AMAZON WEB SERVICES

LARGE SCALE DATA MANAGEMENT

CLOUD COMPUTING AND BIG DATA ECOSYSTEMS DESIGN

NUEVAS TENDENCIAS EN SISTEMAS DISTRIBUIDOS

Table of Contents

Bibliography

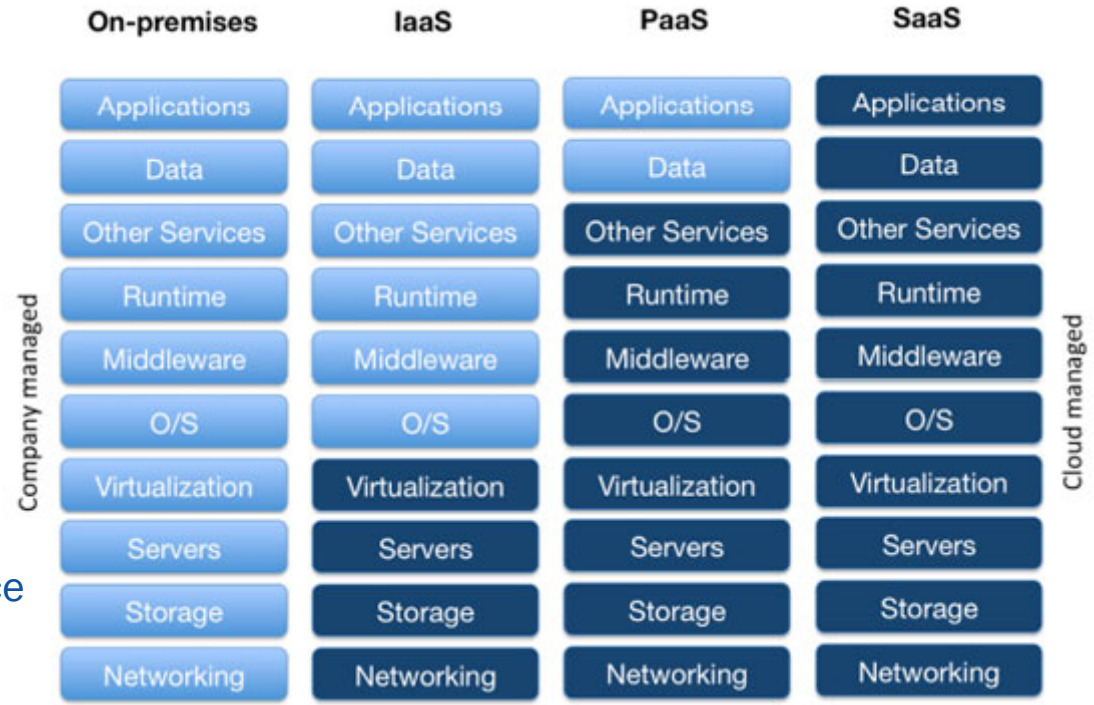
- AWS Documentation. https://aws.amazon.com/documentation/
- Amazon Web Services AWS LiveLessons, Richard Jones, Published by Addison-Wesley Professional (2016)
- Cloud Computing: Theory and Practice. Dan C. Marinescu, Ed. Morgan Kaufmann (2013)
- Cloud Computing: Concepts, Technology & Architecture. Thomas Erl, Ricardo Puttine, Zaigham Mahmood. Prentice Hall (2013)
- Programming Amazon EC2. J. van Vliet; F. Paganelli. O'Reilly (2011)
- Programming Amazon Web Services. J. Murty. O'Reilly (2008)

AWS OVERVIEW

Overview

What is “cloud computing”?

- Main characteristics
  - On-demand
  - Broad Network Access
  - Resource Pooling
  - Rapid Elasticity
  - Measured Service
- Service Models
  - **IaaS** – Infrastructure as a Service
  - **PaaS** – Platform as a Service
  - **SaaS** – Software as a Service



Global Infrastructure

Regions and Availability Zones

- The AWS Cloud infrastructure is built around **Regions** and **Availability Zones** ("AZs").
  - A **Region** is a physical location in the world where AWS has multiple Availability Zones.
  - The AWS Cloud operates 43 Availability Zones within **16 geographic Regions** around the world, with announced plans for 11 more Availability Zones and 4 more Regions.

Global Infrastructure

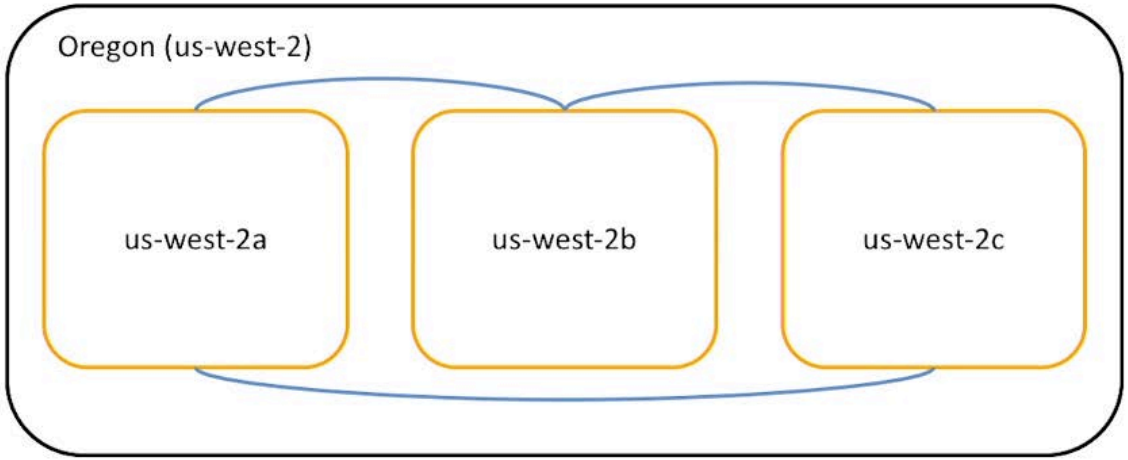
Regions and Availability Zones

- Regions are the **primary building block** of Amazon Web Services
  - When you go to launch resources, such as virtual machines, storage, load balancers, databases… you generally will pick a region
- Regions are chosen:
  - Based on **latency to your end users**
  - Based on **cost**
  - Based on **security compliance**
- Large sites such as Netflix, serving a global audience, choose **multiple regions**
  - Very common practice for disaster recovery
  - Allows you to serve different audiences, US audience vs. European audience

Global Infrastructure

Regions and Availability Zones

- **Availability Zones** consist of one or more discrete data centers, each with redundant power, networking and connectivity, housed in separate facilities.
  - They are connected with private fiber.
  - These Availability Zones offer you the ability to operate production applications and databases which are more **highly available**, **fault tolerant** and **scalable** than would be possible from a single data center.



Global Infrastructure

Regions and Availability Zones

- **High Availability Through Multiple Availability Zones**
  - Deploy applications across multiple Availability Zones in the same region for fault tolerance and low latency.
  - Availability Zones are connected to each other with fast, private fiber-optic networking, enabling you to easily architect applications that automatically fail-over between Availability Zones without interruption.
- **Improving Continuity With Replication Between Regions**
  - Increase redundancy and fault tolerance by replicating data between geographic Regions
- **Geographic Expansion**
  - AWS plans to expand with 11 new Availability Zones in four new geographic Regions: Hong Kong, Ningxia (China), Paris, and Stockholm

Global Infrastructure

Edge Locations

- To deliver content to end users with lower latency, AWS provides a global network of 56 *edge locations* and 11 regional edge caches across 21 countries and 48 cities for content delivery
- An edge location is where end users access services located at AWS.
- They are located in most of the major cities around the world and are specifically used by CloudFront (CDN) to distribute content to end user to reduce latency and by Route 53, the AWS's global DNS service.

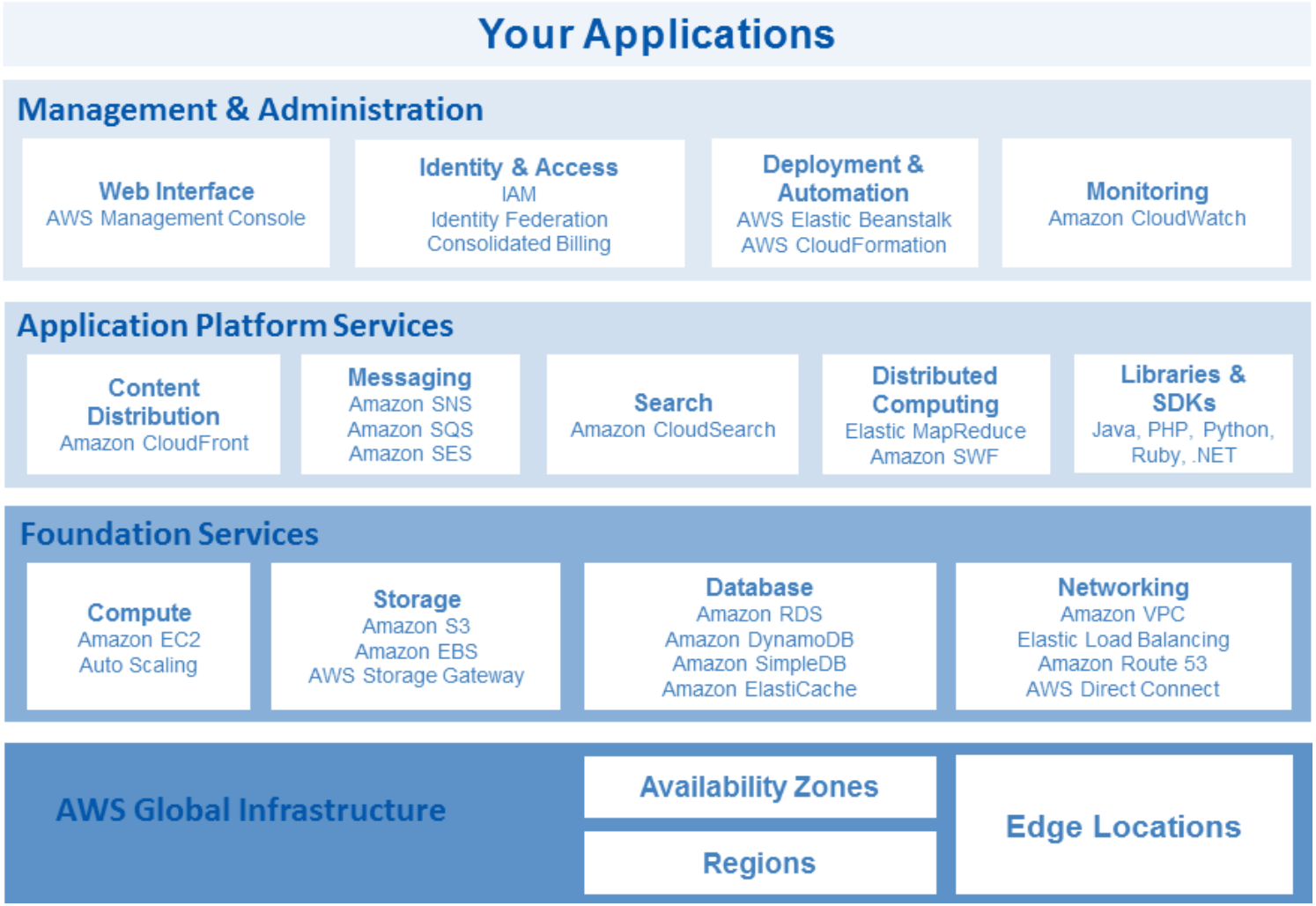
Global Infrastructure

Scope of Services

- **Global**: don't need to choose a particular region o AZ to use them.
  - AWS IAM
  - Amazon CloudFront
  - Amazon Route53
- **Regional**: we need to choose the region, and that particular service is inherently highly available and fault tolerant and durable.
  - Amazon DynamoDB
  - Amazon Simple Storage Service
  - Elastic Load Balancing
  - Amazon Virtual Private Cloud
- **Availability Zone**: we have to choose an AZ
  - Amazon Elastic Block Store
  - Amazon Elastic Compute Cloud
  - Subnets

Services overview

AWS Architecture



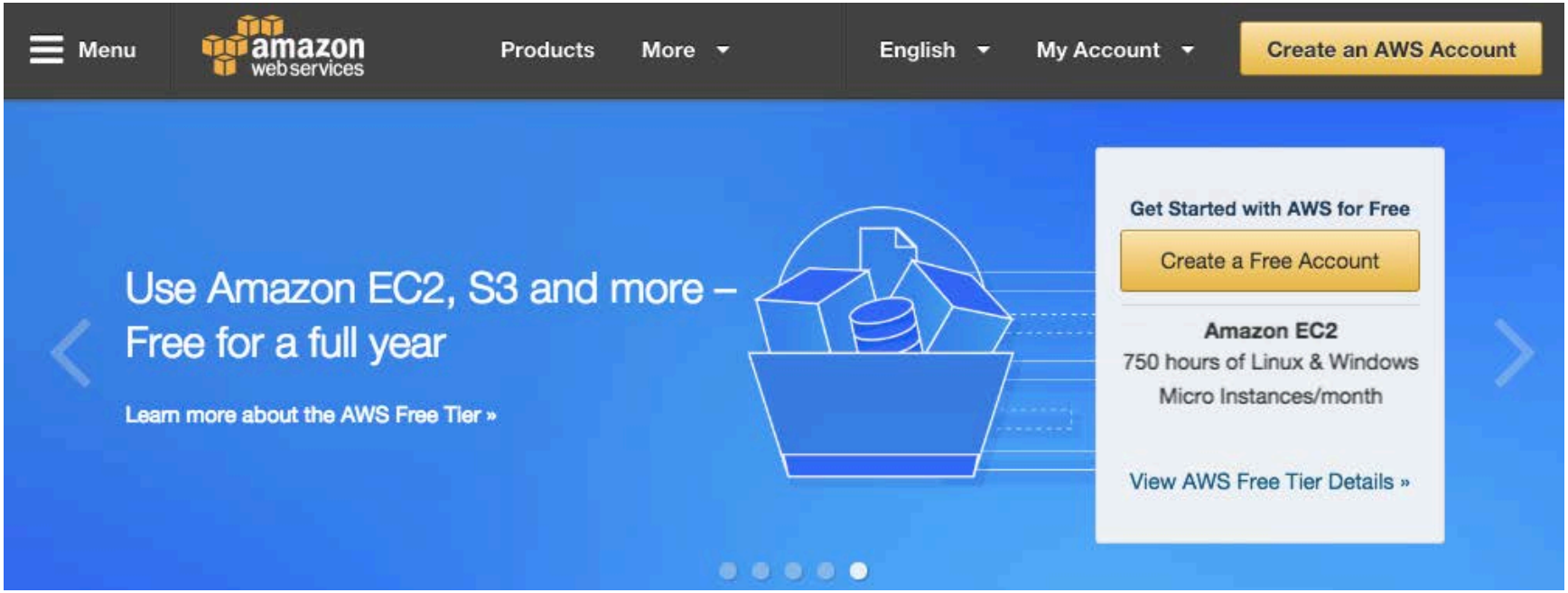
Services Overview

- **Compute**
  - Amazon EC2
  - Amazon VPC
  - AWS Batch
  - AWS Elastic Beanstalk
  - AWS Lambda
  - Auto Scaling
  - Elastic Load Balancing
- **Storage**
  - Amazon S3
  - Amazon EBS
  - Amazon EFS
  - Amazon Glacier
  - AWS Snowball
  - AWS Storage Gateway
- **Database**
  - Amazon RDS
  - Amazon DynamoDB
  - Amazon ElastiCache
  - Amazon Redshift
- **Networking & Content Delivery**
  - Amazon VPC
  - Amazon CloudFront
  - AWS Direct Connect
  - Elastic Load Balancing
  - Amazon Route 53
- **Management Tools**
  - Amazon CloudWatch
  - AWS CloudFormation
  - AWS Health
  - AWS Management Console
  - AWS Command Line Interface
- **Security, Identity, & Compliance**
  - Identity & Access Management
  - AWS Artifact
  - AWS Certificate Manager
  - AWS Directory Service
  - AWS Organizations
- **Analytics**
  - Amazon EMR
  - Amazon Kinesis
  - Amazon Redshift
  - AWS Data Pipeline
- **Artificial Intelligence**
  - Amazon Machine Learning
- **Mobile Services**
  - Amazon Cognito
  - Amazon Mobile Analytics
  - Amazon SNS
- **Messaging**
  - Amazon SNS
  - Amazon SES
  - Amazon SQS

... And many, many more

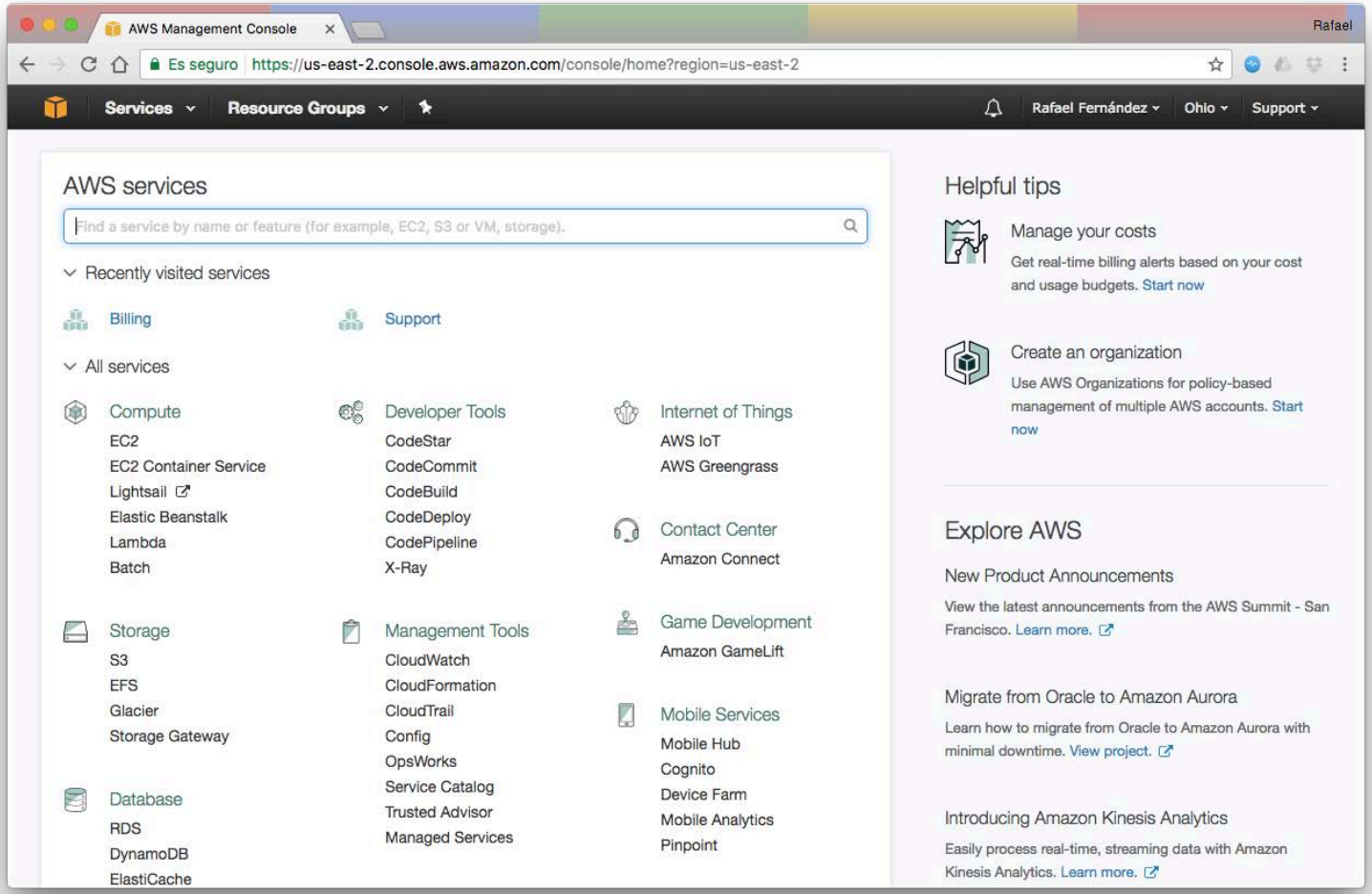
Create an AWS Account

AWS Free Tier



Tools for AWS

AWS Management Console



Tools for AWS

AWS Command Line Interface (CLI)

- Provides commands for a broad set of AWS products
- Is supported on Windows, Mac, and Linux/Unix

```
$ aws s3 cp myfolder s3://mybucket/myfolder –recursive

upload: myfolder/file1.txt to s3://mybucket/myfolder/file1.txt
upload: myfolder/subfolder/file1.txt to s3://mybucket/myfolder/subfolder/file1.txt
…
```

Tools for AWS

AWS Application Programming Interfaces (APIs)

- AWS architecture is designed to be programming language-neutral, using the supported interfaces to access AWS resources
- REST API: an HTTP interface
  - Use standard HTTP requests to access AWS resources
  - Use standard HTTP headers and status codes
- SOAP API:
  - Provides a SOAP 1.1 interface using document literal encoding
  - The most common way to use SOAP is to download the WSDL
  - SOAP support over HTTP is deprecated, but it is still available over HTTPS

Tools for AWS

AWS Software Development Kits (SDKs)

- AWS SDKs:
  - Build applications using language-specific APIs
  - Provides libraries for software developers
    - Java
    - JavaScript
    - .NET
    - PHP
    - Python
    - Ruby
    - ...
- AWS Mobile SDK:
  - Android
  - iOS
- AWS Toolkit
  - Eclipse
  - Visual Studio

SECURITY IN AWS

AWS Identity and Access Management

Introduction

- Authentication
  - Who the user is
- Authorization
  - What users are allowed to do
- Users
- Groups
- Password Policy
  - It is important to set a strong password policy
- Multifactor Authentication
  - Something you have → a MFA device
  - Something you know → your username and your password
  - Something you are → biometrics: fingerprints, iris scan, …

You have to fulfill at least two of those!

- **This is meant for authenticating and authorizing users and groups of users against the Amazon Web Services API (includes CLI, SDK and Management Console), it is not to be used for OS level authentication, or application or database level authentication**

Security in AWS

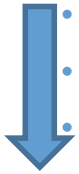
AWS Security Credentials

AWS tools	Security credentials	Extra security
AWS Management Console	Email address and password	Multi-Factor Authentication
	IAM user name and password	Multi-Factor Authentication
AWS CLI	Access keys	Temporary security credentials
AWS API		
AWS SDKs		
Access EC2 Linux instances	Key pairs	

AWS Identity and Access Management

Permissions and Policies

- User accounts, by default, does not allow your user to do anything
  - Permissions (authorization) granted by means of *policies*
- Policies are written in JSON
- There are two different types of policies:
  - Managed Policy: top level first class objects, can be directly created and managed apart from other resources.
    - AWS managed (created and managed by AWS)
    - Customer managed (we create and manage on our own)
  - Inline Policy (you put the policy directly to the user's account rather than referencing a managed policy)
- Policies can be created via
  - A Generator AWS provides
  - Hand written policies (be careful, copy & shit)
- Policies are logically evaluated:
  - Defaults to implicit deny. You can't do anything unless you have an explicit allow
  - Explicit deny. It can't be overridden. Explicit deny will always win.
  - Explicit allow



Rafael Fernández (rfernandez@fi.upm.es)

AWS Identity and Access Management

Amazon Resource Name (ARN)

- ARN is a way of uniquely identifying a particular resource across all of Amazon
- Format pattern:
  - `arn:partition:service:region:account-id:resourceType/resource`
  - `arn:partition:service:region:account-id:resourceType:resource`

- Example ARNs:

  - `arn:aws:iam::123456789012:user/rfernandez`
    - No region is necessary for global services

  - `arn:aws:s3:::mybucket/*`
    - No account id is necessary for S3 buckets

  - `arn:aws:rds:us-west-2:1232456789012:db:mysql-db`

AWS Identity and Access Management

Example policy

```
{
  “Version”: “2012-10-17”,
  “Statement”: [
    {
      “Effect”: “Allow”,
      “Action”: [
        “s3:GetObject”,
        “s3:PutObject”
      ],
      “Resource”: [
        “arn:aws:s3:::mybucket/*”
      ]
    }
  ]
}
```

Array (Collection)

Allow or *Deny*

Collection of *actions*

Collection of *resources*

As ARNs

Rafael Fernández (rfernandez@fi.upm.es)

AWS Identity and Access Management

Permissions and Policies

- Policies are just collections of statements
- It is possible for a user or group of users to have multiple policies attached.
- Statements specify
  - The who (in the case of resource-based policies, otherwise it could be implicit)
  - Actions
    - ec2:RunInstances
    - s3:ListBucket
  - Resources
    - EC2 instances
    - S3 buckets (or objects)
  - Conditions
    - Time of day
    - From specific IP address
    - Resource contains particular tag
    - Etc

AWS Identity and Access Management

Resource Policies

- Policies can also be applied to a resource, such as:
  - Amazon Simple Storage Service (S3) bucket
  - Amazon DynamoDB table
  - Amazon Simple Queue Service (SQS) queue
- Written in JSON
- Can provide cross-account resource sharing
  - If we have an S3 bucket in one account we can allow someone else's account to leverage that bucket if we want to
- Can allow anonymous use

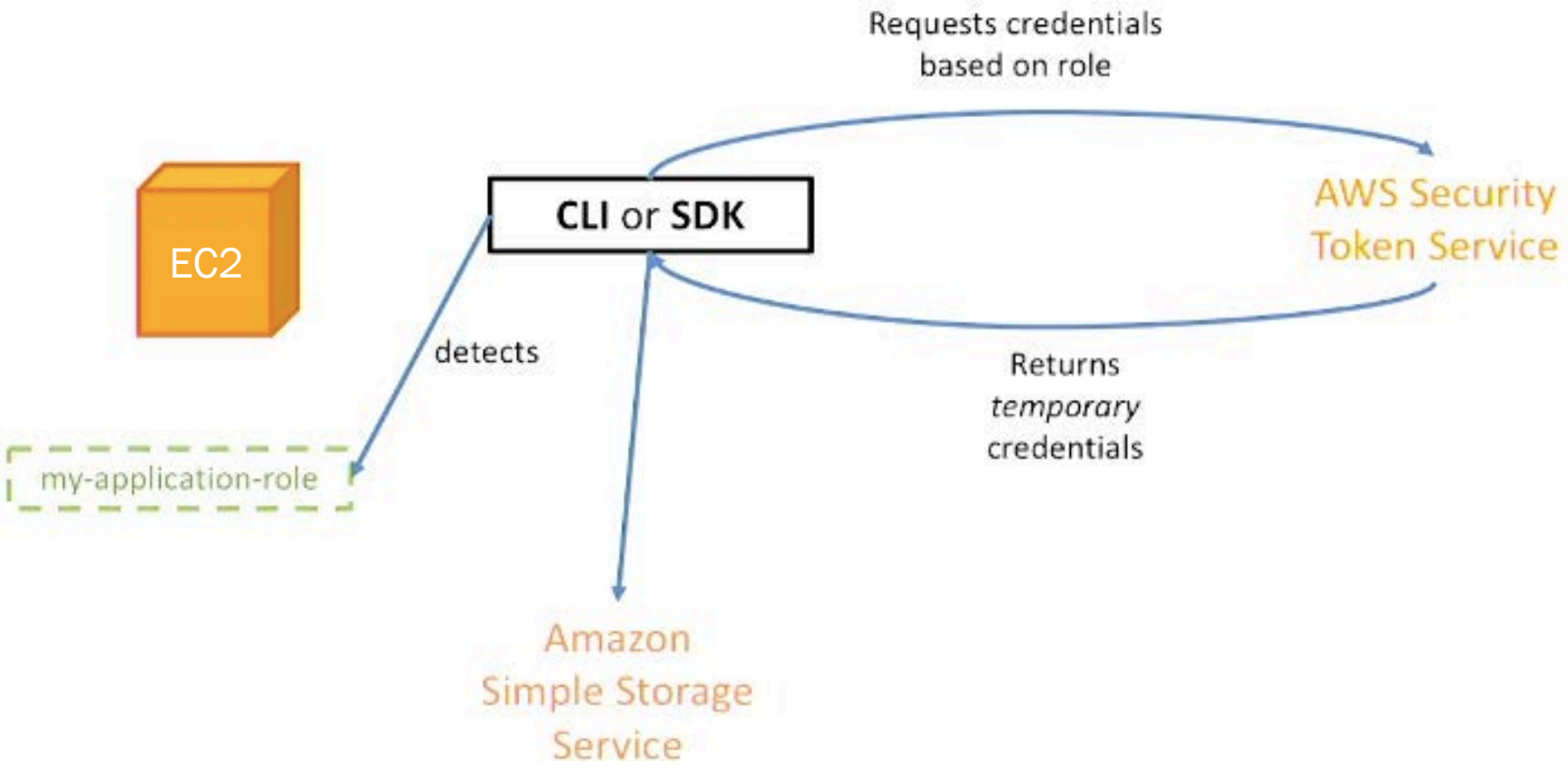
AWS Identity and Access Management

Roles

- Roles can be thought of as an alternate form of authentication
- They provide us with “temporary credentials”
- What we want is a way to authenticate and authorize a user, but without having to give them, or without having to embed or share some long term credentials
- Allow access from:
  - EC2 instance
  - AWS service
  - A user
  - Separate account
    - One you own
    - Third party
- What not to do:
  - Embed access keys in code
  - Embed in environment variables
  - Share with:
    - Third parties
    - Hundreds of enterprise users
    - Thousands of web users

AWS Identity and Access Management

An EC2 Role Example



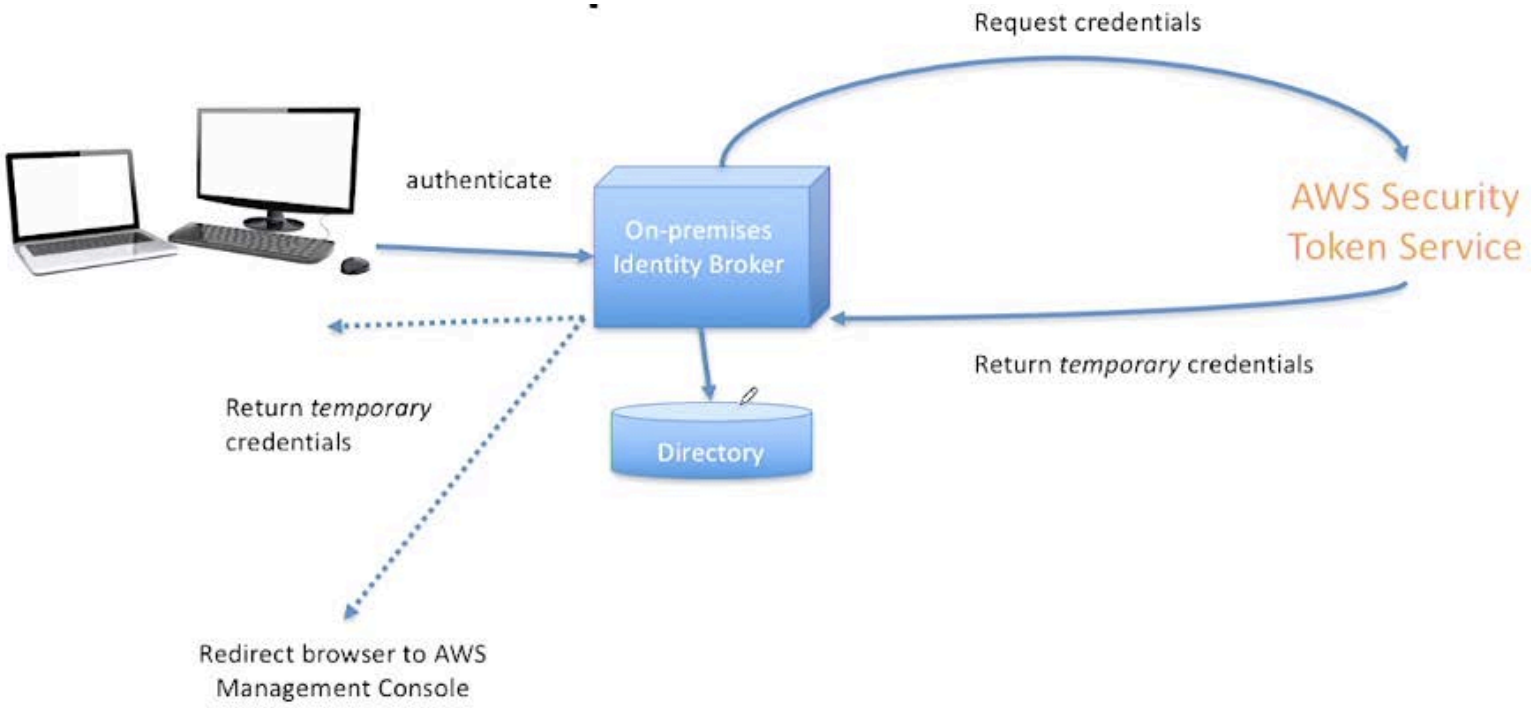
AWS Identity and Access Management

Federated Users

- In large organizations there are dozens, hundreds or thousands of users:
  - Leverage existing user directories:
    - LDAP
    - Active Directory
  - Federated users means that you are taking someone who you already know and giving them temporary access into AWS based on temporary credentials.
  - We can create a Single sign-on service
- We can also federate Web/mobile application users:
  - Apps bypass backend APIs/proxies and allow those mobile applications to go directly to AWS services

AWS Identity and Access Management

Federated Users



AWS Identity and Access Management

Best Practices

Read Security Whitepaper

https://aws.amazon.com/whitepapers

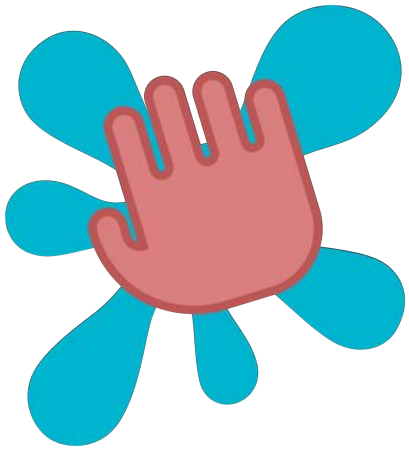
- Leverage groups
  - Especially when we have a large number of users
- Grant least privilege
  - Give a user enough permission to do what they need to do, but not so much that they can be dangerous
- Implement strong password policy
- Leverage roles for cross-account access
- Use deny statements for added security
- Never share your credentials
  - Never email, print, or publish to code repos
- Leverage Multiple accounts for isolation

- **Protect master account (root/administrator) at all costs**
  - Never use it for day-to-day
  - Delete default access keys
  - Enable MFA (with physical key fob)
  - Lock key fob in safe

AWS Identity and Access Management

Hands-on activity

- Let's make a demo to see how to:
  - Create User and group
  - Create Access Keys
  - Create and attach Policies
  - Create Roles
  - Managing an MFA Device
  - Applying Resource Policy



hands-on
TRAINING

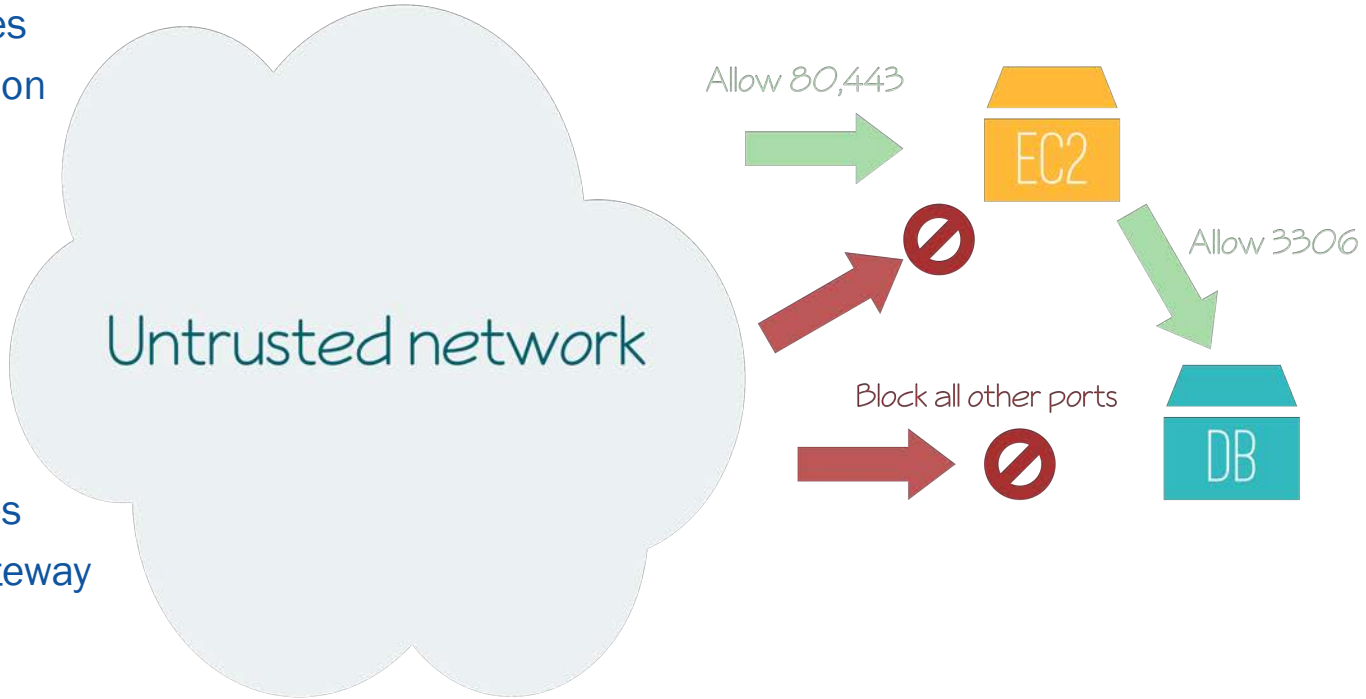
Rafael Fernández (rfernandez@fi.upm.es)

AMAZON VIRTUAL PRIVATE CLOUD

Networking

Introduction to Amazon Virtual Private Cloud (VPC)

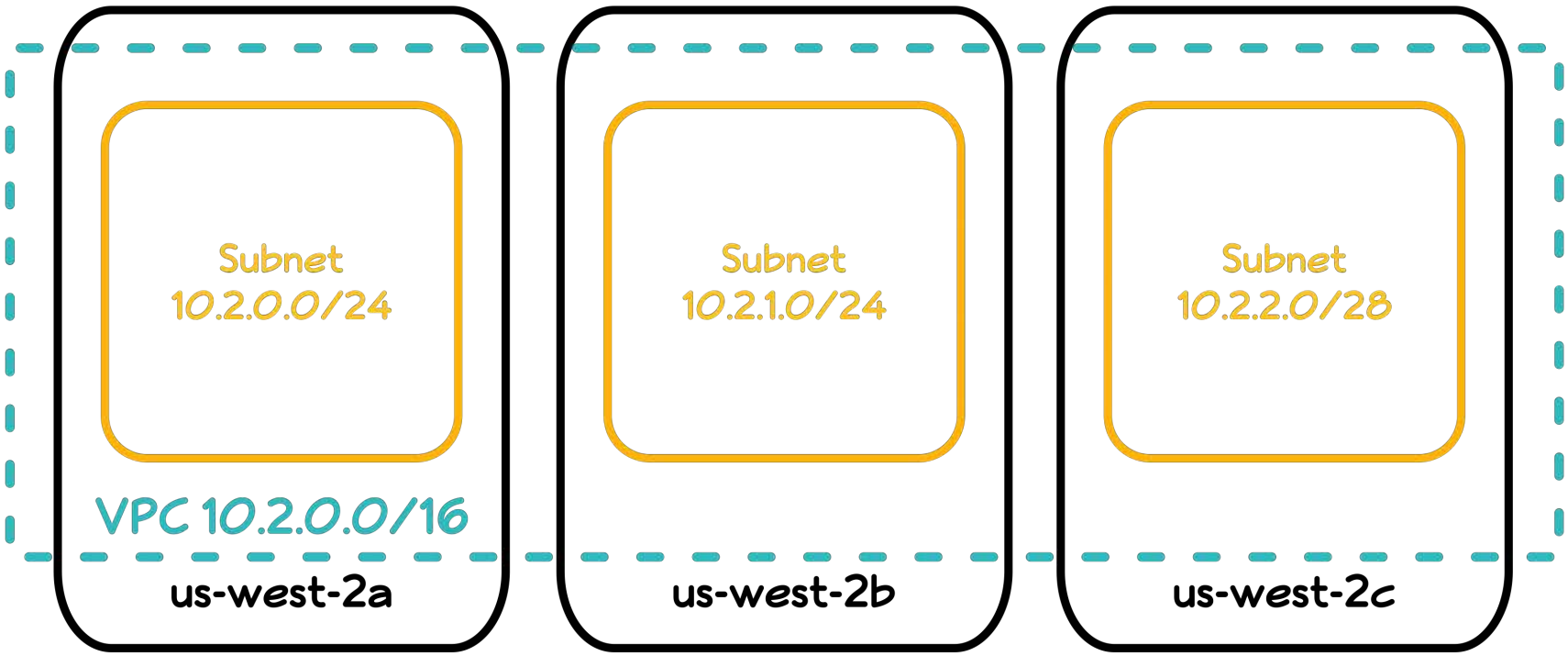
- **Virtual Private Cloud** (VPC) is a virtual network that closely resembles a traditional network that you'd operate in your own data center, with the benefits of using the scalable infrastructure of AWS
- Applications generally need:
  - IP addresses
  - Segmentation
    - Public
    - Private
  - Firewalls
- Concepts:
  - VPC
  - Subnet
  - Route tables
  - Internet gateway
  - Security



Rafael Fernández (rfernandez@fi.upm.es)

Networking

Introduction to Amazon Virtual Private Cloud (VPC)



Rafael Fernández (rfernandez@fi.upm.es)

Networking

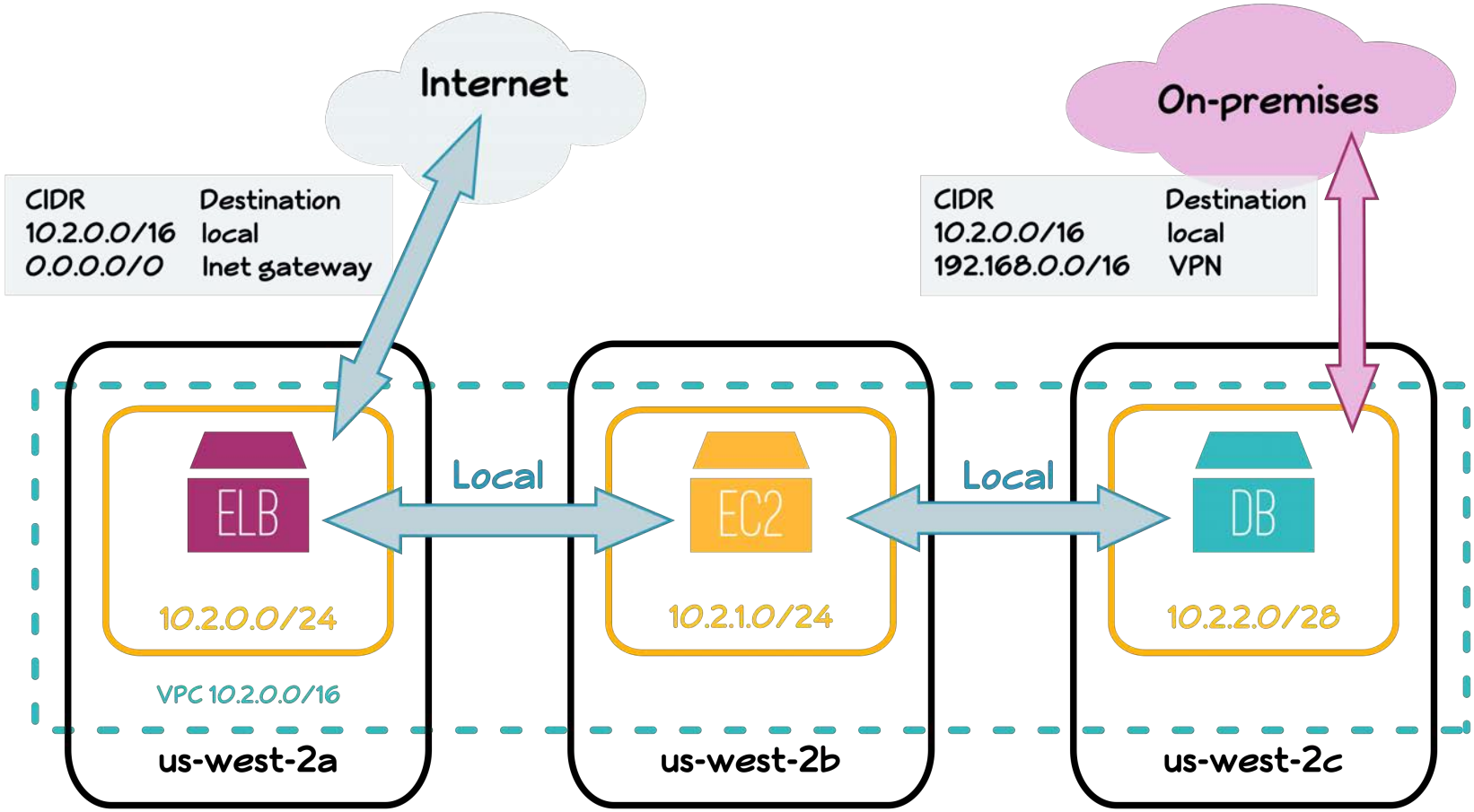
Introduction to Amazon Virtual Private Cloud (VPC)

- VPC: virtual network dedicated to an AWS account. It is logically isolated from other virtual networks in the AWS cloud
  - To select its IP address range, create subnets, and configure route tables, network gateways, and security settings
  - Can span multiple Availability Zones
  - Single CIDR block size between a /28 and /16 netmask

- Subnet: range of IP addresses in your VPC
  - Can launch AWS resources into a subnet
    - Public subnet for resources that must be connected to the Internet
    - Private subnet for resources that won't be connected to the Internet
  - Multiple layers of security to protect the AWS resources in each subnet
    - Security groups
    - Network access control lists (ACL)
  - Must reside entirely within one Availability Zone and cannot span zones

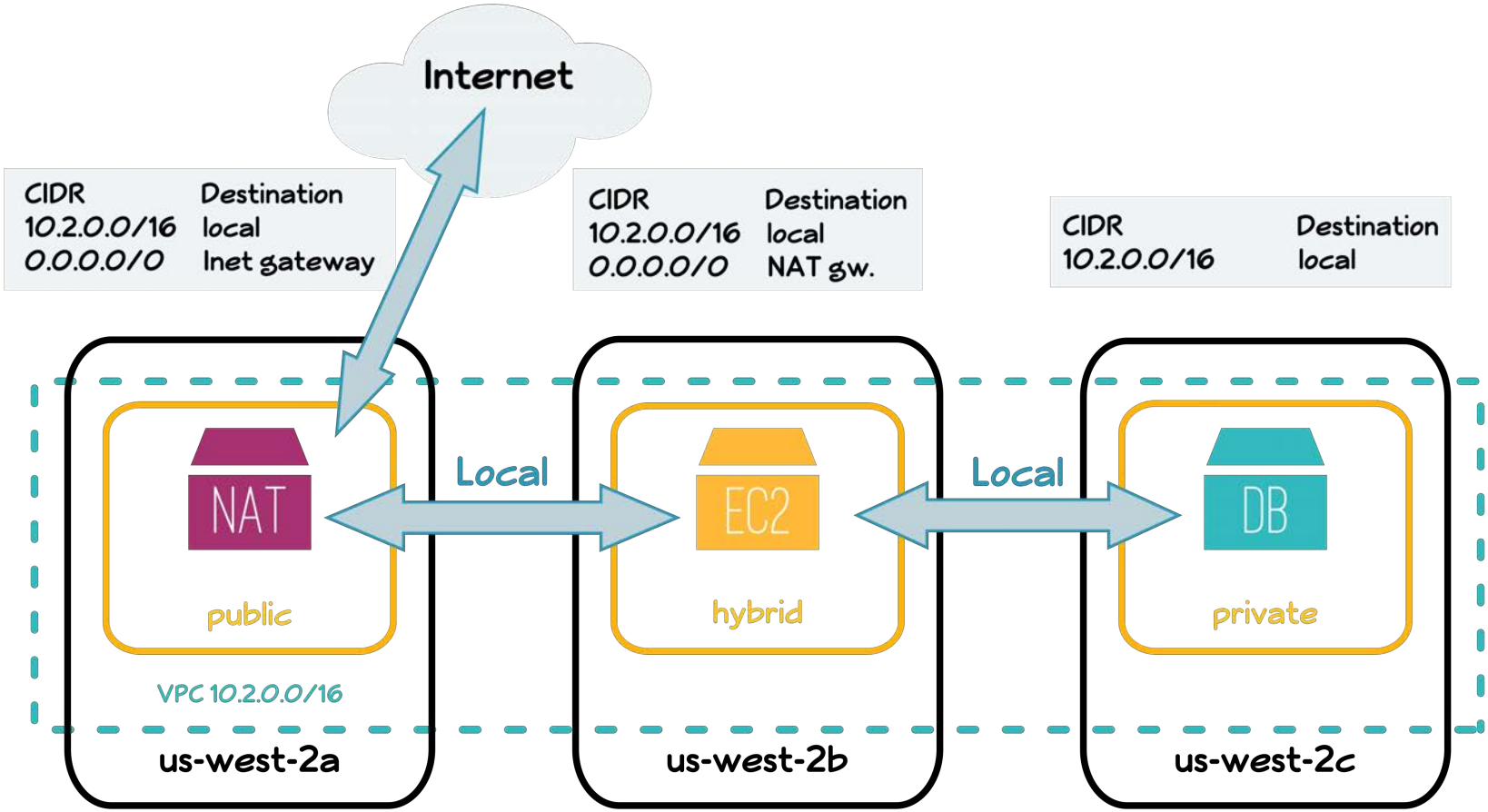
Networking

Routing



Networking

Public, Private, and Hybrid Subnets



Networking

Security: Network Access Control Lists (NACL) and Security Groups

- **Security:**
  - Control inbound and outbound traffic of instances
    - **Network ACLs**: firewall at network level
    - **Security Groups**: firewall at instance level
  - By design,
    - each subnet must be associated with a network ACL
    - each instance must be associated with a security group
- **Network Access Control Lists (NACLs)**
  - Applied to **subnet** as a whole
  - **Stateless**
  - Must specify ingress and egress
  - Allow or Deny (default)
  - Moreover, they have to specify:
    - Protocol,
    - Source IP range
    - Destination port range

Ingress rules

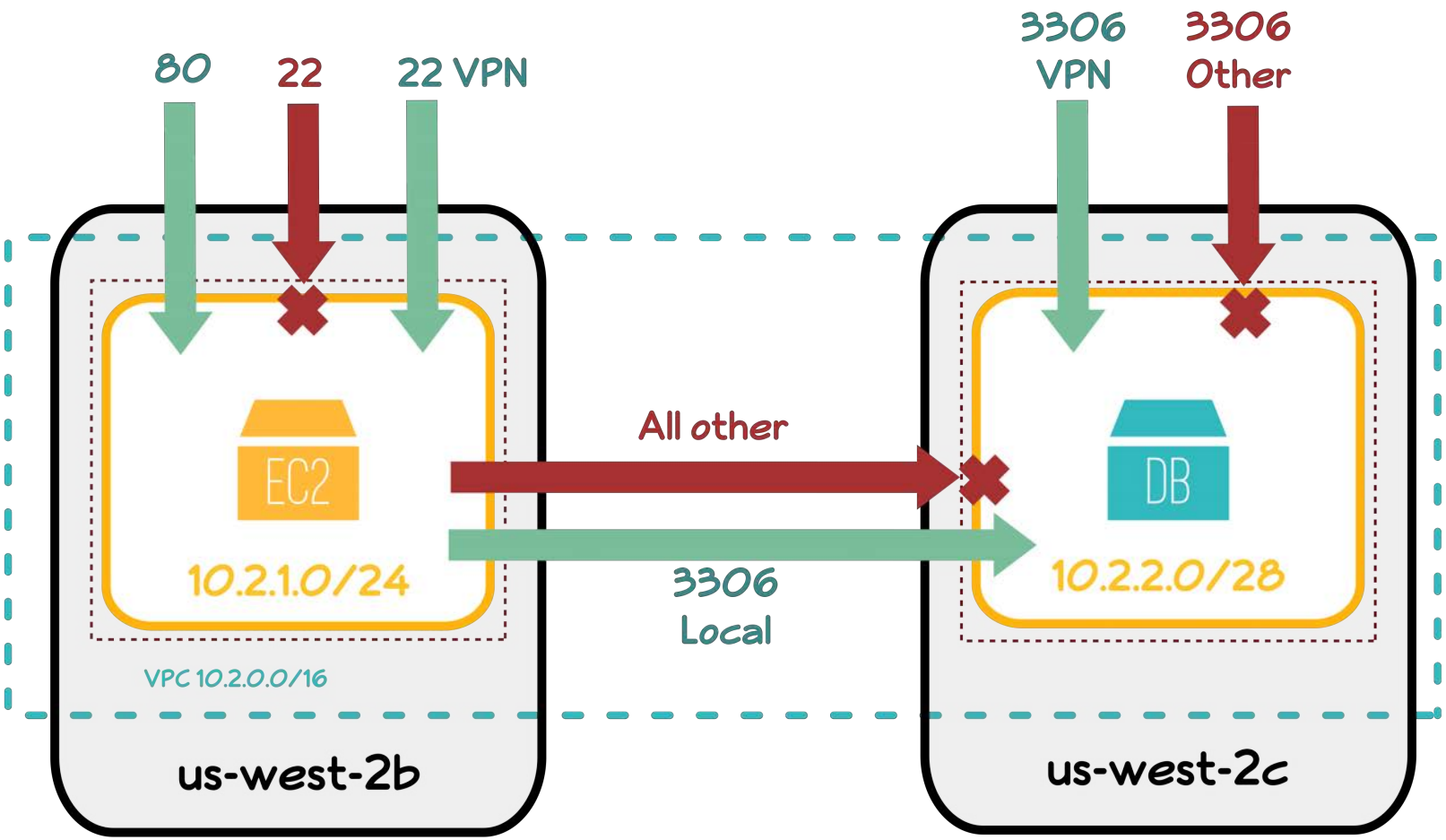
Rule #	Type	Protocol	Port Range	Source	Allow/Deny
100	HTTP	TCP(6)	80	0.0.0.0/0	ALLOW
101	HTTPS	TCP(6)	443	0.0.0.0/0	ALLOW
110	SSH	TCP(6)	22	192.168.0.0/16	ALLOW
*		ALL	ALL	0.0.0.0/0	DENY

Egress rules

Rule #	Type	Protocol	Port Range	Destination	Allow/Deny
100	Custom	TCP(6)	1024-65535	0.0.0.0/0	ALLOW
*		ALL	ALL	0.0.0.0/0	DENY

Networking

Network Access Control Lists (NACL)



Networking

Security Groups

- Security Groups:
  - Applied to instances
  - **Stateful**
  - Can specify ingress or egress
    - **Egress rules apply to initiating connections, not responses**
  - Denied by default
  - They also need:
    - Protocol
    - Source IP range
    - Destination port range

Ingress rules

Type	Protocol	Port Range	Source
HTTP	TCP(6)	80	0.0.0.0/0
HTTPS	TCP(6)	443	0.0.0.0/0
Custom	TCP(6)	9999	sg-ihgfdcba
SSH	TCP(6)	22	192.168.0.0/16

Reference

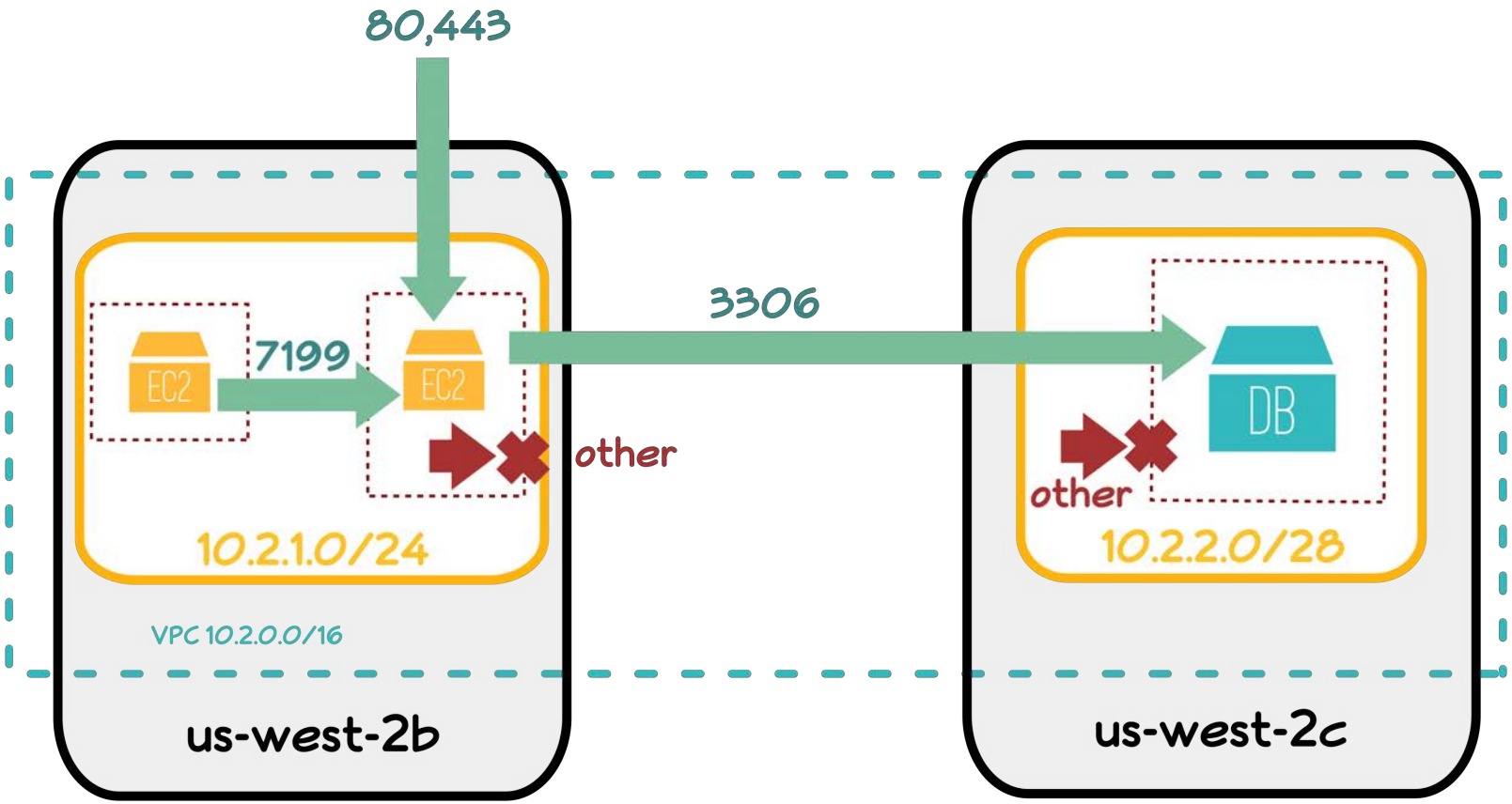
Egress rules

Type	Protocol	Port Range	Destination
MySQL	TCP(6)	3306	sg-abcdfhi

Rafael Fernández (rfernandez@fi.upm.es)

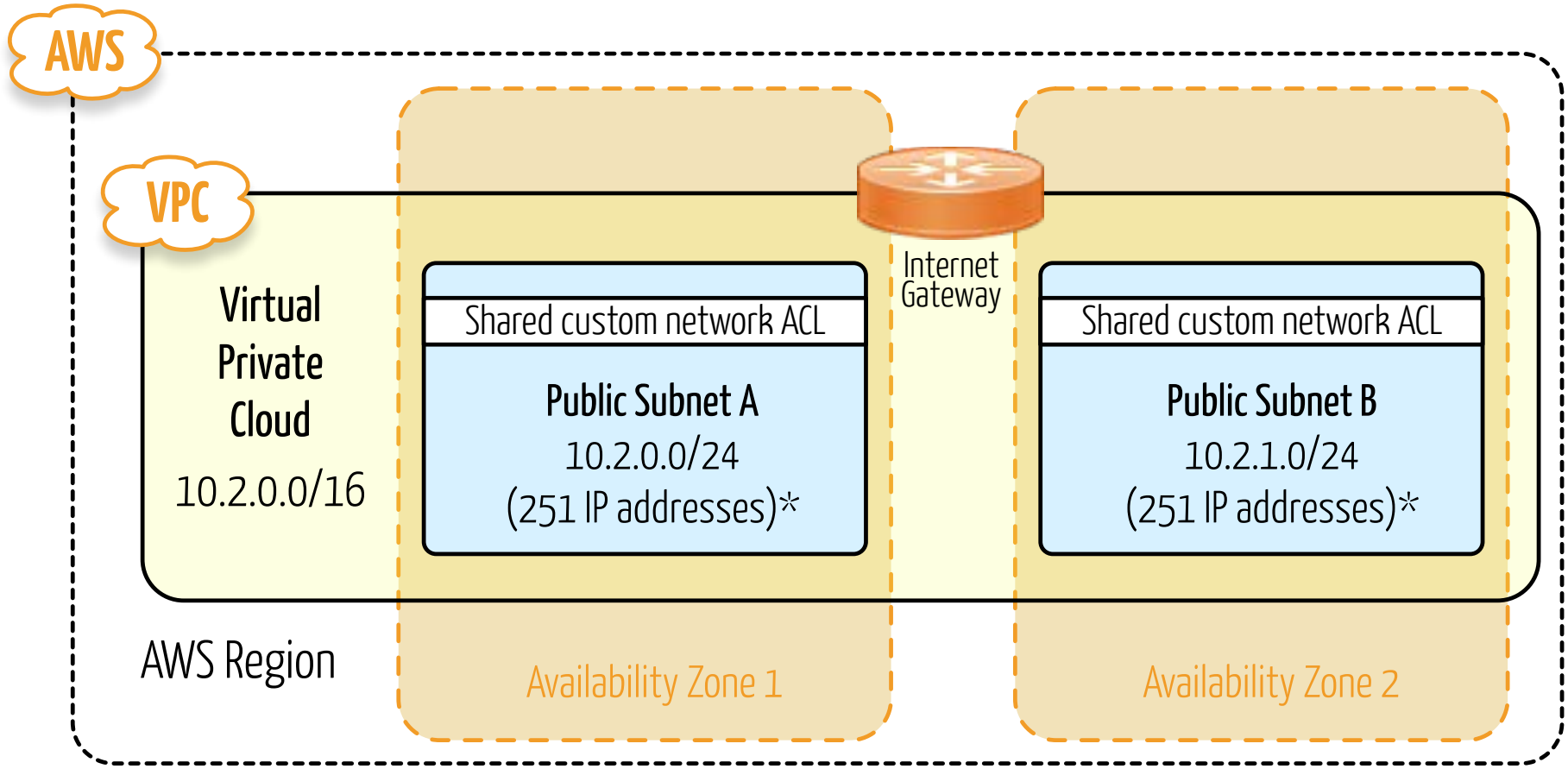
Networking

Security Groups



Networking

Exercise

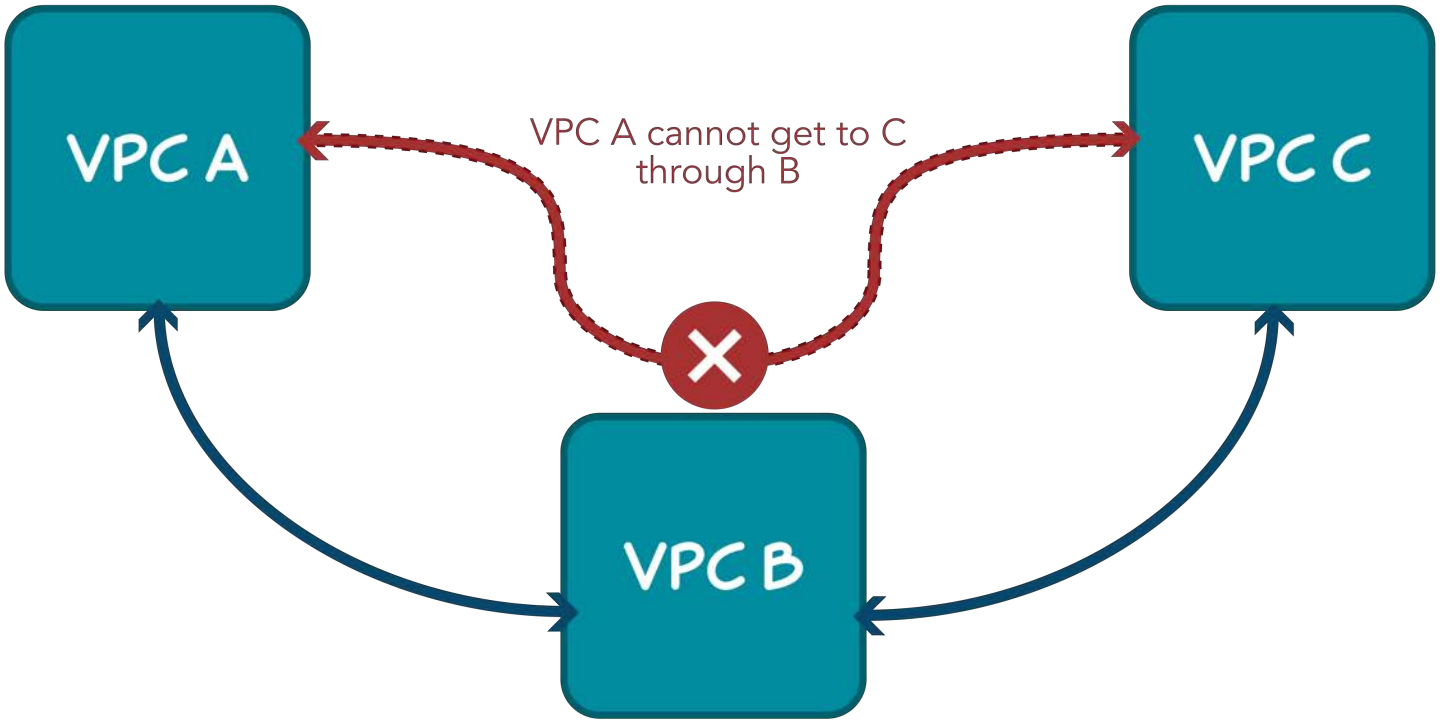


- NACL: Create rules to allow HTTP, HTTPS and SSH connections

Networking

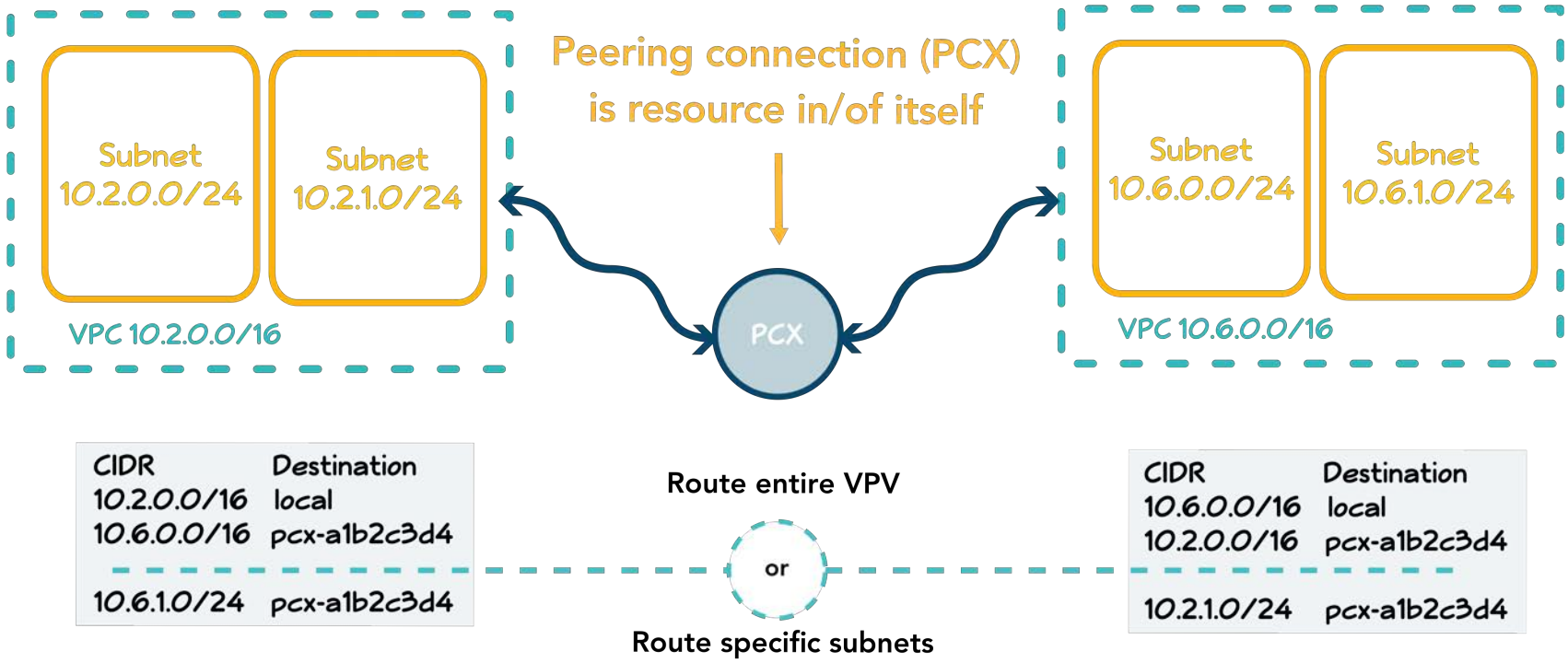
VPC Peering

- Connection between two VPCs
- Helps segregate networks
- No bottleneck or SPOF
- Can peer between accounts
- IP ranges must not overlap
- Always one-to-one
- No transitive peering
- No edge-to-edge routing



Networking

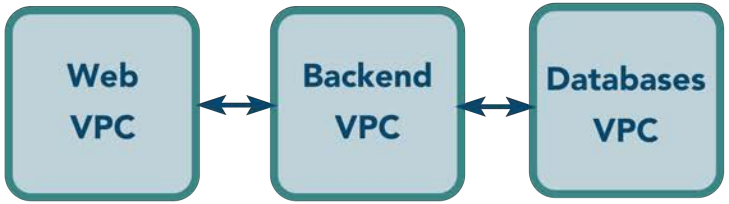
Peering two VPCs



Networking

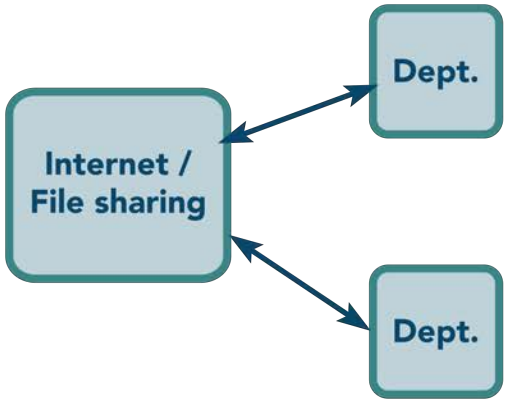
VPC Peering Scenarios

Isolating services

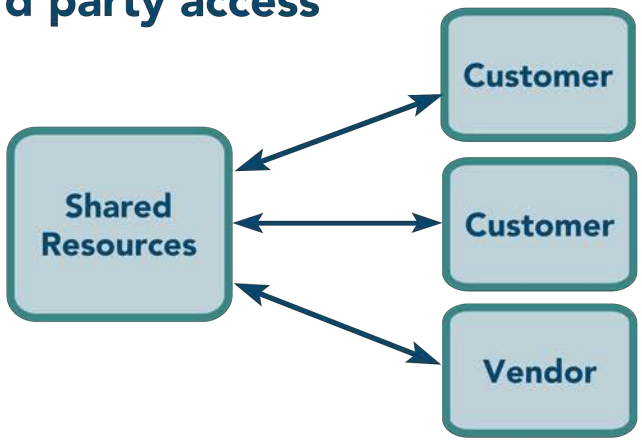


Separating departments

Centralized resources



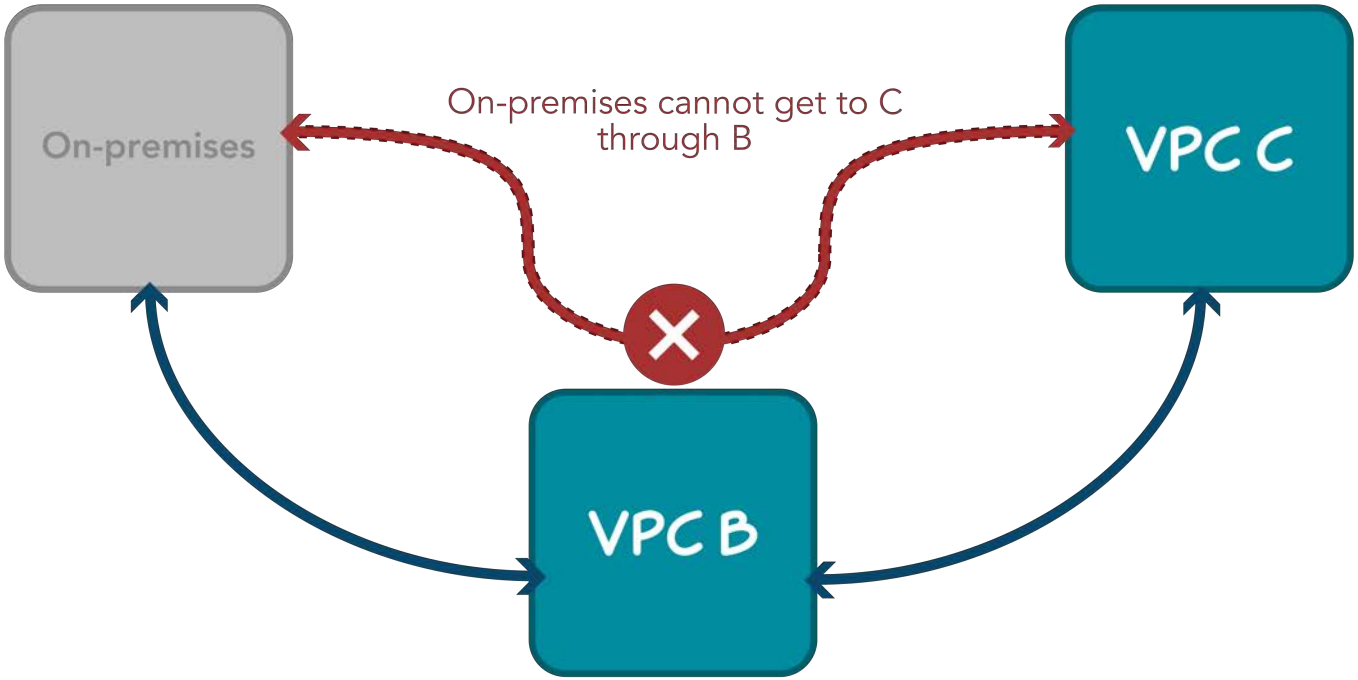
Third party access



Networking

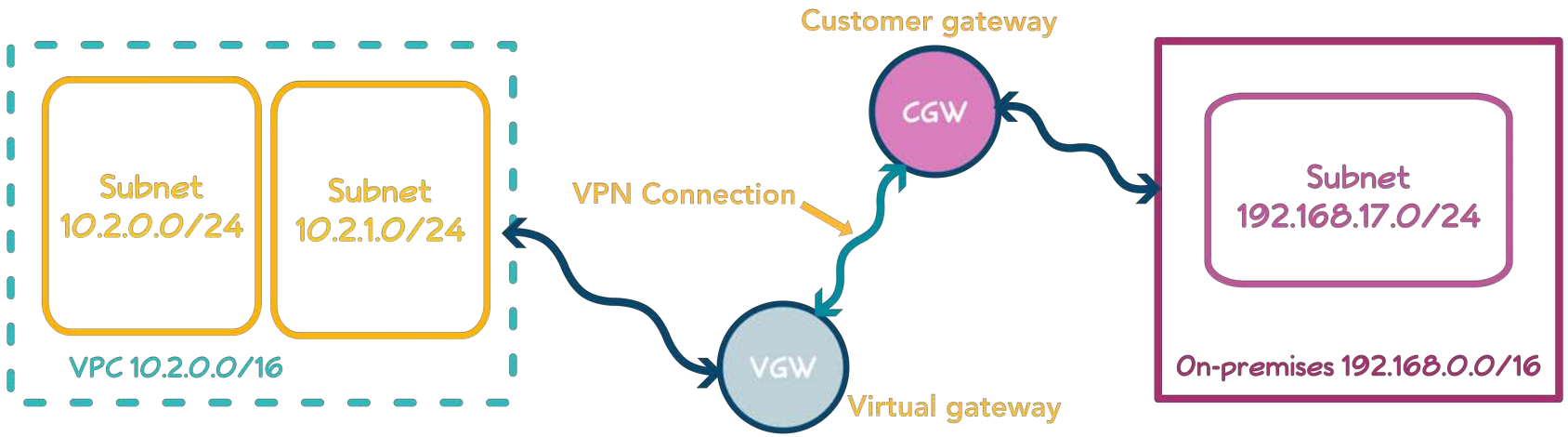
VPN Access

- Extend on-premises network into VPC
- Requires:
  - Virtual Gateway attached to VPC
  - Customer Gateway
    - Physical hardware on-premises
- Beware of IP range overlap in routes
- Always one-to-one
- No transitive peering
- No edge-to-edge routing



Networking

Extending on-premises with VPN



CIDR	Destination
10.2.0.0/16	local
192.168.0.0/16	vgw-a1b2c3d4

CIDR	Destination
10.2.0.0/16	Customer gw.

Route entire VPC

or

Route specific subnets

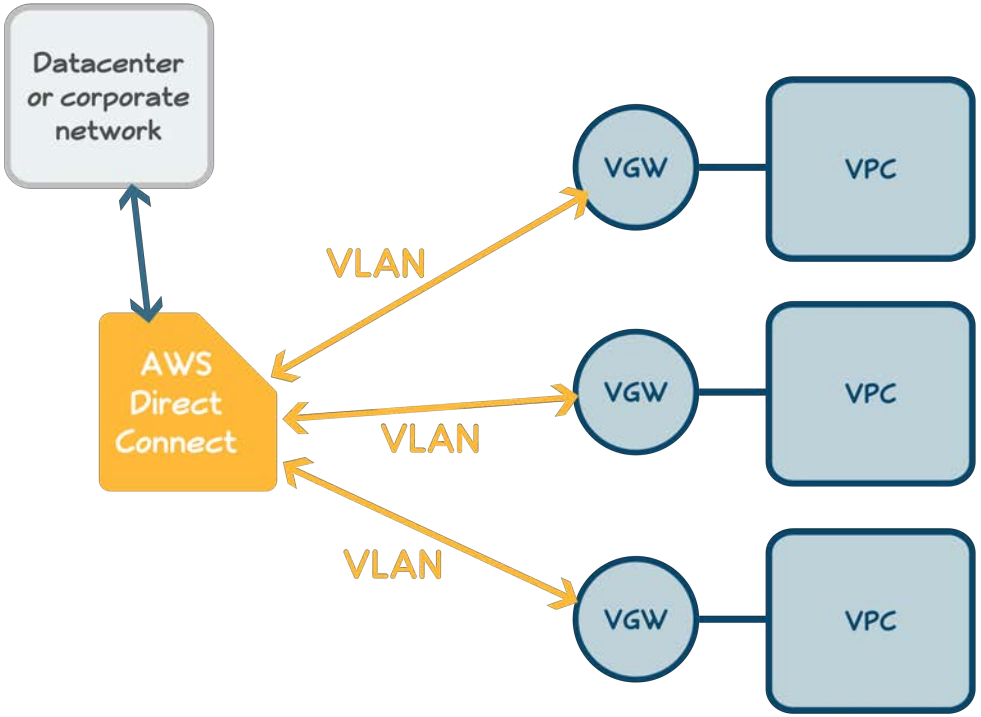
CIDR	Destination
10.2.0.0/16	local
192.168.17.0/24	vgw-a1b2c3d4

CIDR	Destination
10.2.1.0/24	Customer gw.

Networking

AWS Direct Connect

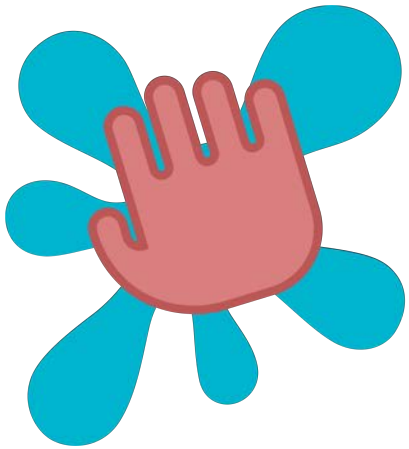
- Dedicated private connection
- 1Gbps or 10Gbps options
- Can establish private connectivity to one or multiple VPCs
- Lower data transfer rate than Internet
- Increase bandwith and throughput
- More consistent network performance



Networking

Hands-on activity

- Let's see how to:
  - Create a VPC
  - Add subnets to the VPC
  - Route to the Internet
  - Create and apply a NACL
  - Create a Security Group, and
  - Peer two VPCs



hands-on
TRAINING

Rafael Fernández (rfernandez@fi.upm.es)

AMAZON ELASTIC COMPUTE CLOUD (EC2)

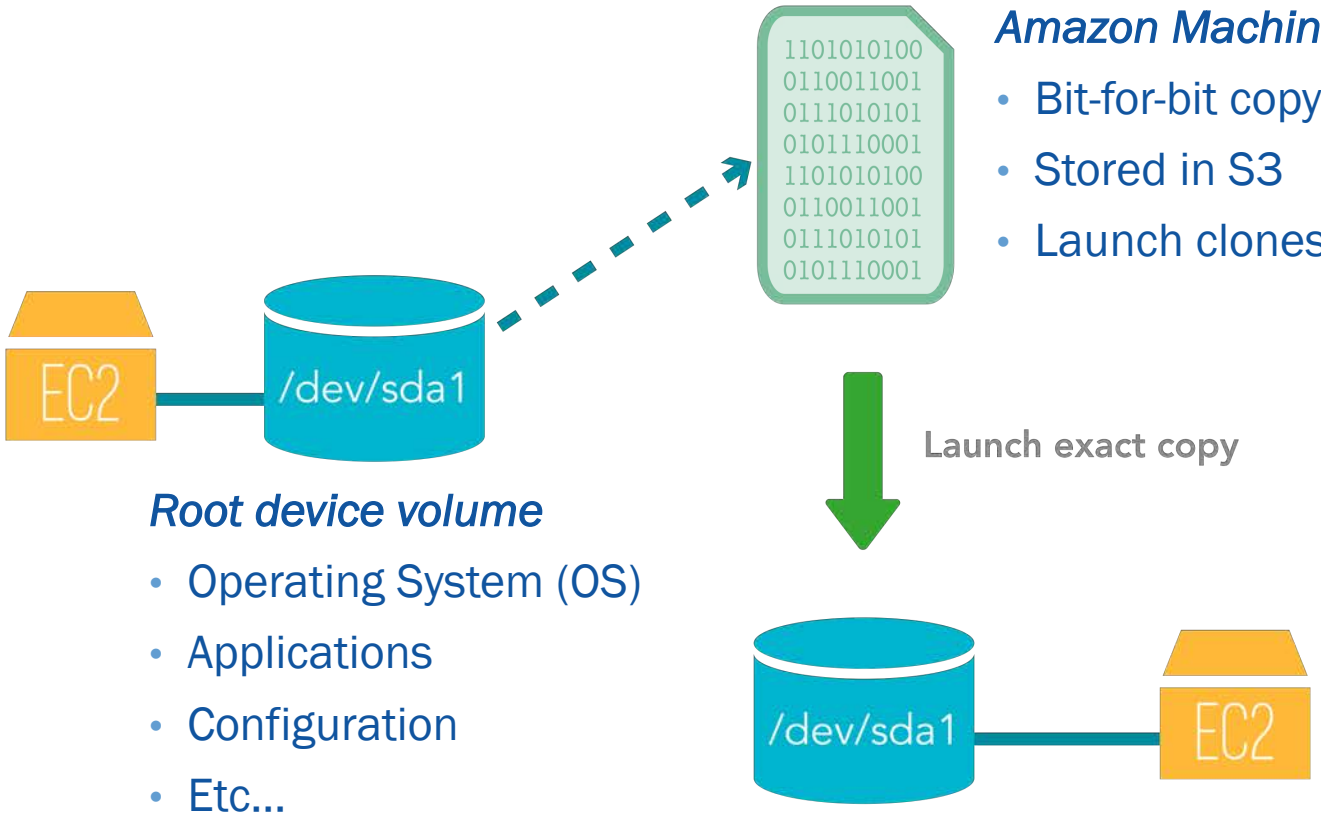
Amazon Elastic Compute Cloud (EC2)

Introduction

- Virtual Machine
- Xen hypervisor
- Various combinations of CPU, memory, disk, IO
- One virtual machine called an *instance*
- Launch 1 to thousands
- Consider them *disposable*

Amazon Elastic Compute Cloud (EC2)

Amazon Machine Images (AMI)



Amazon Machine Image (AMI)

- Bit-for-bit copy of root volume
- Stored in S3
- Launch clones of original

Root device volume

- Operating System (OS)
- Applications
- Configuration
- Etc...

Amazon Elastic Compute Cloud (EC2)

Instances

- An **instance** is an AMI running as virtual server in cloud
- The **instance type** determines the hardware of the host computer used for your instance (compute, memory, and storage capabilities)
  - Dedicated resources of the host computer to an instance:
    - CPU, memory, and instance storage
  - Shared resources of the host computer among instances:
    - Network and the disk subsystem
- The **virtualization type** is determined by the AMI used to launch the instance
  - Paravirtual (PV)
  - Hardware Virtual Machine (HVM)

Instance Family	Instance Types
General purpose	t2.micro | t2.mini | t2.medium |m3.medium | m3.large | m3.xlarge | m3.2xlarge
Compute optimized	c3.large | c3.xlarge | c3.2xlarge | c3.4xlarge | c3.8xlarge
Memory optimized	r3.large | r3.xlarge | r3.2xlarge | r3.4xlarge | r3.8xlarge
Storage optimized	i2.xlarge | i2.2xlarge | i2.4xlarge | i2.8xlarge
GPU	g2.2xlarge

Amazon Elastic Compute Cloud (EC2)

Instances

- General purpose:
  - T2:
    - High Frequency Intel Xeon Processors operating at 2.5GHz with Turbo up to 3.3GHz
    - Burstable CPU, governed by CPU Credits, and consistent baseline performance
    - Lowest-cost general purpose instance type (t2.micro - Free Tier eligible)
    - Balance of compute, memory, and network resources
  - Use cases:
    - Development environments
    - Build servers
    - Code repositories
    - Low-traffic web applications
    - Early product experiments
    - Small databases

Model	vCPU	Memory(GiB)	Storage(GB)
t2.micro	1	1	EBS only
t2.small	1	2	EBS only
t2.medium	2	4	EBS only

Amazon Elastic Compute Cloud (EC2)

Instances

- General purpose:
  - M3:
    - High Frequency Intel Xeon E5-2670 v2 (Ivy Bridge) Processors
    - SSD-based instance storage for fast I/O performance
    - Balance of compute, memory, and network resources
  - Use cases:
    - Small and mid-size databases
    - Data processing tasks that require additional memory
    - Caching fleets
    - Backend servers for SAP, Microsoft SharePoint, and other enterprise applications

Model	vCPU	Memory(GiB)	SSD storage(GB)
m3.medium	1	3,75	1 x 4
m3.large	2	7,5	1 x 32
m3.xlarge	4	15	2 x 40
m3.2xlarge	8	30	2 x 80

Amazon Elastic Compute Cloud (EC2)

Instances

- Compute optimized:
  - C3:
    - High Frequency Intel Xeon E5-2680 v2 (Ivy Bridge) Processors
    - SSD-backed instance storage
    - Support for clustering
    - Support for Enhanced Networking
  - Use cases:
    - High performance front-end fleets
    - Web-servers
    - On-demand batch processing
    - Distributed analytics
    - High performance science and engineering applications
    - Ad serving
    - Batch processing
    - MMO gaming
    - Video encoding

Model	vCPU	Memory(GiB)	SSD storage(GB)
c3.large	2	3,75	2 x 16
c3.xlarge	4	7,5	2 x 40
c3.2xlarge	8	15	2 x 80
c3.4xlarge	16	30	2 x 160
c3.8xlarge	32	60	2 x 320

Amazon Elastic Compute Cloud (EC2)

Instances

- Memory optimized:
  - R3:
    - High Frequency Intel Xeon E5-2670 v2 (Ivy Bridge) Processors
    - SSD Storage
    - Lowest price point per GiB of RAM
    - Support for Enhanced Networking
  - Use cases:
    - High performance databases
    - Distributed memory caches
    - In-memory analytics
    - Genome assembly and analysis
    - Larger deployments of SAP, Microsoft SharePoint, and other enterprise applications

Model	vCPU	Memory(GiB)	SSD storage(GB)
r3.large	2	15	1 x 32
r3.xlarge	4	30,5	1 x 80
r3.2xlarge	8	61	1 x 160
r3.4xlarge	16	122	1 x 320
r3.8xlarge	32	244	2 x 320

Amazon Elastic Compute Cloud (EC2)

Instances

- Storage optimized:
  - I2:
    - High Frequency Intel Xeon E5-2670 v2 (Ivy Bridge) Processors
    - SSD Storage
    - Support for TRIM
    - High Random I/O performance
    - Support for Enhanced Networking
  - Use cases:
    - NoSQL databases like Cassandra and MongoDB
    - Scale out transactional databases
    - Data warehousing
    - Hadoop
    - Cluster file systems

Model	vCPU	Memory(GiB)	SSD storage(GB)
i2.xlarge	4	30,5	1 x 800 SSD
i2.2xlarge	8	61	2 x 800 SSD
i2.4xlarge	16	122	4 x 800 SSD
i2.8xlarge	32	244	8 x 800 SSD

Amazon Elastic Compute Cloud (EC2)

Instances

- GPU:
  - G2:
    - High Frequency Intel Xeon E5-2670 (Sandy Bridge) Processors
    - High-performance NVIDIA GPU with 1,536 CUDA cores and 4GB of video memory
    - On-board hardware video encoder
    - Support for low-latency frame capture and encoding for either the full operating system or select render targets
  - Use cases:
    - Game streaming
    - Video encoding
    - 3D application streaming
    - Other server-side graphics workloads

Model	vCPU	Memory(GiB)	SSD storage(GB)
g2.2xlarge	8	15	1 x 60

Amazon Elastic Compute Cloud (EC2)

Key Pairs

- Public and private keys
- 2048-bit SSH-2 RSA
- Create key pair in AWS
- Choose key pair when lauching
- AWS keeps public key
- AWS does not keep private key
- Linux:
  - Public key added to `.ssh/authorized_keys`
  - Use private key for SSH login
- Windows:
  - Administrator password encrypted
  - Use private key to decrypt

Amazon Elastic Compute Cloud (EC2)

Instance Metadata Service

- http://169.254.169.254/latest/meta-data/
- Available only from an EC2 instance
- Instance can get information about *itself*
- Can retrieve (includes but not limited to):
  - AMI id
  - Hostname
  - Instance-id
  - Instance-type
  - Private IP address
- Often used with scripts for instance-specific configuration

Rafael Fernández (rfernandez@fi.upm.es)

Amazon Elastic Compute Cloud (EC2)

Bootstrapping with Userdata

- Script instance to configure itself
- Key to high-availability & self-healing
- Limited to 16 KB
- Executed only at launch (first boot)
- Can be a list of environment variables
- Linux:
  - `#!/bin/bash`
  - `#!/bin/node`
  - `#!/bin/python`
  - Or other executable
- Windows
  - `<script>`
  - `<powershell>`

Amazon Elastic Compute Cloud (EC2)

Billing options

		Reserved Instances			
	On-Demand	No Up Front	Partial Up Front	Full Up Front	Spot Instances
Commitment	None	1 year	1 or 3 years	1 or 3 years	None
Hourly Price	"MSRP"	Some Discount	More Discount	Most Discount	Market Price
Availability	Not Guaranteed	Guaranteed	Guaranteed	Guaranteed	Not Guaranteed

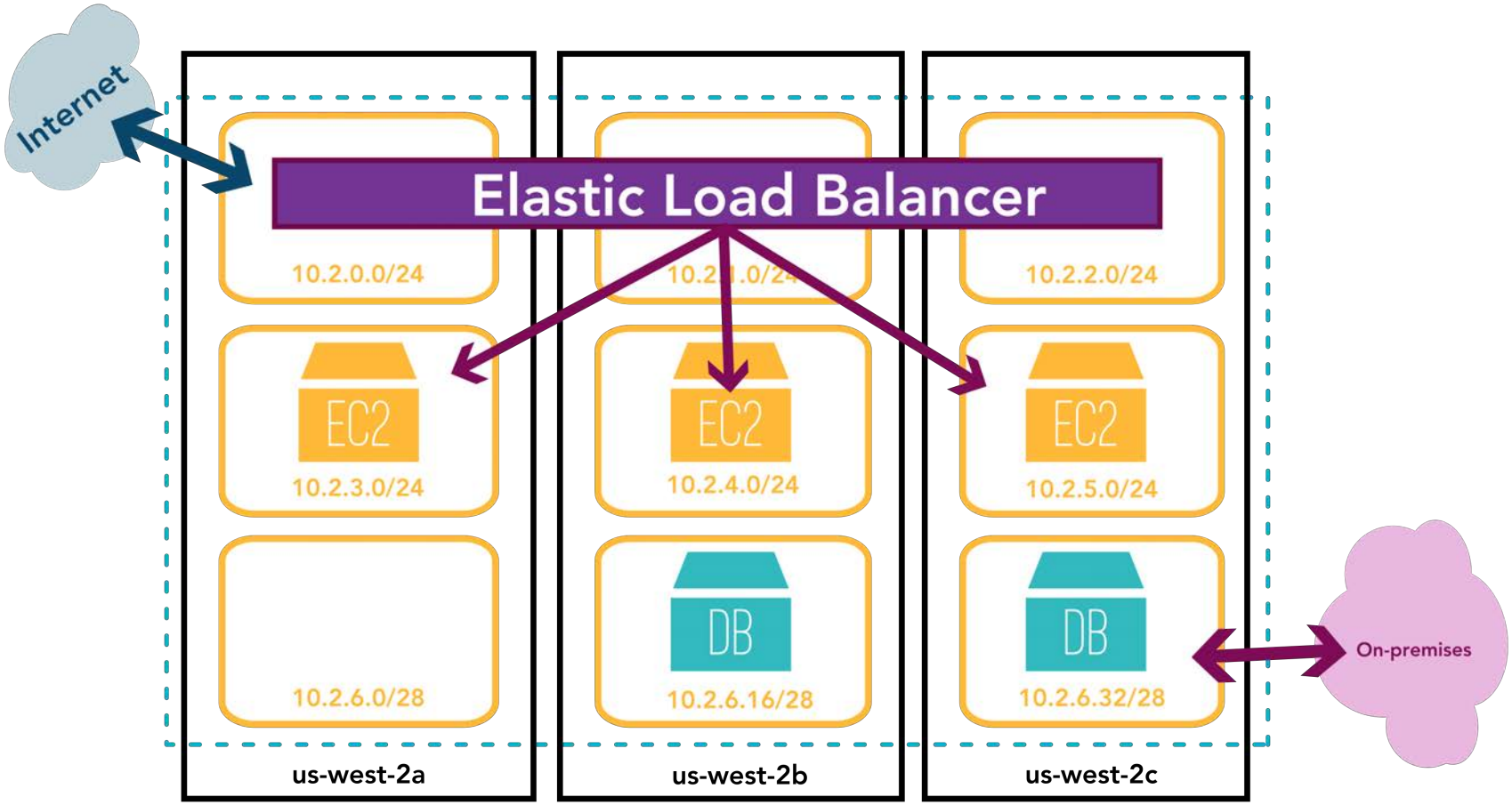
Amazon Elastic Compute Cloud (EC2)

Billing options

- Reserved Instance:
  - Up-front payment
  - 1 or 3 year term
  - Pay hourly fee regardless of use
  - Guaranteed availability
  - Significant savings over on-demand
  - Best for:
    - Constant applications
    - Databases
- Spot Instances:
  - Fee based on supply/demand
  - Hourly fee fraction of on-demand
  - Specify maximum spend
  - Instance can be terminated by AWS
  - Best for:
    - Temporary work
    - Batch processing
    - Testing/experimentation
    - End-of-year billing reports
    - Quarterly sales reports
    - Apps must tolerate interruption

Amazon Elastic Compute Cloud (EC2)

Use case: Highly-Available Web Application



Rafael Fernández (rfernandez@fi.upm.es)

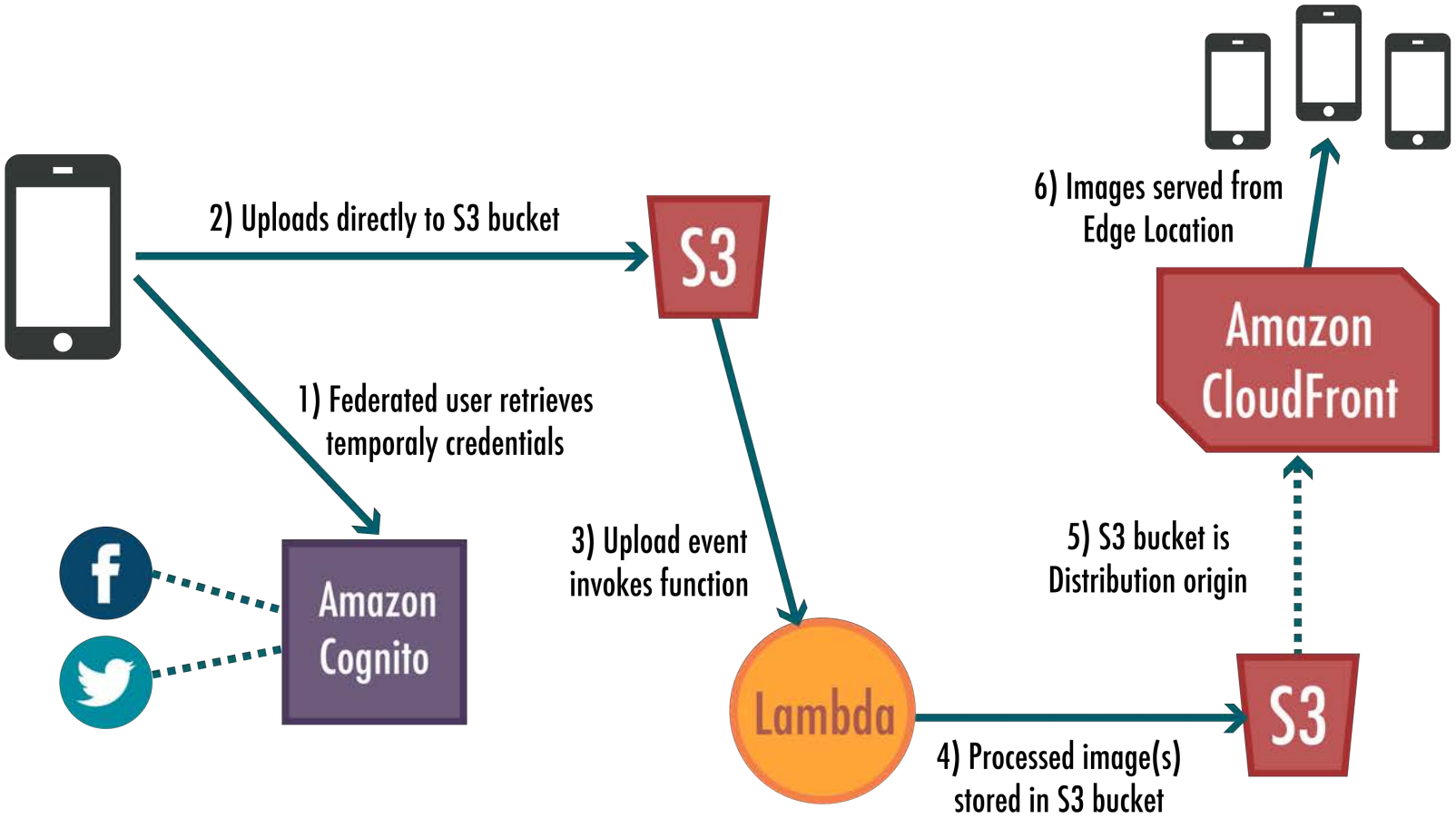
Amazon Elastic Compute Cloud (EC2)

Introduction to AWS Lambda

- “Server-less Infrastructure”
- Pay for compute time per 100ms
- Create functions
  - Inline editor
  - Upload ZIP
- Invoke functions
  - Manually
  - Through API
  - CLI
  - Events

Amazon Elastic Compute Cloud (EC2)

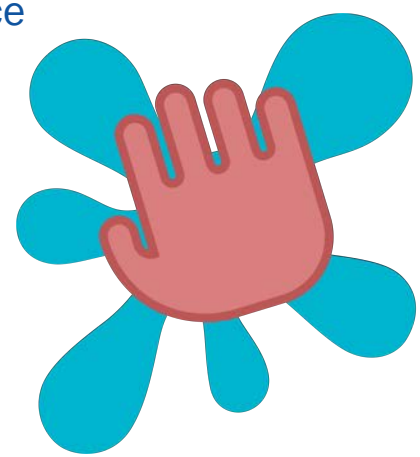
Image Processing with AWS Lambda



Computing in AWS

Hands-on activity

- Let's see how to:
  - Launch a Linux Instance
  - Use Instance Metadata Service
  - Launch a Windows Instance
  - Stop and terminate instances
  - Create a Hello World in AWS Lambda



hands-on
TRAINING

Rafael Fernández (rfernandez@fi.upm.es)

STORAGE IN AWS

Storage in AWS

Overview of AWS Storage Options

- **Instance Store**
  - Built-In
  - Free*
  - High IOPs
  - Ephimeral*
  - No snapshots

- **Block Storage**
  - Independent
  - €/GB/Month
  - Up to 16TB
  - Durable
  - Snapshots

- **Object Storage**
  - Object-storage
  - 3cents/GB/Month
  - “Write once read many”
  - “storage for the internet”
  - All transfers use HTTP

- **Cold Storage**
  - Archival storage
  - 1cent/GB/Month
  - Transition from S3
  - Slow retrieval*

Rafael Fernández (rfernandez@fi.upm.es)

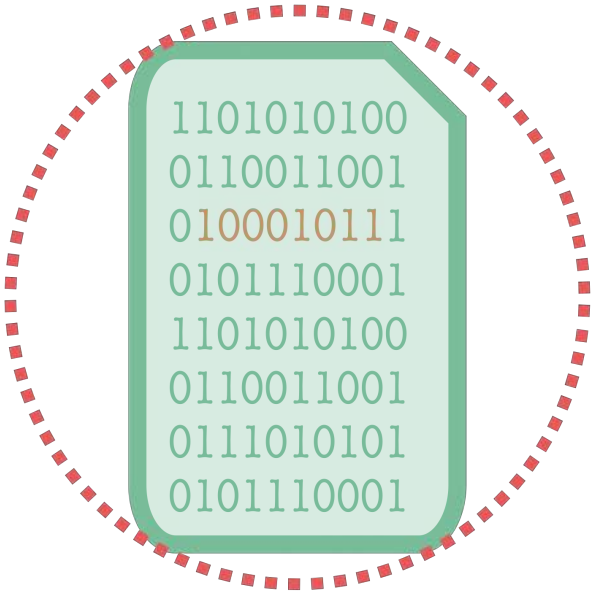
Storage in AWS

AWS Storage gateway

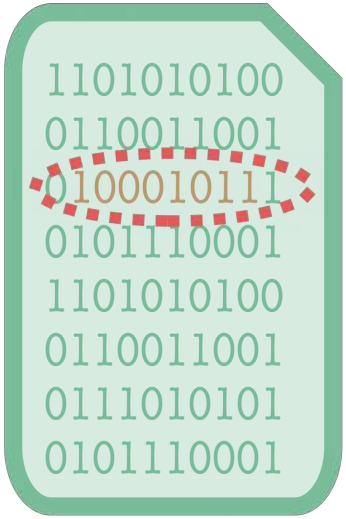
- Connect on-premises appliance with cloud-based storage
- Installed as VM on Vmware ESXi or Microsoft Hyper-V
- Supports industry-standard storage protocols
- Stores all data encrypted in Amazon S3 or Amazon Glacier
- Three configurations:
  - Gateway-Cached Volumnes
    - Stores frequently-accessed data locally
  - Gateway-Stored Volumnes
    - Stores primary data locally, point-in-time snapshots to S3
  - Gateway-Cirtual Tape Library
    - Exposes iSCSI interface, each tape stored in Amazon S3 or Amazon Glacier

Storage in AWS

Object Storage vs. Block Storage



Change 1 character in a 1GB file



Object Storage	Block Storage
Upload entire 1GB file	Update volume blocks with only 8 bits
Uses HTTP for transfer	Uses more efficient protocol
Cannot be mounted	Can be mounted

Rafael Fernández (rfernandez@fi.upm.es)

Storage in AWS

Bucket Security with Resource Policies

- Same as IAM policies
- Applied at resource level
  - Simple Storage Service
  - Simple Queue Service
  - DynamoDB
  - Many more…
- Specify a principal (who)
  - Another account
  - IAM user, group, role
  - AWS Service
  - Anonymous

Storage in AWS

Example S3 Bucket Public Read Policy

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Principal": "*",
         "Action": [
            "s3:GetObject"
         ],
         "Resource": [
            "arn:aws:s3:::mybucket/*"
         ]
      }
   ]
}
```

Allow

Anyone

To download

Any object in the bucket

Storage in AWS

Example S3 Bucket Public Read Policy

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Principal": {
            "AWS": "arn:aws:iam::1234567890:root"
         },
         "Action": [
            "s3:ListObjects",
            "s3:PutObject",
            "s3:GetObject"
         ],
         "Resource": [
            "arn:aws:s3:::mybucket",
            "arn:aws:s3:::mybucket/*"
         ]
      }
   ]
}
```

Allow

another account

to list bucket contents

to upload and download

the bucket itself

any object in the bucket

Amazon Glacier

Introduction to Amazon Glacier

- **Simple Storage Service**
  - Object-storage
    - Buckets/Objects
  - 3 cents/GB/Month
  - “Write once read many”
  - Download at will anytime

- **Cold Storage**
  - Archival storage
    - Vaults/Archives
  - <1 cent. /GB/Month
  - “Write once read rarely”
  - Write archives
    - Transition from S3
    - Direct upload
  - Download via retrieval request
    - 3-5 hour wait
    - $.05/1000 pulls

Rafael Fernández (rfernandez@fi.upm.es)

Amazon Glacier

Lifecycle Rules

logs/*

prefix

Archive to the Glacier Storage Class
all objects under logs/
after 7 days
permanently delete after 2600 days

- Manage storage class
  - STANDARD
  - STANDARD_IA
  - GLACIER
- Max 1000 rules per bucket

Instance Store Volumes

Introduction

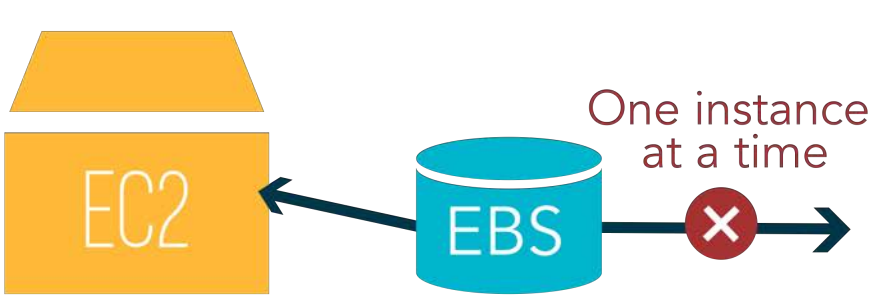
m3.medium	m3.large	m3.xlarge	d2.8xlarge	hi1.4xlarge
SSD 1x4GB	SSD 1x32GB	SSD 2x40GB	HDD 24x2TB	SSD 2x1TB

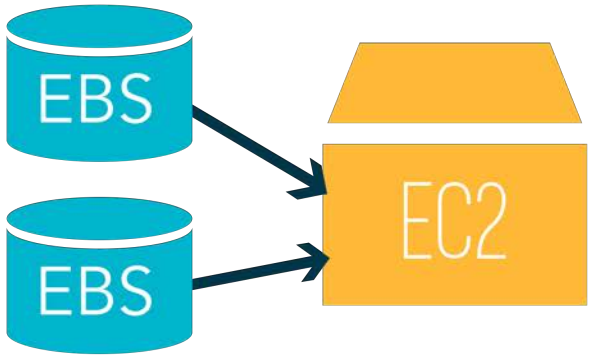
t1, t2, m4 , and c4 series have no instance store volumes

- Built-in
- Included in machine cost
- High IOPs (100k+)
- Ephemeral
  - Ephemeral stores are deleted when instance stopped or terminated
- No snapshots

Amazon Elastic Block Store (EBS)

Introduction to EBS





- Data independent from instance
- Pay for **provisioned** storage
- Exist in single AZ
- Can be encrypted
- Connected over network

- Can detach
- Can attach to different instance
- Can be encrypted
- Can be used in RAID or Logical Volume Manager (LVM)

Rafael Fernández (rfernandez@fi.upm.es)

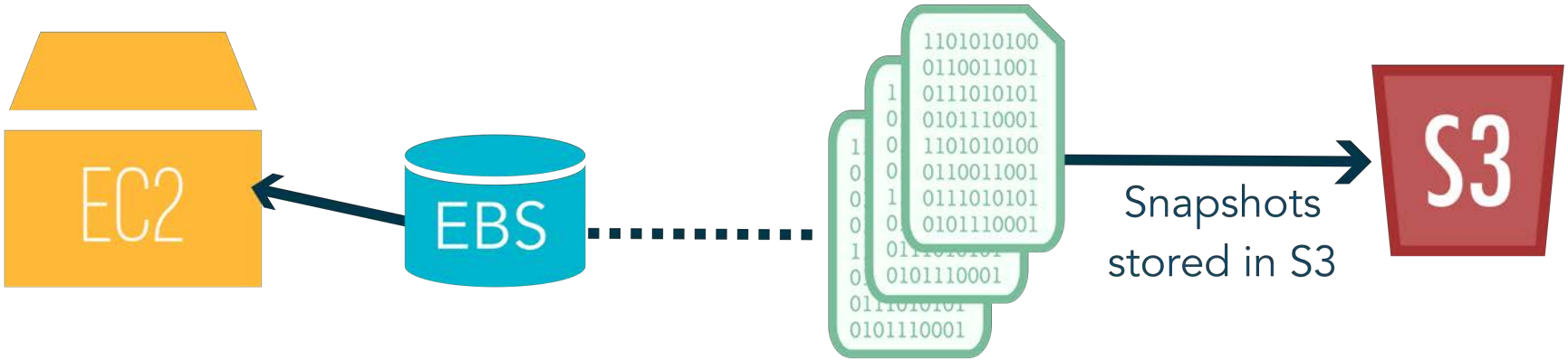
Amazon Elastic Block Store (EBS)

EBS Volume Types and Performance

SSD		HDD	
General purpose	Provisioned IOPs	Throughput Optimized	Cold
1GB – 16TB	4GB – 16TB	500GB – 16TB	500GB– 16TB
10.000 Max IOPs Baseline 3 IOPs/GB	20.000 Max IOPs Specify (max 50:1 IOPs:size)	500 Max IOPs	250 Max IOPs
160 MB/s	320 MB/s	500 MB/s	250 MB/s
16KB I/O size		1 MB I/O size	

Amazon Elastic Block Store (EBS)

EBS Snapshots



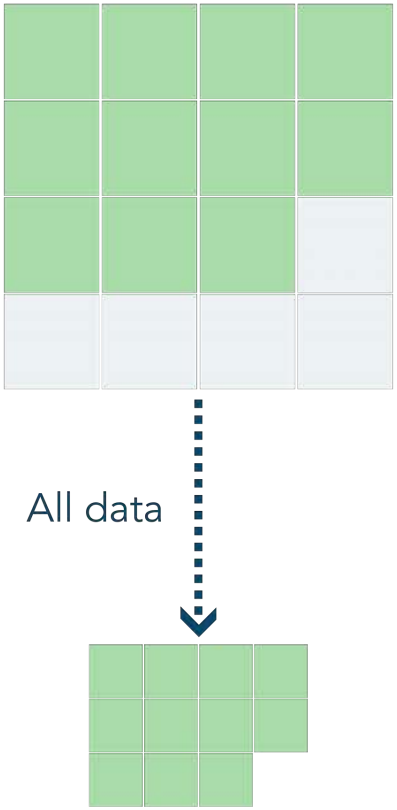
- EBS volumes exits in sigle AZ
- Durable to loss of device
- Not durable to loss of AZ

- Snapshot exits in “Admin Zone”
- Durable to loss of 2 Azs
- Easily copied to other region
- Can be shared

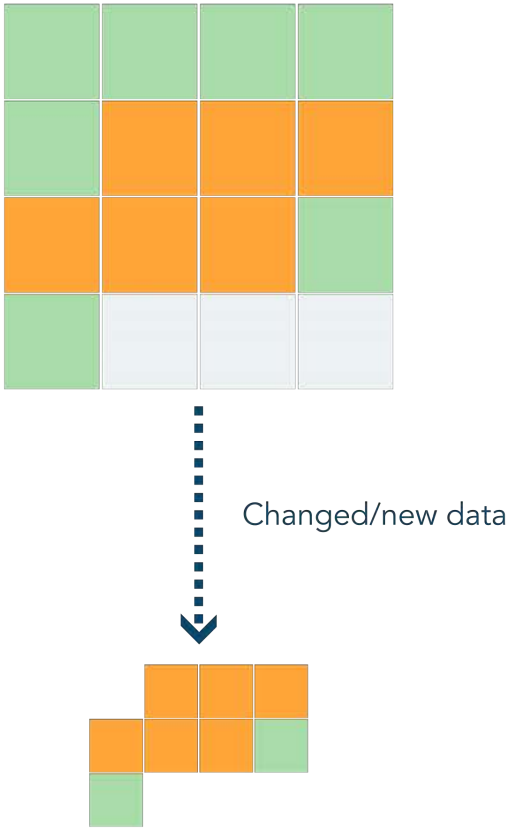
Amazon Elastic Block Store (EBS)

ESB Snapshots

- First snapshot full backup



- Subsequent *incremental*



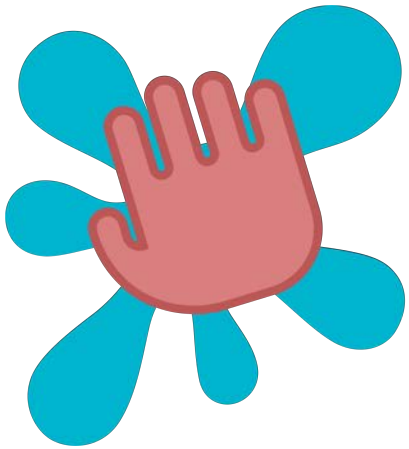
Use case

Video Transcoding and Archival

Storage in AWS

Hands-on activity

- Let's see how to:
  - Create buckets and objects in S3
  - Create a bucket policy for public read
  - Add Lifecycle Rules to Glacier
  - Create an EBS Volume
  - Mount an EBS Volume on linux and windows
  - Create an EBS Snapshot



hands-on
TRAINING

Rafael Fernández (rfernandez@fi.upm.es)

DATABASES IN AWS

Databases in AWS

Overview of AWS Database Options

- **Amazon Relational Database Service**
  - MySQL
  - SQL Server
  - PostgreSQL
  - MariaDB
  - Oracle
  - Amazon Aurora

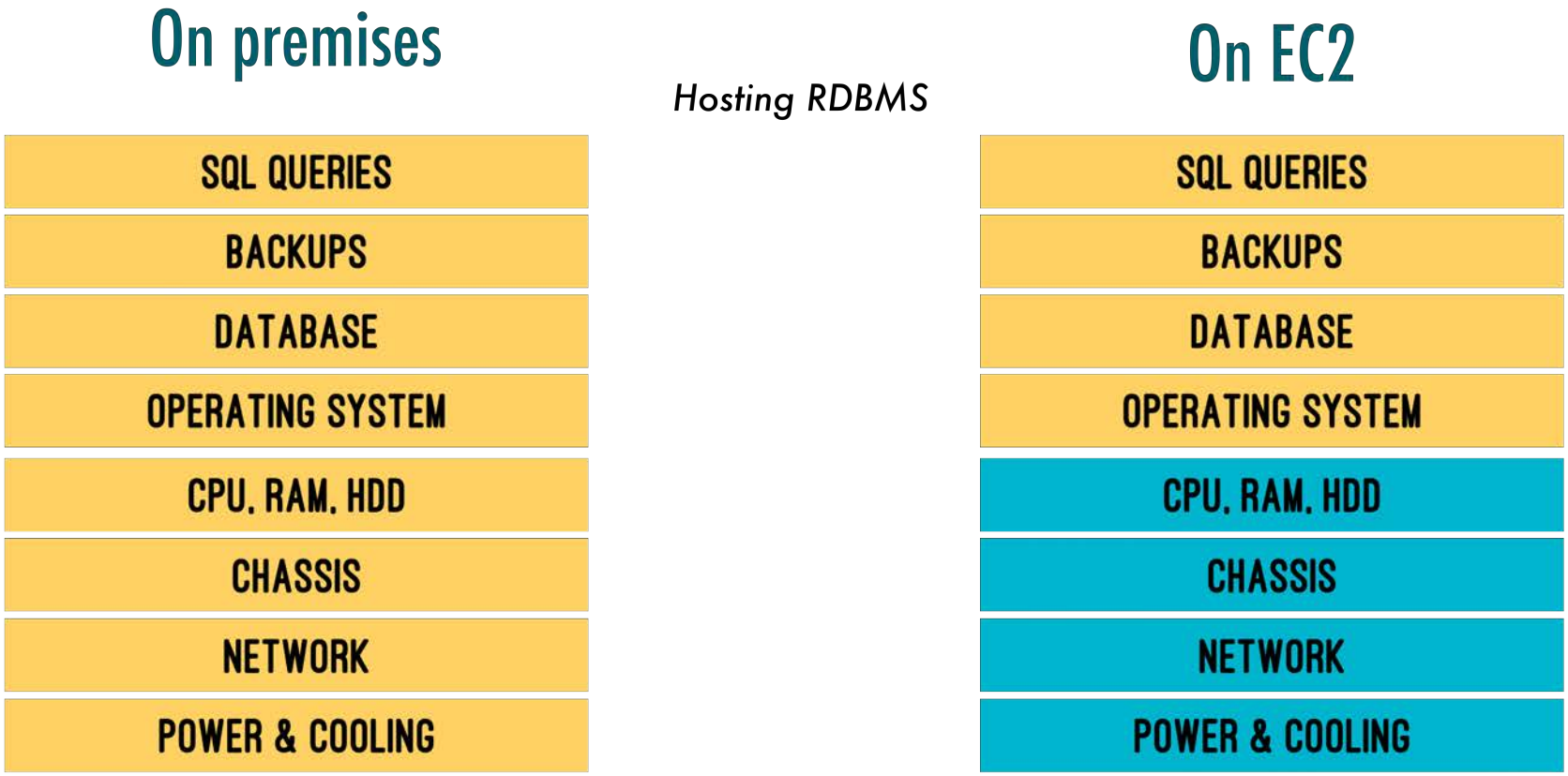
- **Amazon ElastiCache**
  - Memcached
  - Redis
  - Clustered
  - Automated maintenance

- **Amazon DynamoDB**
  - NoSQL
  - Highly-scalable
  - Fault-tolerant
  - Replicated
  - Event-driven

- **Amazon Redshift**
  - SQL compliant
  - Petabyte-scale
  - Fault-tolerant
  - Replicated
  - Encryption

Amazon Relational Database Service

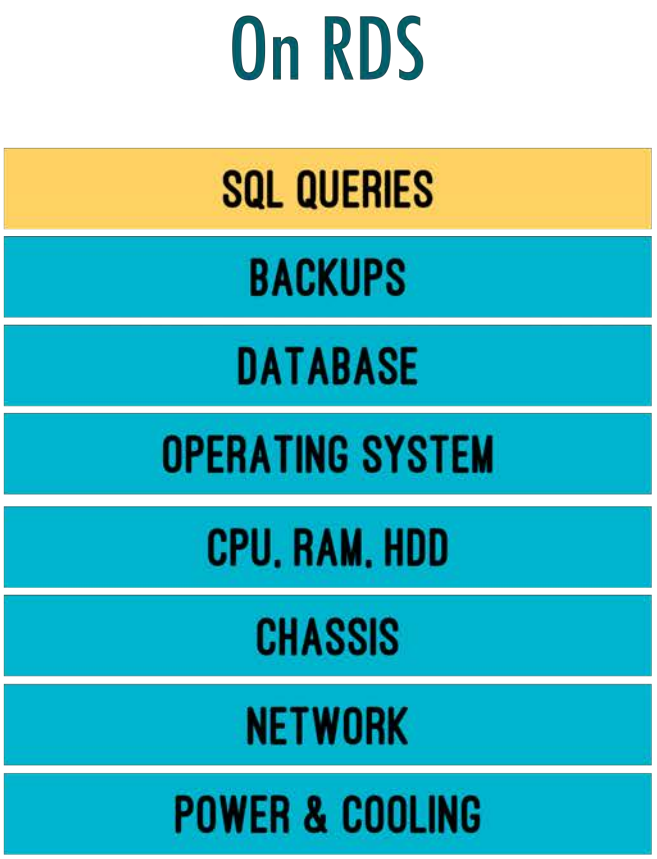
Managing a Relational Database



Rafael Fernández (rfernandez@fi.upm.es)

Amazon Relational Database Service

Managing a Relational Database

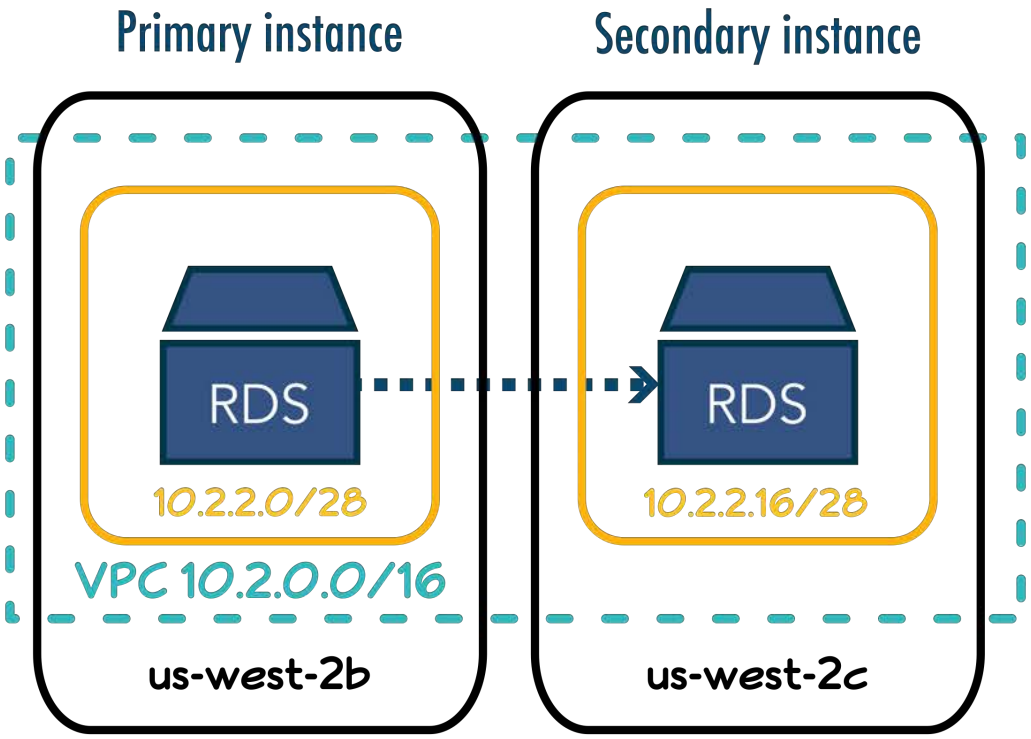


- Reduced operational burden
- Focus on application
- Data durability built-in
- Easy geographical diversity
- Choice of
  - MySQL
  - Amazon Aurora
  - SQL Server
  - Oracle
  - PostgreSQL
  - MariaDB
- Read replicas
- Automated
  - Backup
  - Patches

YOUR RESPONSIBILITY AWS MANAGED

Amazon Relational Database Service

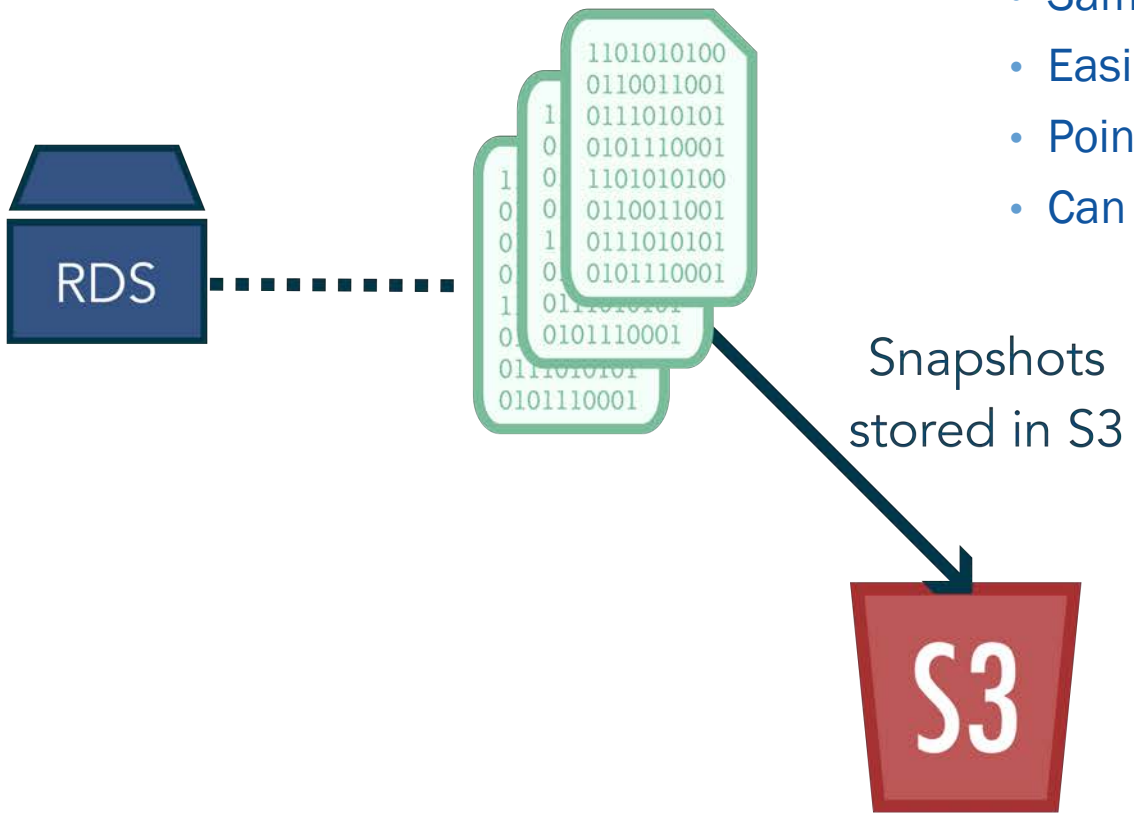
Multi-AZ Deployments



- High availability
- RTO/RPO in minutes
- Synchronous replication
- Physically distinct
- Automatic failover
  - Loss of network
  - Compute failure
  - Storage failure
- Production best practice

Amazon Relational Database Service

Automated backups

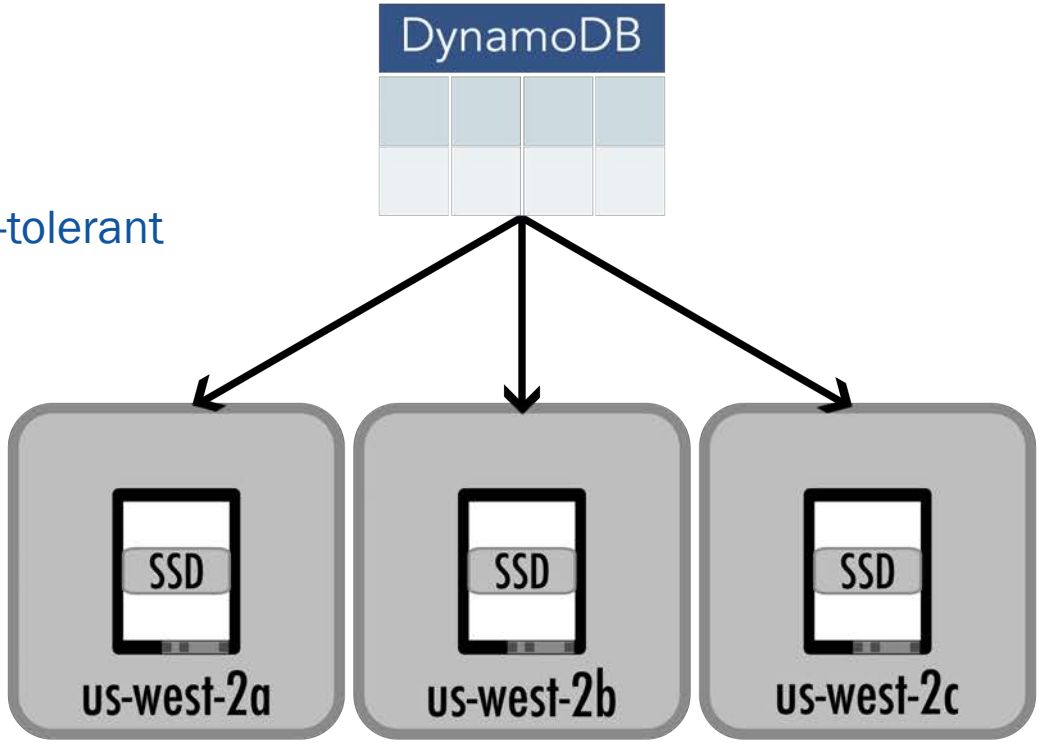


- Set backup window
- Same as EBS snapshots
- Easily Restore
- Point-in-time restore
- Can copy to other regions

Amazon DynamoDB

Introduction to Amazon DynamoDB

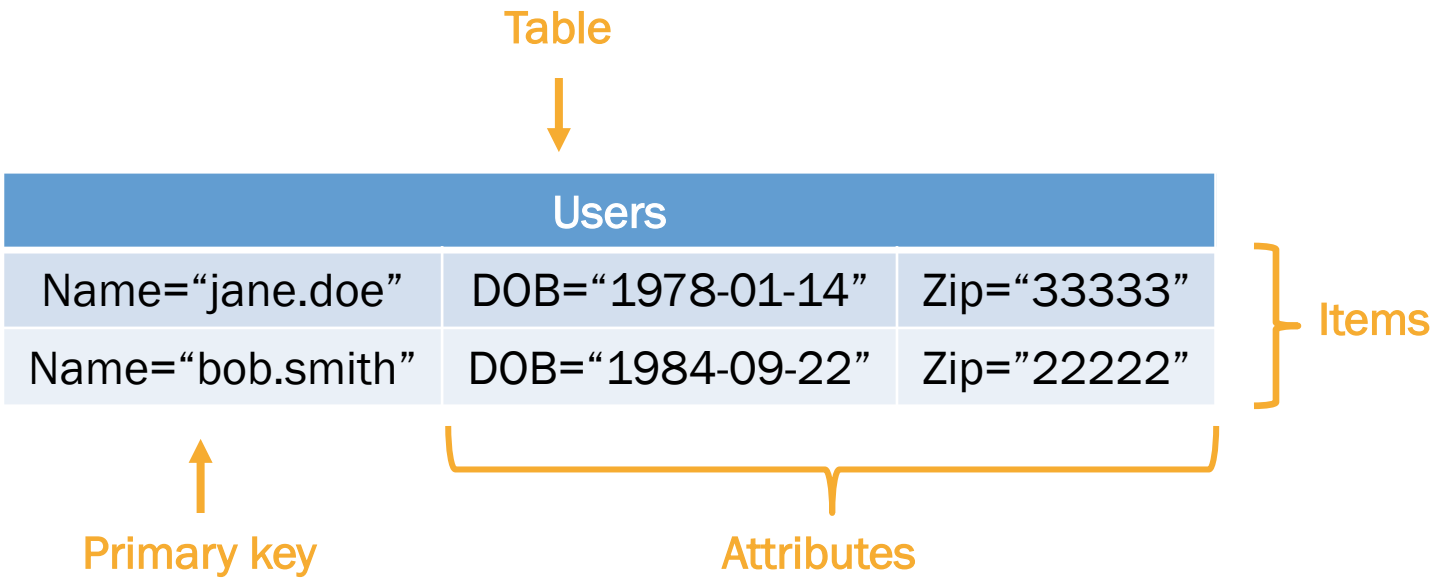
- NoSQL Data store
- SSD-backed
- Single-digit millisecond response
- Inherently highly-available & fault-tolerant
- Replicated multiple AZs
- Provisioned throughput
  - Reads
  - Writes



Rafael Fernández (rfernandez@fi.upm.es)

Amazon DynamoDB

Data Models in Amazon DynamoDB



- Tables, items, attributes
- No joins/relationships
- Schema-less
- Unique primary key required

- Secondary indexes
- No table size limit
- 400KB item size limit

Rafael Fernández (rfernandez@fi.upm.es)

Amazon DynamoDB

Primary Keys in Amazon DynamoDB

Users		
Name="jane.doe"	DOB="1978-01-14"	Zip="33333"
Name="bob.smith"	DOB="1984-09-22"	Zip="22222"

Query Name = "jane.doe"

Primary key as *partition key*

Relationships		
Name="jane.doe"	Friend="amy.benson"	Rel="Cousin"
Name="jane.doe"	Friend="bob.smith"	Rel="Friend"

Query Name = "jane.doe"
Sort key Friend

partition key *sort key*

Primary key as *"composite" primary key*

Amazon ElastiCache

Introduction

- Fully managed like RDS
- Automated minor updates
- Supports clustering

- Redis
  - In-memory DB
  - Multiple data types
  - Backed to disk
  - Automated backups & patches

- Memcached
  - In-memory cache
  - Simple key/value
  - Not backed to disk
  - Automated patches

Rafael Fernández (rfernandez@fi.upm.es)

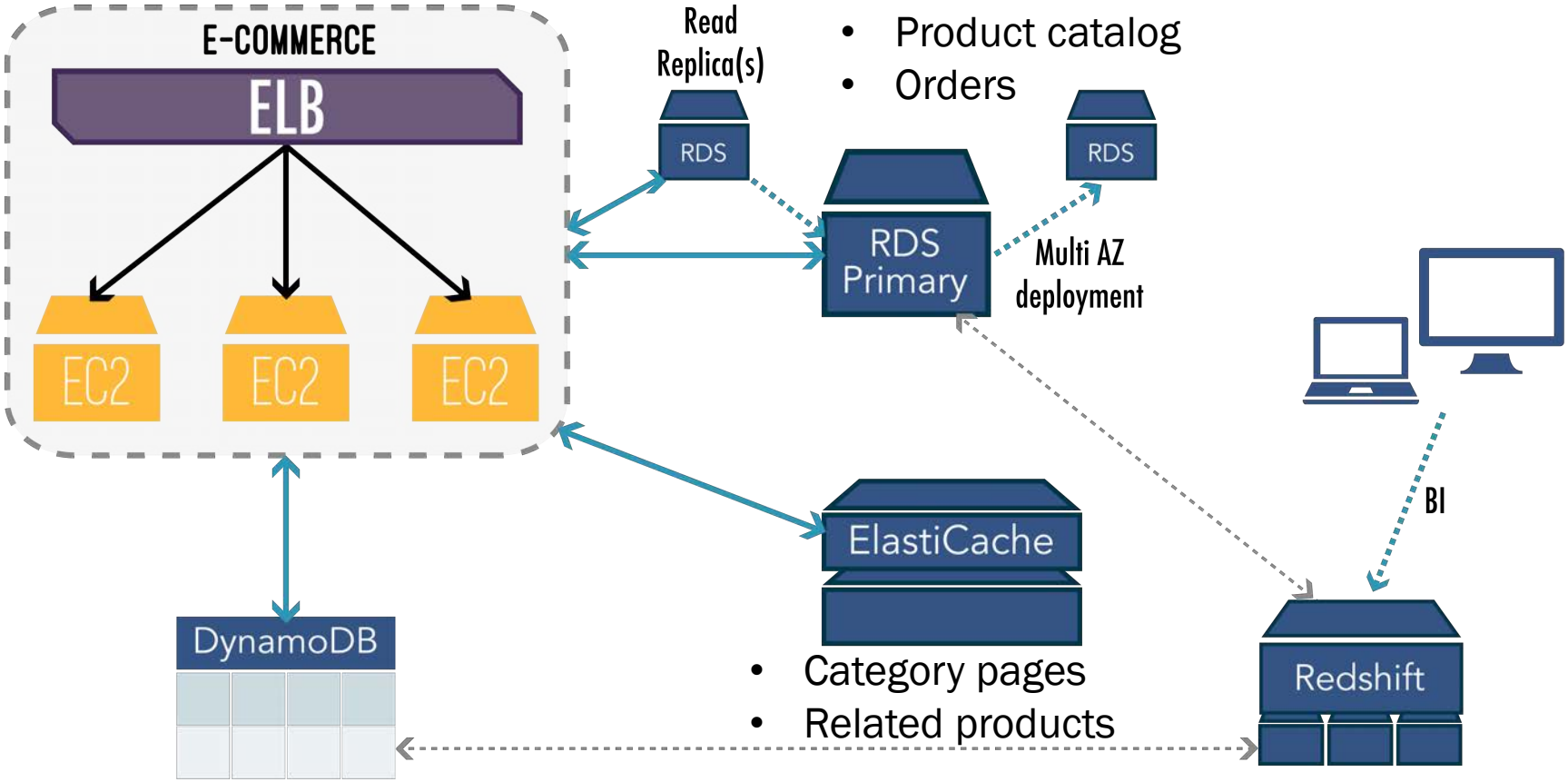
Amazon Redshift

Introduction

- Fully managed like RDS
- Petabyte scale data warehouse
- Based on Postgres 8.0.2
- SQL compliant
- Connect with JDBC/ODBC
- Parallel queries
- Ideal for OLAP & BI applications

Use case

E-Commerce



- User accounts
- Product catalog
- Orders

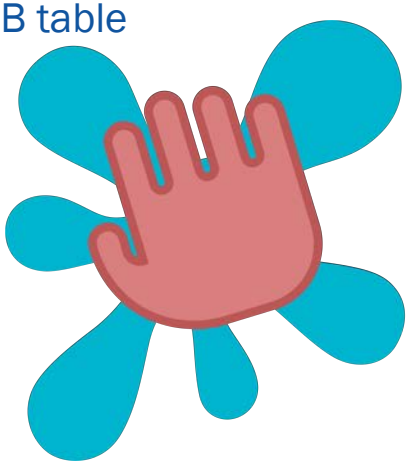
- Category pages
- Related products

- Campaign events
- Affiliate tracking
- Product view history
- Search history
- Shopping carts

Databases in AWS

Hands-on activity

- Let's see how to:
  - Launch an Amazon RDS instance
  - Creating an Amazon DynamoDB table
  - Scan and query Operations



hands-on
TRAINING

Rafael Fernández (rfernandez@fi.upm.es)

ANALYTICS IN AWS

Analytics in AWS

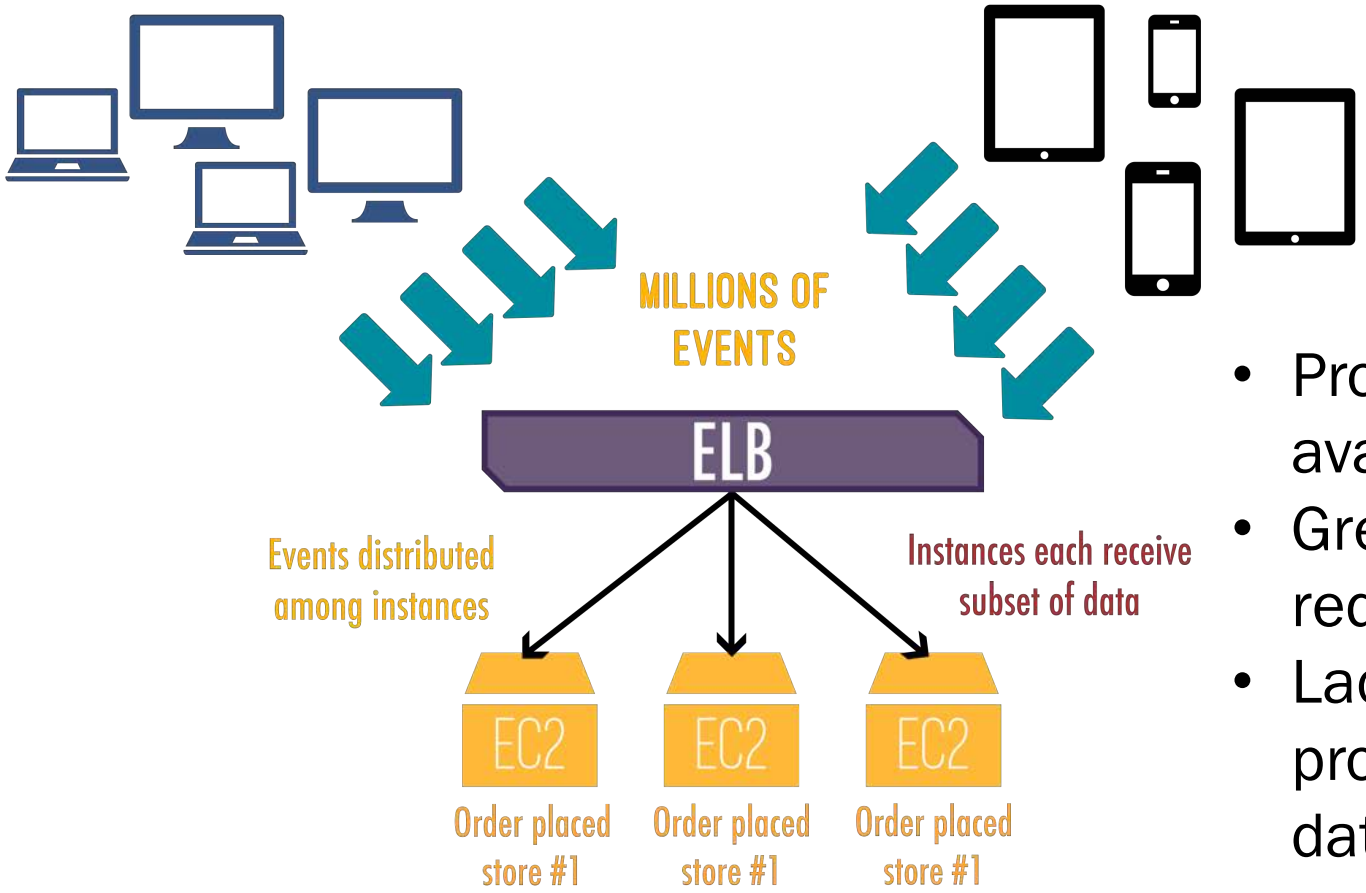
A Kinesis Use Case

- **What we have**
  - 11.000 locations worldwide
  - Nearly 40.000 delivery drivers
  - Millions of customers
  - Serving > 1.5 million pizzas per day
  - Antiquated ordering system
  - Daily, weekly, monthly reports
  - Customers have no insight
- **What we want**
  - Additional 5.000 locations
  - Additional 10.000 drivers
  - Serving > 2 million pizzas per day
  - Mobile ordering
  - Real-time reporting
    - Per location
    - Per pizza steps in process
  - Mobile push notifications

Rafael Fernández (rfernandez@fi.upm.es)

Analytics in AWS

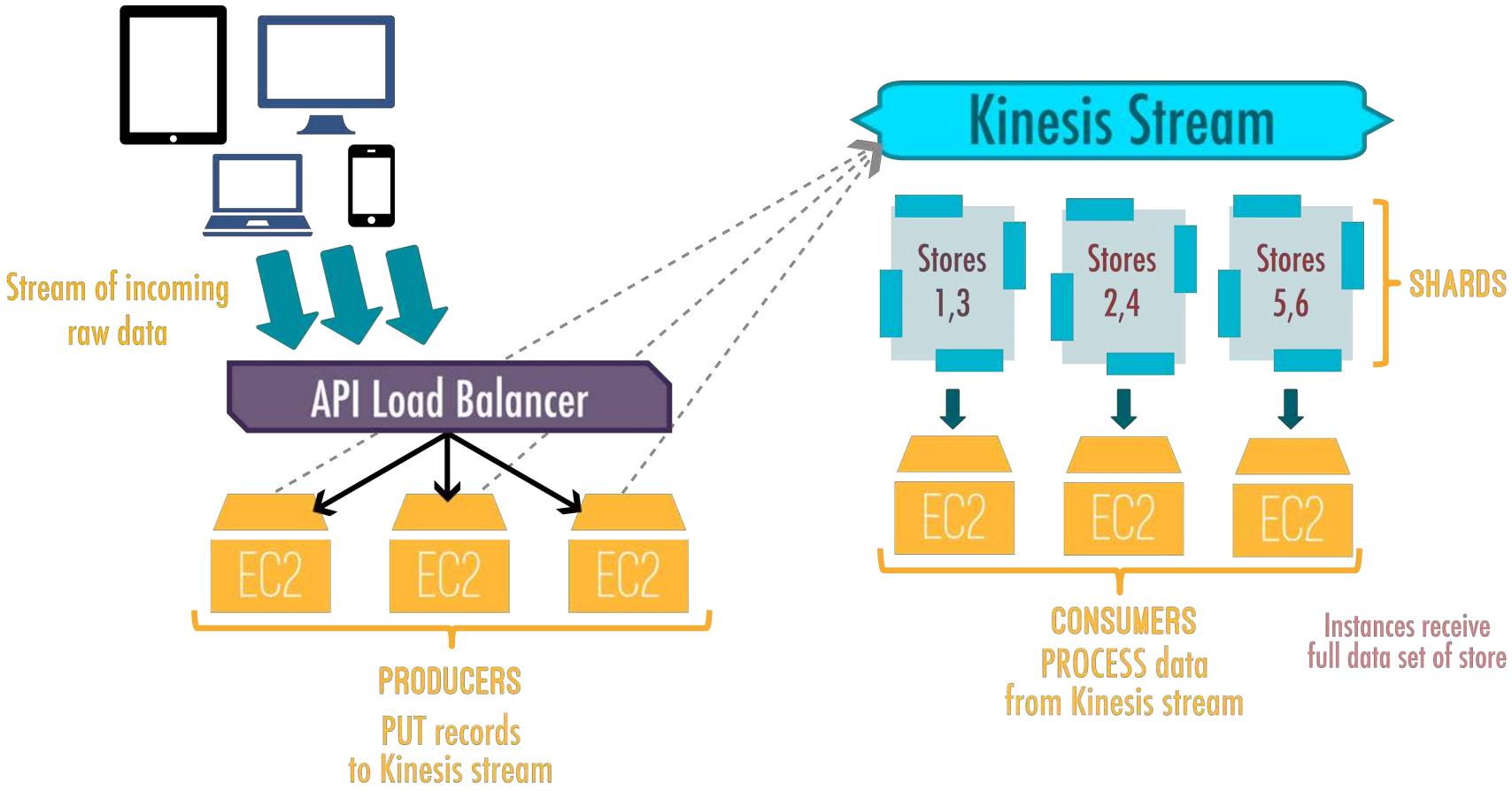
Receiving data at scale



- Provides high-availability
- Great for handling requests
- Lacking ability to process real-time data

Analytics in AWS

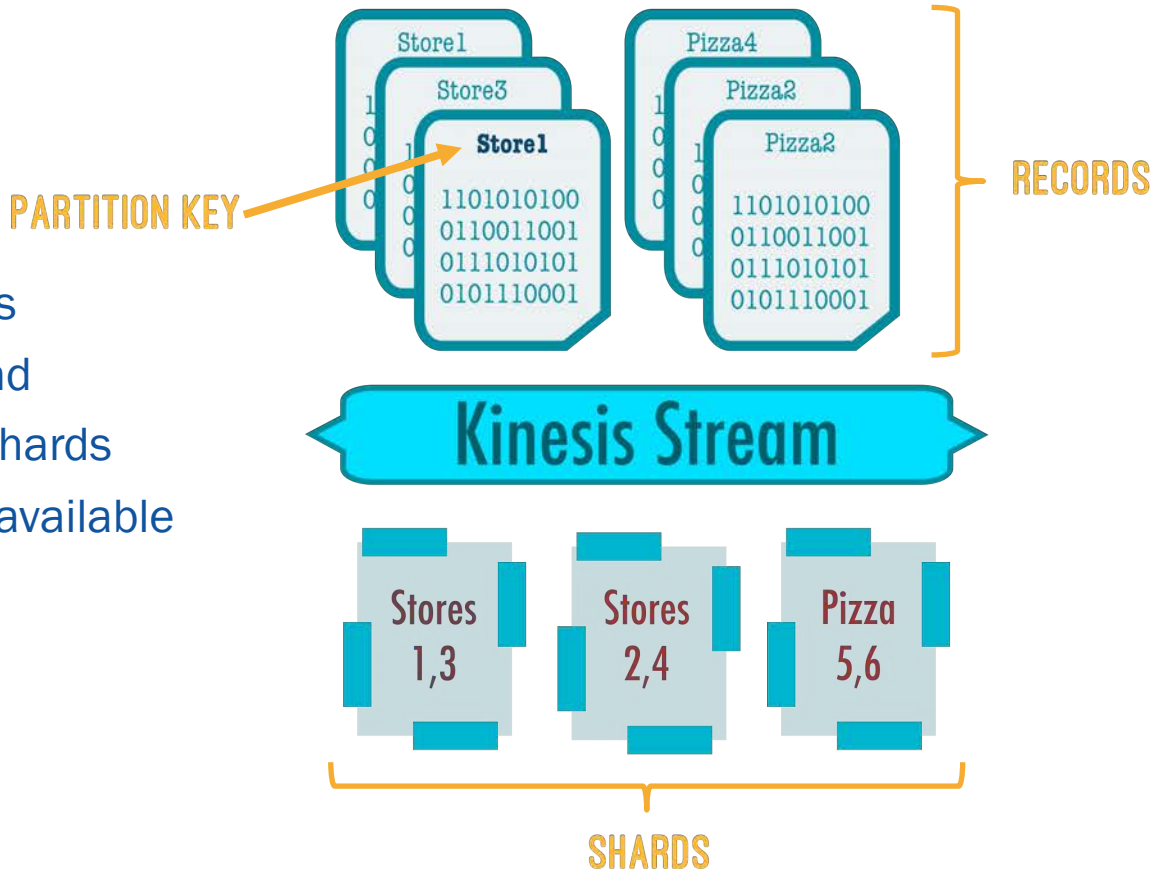
Real-time stream processing



Amazon Kinesis

Kinesis Details

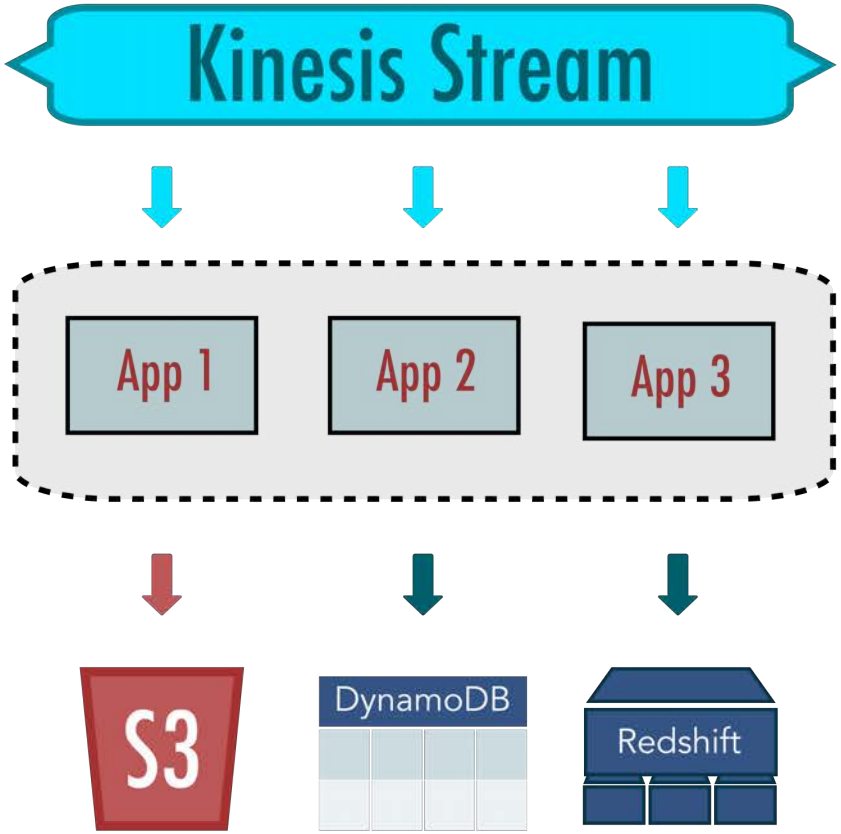
- Fully managed service
- Collect **terabytes** per hour
- Maintains **message order**
- Records kept 24hrs – 7 days
- 1000 PUTs per shard/second
- Scale by adding/removing shards
- Kinesis Client Library (KCL) available



Amazon Kinesis

Kinesis Details

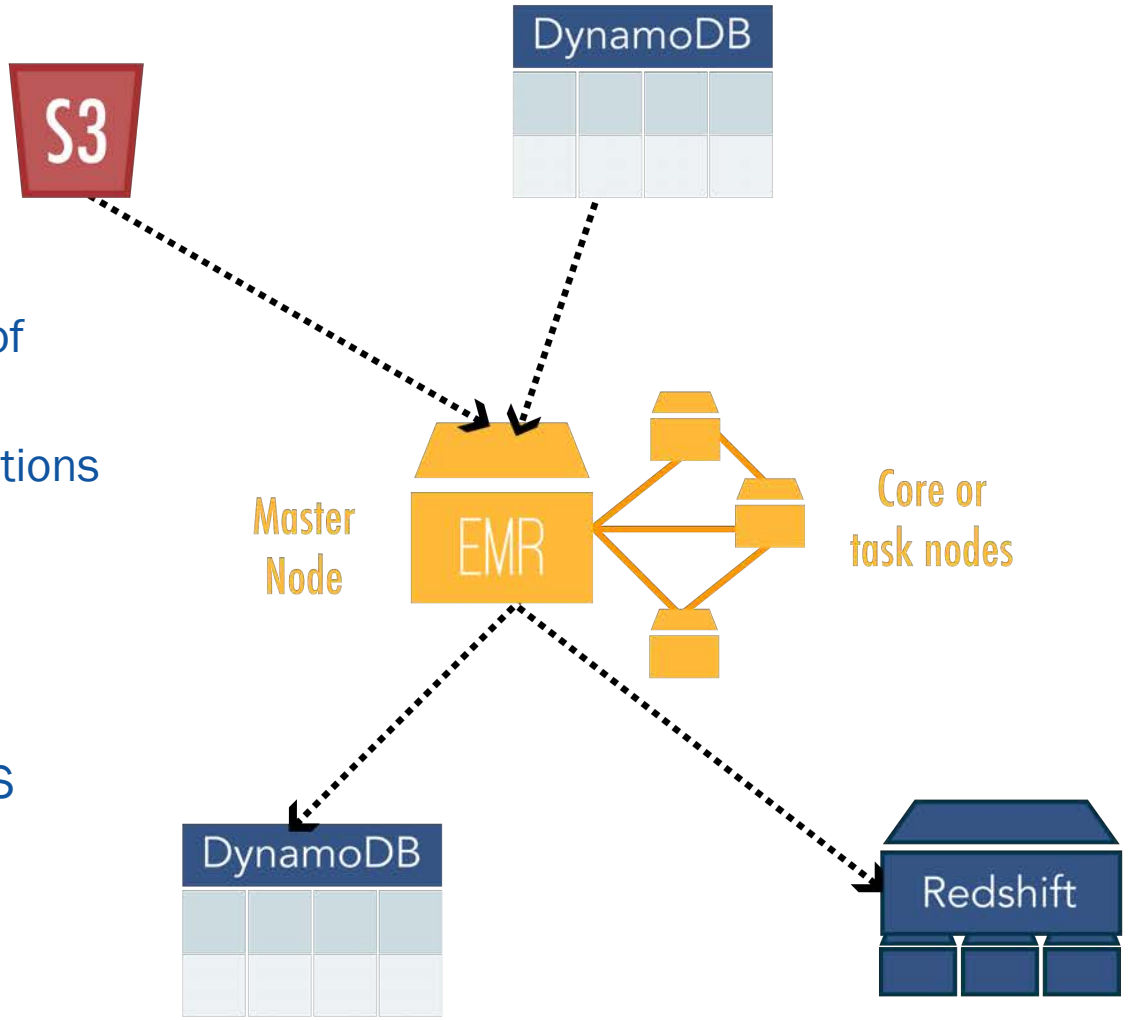
- One stream can be processed by multiple applications
  - Aggregation
  - Real-time metric extraction
  - Sliding Window Analysis
  - Event processing/notification
  - Easy integration with other data stores



Amazon Elastic MapReduce

Big Data with Amazon EMR

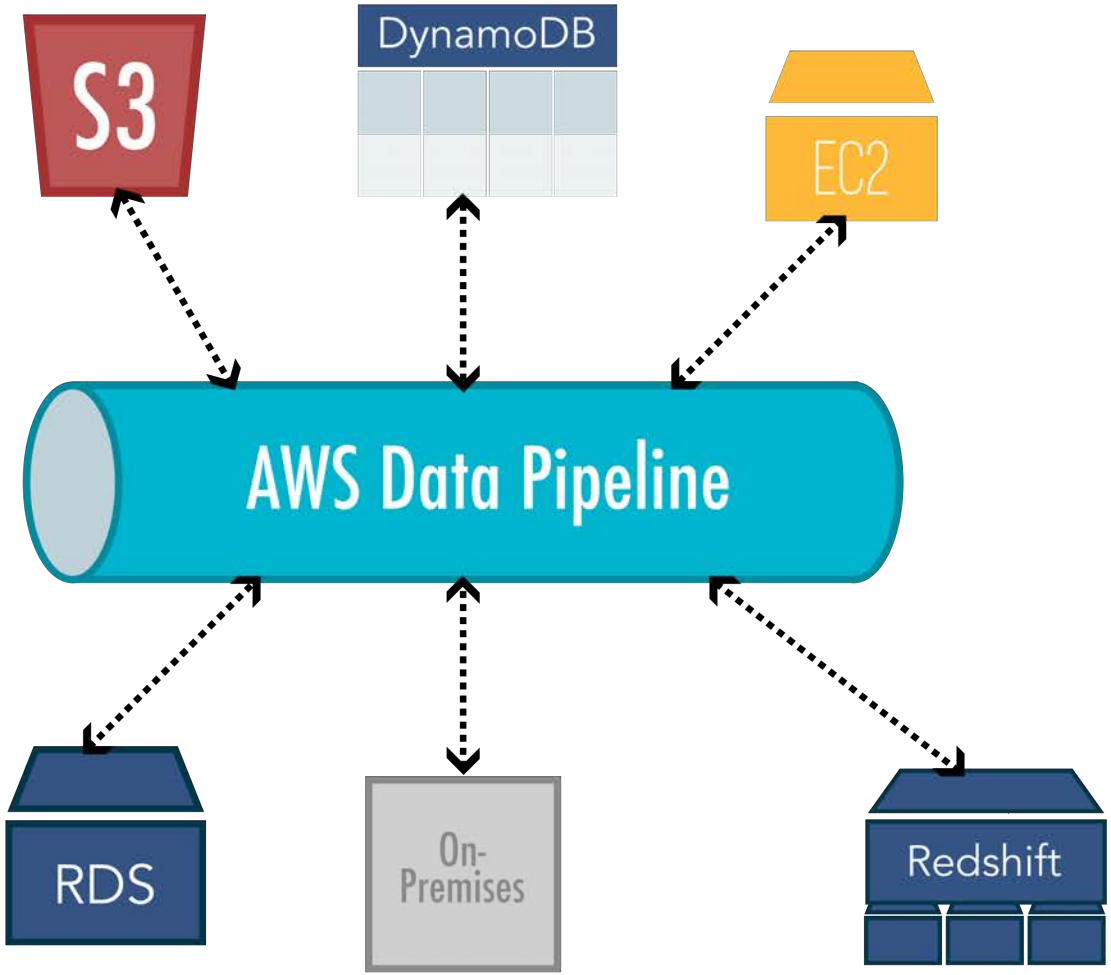
- Simplifies Big Data processing
- Managed Hadoop framework
  - Or Spark, Presto, Hbase
- Provision single or thousands of instances
- Ideal for data-intensive applications
  - Data mining
  - Log analysis
  - Scientific simulation
  - Genomics
- Integrates easiy with other AWS services



Analytics in AWS

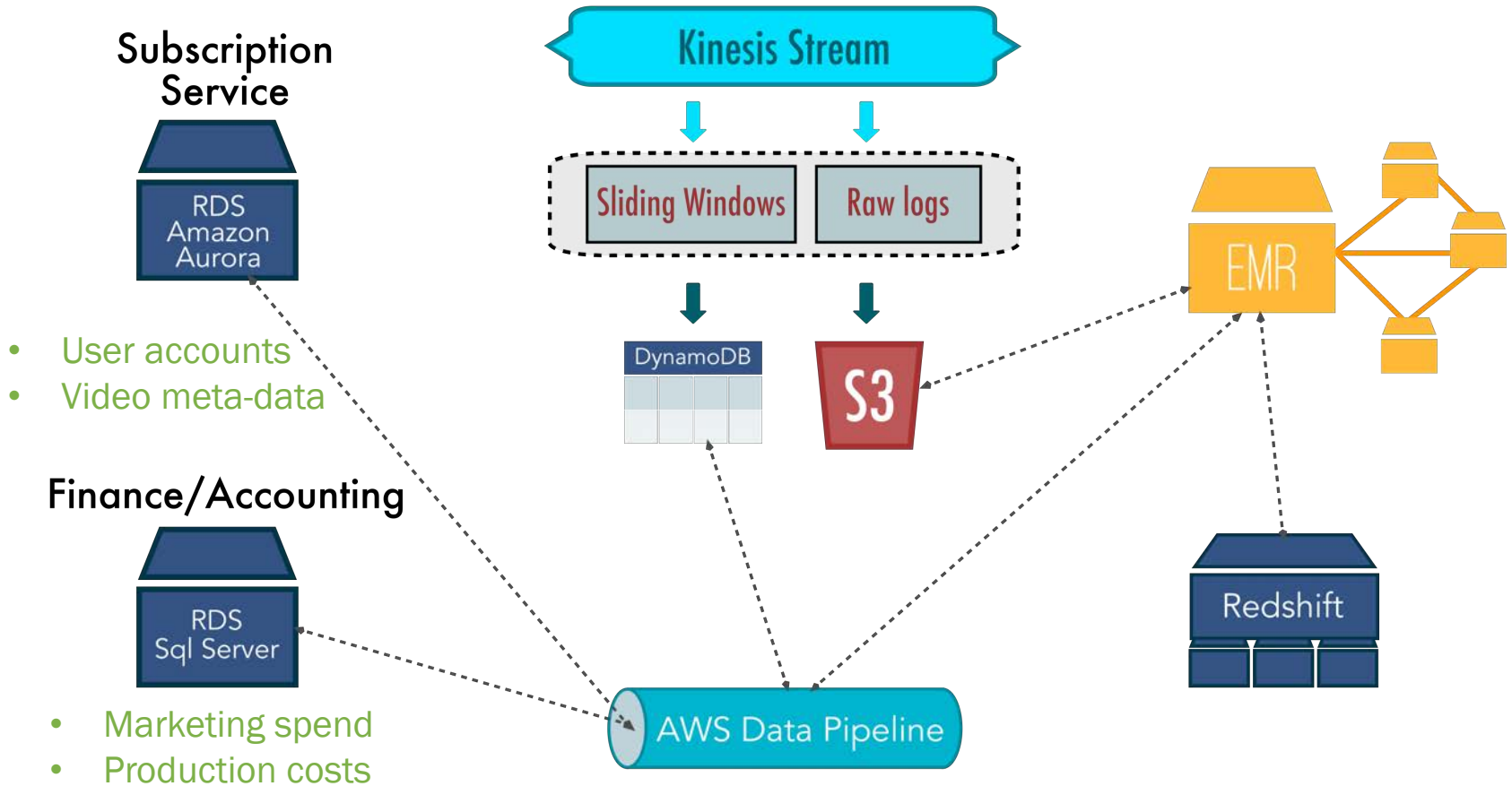
AWS Data Pipeline

- Helps move data between various data sources
- Supports sources outside AWS
- Leverages Amazon EC2 or EMR to transform data
- Configure to take actions
  - Execute SQL queries
  - Execute custom applications
- Supports scheduling
- Built-in error handling
- Full execution logs stored in Amazon S3



Analytics in AWS

Video Subscription Service Business Intelligence



- User accounts
- Video meta-data

- Marketing spend
- Production costs
- Royalties

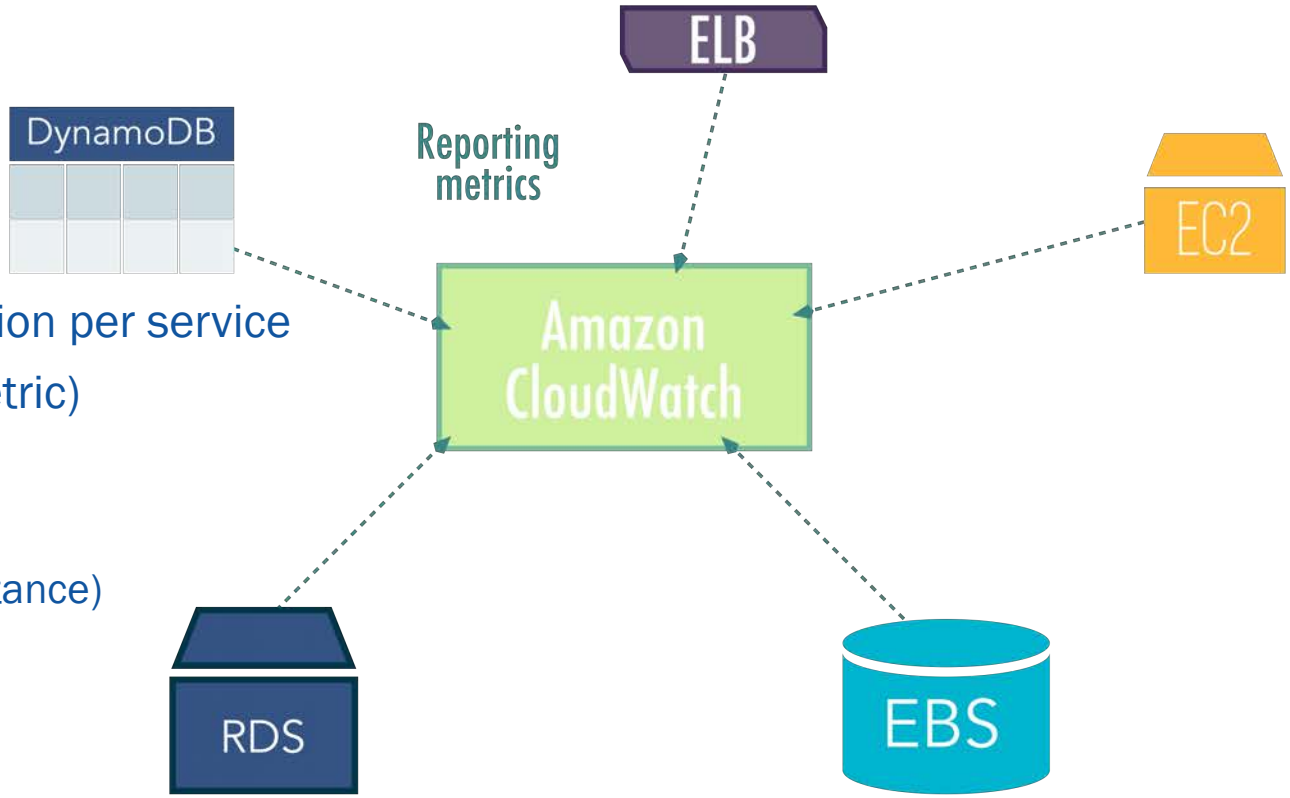
Rafael Fernández (rfernandez@fi.upm.es)

DEVELOPER AND MANAGEMENT TOOLS

Monitoring

Amazon CloudWatch

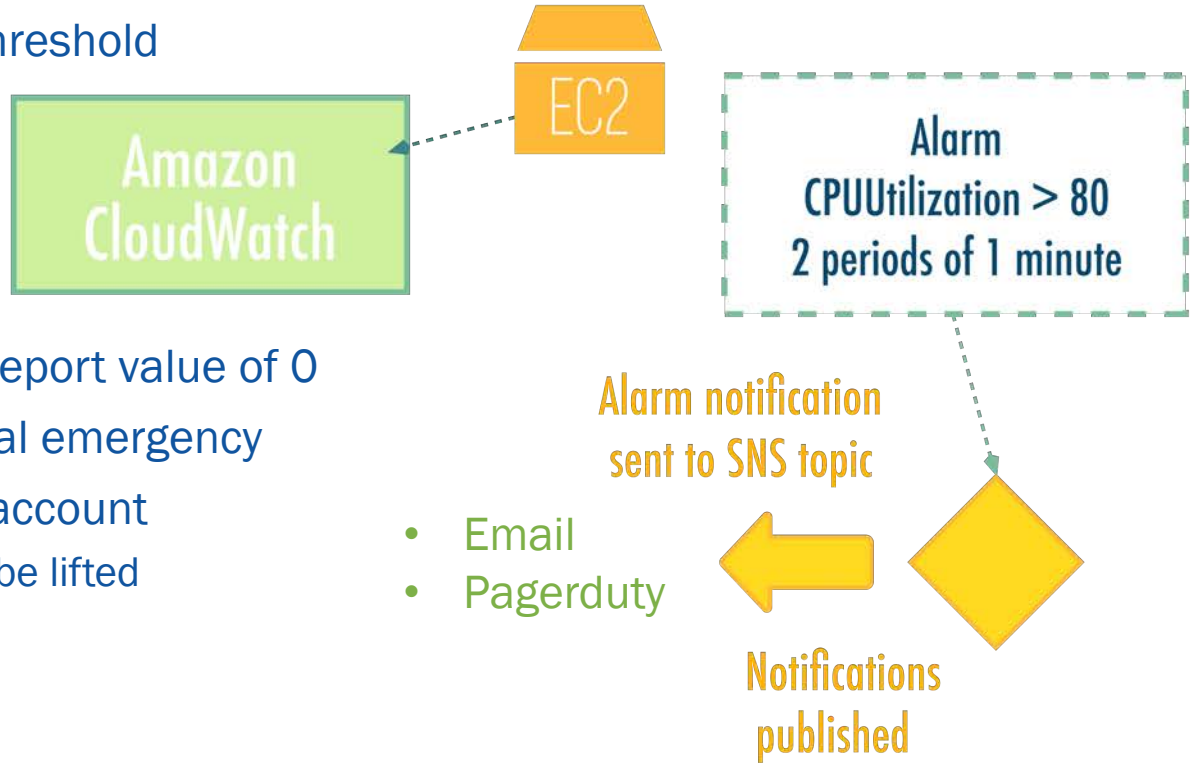
- Collect metrics
- Stored for 2 weeks
- Unique metrics collection per service
- Custom metrics ($/metric)
- EC2
  - Default 5 min interval
  - Detailed 1 min ($/instance)
  - No memory metrics
- ELB
  - Default 1 min interval
- RDS
  - Memory, connections, disk
- DynamoDB
  - Read/Write throughput



Monitoring

Amazon CloudWatch Alarms

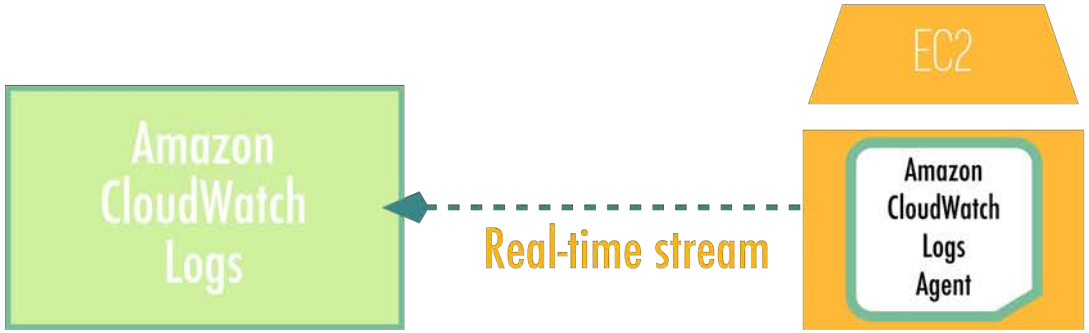
- Triggered on breach of threshold
- Three possible states:
  - OK
  - ALARM
  - INSUFFICIENT_DATA
- Some resources do not report value of 0
- Doesn't necessarily signal emergency
- Up to 5.000 alarms per account
  - Some limits in AWS can be lifted



Monitoring

Log Collection with Amazon CloudWatch Logs

- Collect logs by streaming
- Configure agent on instances
- Default retention indefinite
- Can archive to Amazon S3
- Can stream to Amazon Elasticsearch
  - Includes Kibana



Monitoring

Searching and Filtering Log Data

- Search using specific syntax
- Can search JSON fields
- Subscription filters stream to
  - Kinesis
  - Lambda
- Create metric filters
  - Counts as custom metric
  - Create alarms such as
    - Number of 404s
    - Bytes transferred
    - Customer conversions

The filter pattern "`ERROR Exception`" would match:

```
[ERROR] Caught IllegalArgumentException
[ERROR] Unhandled Exception
```

Infrastructure as Code

Amazon CloudFormation

- Solution architectures often complex
- Challenges with manual process:
  - Reliability
  - Reproducibility
  - Documentation
- Challenges with scripts
  - Dependencies
  - Parallelization
- Solution: Automate with AWS CloudFormation

- *Template*
  - JSON-formatted text file describes resources
- *AWS CloudFormation Engine*
  - Processes template, creating/updating resources
- *Stack*
  - Collection of resources created by template

Infrastructure as Code

Amazon CloudFormation

- Declarative vs. Imperative Programming
- Can/should be store in source control
- Templates can be nested
- Write once, deploy many
- Make a library for commonly deployed architectures
- AWS CloudFormation offers access to full breadth of AWS
- No imposed model for development and operations

- **Template declares via JSON:**
  - Parameters
  - Resources
    - Properties
    - Conditions
  - Mappings
  - Outputs

Infrastructure as Code

Example Template

```
{
    "Parameters": {
        "Cidr": {
            "Description": "Specify a CIDR Range",
            "Type": "String",
            "Default": "10.10.0.0/16"
        }
    },
    "Resources": {
        "applicationVpc": {
            "Type": "AWS::EC2::VPC",
            "Properties": {
                "CidrBlock": {"Ref": "Cidr"},
                "EnableDnsSupport": true
            }
        }
    }
}
```

Parameter named "Cidr"

Resource named "applicationVpc"

Reference to parameter "Cidr"

Infrastructure as Code

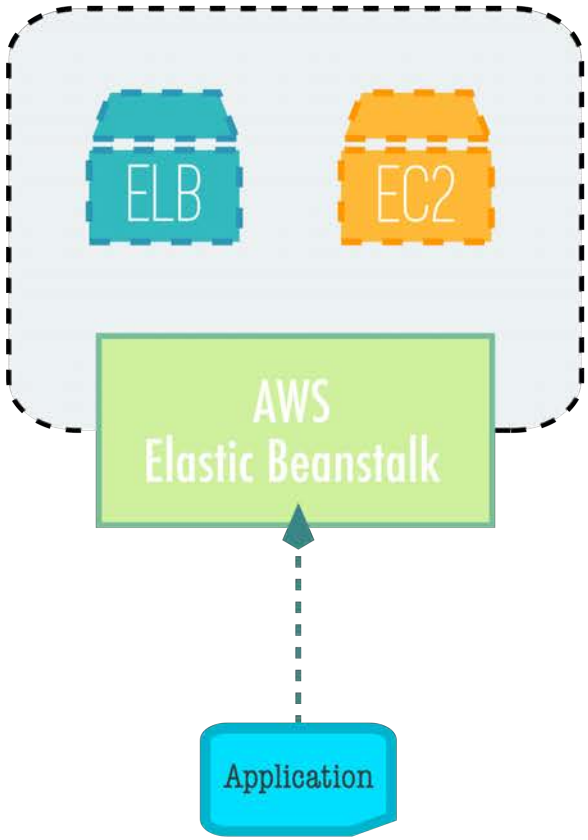
Example Template

Rafael Fernández (rfernandez@fi.upm.es)

Application Management Services

Application Deployment and Management

- AWS Elastic Beanstalk
  - Application management platform
  - Provides easy entry
  - Choice of
    - Java
    - .NET
    - Node.js
    - **Docker**
    - And more…
  - Simply upload application
  - Ideal for developers



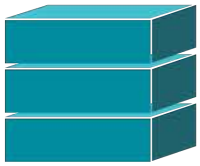
- Automatically handles
  - Capacity provisioning
  - Load balancing
  - Auto scaling
  - Monitoring

Rafael Fernández (rfernandez@fi.upm.es)

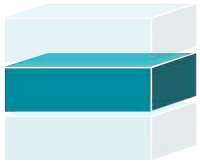
Application Management Services

Application Deployment and Management

- AWS OpsWorks
  - Configuration management platform
  - Supports Chef recipes (“Configuration as code”)
  - More control than AWS Elastic Beanstalk
  - Narrower range of resources than AWS CloudFormation
  - Maintains app health by replacing failed instances
  - Ideal for IT admins and DevOps engineers



STACK
Infrastructure and applications managed together



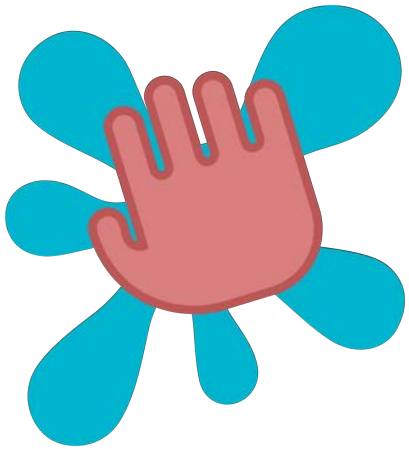
LAYER
Configuration for instances related resources

APPLICATIONS
Deploy to specific instances via Chef recipes

DevOps in AWS

Hands-on activity

- Let's see how to:
  - Create Resources with AWS Cloudformation
  - Launch an Application on AWS Elastic Beanstalk



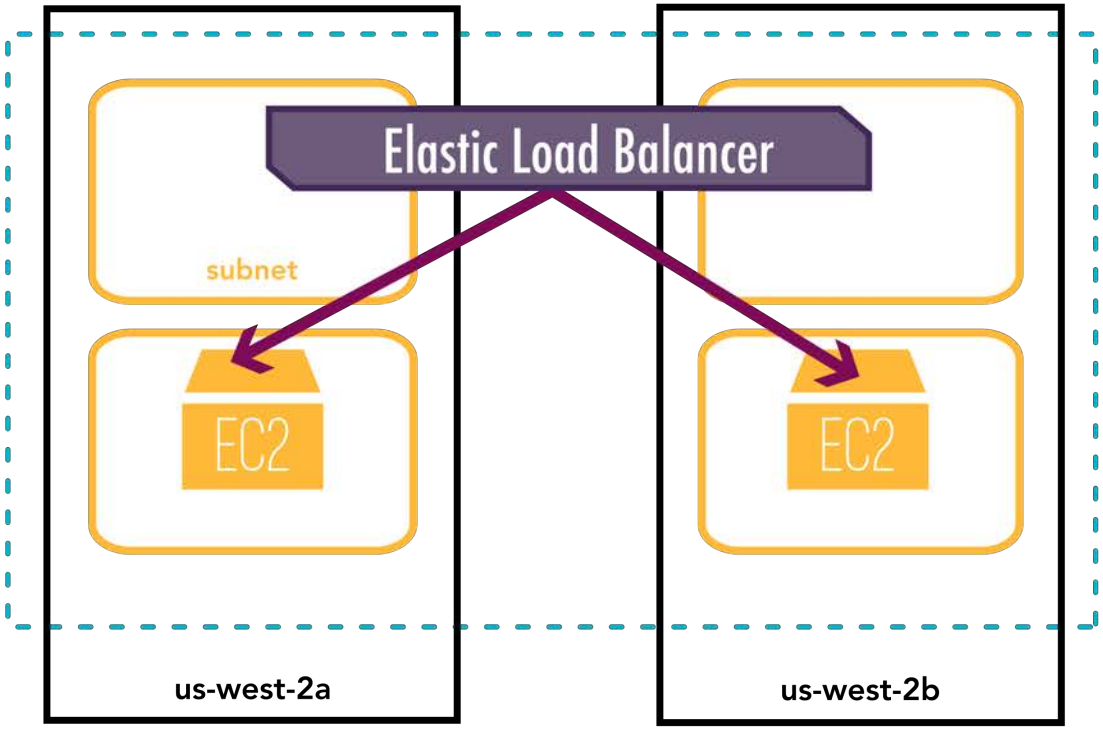
hands-on
TRAINING

Rafael Fernández (rfernandez@fi.upm.es)

HIGH AVAILABILITY AND FAULT TOLERANCE

High availability and fault tolerance

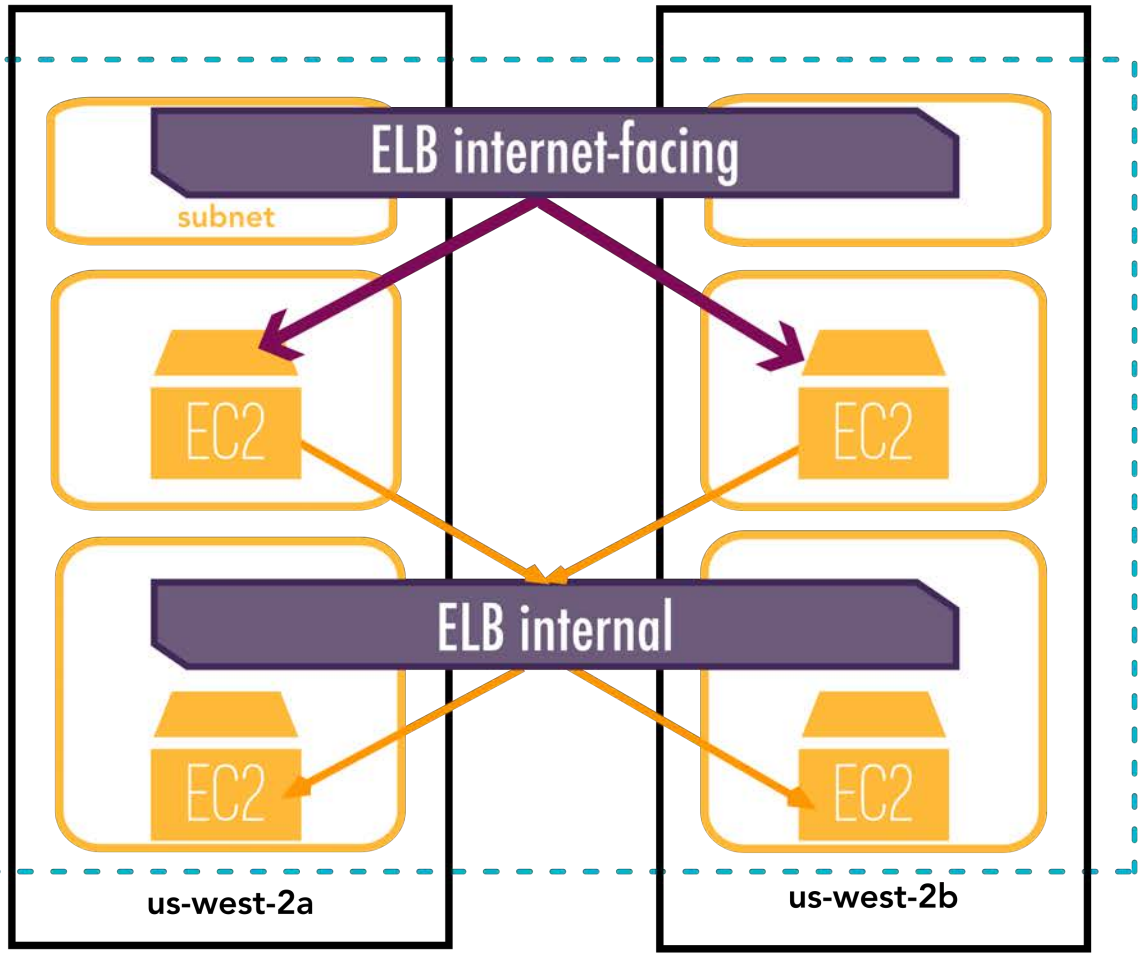
Introduction to Elastic Load Balancing (ELB)



- Inherently highly-available and fault-tolerant
- Spans region, use every AZ
- Provides DNS endpoint
- DO NOT rely on IP addresses
- €/hour + €/GB bandwidth

High availability and fault tolerance

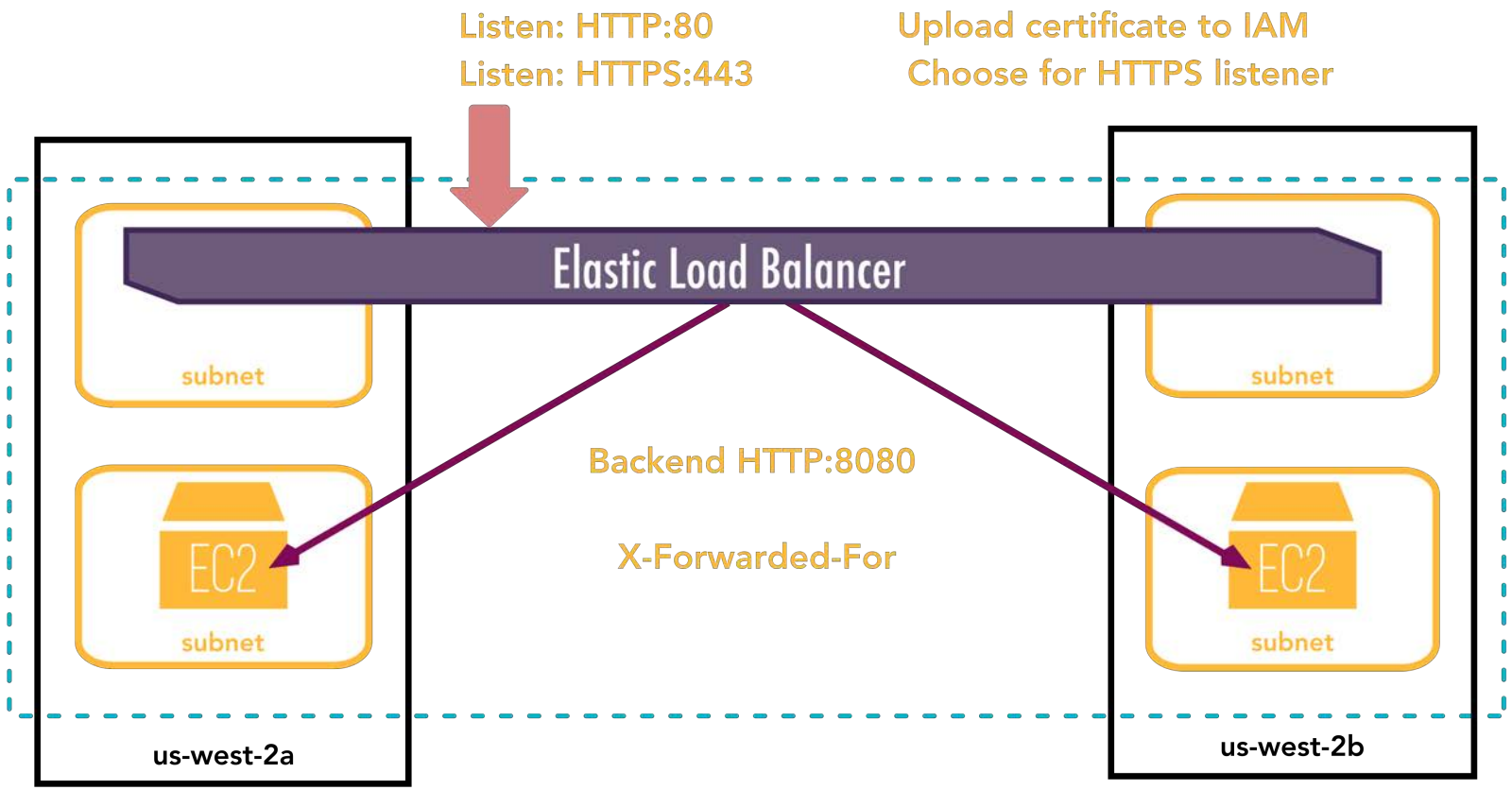
ELB Scheme



- Functionally identical
- **Internet-facing: public IP**
  - Best for public available services
- **Internal: private IP**
  - Best for private services

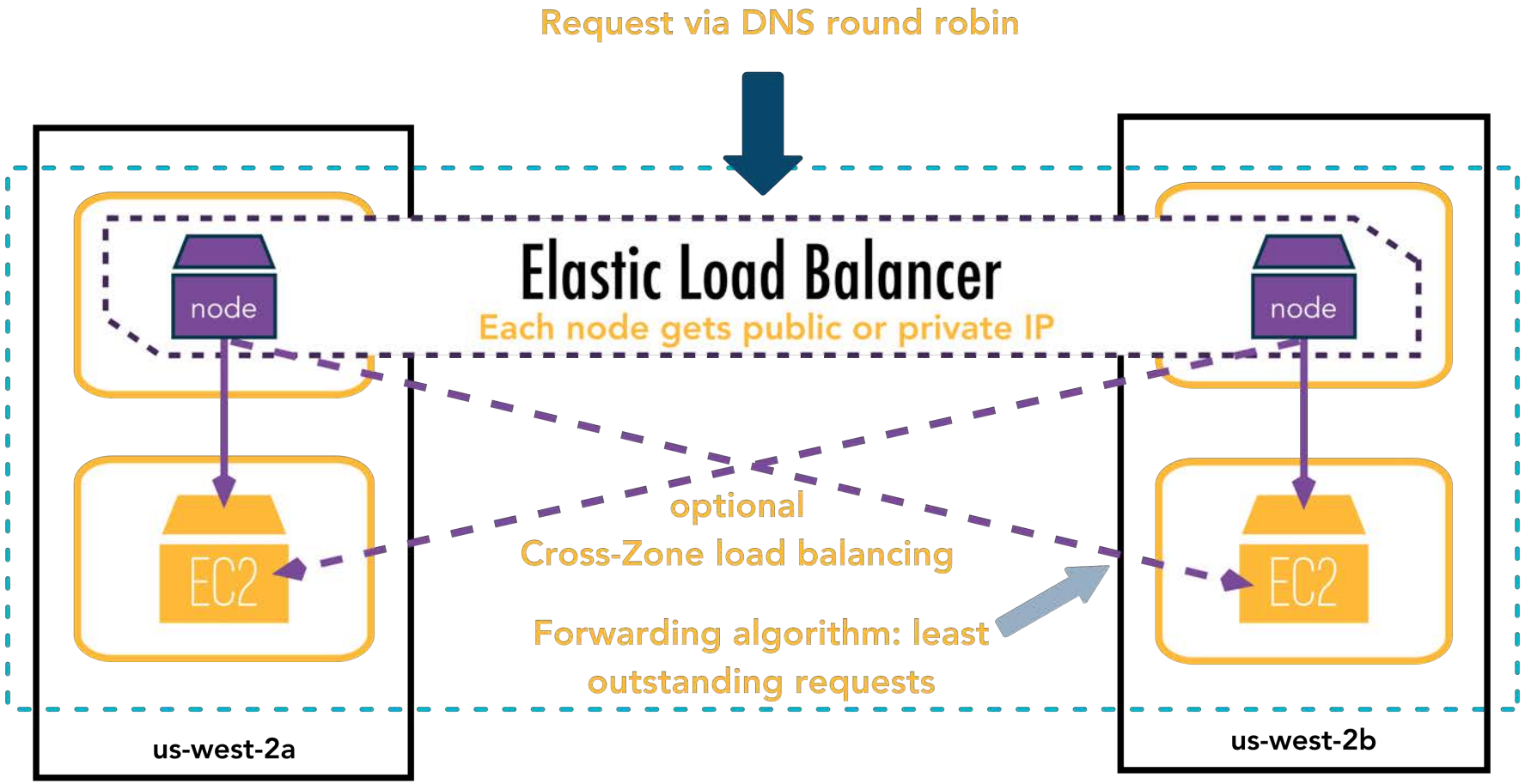
High availability and fault tolerance

Listeners and SSL certificates



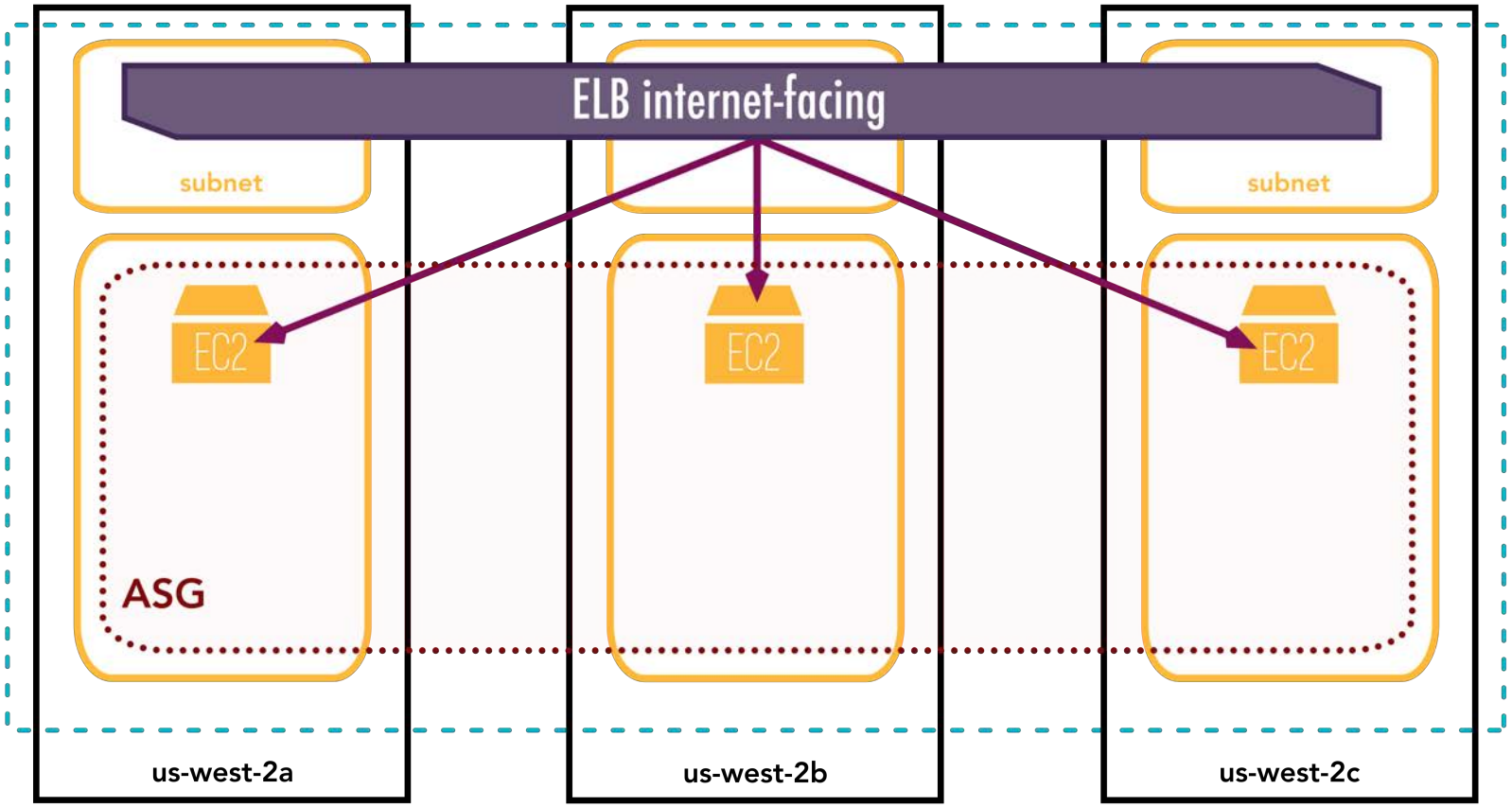
High availability and fault tolerance

Load balancing algorithms



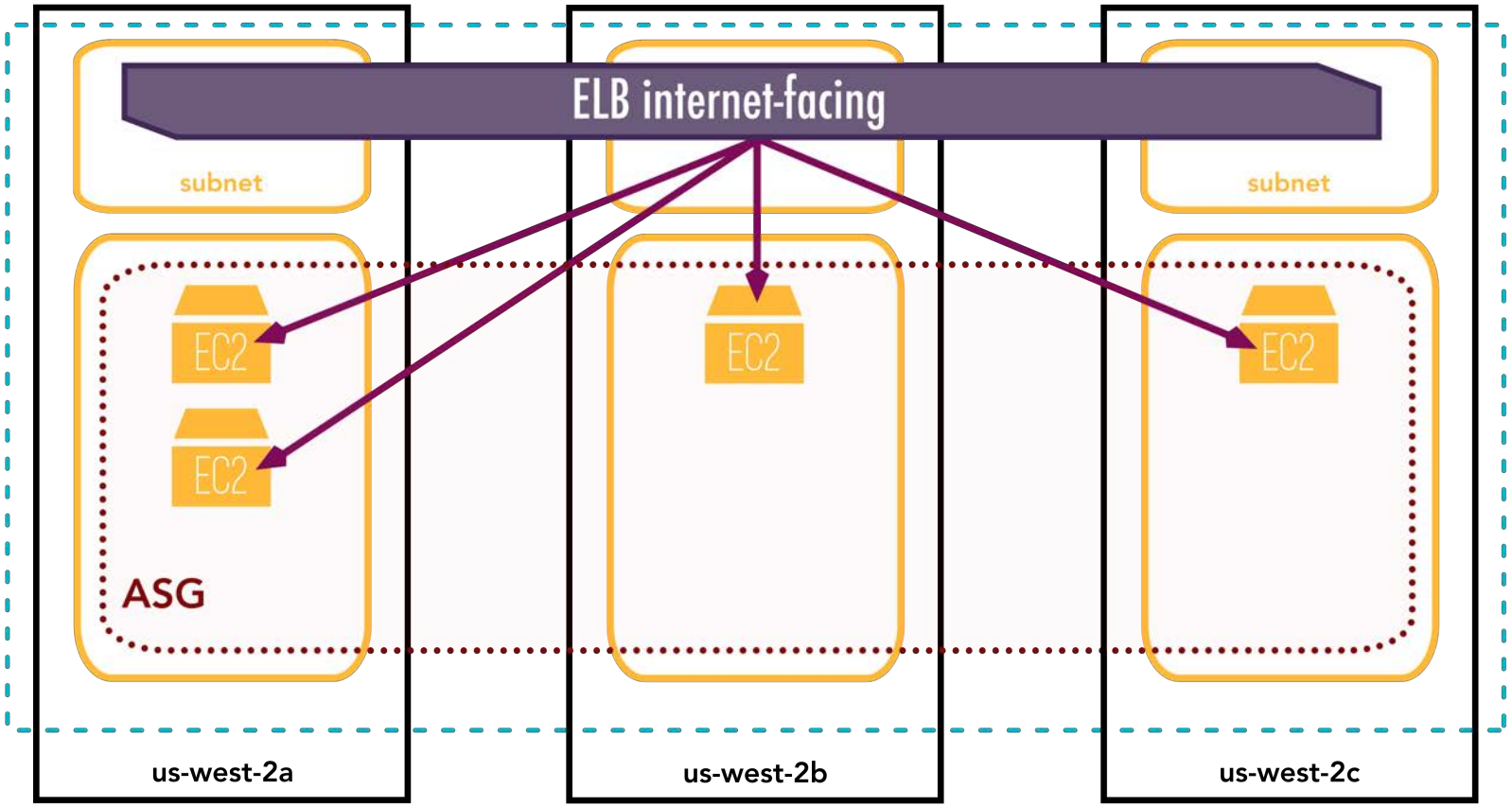
High availability and fault tolerance

Introduction to Auto-Scaling



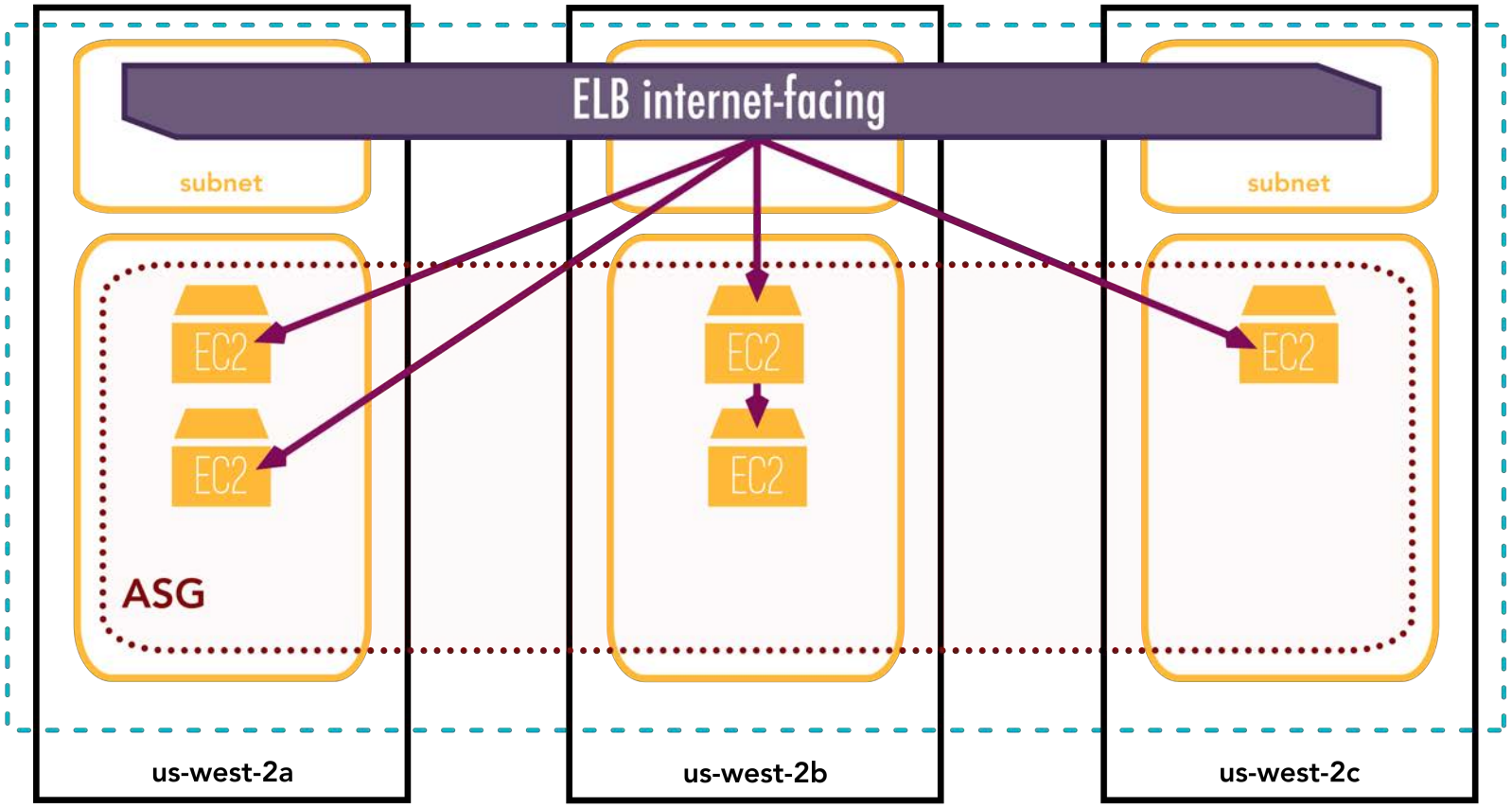
High availability and fault tolerance

Introduction to Auto-Scaling



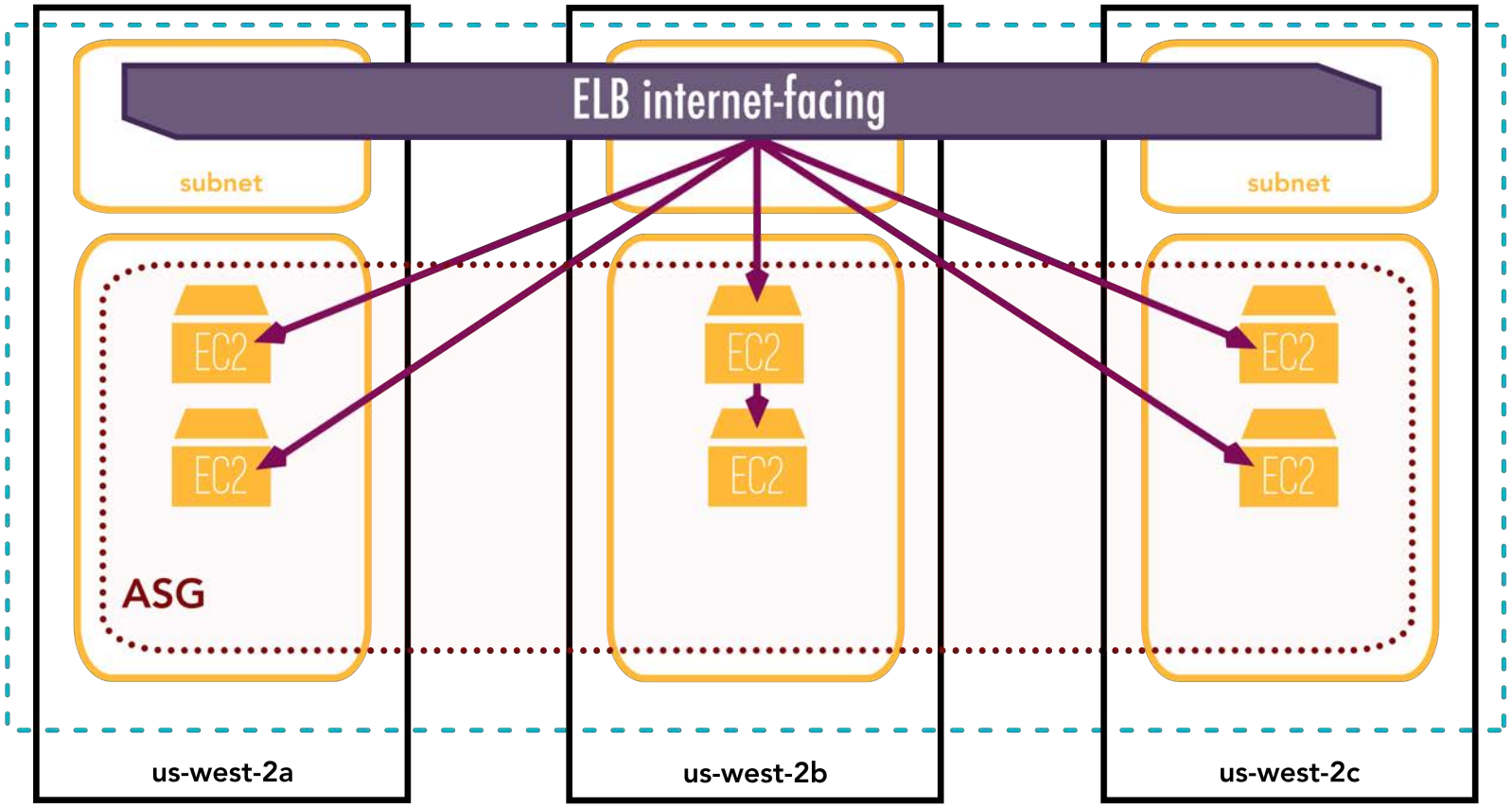
High availability and fault tolerance

Introduction to Auto-Scaling



High availability and fault tolerance

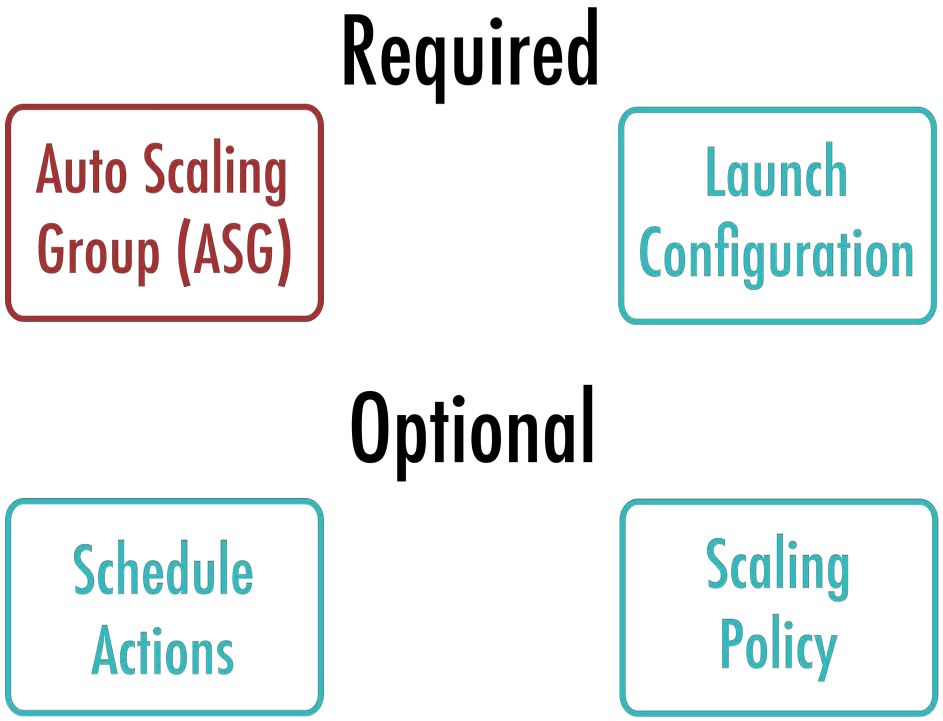
Introduction to Auto-Scaling



High availability and fault tolerance

Introduction to Auto-Scaling

- Replaces failed instances
- Change capacity according to load
- Maintain fixed size fleet
- Works with Amazon CloudWatch
- Events
  - SNS
  - Lambda



High availability and fault tolerance

Introduction to Auto-Scaling

Rafael Fernández (rfernandez@fi.upm.es)

Schedule Action

- Intervals
- One-time event
- Useful for known events
- Predictable loads

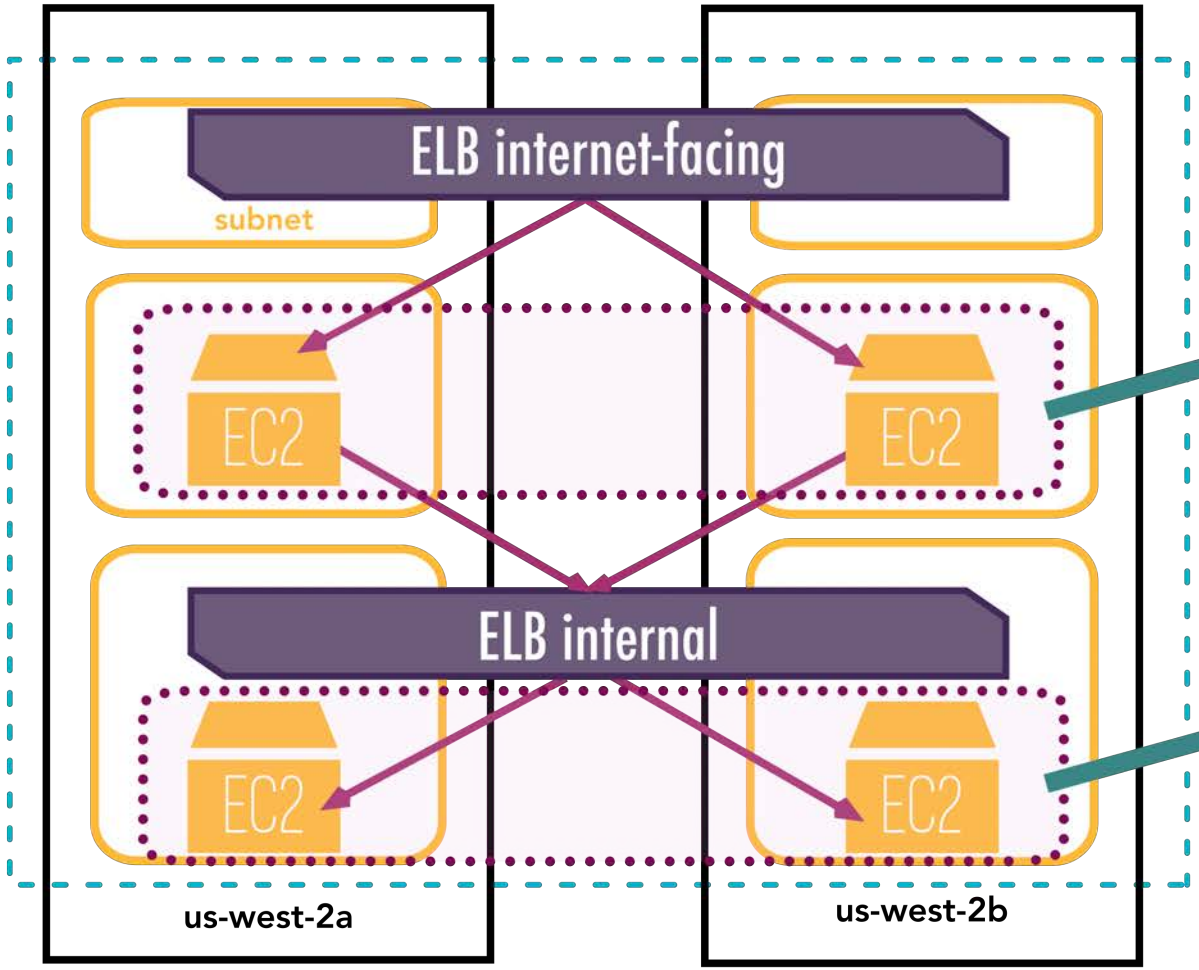
Optional

Scaling Policy

- Scale based on load
- Useful for unpredictable load
- Need to be triggered
  - Via CloudWatch alarms
  - Manually
  - Programatically

High availability and fault tolerance

Self-Healing services



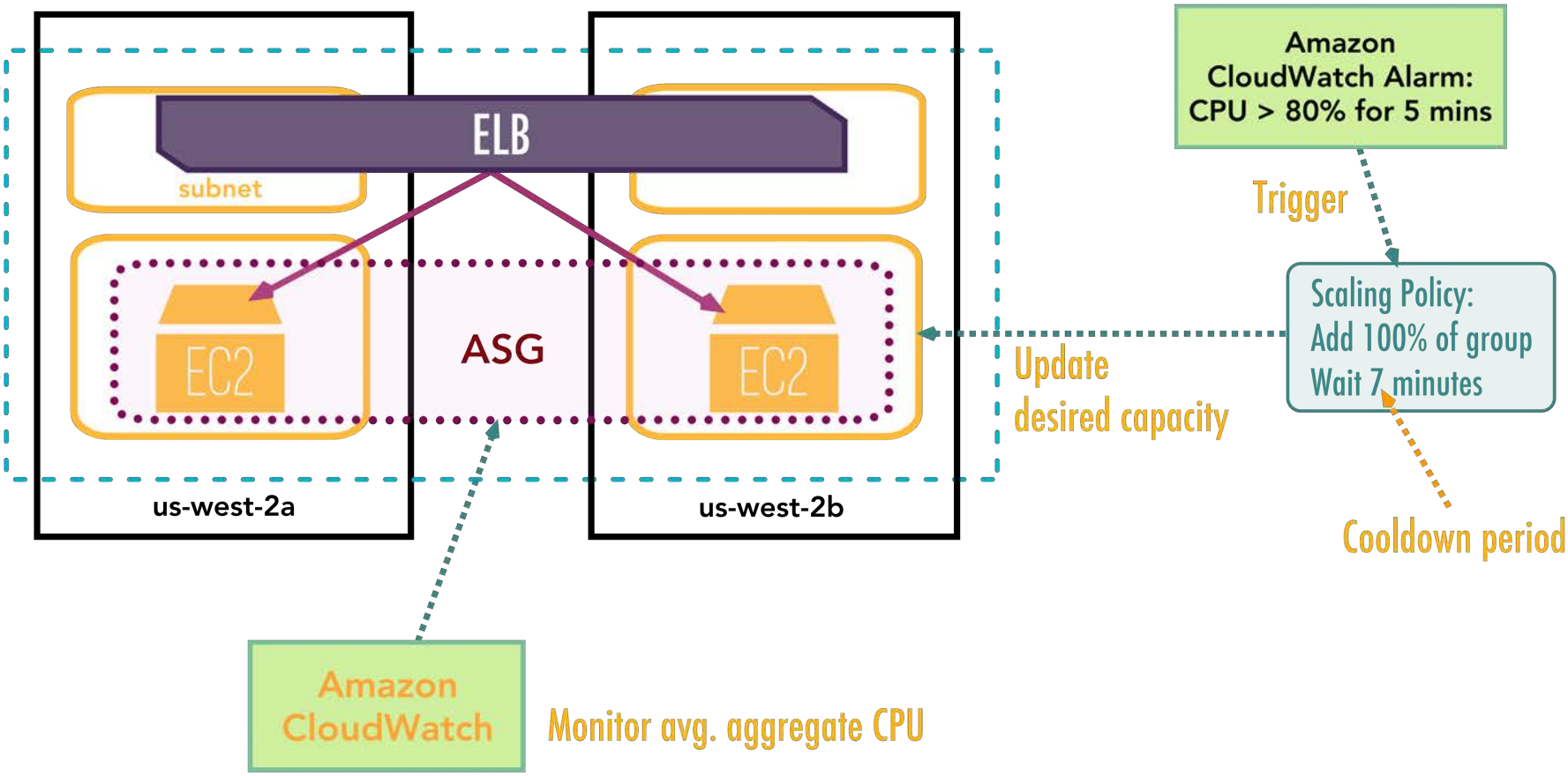
Launch configuration
WebServer AMI
c4.large
Userdata:
1. Environment variables
2. Start process

Launch configuration
Application AMI
m4.largeseries
Userdata:
1. Update code base
2. Start process
3. Notify

Rafael Fernández (rfernandez@fi.upm.es)

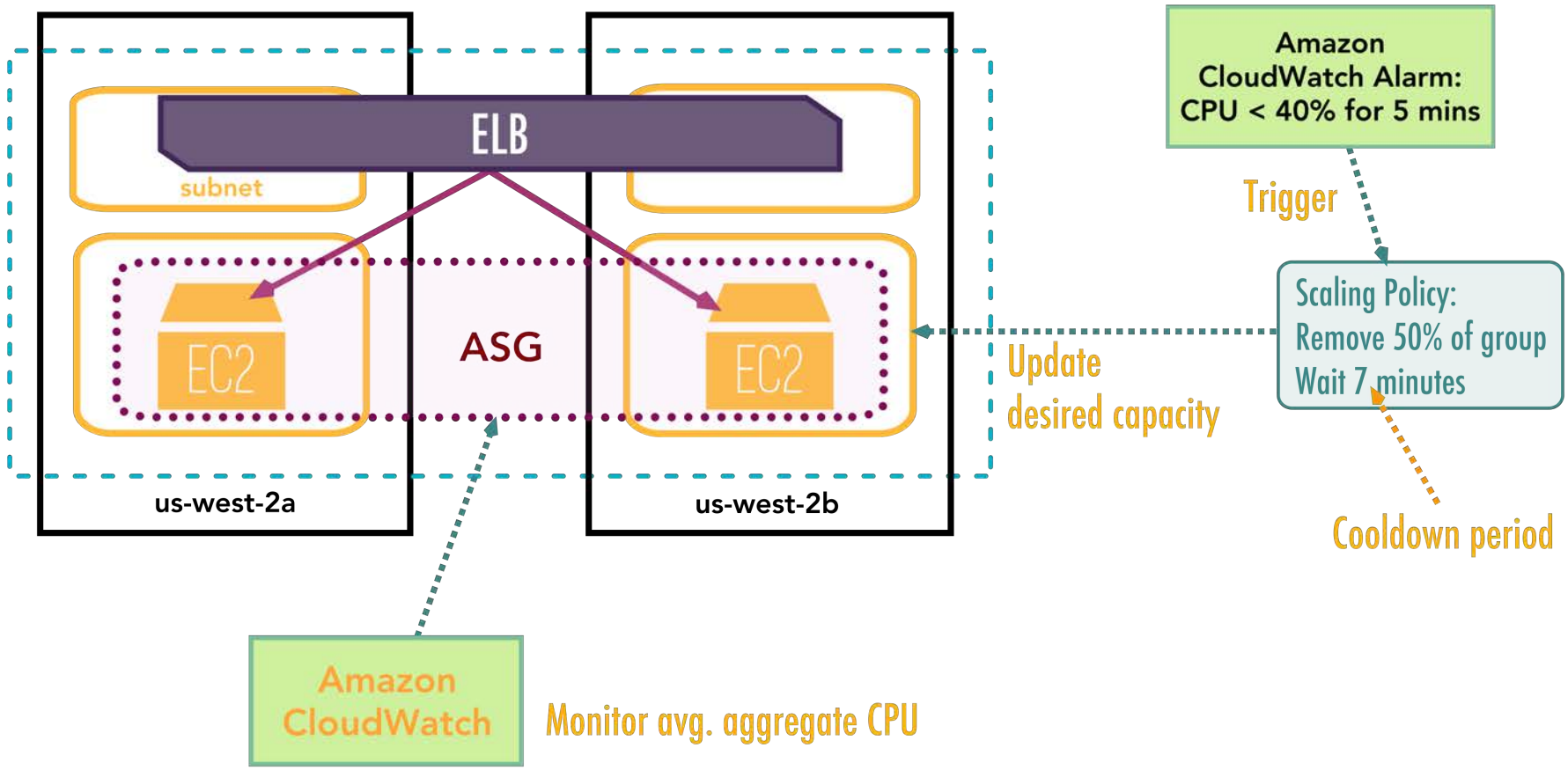
High availability and fault tolerance

Demand-based Scaling Out



High availability and fault tolerance

Demand-based Scaling In



HA and Fault Tolerance in AWS

Hands-on activity

- Let's see how to:
  - Create an Elastic Load Balancer
  - Create an Auto Scaling Group

Rafael Fernández (rfernandez@fi.upm.es)