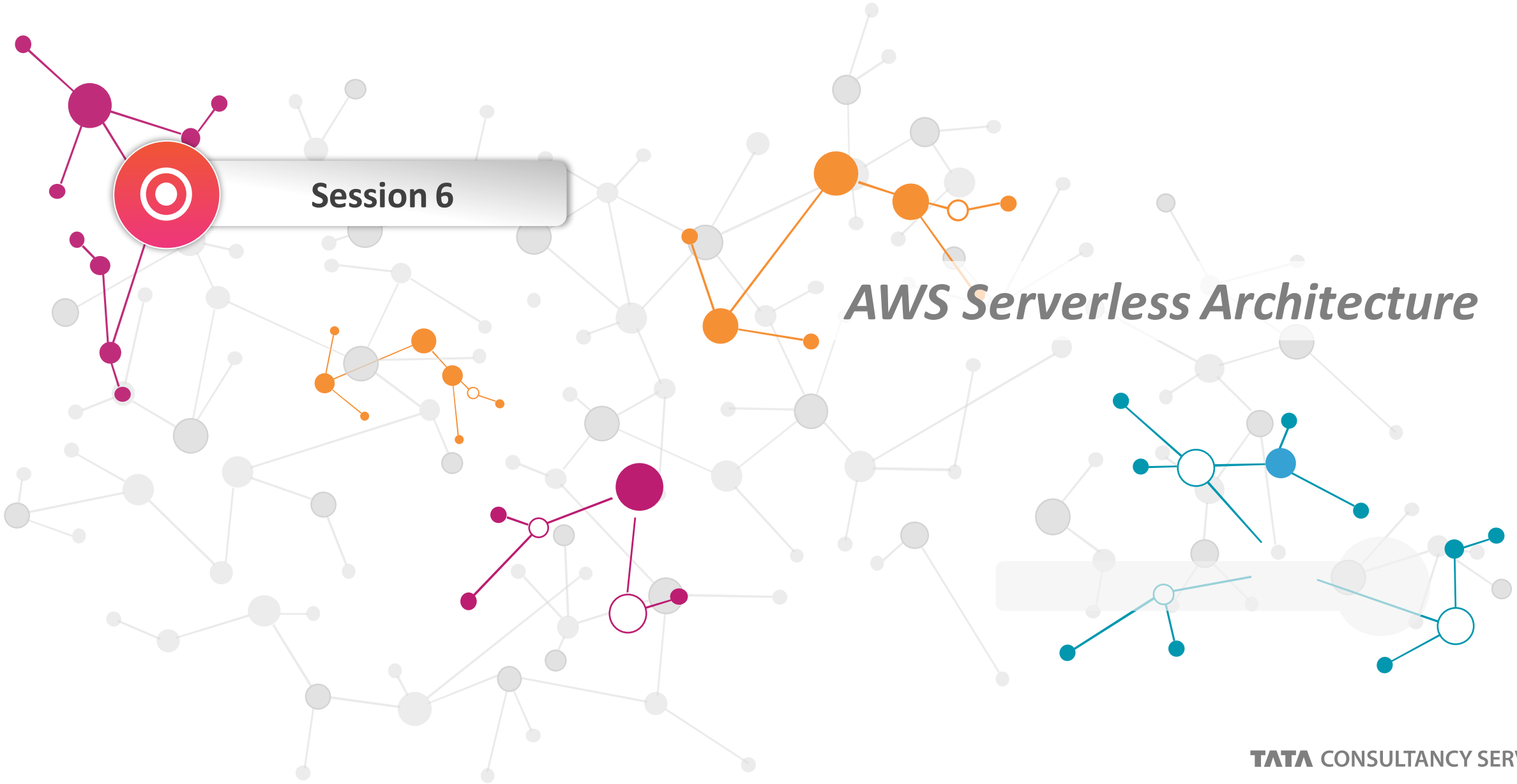
AWS Academy

North America
Talent Development Team

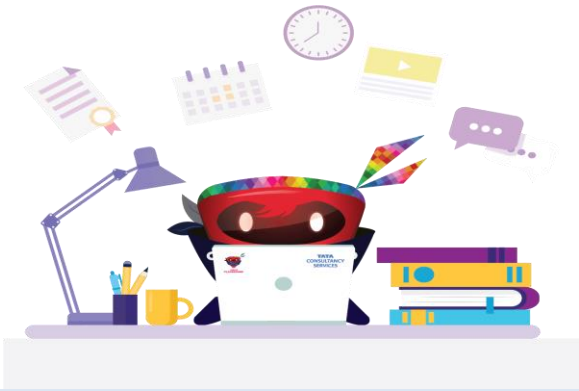
Date



AWS Serverless Architecture

- API Gateway
- Elastic Beanstalk
- Lambda
- Cognito
- Decoupled and Multi-Tier architecture

Module # 1

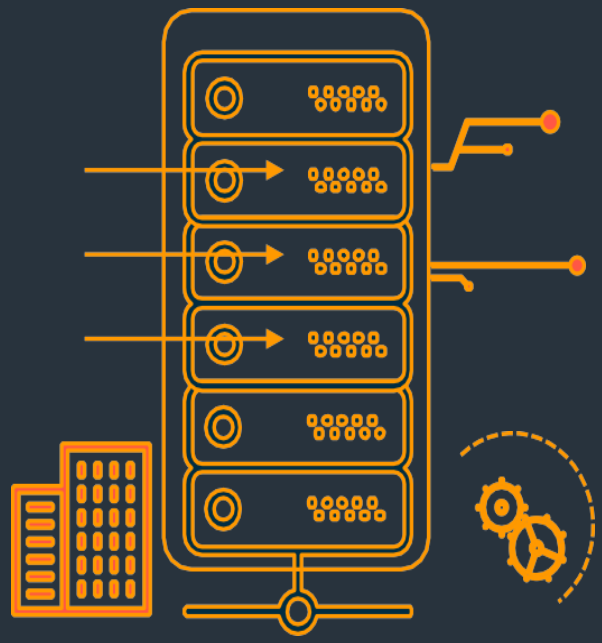


Serverless

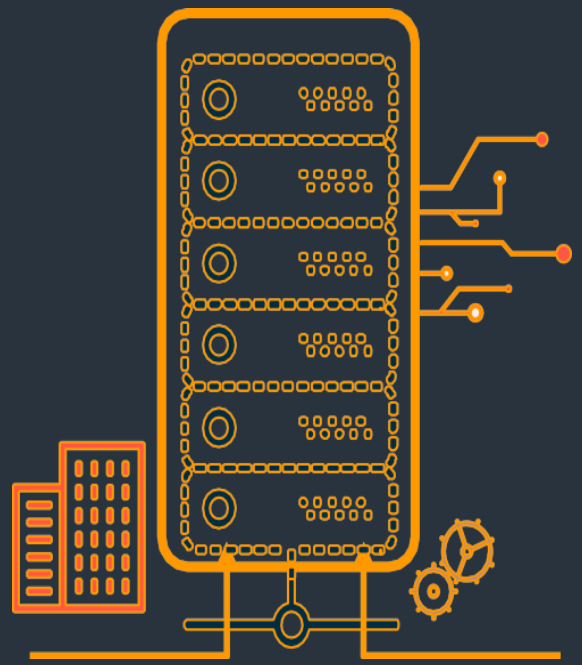
Evolution of computing

Evolution of computing

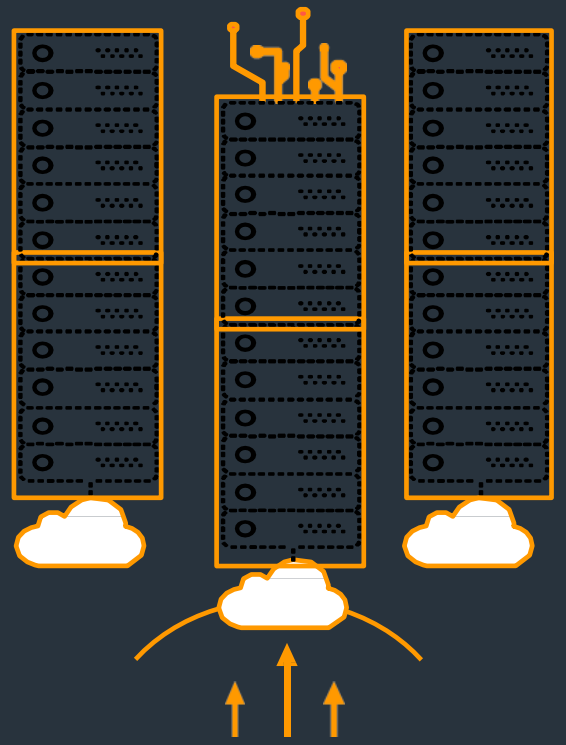
Physical Servers in Datacenters



Virtual Servers in Datacenters



Virtual Servers in the Cloud

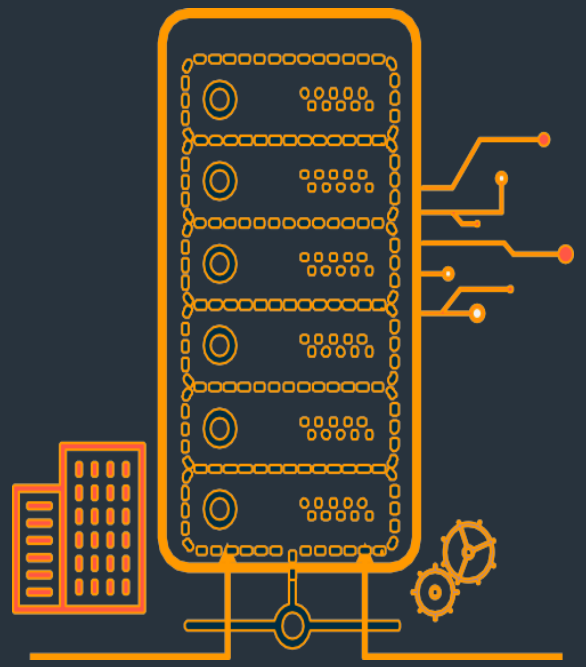


Each progressive step was better

- Higher utilization
- Faster provisioning speed
- Improved uptime
- Disaster recovery
- Hardware independence

- Trade CAPEX for OPEX
- More scale
- Elastic resources
- Faster speed and agility
- Reduced maintenance
- Better availability and fault tolerance

Virtual Servers in Datacenters

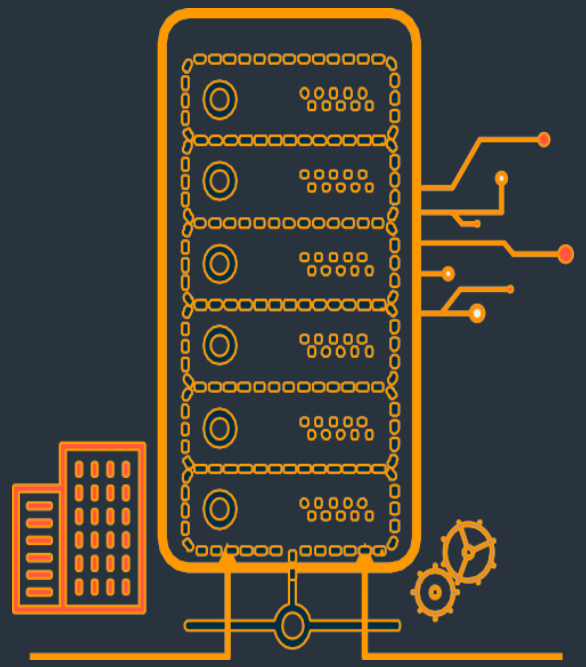


Challenges still existed

- Trade CAPEX for OPEX
- More scale
- Elastic resources
- Faster speed and agility
- Reduced maintenance
- Better availability and fault tolerance

- Still need to administer virtual servers
- Still need to manage capacity and utilization
- Still need to size workloads
- Still need to manage availability, fault tolerance
- Still expensive to run intermittent jobs

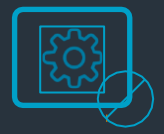
Virtual Servers in Datacenters



What is Serverless?

Build and run applications without thinking about Servers

What is serverless?



No infrastructure provisioning, no management

Automatic scaling

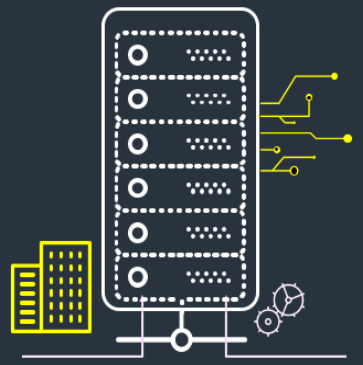
Pay for value

Highly available and secure

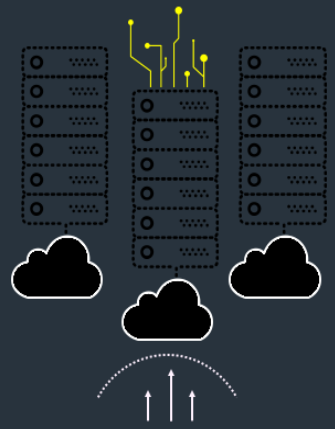
Evolving to Serverless

Physical Servers in Datacenters

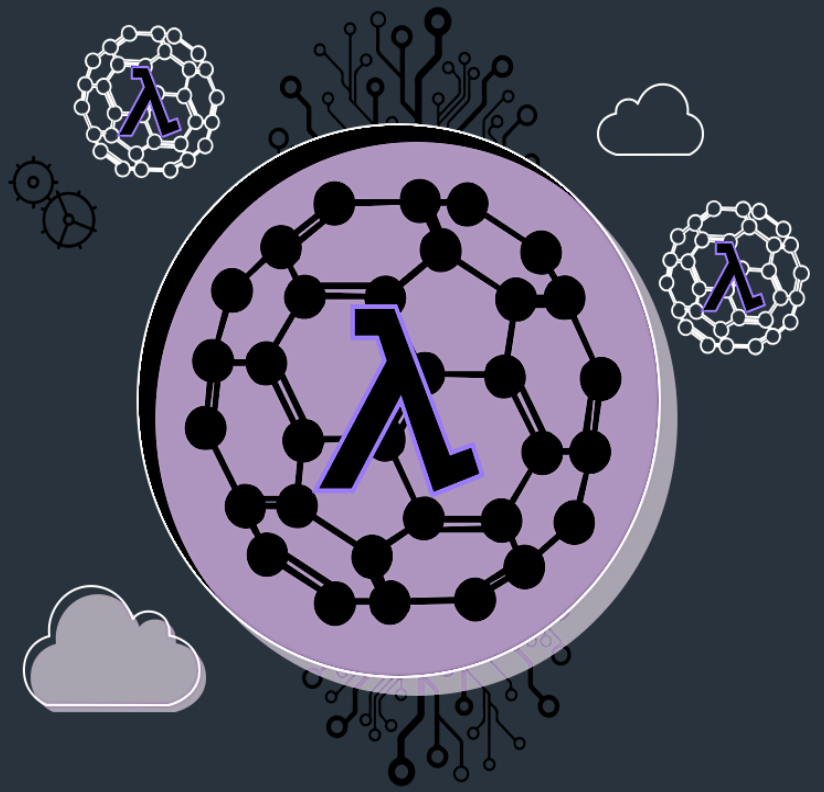
Virtual Servers in Datacenters



Virtual Servers in the Cloud



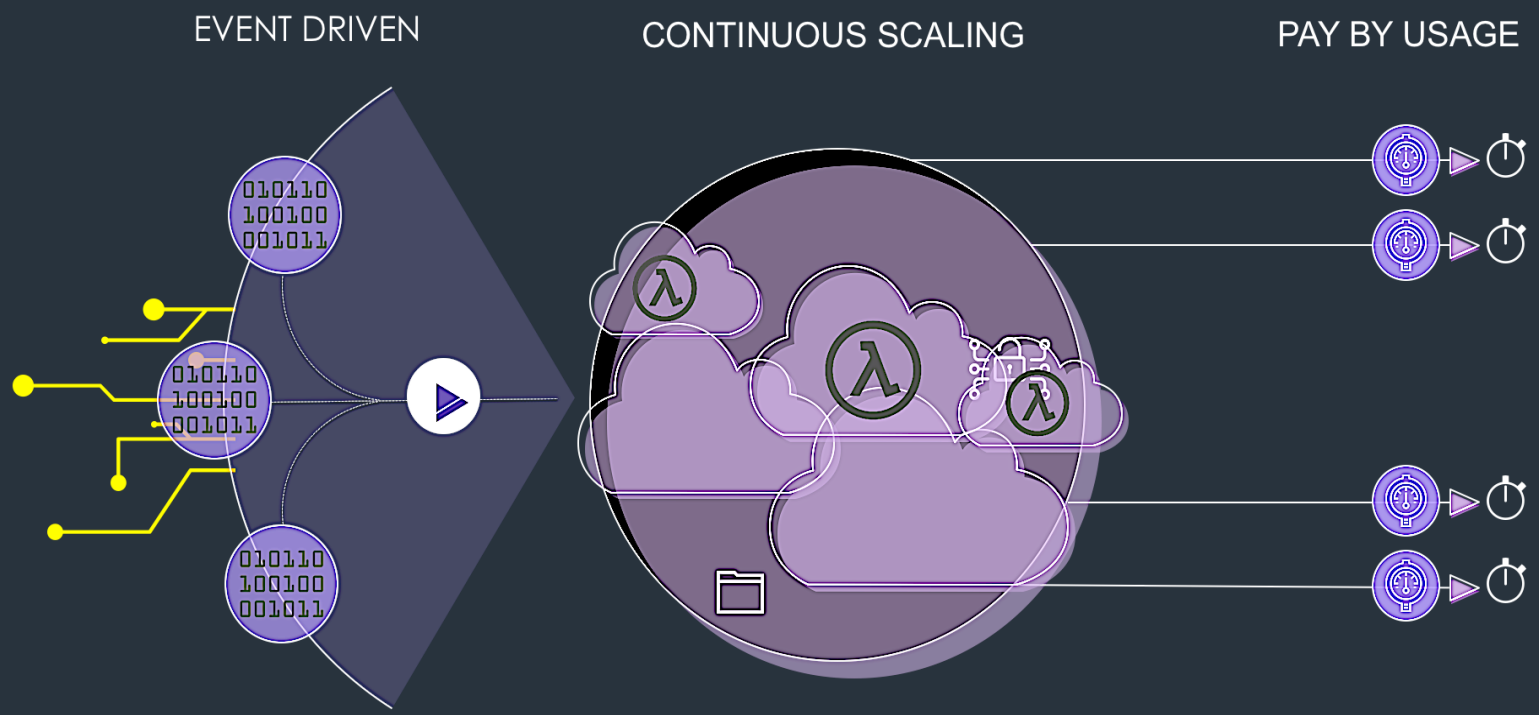
SERVERLESS



No servers is easier to manage than virtual servers

- *Provisioning and Utilization*
- *Availability and fault tolerance*
- *Scaling*
- *Operations and Management*

Deliver on demand, never pay for idle

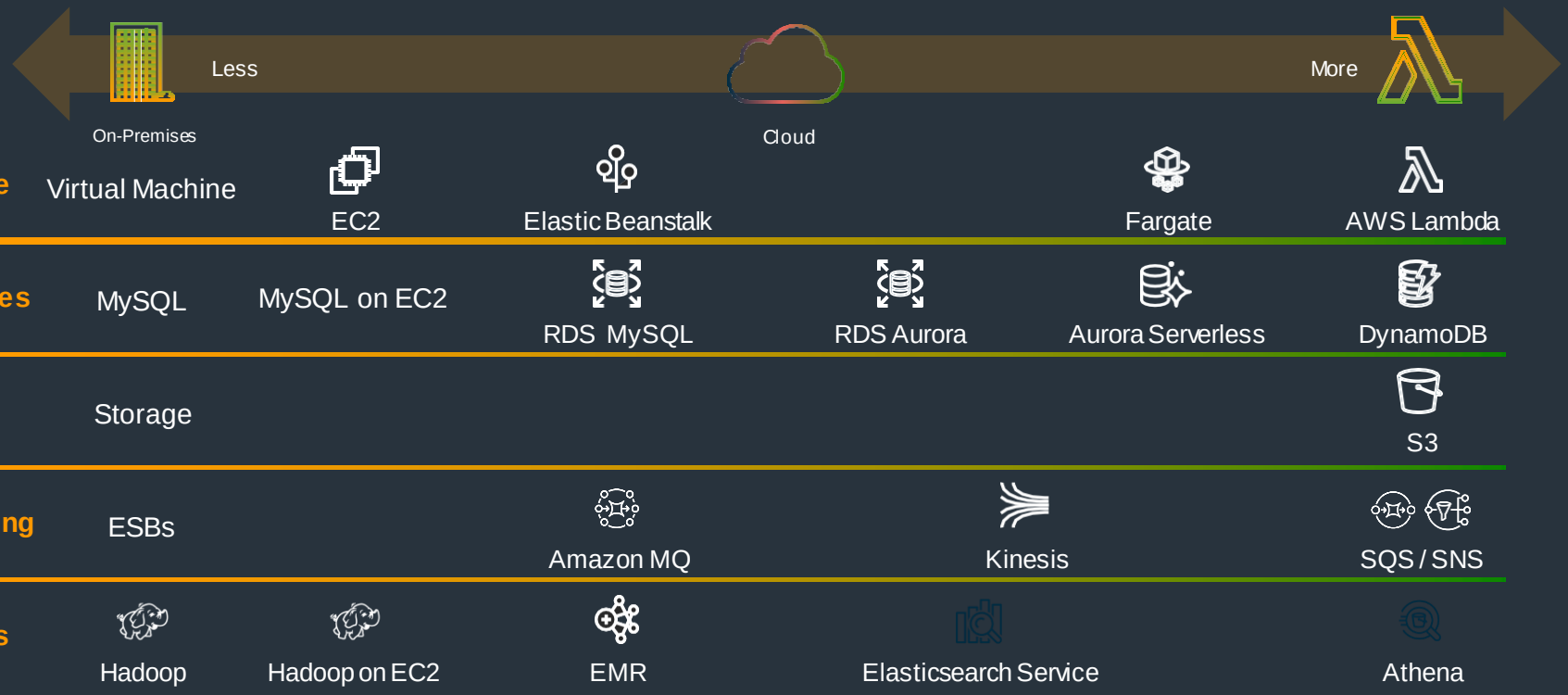


When the impact of change is small, release velocity can increase

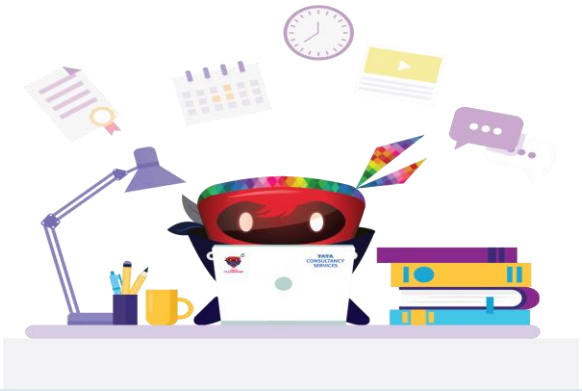
Monolith
Does everything

Microservices
Does one thing

AWS operational responsibility models



Module # 2



Serverless

API GATEWAY

What is API GATEWAY?

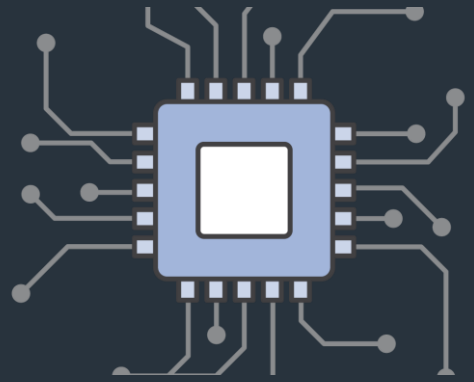
- Amazon API Gateway is a fully managed service that makes it easy for developers to publish, maintain, monitor, and secure APIs at any scale.

- Create an API that acts as a “front door” for applications to access data, business logic, or functionality from your back-end services, such as applications running on:

  - Amazon EC2, Amazon ECS or AWS Elastic Beanstalk, code running on AWS Lambda, or any web application.

- Amazon API Gateway handles all of the tasks involved in accepting and processing up to hundreds of thousands of concurrent API calls, including traffic management, authorization and access control, monitoring, and API version management

Introduction to Amazon API Gateway

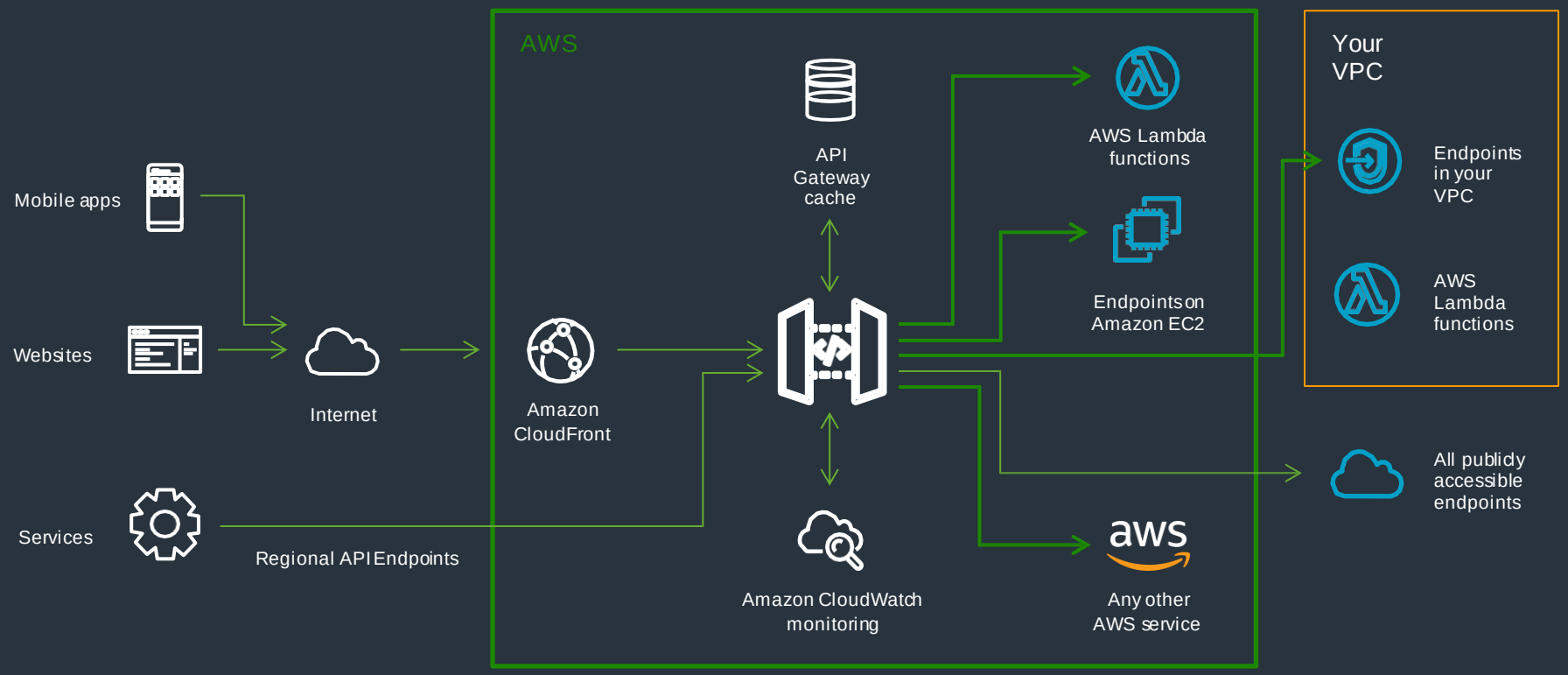


Create a unified API frontend for multiple micro-services

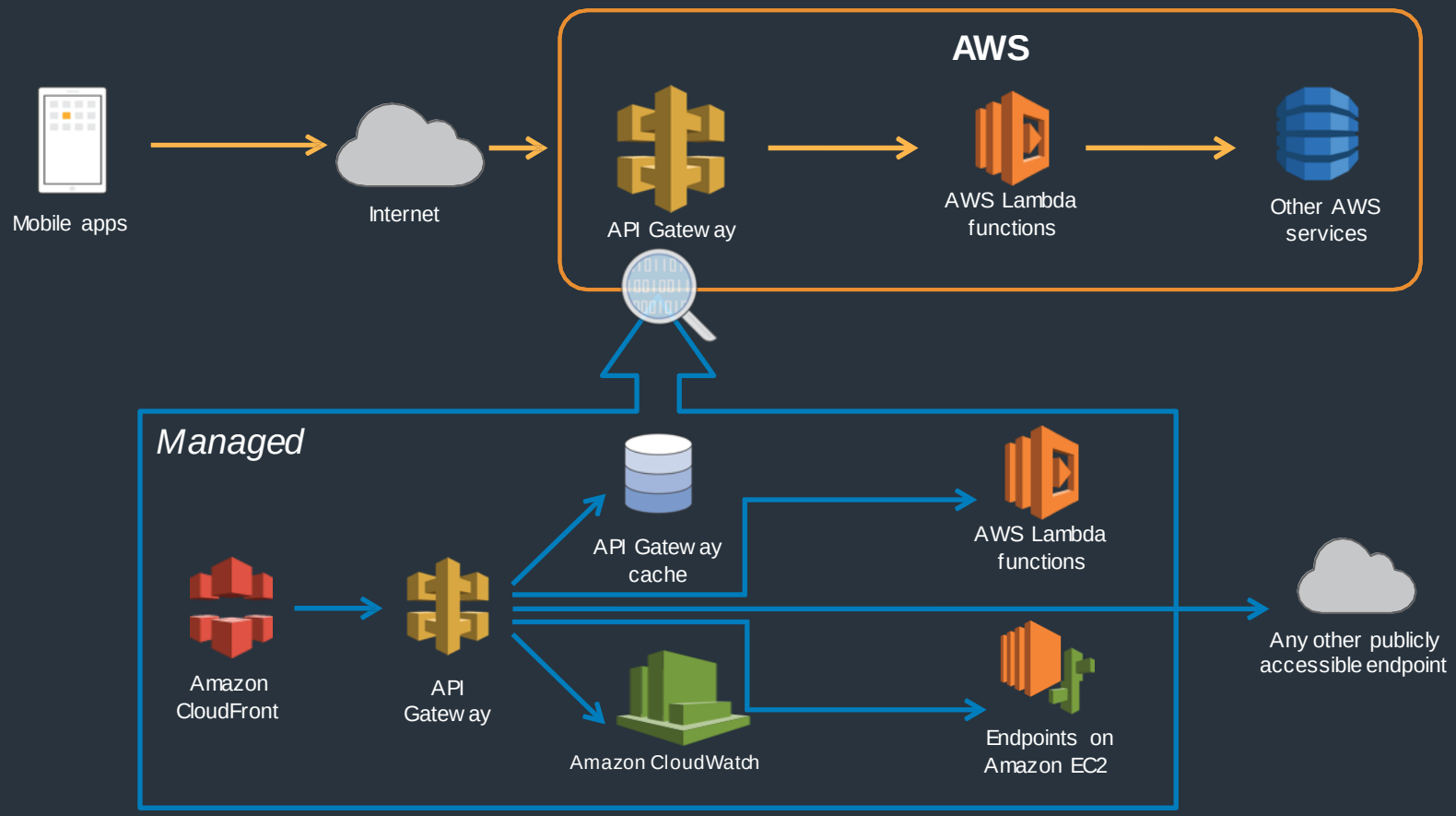
DDoS protection and throttling for your backend

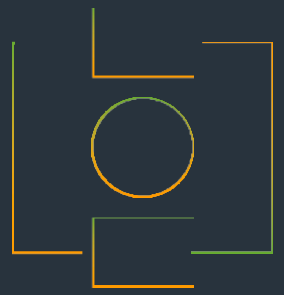
Authenticate and authorize requests to a backend

Manage APIs with API Gateway



The serverless stack – zoom in

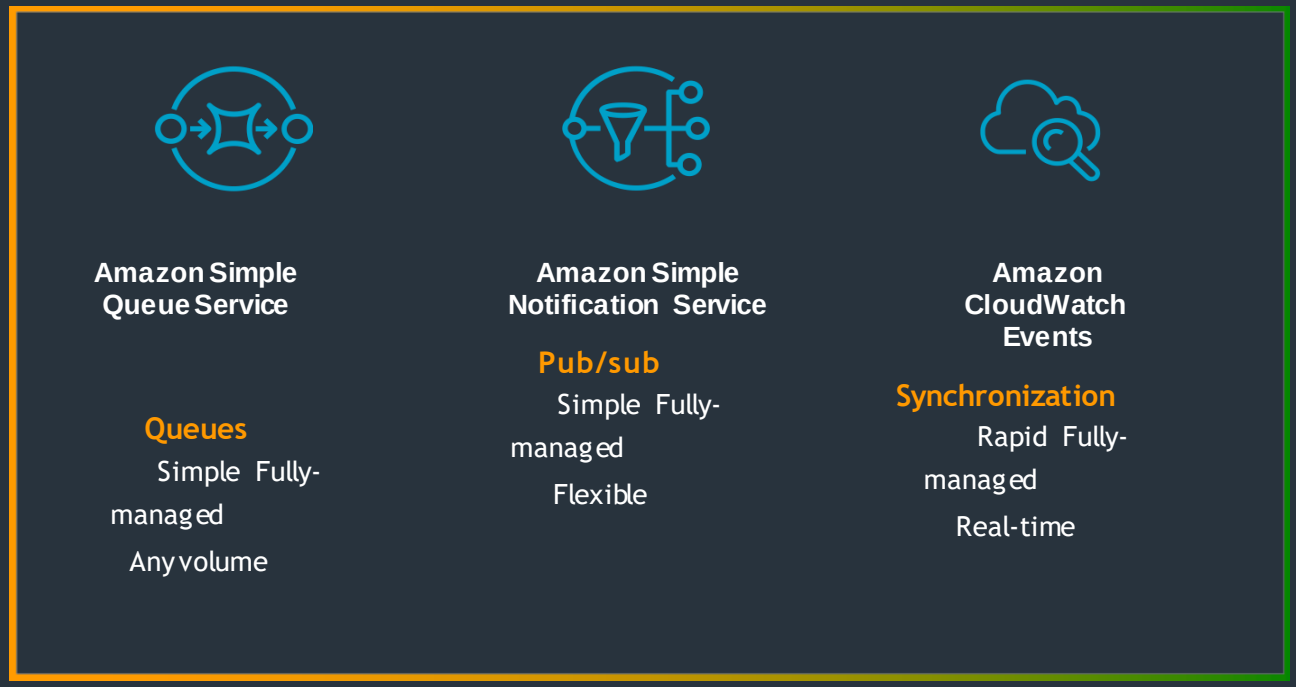




Event-driven architectures

Decouple state from code using messaging

Messaging



Amazon Simple Queue Service

Queues

Simple Fully-managed

Any volume

Amazon Simple Notification Service

Pub/sub

Simple Fully-managed

Flexible

Amazon CloudWatch Events

Synchronization

Rapid Fully-managed

Real-time

And data streams

Data Stream Capture

Amazon Kinesis Data Streams

Ingest Data streams Data processing Real-time

Amazon Dynamo DB

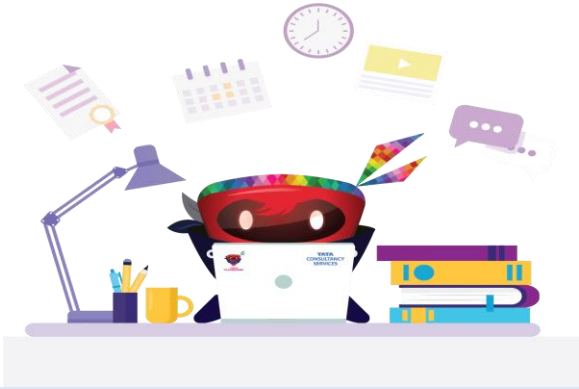
Data Store

Microservices

Performance at scale

Fast and Flexible

Module # 3



Serverless Architecture

AWS Lambda

Why Lambda?

“Productivity focused compute platform to build powerful, dynamic, modular applications in the cloud”

1

No Infrastructure to manage

Focus on business logic, not infrastructure. You upload code; AWS Lambda handles everything else.

2

High performance at any scale; Cost-effective and efficient

Automatically scales your application by running code in response to each trigger. Your code runs in parallel and processes each trigger individually, scaling precisely with the size of the workload.

3

Bring Your Own Code

Run code in a choice of standard languages. Use threads, processes, files and shell scripts normally.

How Lambda works

Invoked in response to events
- Changes in data
- Changes in state

S3 event notifications

DynamoDB Streams

Kinesis events

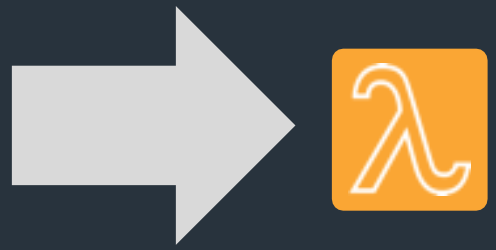
SNS events

CloudTrail events

Cognito events

Custom events

Author in familiar language using any libraries; Execute only when needed, automatic scale



"Lambda functions"

Access any service, including your own

Any custom

Any AWS

Such as...

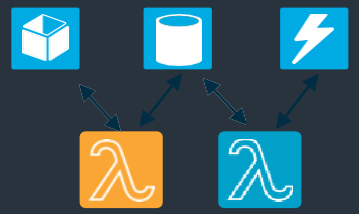
SNS DynamoDBLambda

Redshift

Kinesis

S3

Lambda functions



Simple resource model

- Set memory to any size from 128MB to 1GB, in 64MB steps
- Receive an equivalent portion of other resources (disk, network, compute power, etc.)
- Lambda tells you how much memory you used, so you can tune this setting.

Flexible invocation paths

- Lambda functions can be invoked “on demand” through CLI and Console
- Subscribe to one or many event sources
- Reuse the same Lambda function with multiple event sources

Granular permissions control (using IAM)

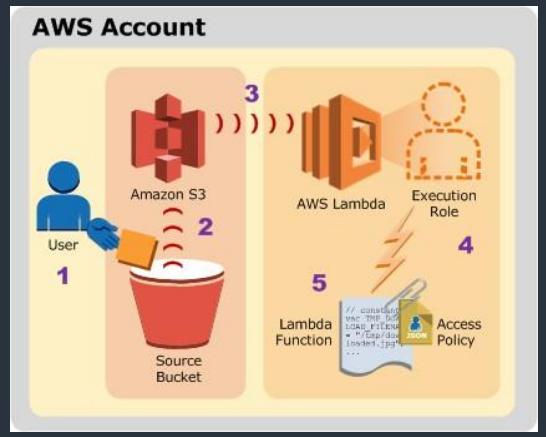
- Define what permissions the function has
- Uses IAM role (execution role) for granular permission control

Recommended minimum permission – log to CloudWatch
E.g. “read from <X> DDB table only in the context of <Y> function”

Under the covers - Invocation models

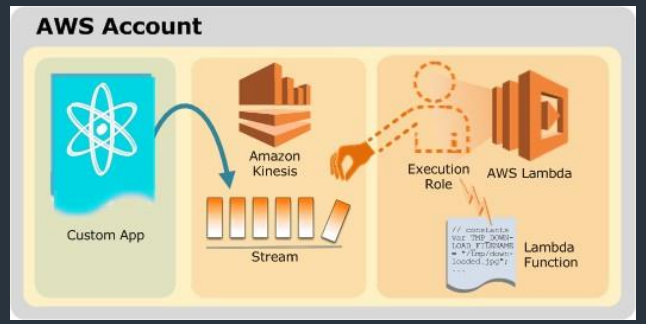
Push model - Invoke API call

- “RequestResponse” mode returns a response immediately
- “Event” mode puts function call into a queue, picked up by a poller and then invoked; Used by S3
- Caller (e.g. S3) derives permission through resource policies on Lambda function



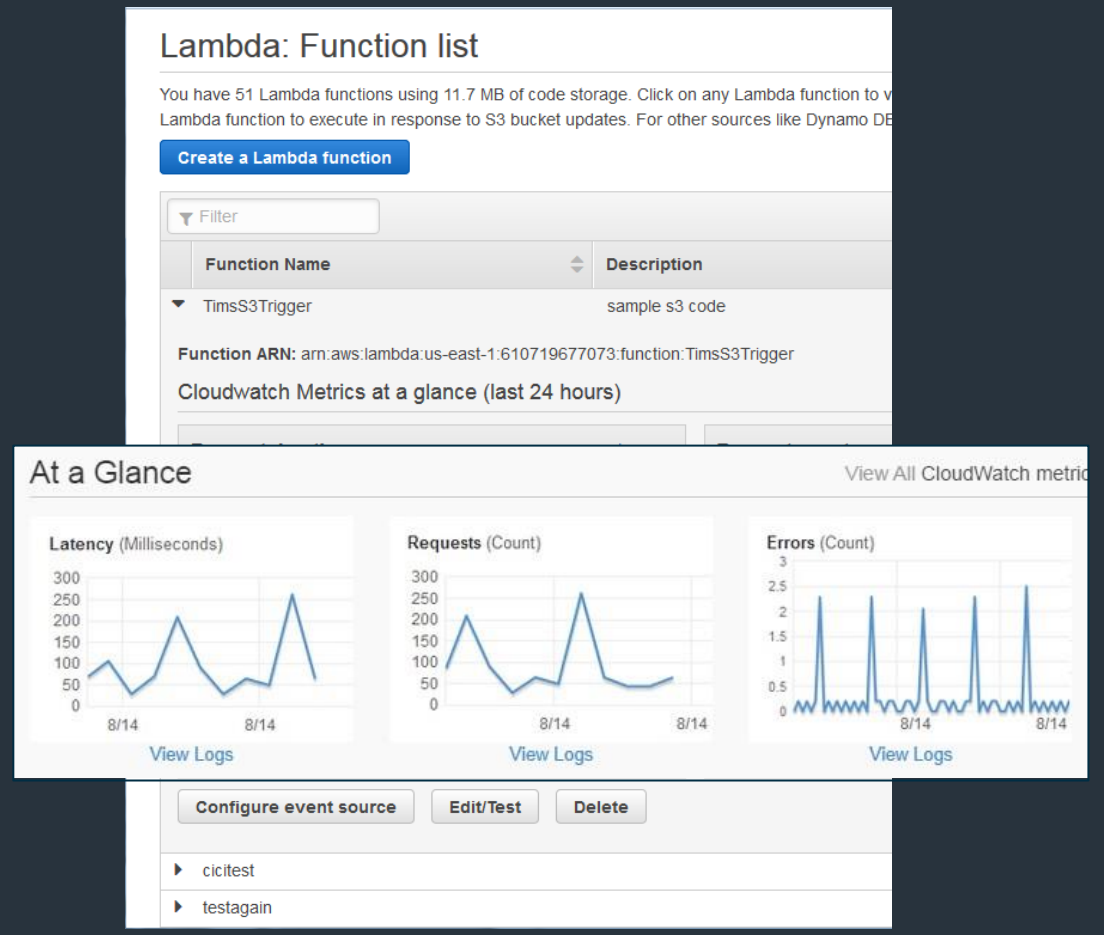
Pull model - Subscribe to event source

- points poller at a particular data stream; Used by DDB/Kinesis
- Poller derived permission from execution role to read from particular DDB update stream

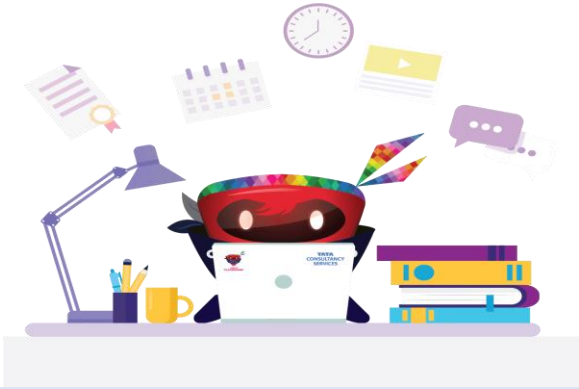


Monitoring and debugging Lambda Functions

- AWS Lambda console includes a dashboard for functions
  - Lists all Lambda functions
  - Easy editing of resources, event sources and other settings
  - At-a-glance metrics
- Metrics automatically reported to Amazon CloudWatch for each Lambda function
  - Requests
  - Errors
  - Latency
  - Throttles
- Logs captured by Amazon CloudWatch Logging service



Module # 4



Serverless Architecture

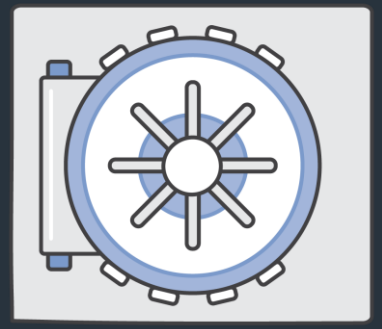
Amazon Cognito

What do customers want to do?

Focus on differentiating features in their applications

Provide flexible and modern options for authenticating

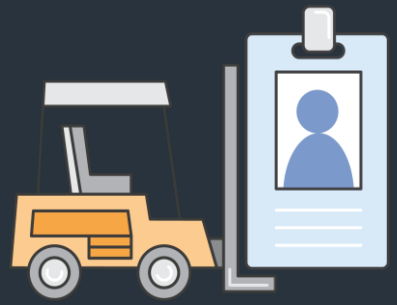
Build standards-based, API-driven interoperable platforms



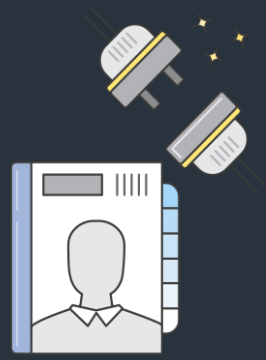
Protect the security and privacy of their customers

Introducing Amazon Cognito

Simple and secure user sign-up, sign-in, and access control for web and mobile apps.



Offload undifferentiated identity heavy lifting

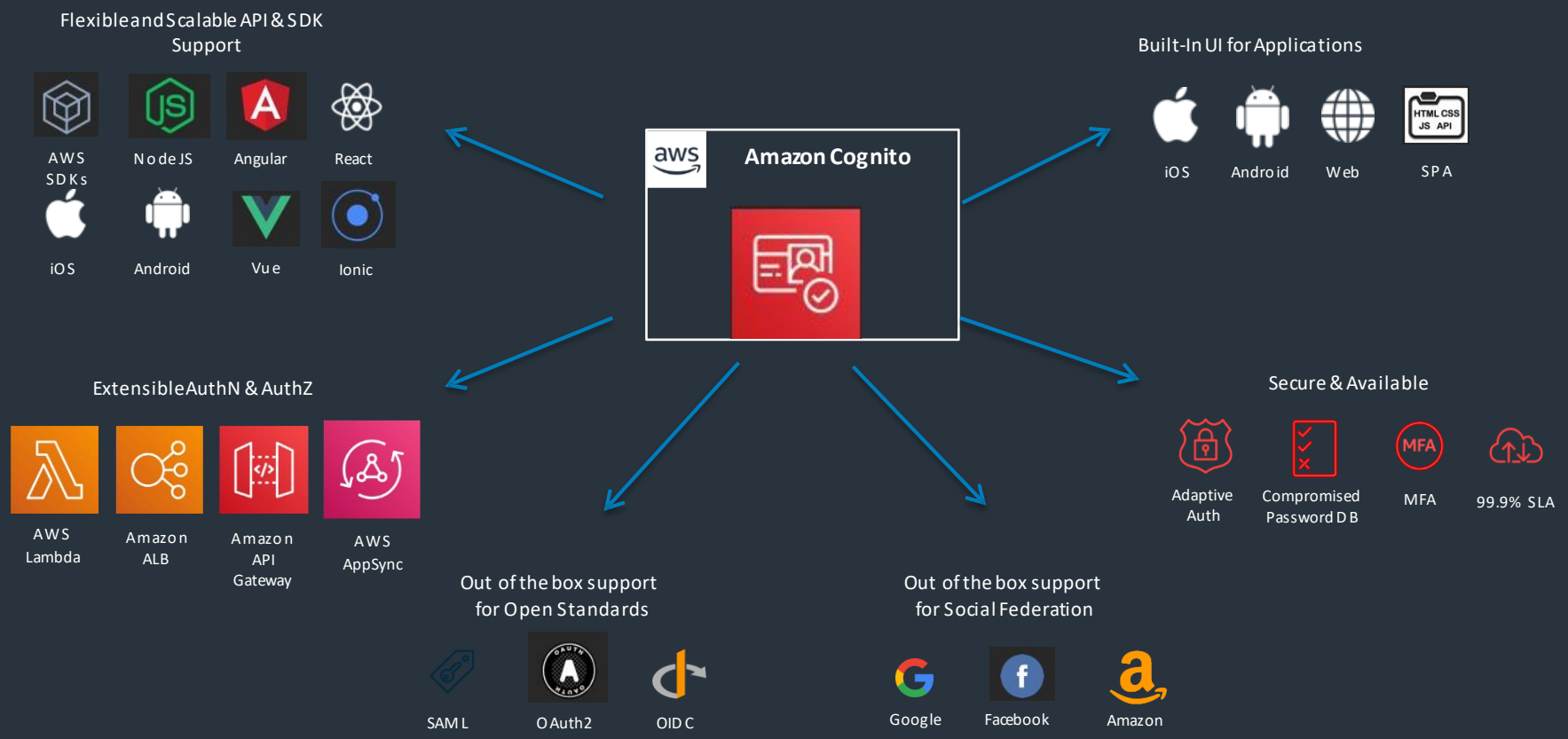


Use your choice of existing or cloud native identities.

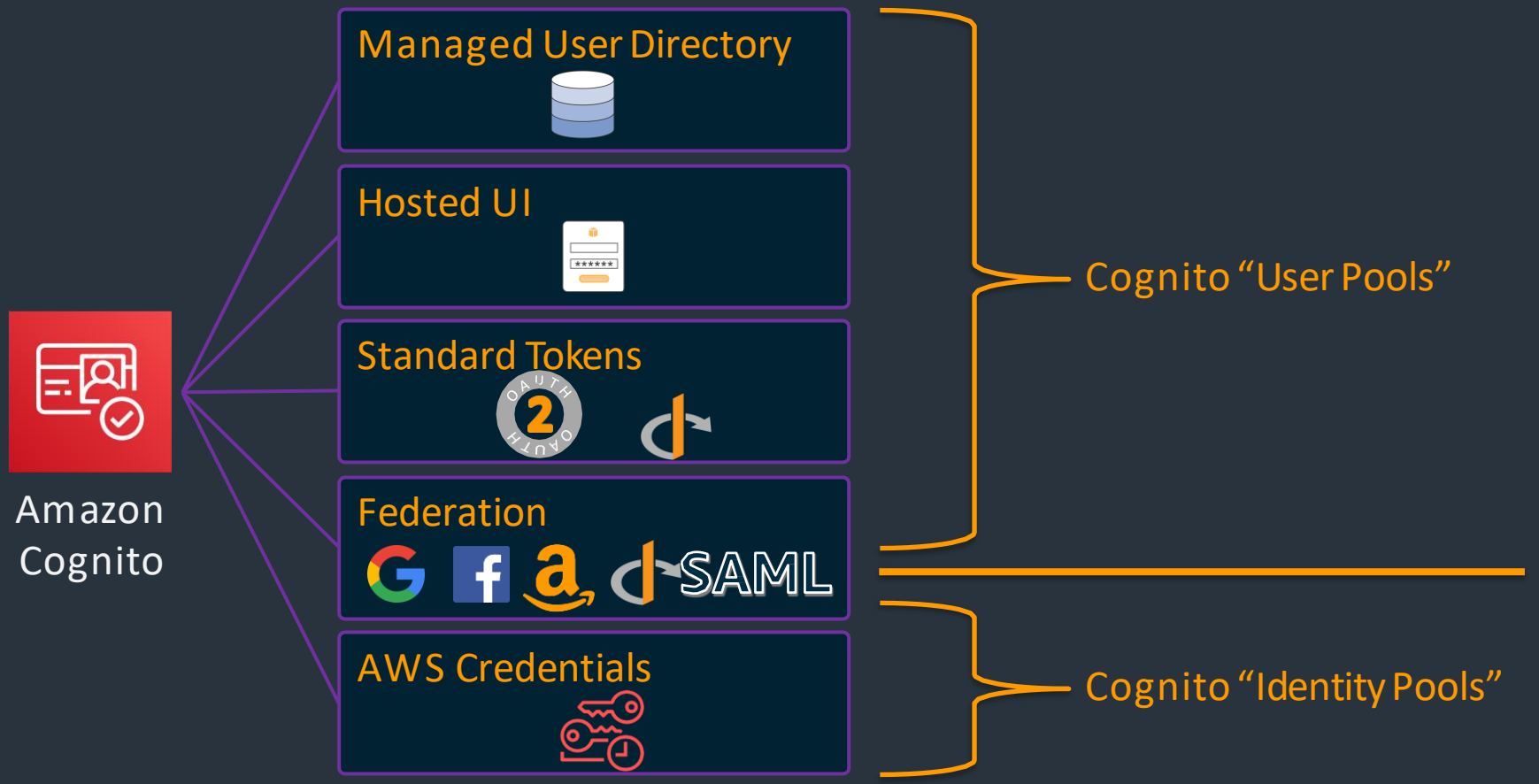
Use standards-based authentication

Provide advanced security for your apps and users

Cognito: Flexible and Fully Managed Application Identity



Amazon Cognito Overview

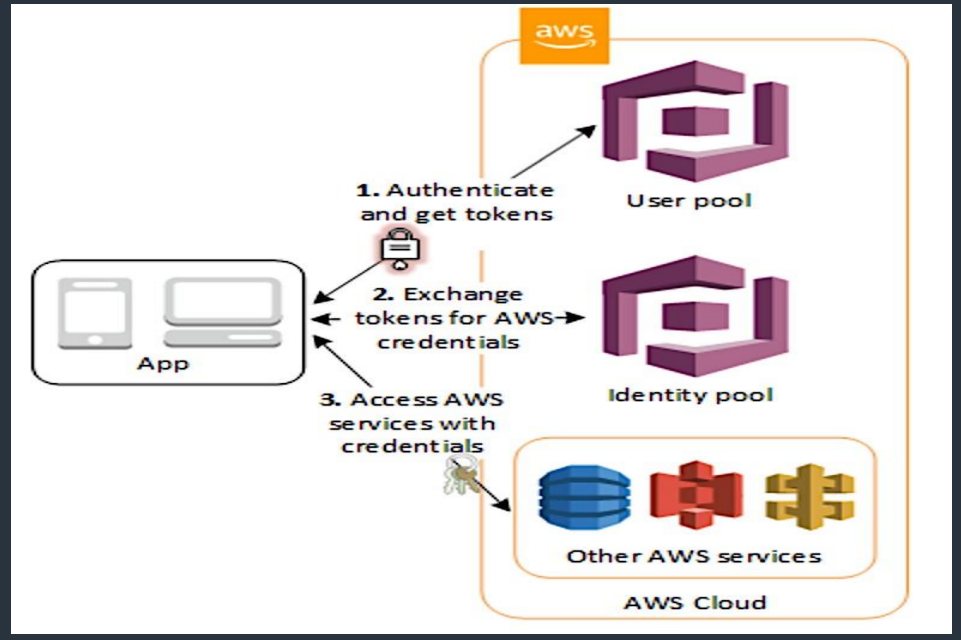


User pools and Identity pools

Your users can sign in directly with a user name and password, or through a third party such as Facebook, Amazon, or Google.

The two main components of Amazon Cognito are user pools and identity pools.

- User pools are user directories that provide sign-up and sign-in options for your app users. Identity pools enable you to grant your users access to other AWS services.
- You can use identity pools and user pools separately or together.



Amazon Cognito User Pools

1

Easy User Management

Add sign-up and sign-in easily to your mobile and web apps

2

Managed User Directory

Launch a simple, secure, low-cost, and fully managed service to create and maintain a user directory that scales to 100s of millions of users

3

Enhanced Security Features

Verify phone numbers and email addresses and offer multi-factor authentication

Managed user directory

- Serverless directory
  - Nothing to manage
  - API driven
  - Multi-AZ redundancy
- User & group storage
  - Profile information (name, email, etc)
  - Credential & device information (SRP verifier, MFA, etc)
  - Extensible with custom attributes

Managed User Directory (User Pool)

Login

- Usernames / Passwords
- Profiles (attributes / groups)

Admin controls

- Create user, import, search, disable, set policies, etc.

User Flows

- Sign up, Confirm, Sign in, Forgot password, etc.

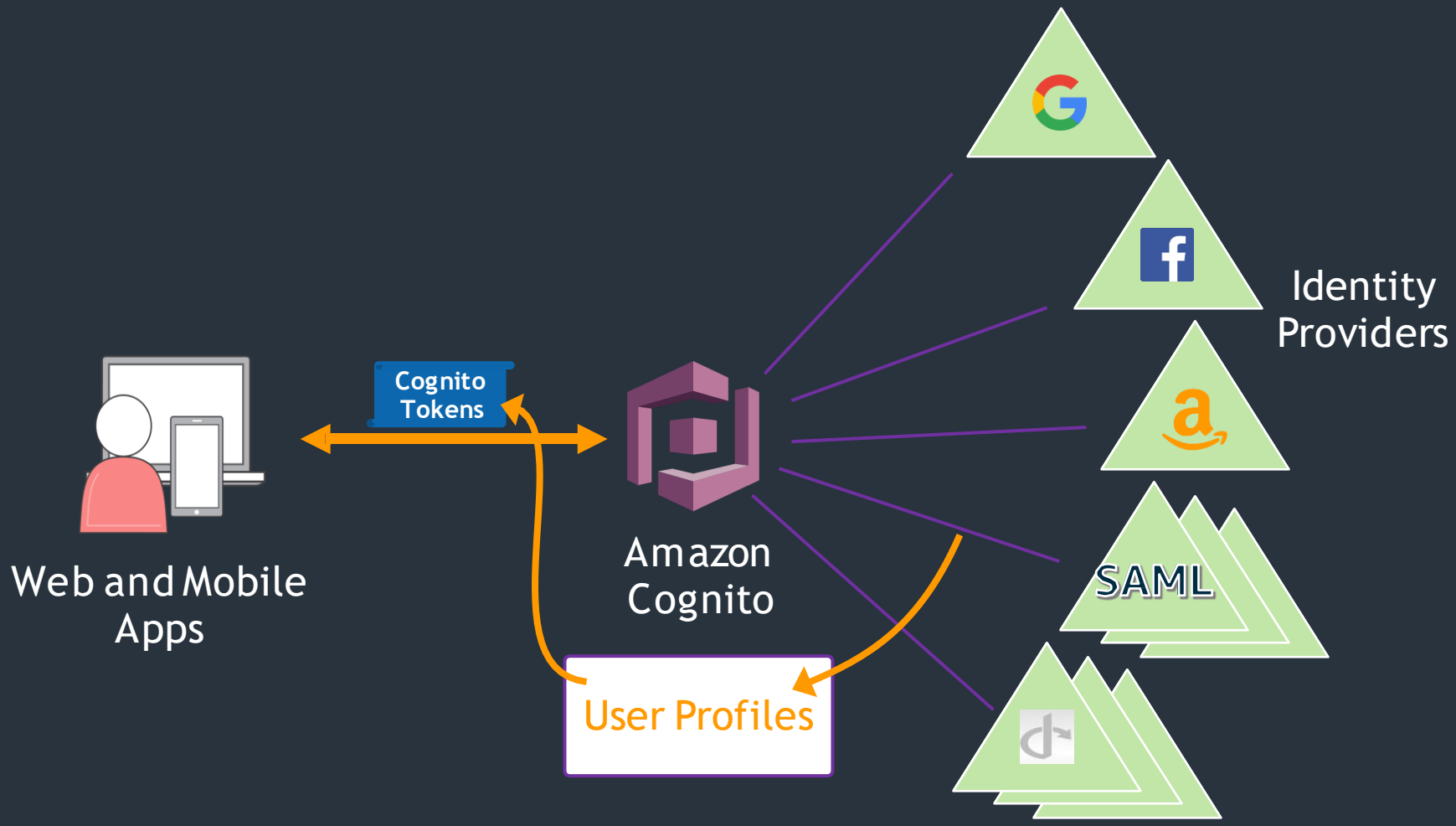
Security

- MFA, Adaptive authentication, Protection for compromised credentials

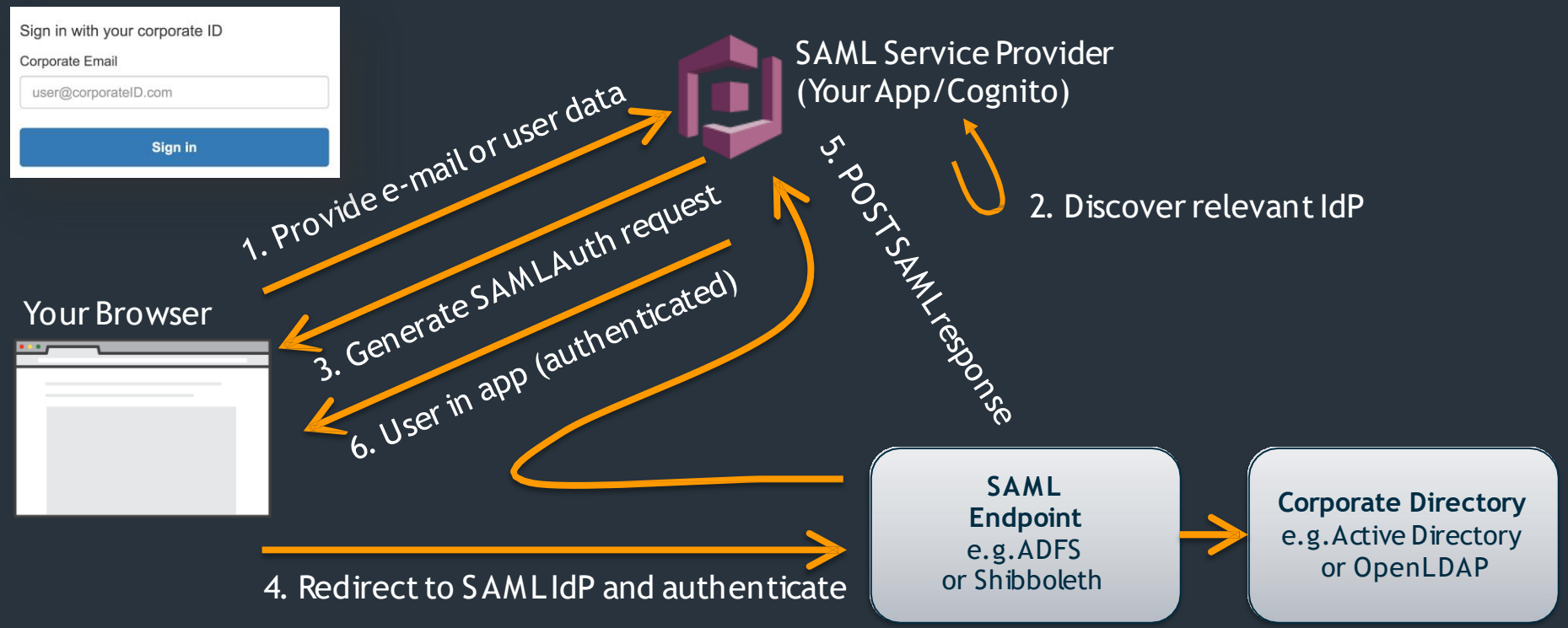
Compliance

- PCI, HIPAA, SOC 1/2/3, ISO 270001, GDPR

Identity Federation



Cognito manages federation process for you

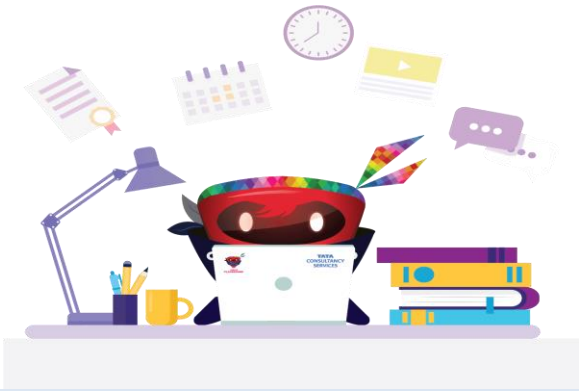


Supports just-in-time user provisioning

Comprehensive user flows

User Sign-Up and Sign-In	Allow users to sign up and sign in using an email, phone number, or username (and password) for your application.
User Profile Data	Enable users to view and update their profile data – including custom attributes
Forgot Password	Provide users the ability to change their password when they forget it with a one-time password challenge
Token Based Authentication	Use JSON Web Tokens (JWTs) based on OpenID Connect (OIDC) and OAuth 2.0 standards for user authentication in your backend
Email or Phone Number Verification	Require users to verify their email address or phone number prior to activating their account with a one-time password challenge
SMS Multifactor Authentication	Require users to complete a second factor of authentication by inputting a security code received via SMS as part of the sign-in flow

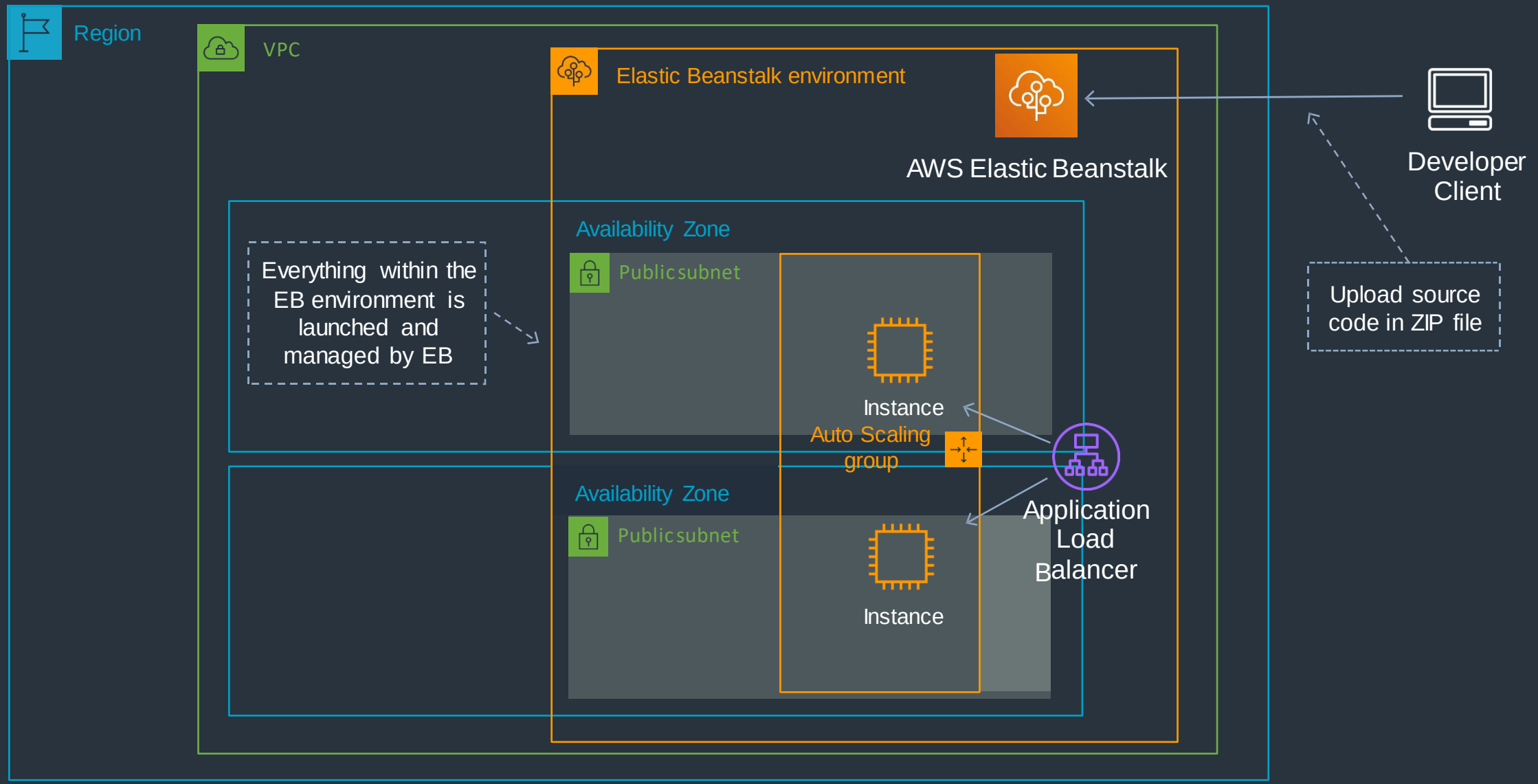
Module # 5



Platform Services

AWS Elastic Beanstalk

AWS Elastic Beanstalk



AWS Elastic Beanstalk

- AWS Elastic Beanstalk can be used to quickly deploy and manage applications in the AWS Cloud.
- Developers upload applications and Elastic Beanstalk handles the deployment details of capacity provisioning, load balancing, auto-scaling, and application health monitoring.
- AWS Elastic Beanstalk leverages Elastic Load Balancing and Auto Scaling to automatically scale your application in and out based on your application's specific needs.
- Considered a Platform as a Service (PaaS) solution.

AWS Elastic Beanstalk

- Supports Java, .NET, PHP, Node.js, Python, Ruby, Go, and Docker web applications.
- Supports the following languages and development stacks:
  - Apache Tomcat for Java applications.
  - Apache HTTP Server for PHP applications.
  - Apache HTTP Server for Python applications.
  - Nginx or Apache HTTP Server for Node.js applications.
  - Passenger or Puma for Ruby applications.
  - Microsoft IIS 7.5, 8.0, and 8.5 for .NET applications.
  - Java SE.
  - Docker.
  - Go.

AWS Elastic Beanstalk

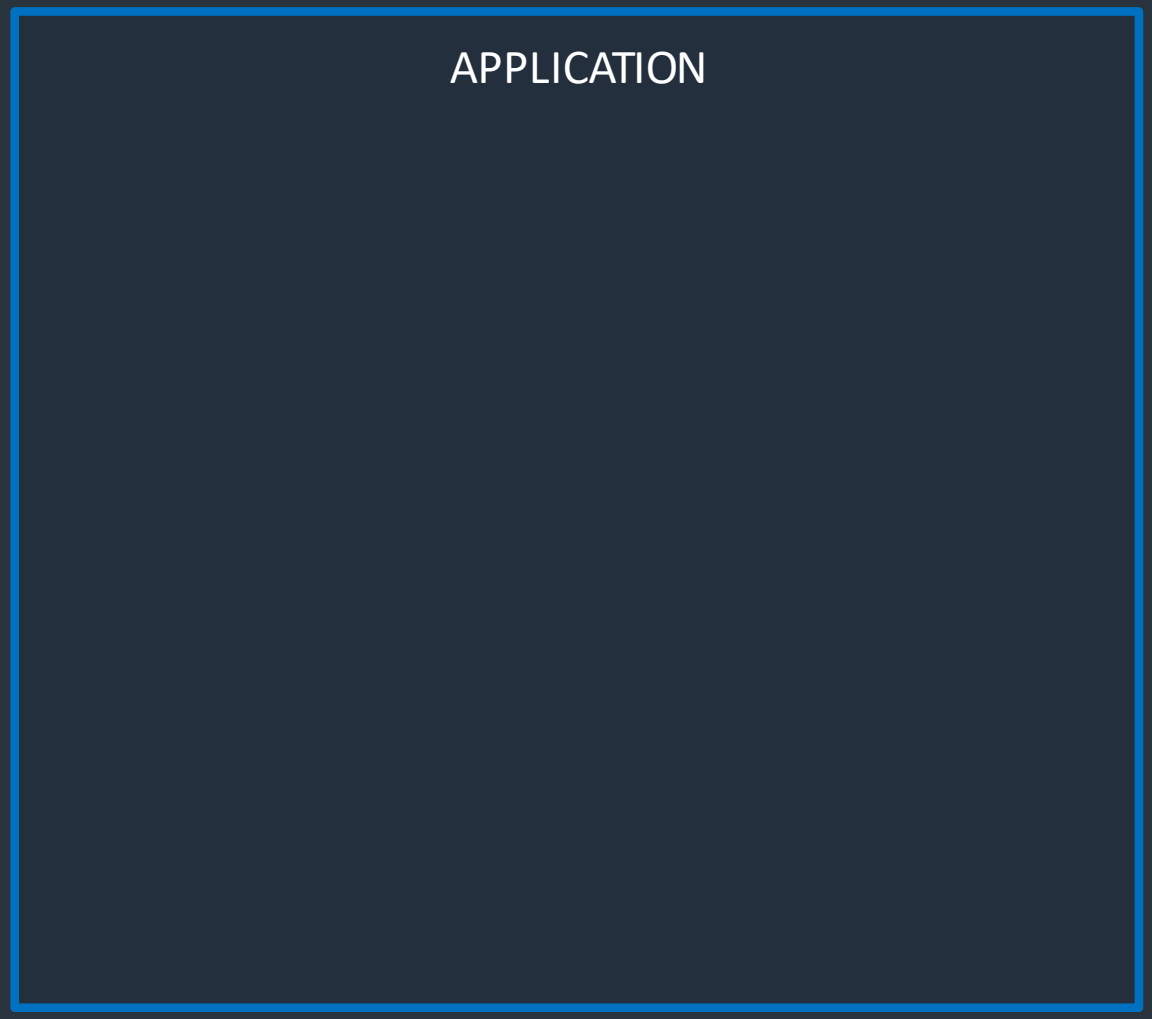
- You maintain full control of the underlying resources.
- You pay only for the resources provisioned, not for Elastic Beanstalk itself.
- Elastic Beanstalk automatically scales your application up and down.
- The Managed Platform Updates feature automatically applies updates for your operating system, Java, PHP, Node.js etc.
- Elastic Beanstalk monitors and manages application health and information is viewable via a dashboard.
- AWS CloudFormation is used by Elastic Beanstalk to deploy the resources.
- Integrated with CloudWatch and X-Ray for performance data and metrics.

AWS Elastic Beanstalk

There are several layers:

Applications:

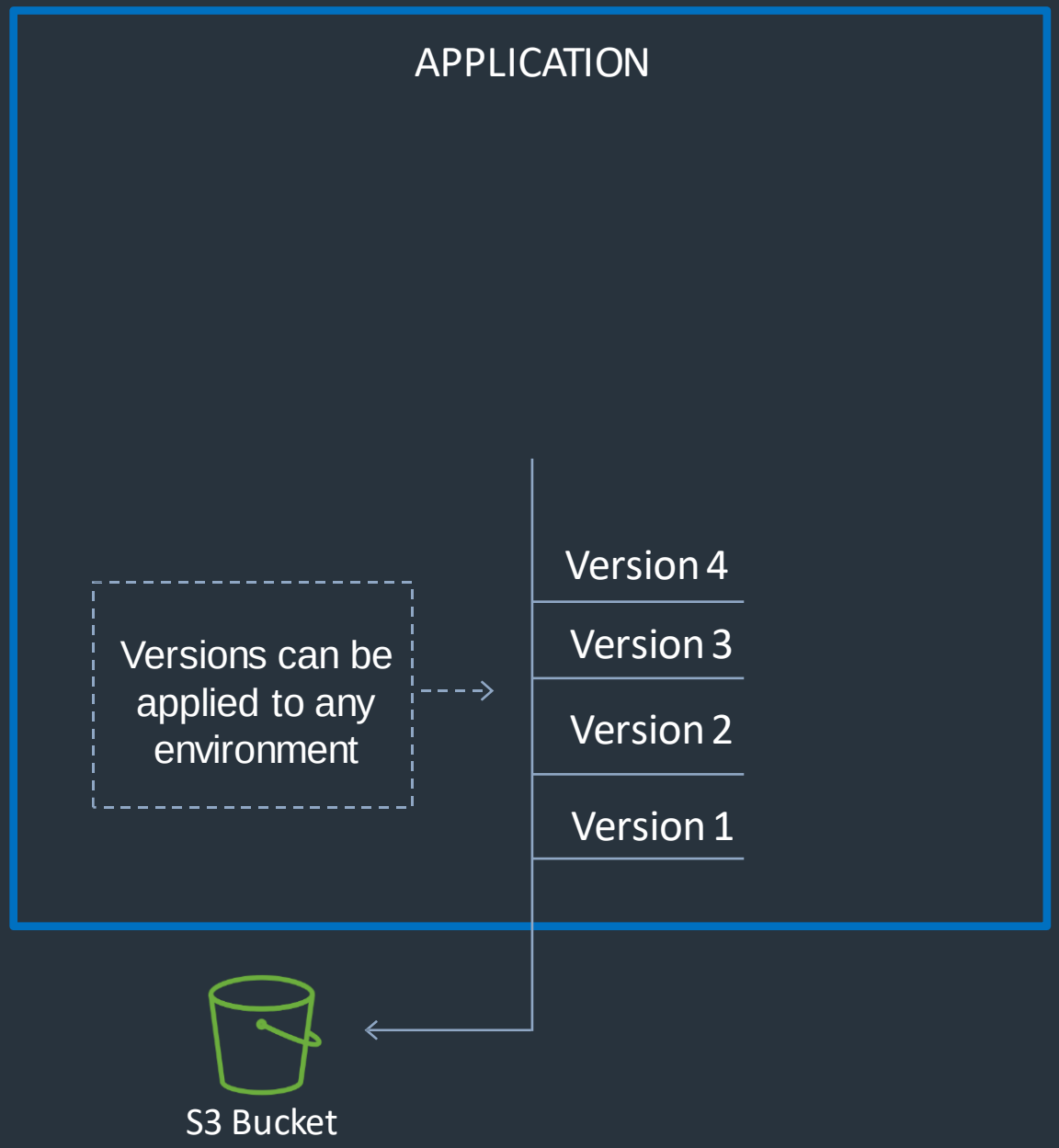
- Contain environments, environment configurations, and application versions.
- You can have multiple application versions held within an application.



AWS Elastic Beanstalk

Application version

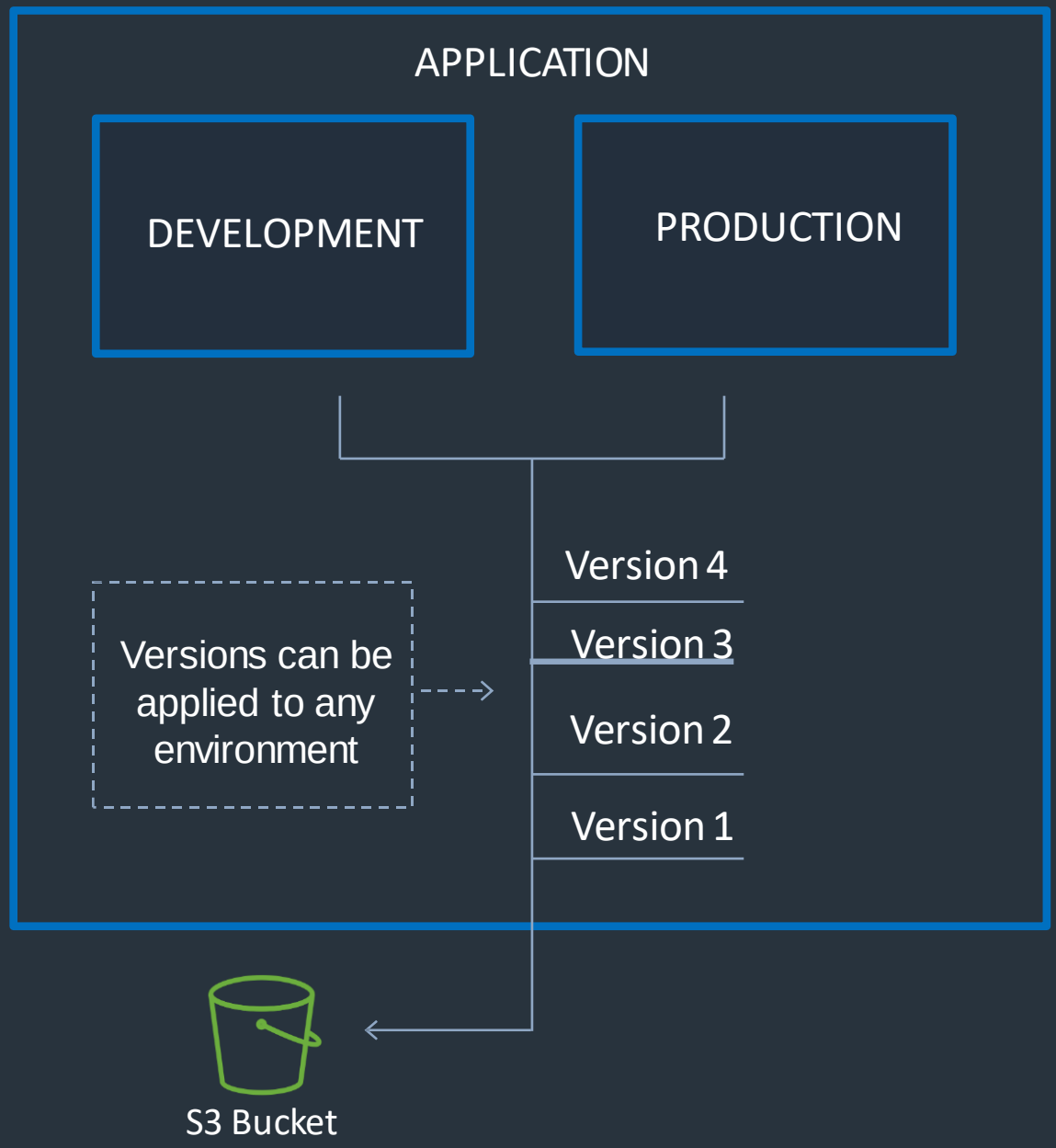
- A specific reference to a section of deployable code.
- The application version will point typically to an Amazon S3 bucket containing the code.



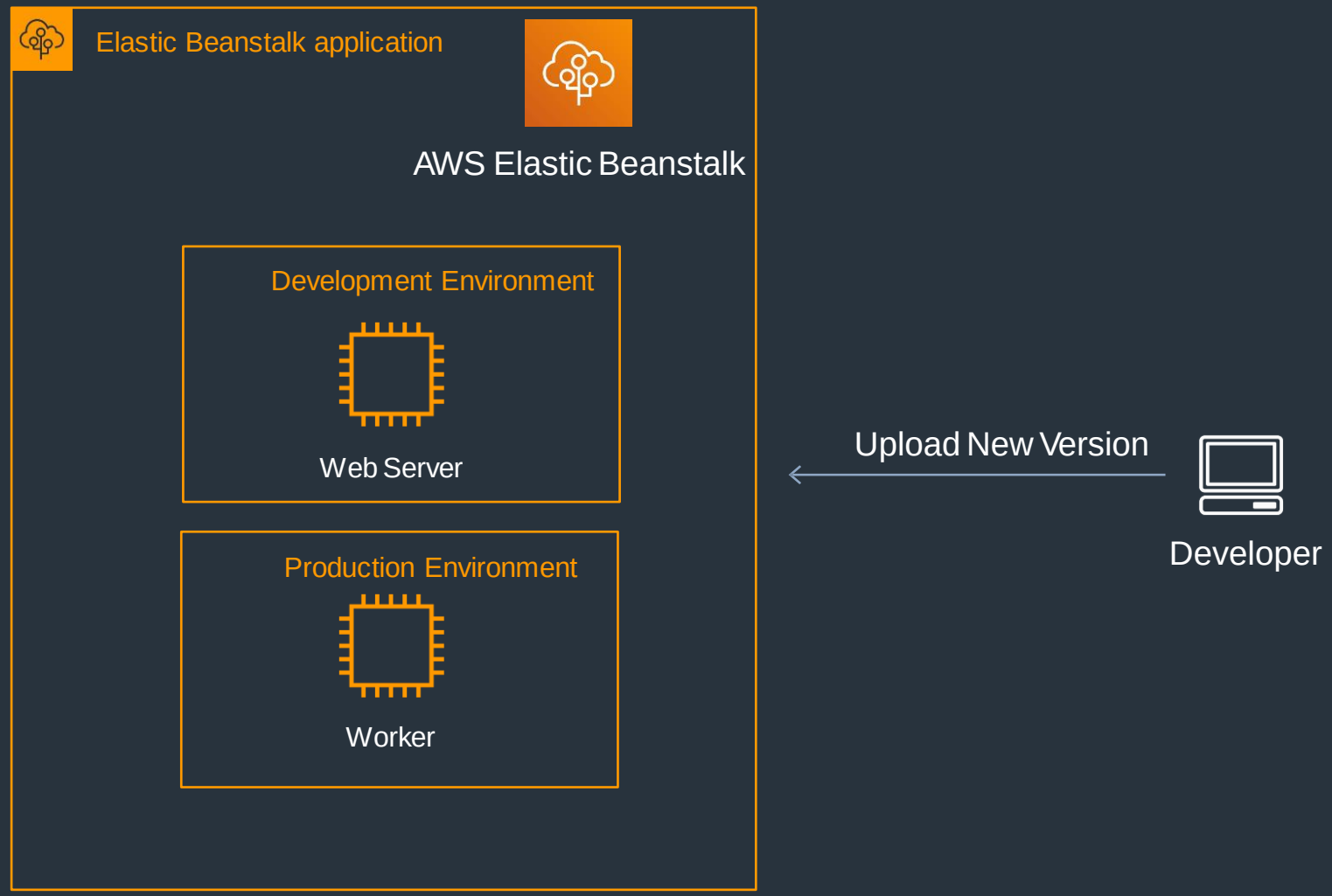
AWS Elastic Beanstalk

Environments:

- An application version that has been deployed on AWS resources.
- The resources are configured and provisioned by AWS Elastic Beanstalk.
- The environment is comprised of all the resources created by Elastic Beanstalk and not just an EC2 instance with your uploaded code.



AWS Elastic Beanstalk - Multiple Environments

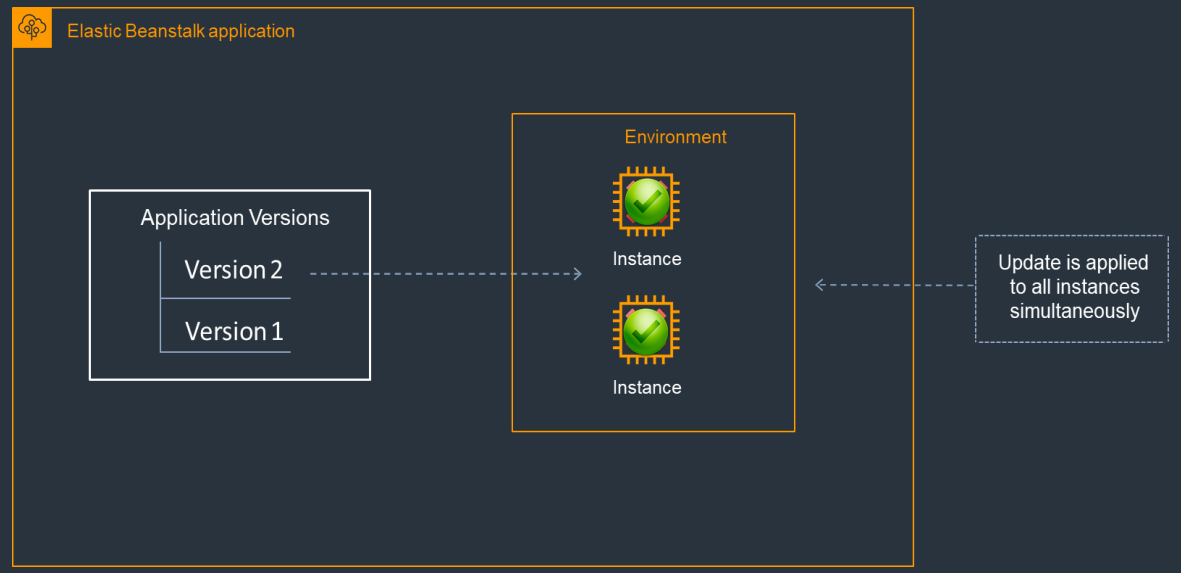


AWS Elastic Beanstalk Deployment Policies

- All at once:
  - Deploys the new version to all instances simultaneously.
- Rolling:
  - Update a batch of instances, and then move onto the next batch once the first batch is healthy.
- Rolling with additional batch:
  - Like Rolling but launches new instances in a batch ensuring that there is full availability.
- Immutable:
  - Launches new instances in a new ASG and deploys the version update to these instances before swapping traffic to these instances once healthy.
- Blue/green:
  - Create a new "stage" environment and deploy updates there.

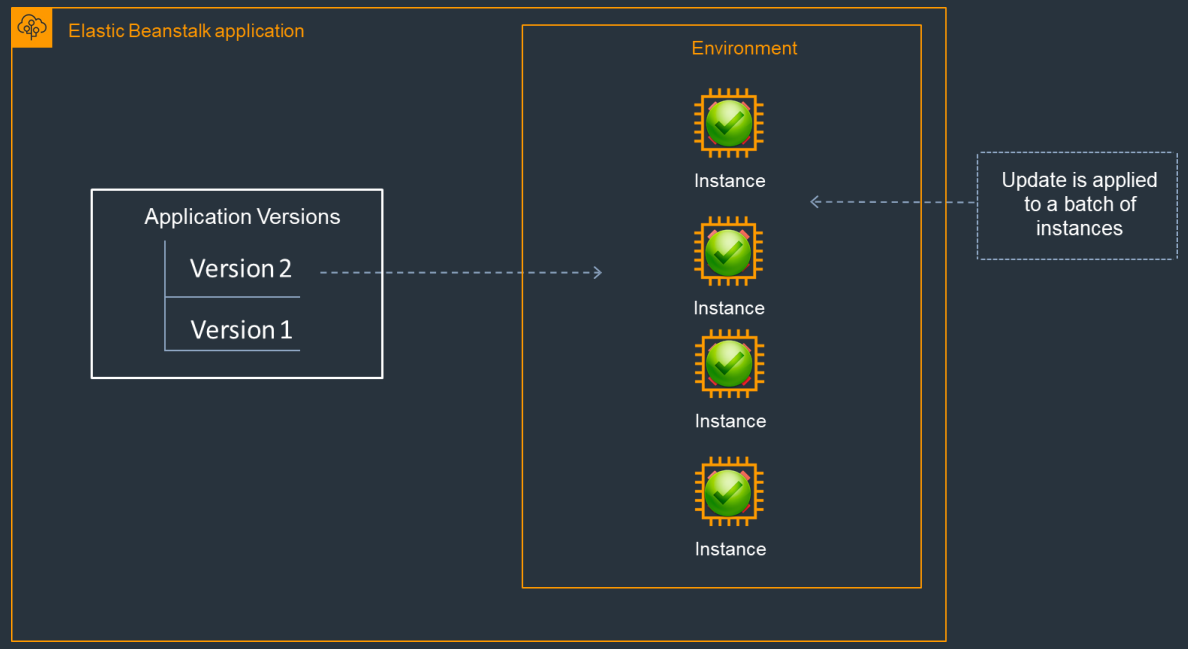
AWS Elastic Beanstalk – All at Once Update

- Deploys the new version to all instances simultaneously.
- All of your instances are out of service while the deployment takes place.
- Fastest deployment.
- Good for quick iterations in development environment.
- You will experience an outage while the deployment is taking place - not ideal for mission-critical systems.
- If the update fails, you need to roll back the changes by re-deploying the original version to all of your instances.
- No additional cost.



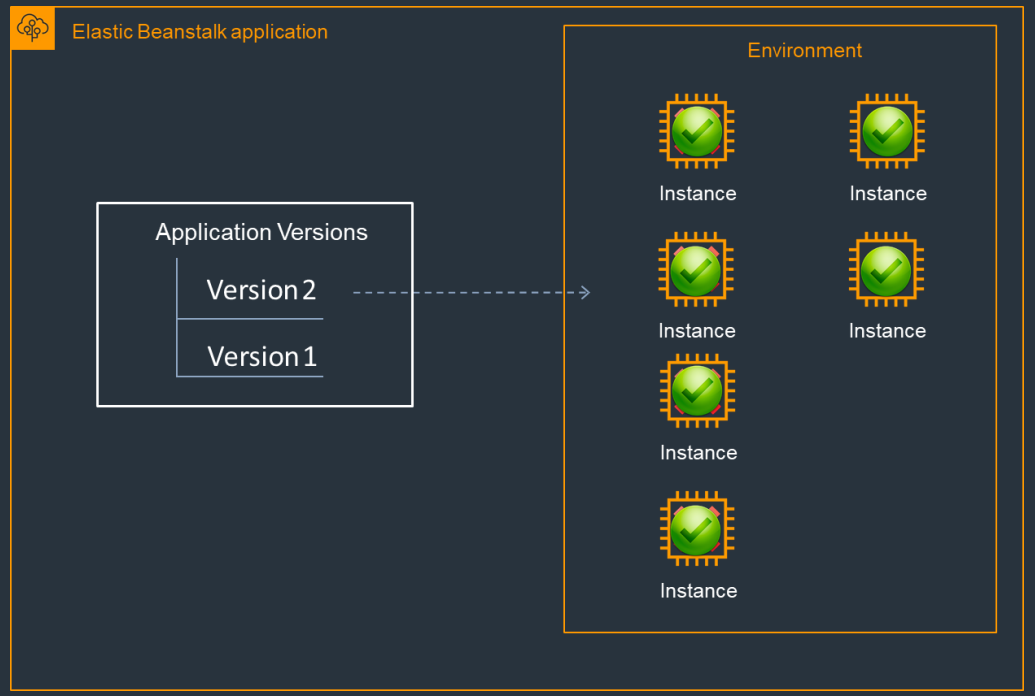
AWS Elastic Beanstalk – Rolling Update

- Update a few instances at a time (batch), and then move onto the next batch once the first batch is healthy (downtime for 1 batch at a time).
- Application is running both versions simultaneously.
- Each batch of instances is taken out of service while the deployment takes place.
- Your environment capacity will be reduced by the number of instances in a batch while the deployment takes place.
- Not ideal for performance-sensitive systems.
- If the update fails, you need to perform an additional rolling update to roll back the changes.
- No additional cost.
- Long deployment time.



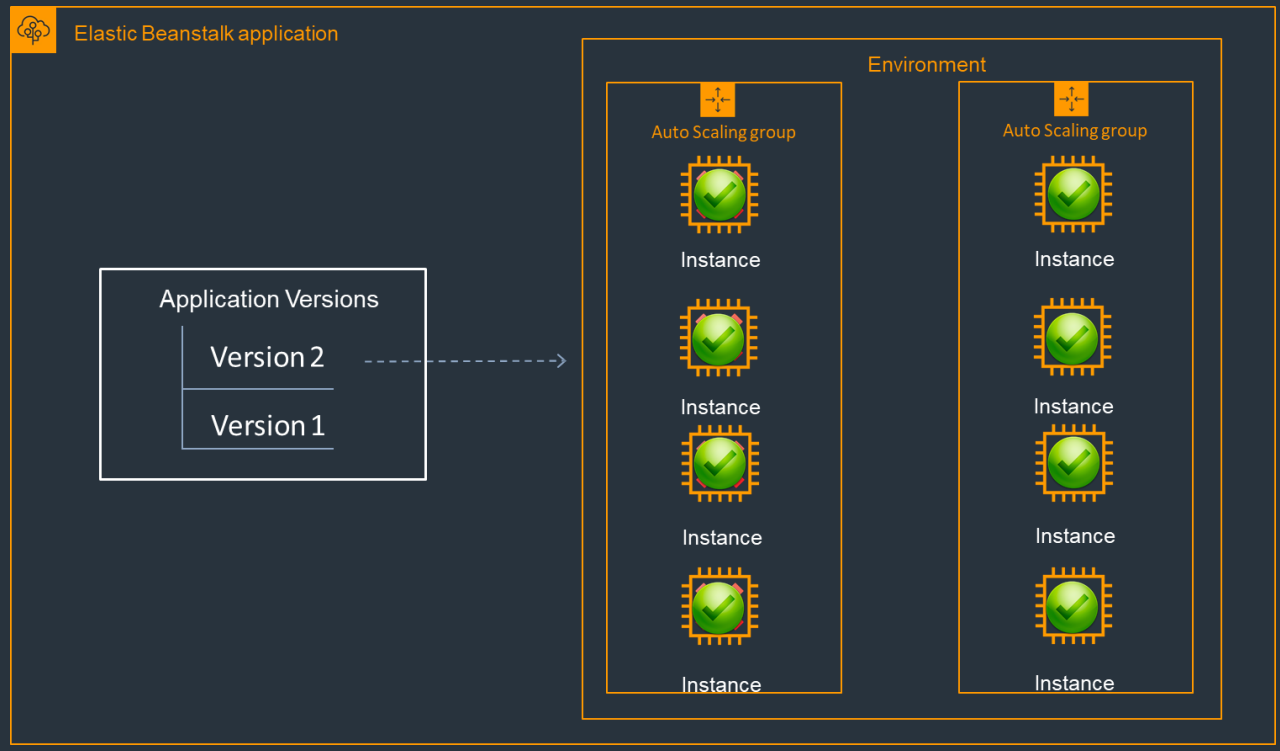
AWS Elastic Beanstalk – Rolling with Additional Batch Update

- Like Rolling but launches new instances in a batch ensuring that there is full availability.
- Application is running at capacity.
- Can set the batch size.
- Application is running both versions simultaneously.
- Small additional cost.
- Additional batch is removed at the end of the deployment.
- Longer deployment.
- Good for production environments.



AWS Elastic Beanstalk – Immutable Update

- Launches new instances in a new ASG and deploys the version update to these instances before swapping traffic to these instances once healthy.
- Zero downtime.
- New code is deployed to new instances using an ASG.
- High cost as double the number of instances running during updates.
- Longest deployment.
- Quick rollback in case of failures.
- Great for production environments.



AWS Elastic Beanstalk – Blue/green

- This is not a feature within Elastic Beanstalk
- You create a new "staging" environment and deploy updates there.
- The new environment (green) can be validated independently and you can roll back if there are issues.
- Route 53 can be setup using weighted policies to redirect a a percentage of traffic to the staging environment.
- Using Elastic Beanstalk, you can "swap URLs" when done with the environment test.
- Zero downtime.

Thank You!

Enabling Certainty

TATA CONSULTANCY SERVICES