



Intro to AWS and Infrastructure as Code

Andrew May
Ohio LinuxFest

About me

- Cloud Solutions Lead at Leading EDJE
- 5 years AWS experience
- Regular presenter at Columbus AWS Meet-up
- AWS Certified





Cloud Computing



Intro to AWS

Basic Security

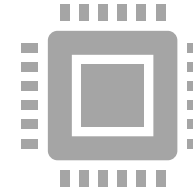
Setting up Account

Compute

Autoscaling and Load Balancers

Storage

Databases



Infrastructure as Code


CloudFormation

Terraform

Agenda for Today

Introductions

- Name
- What you want to get out of this course
- Level of AWS experience (if any)



What is Cloud Computing?

Cloud computing is the **on-demand** delivery of compute power, database storage, applications, and other IT resources through a cloud services platform **via the internet** with **pay-as-you-go** pricing.
(at least according to AWS)



Everything is an API

- Use an API to create Infrastructure on-demand
- Everything (at least on AWS) uses the same API:
 - Web Console
 - Command Line Interface
 - SDK
- APIs enable Infrastructure as Code

Compare to On-Premise

On-Premise

- Capital expenditure
- Planning
 - Max capacity
 - Procurement cycle
- Datacenter management
 - HVAC
 - Space
 - Staff
 - Security

Cloud Platform

- Operational expenditure
- Infrastructure on-demand
 - Scale as needed
 - Stop guessing about capacity
- No up-front costs*
- Speed and agility
- Global on-demand

IaaS: Infrastructure as a Service

PaaS: Platform as a Service

SaaS: Software as a Service

Cloud Models

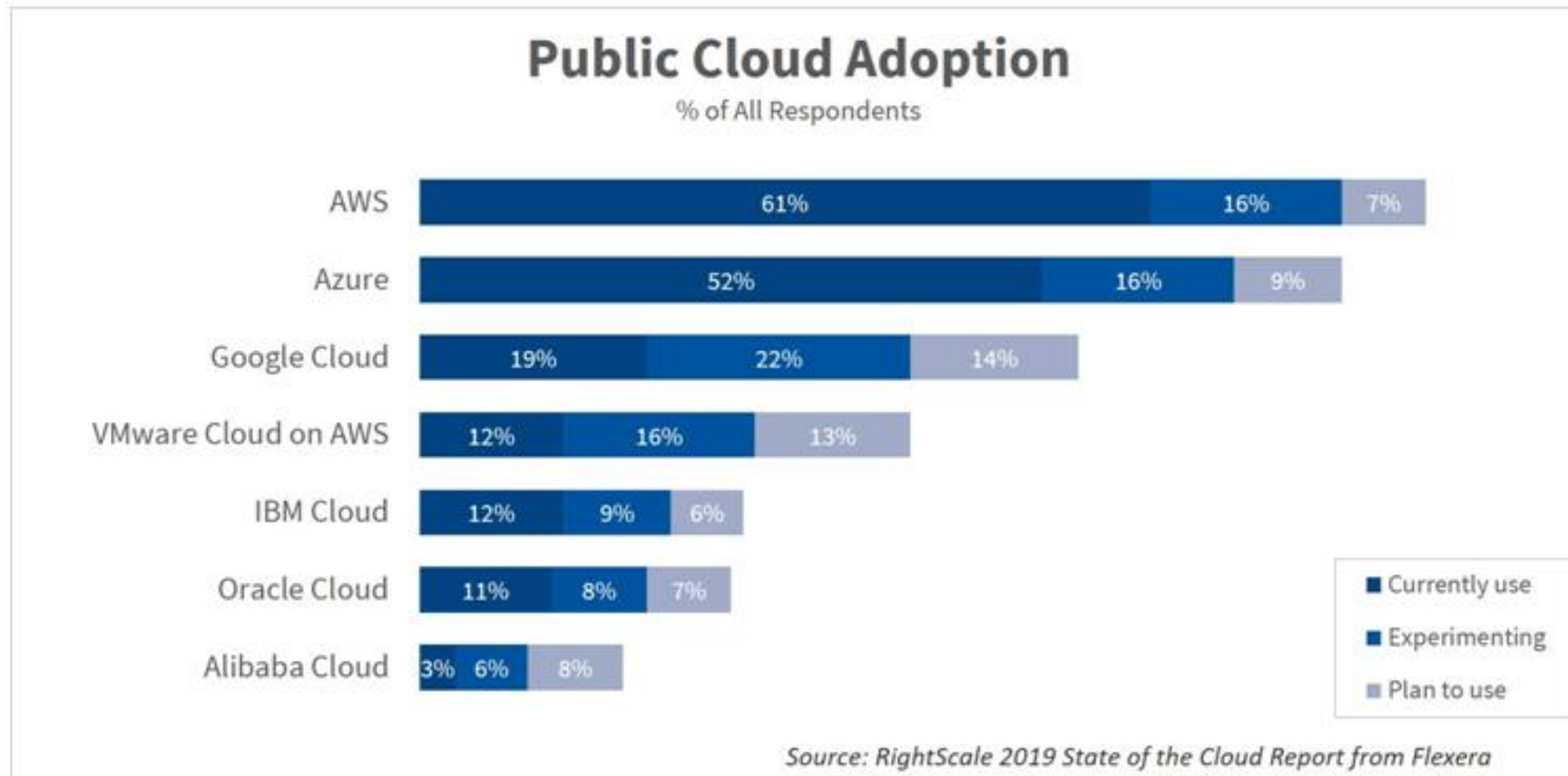
All-in Cloud

Hybrid

Private Cloud (on-premise)

Deployment Models

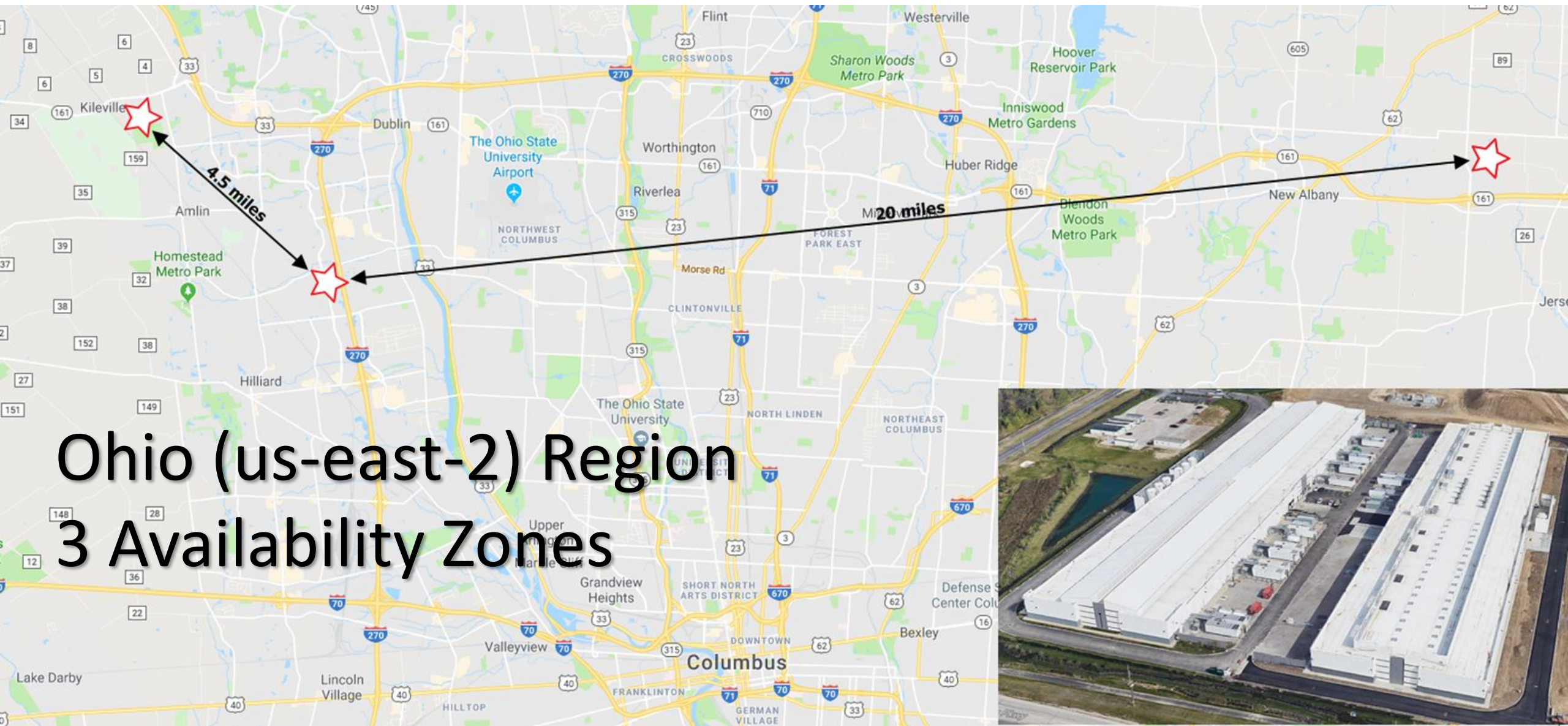
Major Cloud Providers

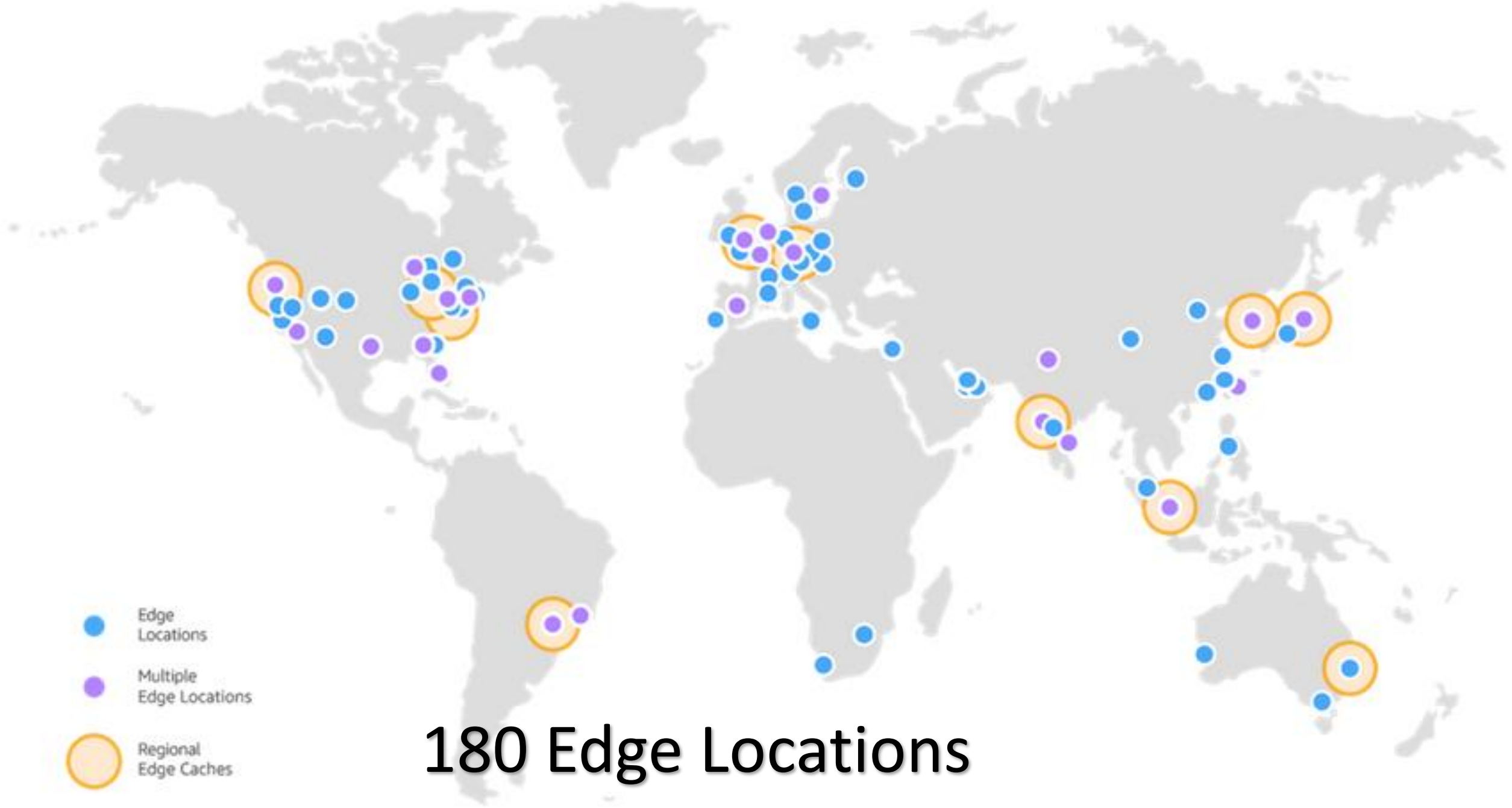




Intro to AWS



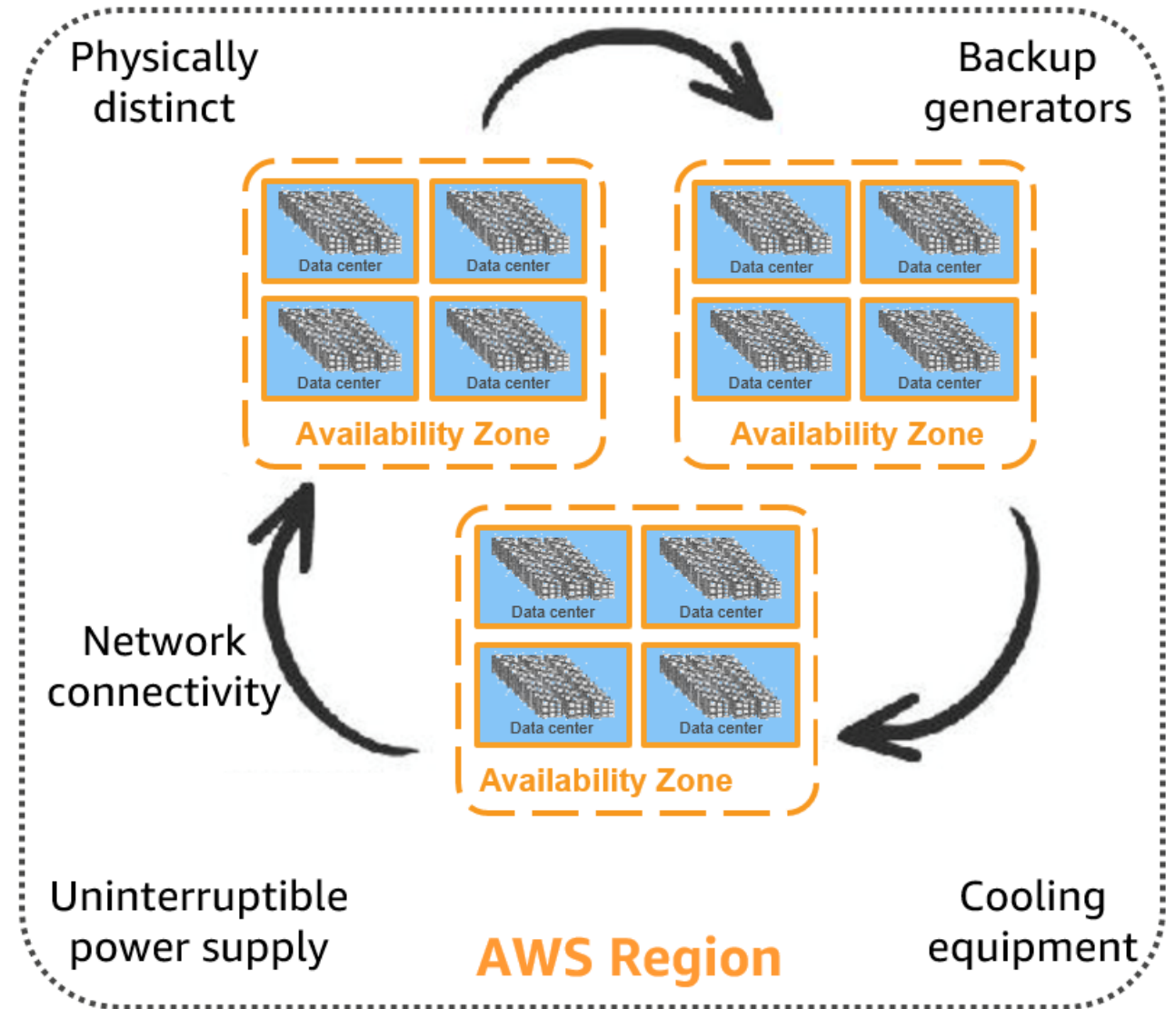




180 Edge Locations

Region High Availability

- Many services span the availability zones within a region to provide high availability.
- In some cases (e.g. VPC, EC2) you must explicitly declare the AZs you're going to use.
- Others (e.g. DynamoDB) span AZs transparently.



Service Categories



Analytics



Application Integration



AR & VR



AWS Cost Management



Blockchain



Business Applications



Compute



Customer Engagement



Database



Developer Tools



End User Computing



Game Tech



Internet of Things



Machine Learning



Management & Governance



Media Services



Migration & Transfer



Mobile



Networking & Content
Delivery



Robotics



Satellite



Security, Identity &
Compliance



Storage

Compute



Compute

Amazon EC2

Virtual Servers in the Cloud

Amazon EC2 Auto Scaling

Scale Compute Capacity to Meet Demand

Amazon Elastic Container Registry

Store and Retrieve Docker Images

Amazon Elastic Container Service

Run and Manage Docker Containers

Amazon Elastic Kubernetes Service

Run Managed Kubernetes on AWS

Amazon Lightsail

Launch and Manage Virtual Private Servers

AWS Batch

Run Batch Jobs at Any Scale

AWS Elastic Beanstalk

Run and Manage Web Apps

AWS Fargate

Run Containers without Managing Servers or Clusters

AWS Lambda

Run your Code in Response to Events

AWS Outposts

Run AWS services on-premises

AWS Serverless Application Repository

Discover, Deploy, and Publish Serverless Applications

VMware Cloud on AWS

Build a Hybrid Cloud without Custom Hardware

Database



Database

Amazon Aurora
High Performance Managed Relational Database

Amazon DynamoDB
Managed NoSQL Database

Amazon DocumentDB (with MongoDB compatibility)
Fully managed document database

Amazon ElastiCache
In-memory Caching System

Amazon Neptune
Fully Managed Graph Database Service

Amazon Quantum Ledger Database (QLDB)
Fully managed ledger database

Amazon RDS
Managed Relational Database Service for MySQL, PostgreSQL, Oracle, SQL Server, and MariaDB

Amazon RDS on VMware
Automate on-premises database management

Amazon Redshift
Fast, Simple, Cost-effective Data Warehousing

Amazon Timestream
Fully managed time series database

AWS Database Migration Service
Migrate Databases with Minimal Downtime

Network and Content Delivery



Networking & Content Delivery

Amazon VPC
Isolated Cloud Resources

Amazon API Gateway
Build, Deploy, and Manage APIs

Amazon CloudFront
Global Content Delivery Network

Amazon Route 53
Scalable Domain Name System

AWS PrivateLink
Securely Access Services Hosted on AWS

AWS App Mesh
Monitor and control microservices

AWS Cloud Map
Application resource registry for microservices

AWS Direct Connect
Dedicated Network Connection to AWS

AWS Global Accelerator
Improve application availability and performance

AWS Transit Gateway
Easily scale VPC and account connections

Elastic Load Balancing
Distribute incoming traffic across multiple targets

Security, Identity and Compliance



Security, Identity & Compliance

AWS Identity & Access Management
Manage User Access and Encryption Keys

Amazon Cognito
Identity Management for your Apps

Amazon GuardDuty
Managed Threat Detection Service

Amazon Inspector
Analyze Application Security

Amazon Macie
Discover, Classify, and Protect your Data

AWS Artifact
On-demand access to AWS compliance reports

AWS Certificate Manager
Provision, Manage, and Deploy SSL/TLS Certificates

AWS CloudHSM
Hardware-based Key Storage for Regulatory Compliance

AWS Directory Service
Host and Manage Active Directory

AWS Firewall Manager
Central Management of Firewall Rules

AWS Key Management Service
Managed Creation and Control of Encryption Keys

AWS Resource Access Manager
Simple, secure service to share AWS resources

AWS Secrets Manager
Rotate, Manage, and Retrieve Secrets

AWS Security Hub
Unified security and compliance center

AWS Shield
DDoS Protection

AWS Single Sign-On
Cloud Single Sign-On (SSO) Service

AWS WAF
Filter Malicious Web Traffic

Storage



Storage

Amazon Simple Storage Service (S3)

Scalable Storage in the Cloud

Amazon Elastic Block Store (EBS)

EC2 block storage volumes

Amazon Elastic File System (EFS)

Fully managed file system for EC2

Amazon FSx for Lustre

High-performance file system integrated with S3

Amazon FSx for Windows File Server

Fully managed Windows native file system

Amazon S3 Glacier

Low-cost Archive Storage in the Cloud

AWS Backup

Centralized backup across AWS services

AWS Snow Family

Physical devices to migrate data into and out of AWS

AWS Storage Gateway

Hybrid Storage Integration



Security Basics

AWS Accounts

Accounts are isolated, unless explicit cross-account permissions are granted

Accounts span regions

- Government and China regions work differently

For this workshop you will each have your own account

- These belong to an organization where I own the root account

Anyone can create their own account

- You need to supply a credit card
- You will get some free services for a year, but some will cost money

Types of AWS Users

Root user (account owner)

- Always an administrator
- Authenticated with email address and password
- Each account must have a unique email address

IAM (Identity and Access Management) User

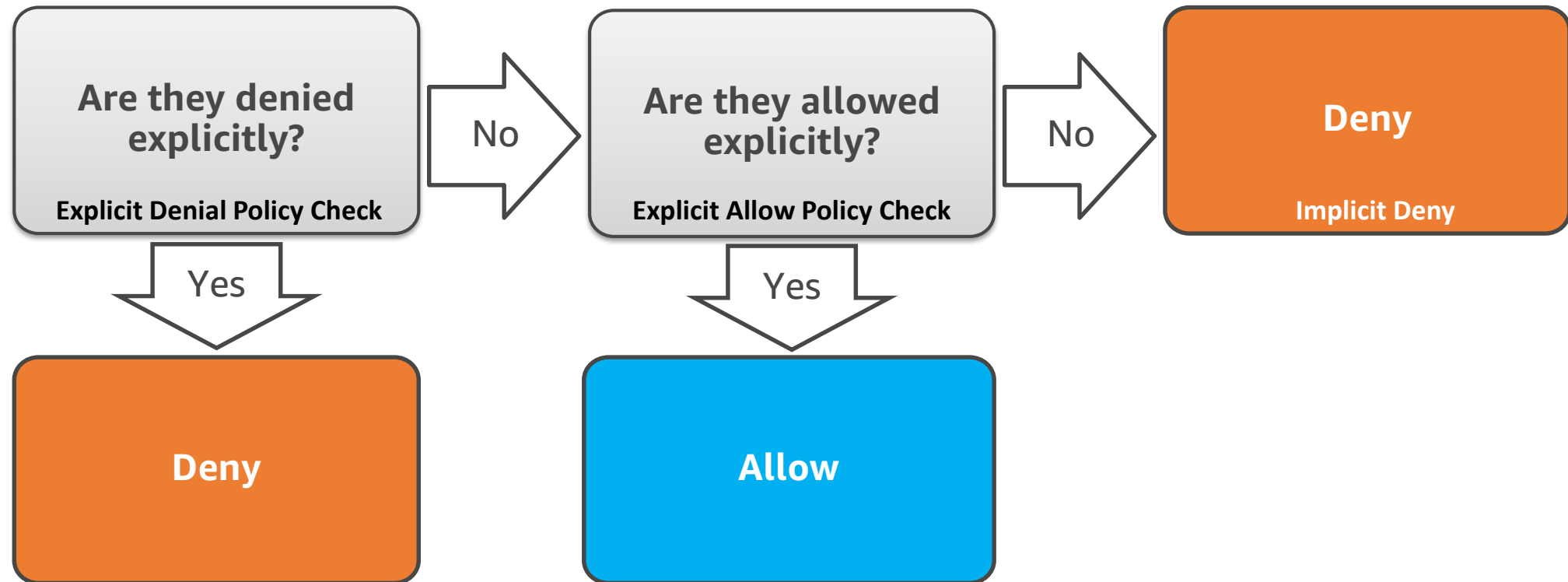
- Username and password (AWS Console)
- Access Key/Secret Key (CLI/SDK)

IAM Role

- Used to delegate access to AWS resources
e.g. allow the Lambda service to access a DynamoDB table

IAM Permissions

How IAM determines permissions:



Example IAM Statement

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": [
      "sns:Publish"      This matches name of API, operation in CLI and method in SDK
    ],
    "Resource": [
      "arn:aws:sns:us-east-1:1234567:aws-iot-button-sns"
    ]
  }
}
```

ARN = Amazon Resource Name

Region Account Id SNS Topic Name

Shared Responsibility Model



AWS is responsible for the security **of** the cloud



You are responsible for security **in** the cloud

**Customer
Responsibility**

Customer Data

Applications, IAM

Operating System, Network, and Firewall Configuration

Client-side data
encryption and data integrity
authentication

Server-side encryption
(File system and/or data)

Network traffic protection
(Encryption/ integrity /identity)

**AWS
Responsibility**

AWS Foundation Services

Compute

Storage

Database

Networking

AWS Global
Infrastructure

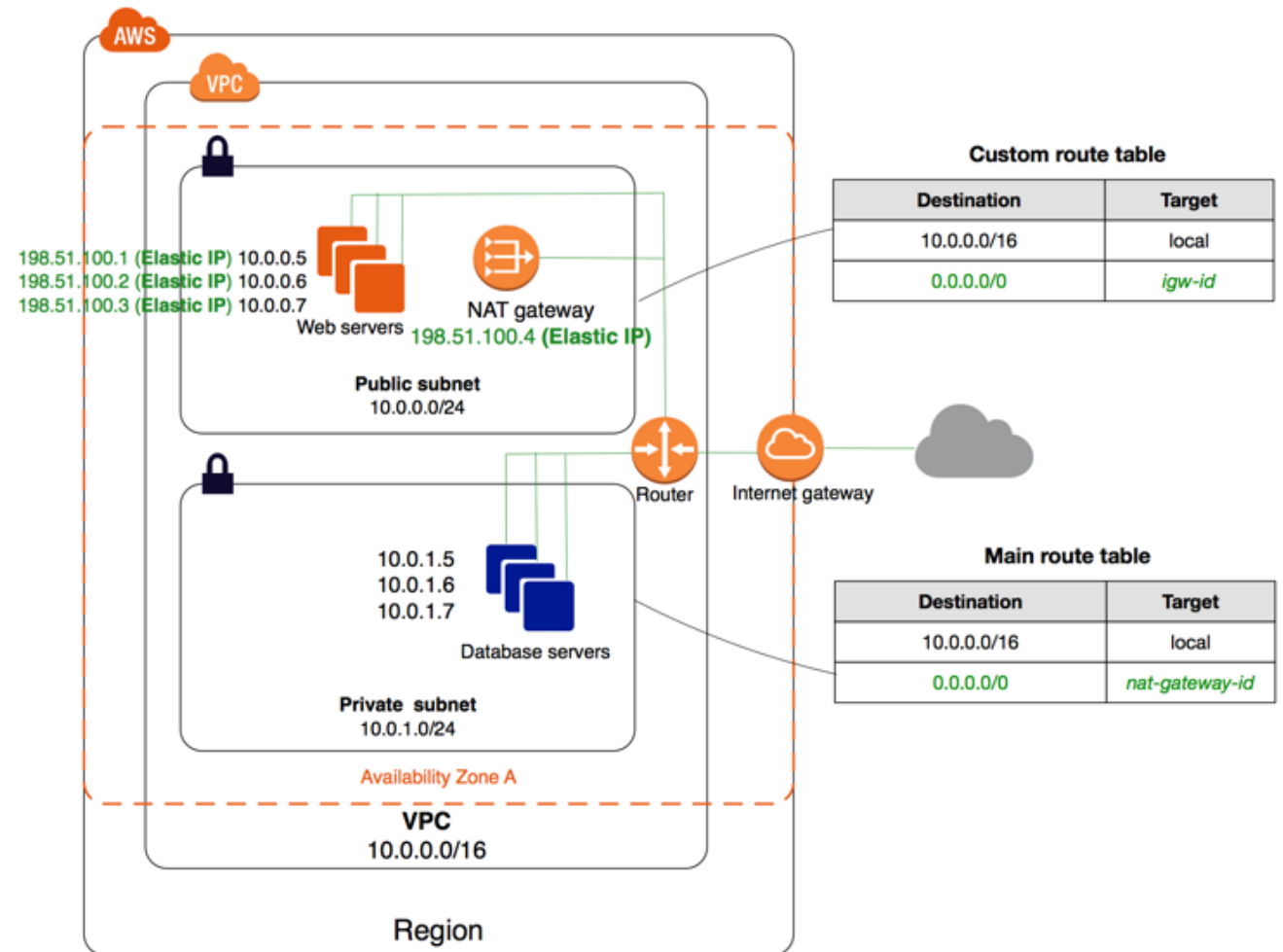
Regions

Availability Zones

Edge Locations

Virtual Private Cloud

- Define Virtual Networks in AWS
- Split into subnets
 - Subnets tied to an AZ
 - Typically use multiple AZs for HA
- Subnets can be public/private/VPN
 - Controlled by route table
- Internet Gateways provide Internet access
- NAT Gateways (or instances) used with private subnets



Security Groups & Network ACLs

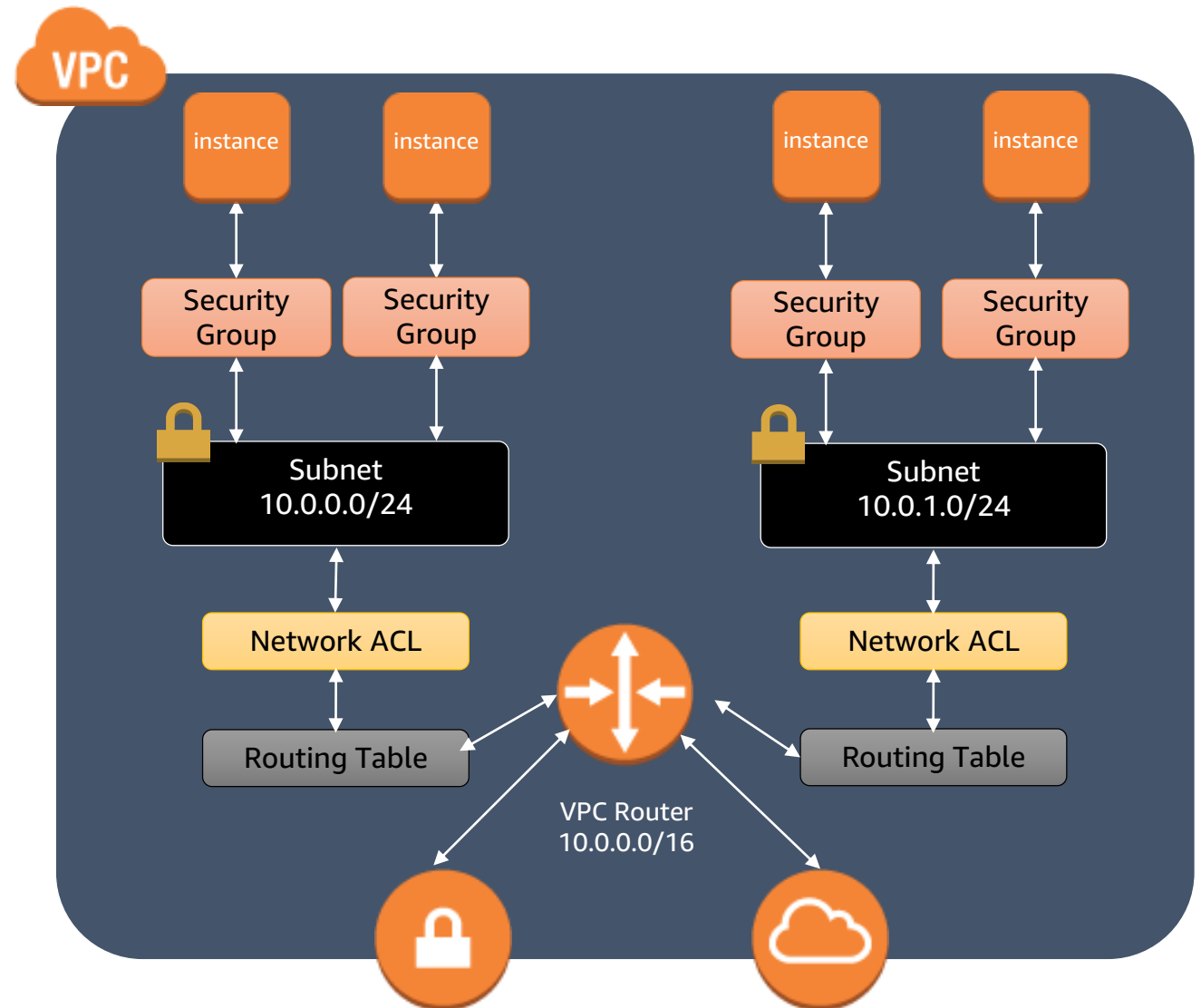
Security Groups are stateful firewalls at instance level

Lots of different types of resources that run in VPCs can use Security Groups

Network Access Control Lists (NACLs) are stateless firewall rules at subnet boundaries

Need to allow access in both directions

Can block IPs





EC2
Elastic Compute
Cloud

EC2

- Cloud based virtual machines – building block of IaaS
- EC2 portion of AWS Console includes related services, including:
 - Security Groups (firewall)
 - Autoscaling groups
 - Load Balancers
- Many different options available with different CPU/Memory/IO/GPU
<https://www.ec2instances.info>
- Linux (various) and Windows Server AMIs (Amazon Machine Images)
- X86_64 (Intel and AMD), and ARM

Instance Types

- Arranged by:
 - Family – General Purpose, Compute, Memory, Accelerated, Storage
 - Class – T, C, M...
 - Generation – C1, C2, C3, C4, C5
 - Size – c5.large (2 vCPU/4 GiB), c5.xlarge, ..., c5.24xlarge (96 vCPU, 192 GiB)
- Additional variations for AMD (T3a, M5a)
- And attached storage (M5d, M5ad)



Pricing

On-Demand

**Spot
Instances**

**Reserved
Instances**

**Dedicated
Hosts**

- Per-second billing (Amazon Linux and Ubuntu only)
- Per-hour billing (All other OSs)

- Per-hour billing

T2/T3 instances

- Low cost instances that have burstable performance
- CPU credits accumulate over time
- With Unlimited mode you will not be throttled, but charged extra if you run out of credits
- Not a good option if you're consistently using most of the CPU
- Smallest X86_64 instance types with as little as 0.5 GiB memory
 - t3.nano: \$0.0052 / hr
 - t3a.nano (AMD): \$0.0047 / hr
- Free Tier: t2.micro only

Amazon Machine Images

Initial Software for instances

- OS
- Initial state of any patches
- System or Application software

Types

- Community
(includes those provided by AWS)
- Marketplace
- Private

Amazon Linux 1

- Originally based on RHEL 5/6 or CentOS
- RPM/Yum packaging
- System V init
- Fair selection of packages
- Supported until June 2020

Amazon Linux 2

- Based on CentOS 7?
- RPM/Yum packaging
- systemd
- Limited packages, very small extras
- Supported until June 2023

Lab: Nginx on EC2

The goal:

Welcome to **nginx** on Amazon Linux!

This page is used to test the proper operation of the **nginx** HTTP server after it has been installed. If you can read this page, it means that the web server installed at this site is working properly.

Website Administrator

This is the default `index.html` page that is distributed with **nginx** on Amazon Linux. It is located in `/usr/share/nginx/html`.

You should now put your content in a location of your choice and edit the `root` configuration directive in the **nginx** configuration file `/etc/nginx/nginx.conf`.

[[Powered by nginx](#)]

Steps

- Sign in using details on sheet
- Create SSH Key Pair (EC2 -> Key Pairs)
- Launch EC2 instance
 - Select Amazon Linux 2
 - Use free-tier eligible t2.micro
 - Pay attention to Security Group (which ports do you want open?)
 - Either SSH to instance and install/start nginx
 - Or use script in UserData to install/start nginx at start-up
- Terminate EC2 instance
- Time: ~30 minutes

EC2 Block Storage Options

Ephemeral / Instance:

- Physically attached to host
- When used as boot device, cannot stop/start instance
- Some instance types use high performance NVMe SSDs
- Storage size typically tied to instance size (CPU/Memory)

Elastic Block Storage (EBS):

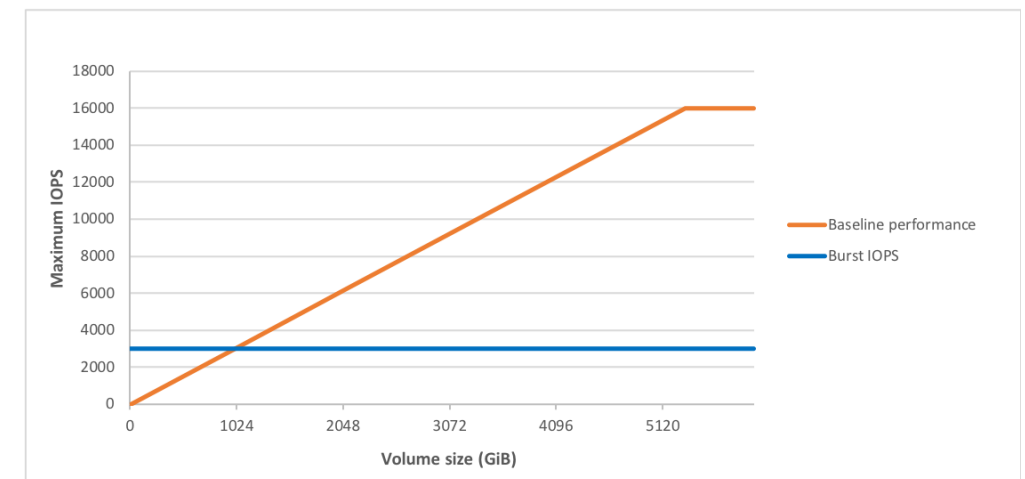
- Similar to Storage Area Network
- Attached to single EC2
- Can be detached and attached to another instance (same AZ)
- Snapshots can be transferred to other AZs and Region
- A variety of performance options
- Sized independently

EBS Performance

	Solid-State Drives (SSD)		Hard Disk Drives (HDD)	
	General Purpose	Provisioned IOPS	Throughput-Optimized	Cold
Volume size	1 GiB - 16 TiB	4 GiB - 16 TiB	500 GiB - 16 TiB	500 GiB - 16 TiB
Max IOPS/volume	16,000	64,000	500	250
Max throughput/volume	250 MiB/s	1,000 MiB/s	500 MiB/s	250 MiB/s

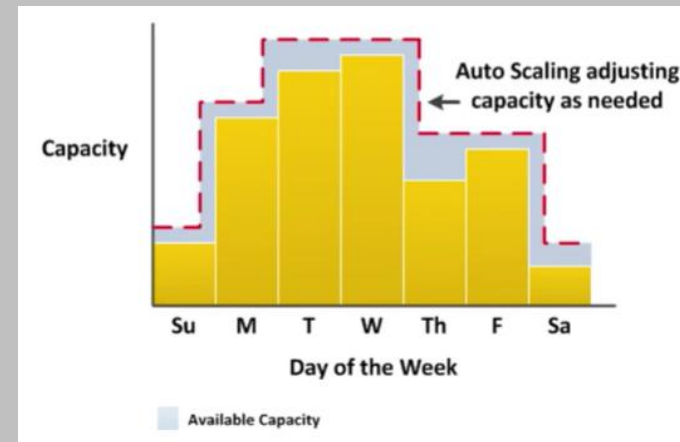
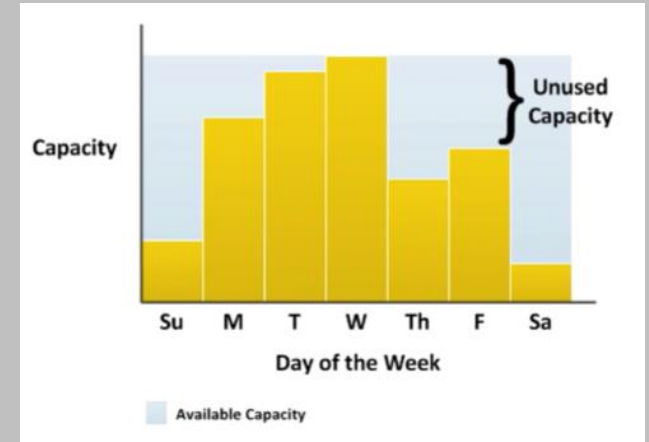
General Purpose SSD (gp2) IOPS scales based upon storage size:

- 3 IOPS/GiB
- Minimum of 100 IOPS
- Performance burst to 3000 IOPS
- Burst duration based upon credits (based upon size)
- Over 1000 GiB no bursting



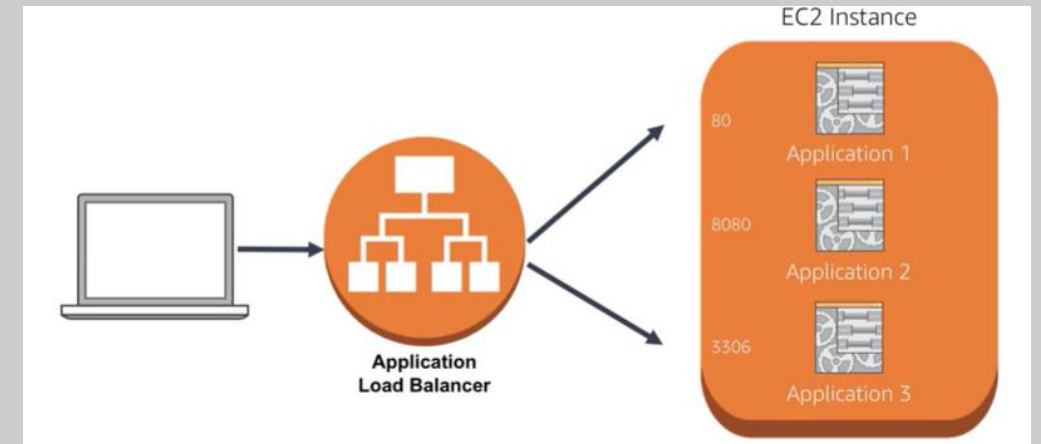
Autoscaling

- Use EC2 autoscaling to avoid unused capacity and save on cost
- Can be manual, scheduled or triggered based upon demand
- Target based autoscaling can scale up or down based upon the desired metric value (e.g. 75% CPU Utilization)
- Using small instances gives more scaling flexibility
- Build applications that can tolerate scaling – store state elsewhere



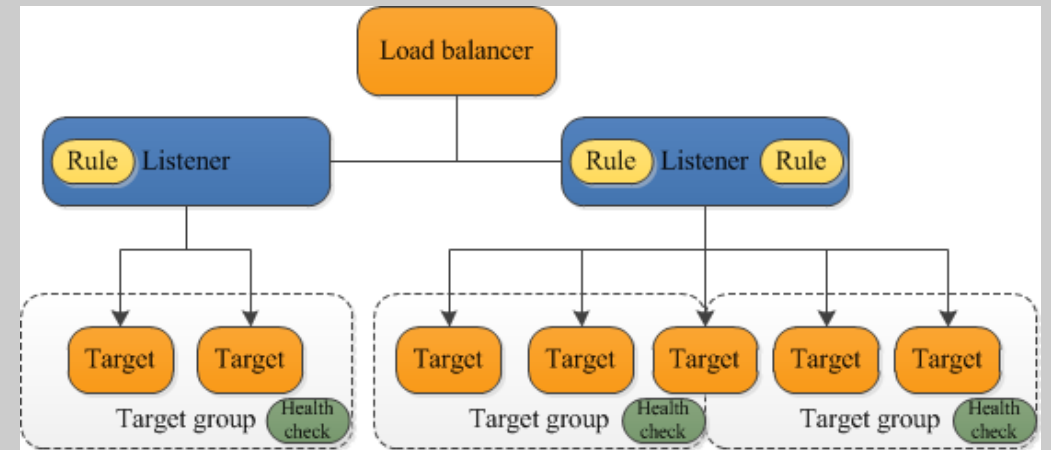
Load Balancing

- Distributes traffic to multiple targets
 - Copies of the same application
 - Different applications based upon host/path/port
 - EC2, ECS/EKS (Docker)
- Application Load Balancer (Layer 7 – HTTP/HTTPS)
- Network Load Balancer (Layer 4 – TCP)
- Classic ELB (Layer 4 or 7)
- TLS termination

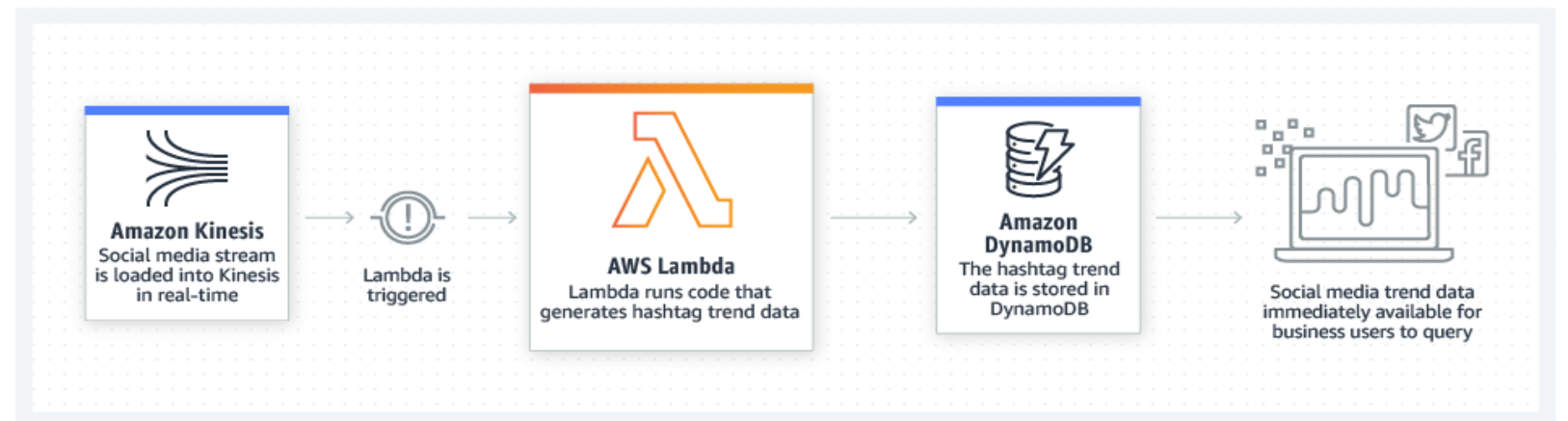
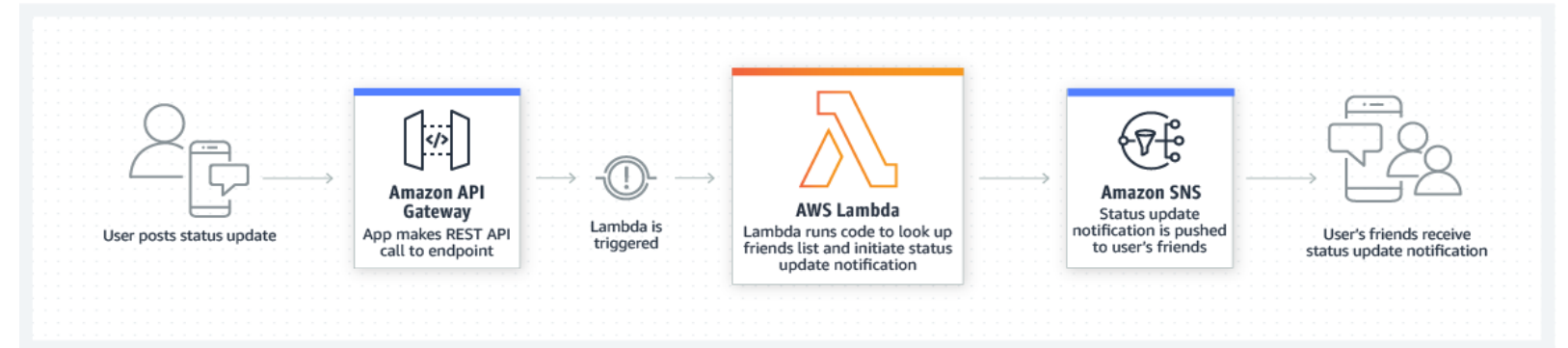
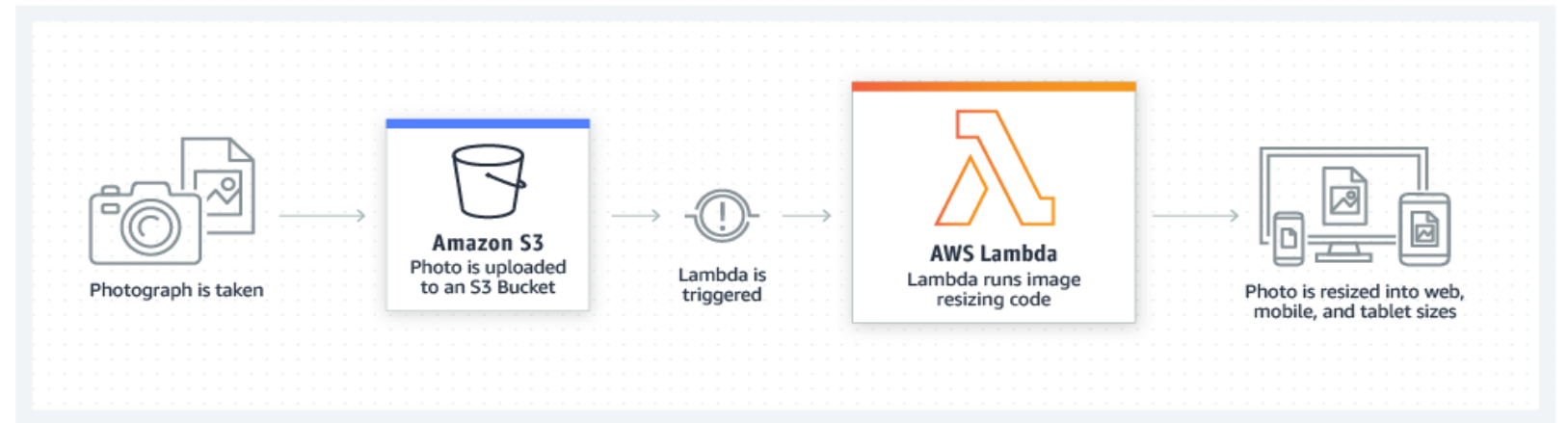


Parts of an ALB

- Load Balancer
- Listeners
 - HTTP
 - HTTPS
- Target Groups
 - Each Listener has a default
 - Health check
- Listener Rules
 - Direct traffic to Target Group based upon Host/Path/Header...
- Targets



AWS Lambda Serverless Compute



Lambda Usage

- Upload code in a variety of languages as zip file
- Lambda calls an entrypoint function with an event
 - Type of event varies based upon calling service
- Each instance handles one request at a time
- Automatic scaling
- Pay only for time running in 100ms increments
 - Amount depends upon allocated memory/CPU
- Can be entire application or glue between services

Storage

Block Storage options

Elastic Block Storage (EBS) and Instance Storage

- Attached to single EC2

Elastic File System

- Shared high performance block storage for Linux
- NFS v4
- Scales to Petabytes of storage
- Higher throughput than EBS (higher latency)

FSx for Windows File Server

FSx for Luster

- High performance for HPC, ML, Media

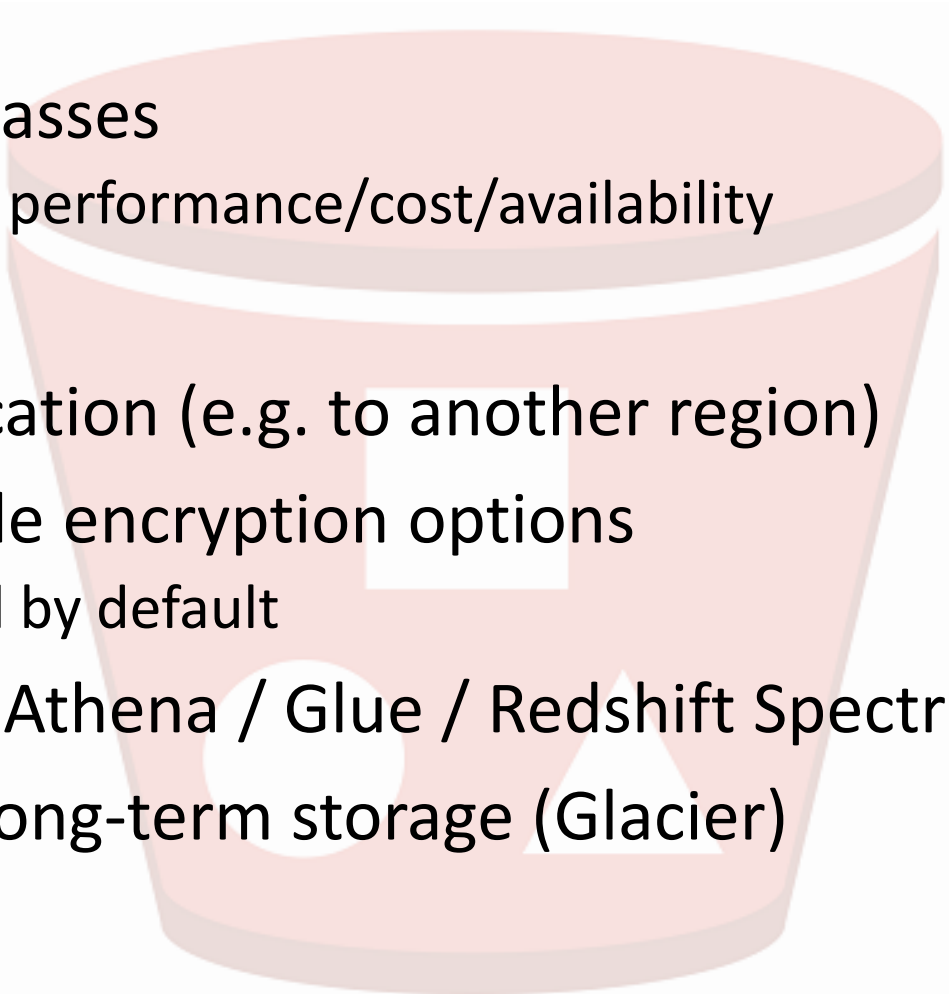
Object Storage – Simple Storage Service (S3)

- Write to S3 using APIs, not as a mounted file system
- Operate on entire objects (although you can retrieve ranges)
- Objects organized into Buckets and identified by Key
 - Key can look like a path, but there are no directories
- Objects from 0 bytes to 5 terabytes, no limit on size of bucket
- Replicated multiple times across region
- Buckets can be public and serve websites
 - Every object has a URL even if it's not public
 - Some companies have had issues with S3 buckets being public...

S3 Oddities (a selection)

- Every bucket must be uniquely named across all AWS accounts
 - Bucket names must be DNS valid
- Buckets shows as global in AWS Console, but belong to a region
- Multiple endpoints to access buckets
 - Path style:
 - `s3.amazonaws.com/bucketname/key` (always resolves to us-east-1 initially)
 - `s3.region.amazonaws.com/bucketname/key`
 - `s3.dualstack.region.amazonaws.com/bucketname/key` (IPv6 support)
 - Virtual Hosting:
 - `bucketname.s3.amazonaws.com/key` (resolves to correct region)
 - `bucketname.s3-region.amazonaws.com/key`

S3 Features

- Multiple storage classes
 - Different levels of performance/cost/availability
 - Object versioning
 - Cross bucket replication (e.g. to another region)
 - Multiple server-side encryption options
 - But not encrypted by default
 - Big data analytics (Athena / Glue / Redshift Spectrum)
 - Archive to offline long-term storage (Glacier)
- 

Databases

Managed vs. Unmanaged

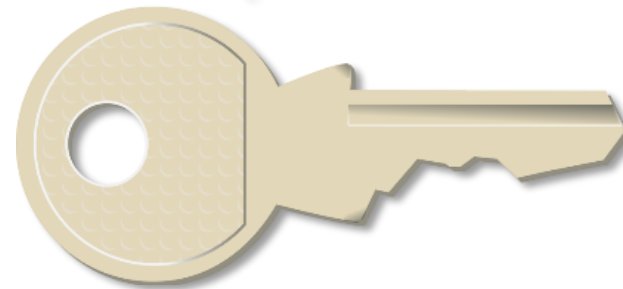
Unmanaged:

Scaling, fault tolerance, and availability are managed by you.



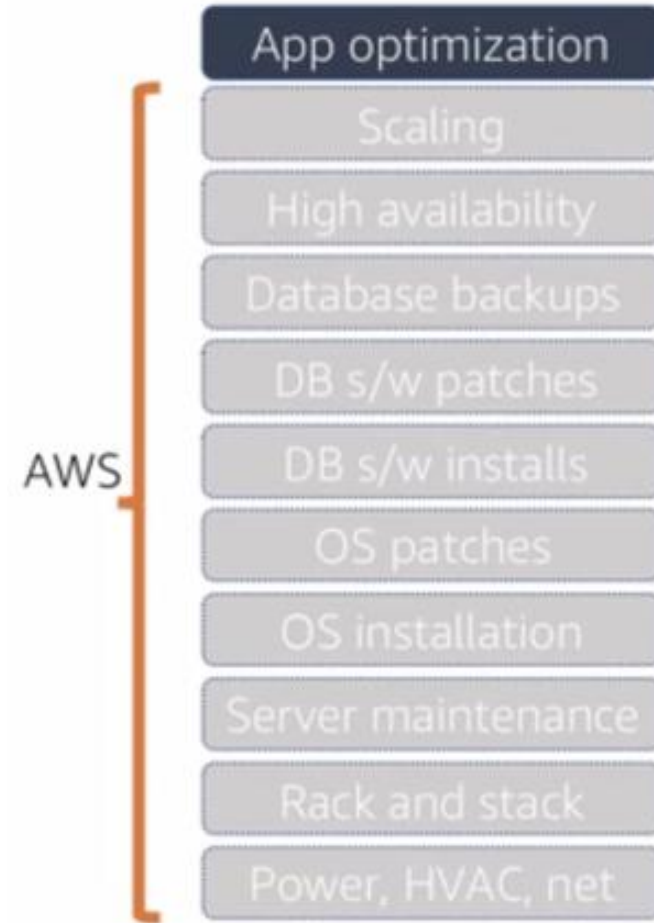
Managed:

Scaling, fault tolerance, and availability are typically built in to the service.



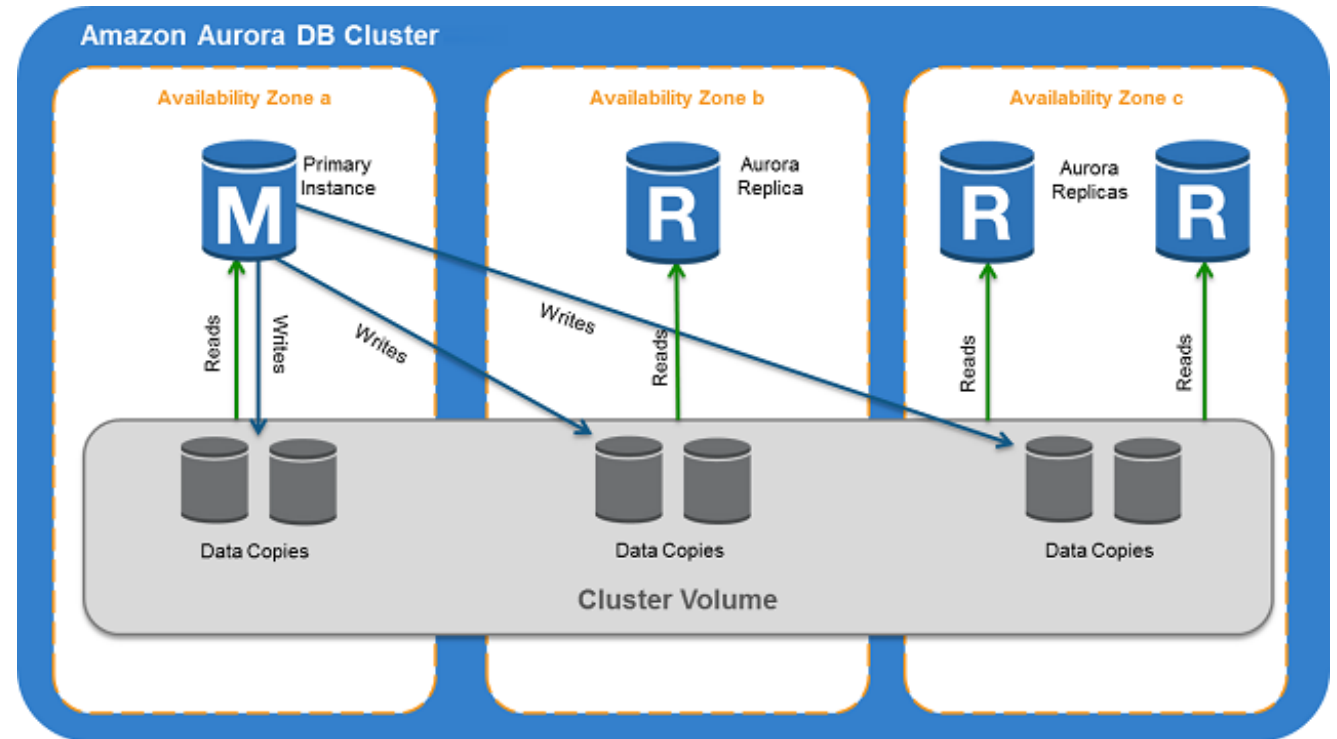
Relational Database Service (RDS)

- PostgreSQL
- MySQL
- MariaDB
- Oracle
- SQL Server
- Aurora (MySQL / Postgres)



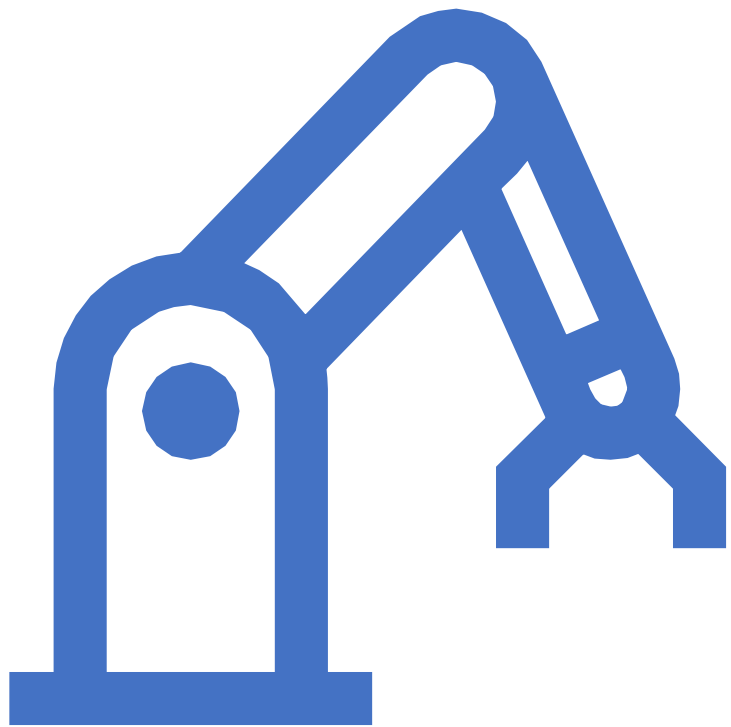
Amazon Aurora

- MySQL and PostgreSQL options
- Storage replicated across 3 AZs
- Read replicas in same region access shared storage
- Up to 64 TiB storage
- Performance improvements



NoSQL options

DynamoDB	Key-Value, high performance, serverless
DocumentDB	MongoDB compatible, fully managed
Neptune	Graph, Gremlin/SPARQL, fully managed
Timestream	Timeseries, still in preview from 2018
Quantum Ledger DB	Ledger, journal of changes, serverless
Elasticache	Memcached and Redis, fully managed
Elasticsearch	Search/Logs, Kibana UI



Infrastructure as Code

CloudFormation and Terraform

Infrastructure as Code

Infrastructure as Code (IaC) is the management of infrastructure (networks, virtual machines, load balancers, etc.) in a descriptive model, using the same versioning as used for source code.

Like the principle that the same source code generates the same binary, an IaC model generates the same environment every time it is applied.

IaC is a key DevOps practice.

Source: <https://docs.microsoft.com/en-us/azure/devops/learn/what-is-infrastructure-as-code>



Key Characteristics

- Repeatable - will consistently generate the same infrastructure
- Reusable - e.g. create multiple environments with similar infrastructure
- Reviewable - can perform reviews of infrastructure changes
- Versioned - stored in version control and this can be used to audit changes and revert if necessary
- Updatable - most IaC technologies track state and apply incremental changes when a change is made
- Not necessarily code - often a template of some form



What we'll be covering

- Terraform overview/demos
- Terraform lab
- Introduction to the infrastructure you'll be building
- CloudFormation overview/demos
- CloudFormation lab

<https://github.com/andrewdmay/infrastructure-as-code>



IaC tools for AWS

AWS CloudFormation



- AWS provided/supported IaC tools
- YAML or JSON templates
- Support for most AWS technologies
- State automatically managed in AWS service
- Limited logic*

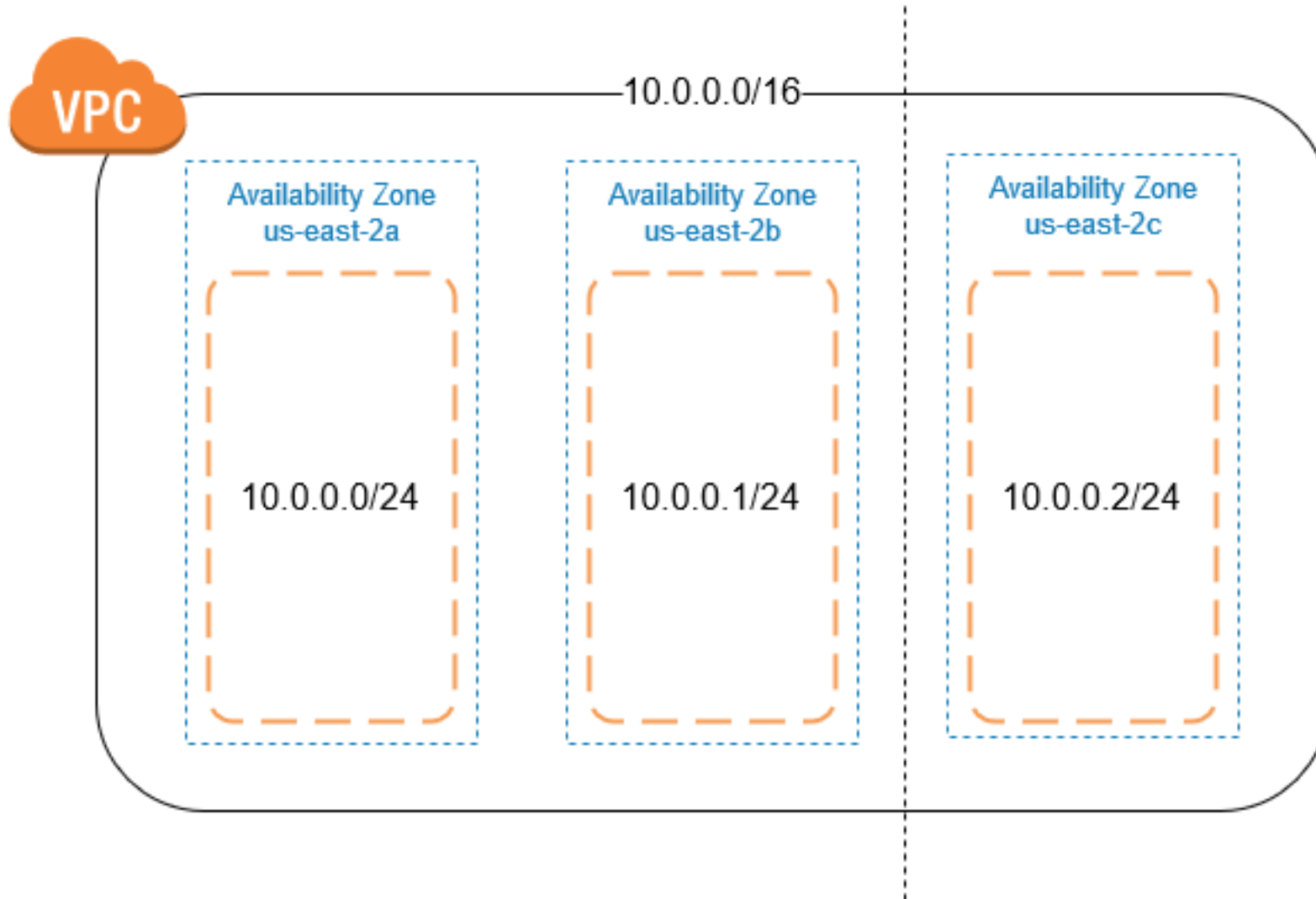
**see AWS CDK*

Terraform



- Created by Hashicorp
- Custom configuration language (or JSON)
- Providers support different Clouds (including AWS) and other services (e.g. a Kubernetes cluster)
- **Templates are NOT reusable across cloud platforms**
- Needs somewhere to store state
- Loops and other logic supported

Example Infrastructure



Virtual Private Cloud

- 2 or 3 availability zones
- Public subnet in each AZ

This is not a realistic VPC configuration, but it's a simple way to get started

Terraform

Terraform Providers

- ACME
- Akamai
- Alibaba Cloud
- Archive
- Arukas
- Avi Vantage
- Aviatrix
- AWS
- Azure
- Azure Active Directory
- Azure Stack
- Bitbucket
- GitLab
- Google Cloud Platform
- Grafana
- Gridscale
- Hedvig
- Helm
- Heroku
- Hetzner Cloud
- HTTP
- HuaweiCloud
- Icinga2
- Ignition
- OVH
- Packet
- PagerDuty
- Palo Alto Networks
- PostgreSQL
- PowerDNS
- ProfitBricks
- RabbitMQ
- Rancher
- Rancher2
- Random
- RightScale

Over 100 official providers listed, with more community providers available

Provider:

A plugin for Terraform that makes a collection of related resources available. A provider plugin is responsible for understanding API interactions with some kind of service and exposing resources based on that API.



Terraform Resources and Data Sources

Resource:

In Terraform's configuration language: A block that describes one or more infrastructure objects. Resources can be things like virtual networks, compute instances, or higher-level components such as DNS records.

In other Terraform-related contexts: An infrastructure object of a type that *could* be managed by Terraform.

Data Source:

A resource-like object that can be configured in Terraform's configuration language.

Unlike resources, data sources do not create or manage infrastructure. Instead, they return information about some kind of external object in the form of readable attributes. This allows a Terraform configuration to make use of information defined outside of Terraform, or defined by another separate Terraform configuration.

<https://www.terraform.io/docs/glossary.html>

Terraform State

It's necessary to store the state of the Terraform managed resources somewhere:

- Local (terraform.tfstate files)
- Remote - a variety of backends including S3, Azure Blob Storage, Postgres
- Terraform Enterprise (on-premise)
- Terraform Cloud (free for 5 users)



Terraform: Simple VPC

Terraform works with the contents of a directory, so we can't simply switch templates to go between 2 and 3 subnets as we did with CloudFormation.

Templates in `aws/terraform/simple-vpc` and `aws/terraform/parameterized-vpc`

- Two subnets - `vpc-2-subnets.tf` - similar to our CloudFormation example
- Parameterized - `vpc-parameterized` - support for any number of subnets



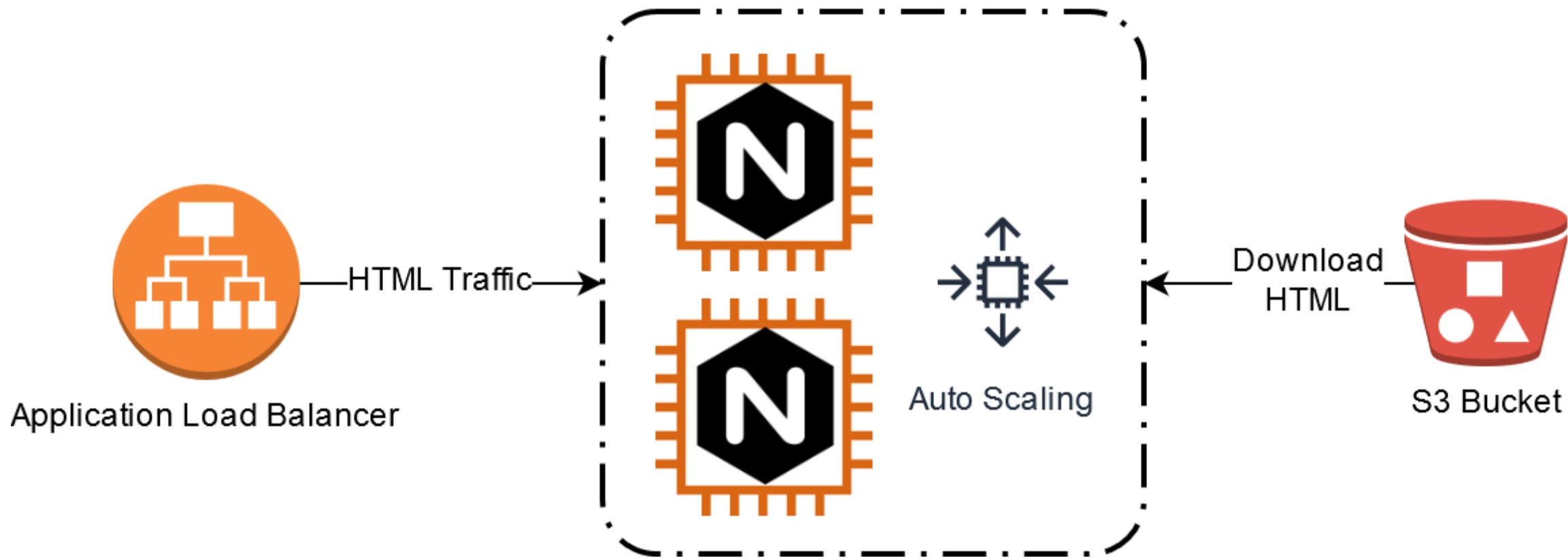
Terraform: Modules

- If you just have a single .tf file in a directory, that is the ROOT module
- You can split your configuration up into separate directories to have “nested” modules (recommended for these to be in the modules/ directory)
- Modules can also come from the Terraform registry:
<https://registry.terraform.io/>



Our Infrastructure

At a High Level



AWS Resources

IAM Role

(download from S3)

ALB Security Group

(HTTP from 0.0.0.0/0)

Application Load Balancer

Launch Template

(Template for EC2)

Instance Profile

(use role with EC2)

ASG Security Group

(HTTP from ALB)

Default Target Group

(EC2 instances)

Auto Scaling Group

(Group of EC2s)

ALB Listener

(HTTP)

<https://github.com/andrewdmay/infrastructure-as-code>

- Branches
 - iac-start: Template with parameters and outline
 - iac-iam: IAM Role/Profile
 - iac-sg: IAM, Security Groups
 - iac-alb: IAM, Security Groups and ALB/Target Group/Listener
 - iac-solutions: Complete template

README contains links to Terraform Documentation

Note: the CloudFormation templates are in these same branches, but we'll work on them later

CloudFormation

CloudFormation Basics

- A template is used to create a “Stack”
 - Many stacks can be created with the same template
- The template must define at least one “Resource”
- The template may have “Parameters”
- “Outputs” can be used to share key values from the generated resources
- Simple logic can be defined using “Conditions”
- Values that vary based upon a parameter (or intrinsic value) can be configured using a “Mapping”



CloudFormation: Simple VPC

- Templates in `aws/cloudformation/simple-vpc` directory
- Most basic (non-parameterized) templates:
 - `vpc-2-subnets.yml` - VPC with 2 subnets (works in us-east-2 (Ohio) only)
 - `vpc-3-subnets.yml` - same as above with 3 subnets
- Single template with support for 2 or 3 subnets using parameters, conditions, functions and outputs:
 - `vpc-parameterized.yml`



CloudFormation: CLI and SDK

In addition to using the Web Console to create stacks you can use the CLI or the SDK to interact with CloudFormation, and this is what you would typically use for CI/CD:

```
aws cloudformation create-stack --stack-name VPC \  
--template-body file:///vpc-parameterized.yml
```

Various utilities are built on top of the SDK for interacting with CloudFormation
(I wrote one in Python)



CloudFormation: Advanced features

- Nested stacks
- Serverless Application Model (SAM)
- CloudFormation macros
- StackSets
- Custom Resources



<https://github.com/andrewdmay/infrastructure-as-code>

- Branches (same as before)
 - iac-start: Template with parameters and outline
 - iac-iam: IAM Role/Profile
 - iac-sg: IAM, Security Groups
 - iac-alb: IAM, Security Groups and ALB/Target Group/Listener
 - iac-solutions: Complete template

README contains links to Terraform Documentation