AWS Certified Solutions Architect Crash Course

This Class...

Assumes that You...

- Have had previous fundamentals training
- Or have some experience
- Are currently familiar with
  - Core AWS services
  - Cloud computing concepts
  - Multi-tier architectures
  - Software development/operations
  - Basic networking (TCP/IP, ports, etc)

This Class...

Covers

- Technical scenarios
- The Well Architected Framework
- The Exam guide
- The Exam itself
- Exam strategy
- Sample questions

Does Not Cover

- *Fundamentals*
- In-depth service details
- Specific implementations
- Every service…

Introduction

The Exam

Registering for the Exam

Taking the Exam

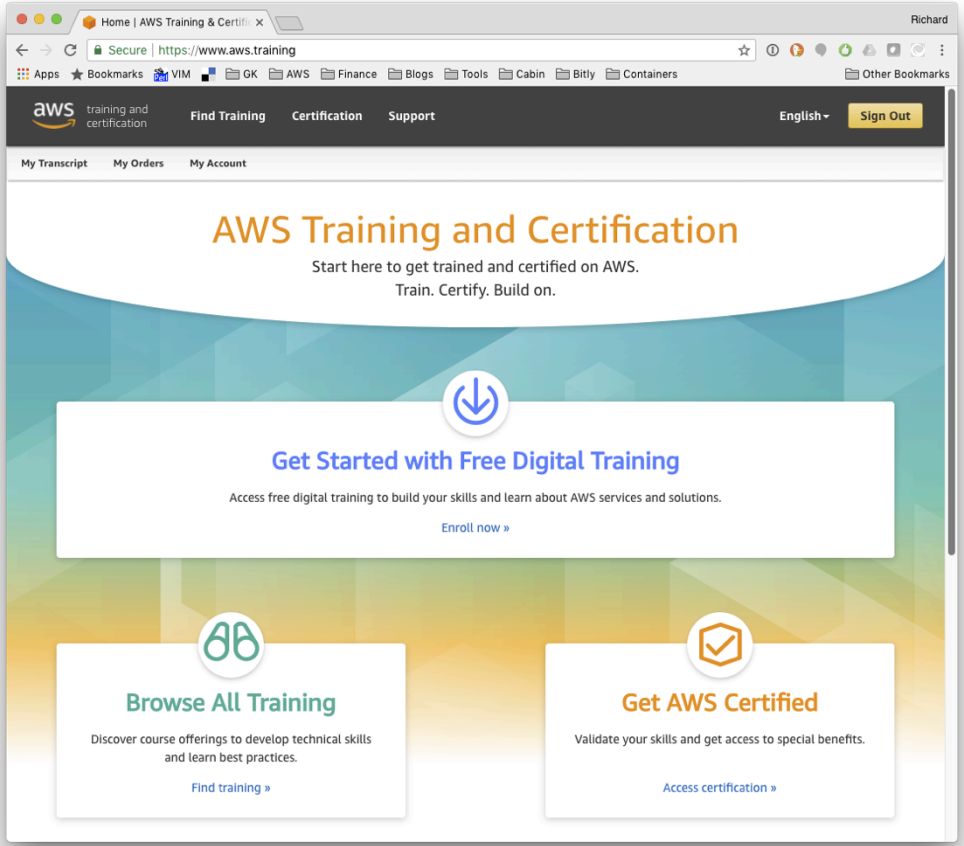
Review the Exam Guide

Best Practices

The Exam

- Multiple choice and answer
- 65 questions
- 130 minutes
- Practice exam $20
- Exam $150
- Currently English only

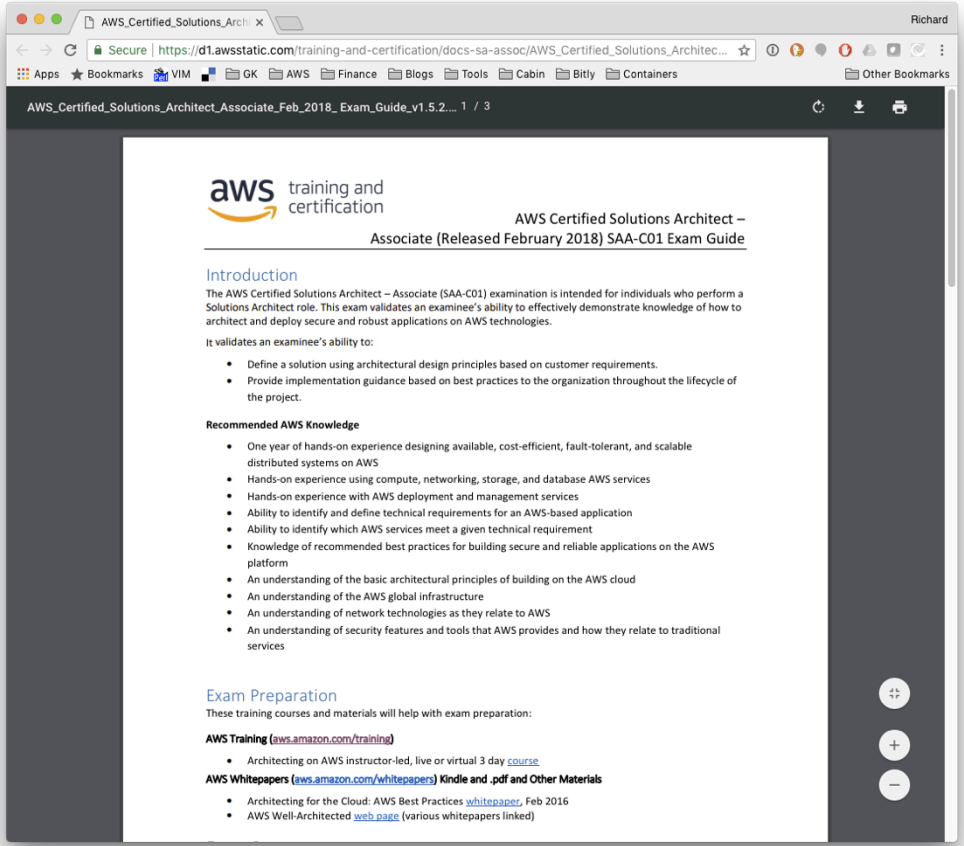
Registering for the Exam



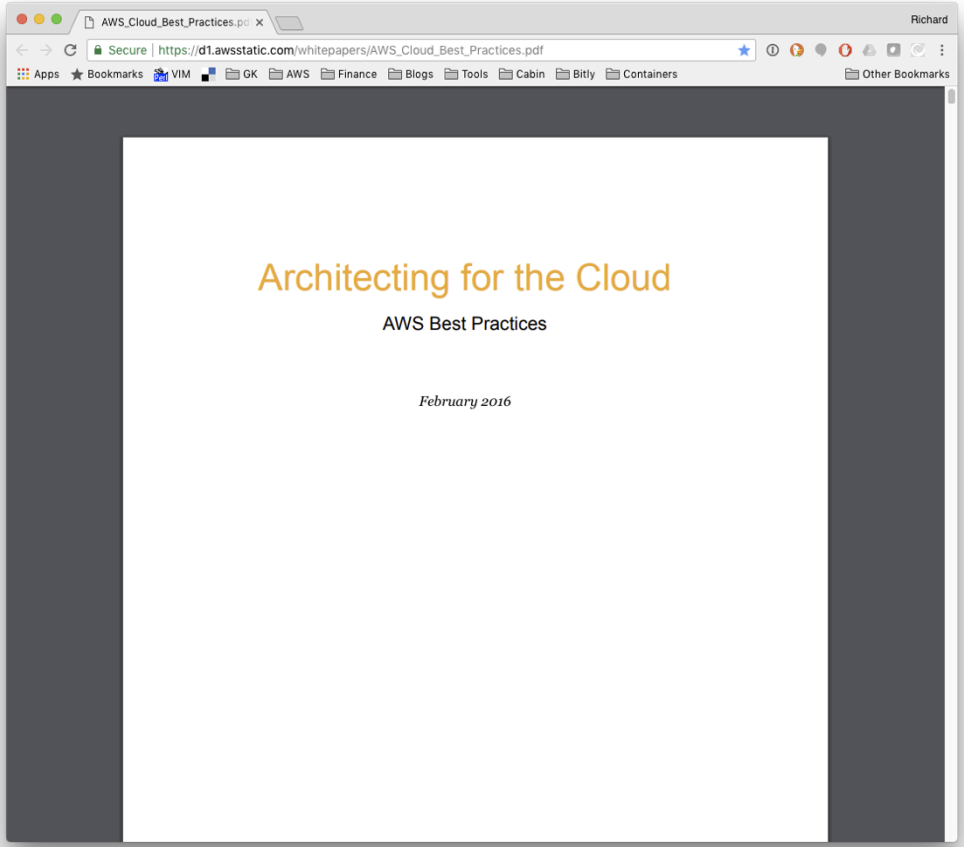
Taking the Exam

- Bring two forms of ID
- Cancellations or rescheduling:
  - >= 48 hours before, no fee
  - < 48 hours, $60 charge
  - Cannot < 24 hours of exam
- Missing the exam
  - Fee is non-refundable
  - Does not count as “failed”
- Leave belongings in a locker
- No ability to research
- Scratch paper/pencils provided
  - Cannot take with you

Review the Exam Guide



Best Practices



Best Practices

- Programmable, disposable resources
- Offload operational burdens
- Leverage built-in security

Best Practices—Design Principles

- Enable scalability
- Treat resources as disposable
- Automate everything
- Loosely couple components
- Prefer services over servers
- Choose the right database
- Avoid single points of failure
- Optimize for Cost
- Leverage caching
- Security in layers

Reliability

High Availability

Reliable Storage

Decoupling Mechanisms

Multi-Tier Architectures

Terms

reliable

adjective

consistently good in quality or performance;

able to be trusted

Terms

resilience

noun

the capacity to recover quickly from difficulties

Reliability

> “...the ability of a system to recover from infrastructure or service disruptions, dynamically acquire computing resources to meet demand, and mitigate disruptions or transient network issues.”

Service Availability

Percentage of uptime over a month, quarter, year, etc.

Availability = Normal Operation Time / Total Time

99.9% ("three nines")	8.76 hours
99.95% ("three and a half nines")	4.38 hours
99.99% ("four nines")	52.56 minutes
99.999% ("five nines")	5.26 minutes

- Consider what each application *really* needs
- Higher availability leads to
  - Higher cost
  - Higher complexity
- What is the real impact of downtime?
- Include planned maintenance

Hindrances to High Availability

- Single points of failure (SPOF)
- Lack of automation
- Lack of elasticity

Estimating Availability

MTBF

Mean Time Between Failures

MTTR

Mean Time to Recover

Availability = MTBF / (MTBF + MTTR)

MTBF = 150 days

MTTR = 1 hour

216000 / (216000 + 60)

99.97% Availability

Prepare for Failure

"Manure occureth."

- Hardware failure
- Deployment failure
- Increased load
- Invalid data "poison pill"
- Credential expiration
- Dependency failure
- Infrastructure failures

What happens when...

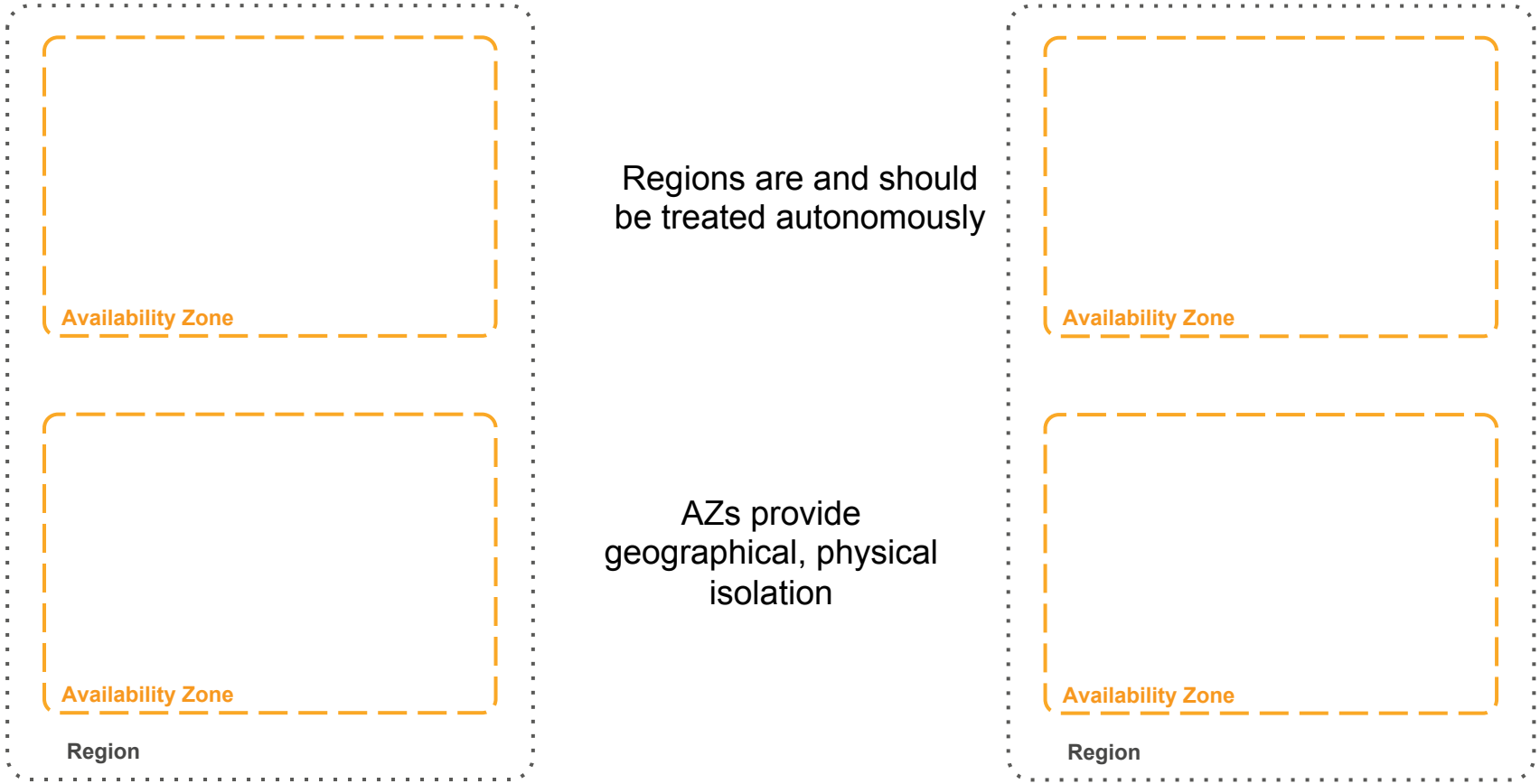
- an instance terminates?
- a disk becomes full?
- a network link is unavailable?
- data transmission slows?
- a data center is offline?
- during a DDoS attack?
- a security group rule is changed?
- authorization is revoked?

Design for Availability

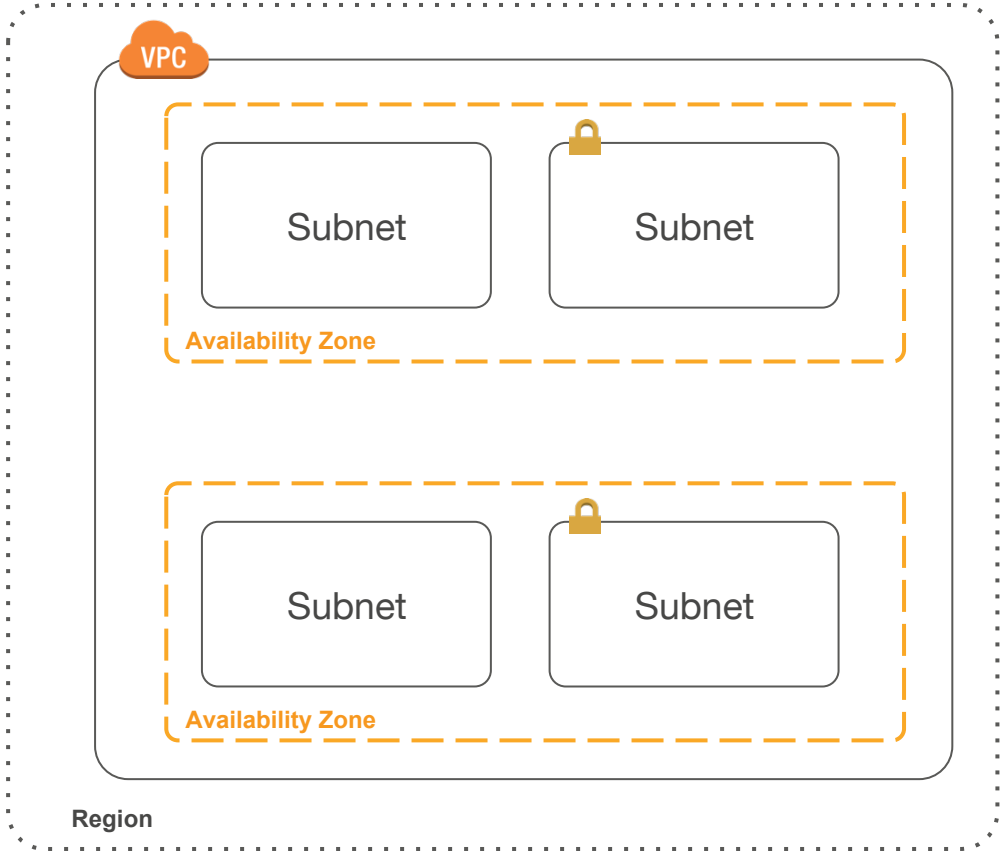
Five practices to improve availability

- Fault isolation zones
- Redundancy
- Microservice architecture
- “Recovery-Oriented Computing”
- Distributed systems best practices

Fault Isolation Zone



Architecture begins with Amazon VPC

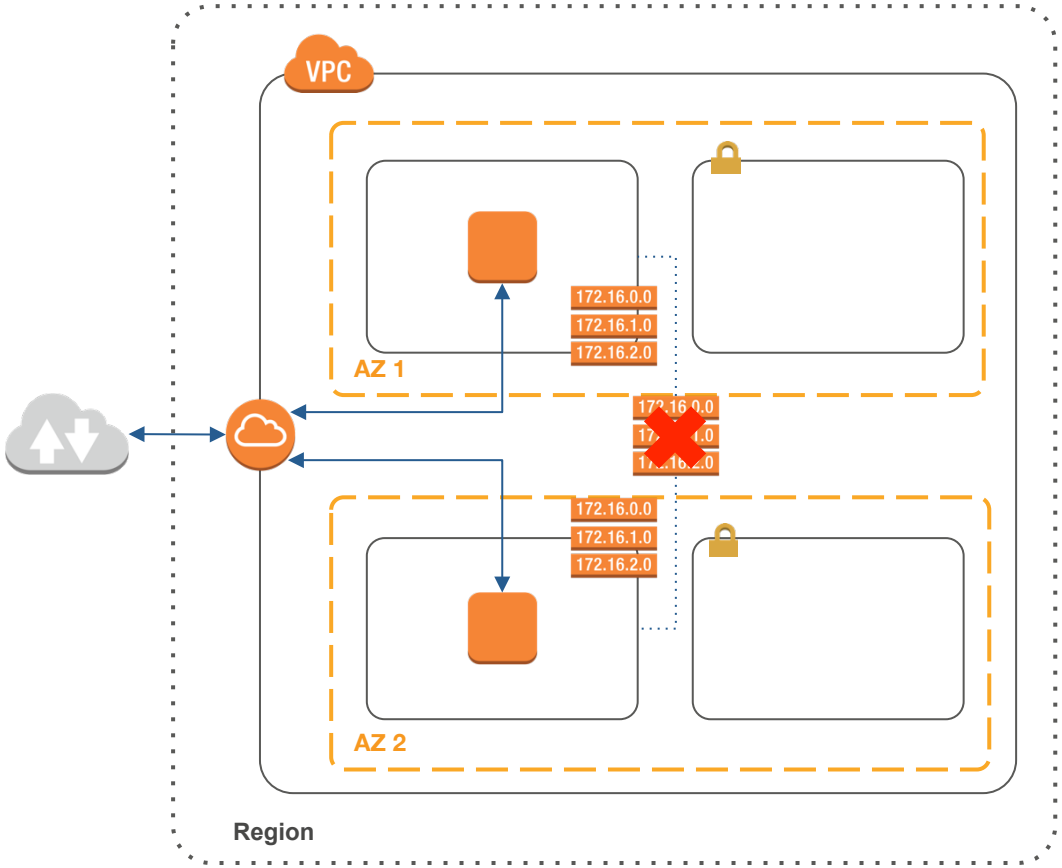


- VPC provides logical isolation
- Subnets provide
  - Isolation between tiers
  - Logical mapping to physical zone

VPC & Subnet Considerations

- IP address space for >1 VPC/Region
- VPC peering
  - Cross account
  - Cross region
- Plan for scalability

Resiliency of Internet Connectivity

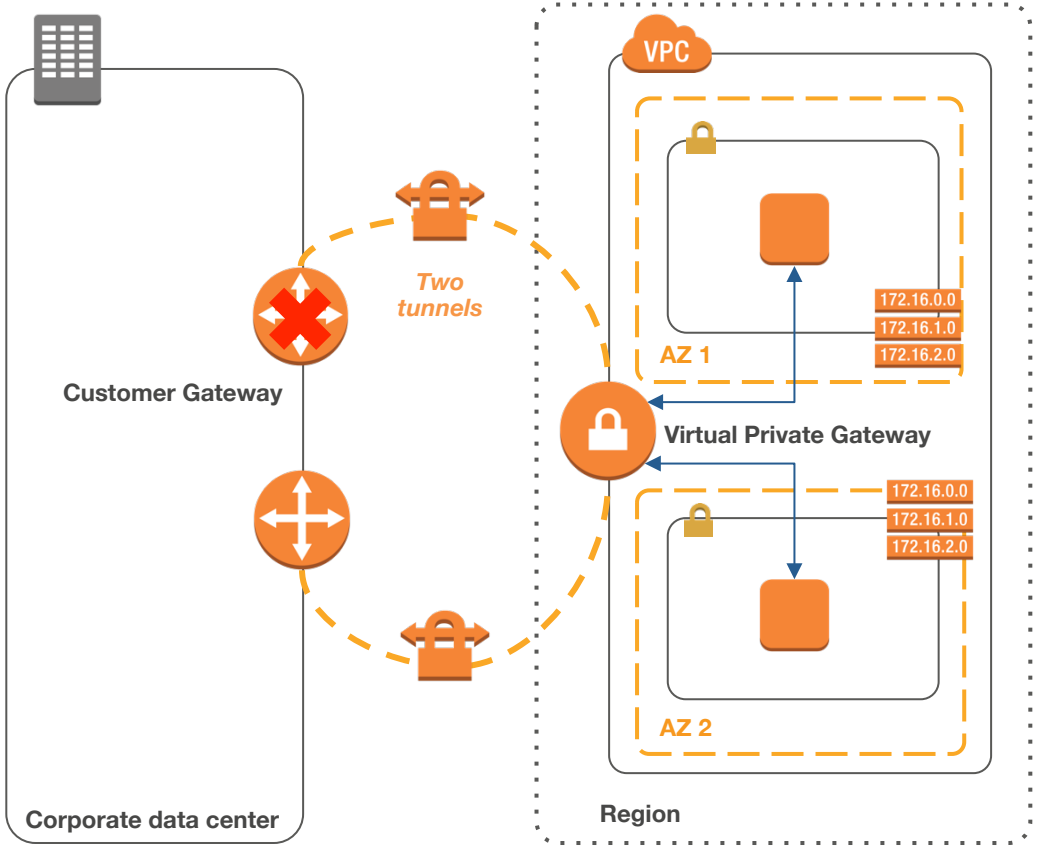


- IGW is horizontally scaled, redundant, and highly available
- What happens when
  - route table disassociated
  - or misconfigured?

High availability achieved through

- Independent route tables

Resiliency of VPN Connectivity

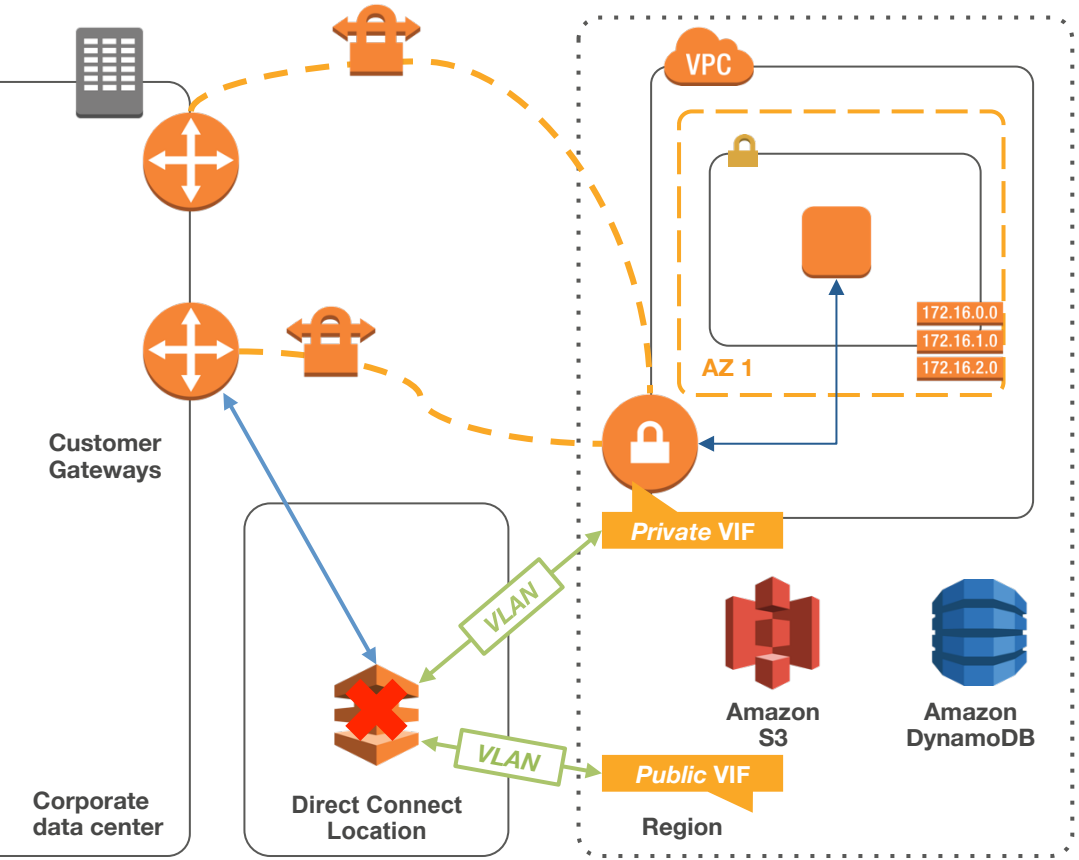


- VGW powered by physically distinct routers in distinct facilities
- What happens when customer router fails?
- What happens when customer internet connection fails?

High availability achieved through

- multiple customer routers
- configured via BGP
- redundant internet connections

Direct Connect HA via VPN

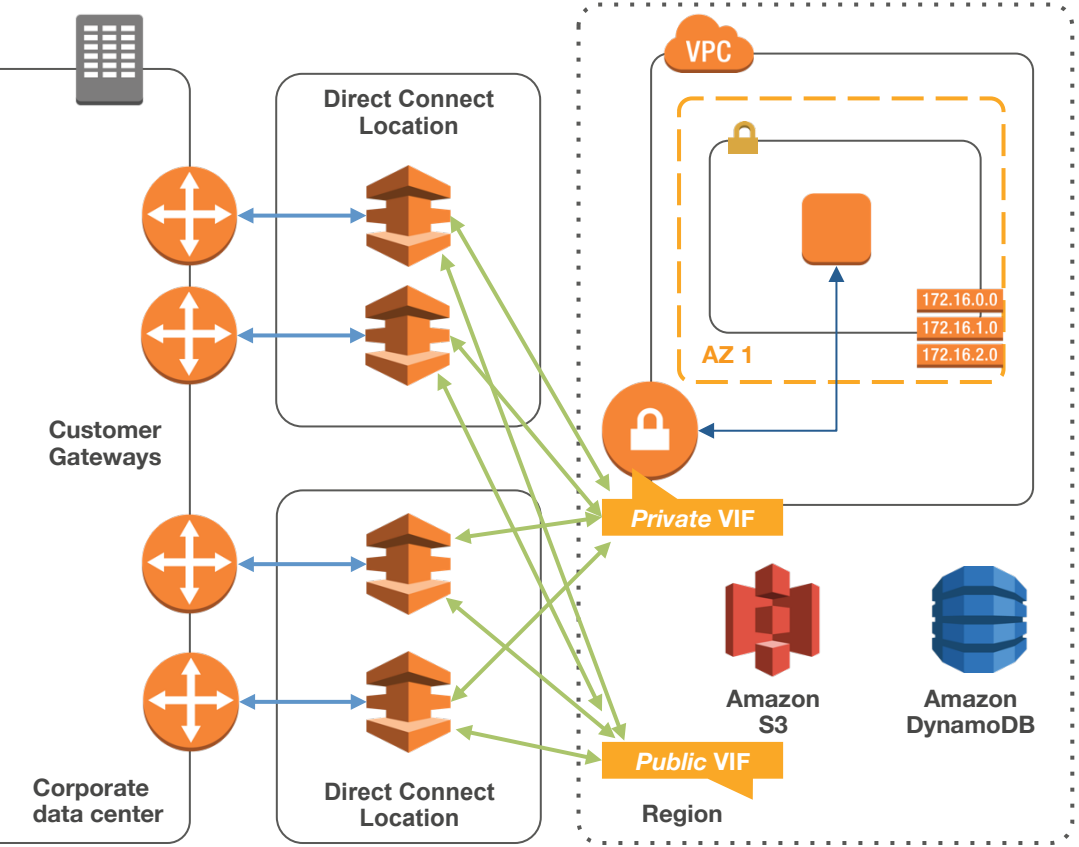


- What happens when
  - customer router fails?
  - Direct connect fails?

High availability achieved through

- failover to VPN
- configured with BGP

Direct Connect HA via Redundant Connections



- What happens when
  - Direct Connect facility fails?
  - Provider network fails?

High availability achieved through

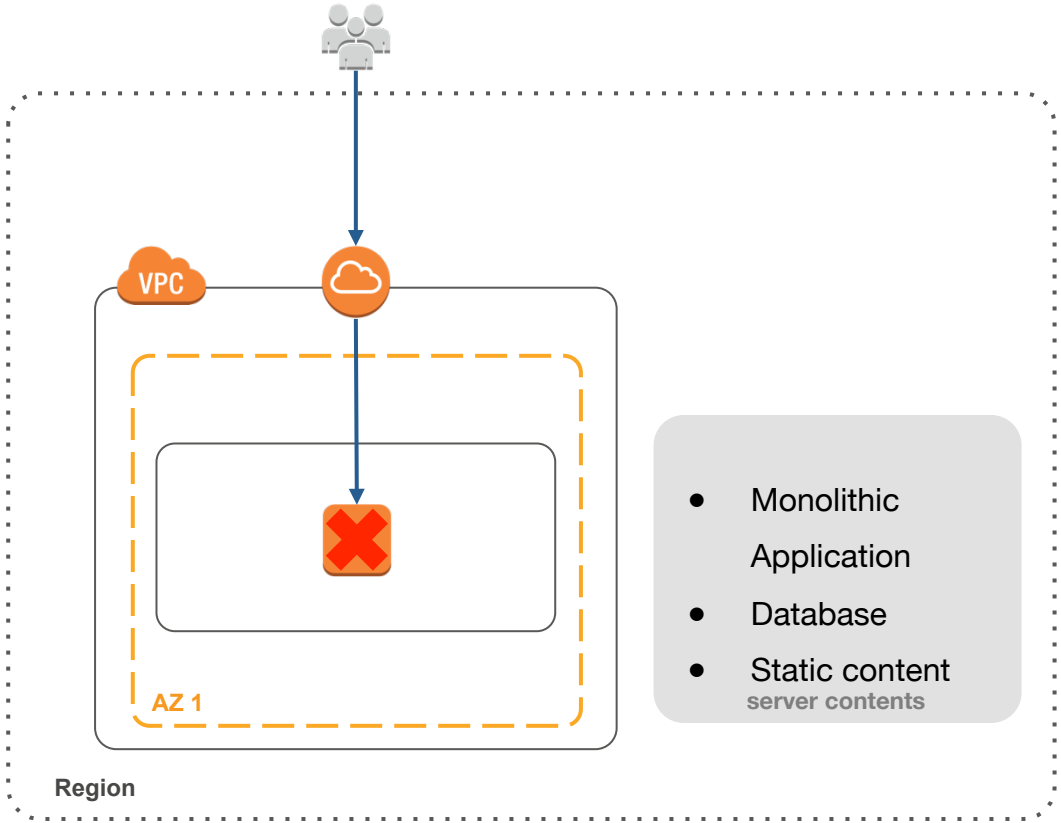
- multiple customer routers
- redundant
  - Direct Connect connections
  - facilities
  - providers

Application Resiliency

> “One of the bedrock principles for service design in AWS is the avoidance of single points of failure ... [We] build software and systems that use multiple Availability Zones and are resilient to failure of a single zone...single compute node, single storage volume, or single instance of a database.”

Reliability Pillar: AWS Well-Architected Framework (AWS Whitepaper) (Kindle Location 413). Amazon Web Services. Kindle Edition.

Application Resiliency



What happens when machine fails?

Availability depends on time to

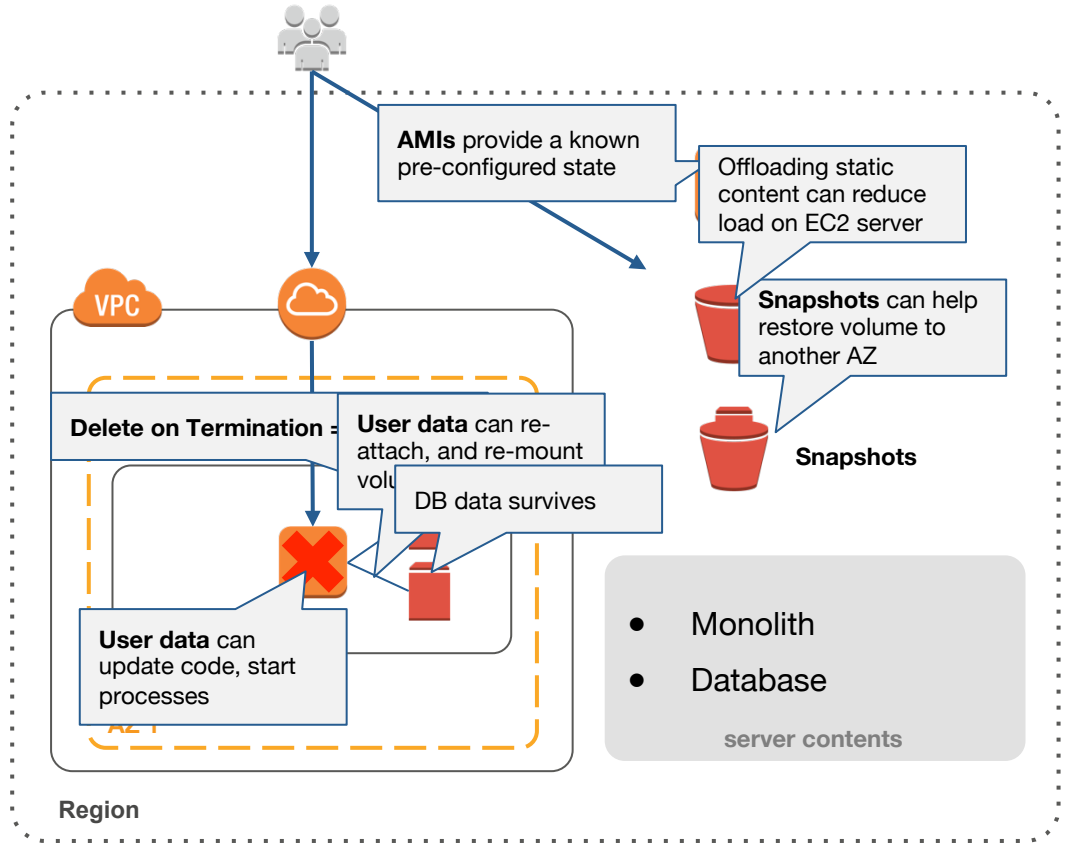
- replace machine
- recover data
- install dependencies
- configure/start processes

Recovery time is

- Increased via manual operations
- Decreased via automation

WHAT ABOUT THE DATA?!

Application Resiliency



How do we recover from

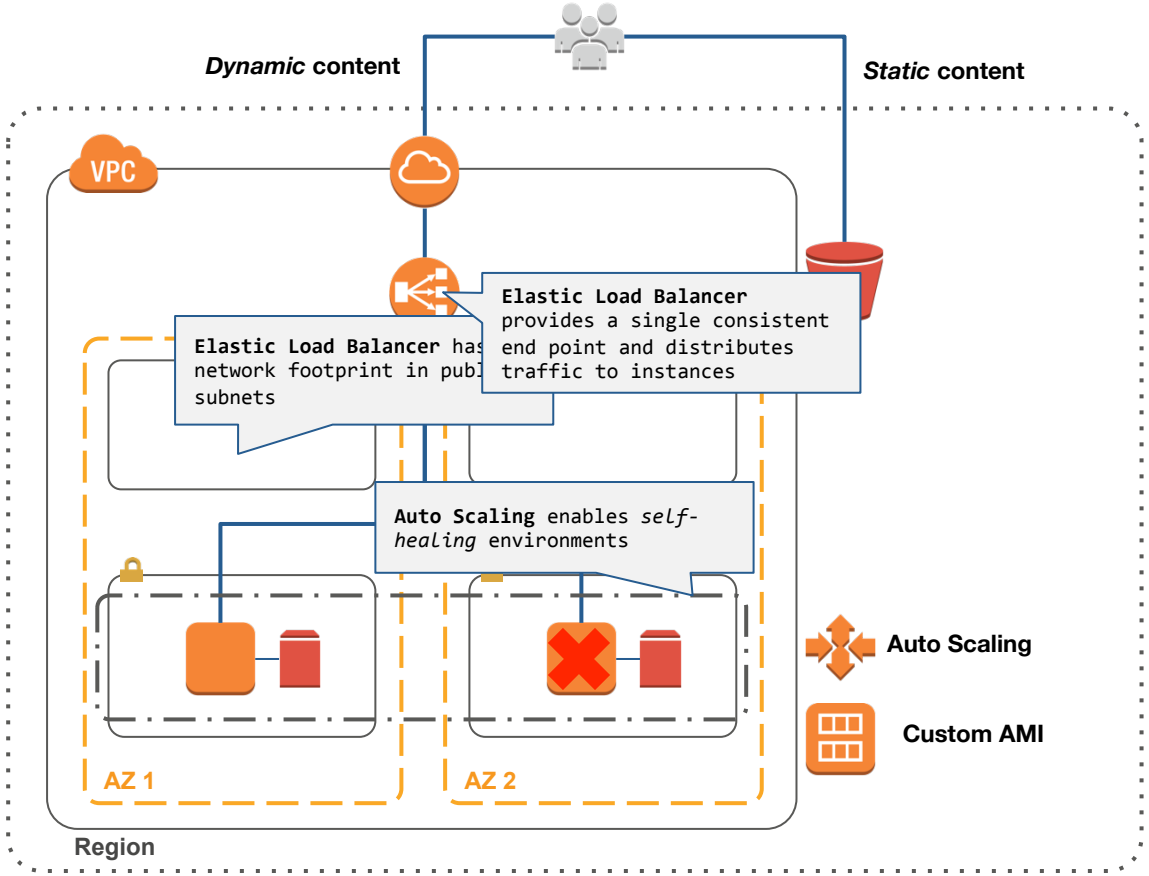
- application process failing?
- instance failing?
- data volume deletion?
- AZ failure?

AMIs provide a preconfigured state

User data provides bootstrapping

Snapshots provide increased durability

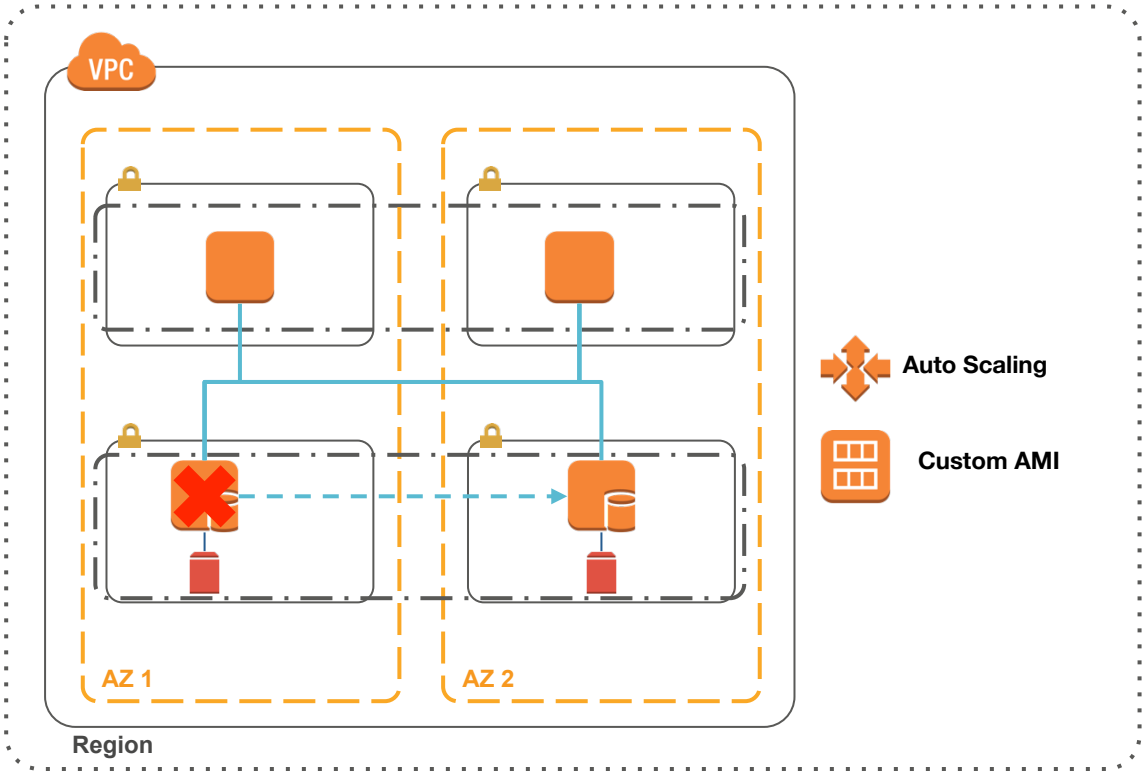
Application Resiliency



Availability improved through

- Fault isolation via multiple AZs
- Redundant components via multiple instances
- Recovery oriented computing via Auto Scaling

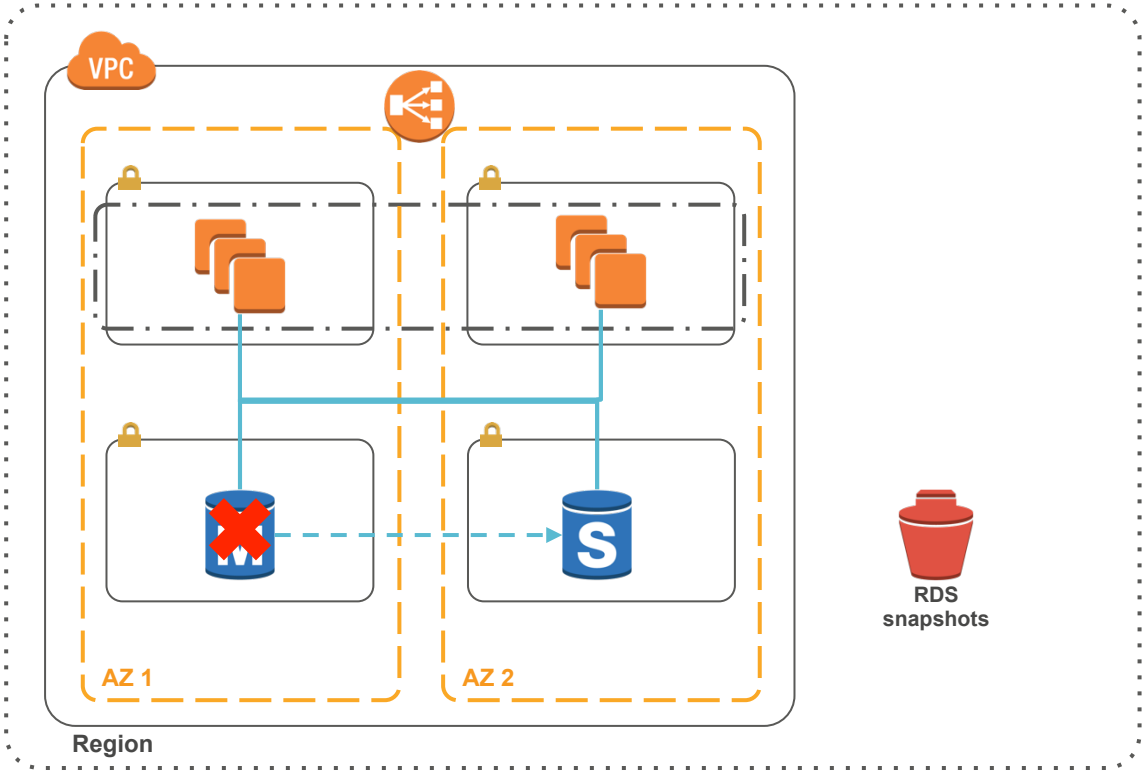
Database Resiliency



What does DIY on EC2 really entail?

- full operational burden
- ***customer*** responsible for
  - OS patches
  - DB engine updates
  - OS & DB security
  - Failover
  - Backups
  - Replication

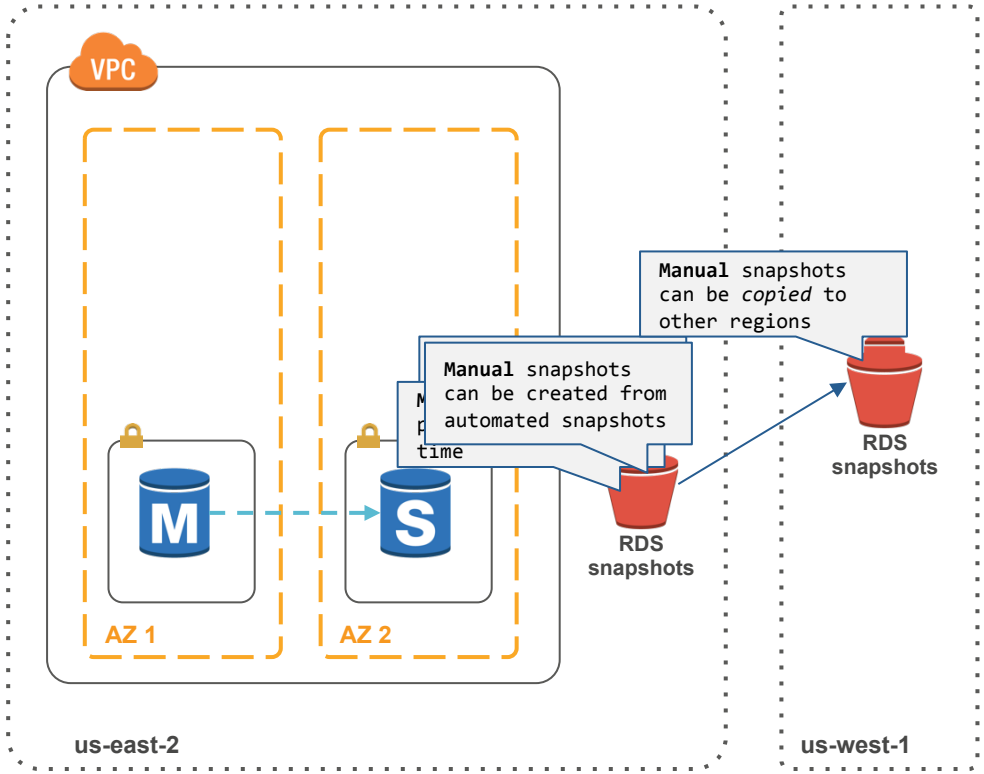
Database Resiliency with RDS



What are the benefits to RDS?

- little operational burden
- ***AWS*** responsible for
  - OS patches
  - DB engine updates
  - OS & DB security
  - Failover
  - Backups
  - Replication
- Customer focuses on
  - Application
  - Queries
  - Data

Data Durability



Greater durability achieved through

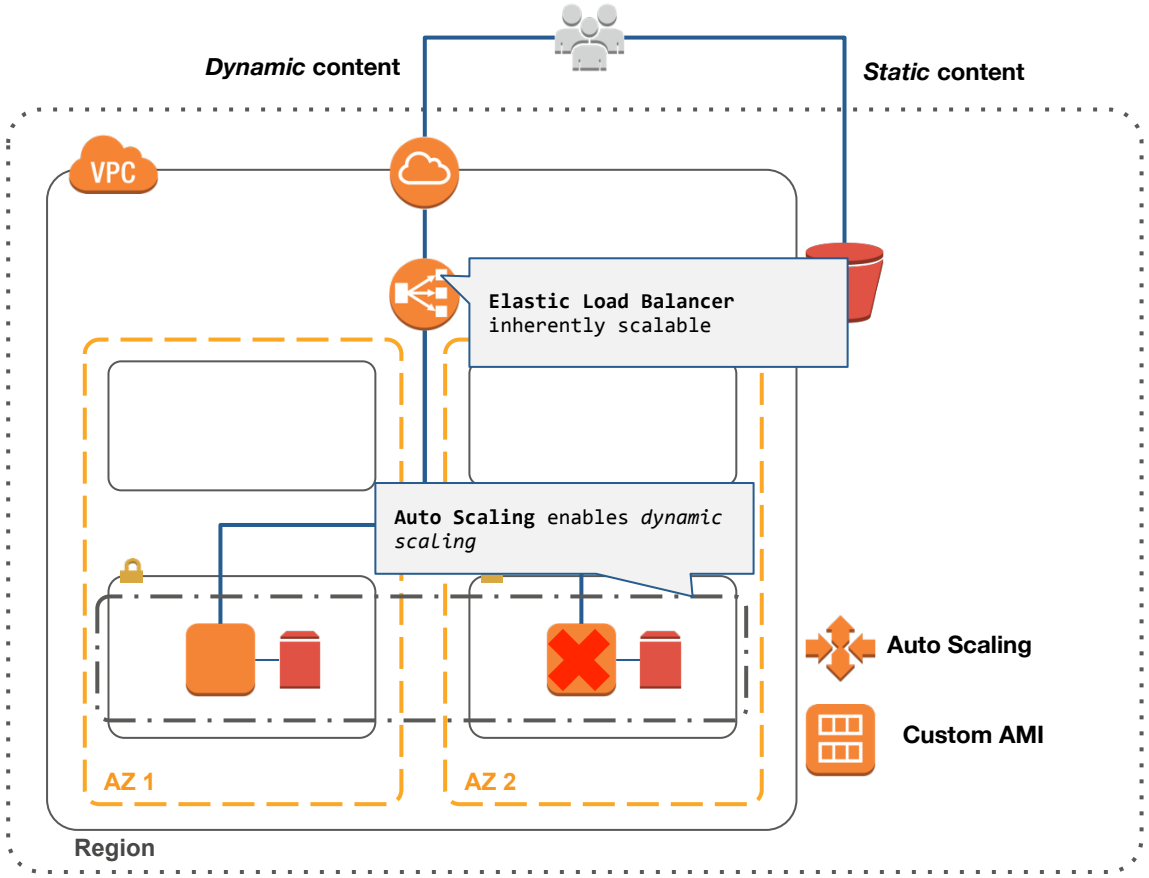
- leveraging multiple regions
- copying snapshots cross-region

Snapshots can also be shared *across accounts*.

Capacity

> “Never run an application of notable importance on single machine.”

Capacity



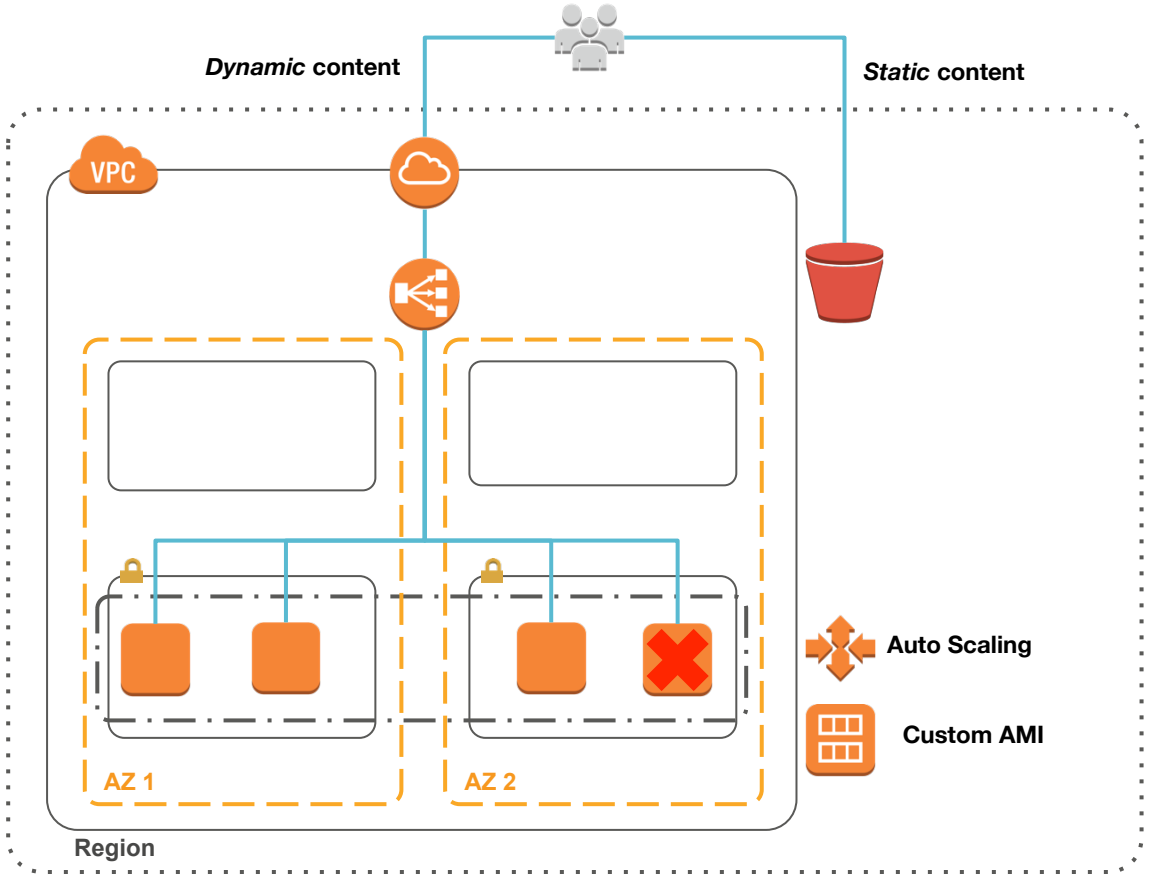
What if an instance or AZ fails during peak demand?

Can remaining servers handle the current load?

One instance or AZ failing would result in 50% loss of capacity.

With only two, each one would need to meet 100% of capacity.

Capacity



What if during peak demand…

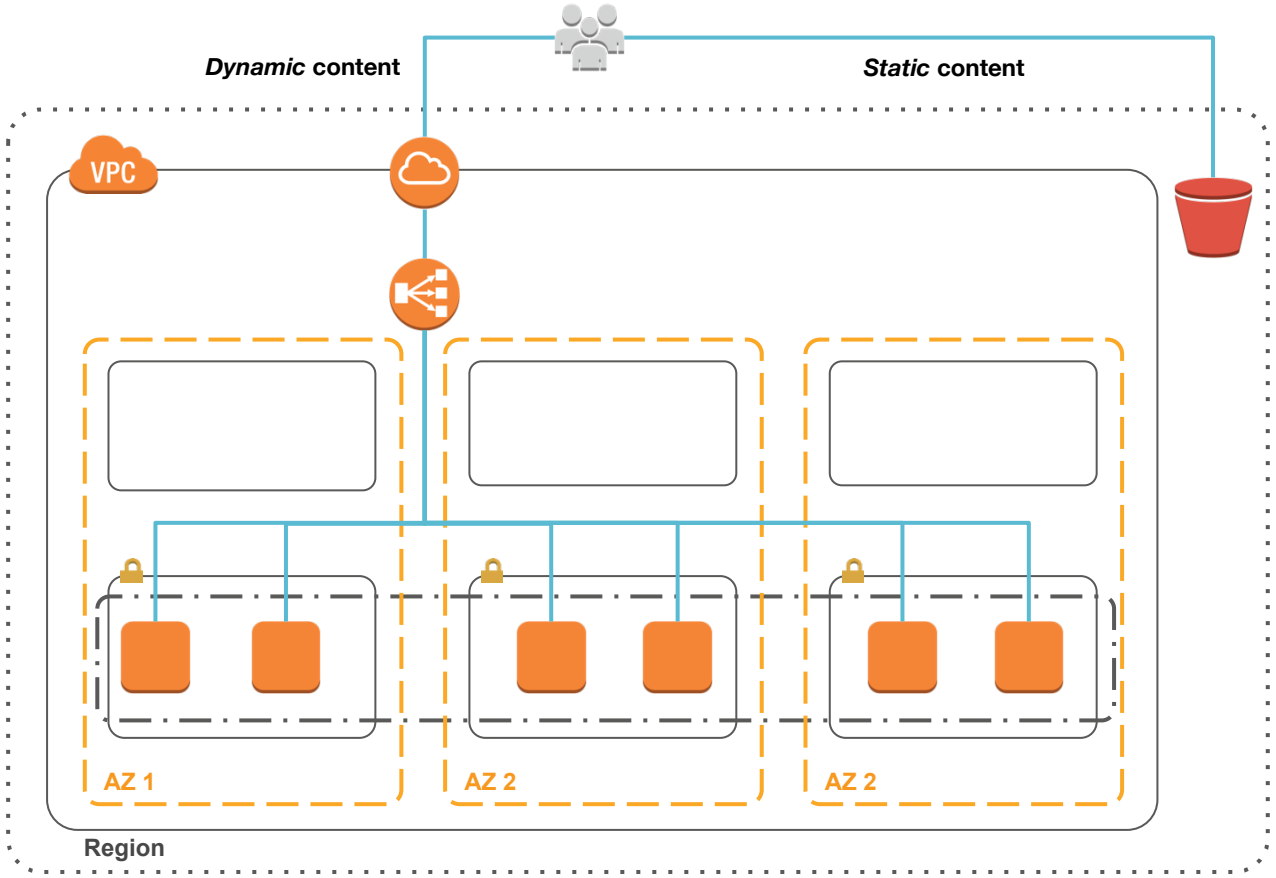
- an *instance* fails?

  25% loss in capacity

- an *AZ* fails?

  50% loss in capacity!

Capacity



What if during peak demand…

- an *instance* fails?

  16% loss in capacity

- an *AZ* fails?

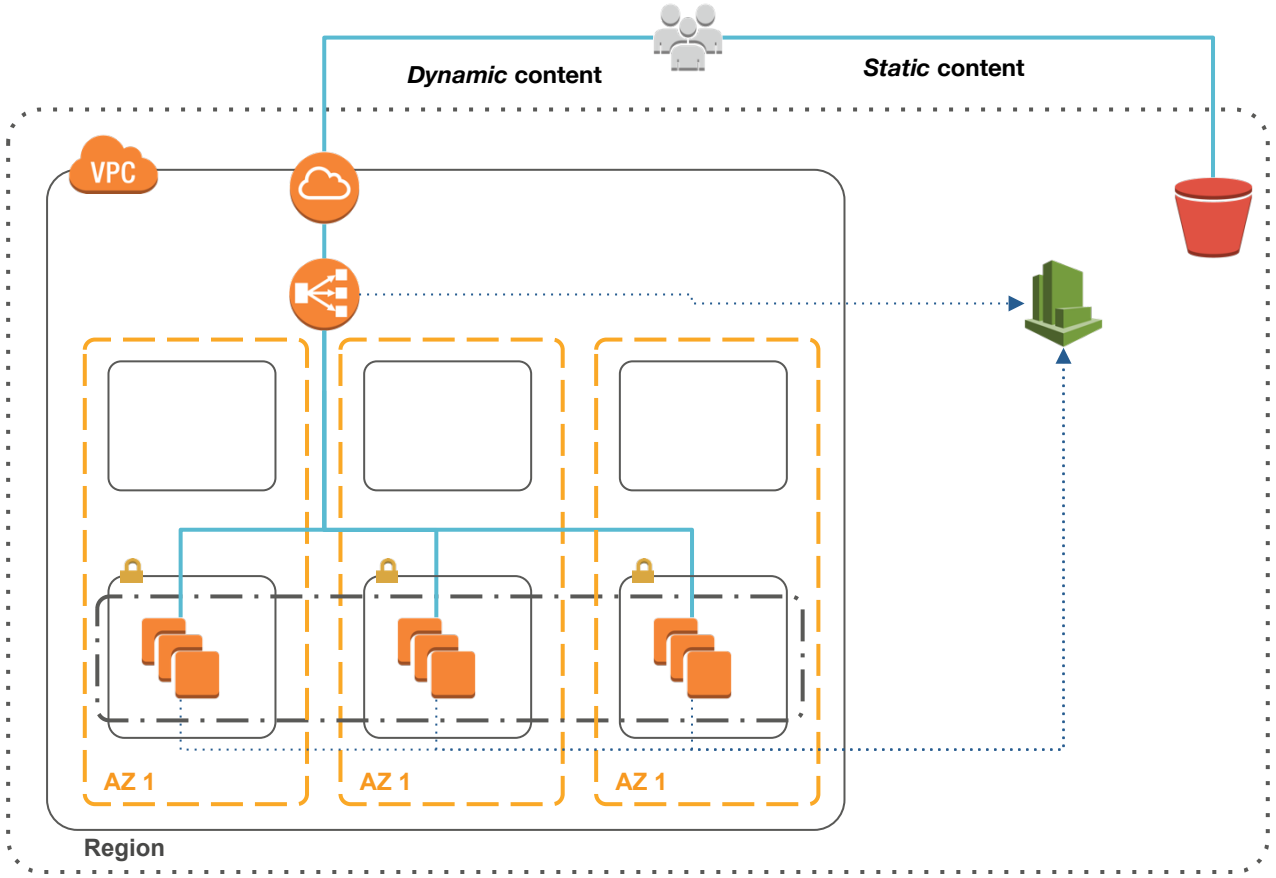
  33% loss in capacity

Can the remaining 84 or 66% fulfill the demand?

Monitoring, Alarms, and Alerts

> “The worst failure mode is the ‘silent’ failure, where the functionality is no longer working, but there is no way to detect it except indirectly. Your customer knows before you do.”

Monitoring



How do you know…

- latency?
- number healthy hosts?
- average CPU?
- when an issue arises?

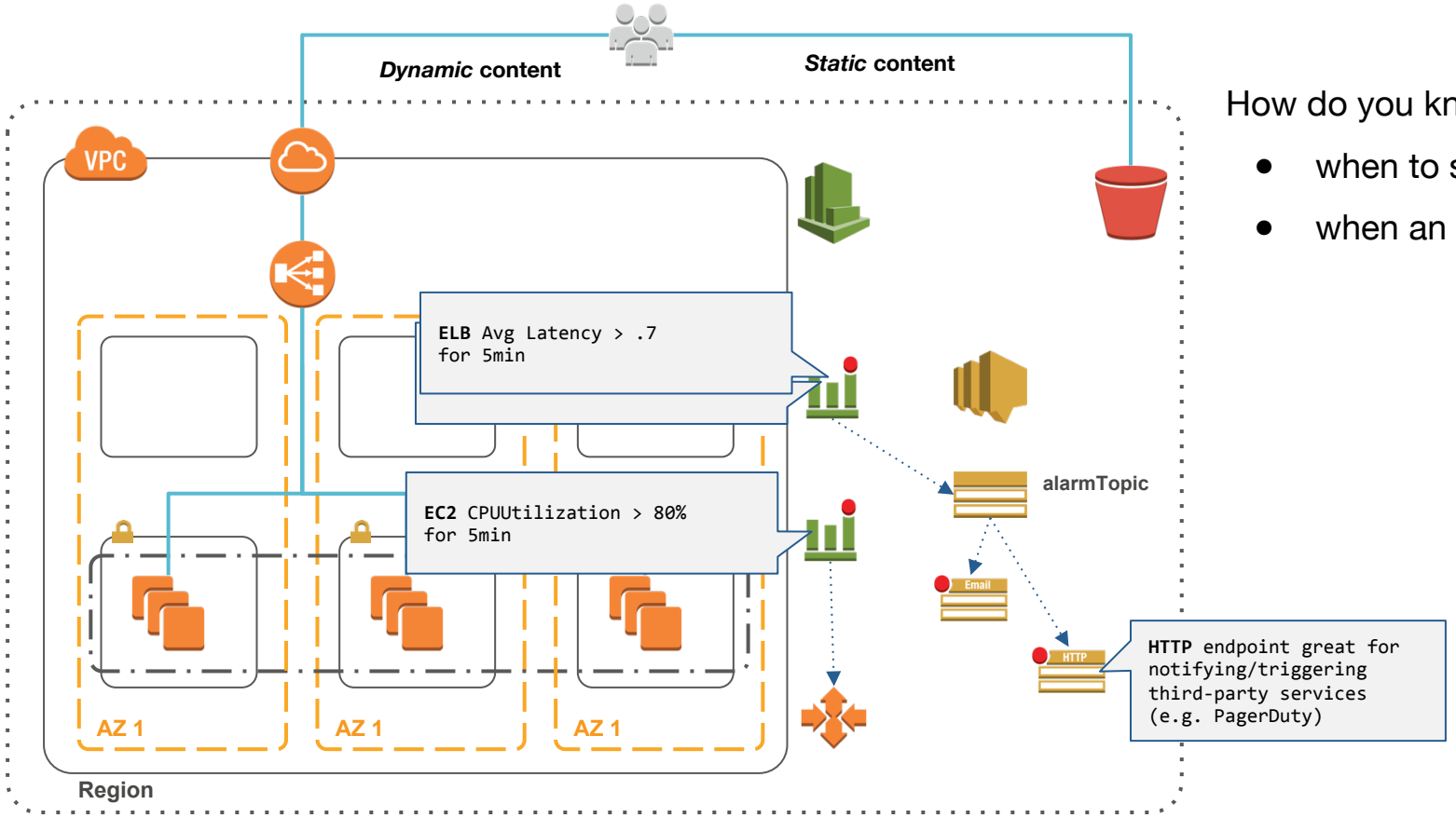
ELB Reports

- Healthy Host Count
- Status Codes
- Latency
- Request Count

EC2 Reports

- CPU Utilization
- Disk R/W Ops
- Network IO
- Status Check Failures

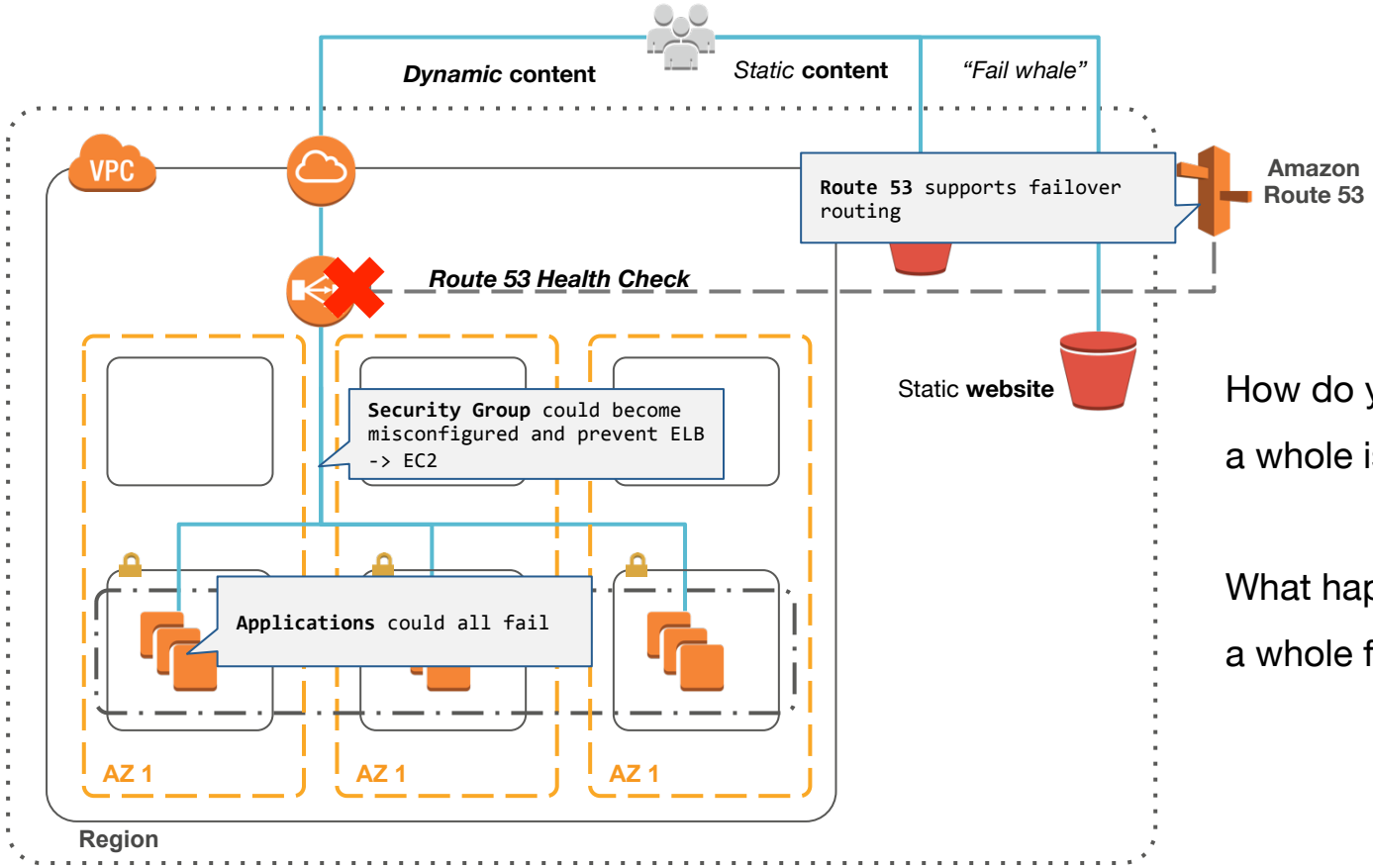
Alarms and Alerts



How do you know…

- when to scale?
- when an incident arises?

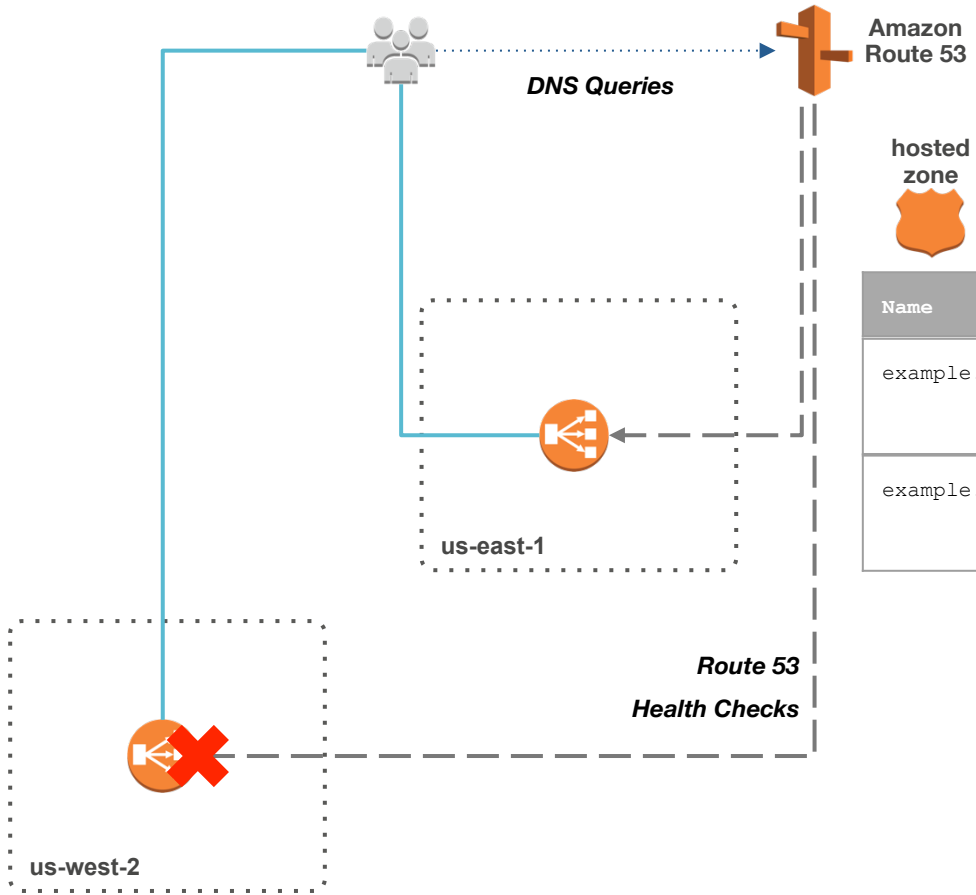
Monitoring External Endpoints



How do you know if the service as a whole is working?

What happens when the service as a whole fails?

Multi-Region Failover



Name	Target	Routing Policy	Record Type
example.com	ALIAS xxxx.us-west-2.elb.amazonaws.com	Failover	Primary
example.com	ALIAS xxxx.us-east-1.elb.amazonaws.com	Failover	Secondary

Performance

Network performance

Storage performance

Compute performance

Serverless architectures

Performance

> “...the efficient use of computing resources to meet requirements and how to maintain that efficiency as demand changes and technologies evolve.”

Performance Design Principles

- Democratize advanced technologies
  Leverage services so you can focus on product development

- Go global in minutes
  Provide lower latency, better customer experience at lower cost.

- Serverless architectures
  Remove the need to maintain servers and lower transactional costs.

Performance Design Principles

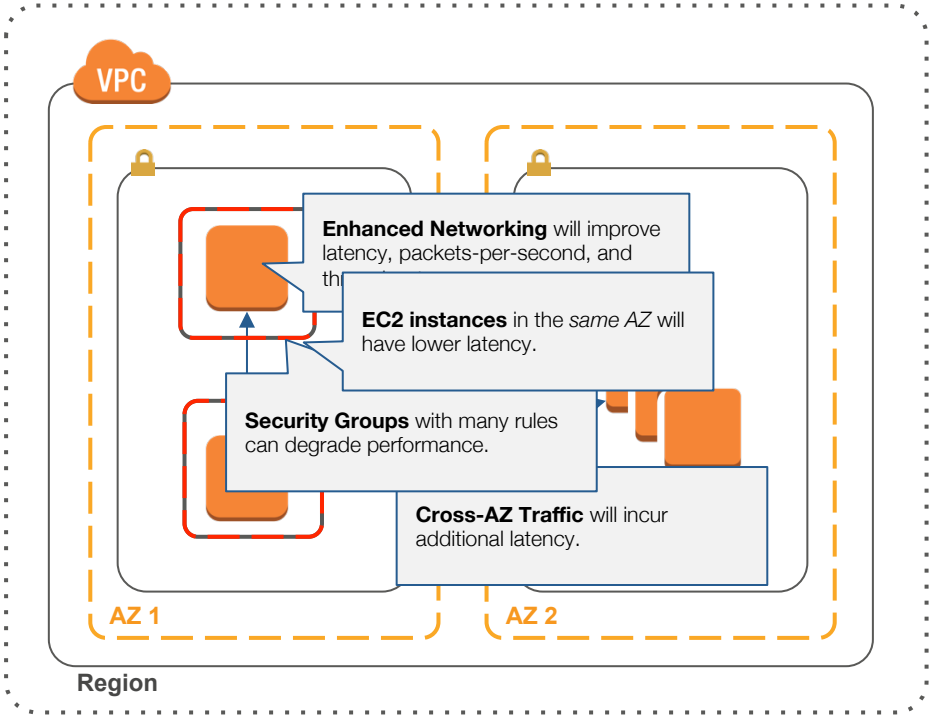
- Experiment more often
  With on-demand resources nothing is fixed, and we are not locked into decisions.

- Mechanical sympathy
  Use the right tool for the job.

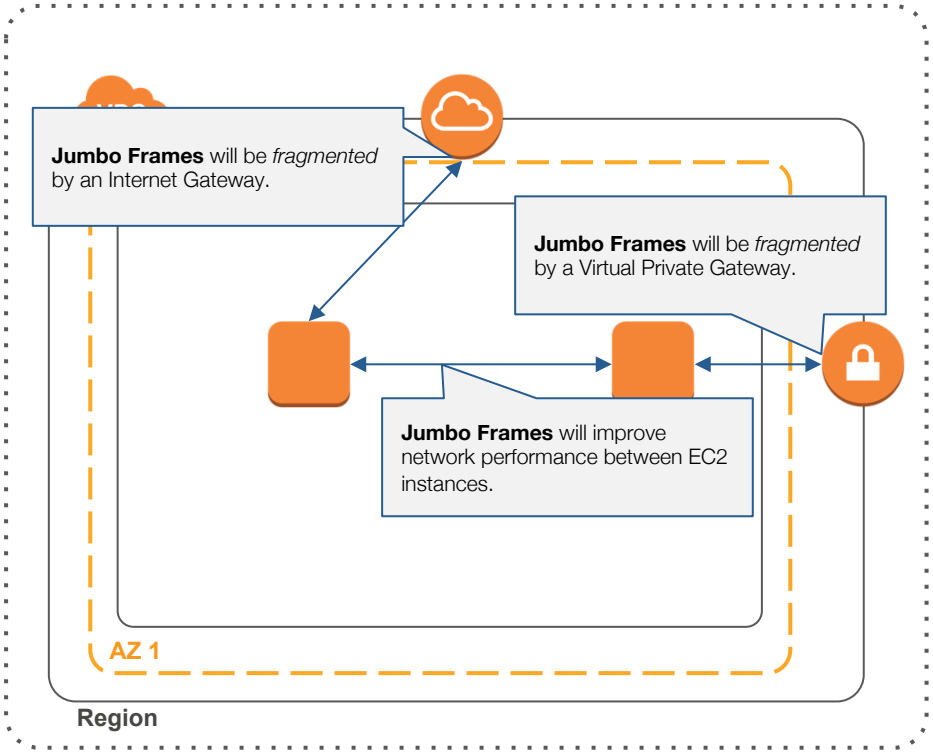
Performance Areas

- Selection
- Review
- Monitoring
- Trade-offs

Network: Inter-Instance Communication

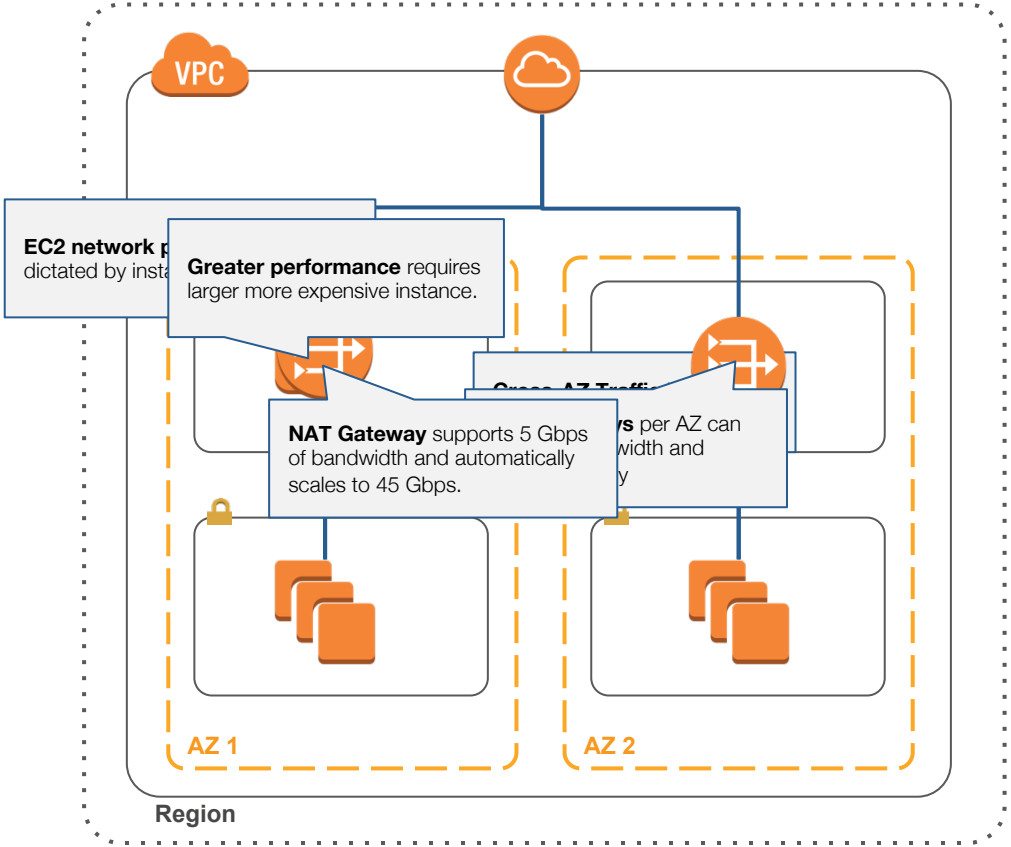


Network: Jumbo Frames



Do Not Fragment flag will cause packets to be dropped.

Network: NAT Instance vs Gateway



AWS Compute Options

- Instances
  General default

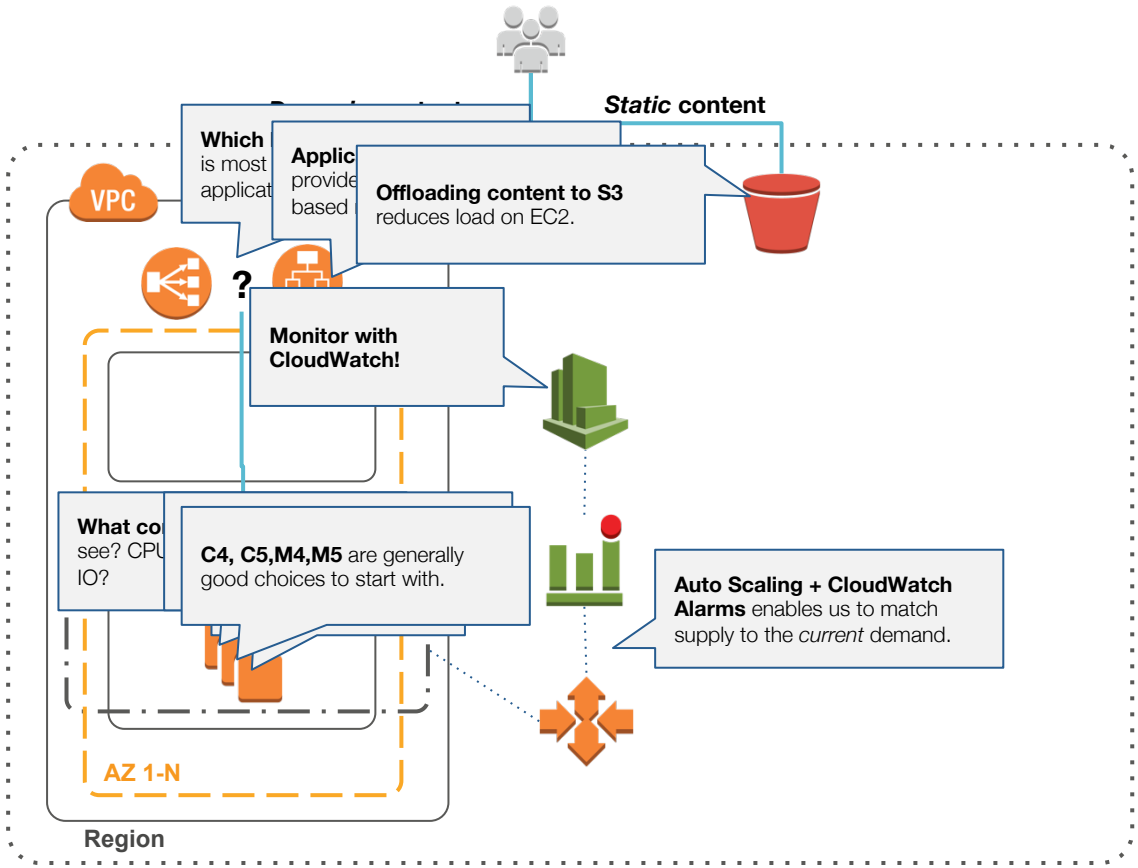
- Containers
  Improve instance utilization

- Functions
  Great for event-drive or parallelizable tasks

Amazon EC2 Instance Types

- General Purpose
  - T2
  - M3,M4,M5
- Compute Optimized
  - C4,C5
- Graphics/HPC
  - P2,P3
  - G3
- Memory Optimized
  - X1
  - R3,R4
- Storage
  - I3
  - H1
  - D2

Instance-Based Performance



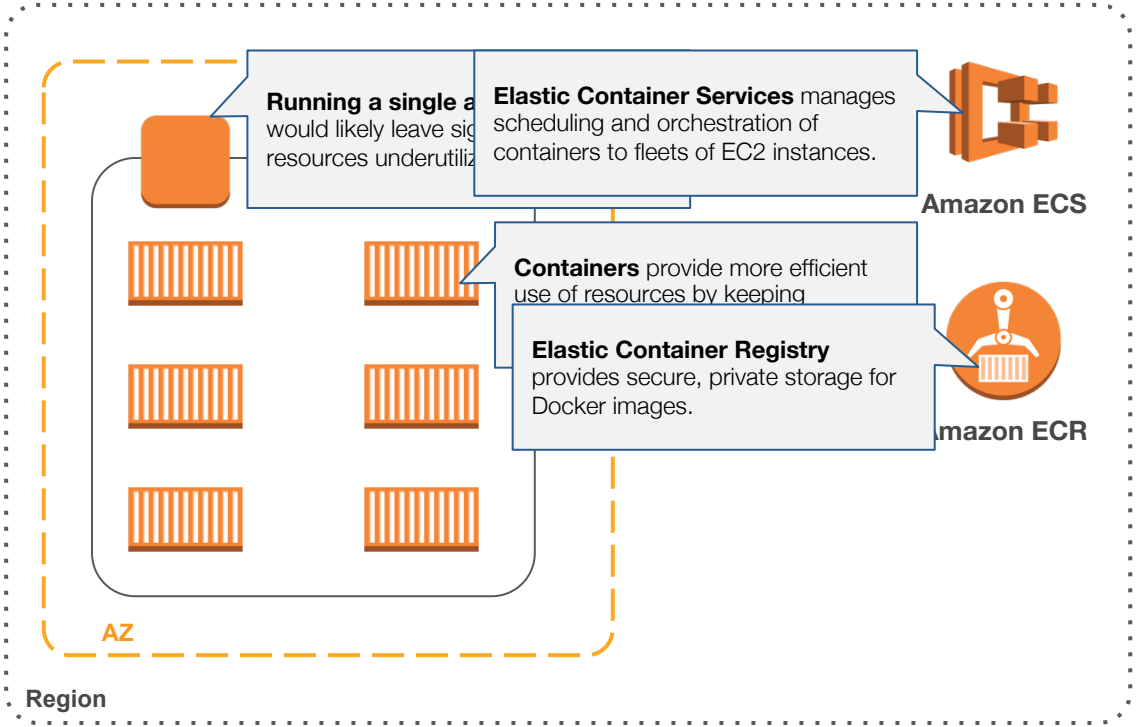
Capture key performance indicators (KPIs).

Technical *and* business metrics.

Custom metrics:

- Thread/process count
- JVM Heap size
- Garbage collection
- Disk usage

Improving Performance with Containers

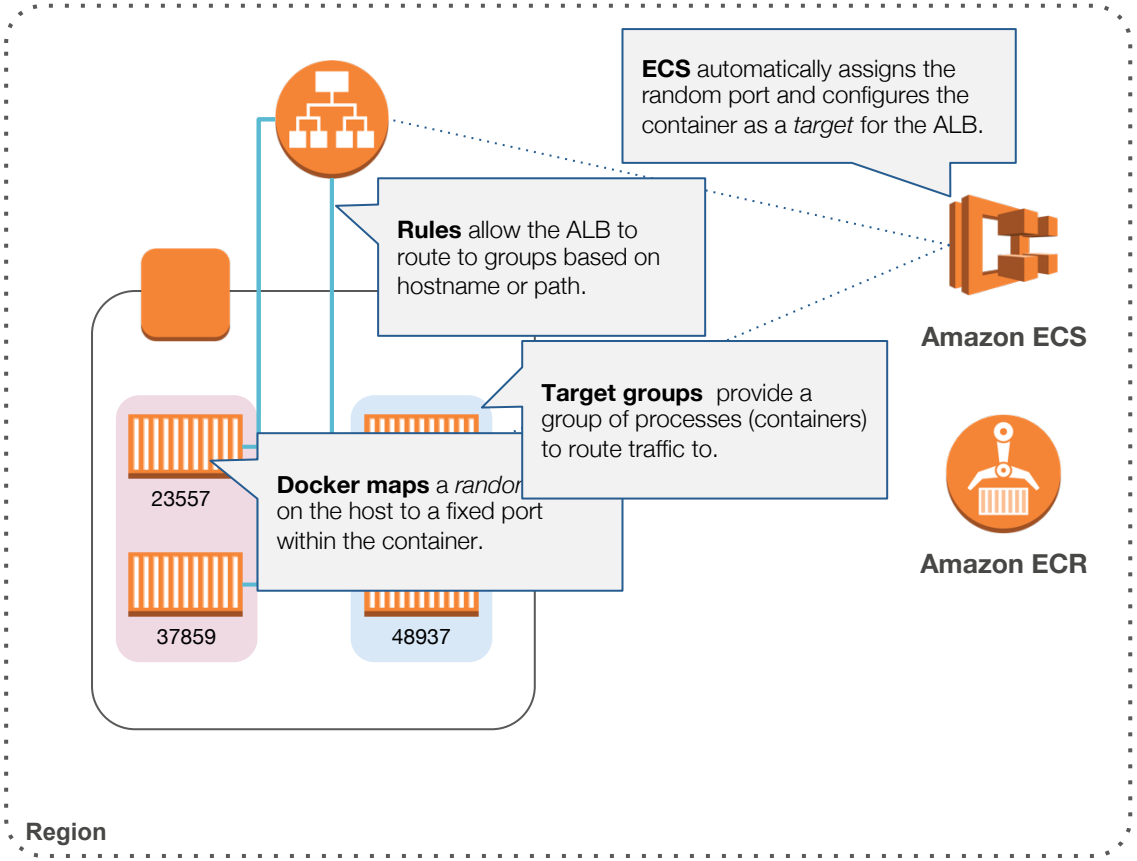


How can we make the most of the CPU, memory, and network IO available to our instances?

Leverage containers!

- Amazon ECS
- Amazon EKS

Dynamic Port Mapping



How can we place more than one container for the same process on a single EC2 instance?

Dynamic port mapping!

- Amazon ECS
- Amazon EKS
- Application Load Balancer

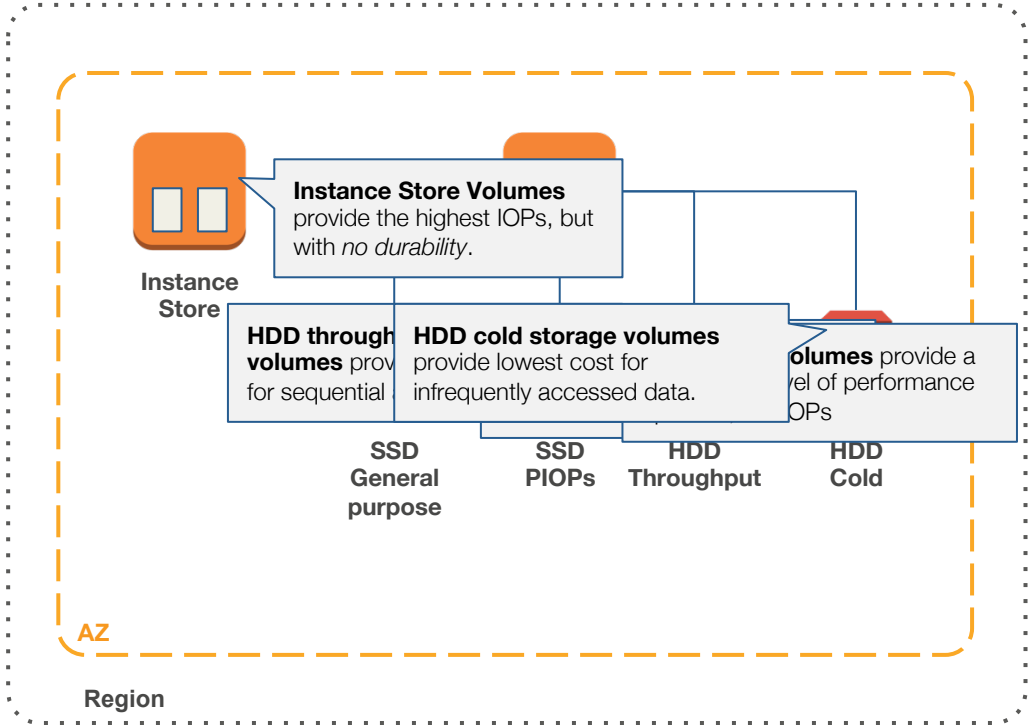
Storage Performance

> “Performance can be measured by looking at throughput, input/output operations per second (IOPS), and latency”

Storage Options

- **Block Storage**
  - Elastic Block Store (EBS)
    - SSD general purpose (gp2)
    - SSD provisioned IOPs (io1)
    - HDD throughput-optimized (st1)
    - HDD cold storage (sc1)
  - EC2 instance store (ephemeral)
- **File System**
  - Elastic File Systems (EFS)
- **Object Storage**
  - Amazon Simple Storage Service (S3)
- **Archival**
  - Amazon Glacier

Block Store Volume Performance



What is the best access method for our application? Block, File, or Object?

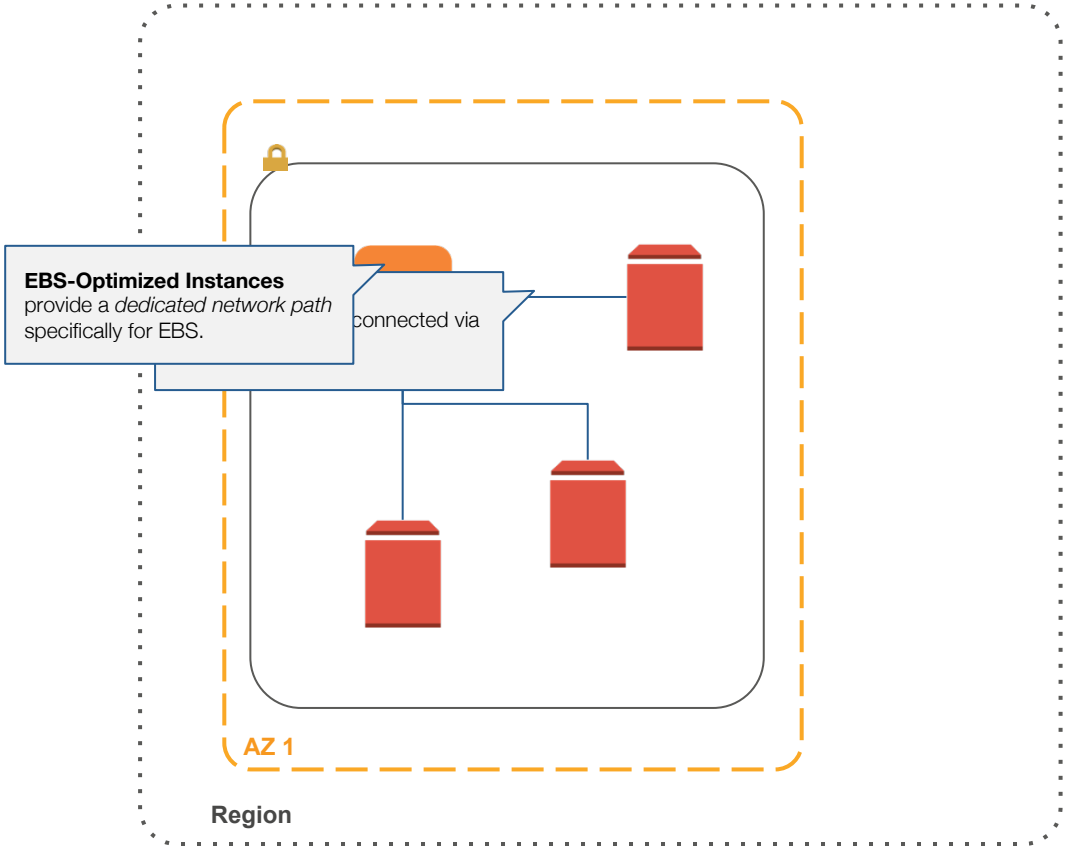
Will our app perform random or sequential IO?

What throughput rates might be required?

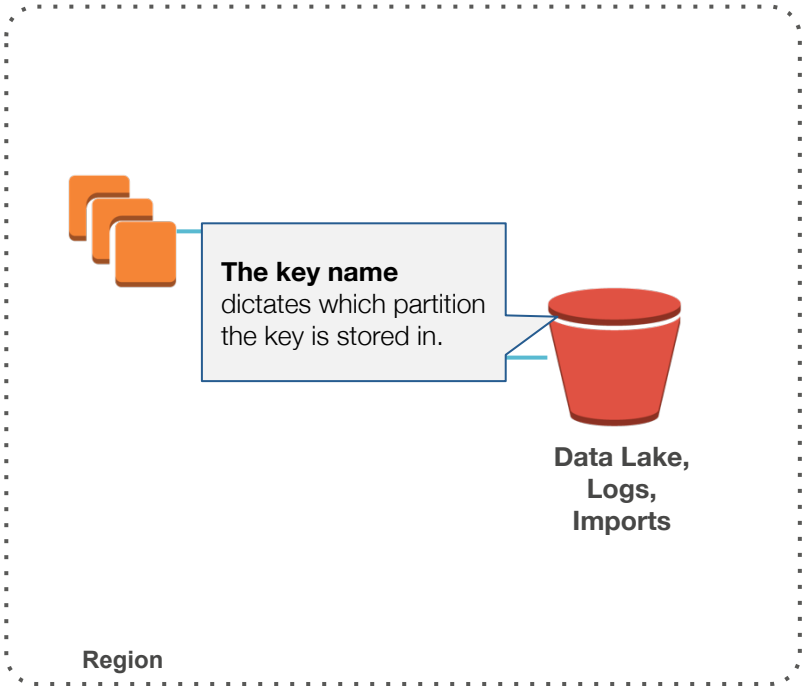
How frequently will the data be read?

How frequently will the data be updated? Write once read many? Constantly? Never?

EBS-Optimized Instances



S3 Performance with a Mix of Request Types



If our requests are typically a mix of GET, PUT, DELETE, or GET Bucket (list objects), how can we ensure consistent performance for a bucket?

How do we avoid SlowDown exceptions?

Entropy!

Add random hash prefixes.

S3 Performance with a Mix of Request Types

```
mybucket/2018-03-01/vendor-1234/imports.json
mybucket/2018-03-01/vendor-2346/imports.json
mybucket/2018-03-01/ve
mybucket/2018-03-01/ve
mybucket/2018-03-01/
...
...
mybucket/2018-03-02/vendor-2832/imports.json
```

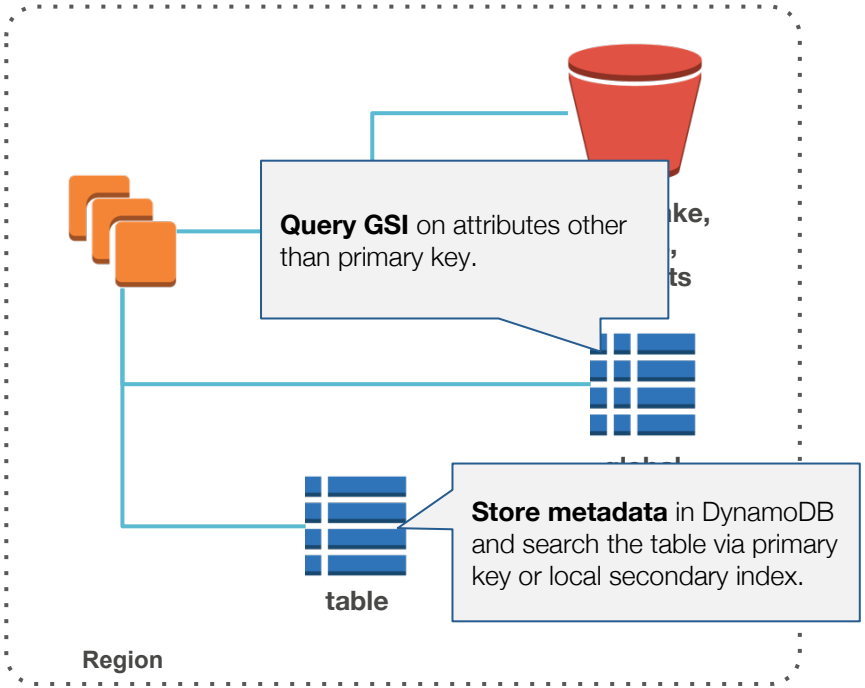
Lexicographical sequences introduces a performance problem by forcing too much data into too few partitions.

```
mybucket/fe98-20                                    json
mybucket/7ag6-2                                     json
mybucket/f93b-2018-03-01/vendor-8267/imports.json
mybucket/d8cc-2018-03-01/vendor-7892/imports.json
mybucket/302f-20                                    json
...
...
mybucket/9ffa-01                                    son
```

Random hashes spread load evenly across partitions for better performance.

Random hashes improve performance, but remove predictable prefixes, making search more difficult.

Searching for S3 Objects



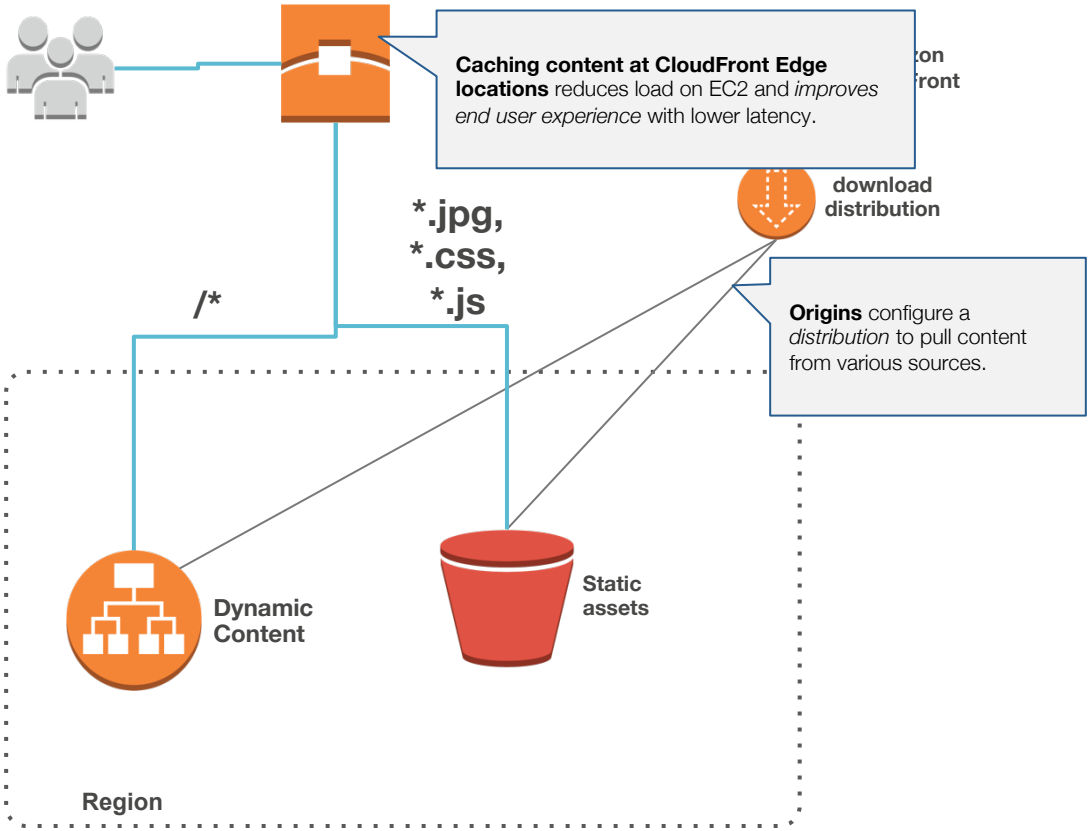
S3 list objects is not an efficient form of search.

If our S3 key prefixes are randomized how do we find objects?

Store searchable metadata in something indexed:

- DynamoDB
- ElasticSearch

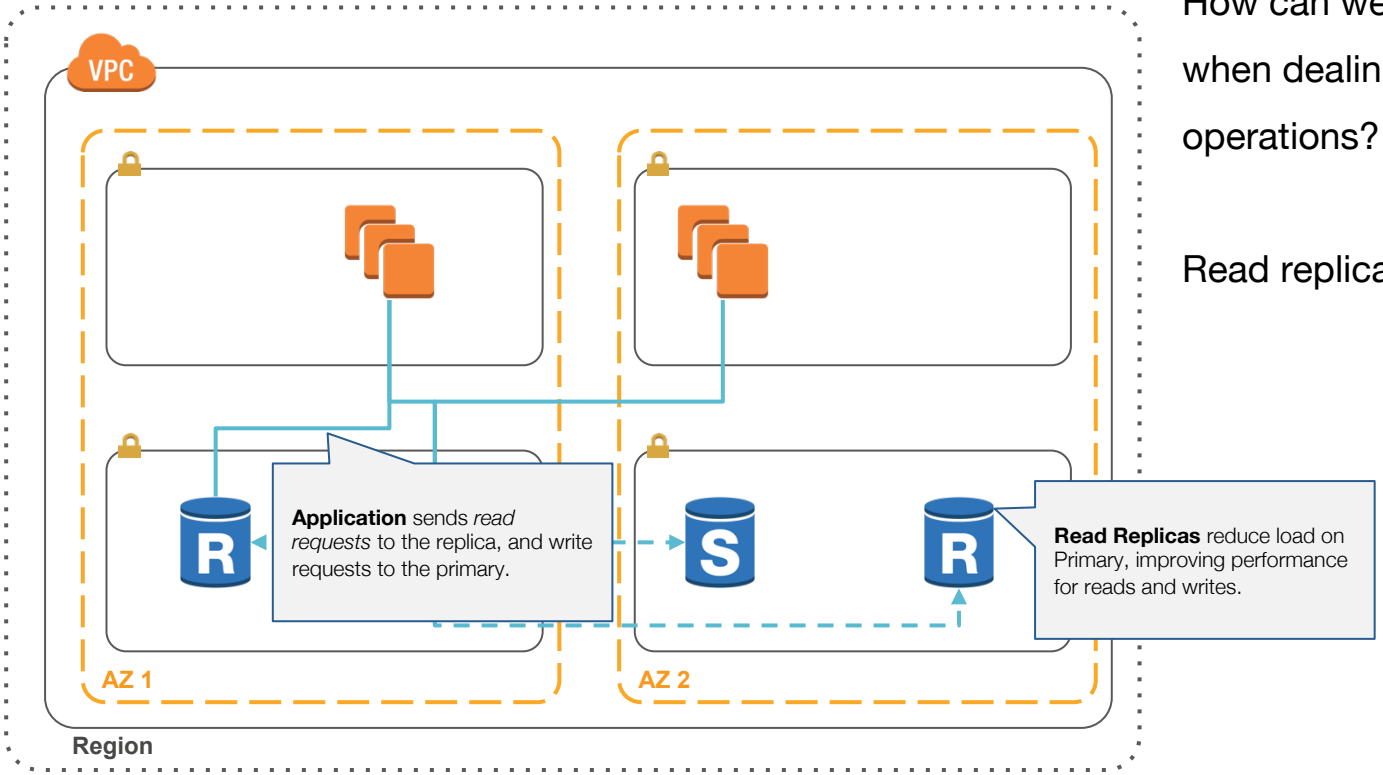
Improving End User Experience



How can we improve the experience of the end user?

Lower their latency through caching!

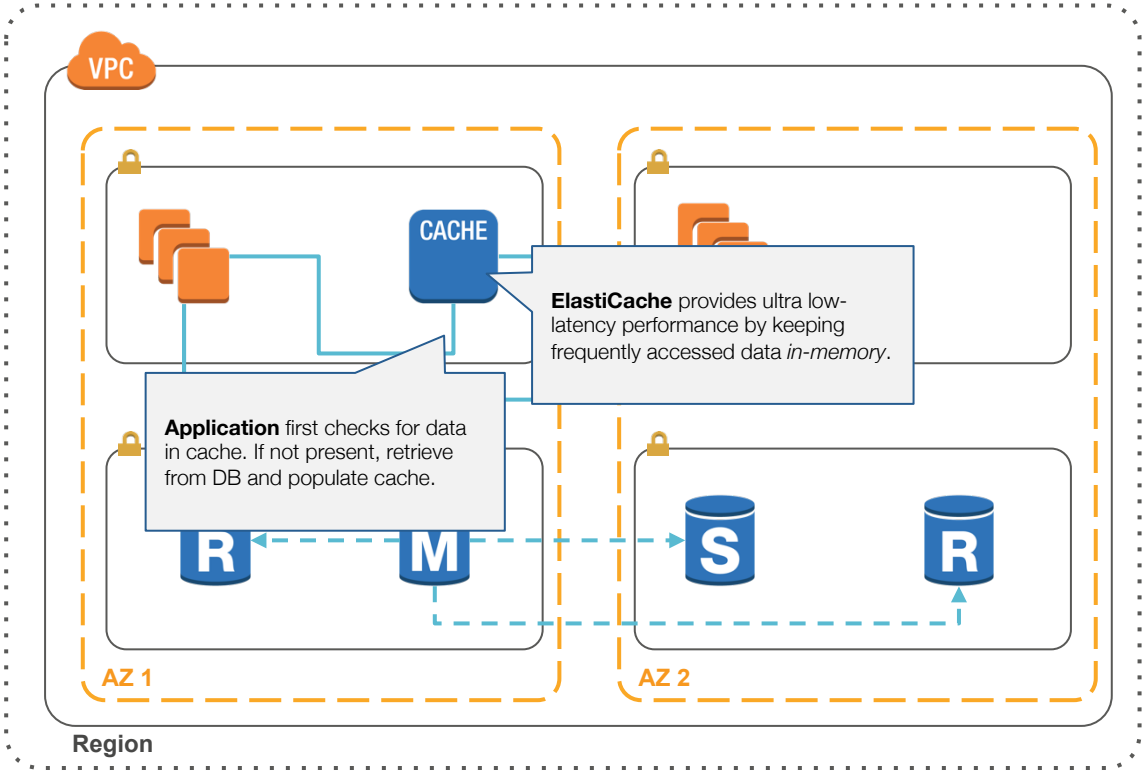
Improving Read Heavy Applications



How can we improve the performance when dealing with many read operations?

Read replicas!

Improving Read-Heavy Applications



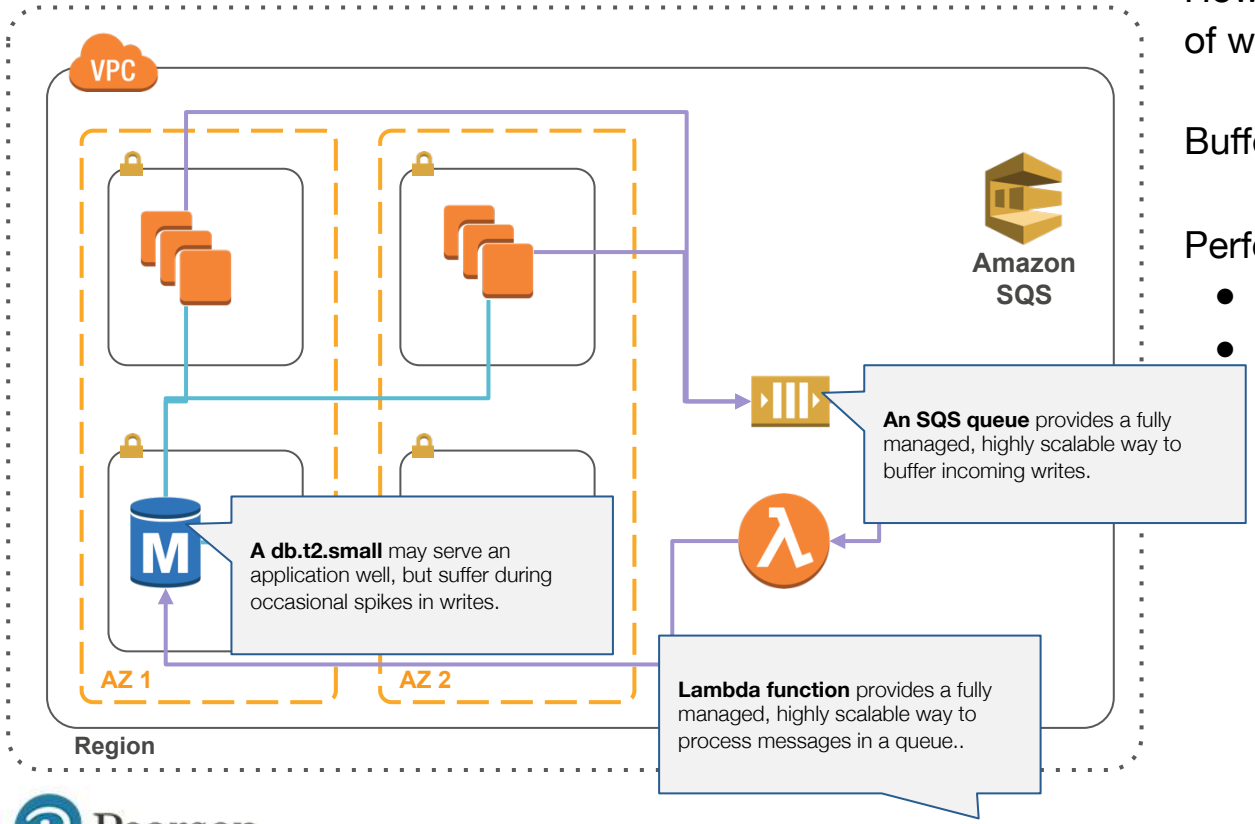
How can we improve the performance when dealing with slow queries?

In-memory cache!

Amazon ElastiCache offers fully managed

- Redis
- Memcached

Improving Write-Heavy Applications



How can we improve the performance of writes to a small RDS instance?

Buffer them.

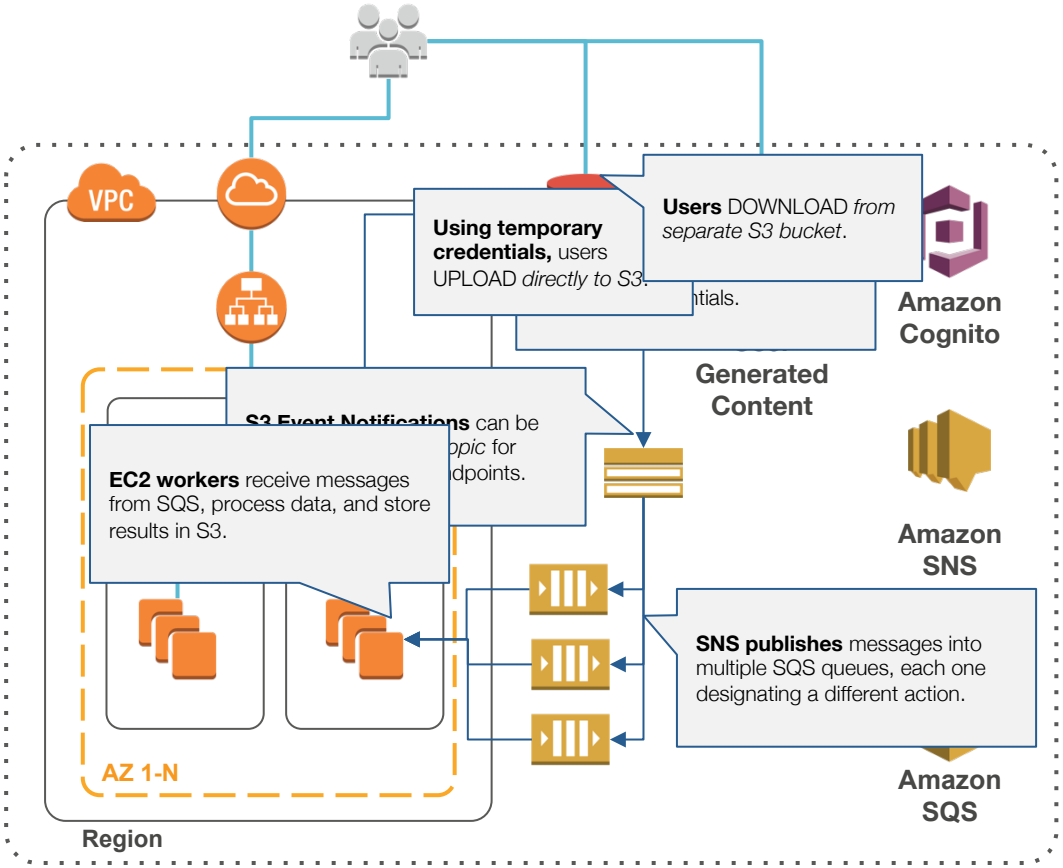
Performance improved with

- Amazon SQS
- AWS Lambda

or

Amazon EC2

Event-Driven Architectures



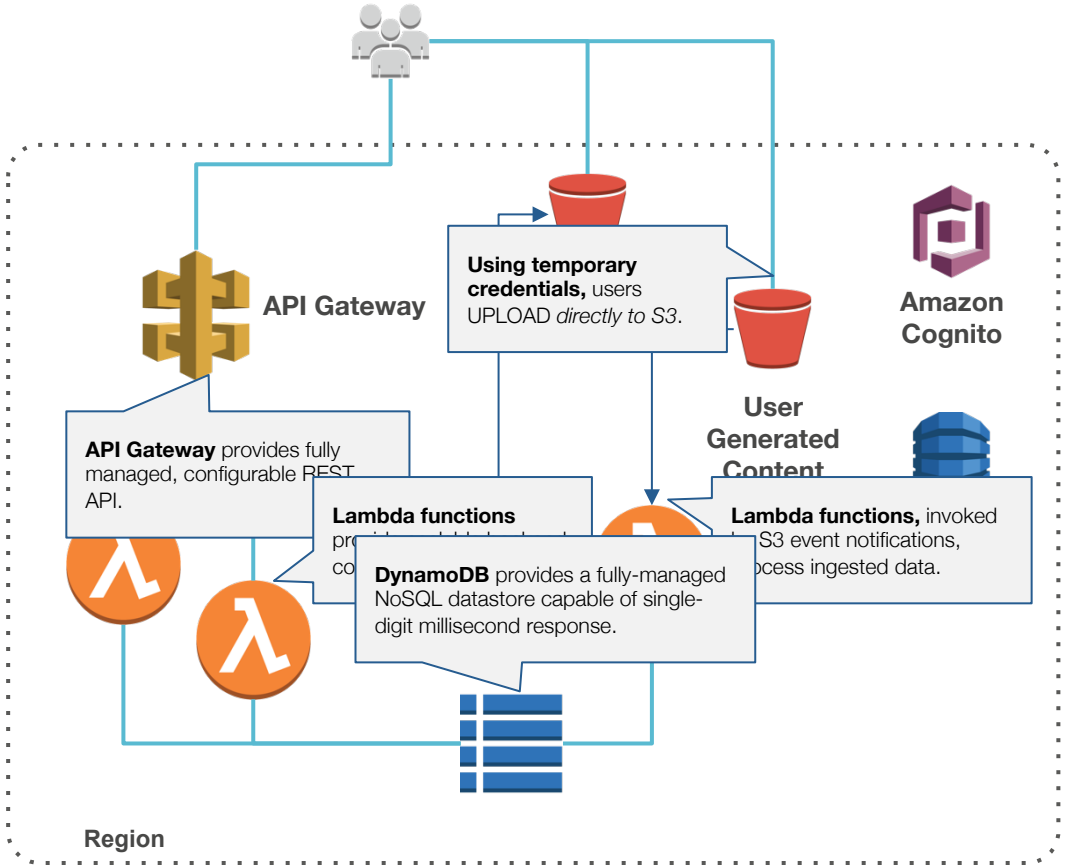
How can we improve the performance of ingestion and processing uploads?

Asynchronous, event-driven architectures.

Performance improved with

- Amazon Cognito
- Amazon S3
- Amazon SQS
- Amazon SNS

Serverless Architectures



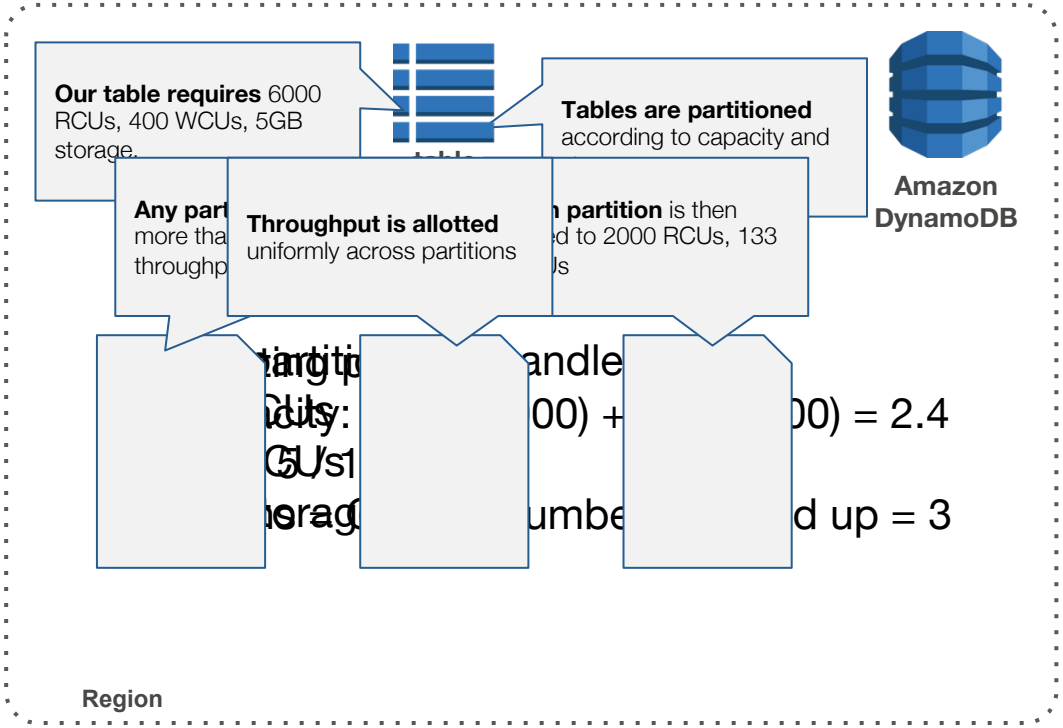
How can we improve the performance of our infrastructure while focusing on application development?

Serverless architectures.

Performance improved, operational burdens handled with

- Amazon Cognito
- Amazon S3
- Amazon API Gateway
- AWS Lambda

Improving DynamoDB Performance



How can we improve the performance of DynamoDB queries?

Understand the nature of DynamoDB partitioning and throughput allocation.

Primary keys that provide an even distribution of storage and request load.

Local secondary indexes

Global secondary indexes

Using query vs scan operation.

DynamoDB Secondary Indexes

Amazon DynamoDB

films

Table

Writes will updates rare at relatively

Changes to the table will be reflected in the global secondary index (eventual consistency).

title	year	rating	runtime
The Matrix	1999		
Eyes Wide Shut	1999		
The Sixth Sense	1999	pg-13	107
Saving Private Ryan			
The Big Lebowski			

How do we query against year across ALL partitions?

Certain films will be read more often than others. Caches will help mitigate hot reads.

Partition Key

Global Secondary Index

year	title
1999	The Matrix
1999	Eyes Wide Shut
1999	The Sixth Sense
1998	Saving Private Ryan
1998	The Big Lebowski

Partition Key

Sort Key

DynamoDB Secondary Indexes

Amazon DynamoDB

film-cast

Table

film	member	role
The Matrix	Keanu Reeves	Neo
The Matrix	Laurence Fishburne	Morpheus
The M		Director
The M		Director

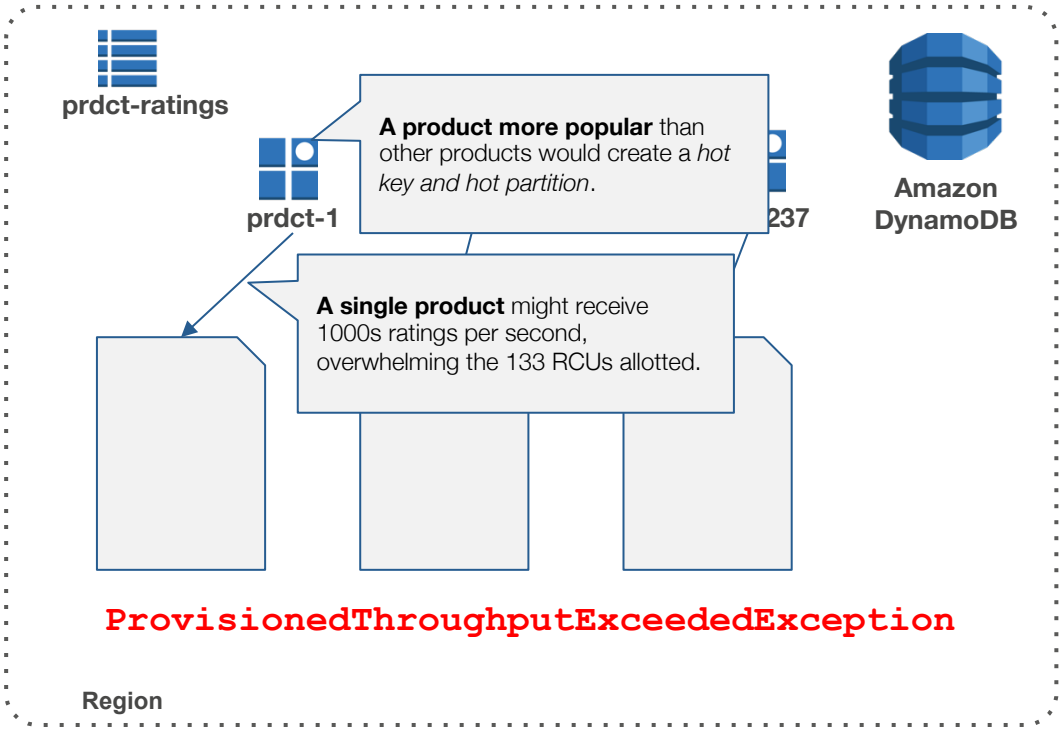
Partition + Sort Key makes a *composite* primary key.

Partition Key

Sort Key

Sort Key

Improving DynamoDB Performance

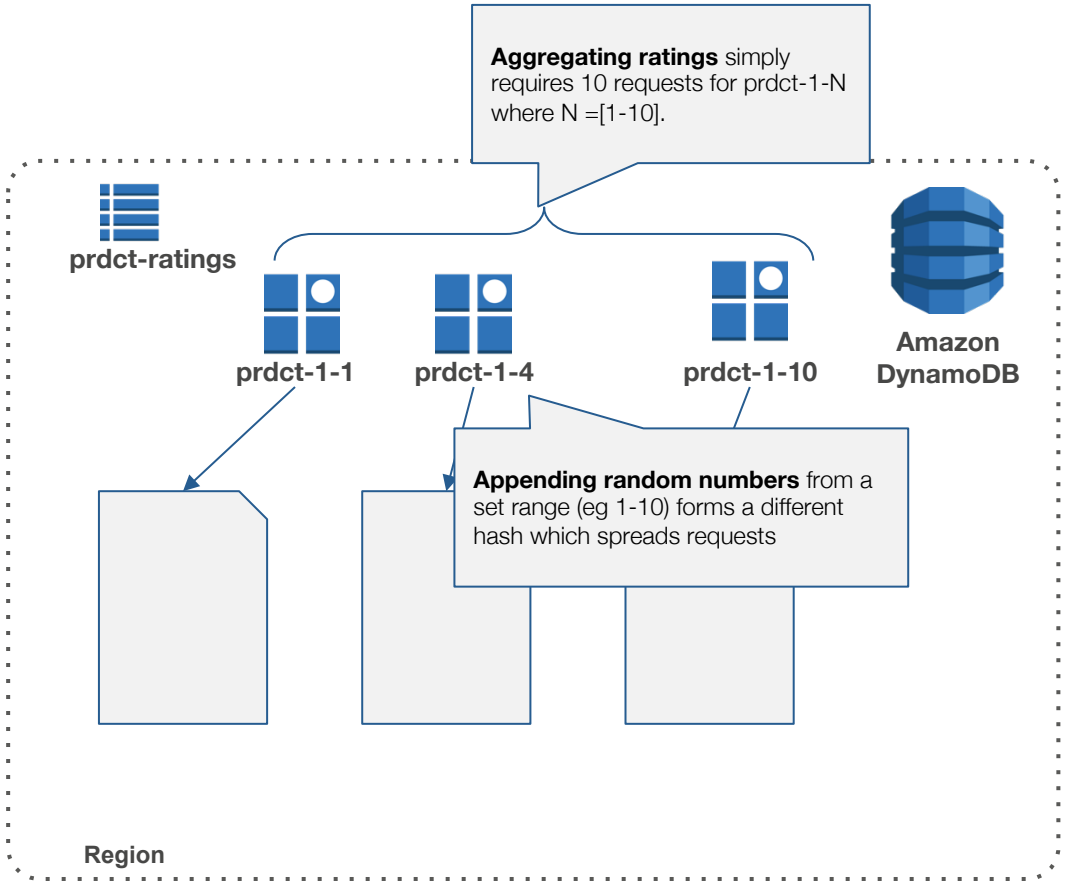


How does the primary key affect performance?

The partition serving the data is determined by the *partition key*.

Scenario: e-commerce site displays product ratings, members rate products on a scale of 1-5.

DynamoDB Write Performance with Random Suffixes



How can we mitigate hot keys or hot partitions?

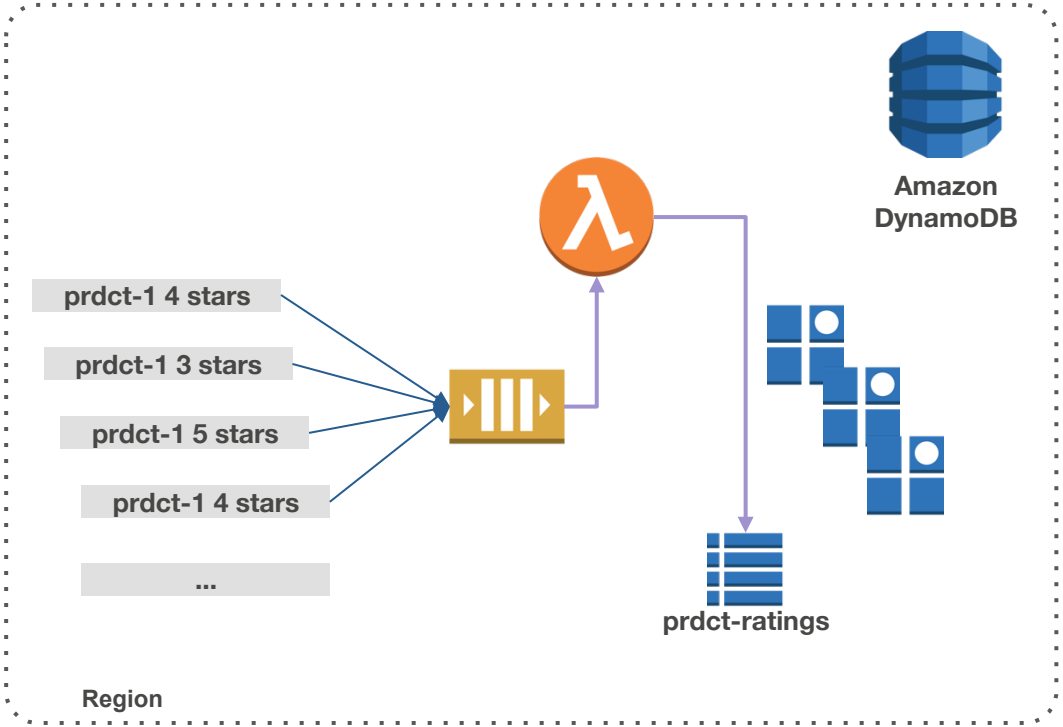
Try a strategy that provides a lot of unique values evenly requested.

For reads: caching in ElastiCache or DAX.

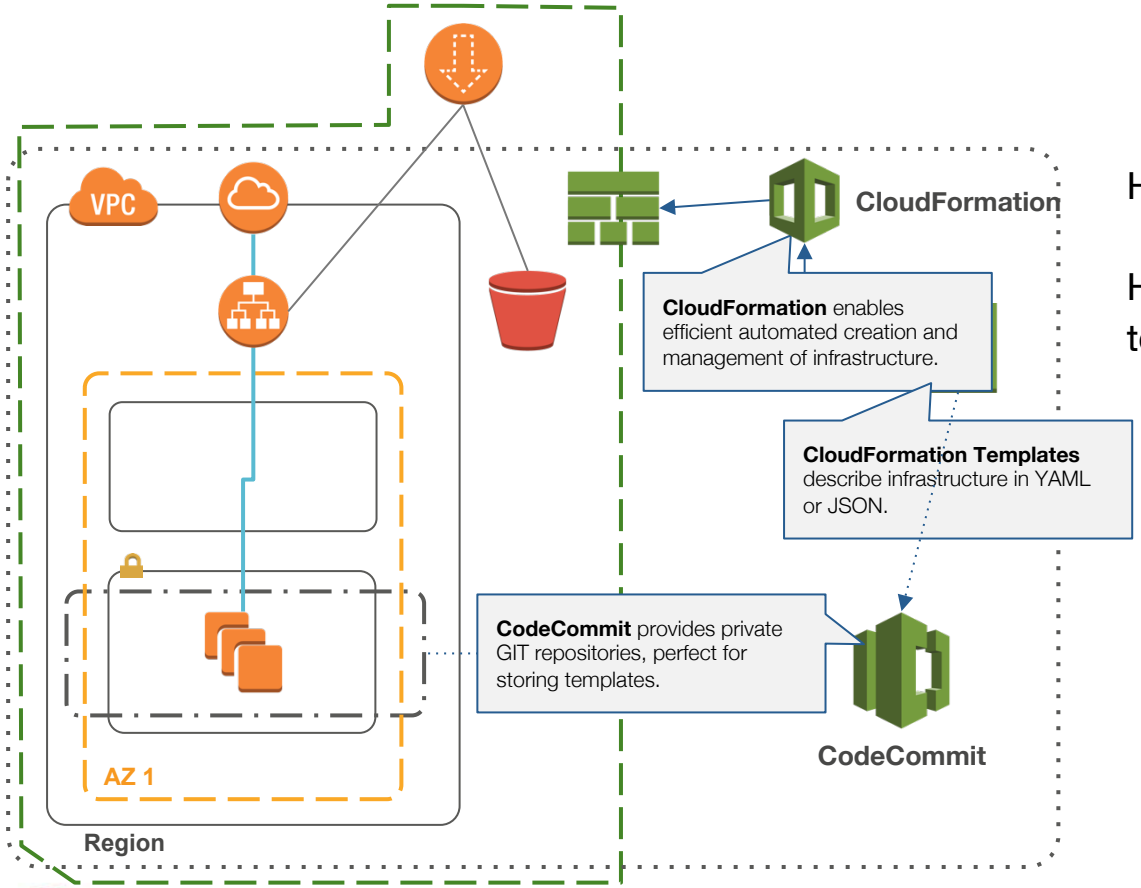
For writes:

- buffering with SQS or Kinesis
- appending random values to the key

DynamoDB Write Performance with a Buffer



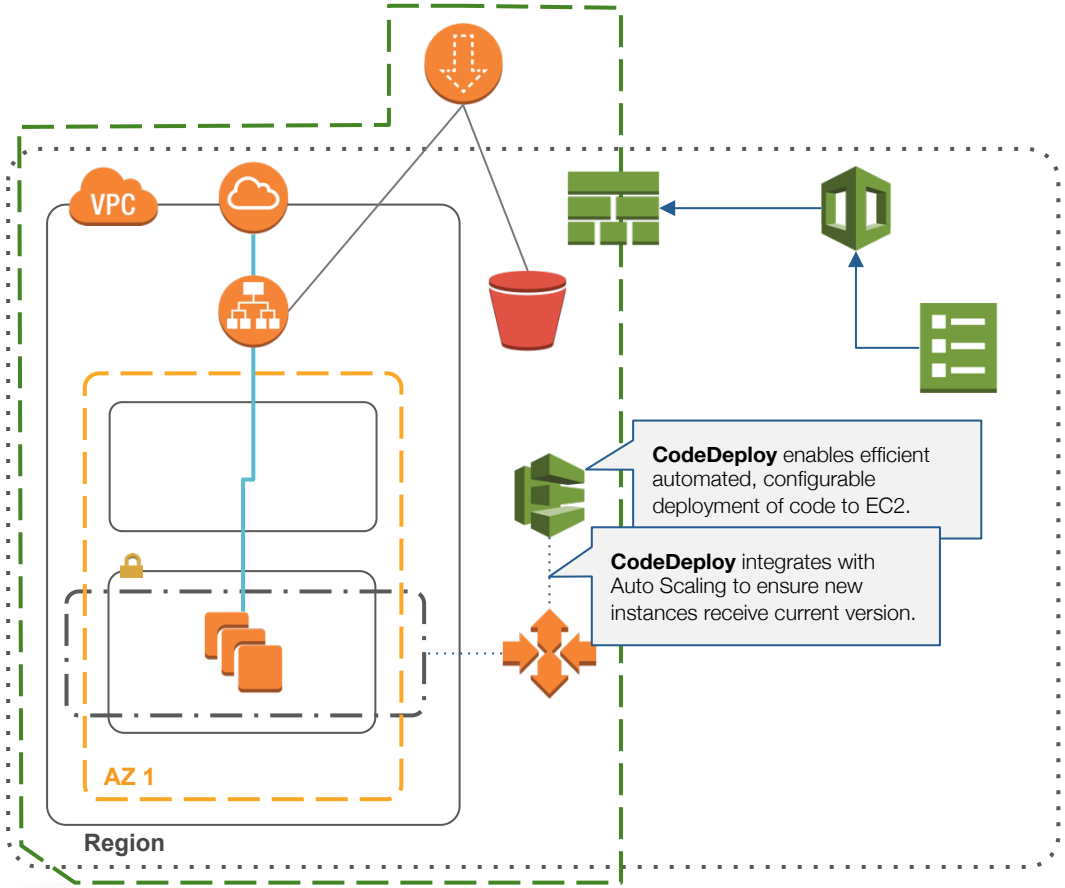
Automate Infrastructure Deployments



How is the infrastructure created?

How do we safely manage changes to existing infrastructure?

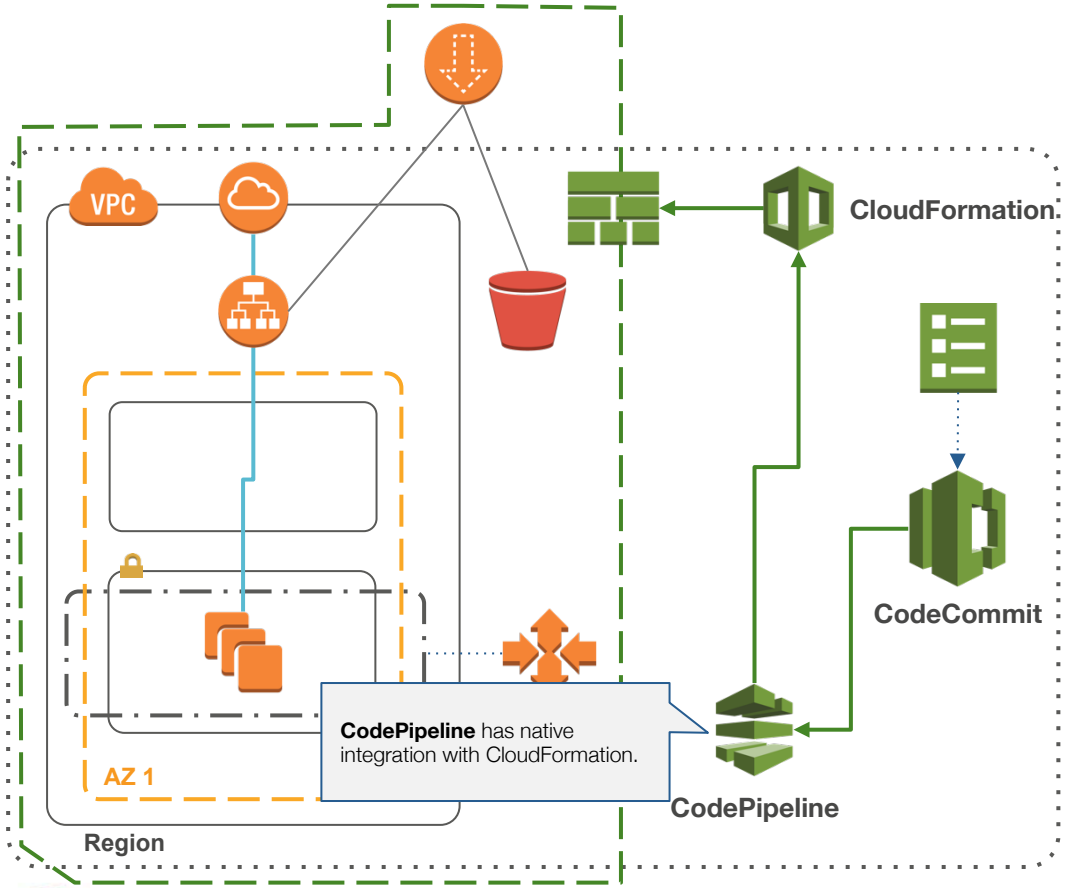
Automate Code Deployments



How is the infrastructure created?

What about code updates?

Automate Testing



How do we verify the infrastructure performs as expected?

TEST IT!

Apply continuous deployment practices to infrastructure.

Security

Identity and Access

Detection

Securing Infrastructure

Securing Data

Responding to incidents

Security Design Principles

- Principle of least privilege, separation of duties
- Reduce or eliminate reliance on long-term credentials
- Log and monitor everything, monitor and alert in real time.
- Apply security in layers with VPC, NACLs, Security groups, OS, and application

Security Design Principles

- Automate security
- Leverage encryption in transit and at rest
- Have a response plan!

Identity and Access Management

Users

Groups

Roles

Policies

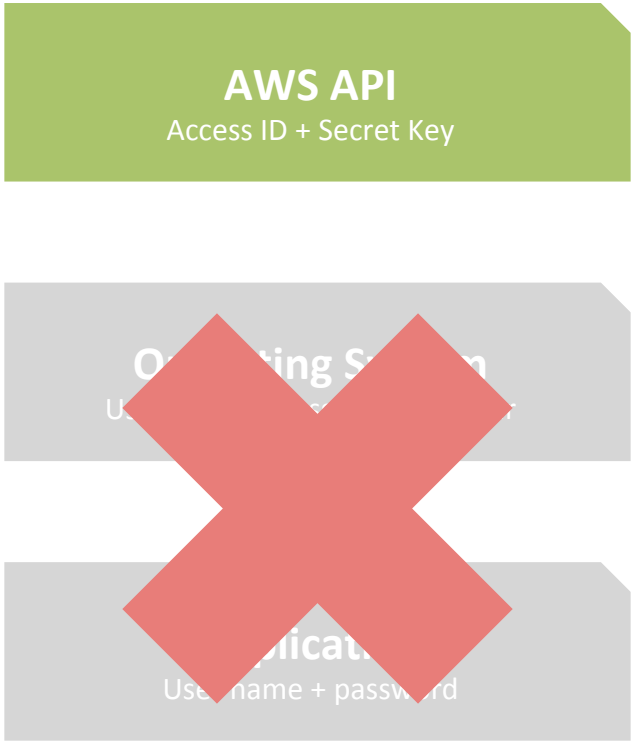
Cross-Account Access

Federation

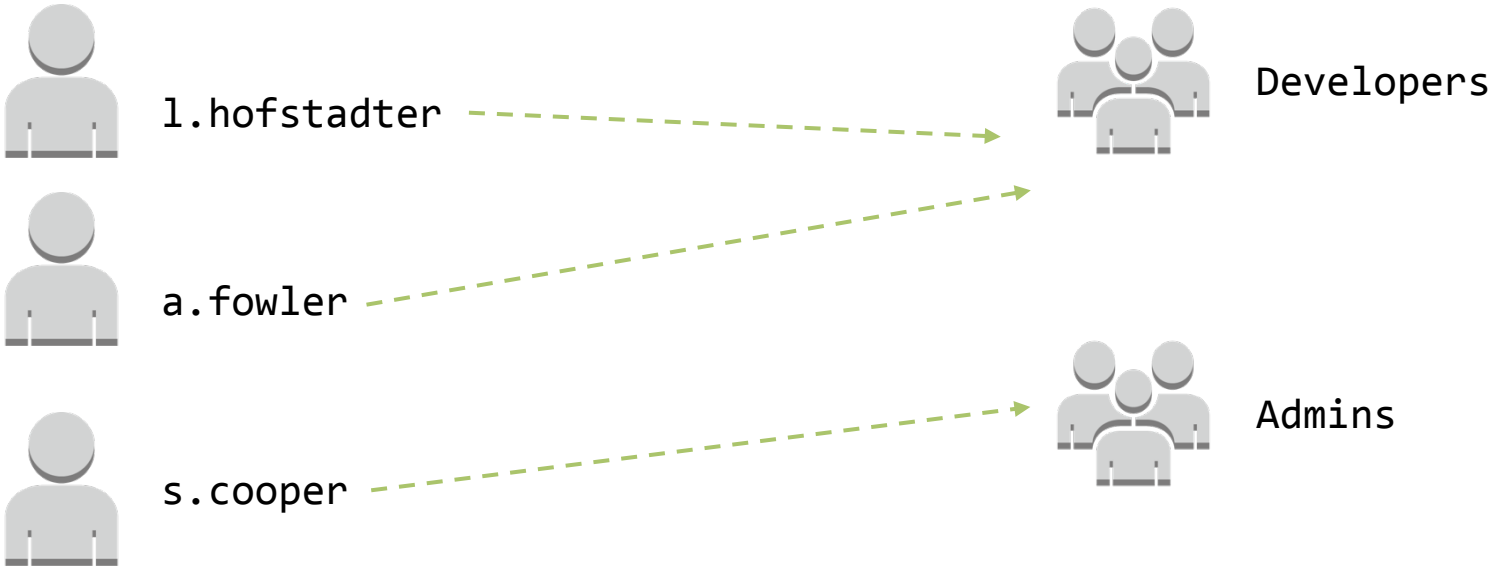
Best Practices

IAM Overview

- Authentication
- Authorization via Policies
- Users
- Groups
- Password Policy
- Multi-Factor Authentication



Users and Groups



- Created and exist with IAM service
- Login to Management Console
- Create long-term *access keys*
- Can enable per-user MFA device

- Cannot be nested

Follow principle of least privilege!

Policies

- Authorization (permissions) granted via *policies*
- Policies written in JSON
- Policy Types
  - Managed Policy
    - AWS managed
    - Customer managed
  - Inline Policy
- Create policies via
  - Generator
  - Hand written policies
- Evaluation logic:
  - Defaults to implicit deny
  - Explicit deny
  - Explicit allow

Example Policy

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

Array (List)

Allow

Upload or Download

To anywhere in this bucket

Roles

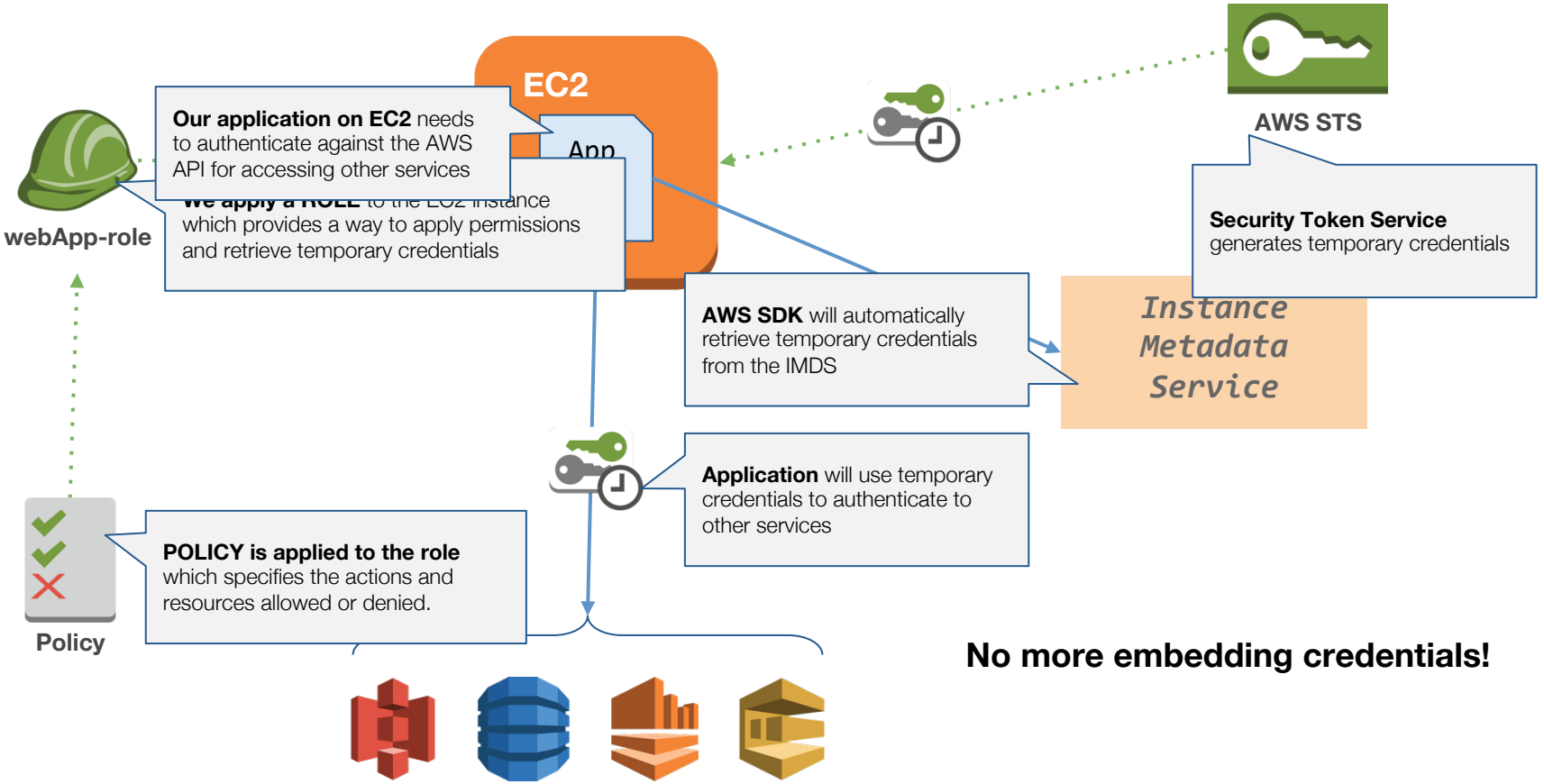
What *not* to do

- Embed access keys in code
- Embed in environment variables
- Share with
  - Third parties
  - Hundreds of enterprise users
  - Thousands (millions?) of web users

Better Practice

- Use *temporary credentials*
- *Delegate* permissions to:
  - EC2 instance
  - AWS service
  - A user (elevate privileges)
  - Separate account
    - One you own
    - Third party

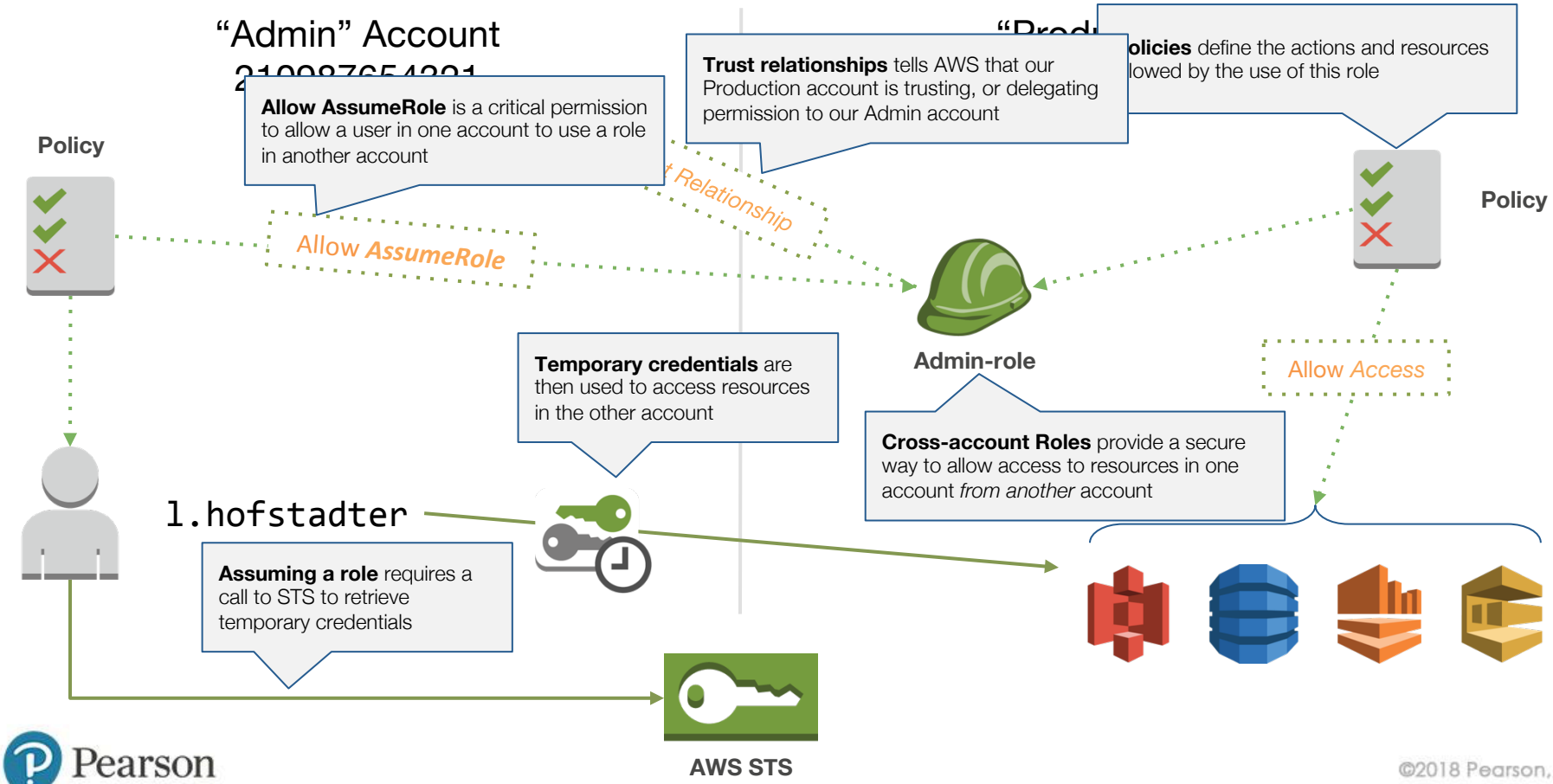
Roles for EC2



No more embedding credentials!

Pearson

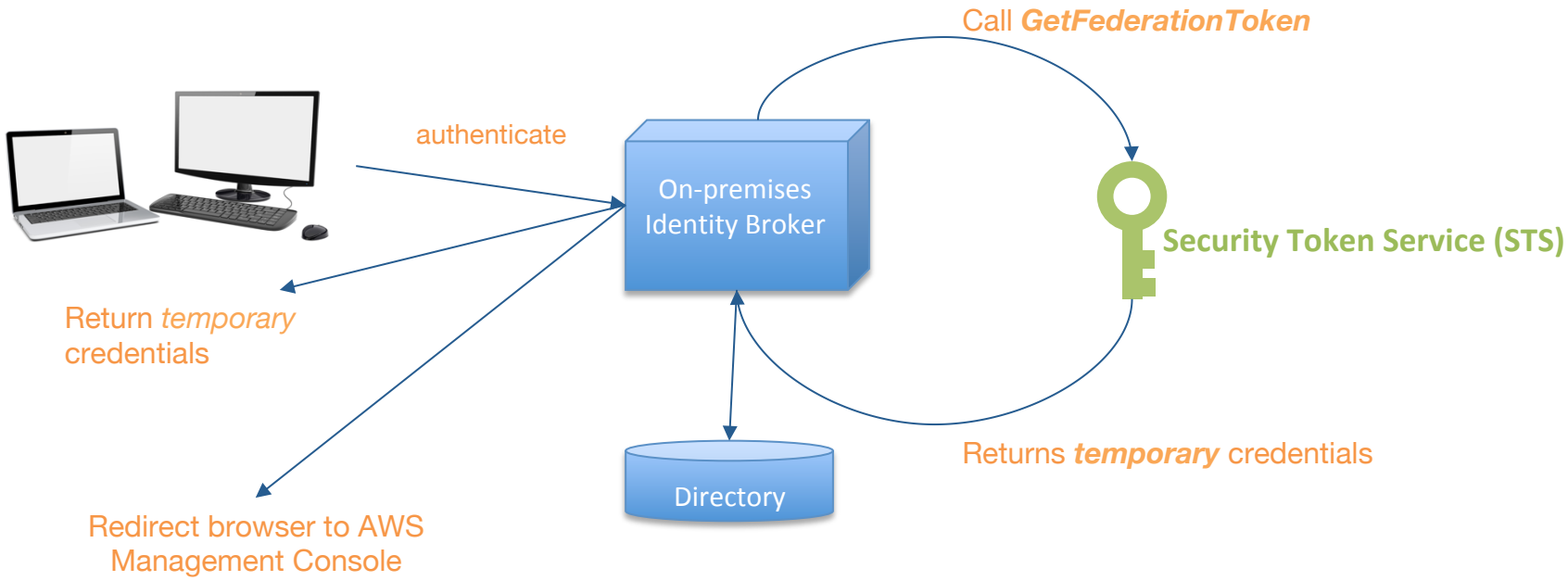
Roles for Cross Account Access



Federated Users

- Organizational users:
  - Leverage existing user directories:
    - LDAP
    - Active Directory
  - Temporary credentials
  - Single sign on
- Web/mobile application users:
  - Apps bypass backend APIs/proxies

Federated Users Example



AWS New Account First Steps

1. Create an admin user in IAM
2. Enable Multi-Factor Authentication on root account
3. Enable Cost and Usage Report
4. Log out of root account
5. Log in with admin user
6. Create additional users, groups, etc

IAM Best Practices

- Root credentials
  - Email address + password
  - Protect at all costs
  - Do not use for day-to-day operations
- Follow principle of least privilege
- Rotate access keys
- Enable Multi-Factor Authentication (MFA)
- Do not share credentials!

Detection

Capturing logs

Analyzing logs

Auditing

Notification

Detection

> “You can use detective controls to identify a potential security threat or incident. They are an essential part of governance frameworks and can be used to support a quality process, a legal or compliance obligation, and threat identification and response efforts.”

Security Pillar: AWS Well-Architected Framework (AWS Whitepaper) (Kindle Locations 197-198). Amazon Web Services. Kindle Edition.

Logs Generation

ELB access logs

Apps on EC2 or ECS/EKS

Lambda logs

S3 access logs

CloudFront access logs

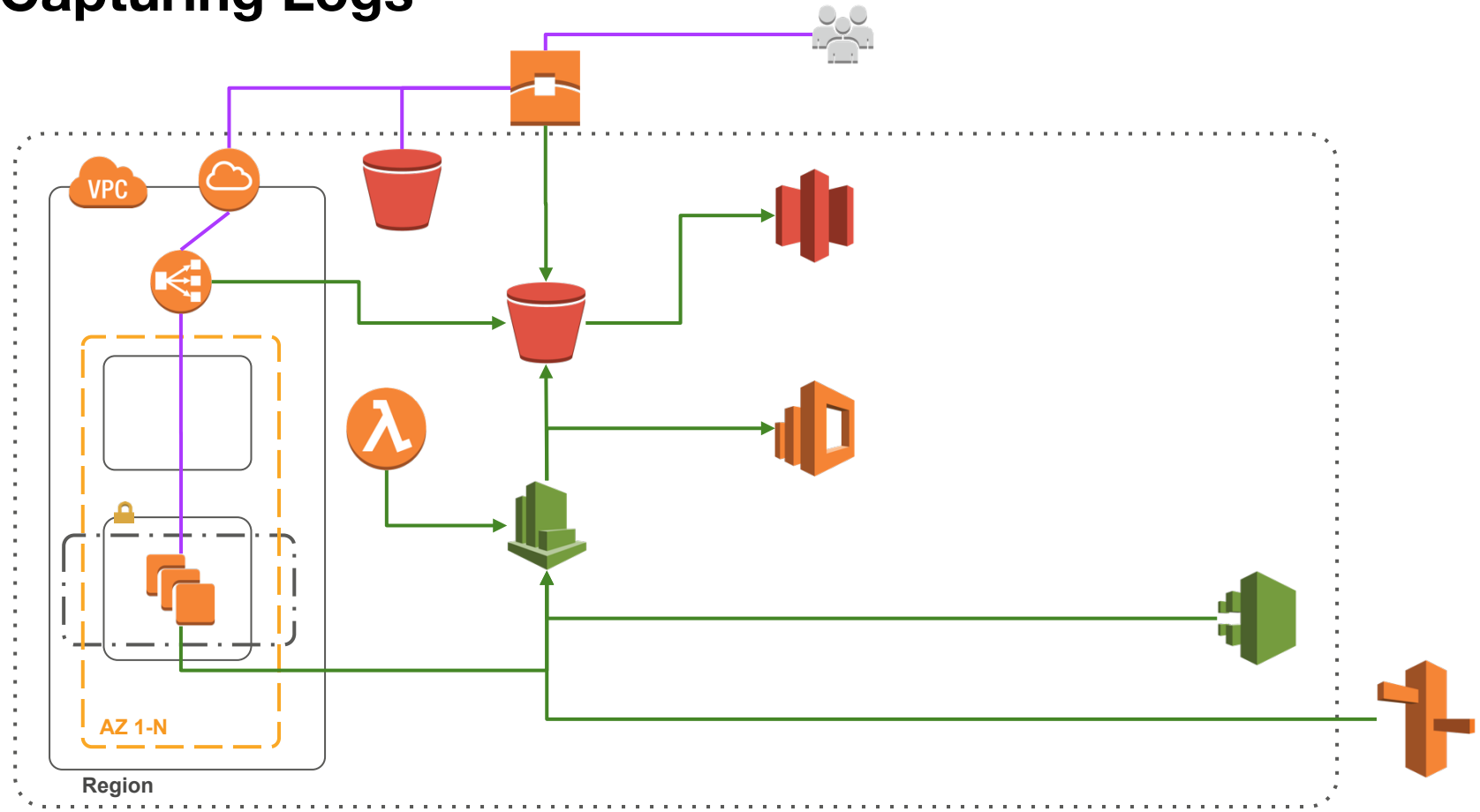
CloudTrail logs

Where can we expect logs to come from in AWS?

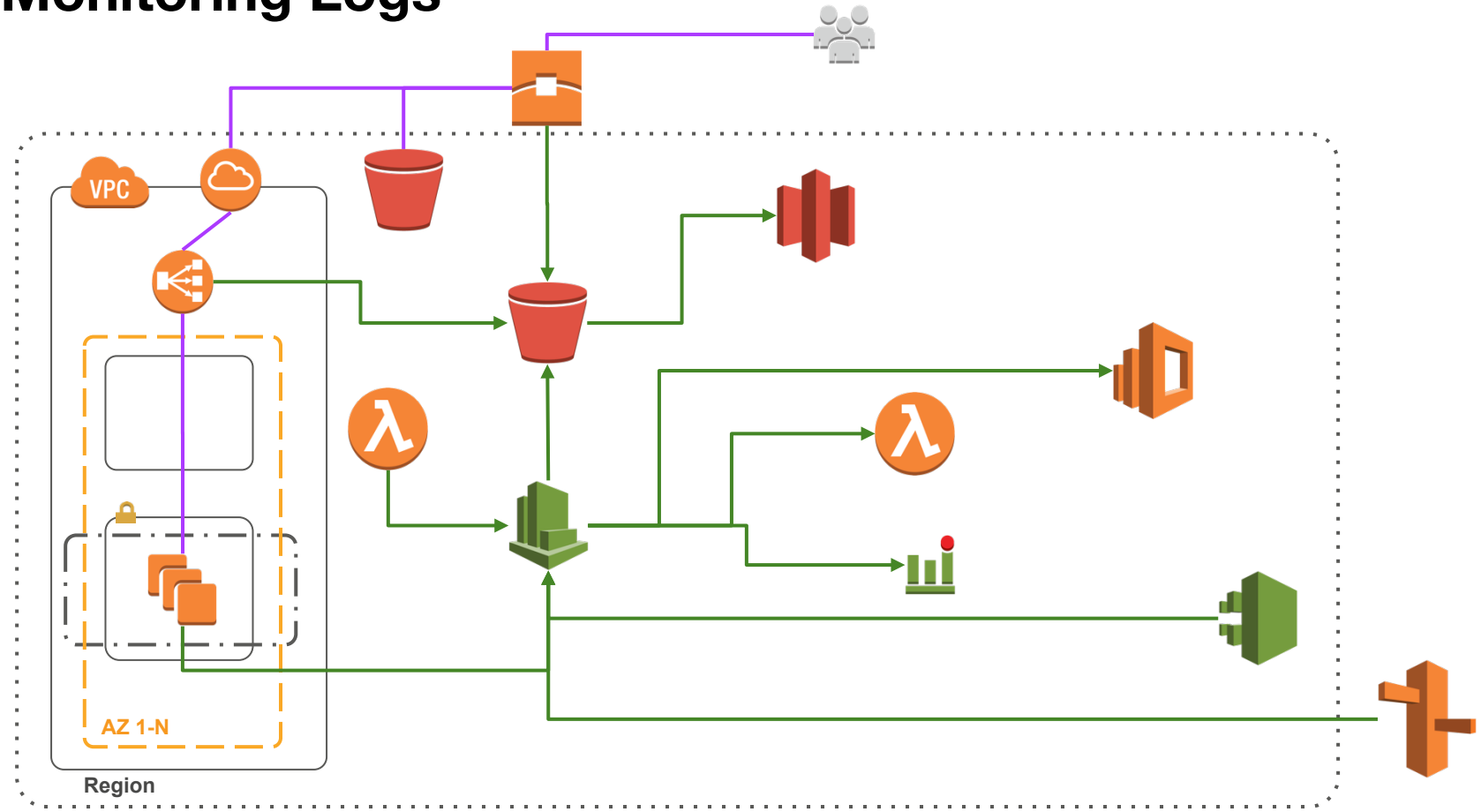
Many places!

Pearson

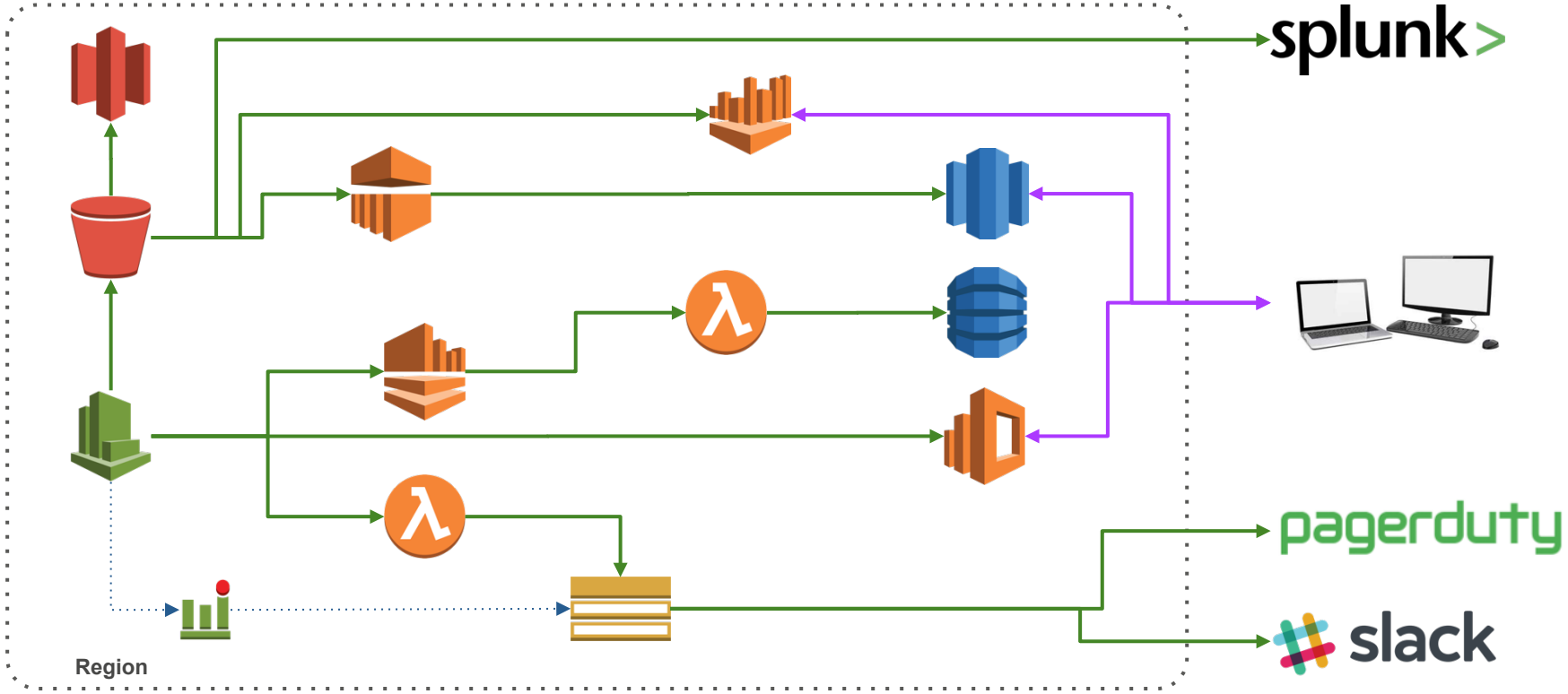
Capturing Logs



Monitoring Logs



Analyzing Logs



Securing Infrastructure

VPC

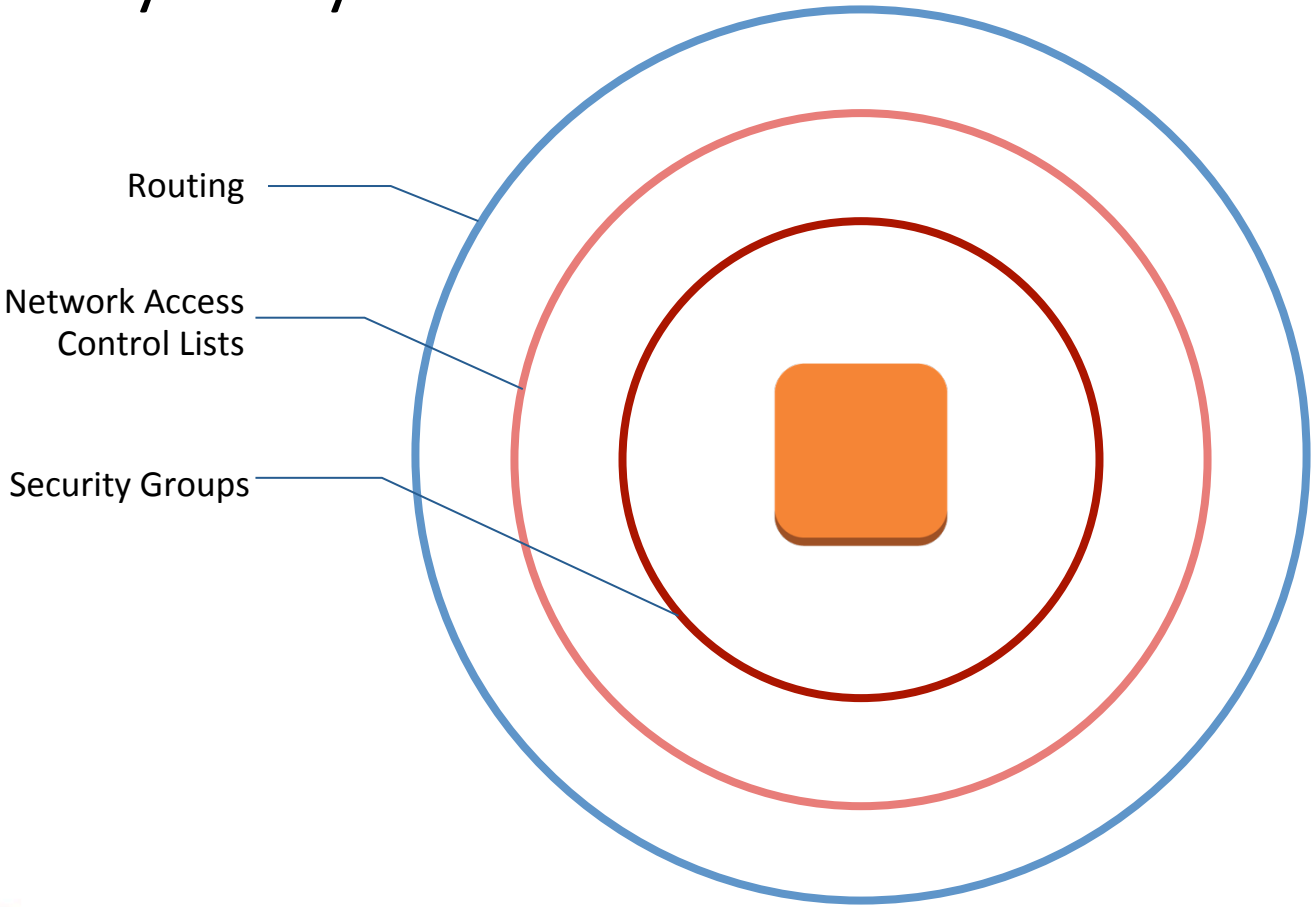
Routing

Network ACLs

Security Groups

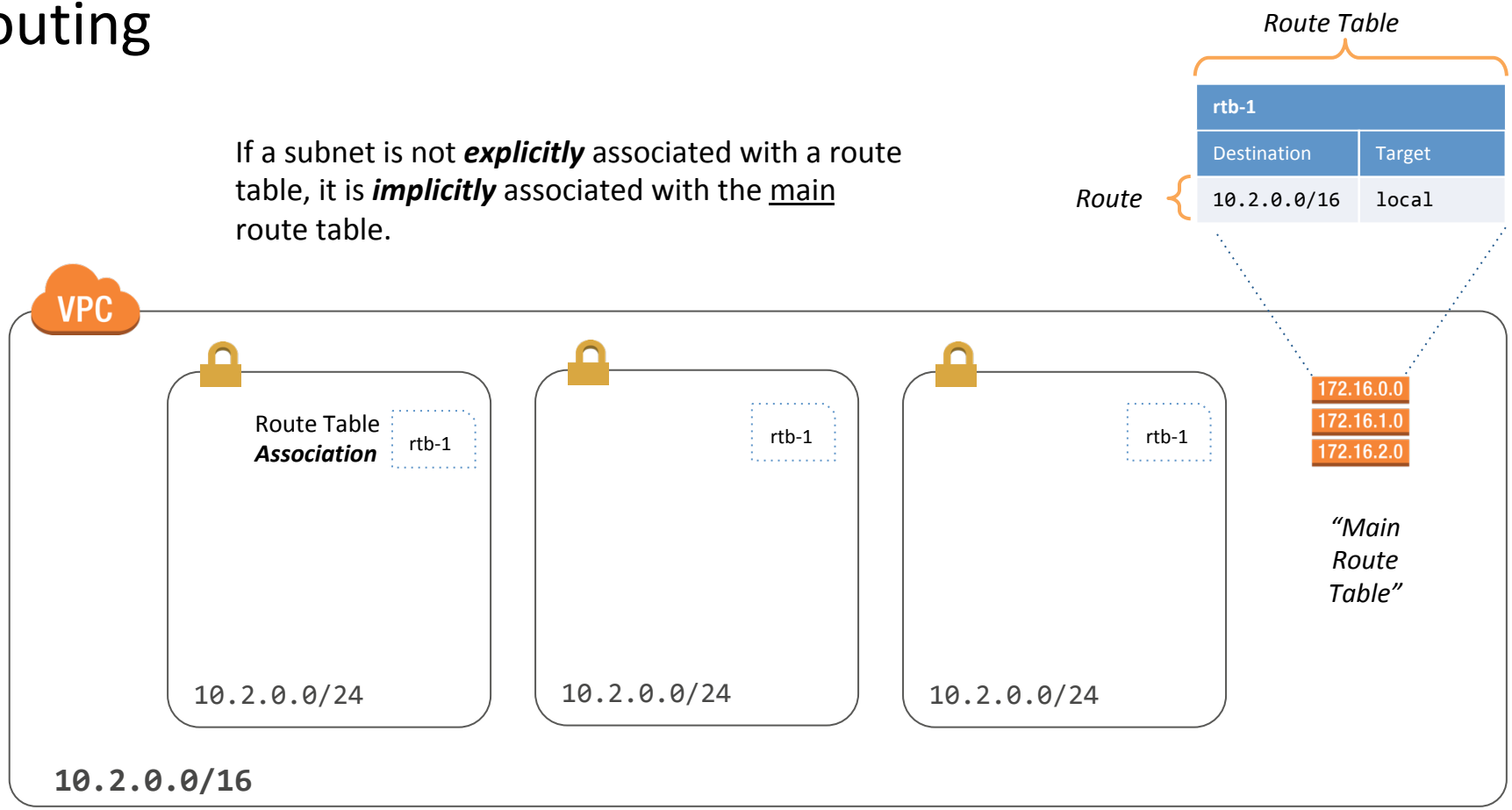
OS-based Security

Security in Layers



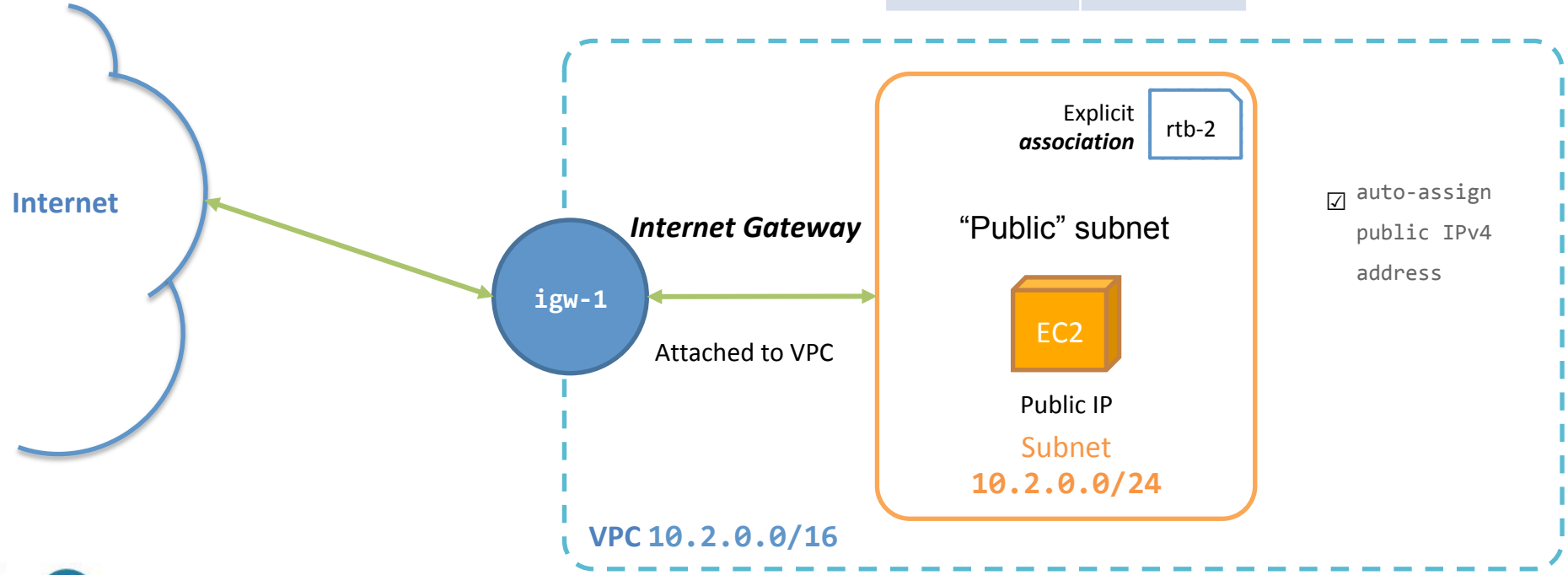
Routing

If a subnet is not ***explicitly*** associated with a route table, it is ***implicitly*** associated with the main route table.



Internet Access

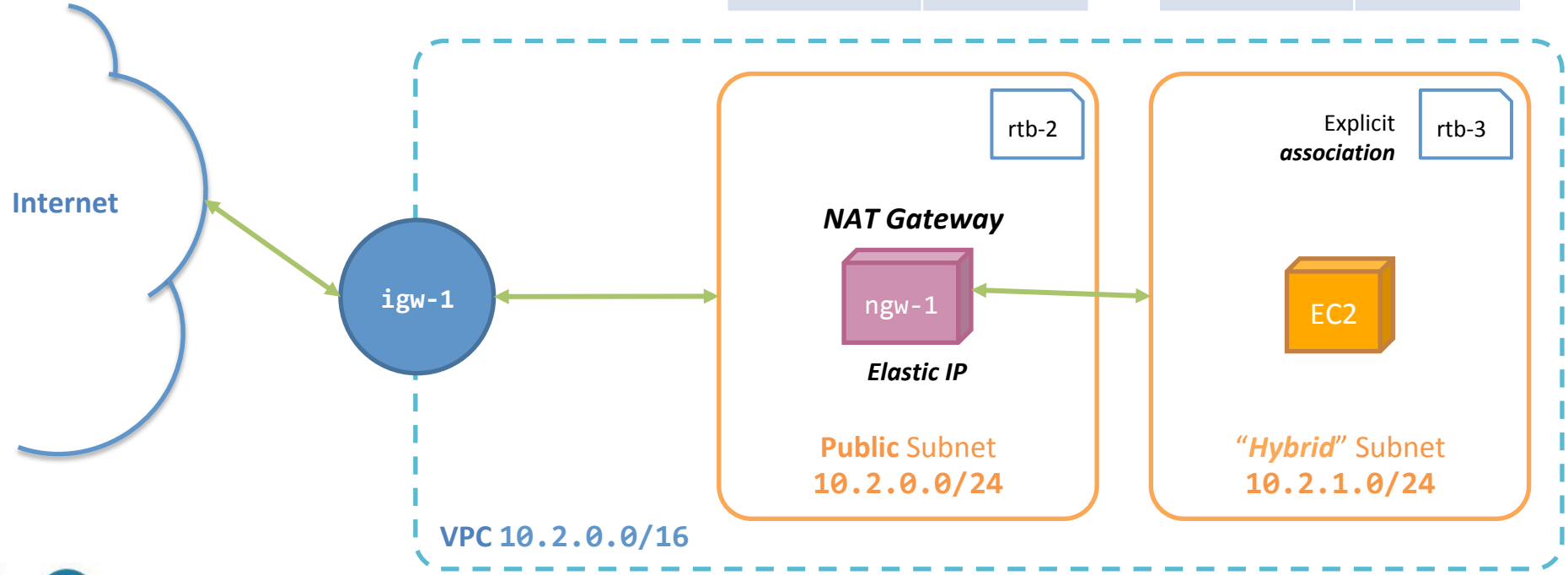
rtb-2	
Destination	**Target**
10.2.0.0/16	local
0.0.0.0/0	**igw-1**



“Hybrid” Subnets with NAT

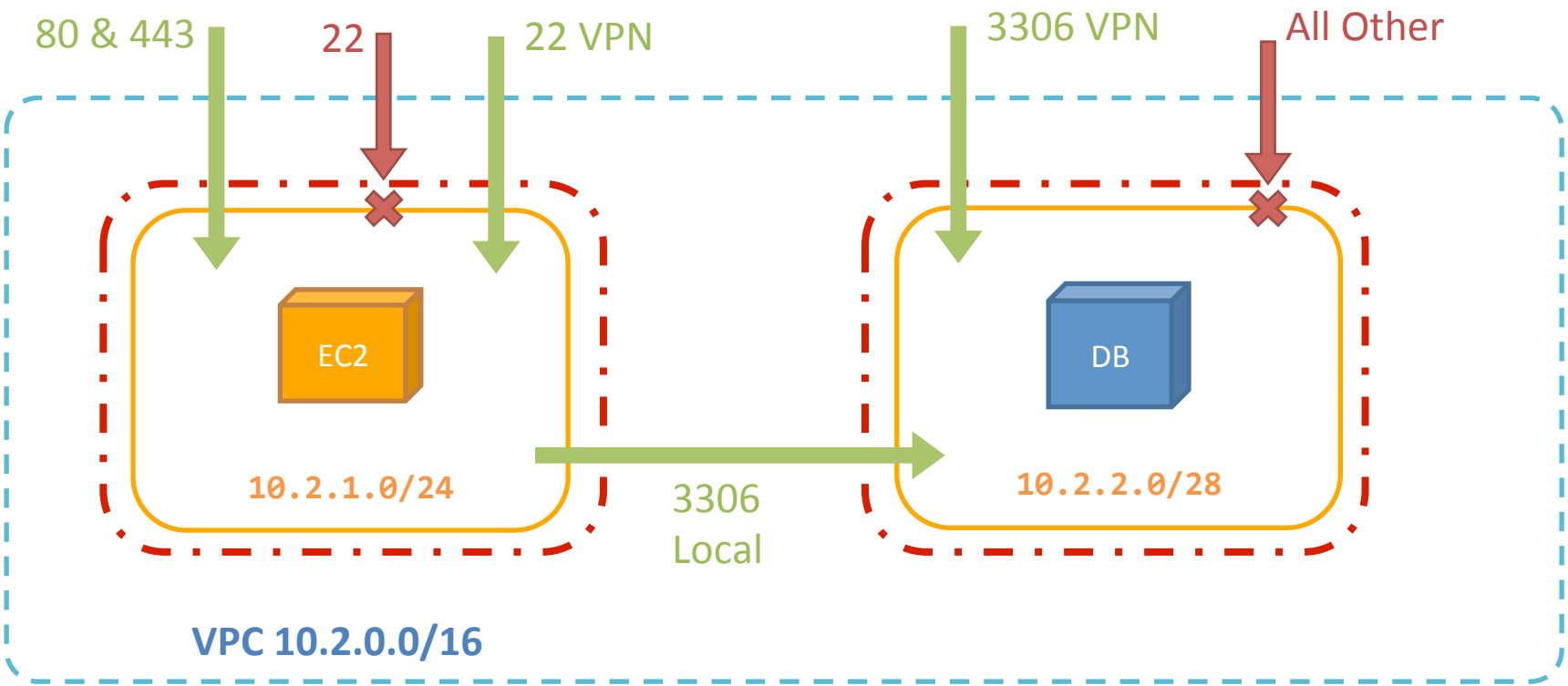
rtb-2	
Destination	**Target**
10.2.0.0/16	local
0.0.0.0/0	**igw-1**

rtb-3	
Destination	**Target**
10.2.0.0/16	local
0.0.0.0/0	**ngw-1**



NAT = Network Address Translation

Network Access Control Lists (NACL)



Network Access Control Lists (NACL)

- Applied to ***subnet*** as a whole
- ***Stateless***
- *Must* specify ingress *and* egress
- All traffic allowed by default
- Allow or Deny
- Specify:
  - Protocol
  - Source IP range
  - Destination port range

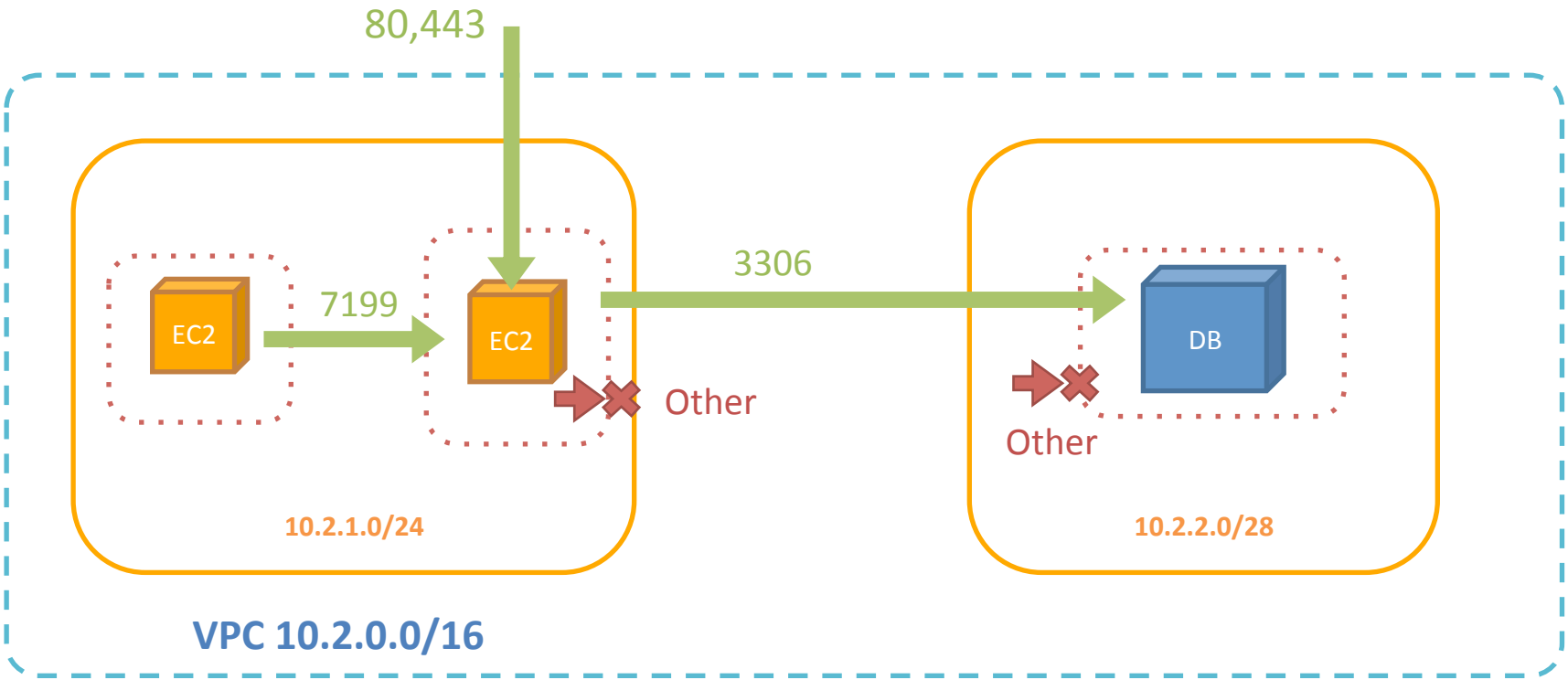
Ingress Rules

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

Security Groups



Security Groups

- Applied to ***instance***
- ***Stateful***
- *Can* specify ingress *or* egress
- *Denied by default*
- Specify:
  - 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-abcdfghi

* Applies to ***initiating*** connections, not responses

Securing Data

Resource Based Policies

Cross-Account Resource Sharing

Publicly Accessible Buckets

Sharing Snapshots

Encryption at Rest

Encryption in Transit

Bucket Security with Resource Policies

Resource Policies

- Same as IAM policies
- Applied at *resource level*
  - Amazon Simple Storage Service
  - Amazon Simple Queue Service
  - Amazon EC2 Container Registry
  - Many more…
- Specify a *Principal* (who)
  - Another account
  - IAM user, group, role
  - AWS Service
  - Anonymous

Example Bucket Public Read Policy

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

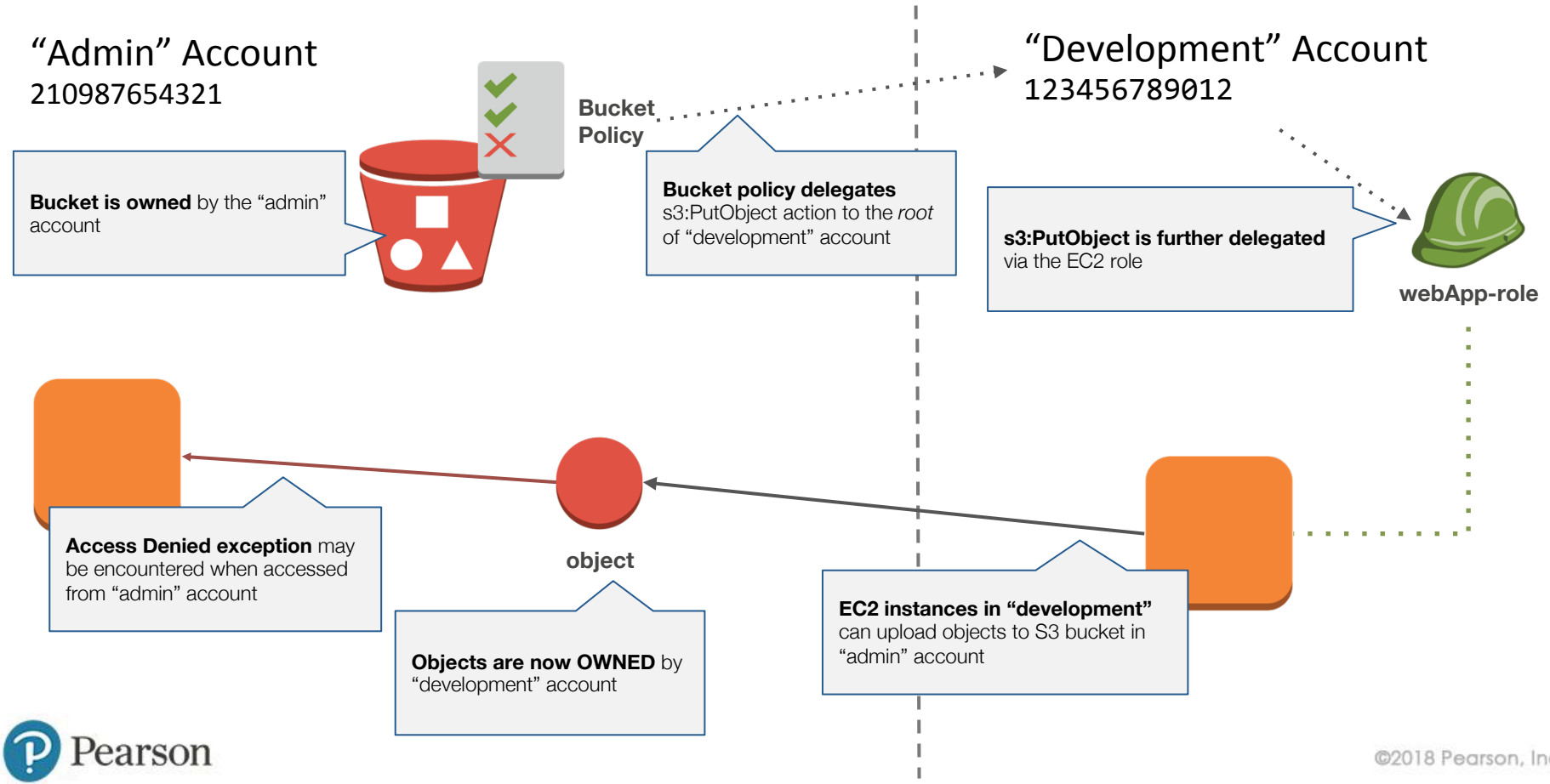
Anyone in the world is allowed to download objects from anywhere in this S3 bucket

Example Bucket Cross-Account Policy

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {"AWS": "123456789012"},
      "Action": "s3:PutObject",
      "Resource": [
        "arn:aws:s3:::app-logs",
        "arn:aws:s3:::app-logs/*"
      ]
    }
  ]
}
```

The specified AWS account is allowed to upload objects to anywhere in this S3 bucket, and *delegate that permission to its users or roles.*

Bucket vs Object Ownership



Enforce Bucket Owner Access to Objects

```
...
    {
      ...
      "Condition": {
        "StringNotEquals":
          {"s3:x-amz-acl":"bucket-owner-full-control"}
      }
    }
...
```

ACLs apply to bucket and objects and provide a coarse-grained way to control permissions.

ACLs can be applied at time of PutObject request or modified later.

```
bash$ aws s3 cp --acl bucket-owner-full-control
```

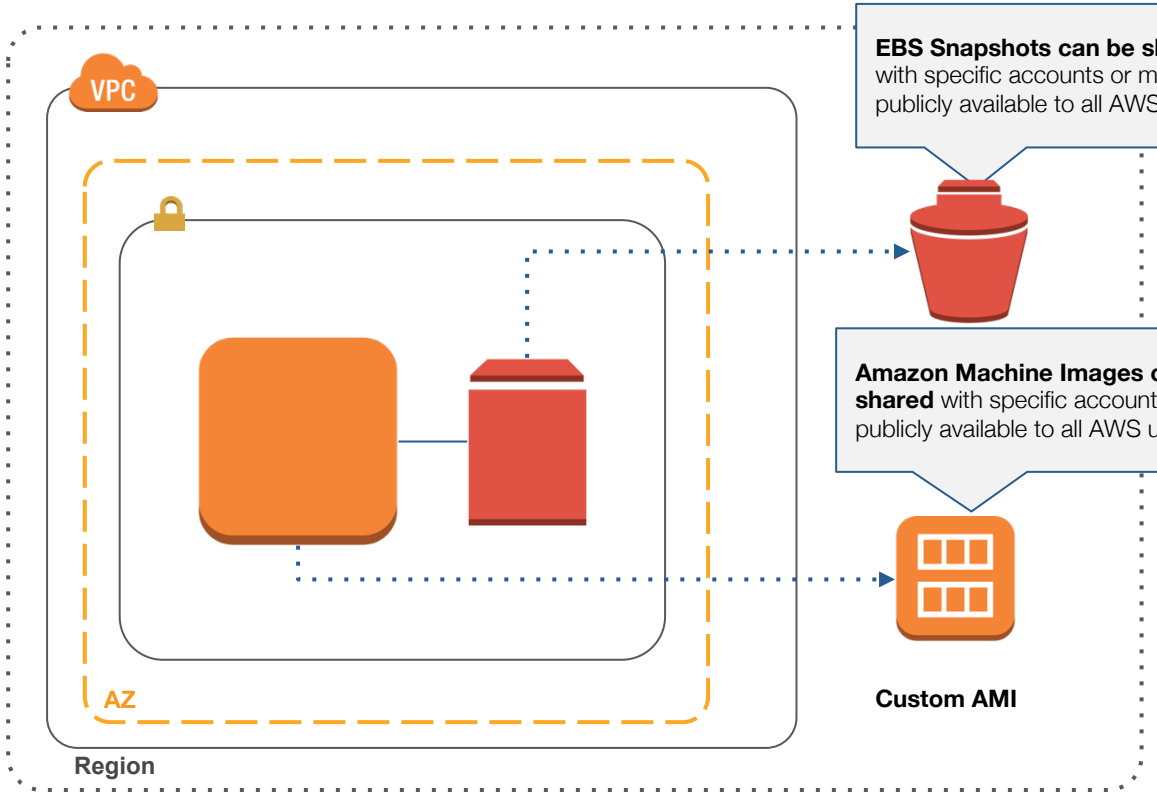
Making a Bucket or Objects Publicly Readable

- Policy using `"Principle":"*"`
- Following **canned ACLs**
  - public-read
  - public-read-write
  - authenticated-read

authenticated-read allows *ANY AND ALL AWS users, including those of other accounts* in other organizations

Application of policies or ACLs are the **customer's responsibility**.

Sharing Snapshots and AMIs



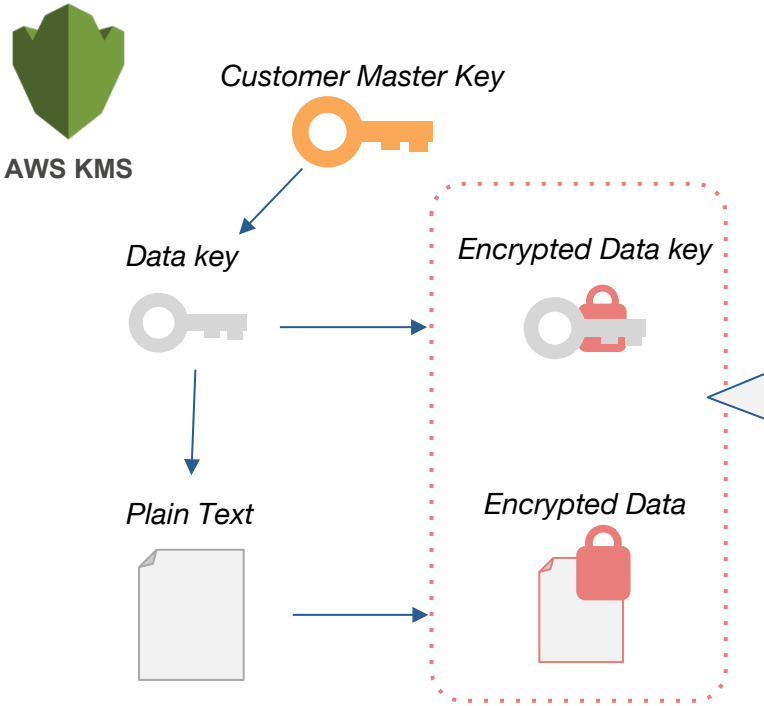
shared AMIs or snapshots at your own risk.

AWS will not vouch for the integrity or y.

Get an AMIs from a trusted source such as Canonical, RedHat, or Microsoft.

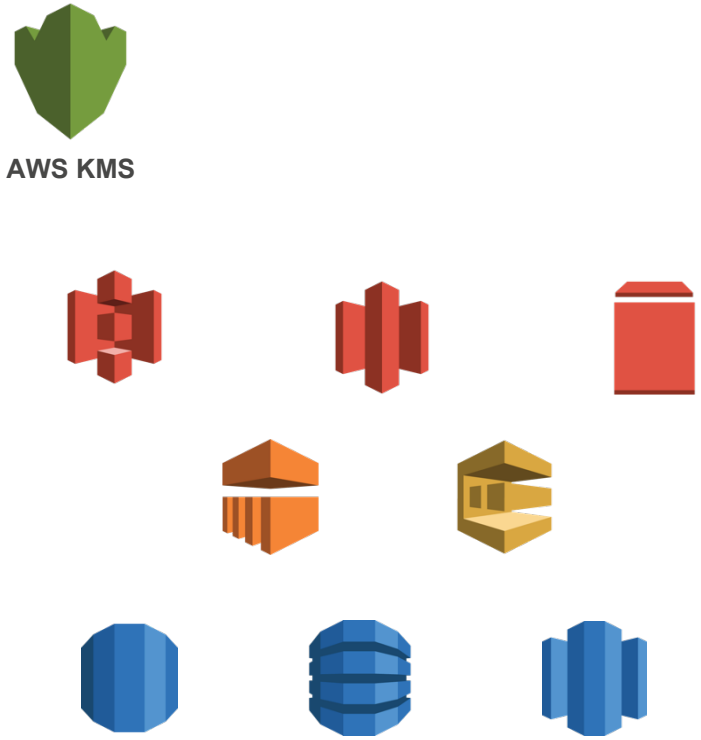
Anyone with access to your snapshot or AMI has **ALL THE DATA**.

Protecting Data at Rest



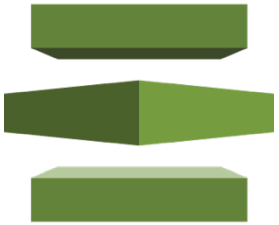
- Key Management Infrastructure
- Full-managed, very secure
- nt, software-based
- use to CloudTrail
- r leaves KMS
- Integrates with numerous services
- SOC 1, SOC 2, SOC 3, PCI DSS lvl 1

Protecting Data at Rest



- S3: key per object
- Glacier: key per archive
- EBS: key per volume
- Redshift: 4-tier envelope encryption
- RDS: builds on EBS encryption

Protecting Data in Transit



AWS Certificate Manager

- Create *free* SSL certs
- Including wildcards
- Import existing certs
- Automatically deploys to
  - Elastic Load Balancers
  - CloudFront
- Automatically renews
- Starfield root authority

Cost Optimization

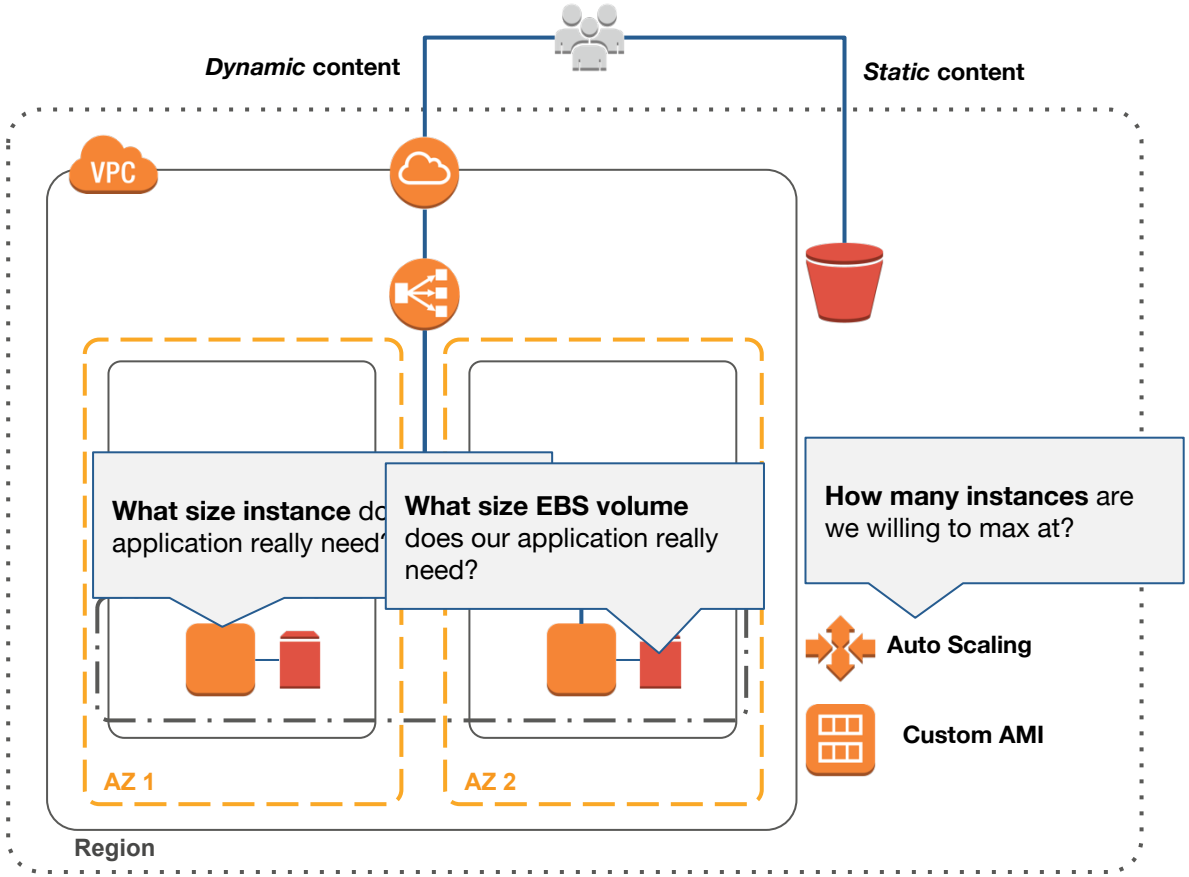
Cost-effective resources

Matching supply with demand

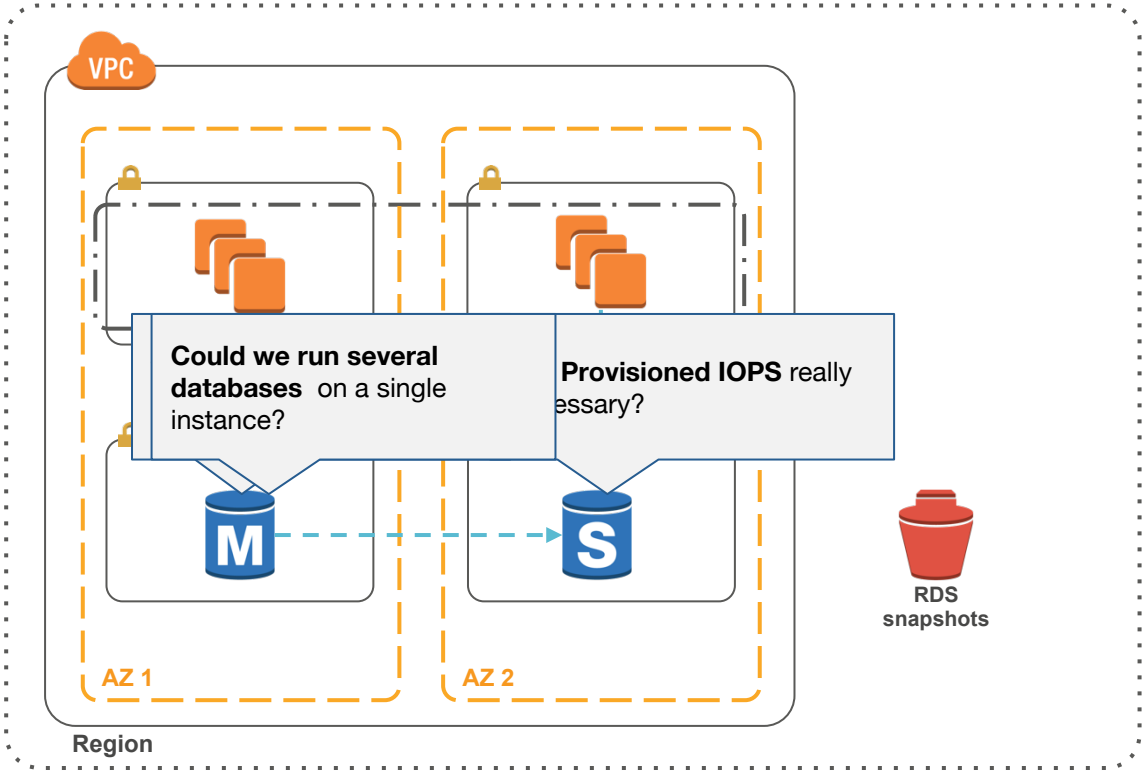
Expenditure awareness

Optimizing over time

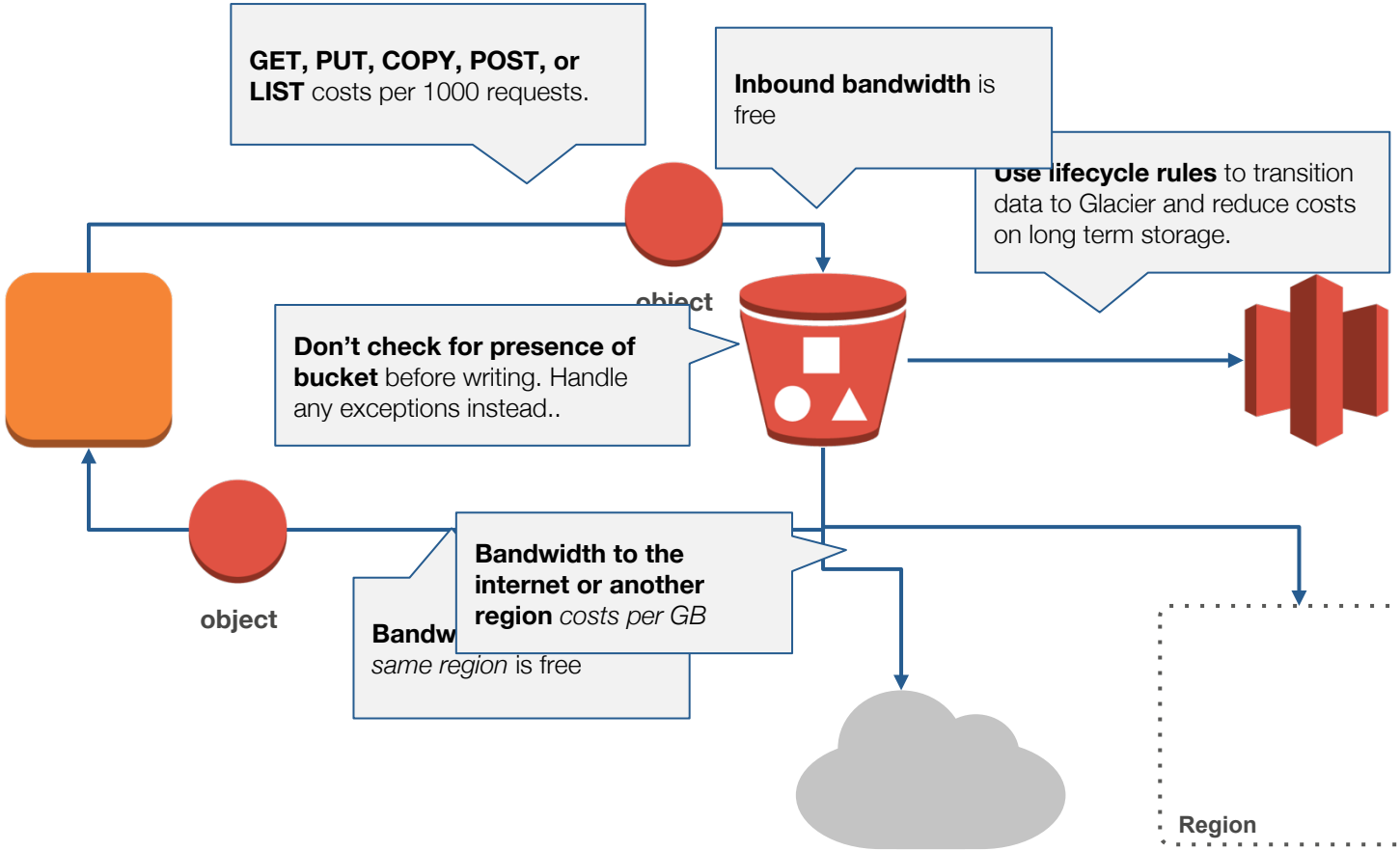
Right Sizing EC2



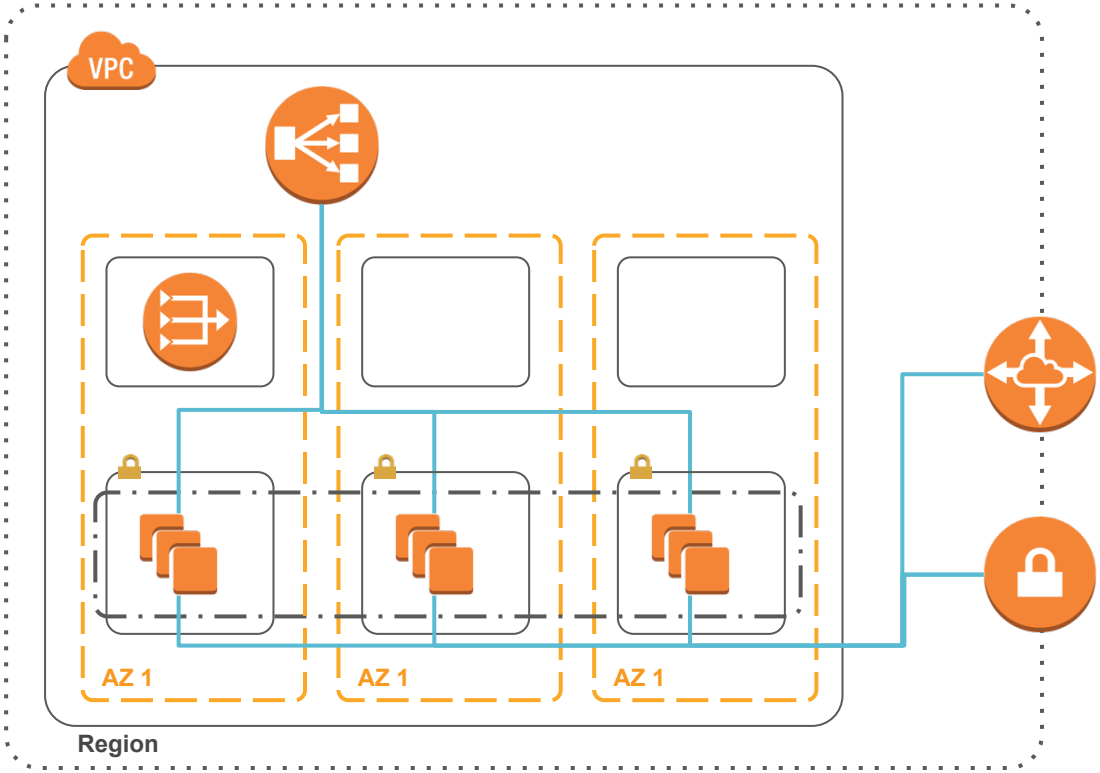
Right Sizing RDS



S3 Costs



Network Costs

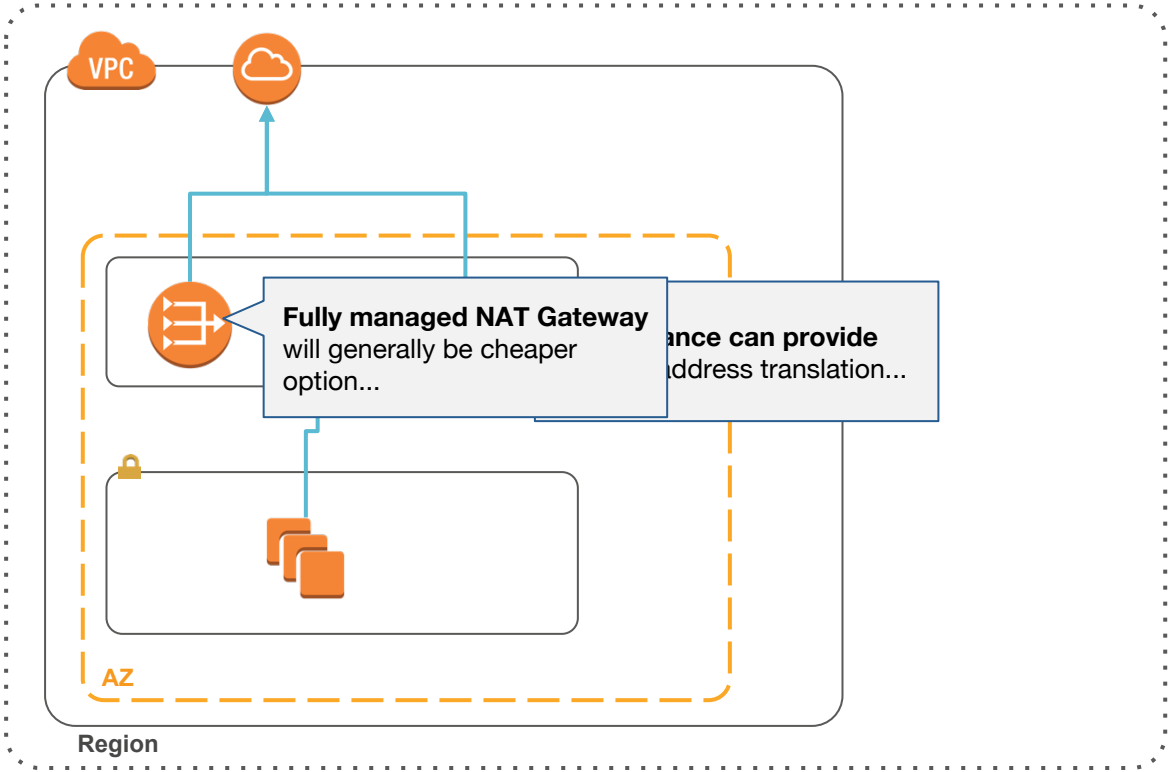


Most managed devices:

- ELB
- NAT Gateway
- VPN Connections
- Direct Connect

Have both hourly and bandwidth charges.

NAT Instance vs Gateway



NAT Gateway capable of 45Gbps

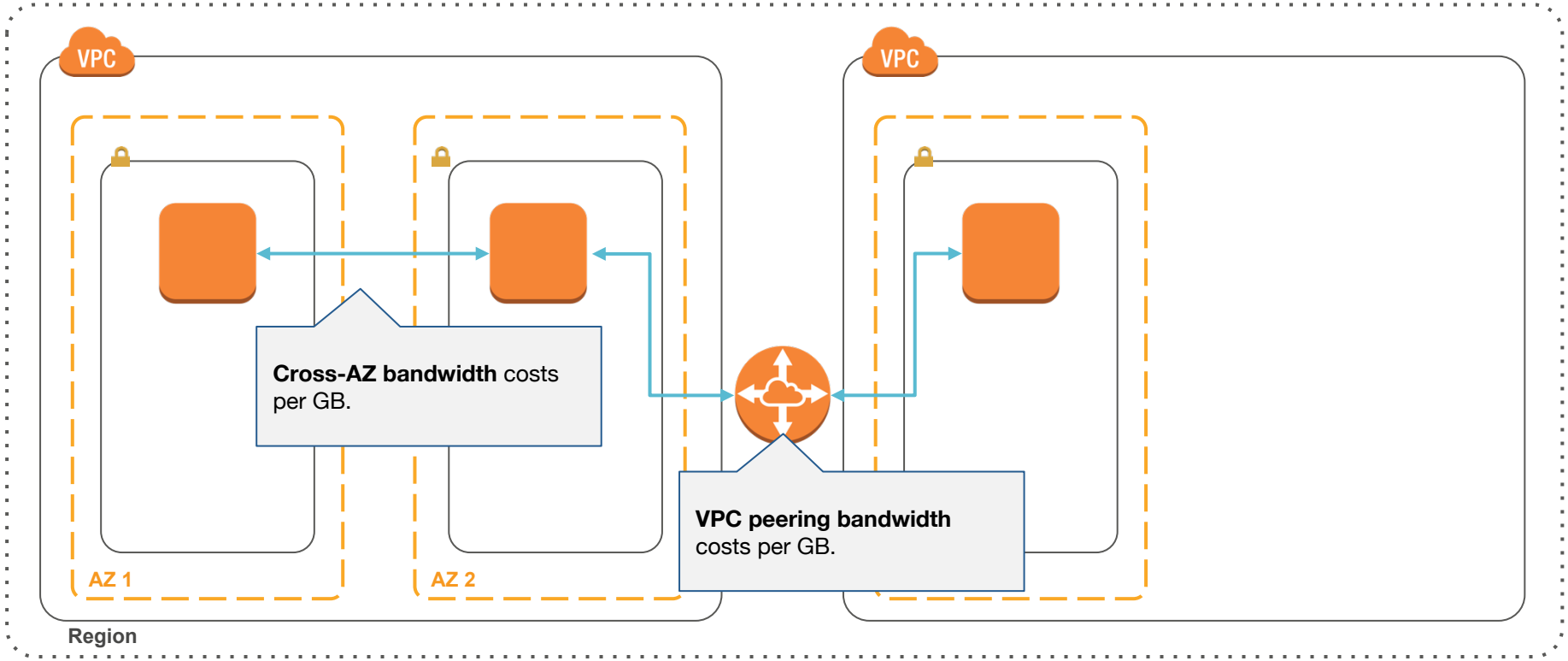
Fully managed

Fault tolerant, horizontally scaled

Closest instance is m4.16xlarge

with 25Gbps ~70x more expensive!

Cross AZ Traffic and VPC Peering



Billing Options

	On-Demand	Reserved Instances			Spot Instances
		No Up Front	Partial Up Front	Full Up Front	
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

Billing Options

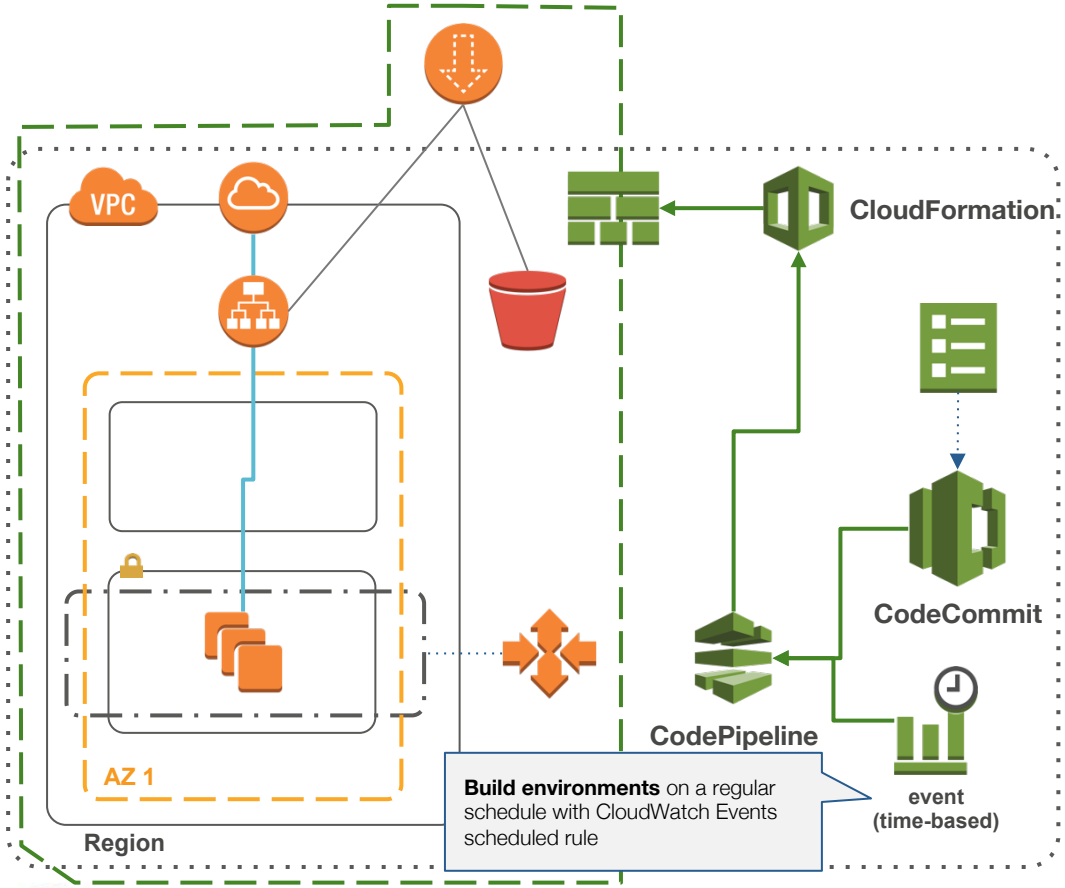
Reserved Instance

- 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

Temporary Environments



Why pay for an environment that's not being used?

Turn out the lights!

Build an environment *when it's needed*.

Delete the environment when it's not needed.

Rinse and repeat.

Operational Excellence

Preparation

Operation

Evolution

Operational Excellence

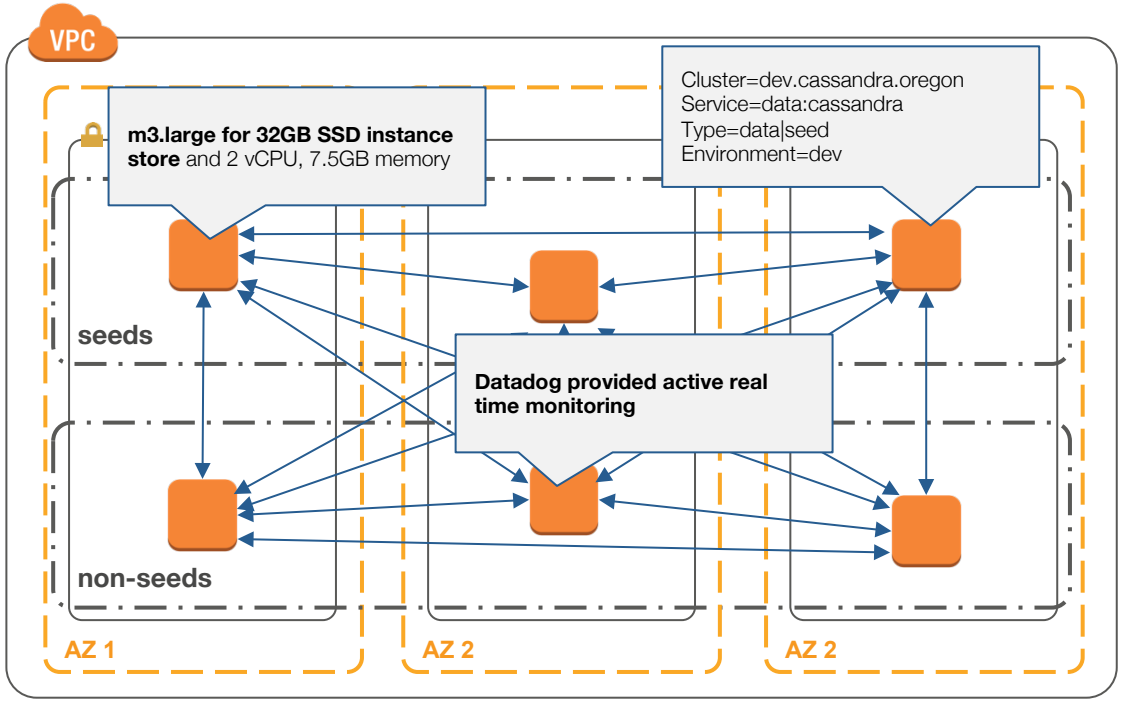
> “To prepare for operational excellence you have to understand your workloads and their expected behaviors. You will then be able design them to provide insight to their status and build the procedures to support them.”

Operational Excellence Pillar: AWS Well-Architected Framework (AWS Whitepaper) (Kindle Locations 95-97). Kindle Edition

Design Principles

- Operations as code
- Automate creation of annotated documentation
- Changes should be frequent, small, reversible
- Refine procedures frequently
- Anticipate failure, test and validate
- Learn from failures

Cassandra



Instance stores are ***ephemeral***

Durability via 3x replication

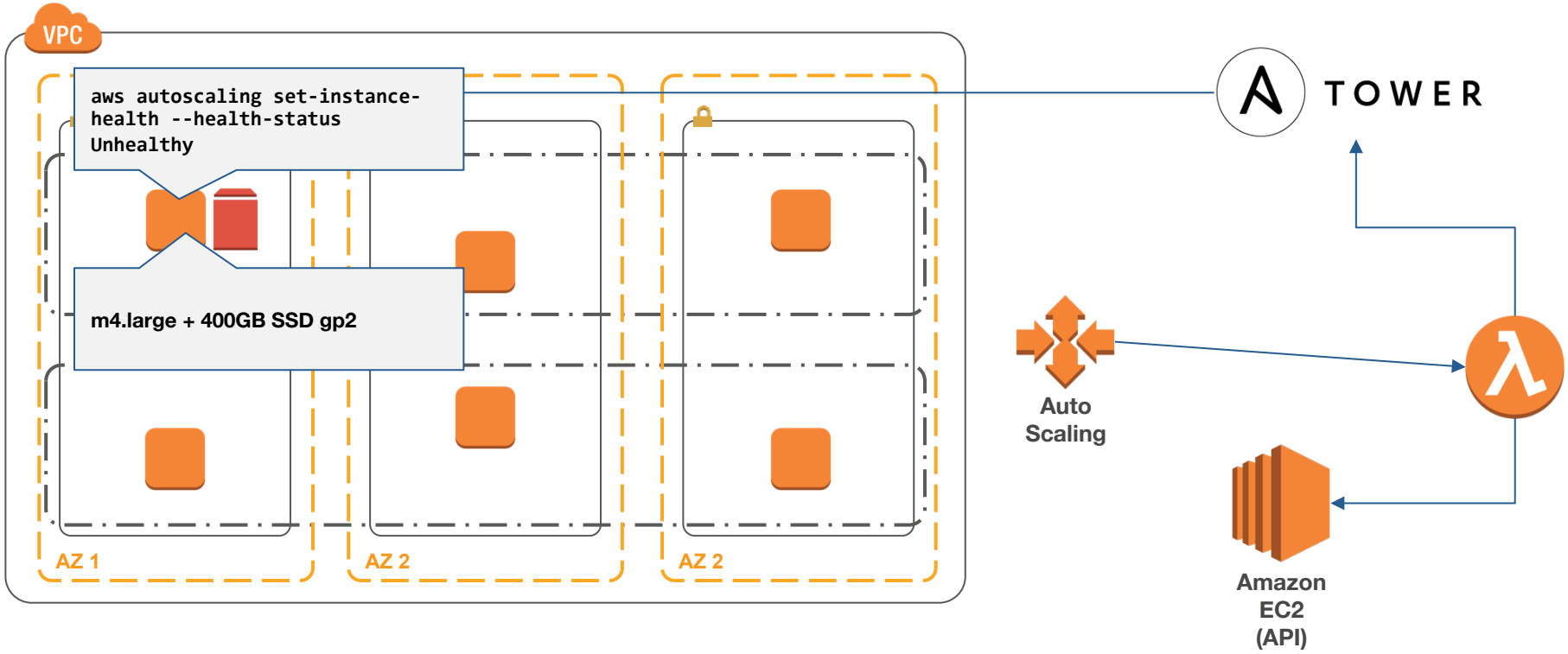
Availability achieved through multiple ASGs

- 1 for seeds
- 1 for non-seeds

Cassandra config and process management via Ansible

Tags for organization and automation

Cassandra



That's All Folks

Thank you for attending!

For more training:

- Amazon Web Services Fundamentals
- Automation in AWS with Cloudformation,CLI, and SDKs
- Networking in AWS

Images provided by Fotolia:

https://us.fotolia.com/id/74146873

https://us.fotolia.com/id/70757791

https://us.fotolia.com/id/109163355

A Note About This Class

This Class Covers

- In-depth technical details

This Class Does Not Cover

- Fundamentals
- Specific implementations
- Every service…

For More In-Depth Training

- AWS Fundamentals
- Automation in AWS with Cloudformation,CLI, and SDKs
- Networking in AWS