as an integration response. To route the integration response to an appropriate method response (of a given HTTP status code) to the client, you can configure the integration response to map required response parameters from integration to method. You then translate the output data format of the backend to that of the frontend, if necessary. API Gateway enables you to define a schema or model for the payload to facilitate setting up the body mapping template.

## Benefits of API Gateway

API Gateway helps you deliver

* Robust
* Secure
* scalable mobile and web application backends.

API Gateway allows you to securely connect mobile and web applications to business logic hosted on AWS Lambda, APIs hosted on Amazon EC2, or other publicly addressable web services hosted inside or outside of AWS. With API Gateway, you can create and operate APIs for backend services. For example, you don't need to develop and maintain infrastructure to handle authorization and access control, traffic management, monitoring and analytics, version management, and software development kit (SDK) generation.

**Important Concepts of an API Gateway:**

**API Gateway API:** A collection of resources and methods that are integrated with backend HTTP endpoints, Lambda functions, or other AWS services. The collection can be deployed in one or more stages. API methods are invoked through frontend HTTP endpoints that you can associate with a registered custom domain name. Permissions to invoke a method are granted using IAM roles and policies or API Gateway custom authorizers. An API can present a certificate to be authenticated by the backend. Typically, API resources are organized in a resource tree according to the application logic. Each API resource can expose one or more API methods that must have unique HTTP verbs supported by API Gateway.

**API developer or API owner:** An AWS account that owns an API Gateway deployment (for example, a service provider that also supports programmatic access.)

**App developer or client developer:** An app creator who may or may not have an AWS account and interacts with the API deployed by the API developer. An app developer can be represented by an API key.

**App user, end user, or client endpoint:** An entity that uses the application built by an app developer. The entity interacts with APIs in Amazon API Gateway. An app user can be represented by an Amazon Cognito identity or a bearer token.

**API key:** An alphanumeric string that is generated by API Gateway on behalf of an API owner or imported from an external source such as a CSV file. The string is used to identify an app developer of the API. An API owner can use API keys to permit or deny access of specific APIs based on the apps in use.

**API deployment and stage:** An API deployment is a point-in-time snapshot of the API Gateway API resources and methods. For a deployment to be accessible for a client to invoke, the deployment must be associated with one or more stages. A stage is a logical reference to a lifecycle status of your API (for example, 'dev', 'prod', 'beta', 'v2'). The identifier of an API stage consists of an API ID and stage name.