# Best practices for AWS security

Julien Simon
Principal Technical Evangelist
julsimon@amazon.fr
@julsimon

amazon web services

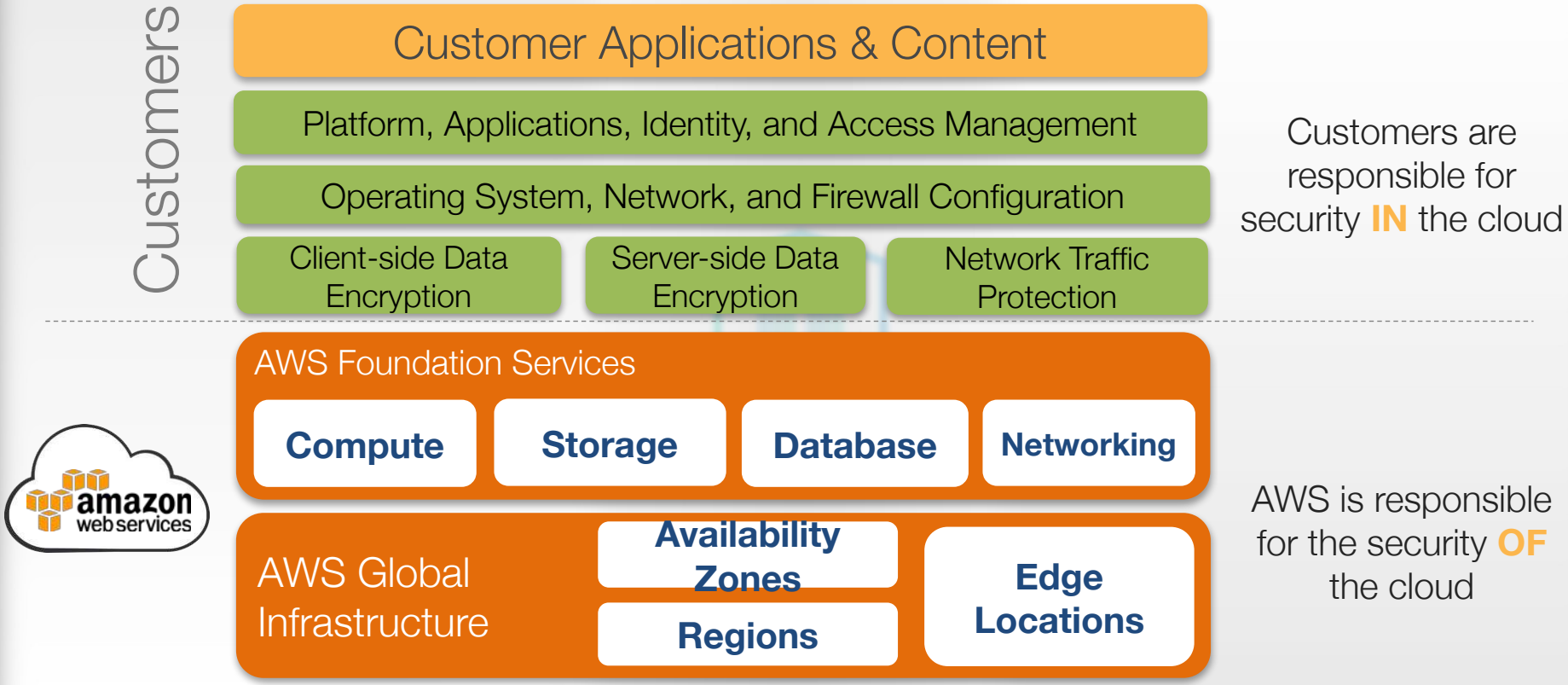
Pop-up Loft
TEL AVIV

# Agenda

- Understand the Shared Security Model
- Encrypt everything
- Manage users and permissions
- Log everything
- Automate security checks

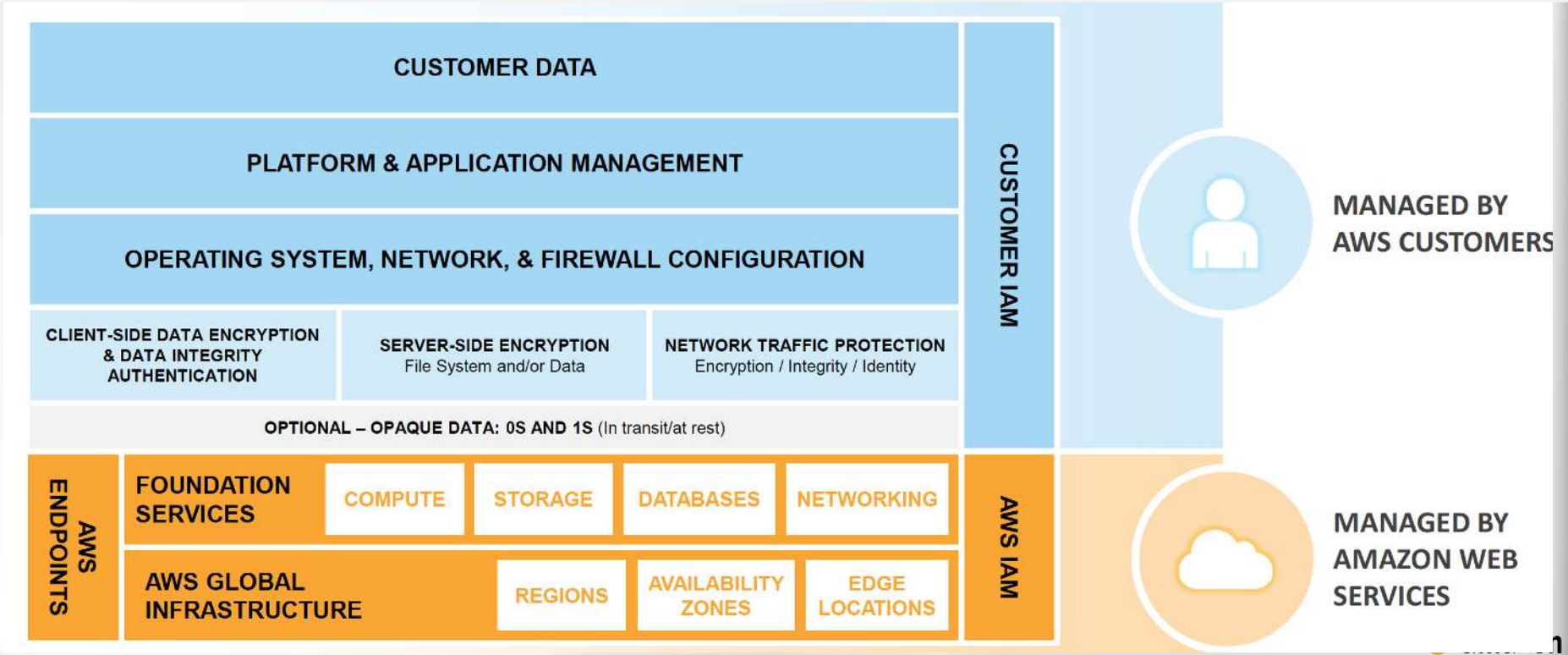
# Shared Security Model

# AWS Shared Responsibility Model



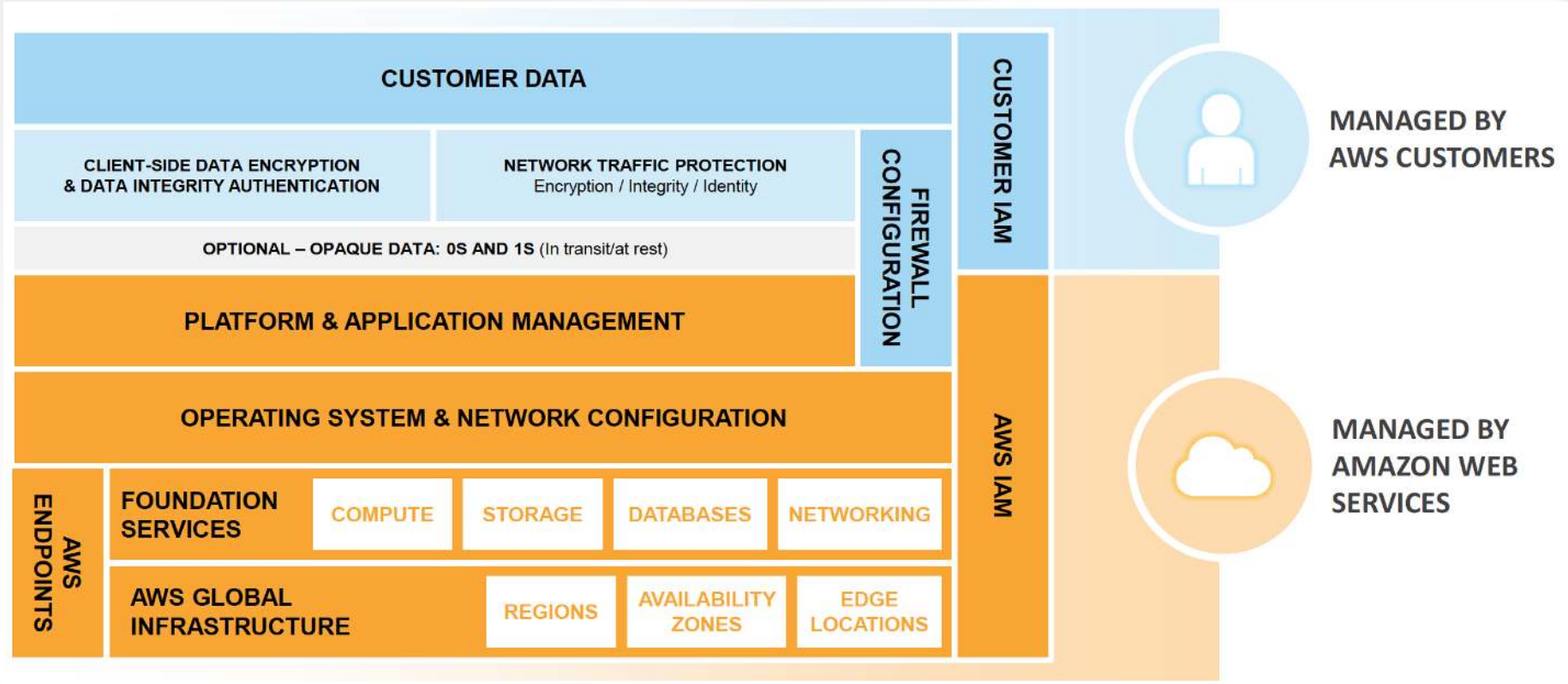
# Shared Security Model: Infrastructure Services

Such as Amazon EC2, Amazon EBS, and Amazon VPC



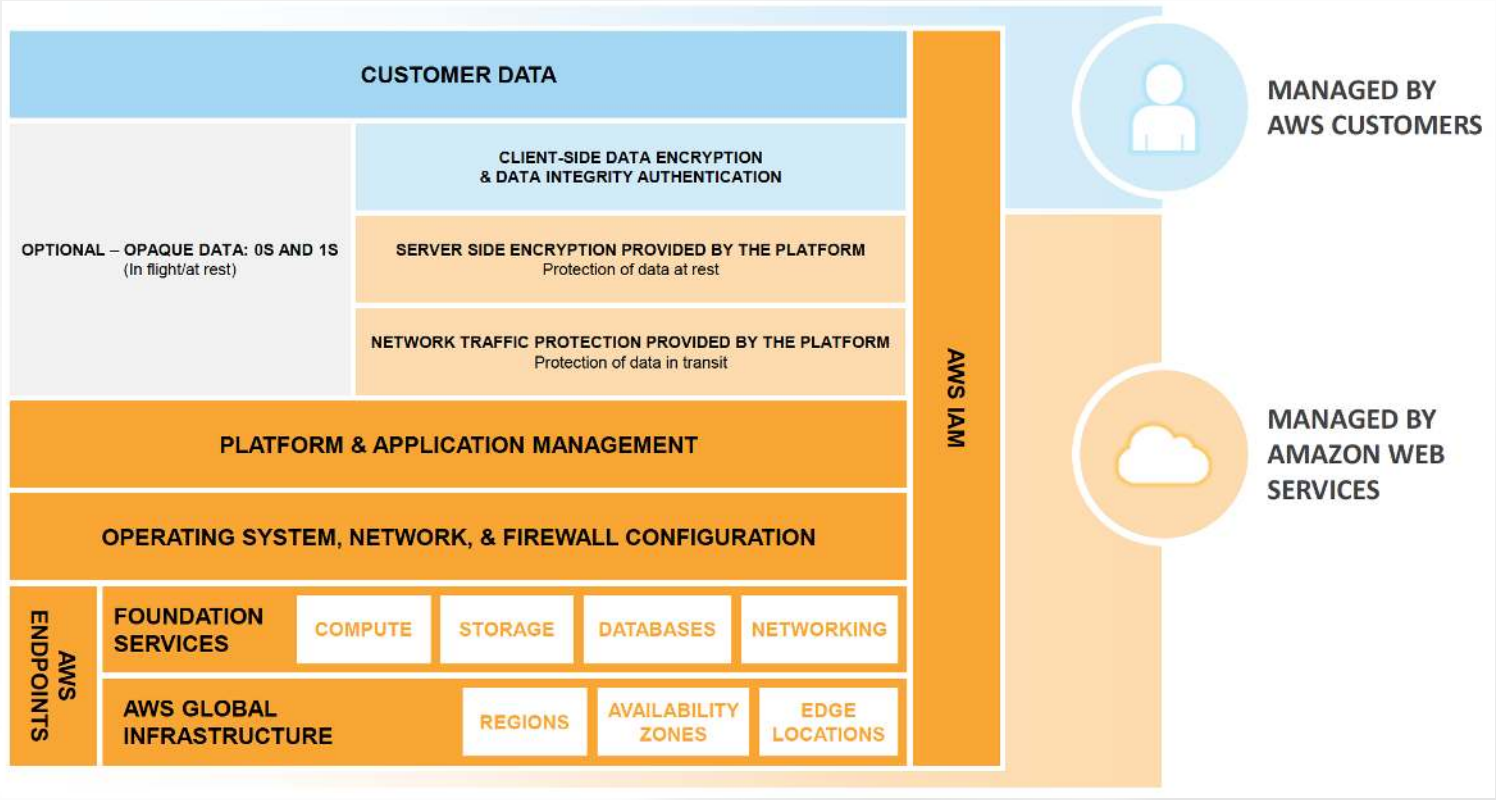
# Shared Security Model: Platform Services

Such as Amazon RDS and Amazon EMR



# Shared Security Model: Managed Services

Such as Amazon S3 and Amazon DynamoDB




amazon web services

# Encrypt everything

# Encryption options

## Native server-side encryption for most services

- S3, EBS, RDS, Redshift, etc.

## Flexible key management

- AWS Key Management Service
- AWS CloudHSM

## 3rd-party encryption

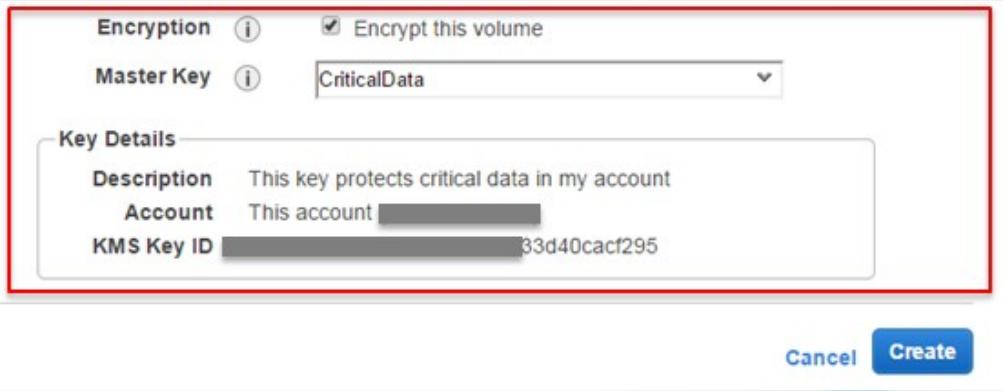
- Trend Micro, SafeNet, Vormetric, Hytrust, Sophos etc.
- AWS Marketplace : https://aws.amazon.com/marketplace/

## Client-side encryption

- Tricky business, please be careful!

# Server-side encryption

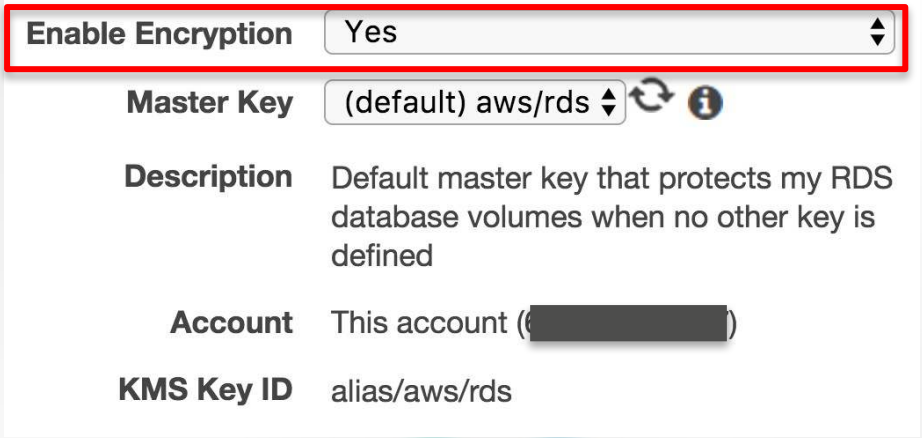
## Amazon EBS




```
create-volume [--dry-run | --no-dry-run] [--size <value>]
[--snapshot-id <value>] --availability-zone <value>
[--volume-type <value>] [--iops <value>]
[--encrypted | --no-encrypted] [--kms-key-id <value>]
[--cli-input-json <value>] [--generate-cli-skeleton]
```

# Server-side encryption

## Amazon RDS

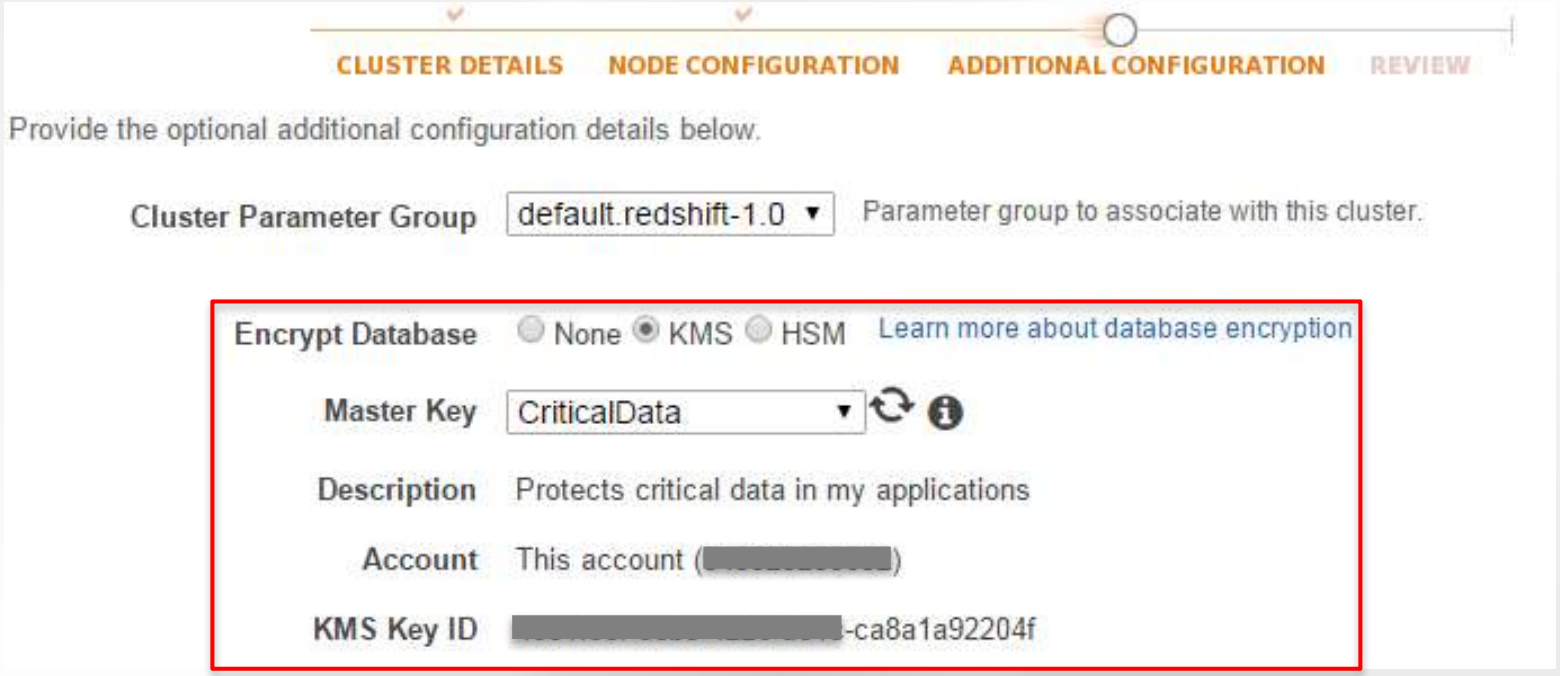



```
aws rds create-db-instance --region us-west-2
--db-instance-identifier myrdsinstance \
--allocated-storage 20 --storage-encrypted \
[ --kms-key-id xxxxxxxxxxxxxxxxx ]
--db-instance-class db.m4.large --engine mysql \
--master-username myawsuser --master-user-password myawsuser
```

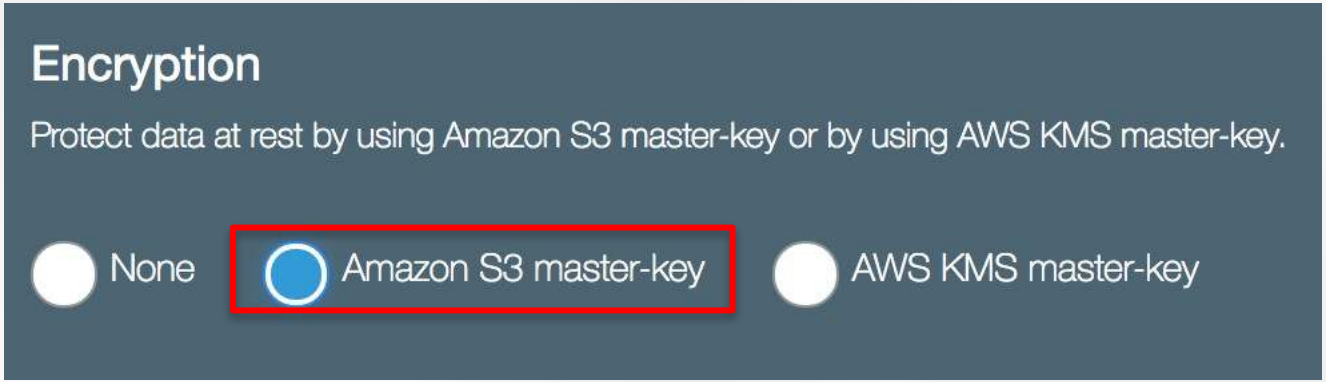
http://docs.aws.amazon.com/cli/latest/reference/rds/create-db-instance.html

# Server-side encryption

## Amazon Redshift



# Server-side encryption on S3 (SSE-S3)



# SSE-S3 with the AWS SDK for Java

```
File file = new File(uploadFileName);
PutObjectRequest putRequest = new PutObjectRequest(bucketName, keyName, file);

// Request server-side encryption.
ObjectMetadata objectMetadata = new ObjectMetadata();
objectMetadata.setSSEAlgorithm(ObjectMetadata.AES_256_SERVER_SIDE_ENCRYPTION);

putRequest.setMetadata(objectMetadata);

PutObjectResult response = s3client.putObject(putRequest);
System.out.println("Uploaded object encryption status is " +
                   response.getSSEAlgorithm());
```

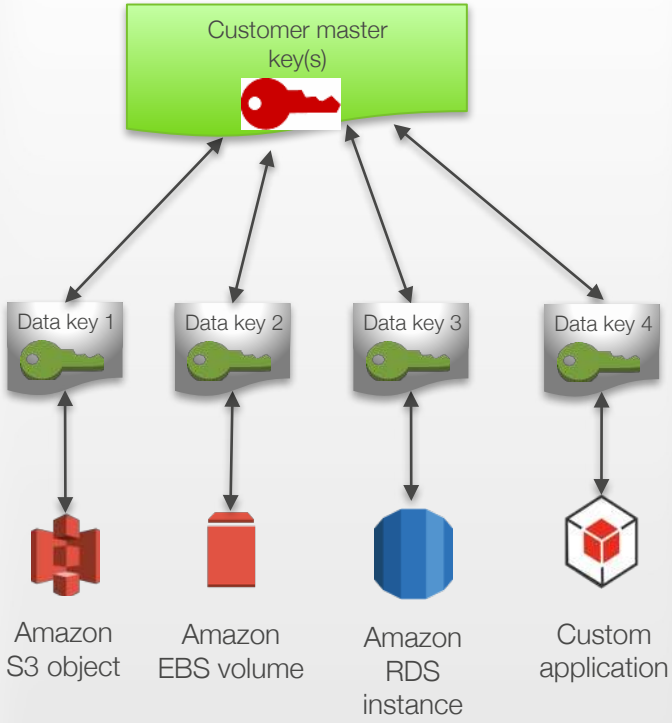
# AWS KMS

Two-tiered key hierarchy using envelope encryption:

- Unique data key encrypts customer data
- AWS KMS master keys encrypt data keys

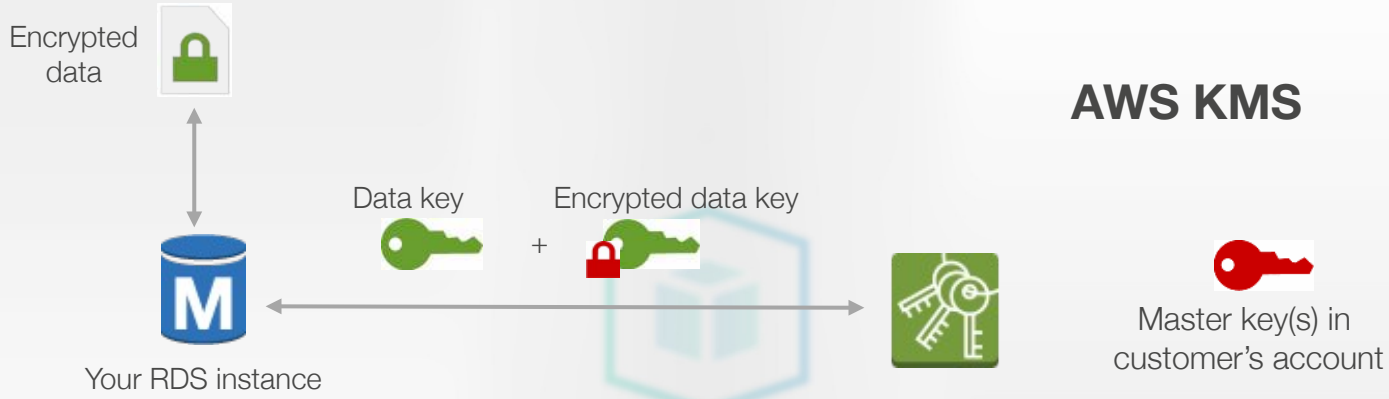
Benefits:

- Limits risk of compromised data key
- Better performance for encrypting large data
- Easier to manage small number of master keys than millions of data keys
- Centralized access and audit of key activity




https://aws.amazon.com/kms/

# How keys are used to protect your data




1. Service requests encryption key to use to encrypt data, passes reference to master key in account
2. Client request authenticated based on permissions set on both the user and the key
3. A unique data encryption key is created and encrypted under the KMS master key
4. Plaintext and encrypted data key returned to the client
5. Plaintext data key used to encrypt data and then deleted when practical
6. Encrypted data key is stored; it's sent back to KMS when needed for data decryption

https://aws.amazon.com/kms/

# Encryption SDKs

- Different from the AWS SDKs

- Java

https://aws.amazon.com/blogs/security/how-to-use-the-new-aws-encryption-sdk-to-simplify-data-encryption-and-improve-application-availability/

- Python (released yesterday!)

https://aws.amazon.com/blogs/security/new-aws-encryption-sdk-for-python-simplifies-multiple-master-key-encryption/

# Manage Permissions

# Identity and Access Management (IAM)

1. Create users



## Advantages

- Unique credentials
- Easier to rotate
- Easier to track

# Identity and Access Management (IAM)

1. Create users
2. Apply the principle of least privilege



## Advantages

- Reduce the risk of human error
- Finer control
- Easier to add permissions than to remove them
- Access Advisor tells you what permissions are actually used

# Identity and Access Management (IAM)

1. Create users
2. Apply the principle of least privilege
3. Factorize permissions with groups



## Advantages

- Simplest way to manage permissions for similar users

# Identity and Access Management (IAM)

1. Create users
2. Apply the principle of least privilege
3. Factorize permissions with groups
4. Use conditional permissions for privileged accounts (time, IP adress, etc).

## Advantages

- Extra security!
- Possible for all APIs

# Identity and Access Management (IAM)

1. Create users
2. Apply the principle of least privilege
3. Factorize permissions with groups
4. Use conditional permissions for privileged accounts (time, IP adress, etc).
5. Enable Cloudtrail to log all API calls

## Advantages

- Keep a log of ALL activity inside your AWS account
- Useful for debugging
- Vital for forensics

# Identity and Access Management (IAM)

6. Use a strong password policy



## Advantages

- Do I really have to explain?

# Identity and Access Management (IAM)

6. Use a strong password policy
7. Rotate security credentials regularly



## Advantages

- Just in case one of your credentials leaked…

https://aws.amazon.com/fr/blogs/security/how-to-rotate-access-keys-for-iam-users/

# Identity and Access Management (IAM)

6. Use a strong password policy
7. Rotate security credentials regularly
8. Enable MFA for privileged users



## Advantages

- Vital for protection against phishing attacks

https://aws.amazon.com/fr/blogs/security/how-to-enable-mfa-protection-on-your-aws-api-calls/

# Identity and Access Management (IAM)

9. Use IAM roles to delegate permissions



## Advantages

- No need to store or share security credentials
- Use cases
  - Cross-account access
  - Federation

# Identity and Access Management (IAM)

9. Use IAM roles to delegate permissions
10. Use IAM roles for EC2 instances



## Avantages

- No need to store, share or rotate security credentials
- Application is granted least-privilege
- Integration with the AWS SDK and the AWS CLI

# Identity and Access Management (IAM)

9. Use IAM roles to delegate permissions
10. Use IAM roles for EC2 instances
11. Delete credentials for the root account



## Avantages

- C'mon, don't use 'root'

# 11 IAM Best Practices

1. Create users!
2. Apply the principle of least privilege
3. Factorize permissions in groups
4. Use conditional permissions for privileged accounts (time, IP adress, etc).
5. Enable Cloudtrail to log all API calls
6. Use a strong password policy
7. Rotate security credentials regularly
8. Enable MFA for privileged users
9. Use IAM roles to delegate permissions
10. Use IAM roles for EC2 instances
11. Delete credentials for the root account

# AWS account: one or many ?

Use a single AWS account when:

- You only need simple controls on who does what
- You don't need to isolate projects or teams
- You don't need to track costs separately

Use multiple accounts when:

- You need total isolation between projects or teams
- You need total isolation for some of your data (such as Cloudtrail logs)
- You want to keep track of costs separately (you can still get a single bill with Consolidated Billing)

# Log Everything

# Logs? Sure, we got logs!

## Infrastructure Logs

- AWS CloudTrail
- VPC Flow Logs

## Service Logs

- Amazon S3
- AWS Elastic Load Balancing
- Amazon CloudFront
- AWS Lambda
- AWS Elastic Beanstalk
- …

## Instance Logs

- UNIX / Windows logs
- NGINX/Apache/IIS
- Your own logs
- …

# CloudTrail

1. Enable Cloudtrail in all regions



## Advantages

- This takes 10 seconds
- It works for all regions, even if you don't use them yet.

# CloudTrail

1. Enable Cloudtrail in all regions
2. Enable log validation



## Advantages

- Guarantees log integrity
- Vital for audits and forensics
- Based on SHA-256 and RSA signature

# CloudTrail

1. Enable Cloudtrail in all regions
2. Enable log validation
3. Encrypt logs



## Advantages

- SSE-S3 by default
- KMS is supported too

# CloudTrail

1. Enable Cloudtrail in all regions
2. Enable log validation
3. Encrypt logs
4. Export logs to Cloudwatch Logs

## Advantages

- Easier to search
- Trigger alerts on specific events

# CloudTrail

1. Enable Cloudtrail in all regions
2. Enable log validation
3. Encrypt logs
4. Export logs to Cloudwatch Logs
5. Centralize logs in a single place

## Avantages

- Single bucket
- Could be in a dedicated account

# CloudTrail partners










https://aws.amazon.com/cloudtrail/partners/

# VPC Flow Logs

- Store all network traffic in Cloudwatch Logs
- They can be enable by VPC, by subnet our by network interface
- It's going to be a lot of data: what do you really need?
  - Everything, Allow, Deny
  - For debugging or for security monitoring?

# AWS Service Logs

Many services let you export their logs to CloudWatch Logs, CloudTrail ou S3.

- Elastic Beanstalk → CloudWatch Logs
  https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/AWSHowTo.cloudwatchlogs.html https://aws.amazon.com/fr/about-aws/whats-new/2016/12/aws-elastic-beanstalk-supports-application-version-lifecycle-management-and-cloudwatch-logs-streaming/
- ECS (instances & containers) → CloudWatch Logs http://docs.aws.amazon.com/AmazonECS/latest/developerguide/using_awslogs.html
- Lambda → CloudWatch Logs
  http://docs.aws.amazon.com/lambda/latest/dg/monitoring-functions-logs.html
- S3 → CloudTrail (S3 data events)
- CloudFront → S3
  http://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/AccessLogs.html
- ELB / ALB → S3
  http://docs.aws.amazon.com/elasticloadbalancing/latest/application/load-balancer-access-logs.html

# EC2 Logs

- You can export them to Cloudwatch Logs with the Cloudwatch Agent.
- Storage costs are pretty low, so it's probably worth it
- Available for Linux and Windows
- Logs can be exported to S3 and ElasticSearch
- Metrics and alarms allow you to keep track of suspicious events

# Automate Security Checks

# You can automate on multiple levels

- Infrastructure / application automation
  - AWS CloudFormation
  - AWS OpsWorks
- DIY automation
  - AWS CloudTrail → CloudWatch Logs → CloudWatch alerts
  - API calls → Amazon CloudWatch Events → SNS / SQS / Kinesis / Lambda
- Compliance automation
  - AWS Inspector
  - AWS Config Rules

# Configuring CloudWatch alarms for CloudTrail

- CloudFormation template with 10 predefined alarms
  - Create, modify or delete Security Groups
  - Modify IAM policies
  - Failed connections to the console
  - Failed API calls caused by permission issues
- Set them up in less than 5 minutes!
- Get e-mail notifications when these events occur in your AWS account

http://docs.aws.amazon.com/fr_fr/awscloudtrail/latest/userguide/use-cloudformation-template-to-create-cloudwatch-alarms.html
https://s3-us-west-2.amazonaws.com/awscloudtrail/cloudwatch-alarms-for-cloudtrail-api-activity/CloudWatch_Alarms_for_CloudTrail_API_Activity.json

# AWS Config Rules

- Config Rules checks that AWS resources are compliant
- You can use:
  - Pre-defined rules: MFA on, CloudTrail on, EBS encryption, etc.
  - Your own rules
- Checks can be:
  - Periodic (1, 3, 6, 12 or 24 hours)
  - Triggered by configuration changes
- Notifications are sent to SNS…
- ... Which means that you can process them with Lambda functions
  - Non-compliant instance? Kill it!

# Amazon Inspector

- This service allows you to check the configuration and the behavior of EC2 instances.
- Agent-based
- Can run from 15  minutes to 24 hours
- Reports and advice on how to fix issues
- Can be automated with the AWS API
- Built-in rule packages

# Amazon Inspector – Rule packages

- ***Common Vulnerabilities and Exposures***
  - http://cve.mitre.org
  - https://s3-us-west-2.amazonaws.com/rules-engine/CVEList.txt (47,050)
- ***CIS Operating System Security Configuration Benchmarks***
  - *Center for Internet Security* http://cisecurity.org
  - http://benchmarks.cisecurity.org
- ***Security Best Practices***
  - SSH, passwords, etc. (Linux uniquement)
  - https://docs.aws.amazon.com/fr_fr/inspector/latest/userguide/inspector_security-best-practices.html
- ***Runtime Behavior Analysis***
  - How instances behave during testing (networking, protocols, etc.)
  - https://docs.aws.amazon.com/fr_fr/inspector/latest/userguide/inspector_runtime-behavior-analysis.html

# AWS Trusted Advisor

## Recommended Actions

**Security Groups - Specific Ports Unrestricted** *Updated: 9/29/14 7:19 AM*

Checks security groups for rules that allow unrestricted access (0.0.0.0/0) to specific ports.

2 of 71 security group rules allow unrestricted access to a specific port.

**IAM Use** *Updated: 9/17/14 12:39 PM*

Checks for your use of AWS Identity and Access Management (IAM).

At least one IAM user, group, or role has been created for this account.

**MFA on Root Account** *Updated: 9/17/14 12:39 PM*

Checks the root account and warns if multi-factor authentication (MFA) is not enabled.

MFA is enabled on the root account.

**Service Limits** *Updated: 9/17/14 12:39 PM*

Checks for usage that is more than 80% of the service limit.

0 of 42 items have usage that is more than 80% of the service limit.


amazon web services

# Please promise me this

- Never share credentials across users / applications
- Never store credentials in source code (they'll end up on Github)
- Never store credentials on EC2 instances
- (Almost) never work with the root account
- One account per user / one role per app with least privilege
- Use MFA for privileged accounts
- Enable CloudTrail in all regions
- Encrypt everything
- Automate security checks and alarms

**"It's not because you're paranoid that they're not after you"**

# Additional resources

Whitepapers
https://d0.awsstatic.com/whitepapers/aws-security-whitepaper.pdf
http://d0.awsstatic.com/whitepapers/architecture/AWS_Well-Architected_Framework.pdf
https://d0.awsstatic.com/whitepapers/Security/AWS_Security_Best_Practices.pdf

AWS re:Invent 2016: Security Services State of the Union (SEC312) - Steve Schmidt, CISO, Amazon Web Services
https://www.youtube.com/watch?v=8ZljcKn8FPA

AWS re:Invent 2016: Automating Security Event Response, from Idea to Code to Execution (SEC313)
https://www.youtube.com/watch?v=x4GkAGe65vE

AWS re:Invent 2016: Automated Governance of Your AWS Resources (DEV302)
https://www.youtube.com/watch?v=2P2I7HlrFtA

AWS re:Invent 2016: Scaling Security Operations and Automating Governance (SAC315)
https://www.youtube.com/watch?v=_yfeCvqHdNg

# Thank You

Julien Simon
julsimon@amazon.fr
@julsimon

**Your feedback is important to us!**

amazon web services

Pop-up Loft
TEL AVIV