



Building a Secure Web App



What are we building in Lab 1?

- We will setup a web application that connects to Twitter and reads information about a user that is identified by a twitter handle.
- This lab is broken up into 3 sections
 1. Deploy an Infrastructure with baseline Security
 2. Further restrict network access and implement Encryption at Rest
 3. Implement Perimeter Protection and enforce Encryption in Transit



What are we building in Section 1?

- We will setup a web application that connects to Twitter and reads information about a user that is identified by a twitter handle.
- In this lab we want to demonstrate:
 1. How to create Virtual Private Cloud with public and private subnets
 2. How to attach Internet Gateway
 3. Setting up of Network Access Control Lists for subnets and security groups for EC2 instances
 4. Launching two EC2 instances from pre-baked AMI that contains our web application
 5. Usage of NAT Gateways
 6. Usage of Application Load Balancer
 7. How to create IAM roles



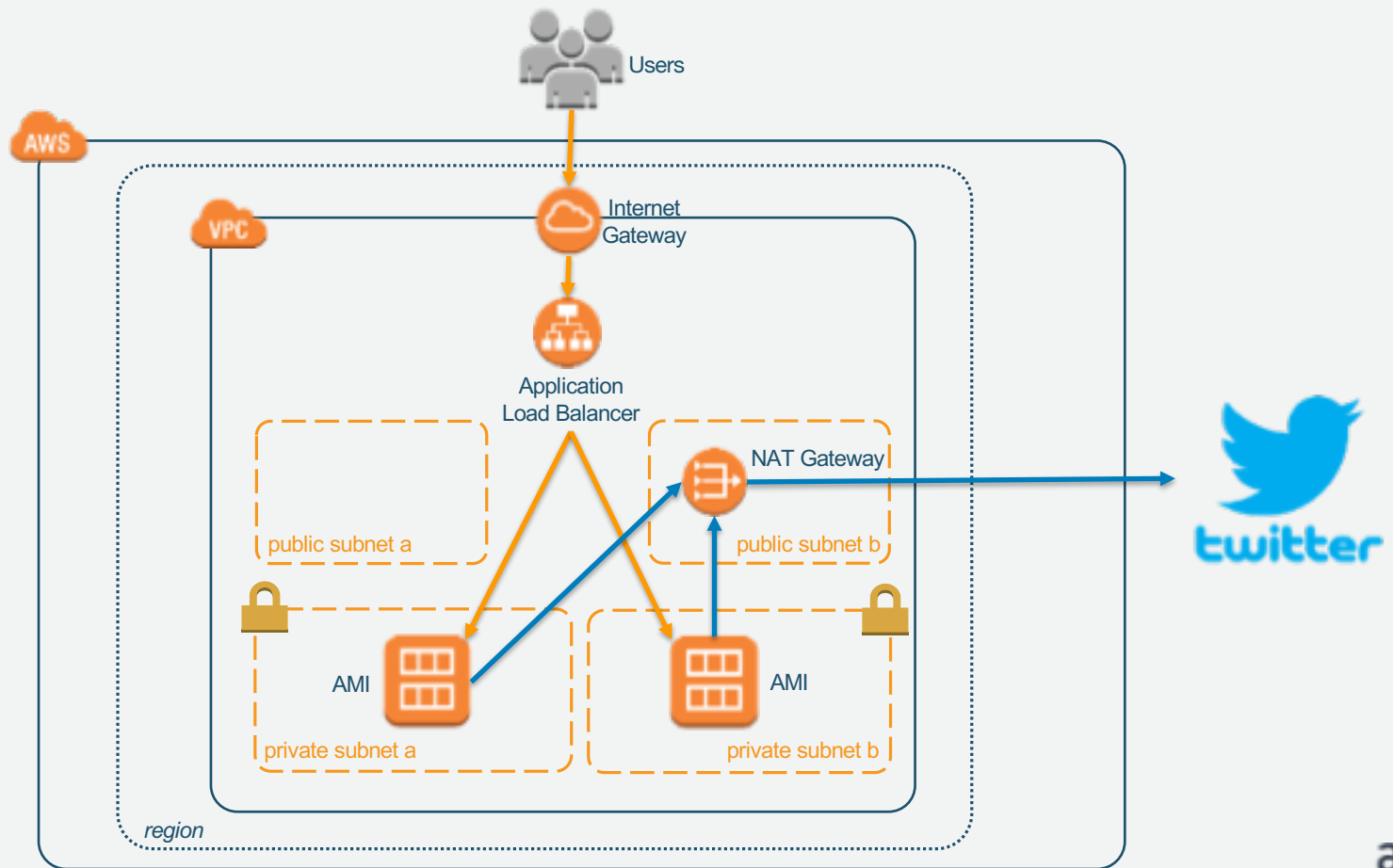
You can skip manual setup of Section 1 by running CloudFormation script:
https://s3.amazonaws.com/security-compliance-immersion-day/ImmersionDayCF_Module1.json





Section 1

Setting up secure EC2 environment



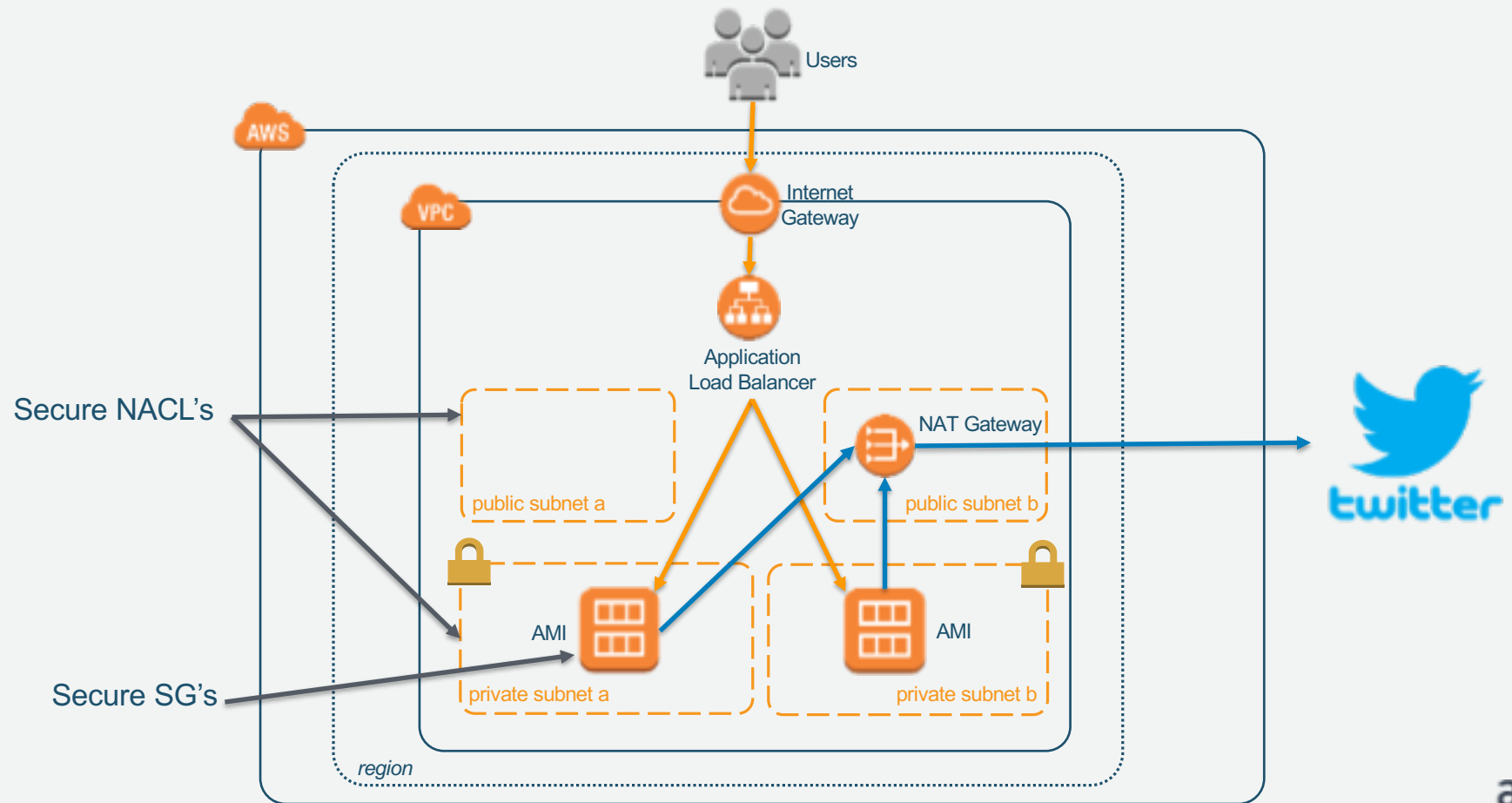
Network Access Control Lists and Security Groups

Security Group	Network ACL
Operates at the instance level (second layer of defense)	Operates at the subnet level (first layer of defense)
Supports allow rules only	Supports allow rules and deny rules
Is stateful: return traffic is automatically allowed, regardless of any rules	Is stateless: return traffic must be explicitly allowed by rules
We evaluate all rules before deciding whether to allow traffic	We process rules in number order when deciding whether to allow traffic
Must be applied to an instance	Automatically applies to all instances in the subnets



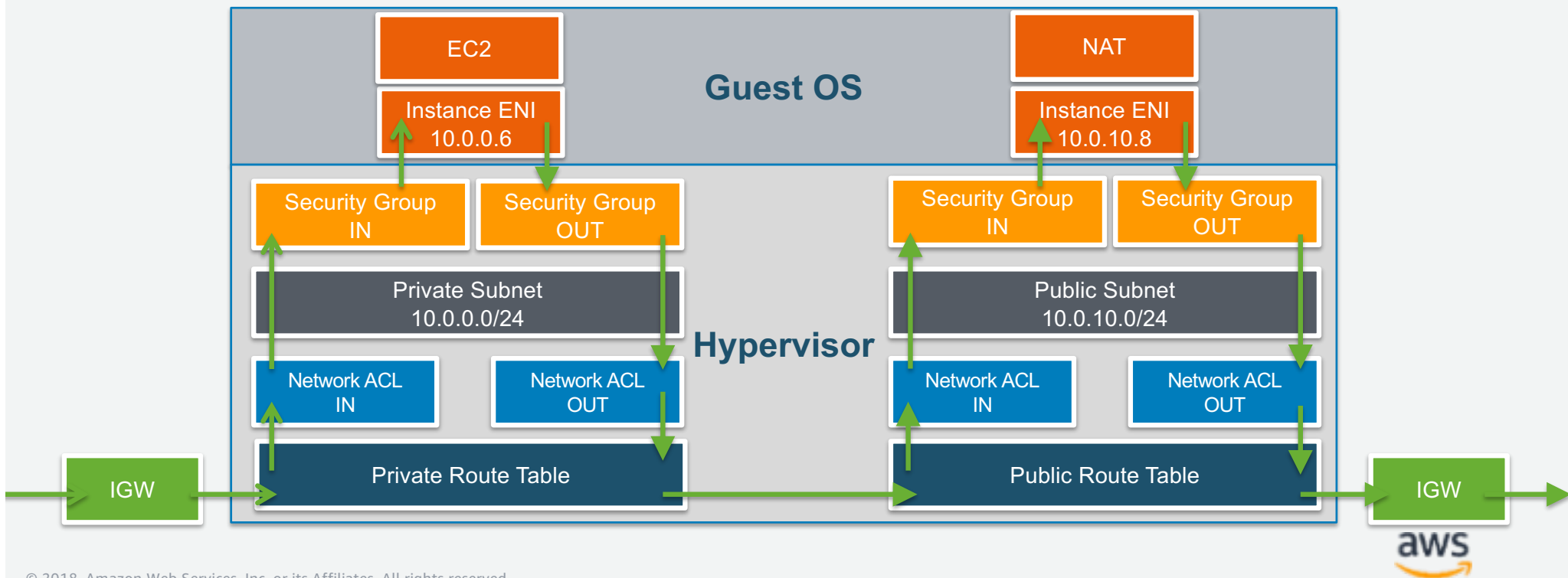
Section 1

Setting up secure EC2 environment



VPC Security Rules Evaluation

- The illustration below shows the packet flow from the Internet to a server in a private VPC subnet, which in turn sends a packet to a NAT gateway, which forwards it to the requested host on the Internet.
- The route table, network ACL and security group rules are processed entirely in the hypervisor layer.



Enforce Consistent Security On Your Hosts

Configure and harden EC2 instances based on security and compliance needs

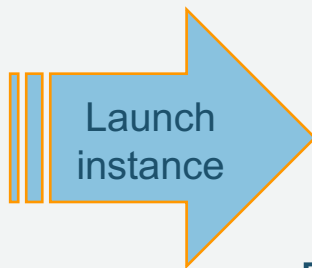
Host-based Protection Software

Restrict Access Where Possible

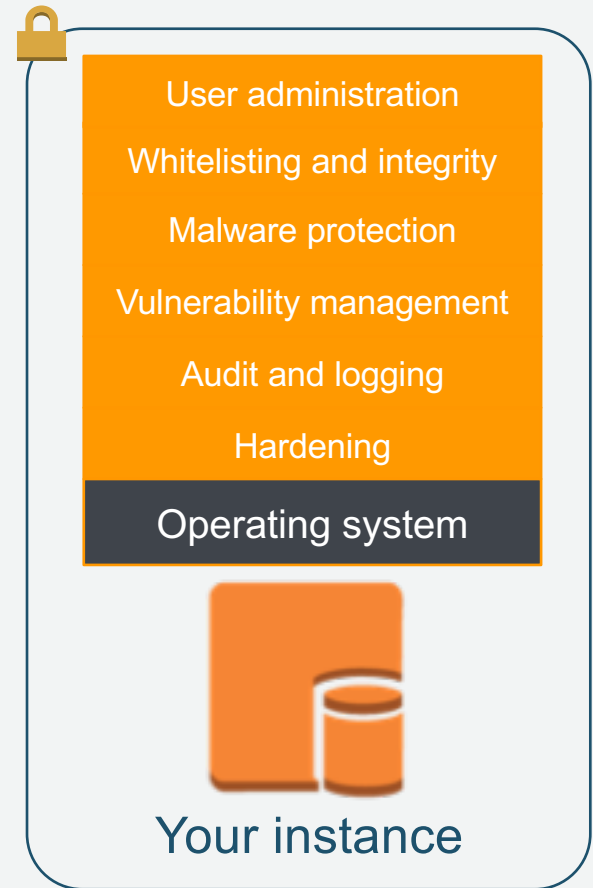
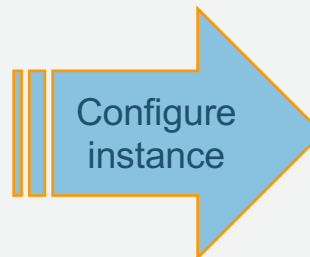
Launch with IAM Role



AMI catalog



Running instance



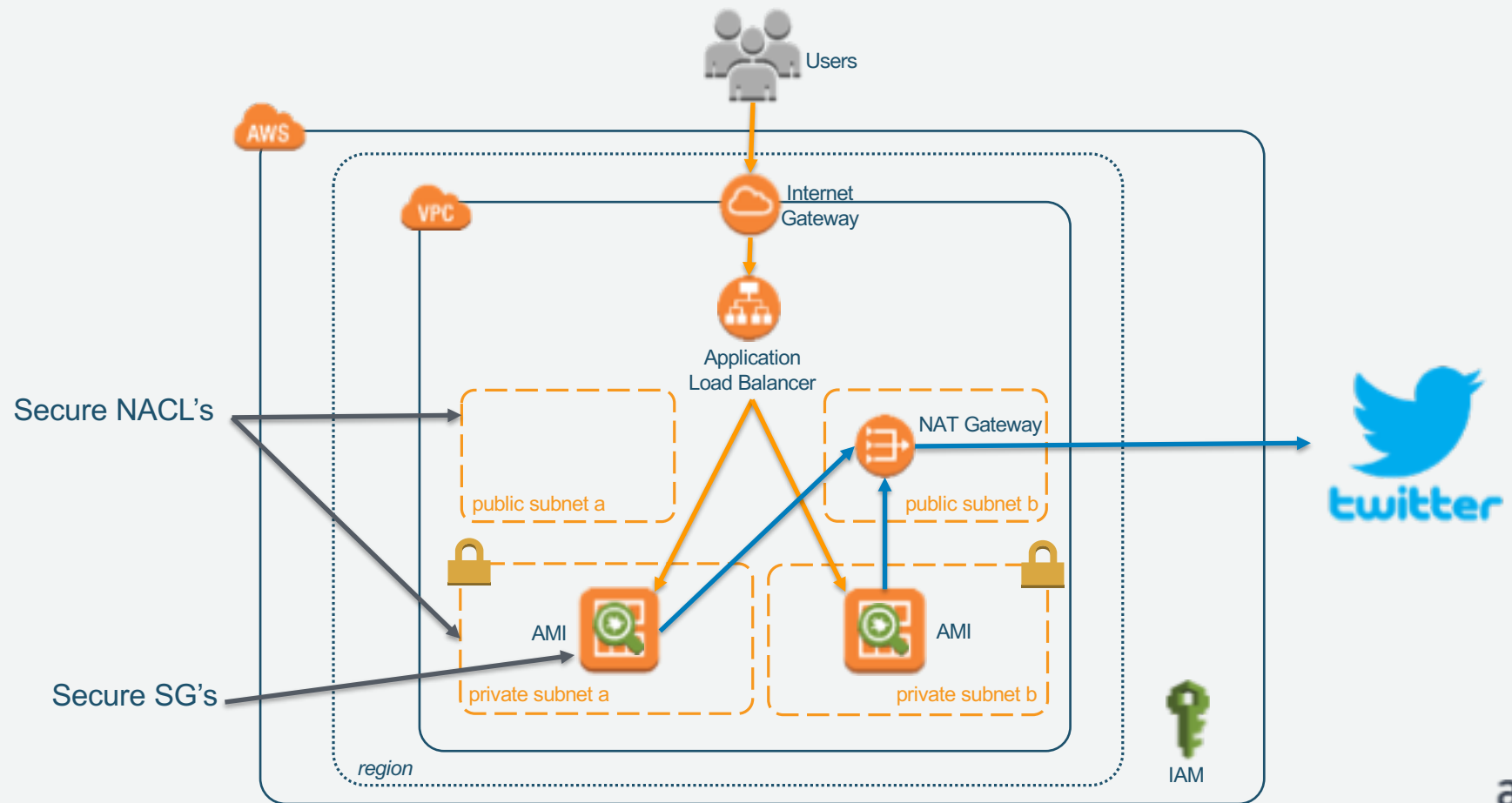
EC2 Roles

- Applications can securely make API requests from your instances
- You can delegate permission to make API requests using IAM roles as follows:
 1. Create an IAM role
 2. Define which accounts or AWS services can assume the role
 3. Define which API actions and resources the application can use after assuming the role
 4. Specify the role when you launch your instance, or attach the role to a running or stopped instance
- Key rotation and revocation can now all be handled in IAM



Section 1

Setting up secure EC2 environment





Step 1 – Set up a VPC

- Select us-east-1 region (N. Virginia)
- Create a new VPC called “*ImmersionDayVPC*” (CIDR block 10.0.0.0/16)
- Create 2 public and 2 private subnets
 1. “*PublicSubnet_1a*” in us-east-1a AZ with CIDR 10.0.0.0/24
 2. “*PublicSubnet_1b*” in us-east-1b AZ with CIDR 10.0.1.0/24
 3. “*PrivateSubnet_1a*” in us-east-1a AZ with CIDR 10.0.2.0/24
 4. “*PrivateSubnet_1b*” in us-east-1b AZ with CIDR 10.0.3.0/24





Step 2 – Set up Security Group

- Demo application running on EC2 instance is a Java-based web application running on Tomcat version 7.
- Create a security group “*ImmersionDaySecGroup*” with only HTTP allowed (port 80 and 8080) and source 0.0.0.0/0.
- The same security group will be used for all instances we start (EC2 and Application Load Balancer).

Step 3 – Create Internet and NAT Gateways

- Create and attach Internet Gateway "*ImmersionDayInternetGateway*" to the VPC
- Create NAT Gateway in the public subnet



Step 4 – Create Route Tables

- Create 2 new route tables:
 1. One for the public subnet "*PublicSubnetRouteTable*"
 2. One for the private subnets "*PrivateSubnetsRouteTable*"
- Associate private/public subnets to these new route tables and modify the routes accordingly:
 1. *PrivateSubnetsRouteTable* should have route to Target = NAT Gateway and Destination = 0.0.0.0/0
 2. *PublicSubnetRouteTable* should have route to Target = Internet Gateway and Destination = 0.0.0.0/0





Step 5 – Create EC2 role

- Go to IAM and select Roles from the left hand side menu options
- Create new role for EC2 instance and name it "ImmersionDayEC2Role"
- Add following permissions to this role:
 1. **AmazonDynamoDBFullAccess** (this gives our EC2 permission to read/write from DynamoDB database)
 2. **AmazonSSMReadOnlyAccess** (this gives our EC2 instance permission to read from System Manager's Parameter Store)





Step 6 – EC2 setup

- Launch one EC2 instance in each private subnet with:
AMI ID = “ami-0aa3c8ac7278c6d3e” (if you are running in **us-east-1**)
or
AMI ID = “ami-0994313865b869a92” (if you are running in **eu-central-1**)
and
AMI Name= “SecurityComplianceImmersionDay”
(t2.micro, no additional EBS volumes)
- Add previously created role “ImmersionDayEC2Role”
- Use the security group that we created previously “*ImmersionDaySecGroup*”
- Tag instances with Key = *instance*, Value = *immersionday*
- If you don’t have a Key/Pair already in us-east-1 or eu-central-1 region, create one and use it with these instances



Step 7 – Set up Application Load Balancer

- Create internet-facing ALB “*ImmersionDayALB*” and connect it with the public subnets:
 - Set listeners to be HTTP on port 80
 - Set HealthCheck path /tweetstats/home on port 8080
 - Create Target Group and add EC2 instances from private subnets on port 8080 to this group
- ALB will be responding to requests made on port 80 and will be forwarding them to target instances on port 8080

Step 8 – Check if all is working fine

- Type in a browser ALB DNS endpoint with “/tweetstats/home” at the end and see if it works, like:

<http://ImmersionDayALB-11111111111.us-east-1.elb.amazonaws.com/tweetstats/home>

- You should see this:





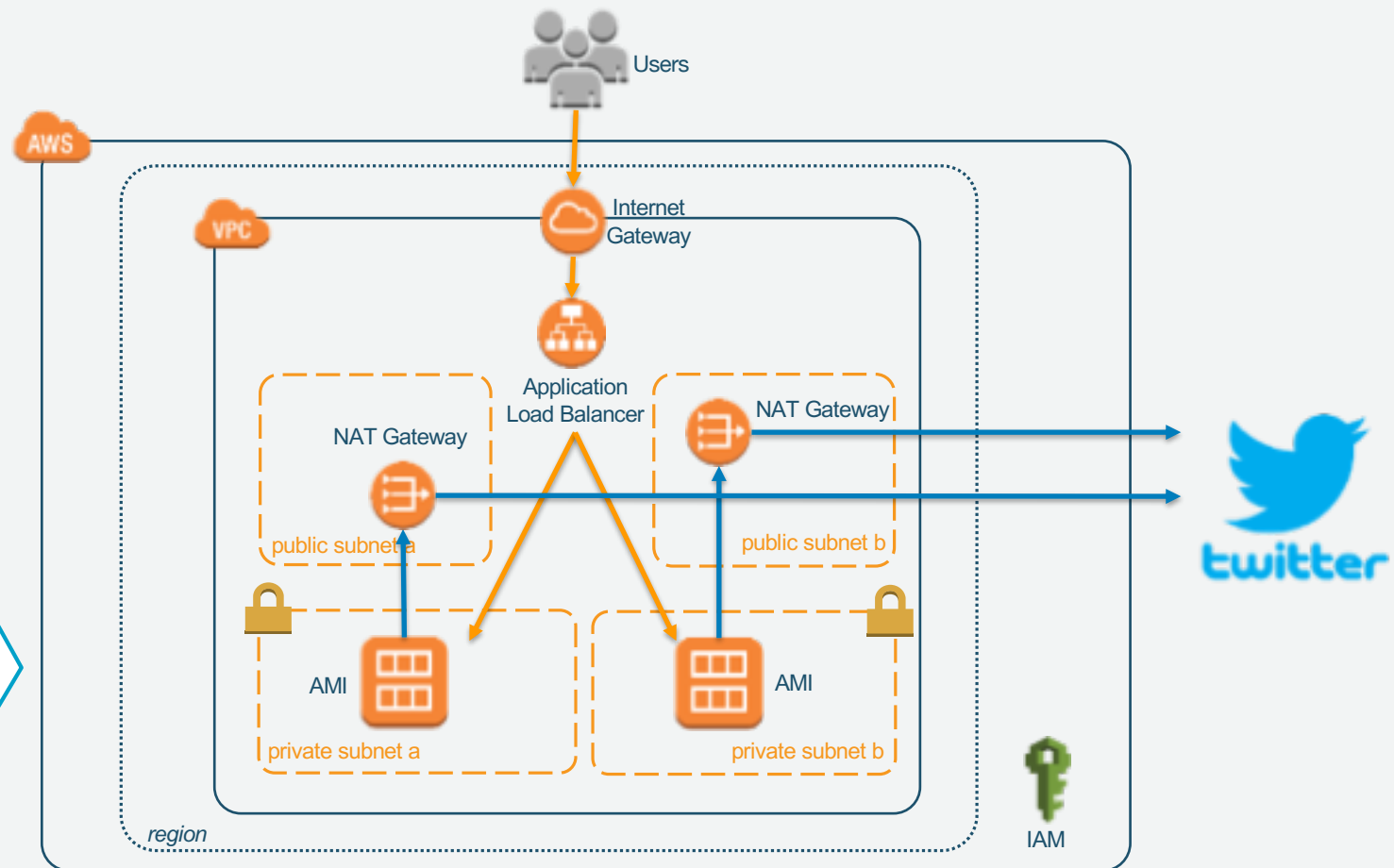
Lab 1

Real-world deployment

Note:

In real-world deployments, we would use 2 NAT Gateways, one in each public subnet in different AZ's. We would configure route tables so that EC2 instances use NAT Gateways from the same AZ.

This way we increase egress throughput and availability.





What are we building in Section 2?

- This lab is continuation of the work we did in Section 1.
- In this lab we want to demonstrate:
 1. How to create VPC endpoints (gateway and interface)
 2. How to create encryption keys with KMS
 3. How to manage secrets with Parameter Store
 4. How to encrypt data in DynamoDB using Client-side encryption



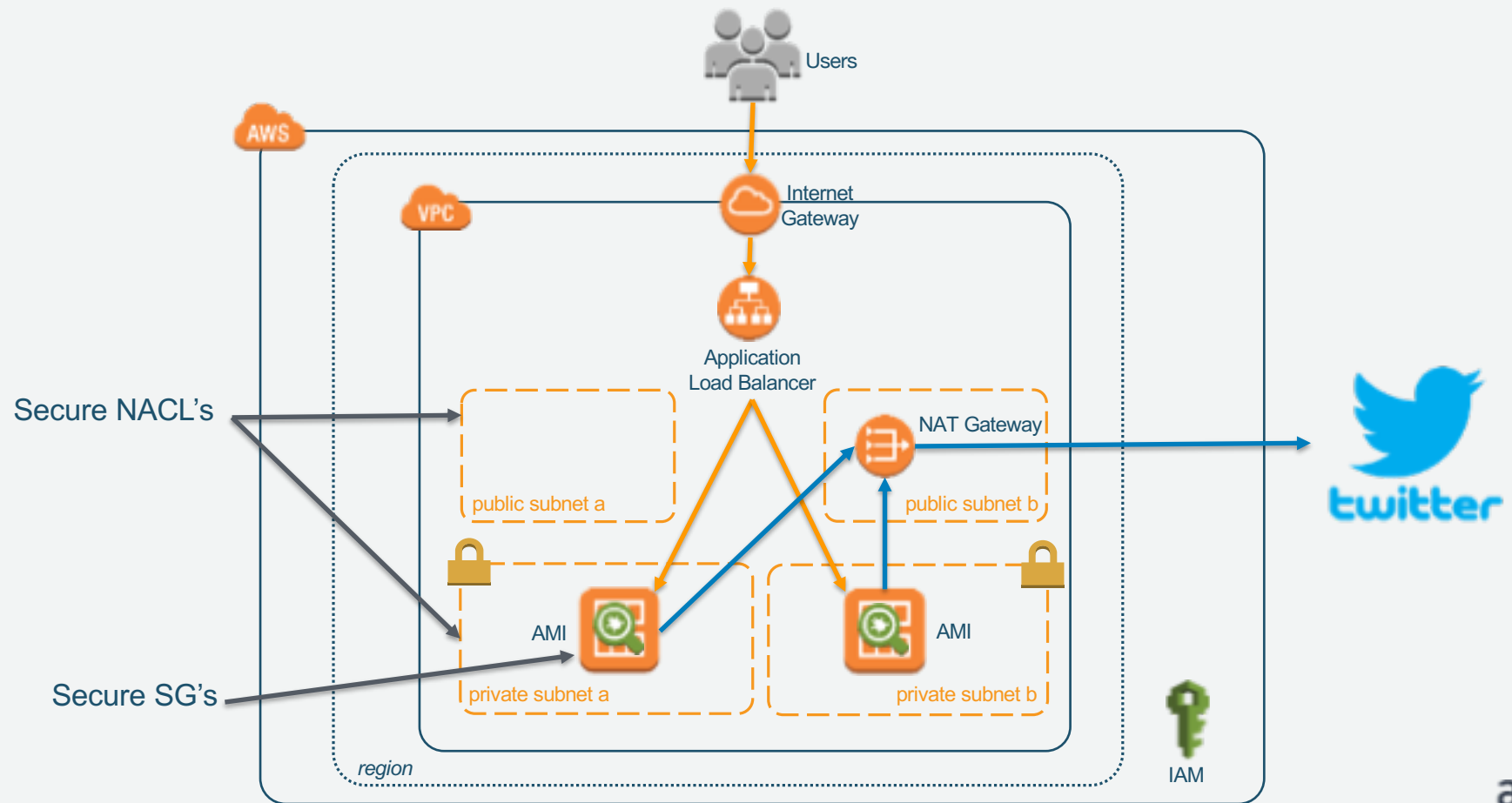
You can skip manual setup of Section 2 by running CloudFormation script:
https://s3.amazonaws.com/security-compliance-immersion-day/ImmersionDayCF_Module2.json

Note: "Step 3 – Set up parameter Store" still needs to be done manually as CloudFormation didn't support creating parameters at the time this was created.





Section 1 Recap



VPC Endpoints can be used to improve security

- Network-based access can be restricted to AWS
- Can apply NACL's and SG's to further restrict communication
- Policy-based access can be restricted to AWS resources

Storing secret parameters

Where do we store configuration data for our applications?

We need to have a place to:

- Centrally store and find configuration data
- Have repeatable, automatable management (e.g. SQL connection strings)
- Have granular access control – view, use and edit values
- Encrypt sensitive data using your own AWS KMS keys

➤ In AWS, there are two ways to do it:

1. AWS Systems Manager **Parameter Store**, or
2. AWS **Secrets Manager**

Comparing AWS Systems Manager Parameter Store and AWS Secrets Manager

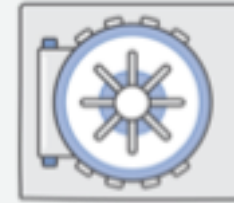
AWS Systems Manager Parameter Store



- Secure storage for configuration data, which can include secrets
- Reference values using the unique name specified during creation
- Use parameters in scripts for configuration and automation
- Parameter Store is free of charge



AWS Secrets Manager



- A service to manage the lifecycle for secrets in your organization
- Helps you meet security and compliance requirements by rotating secrets automatically
- Built-in integrations for Amazon RDS that can rotate database credentials on your behalf
- Extensible via Lambda
- Secrets Manager is pay as you go with \$0.40 per secret per month and \$0.05 per 10,000 API calls



How is data encrypted in DynamoDB?

- We are using *aws-encryption-sdk-java* library to encrypt and decrypt data to and from DynamoDB.

// **Instantiate the encryption SDK**

```
AwsCrypto crypto = new AwsCrypto();
```

// **Set up the master key provider**

```
KmsMasterKeyProvider prov = new KmsMasterKeyProvider  
("arn:aws:kms:" + region.getName() + ":" + response.getAccount() + ":key/" + kmsID);
```

// **Encrypt data**

```
String ciphertext = crypto.encryptString(prov, data, context).getResult();
```

// **Decrypt data**

```
CryptoResult<String, KmsMasterKey> decryptResult = crypto.decryptString(prov, data);
```

We keep this ID in
Parameter Store



How is data encrypted in DynamoDB?

☒ New role: DynamoDBAutoscaleRole

☐ Existing role with pre-defined policies [\[Instructions\]](#)

Role Name*

Encryption At Rest

You may enable encryption for your DynamoDB table to help protect data at rest. [Learn more](#)

☒ Enable encryption

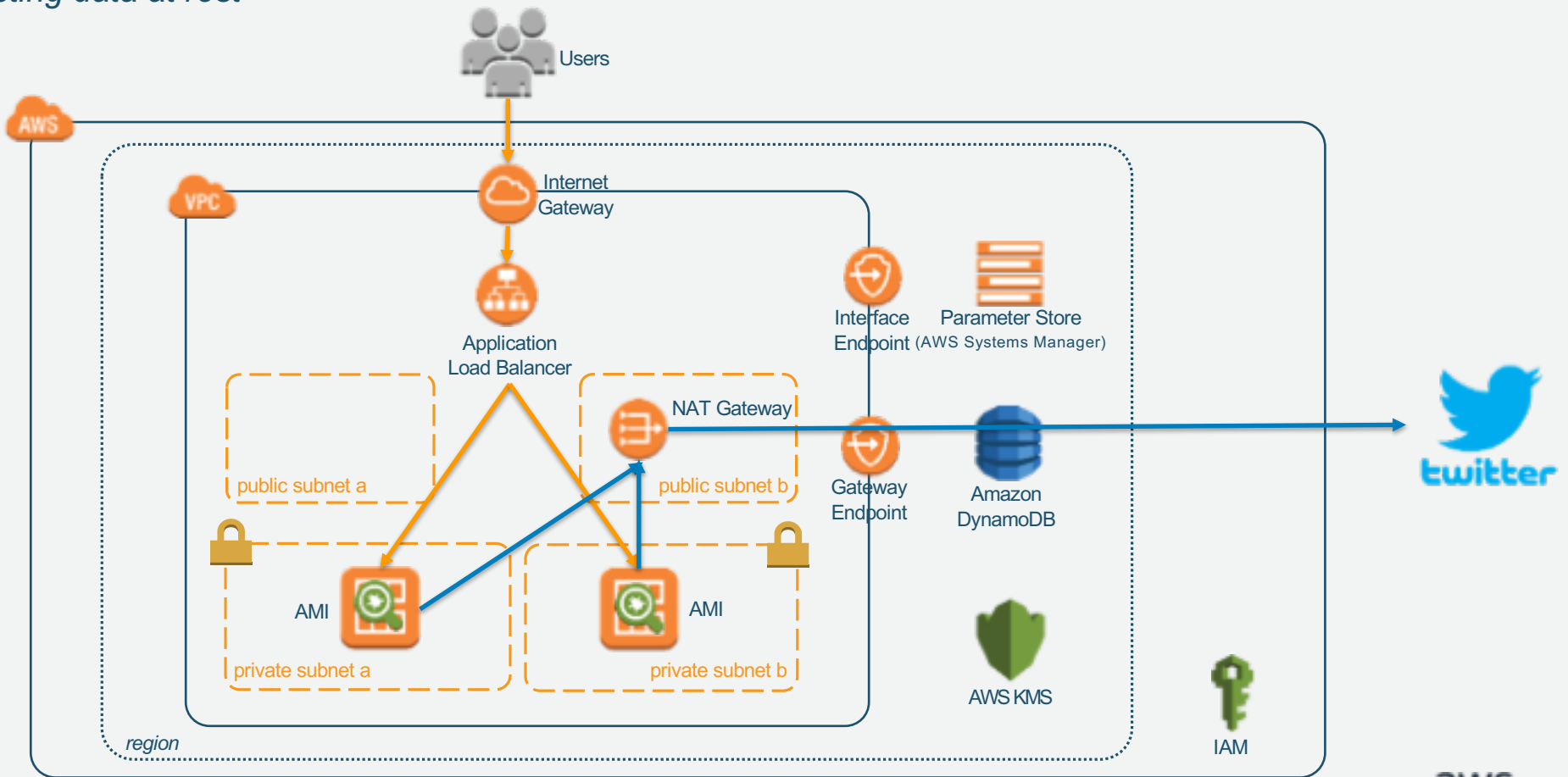
Additional charges may apply if you exceed the AWS Free Tier levels for CloudWatch or Simple Notification Service. Advanced alarm settings are available in the CloudWatch management console.

[Cancel](#) [Create](#)



Section 2

Protecting data at rest





Step 1 – Set up VPC Endpoints

- Go to VPC endpoints and create one of DynamoDB and one for SSM
- Attach both private subnets to these endpoints

Step 2 – Create an encryption key in KMS

- Go to IAM -> Encryption Keys
- Create a new KMS key with name “ImmersionDayKey”
- Key Administrator is you
- Key User is previously defined ImmersionDayEC2Role



Step 3 – Set up Parameter Store

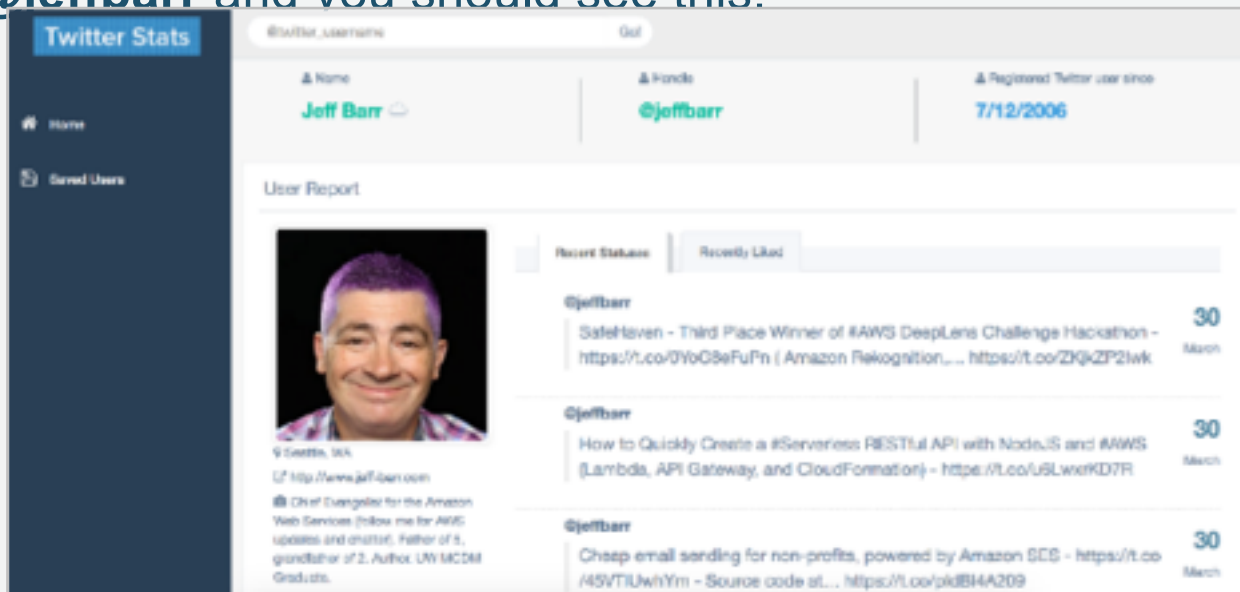
- Go to Systems Manager -> Parameter Store
- Create a new parameter with the name “kmsID”
- It should be type String
- Value should be ID of the encryption key that was created in KMS in the previous step

Step 4 – Check if all is working fine

- Type in a browser ALB DNS endpoint with “/tweetstats/home” at the end and see if it works, like:

<http://ImmersionDayALB-11111111111.us-east-1.elb.amazonaws.com/tweetstats/home>

- Search for **@ieffbarr** and you should see this:



The screenshot displays the Twitter Stats interface for the user @ieffbarr. The left sidebar contains a 'Twitter Stats' header and a 'Saved Users' section. The main content area shows the user's profile with the name 'Jeff Barr', handle '@ieffbarr', and a bio: 'Chief Evangelist for the Amazon Web Services (follow me for AWS updates and chatter). Father of 5, grandfather of 2. Author. UW MCDM Graduate.' Below the profile picture is a 'User Report' section with tabs for 'Recent Statuses' and 'Recently Liked'. The 'Recent Statuses' tab is active, showing three tweets from @ieffbarr, each with a retweet count of 30 and a date of March.

Recent Statuses	Recently Liked
<p>@ieffbarr Safehaven - Third Place Winner of #AWS DeepLens Challenge Hackathon - https://t.co/0YbG8eFuPn (Amazon Rekognition... https://t.co/ZKqk2P2lwk</p>	30 March
<p>@ieffbarr How to Quickly Create a #Serverless RESTful API with Node.js and #AWS (Lambda, API Gateway, and CloudFormation) - https://t.co/uSLwvKD7R</p>	30 March
<p>@ieffbarr Cheap email sending for non-profits, powered by Amazon SES - https://t.co/45VTUwhYm - Source code at... https://t.co/pk0B4A209</p>	30 March



Step 5 – Check if encryption works

- Go to DynamoDB and look at the items in table “saved-tweeter-users-table”.
- Note: DynamoDB and this table is automatically created when our demo application first starts.
- Check if items for “jeffbarr ”are encrypted:

Scan: [Table] saved-tweeter-users-table: name ^ Viewing 1 to 7 items

Scan [Table] saved-tweeter-users-table: name ^

+ Add filter

Start search

	name	desc	fullName	picUrl
<input type="checkbox"/>	jeffbarr	AYADeJk0Chy6BAuC1...	AYADeEwjv8wIQMdFVM...	AYADeHu6h6cW825bFXPRqz...





What are we building in Section 3?

- This lab is continuation of the work we did in Section 2.
- In this lab we want to demonstrate:
 1. How to create web distribution with CloudFront
 2. How to use Web Application Firewall on the CloudFront distribution



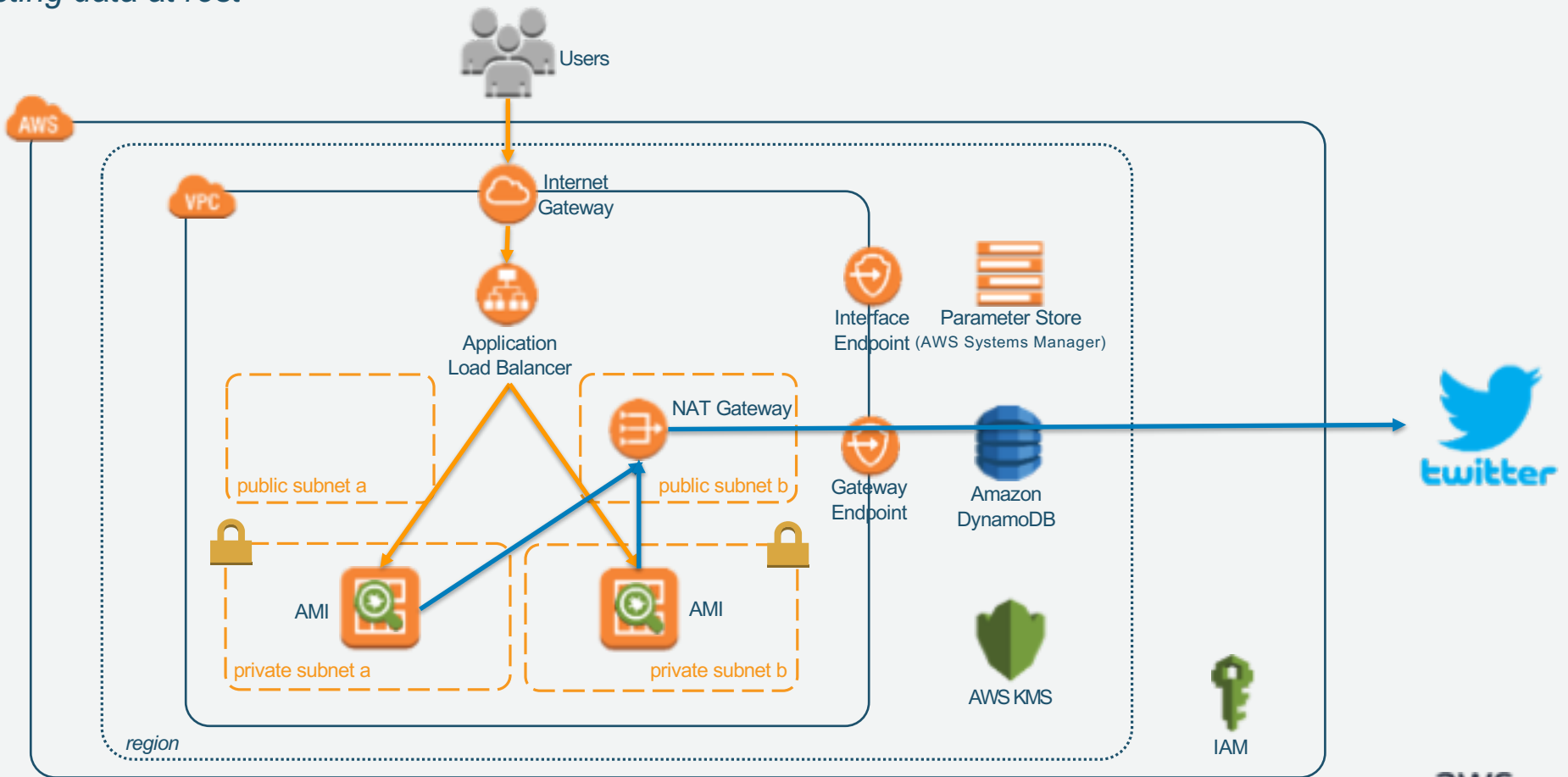
You can skip manual setup of Section 3 by running CloudFormation script:
https://s3.amazonaws.com/security-compliance-immersion-day/ImmersionDayCF_Module3.json





Section 2 - Recap

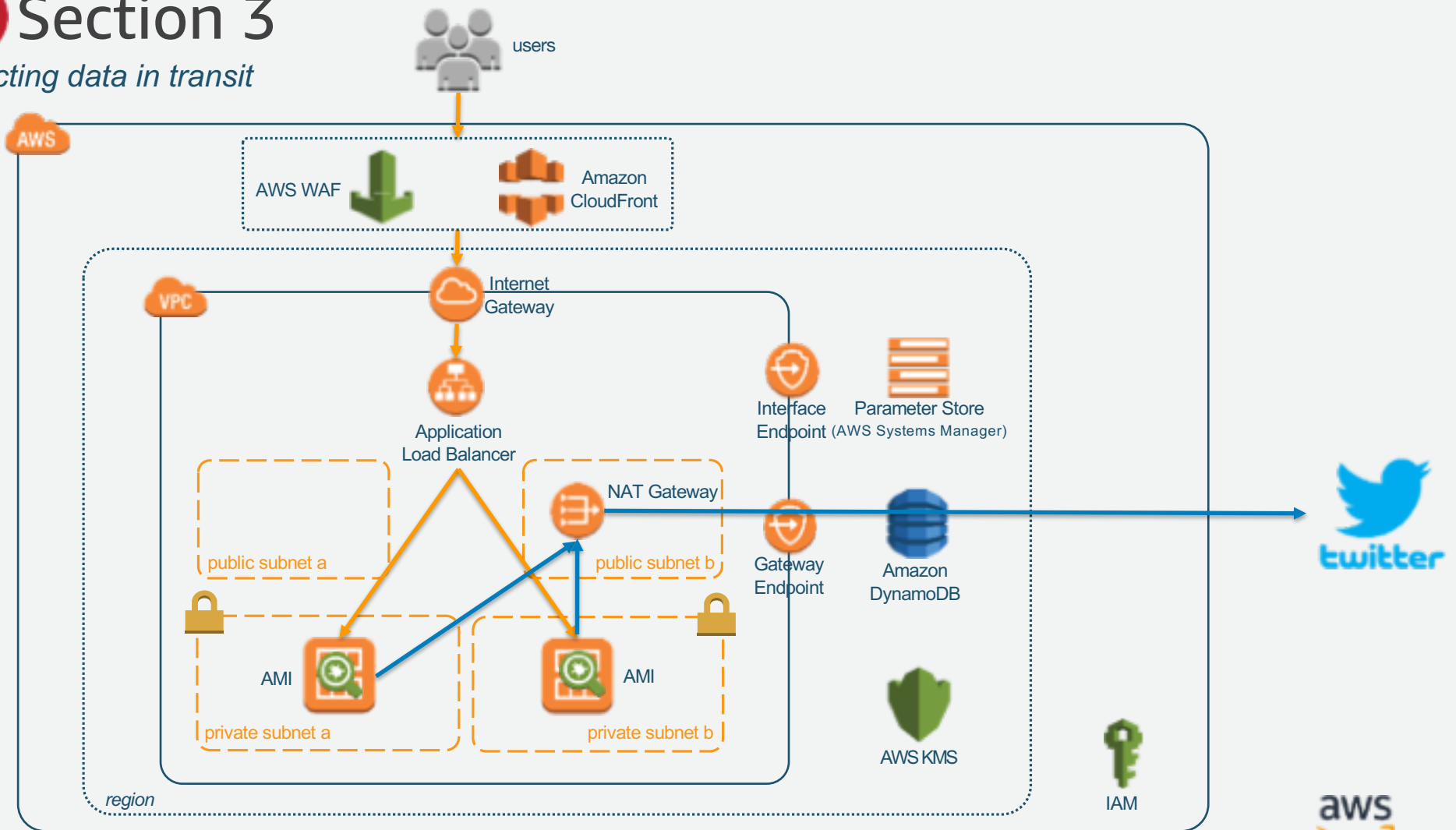
Protecting data at rest





Section 3

Protecting data in transit





Step 1 – CloudFront setup

- Create CloudFront web distribution and point it to the ALB origin (origin protocol is HTTP only)
- Viewer Protocol Policy: Redirect HTTP to HTTPS
- Cache: None
- Object Caching: Use Origin Cache Headers
- Forward Cookies: All
- Query String Forwarding and Caching: Forward all, cache based on all



Now, try to see if CloudFront works:

- Type in a browser your CloudFront endpoint with “/tweetstats/home” at the end and see if it works.
- Note: it takes about 10mins for CloudFront distribution to become active...
 - If it's not active immediately for you, go ahead and enable WAF



Step 2 – WAF setup

- Create WAF Web ACL for CloudFront distribution.

- In the CloudFront WAF, create a ***“String and Regex Matching”*** condition that denies all requests with “immersion” in the query string.

Create string match condition

Filter settings

Specify the settings that you want to use to allow or block web requests. If you add more than one filter to a string match condition, a web request needs to match only one of the filters for the request to match the string match condition. (The filters are ORed together.)

Part of the request to filter on	Query string	i
Match type	Contains	i
Transformation	None	i
Value is base64-encoded	<input type="checkbox"/>	i
Value to match*	immersion	i

Add filter



Step 2 – WAF setup

- The previously created condition should have a rule like below, which means to block all requests that match the condition in the query string

When a request ✕

does ▾

match at least one of the filters in the string match condition ▾

ImmersionDayMatch ▾

Filters in ImmersionDayMatch

Query string contains: "immersion".

Add condition



Now, try to see if WAF for CloudFront works:

- You need to edit CloudFront settings and include WAF!
- Type in a browser your CloudFront endpoint with “/tweetstats/home?**query=immersion**” at the end and see if it works
- You should get “Request Blocked” page.