Cloud Computing Foundation for Developers

IN1580

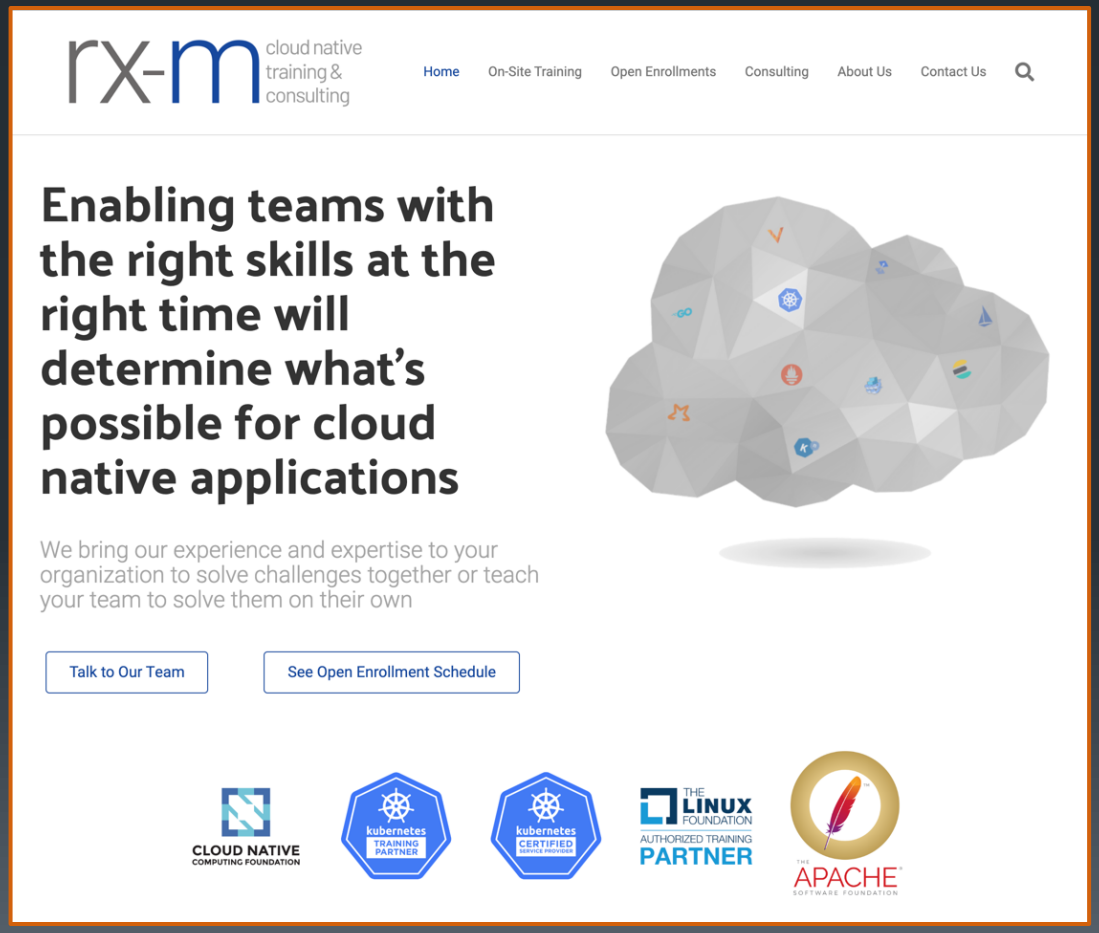
RX-M Cloud Native Consulting

Cloud Foundation

An in-depth introduction to cloud computing fundamentals for developers

RX-M Cloud Native Advisory, Consulting and Training

- Microservice Oriented
  - Microservices Foundation [3 Day]
  - Building Microservices on Kubernetes [3 Day]
  - Building Microservices on AWS [3 Day]
  - Building Microservices on GCP [3 Day]
  - Building Microservices on Azure [3 Day]
  - Building Microservices with Go [3 Day]
  - Building Microservices with Thrift [2 Day]
  - Building Microservices with gRPC [2 Day]
- Container Packaged
  - Docker Foundation [3 Day]
  - Docker Advanced [2 Day]
  - containerd [2 Day]
  - Cri-O [2 Day]
- Dynamically Managed
  - Docker Orchestration (Compose/Swarm) [2 Day]
  - Kubernetes Foundation [2 Day]
  - Kubernetes Advanced [2 Day]
  - Kubernetes for Developers [4 Day]
  - Kubernetes for Operators [5 Day]
  - Kubernetes CKA boot camp [5 Day]
  - Kubernetes CKAD boot camp [4 Day]
  - Kubernetes Day 2 operations [2 Day]
  - Securing Kubernetes [2 Day]
  - Monitoring Kubernetes with Prometheus [2 Day]
  - Mesos Foundation [2 Day]



Course Overview

The IaaS Foundation

1. Overview of cloud computing
2. Cloud services and service models
3. Cloud security and governance
4. AWS case study

Cloud Native Solutions

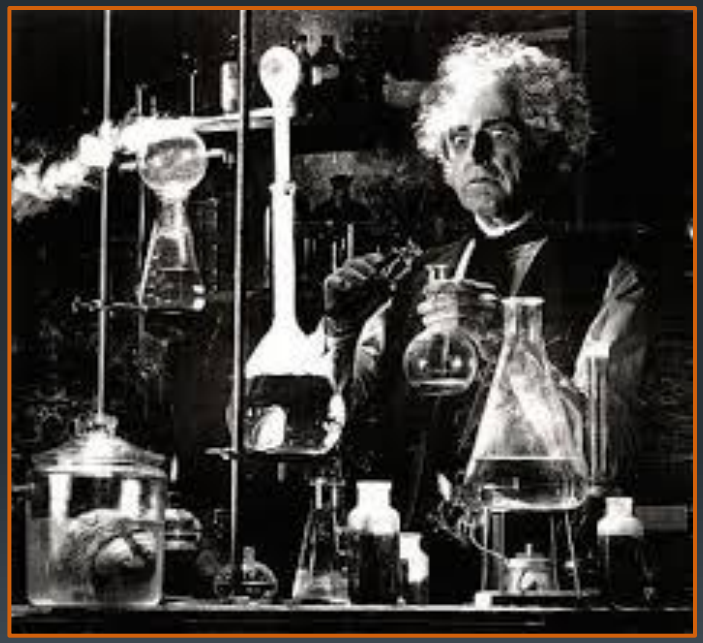
5. Cloud native systems overview
6. CaaS solutions
7. Cloud data stores
8. API gateways

Administrative Info

- Length: 2 Days
- Format: Lecture/Labs/Discussion
- Schedule: 8:30AM – 4:30PM
  15 minute break, AM & PM
  1 hour lunch at noon
  Lab time at the end of each AM and PM session
- Location: Fire exits, Restrooms, Security, other matters
- Attendees: Name/Role/Experience/Goals for the Course

Lecture and Lab

- Our Goals in this class are twofold:
  1. Familiarize you with general concepts and ecosystems
     - This is the primary purpose of the lecture/discussion sessions
     - The instructor will take you on a tour of the museum
       - Like a museum tour you should listen to and interact with the instructor
       - You will not have time to read the slides during the tour, like a museum the instructor will discuss and point out the highlights of the slides (exhibits) which will be waiting for you to read in depth later should you like to dig deeper
  2. Give you practical experience
     - This is the primary purpose of the labs
     - Classes rarely have time for complete real world projects so think of the labs as thought experiments
       - Like hands on exhibits at the museum



Day 1 – The IaaS Foundation

1. Overview of cloud computing
2. Cloud services and service models
3. Cloud security and governance
4. AWS case study

1: Overview of cloud computing

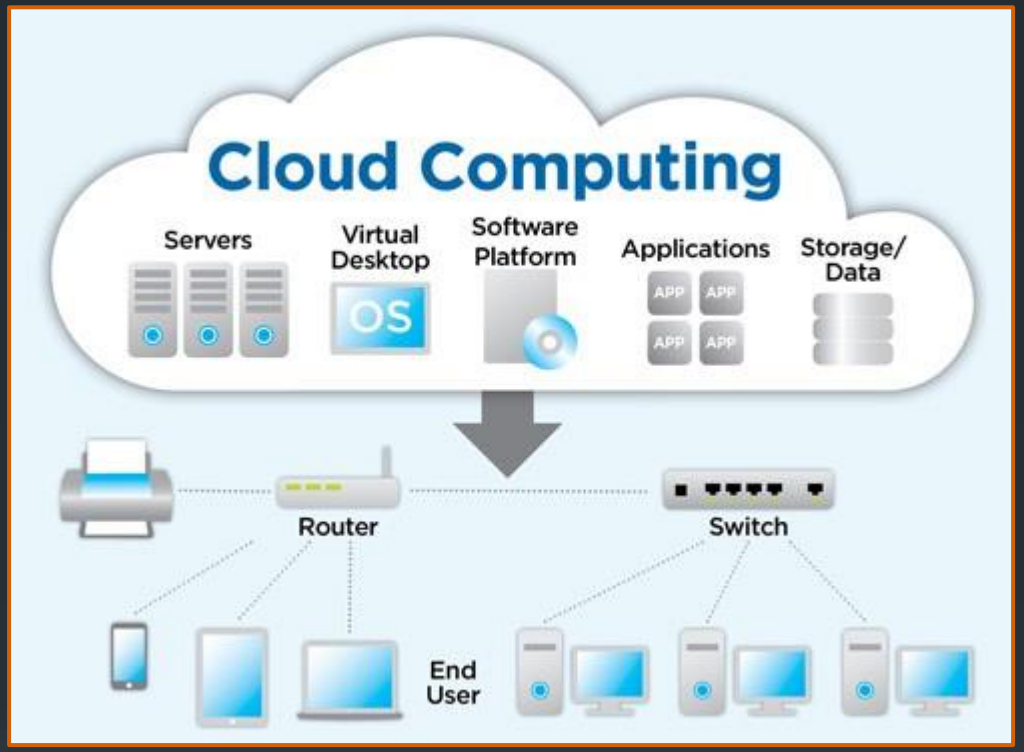
Objectives

- Define cloud computing
- Explain the motivation for cloud adoption
- Explore the economics of the cloud
- Learn when not to use cloud technology
- Introduce the market leading cloud providers and their features

What is Cloud Computing?

Cloud Computing

- Shared pools of resources
  - Servers
  - Networks
  - Storage
  - Applications
  - Services
- On demand
  - Can be rapidly provisioned with minimal management effort, often over the Internet
  - Allows companies to minimize up-front infrastructure costs
- Elasticity
  - Users can scale up as computing needs increase and scale down as demands decrease
- Economies of scale
  - Shared resources achieve economy of scale, similar to a utility
- Infrastructure abstraction
  - Enables organizations to focus on their core businesses instead of expending resources on computer infrastructure and maintenance
- Agility
  - Allows enterprises to get applications running faster, with improved manageability and more ability to rapidly adjust resources to meet unpredictable business demand

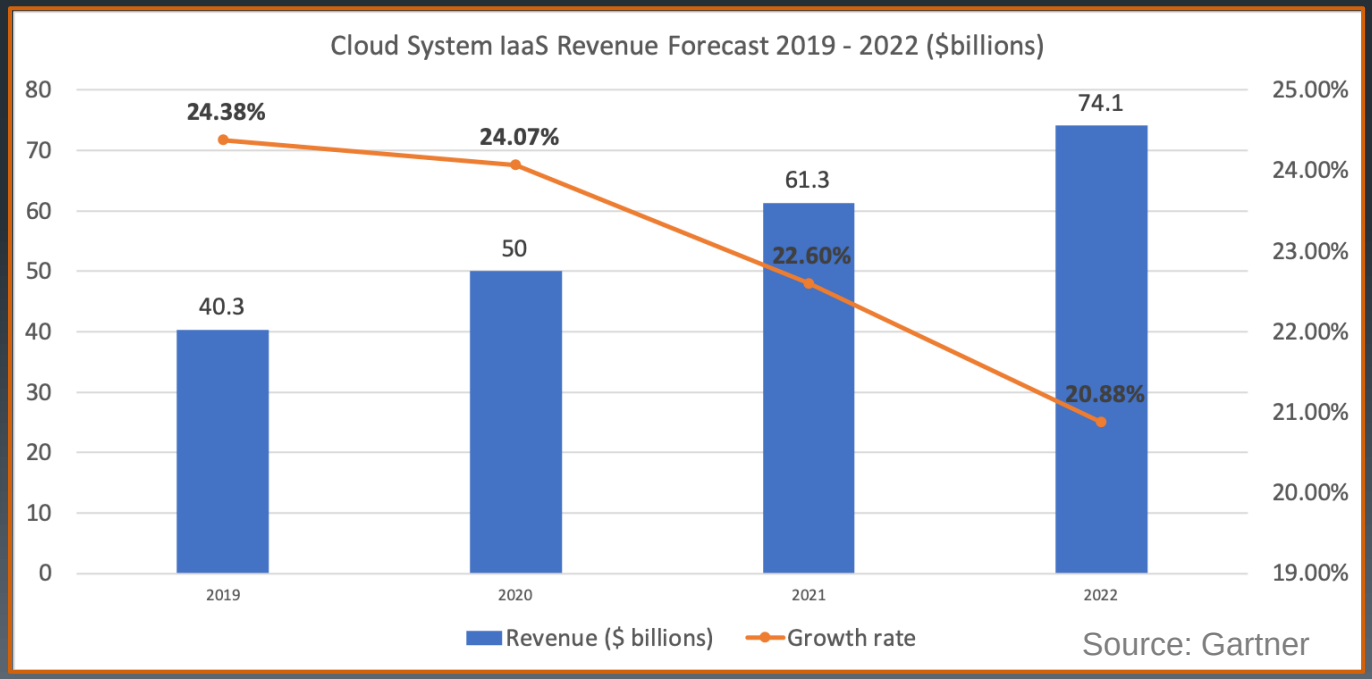


Cloud Enablers

- Technologies that enabled the Infrastructure as a Service cloud:
  - High-capacity networks
  - Low-cost computers
  - Low cost storage devices
  - Widespread adoption of hardware virtualization
  - Service-oriented architecture

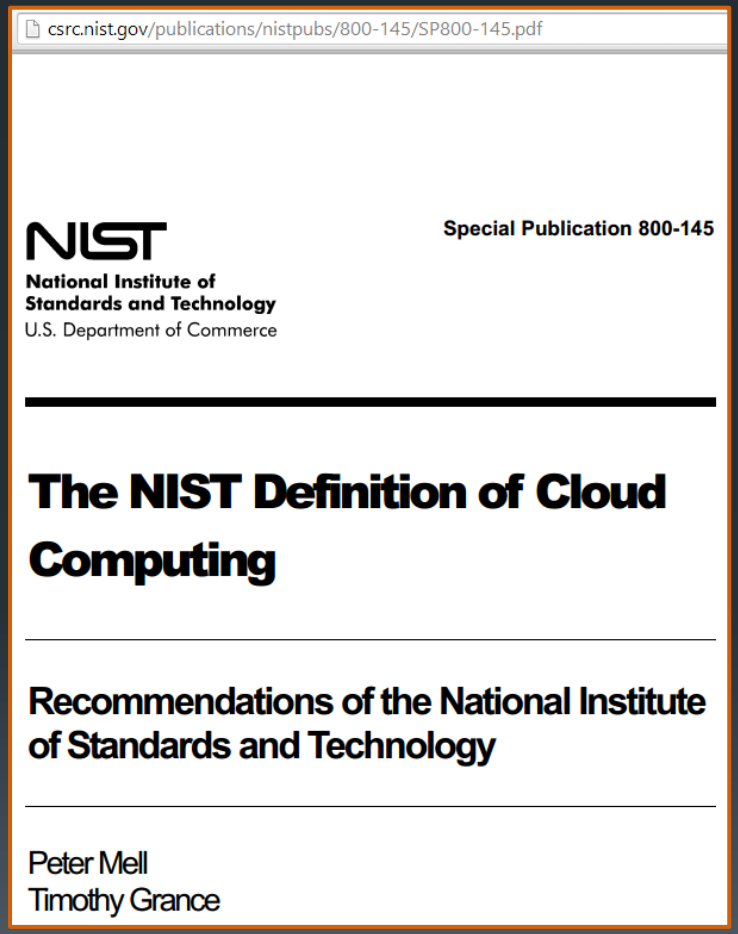
In 2019, the global cloud-enabling technology market was valued at $40.3 billion.

It is expected to reach $74.1 billion by 2022, a CAGR of 22.5%.



Essential Attributes of the Cloud

- The National Institute of Standards and Technology's definition of cloud computing identifies "five essential characteristics":
  - On-demand self-service
    - Users can provision servers, storage and networks without assistance from a service provider
  - Network access
    - Computing capabilities are available over the network by standardized mechanisms, such as mobile phones, tablets, and workstations
  - Resource pooling
    - Computing resources are pooled to serve multiple consumers, with different physical and virtual resources dynamically assigned and reassigned according to consumer demand
  - Elasticity
    - Provisioning is rapid and scales out or in based on need
  - Measured service
    - Metering features used (storage, processing, bandwidth, active accounts, etc.) enable monitoring, controlling, reporting, and billing



Why Use Cloud Computing?

Cloud Motivation

- Cost savings includes:
  - Cyclical replacement of hardware
  - Licensing
  - Maintenance costs
- Traditional server farms can be replaced with centrally hosted virtual servers managed by a fraction of the people
- According to Gartner, IT organizations invest 2/3 of budget into daily operations
- Moving to the cloud frees up 35 to 50 percent of operational/infrastructure resources (Wilcox, 2011)

TCO Example
On-prem Equivalent

Environment: Virtual					
# of VMs	CPU Cores	Memory (GB)	OS	VM Usage (%)	Optimize by
20	8	16	Linux	100%	CPU
5	8	16	Linux	100%	CPU

Storage (TB)		
SAN	NAS	Object
5	5	0

Bandwidth (Mbps)	
Pipe Size	Peak/Average Ratio
1,000	3

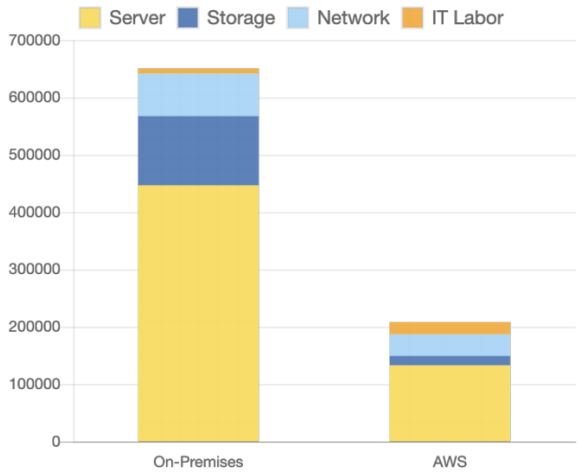
AWS Total Cost of Ownership (TCO) Calculator

On-Premises vs. AWS Summary

You could save **68%** a year by moving your infrastructure to AWS.

Your three year total savings would be $ 442,692

3 Years Cost Breakdown

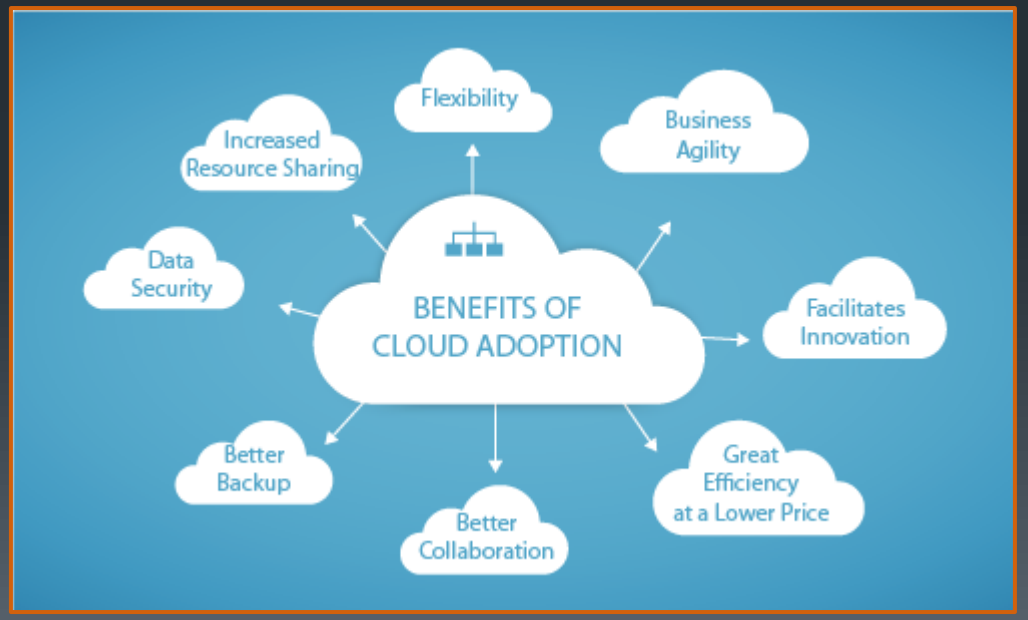


3 Yr. Total Cost of Ownership		
	On-Premises	AWS
Server	$ 446,860	$ 132,889
Storage	$ 120,810	$ 16,291
Network	$ 73,864	$ 37,525
IT-Labor	$ 9,113	$ 21,250
Total	$ 650,647	$ 207,955

AWS cost includes business level support

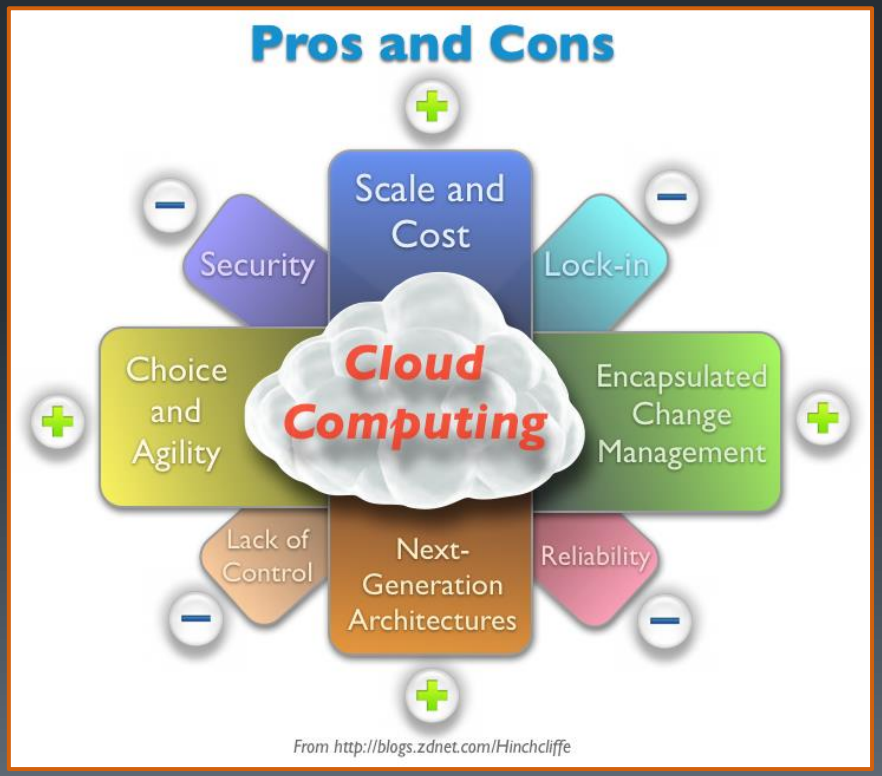
Enabling Innovation

- Public cloud systems offer essentially limitless scalability
- Platforms can be stood up and torn down for testing and experimentation cost effectively and repeatedly
- Resources are accessible from anywhere
- Google is probably better at running a data center than you are
  - Better security
  - Better efficiency
  - Better access
  - Better reliability
- Economies of scale and pay-as-you-go give way to cost savings



When Not to Use Cloud Technology

- Limited customization options
  - Cloud computing is cheaper because of economics of scale (requiring consistency)
  - A restaurant with a limited menu is cheaper than a Michelin Star restaurant with a unique menu every day
    - E.g. no one uses cloud systems for high frequency trading
- Legal and governance issues may constrain cloud use
  - A given cloud provider might not meet a company's legal needs
    - E.g. government certifications and/or data jurisdiction controls
- Management policies may not be in line with needs
  - Cloud providers decide on management policies independently of clients using the cloud
  - Cloud users are limited when it comes to monitoring workloads and infrastructure
    - No direct access to routing equipment of underlying networks
- Infrastructure limitations
  - Cloud vendors may not allow all hardware and/or network functions
    - E.g. no support for UDP multicast
- Noisy neighbors
  - Cloud workloads (particularly when using public clouds) may experience performance variability associated with other users' activity patterns consuming CPU and I/O capacity on shared hardware
- Privacy and confidentiality concerns
  - Data stored in public clouds must be managed with heightened security controls to avoid exposure or loss (encryption at rest and in flight etc.)



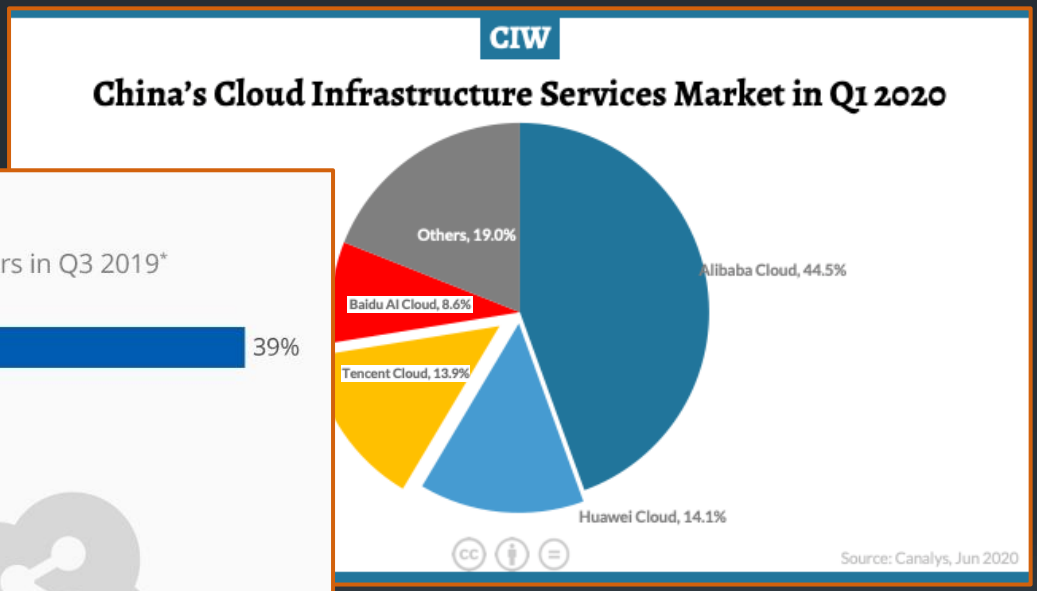
Cloud Providers & Platforms

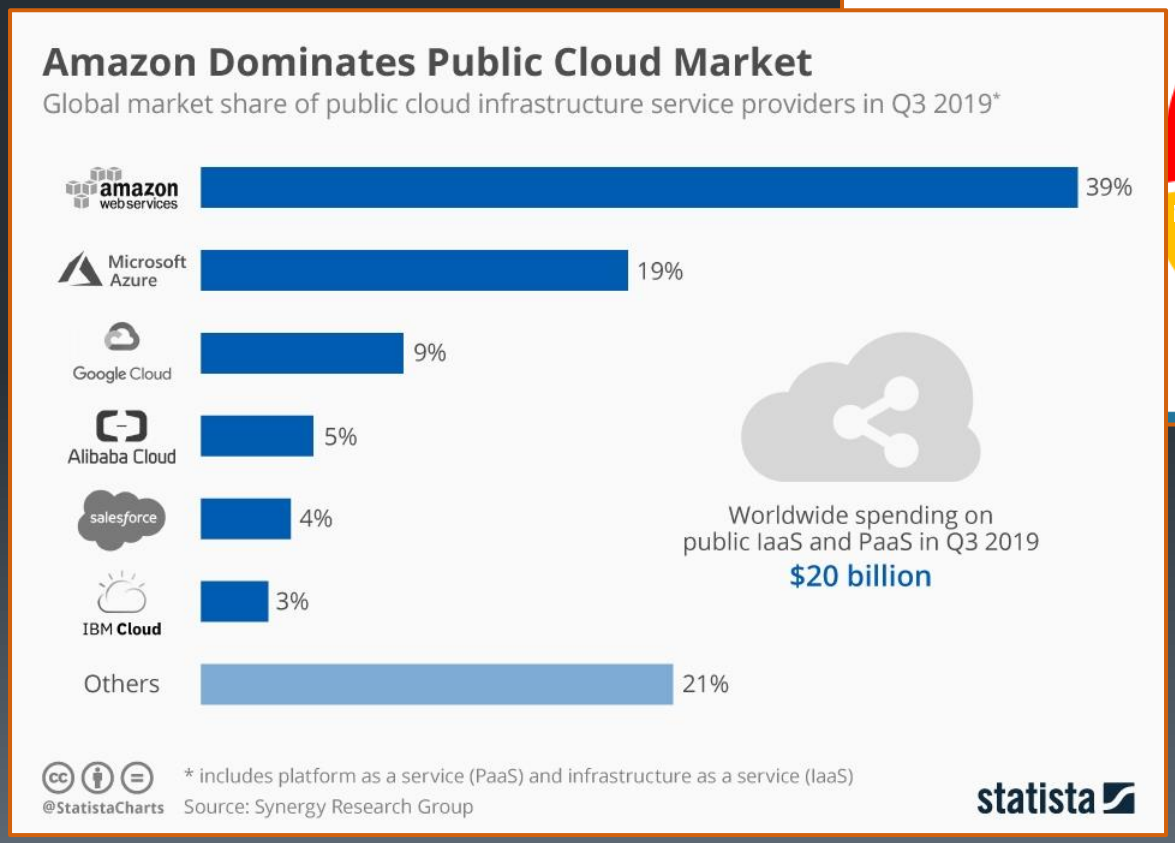
Cloud Platforms

- Amazon Web Services (AWS)
  - The front runner and arguably inventor of the public IaaS cloud
- Microsoft Azure
  - Best option for Windows based enterprises, also working to satisfy the open source Linux world
- Google Cloud Platform (GCP)
  - The original PaaS (App Engine), now betting heavily on Kubernetes and cloud native systems
- OpenStack
  - The lone open source cloud survivor, used in private data centers and powers IBM Cloud, Rackspace and other public clouds

Public Cloud Market Share

- Amazon AWS is the clear market leader
  - AWS posted over $40B annual run rate in May 2020
- Microsoft, IBM and Google trail but have made significant gains
- The Chinese market is also growing rapidly



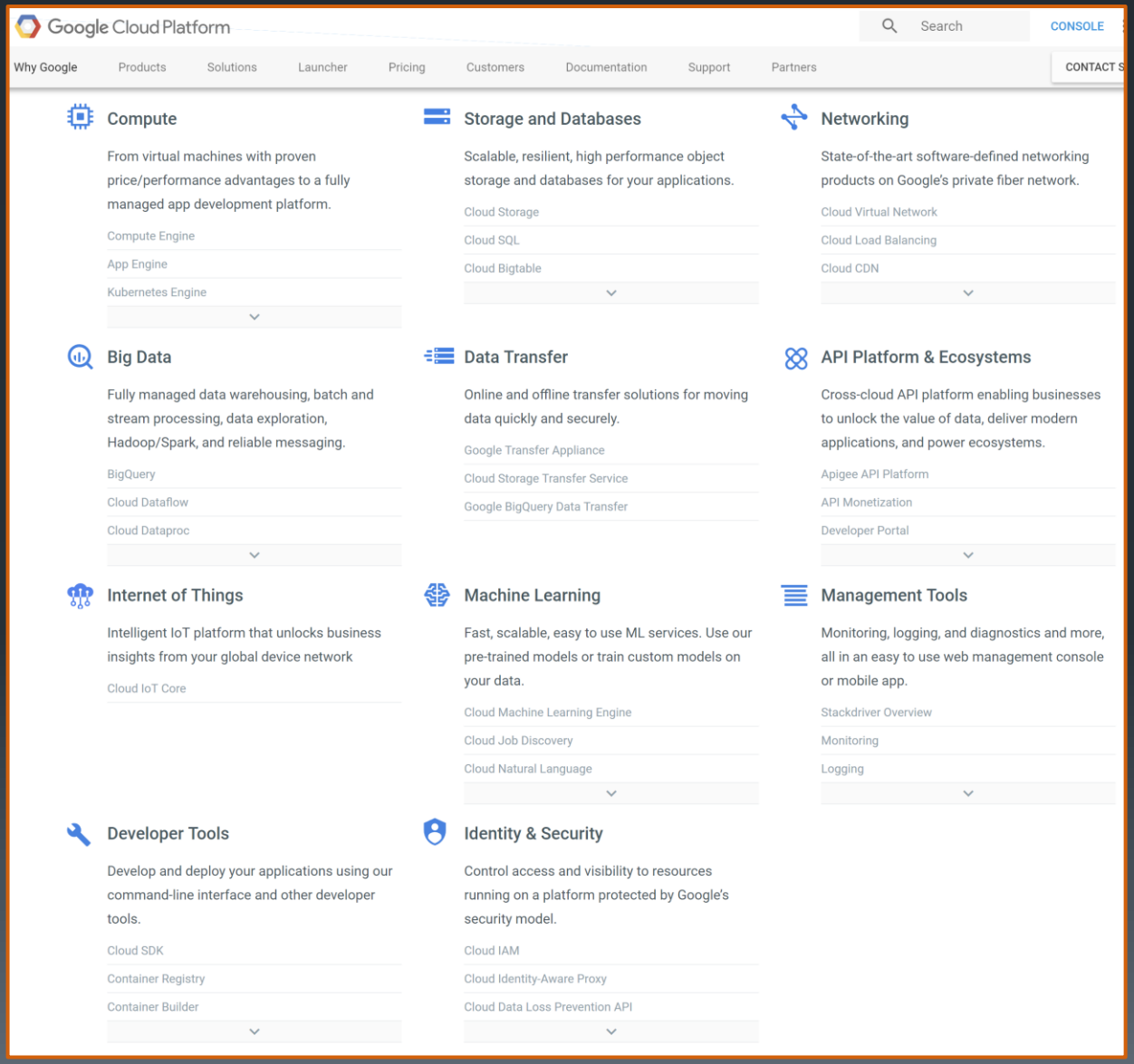


A case study

Google Cloud Platform

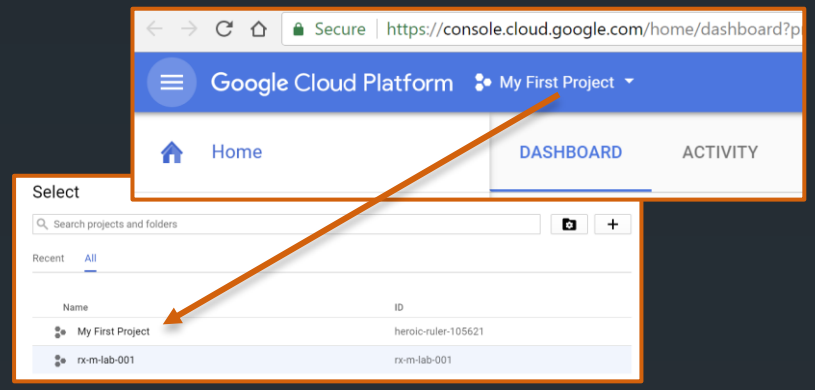
GCP Features

- A cloud platform built on Google's core infrastructure
- Powerful data analytics and machine learning offerings
- Security focused
- Committed to open source and industry leading price-performance



GCP: Organizations and Projects

- Organizations are the top level resource owners in Google Cloud
  - Companies and individuals create organizations in GCP which are then billed for usage
- Projects are created within an organization to define a logical division of the organization
  - Projects can be created for teams, departments, applications, etc.
  - All GCP resources live within a project
  - Projects form a security context for:
    - Adding and removing collaborators
    - Managing permissions for resources
  - To interact with Cloud Platform resources you must provide project information with each request (defaults can be configured)
  - Projects are identified by:
    - Project ID: the customized name you chose when you created the project
    - Project number: a number that's automatically generated by the server and assigned to your project
- GCP interfaces include:
  - Web Console
    - console.cloud.google.com
  - Command-line interface (CLI)
    - The gcloud tool provides command line cloud access
  - Application programming interface (API)
    - All GCP features are available through the GCP REST APIs




```
user@ubuntu:~$ gcloud config list
[compute]
region = us-west1
zone = us-west1-a
[core]
account = student@rx-m.io
disable_usage_reporting = True
project = rx-m-lab-001

Your active configuration is: [default]
user@ubuntu:~$
```

HTTP request

`POST https://www.googleapis.com/compute/v1/projects/{project}/zones/{zone}/instances`

The URL uses Google API HTTP annotation syntax.

Path parameters

Parameters	
`project`	`string` Project ID for this request.
`zone`	`string` The name of the zone for this request.

GCP: Regions and Zones

- Many GCP resources live in zones
  - Compute instances and persistent disks are referred to as zonal resources because they are located within a specific zone
  - Zones live within a single region
- A region is a specific geographical area where you can run your resources
  - Regions have names like: us-central1
  - Each region has one or more zones
  - The us-central1 region is in the Central United States and has zones:
    - us-central1-a
    - us-central1-b
    - us-central1-c
    - **us-central1-f**
- Many resources, like static external IP addresses, are regional resources
  - Regional resources can be used by any resources in that region, regardless of zone
    - To assign a static IP address to an instance, the IP and instance must be in the same region
  - Zonal resources can only be used by other resources in the same zone
    - To attach a disk to an instance, both resources must be in the same zone
- Other resources, such as images, are global resources
  - Global resources can be used by any resource in any location

https://cloud.google.com/compute/docs/regions-zones/

Working with GCP: gcloud

- gcloud is a tool that provides the primary command-line interface to Google Cloud Platform
- You can use this tool to perform almost any platform tasks either from the command-line or in scripts and other automations
- You can use gcloud to create and manage:
  - Google Compute Engine virtual machine instances and other resources
  - Google Cloud SQL instances
  - Google Kubernetes Engine clusters
  - Google Cloud Dataproc clusters and jobs
  - Google Cloud DNS managed zones and record sets
  - Google Cloud Deployment Manager deployments
  - Google App Engine applications
- gcloud is a part of the Google Cloud SDK
  - Install and initialize the GCP SDK to gain access to gcloud
- The SDK installs those gcloud commands that are at the General Availability and Preview levels only
  - Additional functionality available in SDK components named *alpha* and *beta*
    - Alpha and beta provide access to parts of the Cloud Platform at pre-release levels
- gcloud releases have the same version number as the SDK

```
user@ubuntu:~$ gcloud compute instances create vm-wra --image-project=centos-cloud
--image=centos-7-v20171213 --machine-type=n1-highcpu-2
Created [https://www.googleapis.com/compute/v1/projects/rx-m-lab-001/zones/us-west1
-a/instances/vm-wra].
NAME    ZONE        MACHINE_TYPE  PREEMPTIBLE  INTERNAL_IP  EXTERNAL_IP    STATUS
vm-wra  us-west1-a  n1-highcpu-2               10.138.0.2   35.197.85.116  RUNNING
user@ubuntu:~$ gcloud compute instances describe vm-wra
canIpForward: false
cpuPlatform: Intel Broadwell
creationTimestamp: '2018-01-03T12:52:28.671-08:00'
deletionProtection: false
disks:
- autoDelete: true
  boot: true
  deviceName: persistent-disk-0
  index: 0
  interface: SCSI
  kind: compute#attachedDisk
  licenses:
  - https://www.googleapis.com/compute/v1/projects/centos-cloud/global/licenses/cen
tos-7
  mode: READ_WRITE
  source: https://www.googleapis.com/compute/v1/projects/rx-m-lab-001/zones/us-west
1-a/disks/vm-wra
  type: PERSISTENT
id: '5509824327188544611'
kind: compute#instance
```

Summary

- The term *cloud* has many definitions but most accept the following attributes as key to any cloud:
  - On-demand self-service
  - Network accessible
  - Resource pooling
  - Elasticity
  - Measured service
- Cloud systems offer customers many benefits
  - Cost reductions
  - Faster time to innovation
  - Low risk experimentation
- Google Cloud Platform (GCP) is Google's public cloud offering based on the Google infrastructure
  - A representative major public cloud
  - Offering Web, CLI and API based access
  - Resources are organized into Projects within billing Organizations
  - Resources are located in zones within regions

Lab 1

- gcloud Orientation and Compute Engine
  - Use gcloud command line tools to create a simple compute environment

2: Cloud services and service models

Objectives

- Describe and contrast cloud models:
  - Infrastructure as a Service (IaaS)
  - Platform as a Service (PaaS)
  - Containers as a Service (CaaS)
  - Serverless and Functions as a Service (FaaS)
  - Software as a Service (SaaS)

Cloud Services and Types

IaaS

Infrastructure as a Service

Provides infrastructure such as computer instances, network connections, and storage so users can run any software or operating system

- IBM Cloud, OpenStack® cloud, Amazon AWS, Rackspace, Microsoft Azure, Google Cloud Platform

PaaS

Platform as a Service

Allows the consumer to deploy applications through a programming language or tools supported by the cloud platform provider

- Google App Engine, Engine Yard, Heroku, OpenShift

CaaS

Containers as a Service

Allows the consumer to deploy containers directly to the cloud

- Google GKE, AWS ECS and EKS, Azure Kubernetes Service, IBM Kubernetes Service

FaaS

Functions as a Service

Allows users to execute a single purpose program that removes the complexity that is often associated with servers and software stacks.

- AWS Lambda, Google Cloud Functions, IBM OpenWhisk

SaaS

Software as a Service

Allows the consumer to use the software in a cloud environment

- Salesforce CRM, Gmail, Basecamp Project Management

Governance/ Deployment Models

Private cloud
Operated for a single organization

Public cloud

Publically available infrastructure owned by a cloud services company

Hybrid cloud

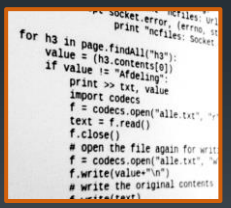
- Combination of the private and public clouds
- Private cloud acquires services of a public cloud when needed

Cloud Application Evolution

- Application architectures and cloud architectures have evolved hand in hand
  - The only reason for the cloud to be, is to run applications
  - More and more applications are targeting cloud systems

- Eras and architectures:

  - IaaS -- Three tier LAMP (Linux, Apache, MySQL, PHP), VM push
  - PaaS -- SOA, Ruby on Rails et al, Web 2.0, code push
  - CaaS -- Microservices, container push
  - FaaS -- Functions & microservices, code and/or container push



Digital Platform Evolution

- The platform underlying rapidly growing Internet applications has changed significantly over the years
  - .com era – bare metal servers
  - Web 2.0 – IaaS
  - APIs – PaaS
  - Microservices – Cloud Native

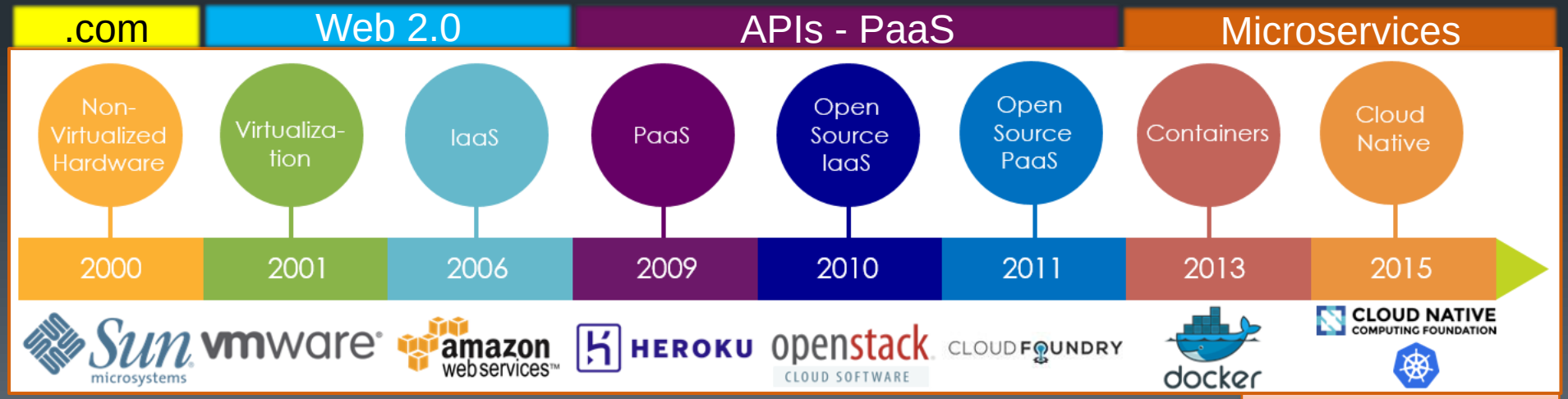
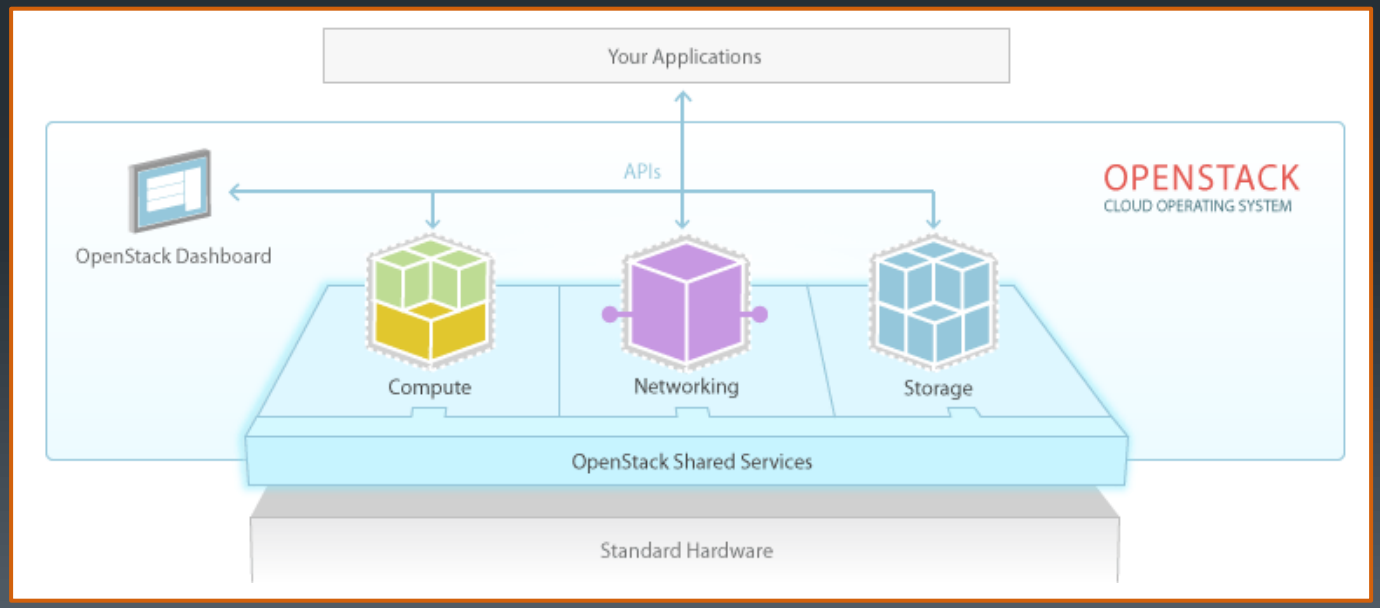


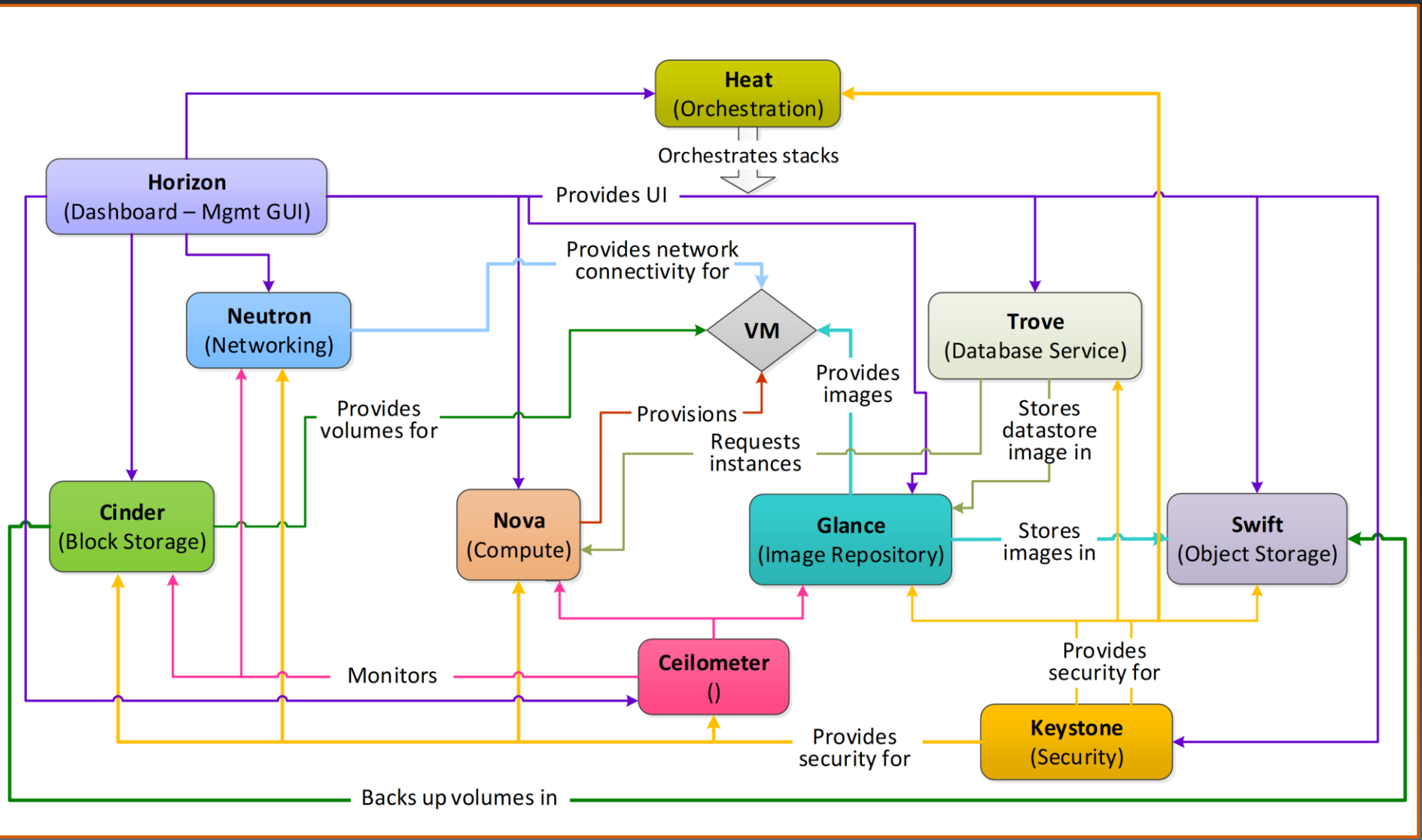
Image courtesy CNCF

Infrastructure as a Service

- Clouds are collections of technology resources organized such that users can simply and dynamically allocate needed resources to perform IT tasks
- This typically involves allocating
  - Compute
  - Network
  - Storage

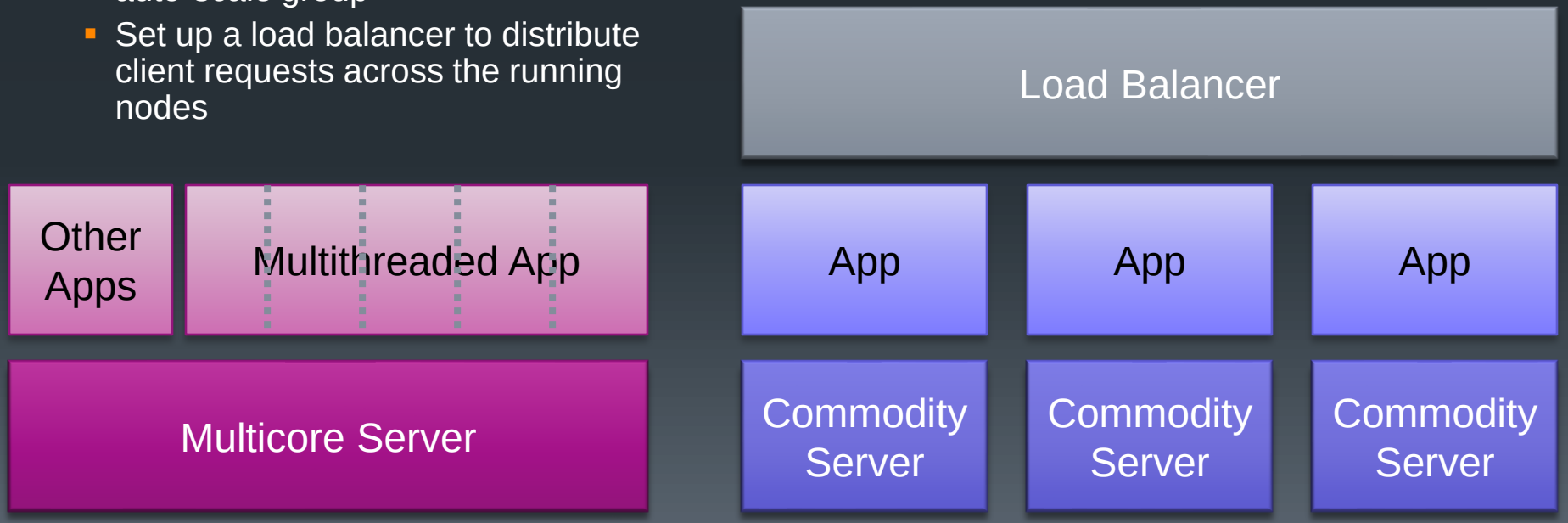


OpenStack IaaS Example



Building for the IaaS Cloud

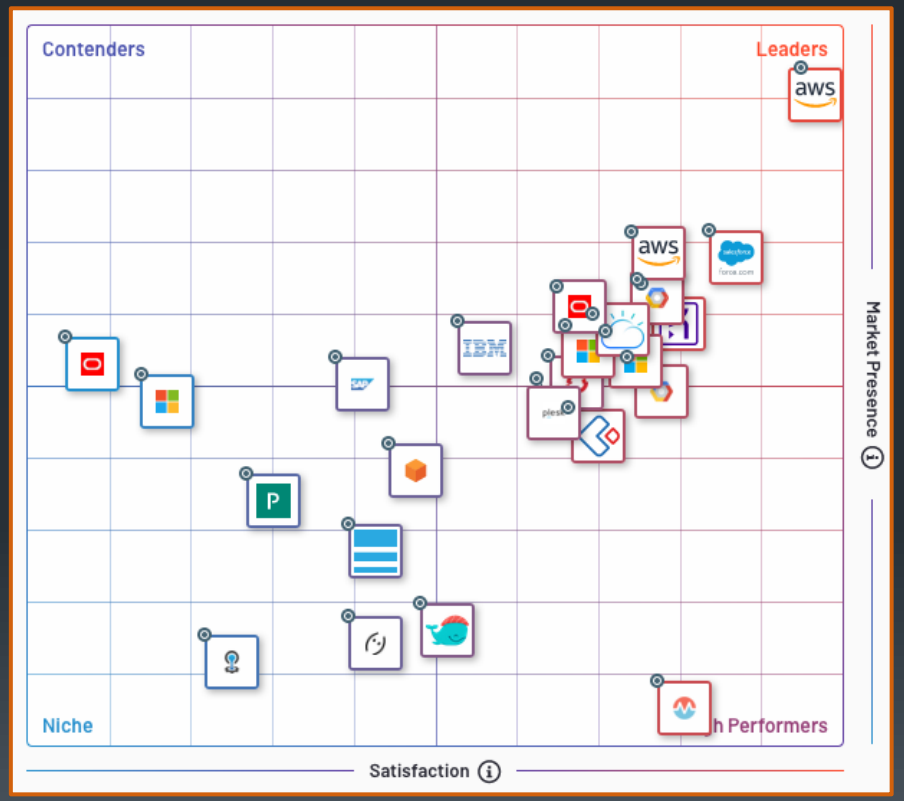
- Approaches to distributed application development vary in and out of the cloud
- Traditional distributed applications often involve sophisticated multithreaded servers
- The cloud approach is to create many copies of small services
- Cloud developers typically start by looking for the simplest possible way to build a scalable solution
- A common last generation approach
  - Create a “virtual machine” that wraps an application
  - Add this machine to a cloud based auto-scale group
  - Set up a load balancer to distribute client requests across the running nodes



Platform as a Service

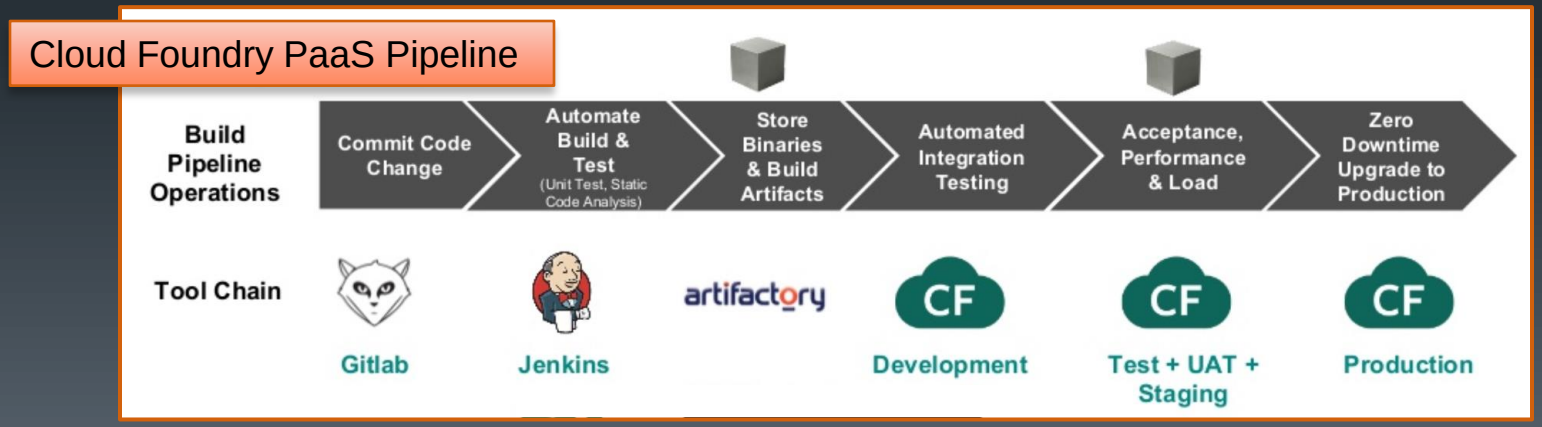
- Platform as a Service (PaaS)
  - A cloud computing platform allowing developers to build and deploy Internet based applications
  - PaaS services (Web Apps) are hosted in the cloud and accessed by users remotely
- Contrasts:
  - IaaS applications are established by uploading VMs into an IaaS cloud
    - *All storage, messaging and web server services are created and maintained as part of the application*
  - PaaS applications are established by pushing code to the cloud as source code control commits
    - *All storage, messaging and web server services are supplied and maintained by the platform*
- Influential PaaS systems:
  - Heroku - one of the first and most successful PaaS systems (launched 2007) acquired by Salesforce in 2010
  - Engine Yard - an early open source PaaS (launched 2006) acquired by Crossover in 2017 (shuttered)
  - Google App Engine (launched 2008)
  - Amazon Elastic Beanstalk (launched 2011)
  - Cloud Foundry - open source open governance Heroku like system targeting Java development (created by VMware and Pivotal)
  - OpenShift - Red Hat backed PaaS solution

Fall 2019



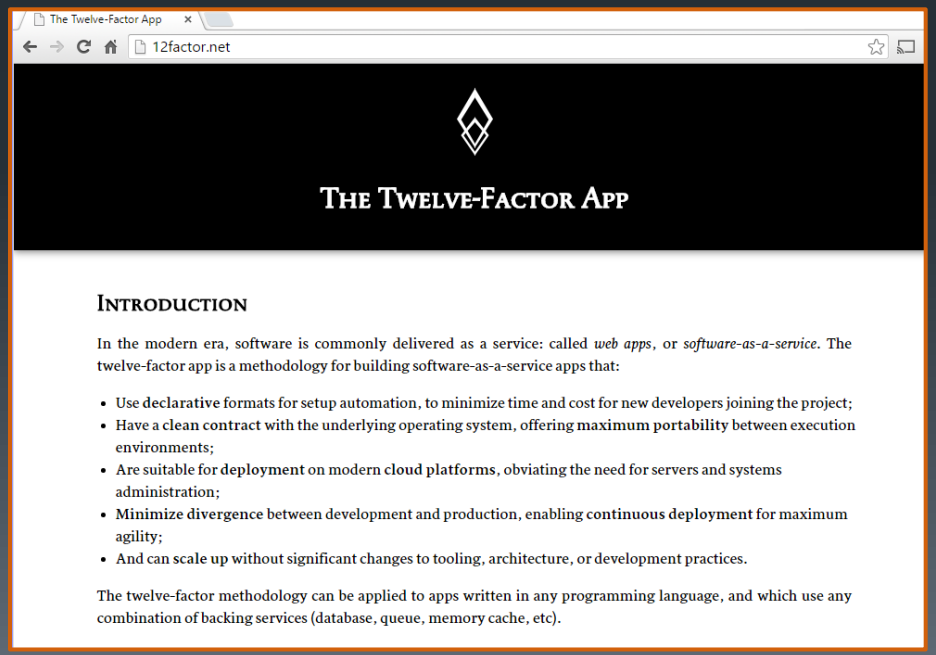
PaaS Attributes

- Platform as a Service
  - Push code
  - PaaS supports only applications fitting one of its buildpack models
    - More recently PaaS systems allow custom buildpacks to be created and even more recently PaaS systems provide basic container push support
  - The PaaS builds, tests, packages and deploys
  - The PaaS platform containerizes the build artifacts
    - Perhaps Java Archives (JARs), a custom format, Docker Images or other



Building for PaaS: 12 Factor Apps

- The Heroku PaaS was a crucible for cloud application experimentation
- Lessons learned and best practices for simple and reliable application deployment on Heroku were captured and published as the 12 factors


The Twelve-Factor App

12factor.net

THE TWELVE-FACTOR APP

INTRODUCTION

In the modern era, software is commonly delivered as a service: called *web apps*, or *software-as-a-service*. The twelve-factor app is a methodology for building software-as-a-service apps that:

- Use **declarative** formats for setup automation, to minimize time and cost for new developers joining the project;
- Have a **clean contract** with the underlying operating system, offering **maximum portability** between execution environments;
- Are suitable for **deployment** on modern **cloud platforms**, obviating the need for servers and systems administration;
- **Minimize divergence** between development and production, enabling **continuous deployment** for maximum agility;
- And can **scale up** without significant changes to tooling, architecture, or development practices.

The twelve-factor methodology can be applied to apps written in any programming language, and which use any combination of backing services (database, queue, memory cache, etc).

THE TWELVE FACTORS

I. Codebase
One codebase tracked in revision control, many deploys

II. Dependencies
Explicitly declare and isolate dependencies

III. Config
Store config in the environment

IV. Backing Services
Treat backing services as attached resources

V. Build, release, run
Strictly separate build and run stages

VI. Processes
Execute the app as one or more stateless processes

VII. Port binding
Export services via port binding

VIII. Concurrency
Scale out via the process model

IX. Disposability
Maximize robustness with fast startup and graceful shutdown

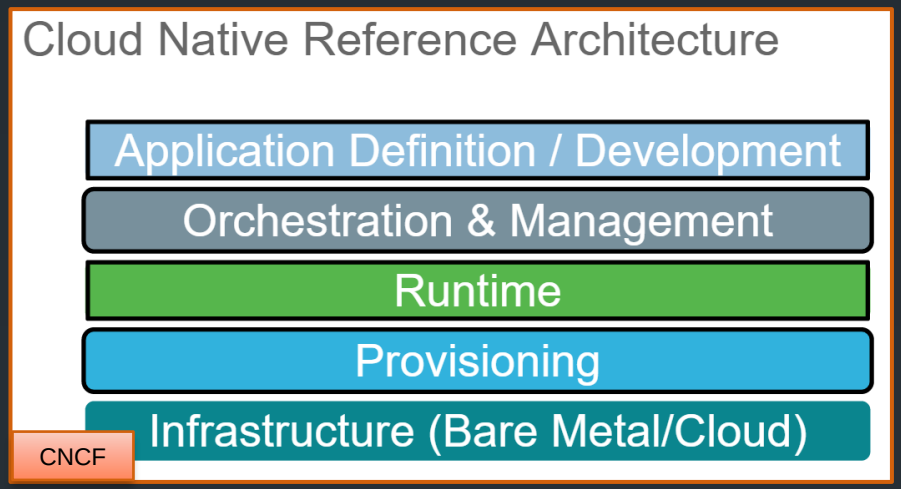
X. Dev/prod parity
Keep development, staging, and production as similar as possible

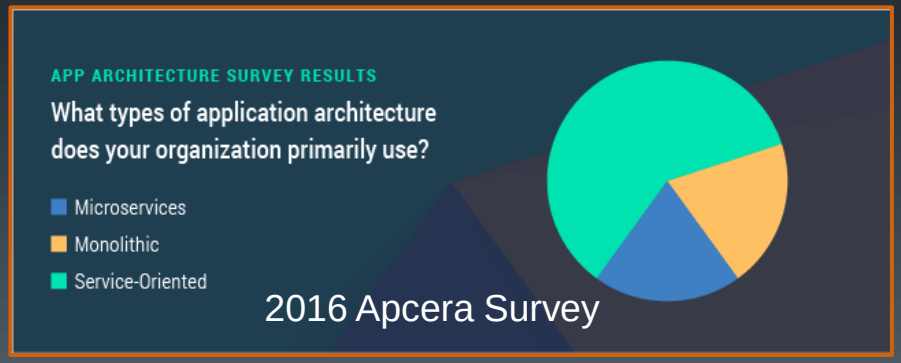
XI. Logs
Treat logs as event streams

XII. Admin processes
Run admin/management tasks as one-off processes

Next Generation Models

- Cloud Native Computing
  - A new computing paradigm optimized for modern distributed systems environments
  - Capable of scaling to tens of thousands of self healing multi-tenant nodes
- Cloud native systems have the following properties:
  - Container packaged
    - Running applications and processes in software containers as an isolated unit of application deployment, and as a mechanism to achieve high levels of resource isolation
    - Improves overall developer experience, fosters code and component reuse and simplifies operations for cloud native applications
  - Dynamically managed
    - Actively scheduled and actively managed by a central orchestrating process
    - Radically improve machine efficiency and resource utilization while reducing the cost associated with maintenance and operations
  - Microservices oriented
    - Loosely coupled with dependencies explicitly described (e.g. through service endpoints)
    - Significantly increase the overall agility and maintainability of applications

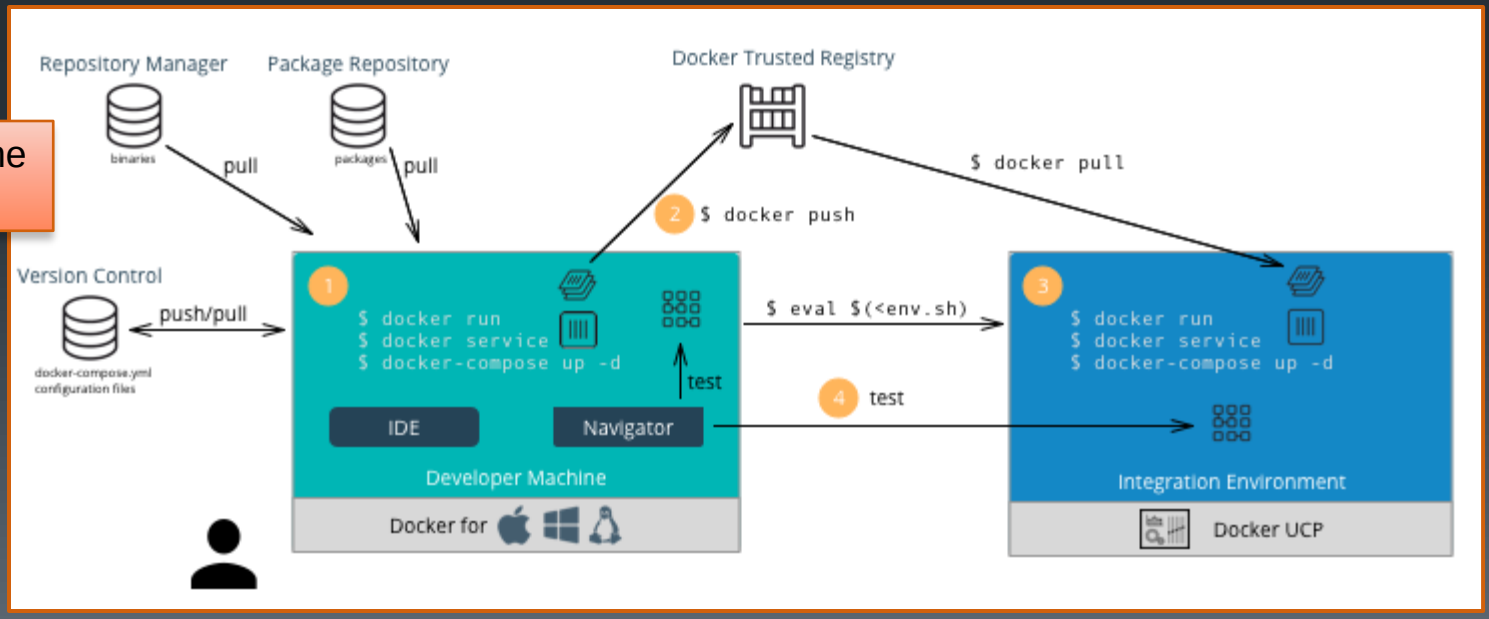




Containers as a Service

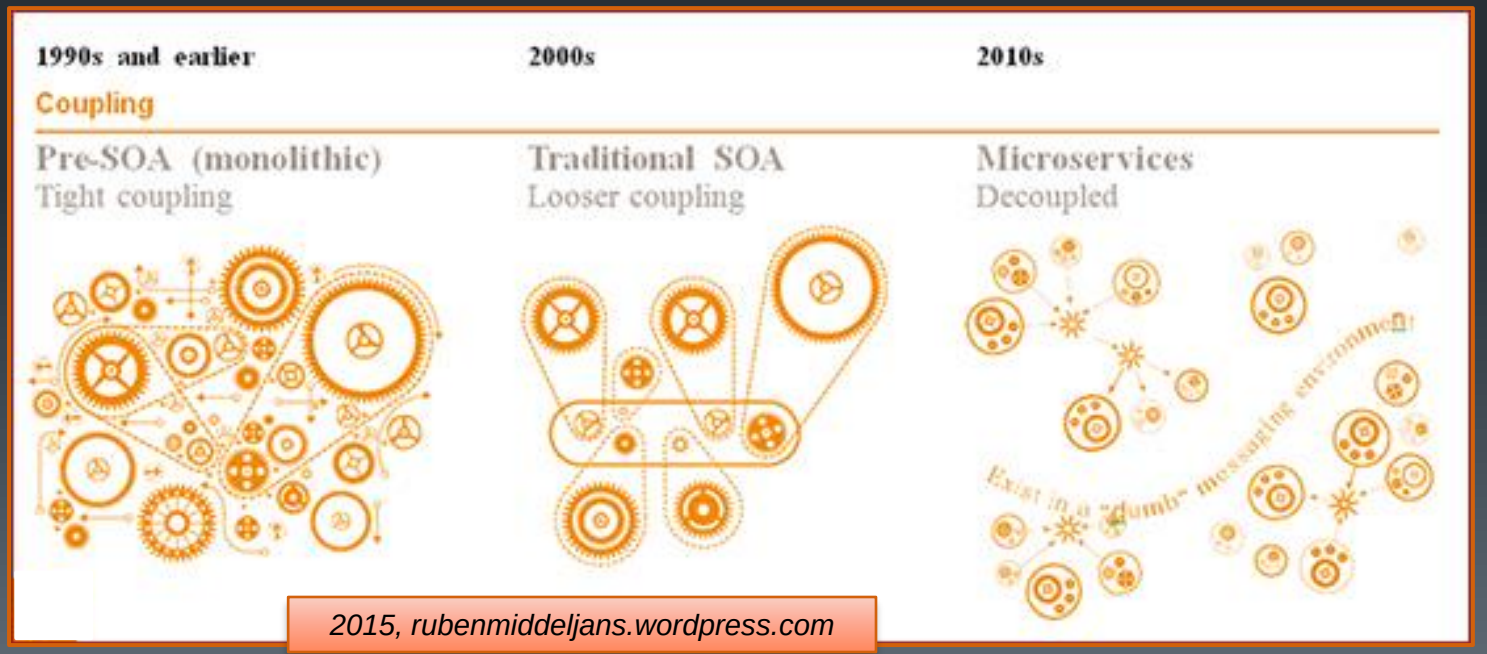
- Containers as a Service (CaaS) allows users to push containers to create running applications
  - IaaS – push VMs
  - PaaS – push code
  - CaaS – push containers
- CaaS deploys and orchestrates
- Containers are created earlier in the pipeline reducing variability
- Hybrid
  - Most popular PaaS systems today use container technology
    - Cloud Foundry uses containers and its own OCI (Open Container Initiative) runc-based Guardian container manager (Greenhouse on Windows)
    - Red Hat OpenShift uses Docker as the container manager and Kubernetes as the orchestration engine
    - Heroku supports container push
    - Elastic Beanstalk supports container push

Docker Dev Pipeline Reference Arch



Building for CaaS: Microservices

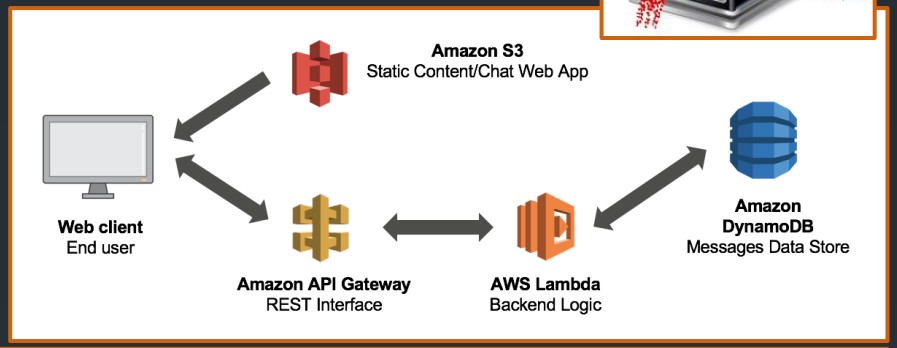
- Microservice:
  - A fine grained atomically deployable service accessed via a platform agnostic network API
- Microservice Architecture (MSA):
  - An architectural pattern used to build distributed software systems where processes communicate with each other over a network
  - Composed of business aligned, goal oriented services
  - Allows designs to evolve and self organize over time
  - Favors symmetry over hierarchy (peer to peer not layers)
  - An architectural approach that seeks to define an application, not an enterprise
- *A deeper look at microservices in section 5*

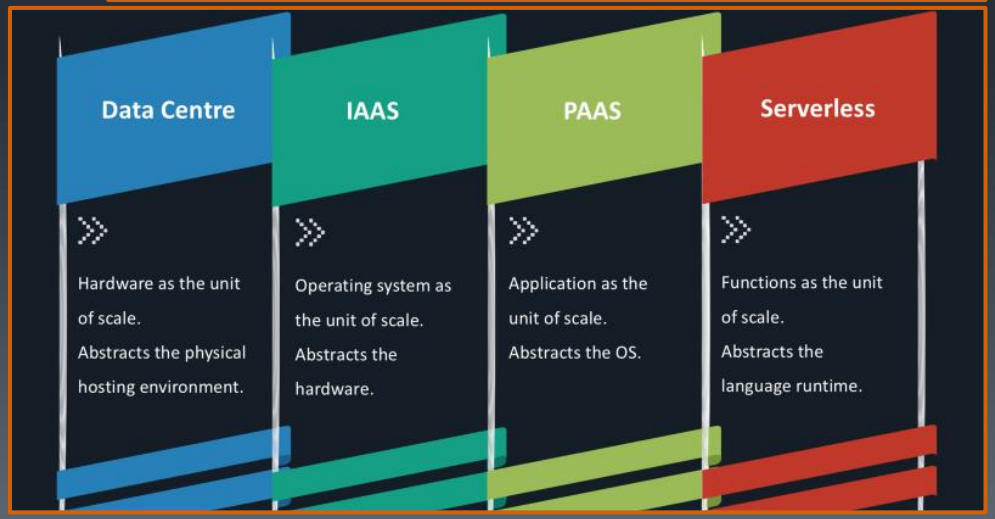


2015, rubenmiddeljans.wordpress.com

Serverless (FaaS)

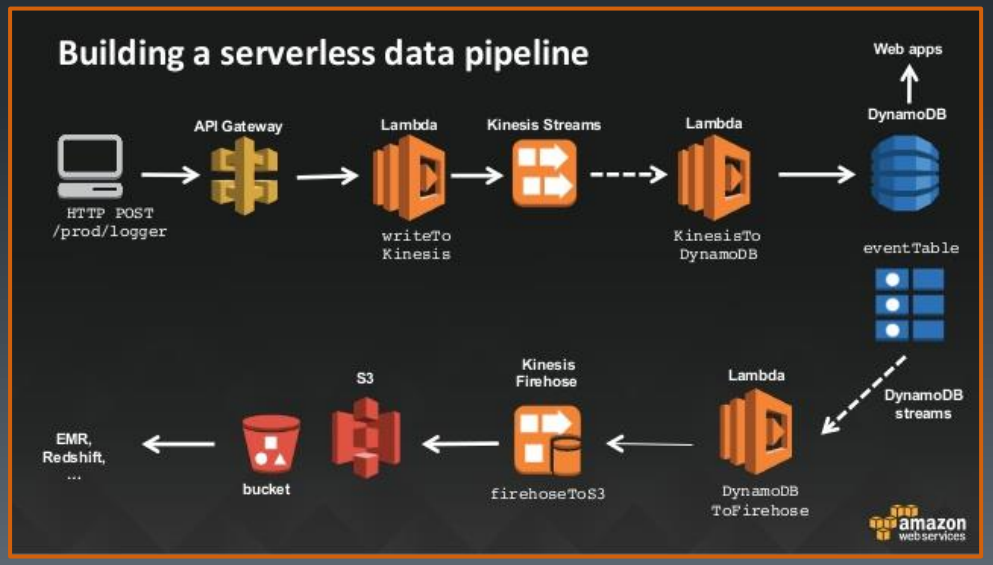
- The cloud has made possible fully "serverless" models of computing
  - On-demand/event-driven
    - Logic can be spun up on-demand in response to events originating from anywhere
  - Composable
    - Applications can be built from these bite-sized bits of business logic
  - Pay for what you use
    - Billing occurs only when code is running
  - Automatic cloud scale
    - Can serve user counts from zero to planet-scale, all without managing any infrastructure
- CNCF working group
  - https://github.com/cncf/wg-serverless





Serverless Applications

- Extensive pipelines can be configured through event chains invoking one or more functions in the progression of processing
- Contrasts:
  - PaaS applications are established by pushing code to the cloud as source code control commits
    - *PaaS applications are monolithic and typically always have at least one server process running that receives external requests*
    - *Since scaling is achieved by booting up more server processes, scalability remains visible to the developer*
  - CaaS applications are established by deploying container images onto a cluster (typically VMs) from an image registry
    - *Container deployment details are created and maintained as part of the application*
    - *Requires existing infrastructure for cluster nodes*
  - FaaS applications are established by defining pipelines and uploading the code to execute when triggered
    - *Avoids consuming idle infrastructure incurring charges when not in use*
    - *Functions are composable-–can be triggered from many different pipelines*



Summary

- Cloud platforms have evolved new layers to support ever more efficient application platforms
  - IaaS
  - PaaS
  - CaaS
- New cloud models are always evolving to abstract away infrastructure and deployments
  - Serverless / FaaS
- Consumers receive functionality more and more directly from the cloud without managing anything
  - SaaS

Lab 2

- Cloud Load Balancing
  - Configure Load Balancing between compute instances on gcloud

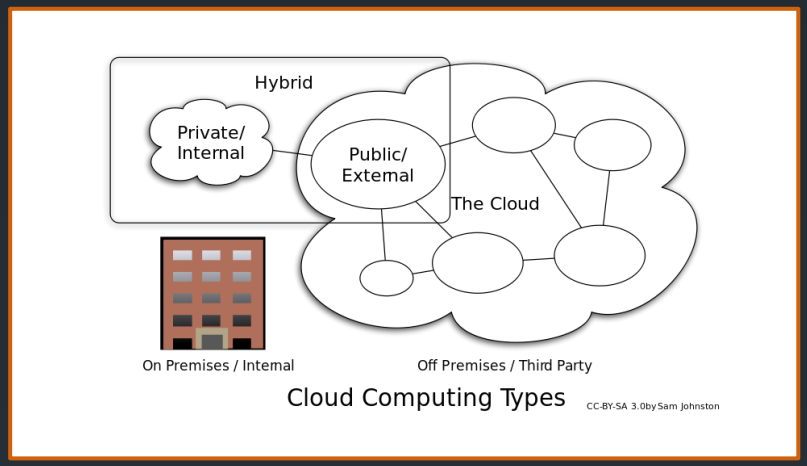
3: Cloud security and governance

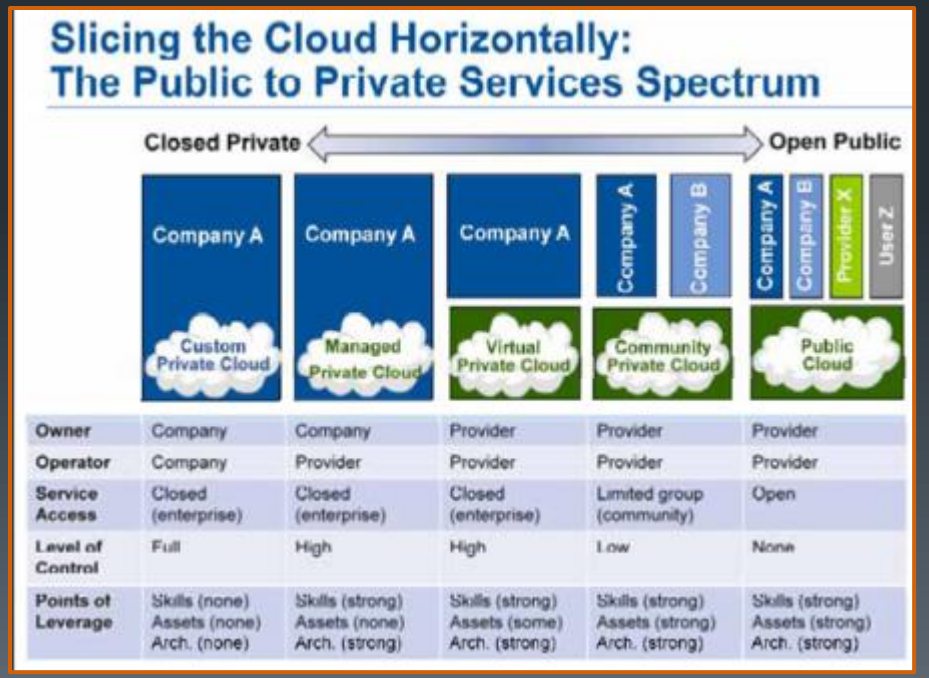
Objectives

- Discuss the three Cloud service types: public, private and hybrid clouds
- List the features of bare metal clouds
- Examine key governance and cloud security concerns

Cloud Types

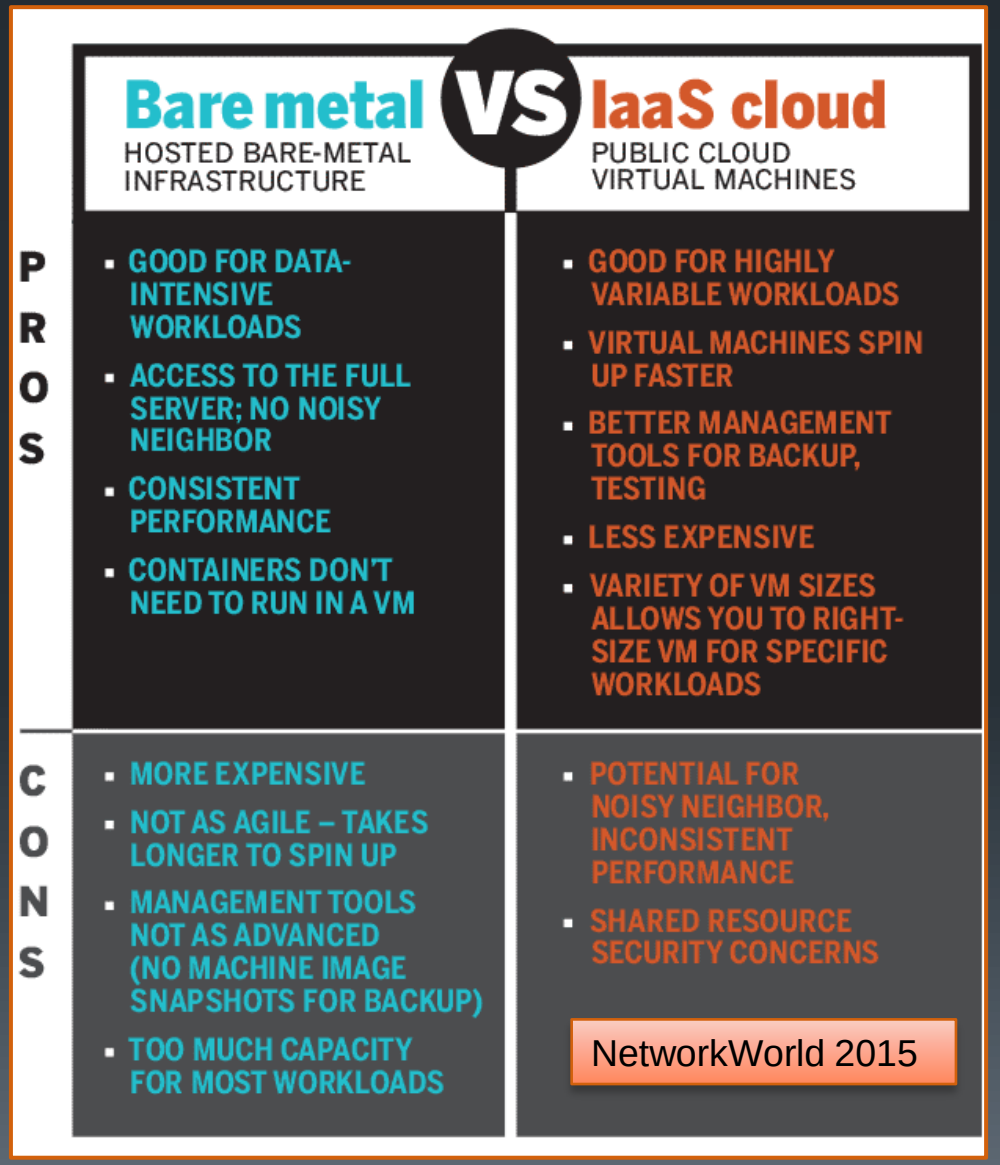
- There are essentially three cloud service types:
  - Public Cloud
    - As the name implies, public clouds are offered to the general public. They're accessible via an Internet connection and shared among multiple, often thousands, of customers. Some of the largest public cloud providers include Amazon Web Services, Microsoft Windows Azure, and Rackspace Cloud.
  - Private Cloud
    - Private clouds are typically created and hosted by a single organization, usually behind the corporate firewall, which provides services on request to employees. Private clouds can also be hosted by third parties, but they remain dedicated to a single customer. They can be more costly than public clouds but offer greater control over your data and better fault tolerance.
  - Hybrid Cloud
    - This term typically applies to organizations whose IT operations combine internal private cloud services with external public cloud services. It may also refer to service offerings used exclusively by an invited group of private customers (also called a "community cloud")





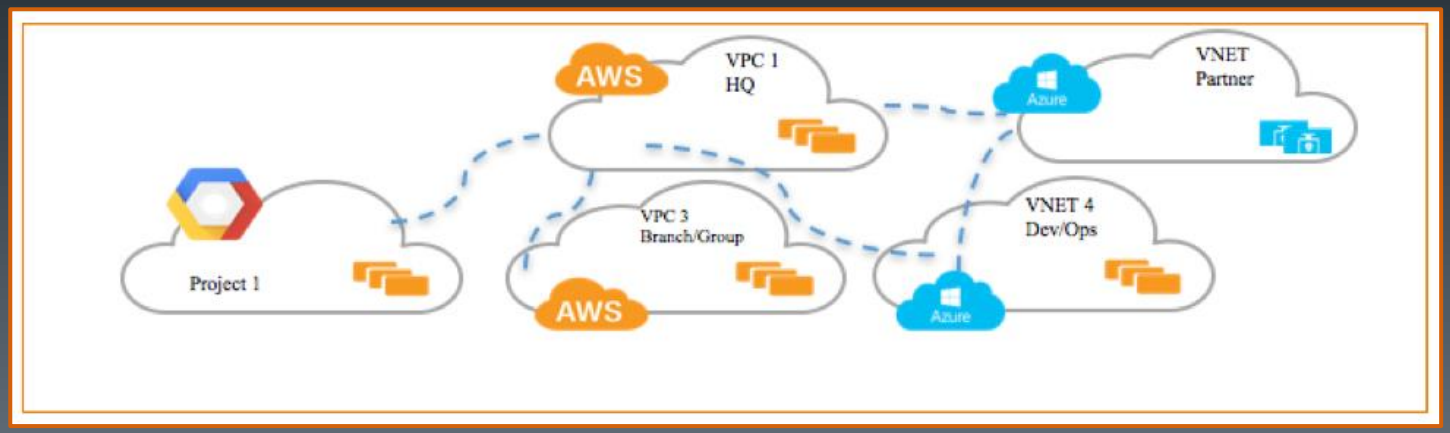
Bare Metal Clouds

- Bare-metal server – a physical computer server with a single tenant
  - Dedicated entirely to the customer who is renting them
    - Unlike many servers in a data center, they are not shared between multiple customers
  - Each logical server offered for rental is a distinct physical piece of hardware that is a functional server on its own
    - Not virtual servers running on shared hardware
- Infrastructure as a Service and Infrastructure as Code offer many advantages making cloud hosting convenient
- Combining the features of cloud hosting and bare-metal servers offers most of the benefits while still conveying the performance advantages of bare metal
  - Cloud based bare metal server offerings are known as bare metal clouds
- Key bare metal cloud providers
  - Packet
  - IBM Cloud
  - Oracle
- Bare metal servers eliminate the shared physical resource exposure of typical cloud VMs
  - Many clouds offer dedicated hosts where only VMs from a single client are allowed to run on the underlying system, providing most of if not all of the security benefits of bare metal



Multicloud

- Multicloud architectures utilize two or more public clouds in a single solution architecture
  - May also include multiple private clouds
- Reasons for deploying a multicloud architecture:
  - Reduced reliance on any single vendor
  - Increased flexibility and cloud service range/choice
  - Disaster avoidance/mitigation
  - Cost optimization
- Models
  - Active/Active – A given workload is load balanced across multiple providers
    - This model prioritizes resilience and maximal uptime
  - Active/Passive – A given workload runs on one provider with a backup on another
    - This model prioritizes disaster recovery, often minimizing cost on the passive side and simplifying active operation
  - Collaborative – Parts of a workload operate on one cloud and other parts operate on a second cloud
    - This model is often used to optimize cloud service consumption (best of breed) where one cloud offers optimal services for one part of an application and another cloud offers optimal services for another part of an application
    - Used in cases where no one provider can be everything for everyone
  - Many other models are possible
- Multicloud environments add complexity to security and governance



Top Clouds by Region

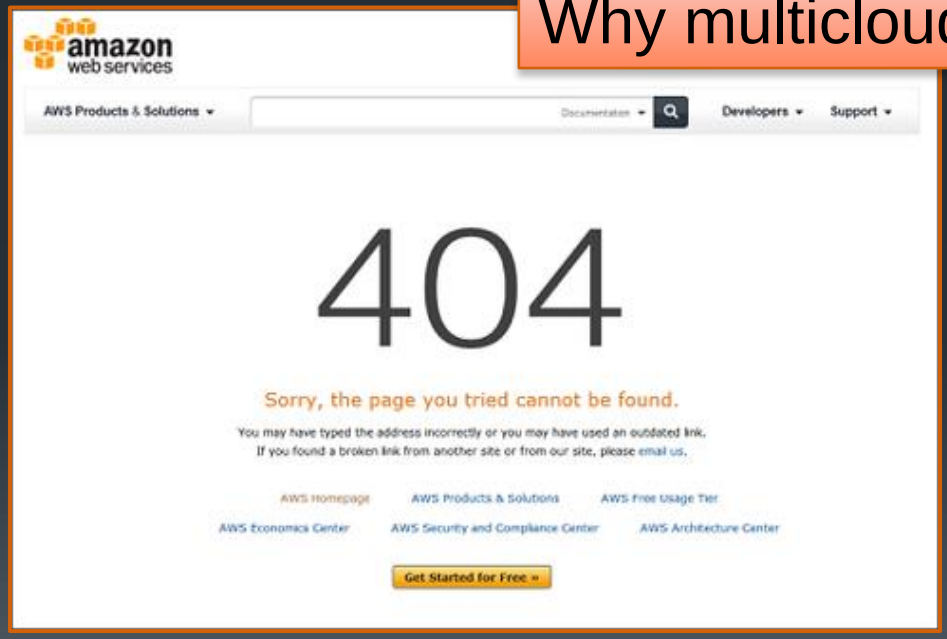
- Availability and adoption varies regionally
  - Amazon AWS is the top provider in all regions but 2
  - Other top providers shuffle depending on region
    - Microsoft and Google clouds typically in the top 3
- Using a single public cloud provider may not be feasible in all scenarios

Courtesy RiskRecon & Cyentia Institute

Selected Cloud Outage Examples

- 2011-04-21 most of the AWS US-East region went down for several hours
- 2012-12-24 many services running on AWS in the US-East region were unavailable for more than 14 hours
  - This included Netflix servers
  - This was the result of a developer deleting data from a production system
- 2015-09-20 AWS experienced significant error rates with DynamoDB in the US-East region, which impacted other AWS services in that region
  - This incident took Netflix, Tinder, Airbnb, Reddit and IMDb offline
- 2017-02-28 US-East-1 region suffered failure in thirty-three of AWS's services including nine services which suffered complete disruption: Athena, EMR, Inspector, Kinesis Firehose, Simple Email Service, S3, WorkMail, Auto Scaling and CloudFormation
  - This caused a chain reaction, taking countless cloud-based applications and websites offline
- These failures were not restricted to a single availability zone within the region
  - This is contrary to the company's vision and marketing which describes lack of interdependence among data centers
  - Servers across many different zones were unavailable for several hours
  - Companies that based their architectures on AWS recommendations for high availability found themselves down
- AWS has continued to improve its service to remove interdependence, but issues across availability zones have continued to surface (though at diminishing scale)

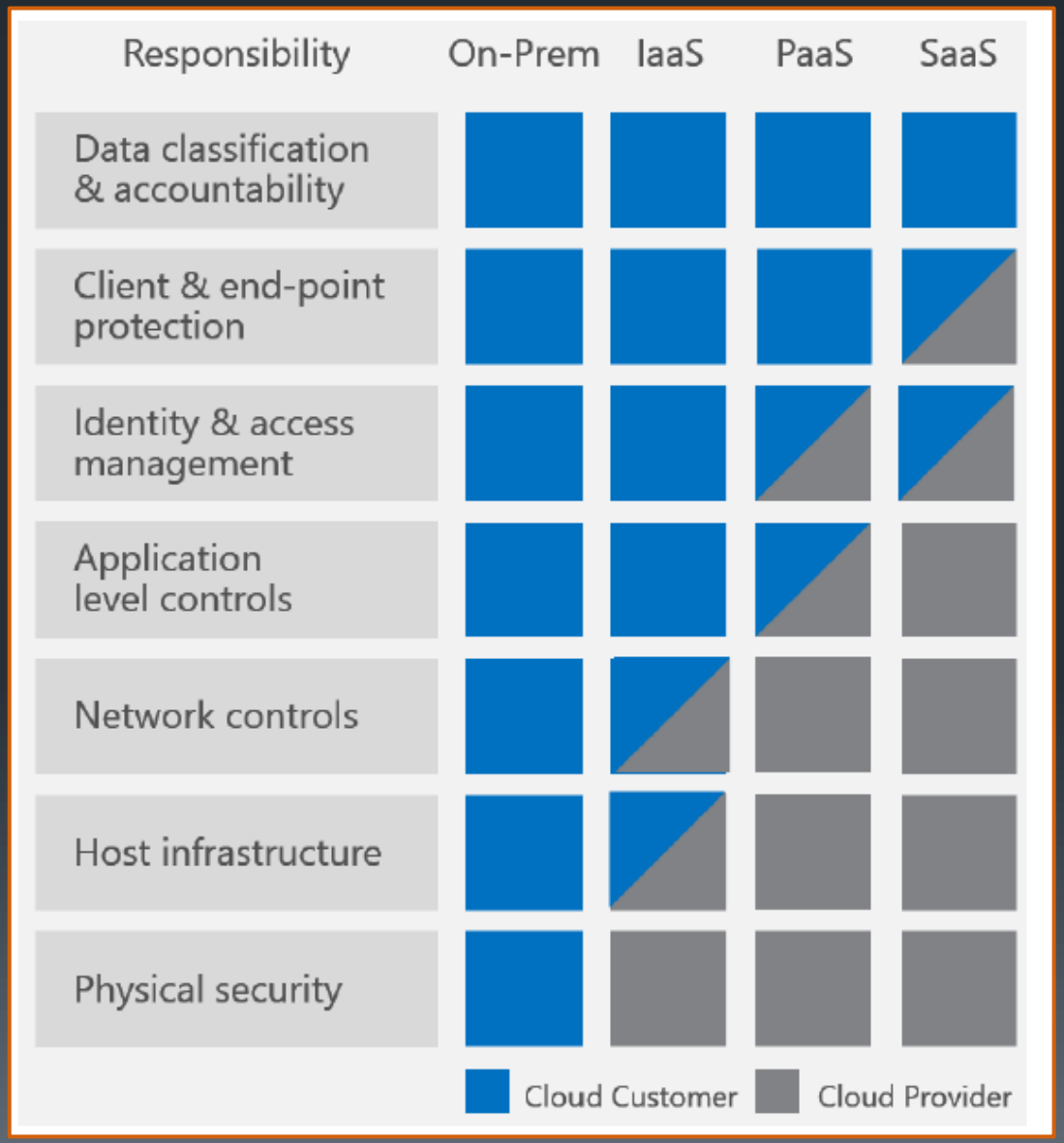
Why multicloud?



Every major cloud provider has had a significant outage once acquiring enough customers, including GCP, Azure, IBM Cloud, etc.

*aaS Shared Security Responsibilities

- Clouds create a shared responsibility model
  - Relieves but does not eliminate customer operational burden
- IaaS providers operate/manage/control:
  - Host OS
  - Virtualization layer
  - Physical security of facilities
- Customers manage:
  - Guest OS (updates/security patches)
  - Application software
  - Configuration of IaaS firewalls (host/network/gateway)
- Customer responsibilities vary based on:
  - Service model (IaaS/PaaS/CaaS/SaaS)
  - Cloud services used
  - Integration of cloud services into customer IT environment
  - Laws and regulations

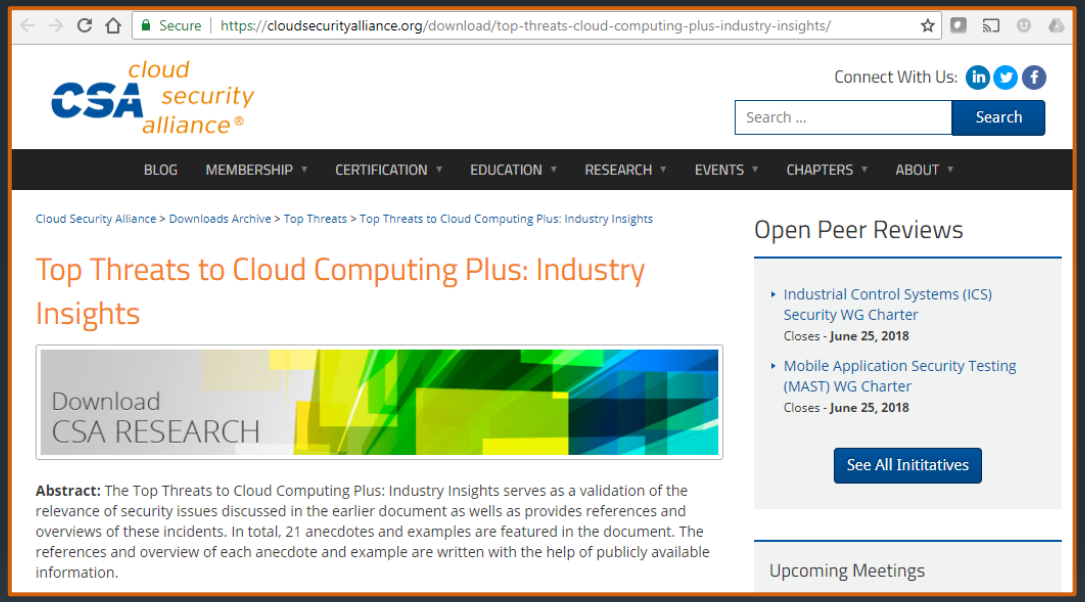


Cloud Governance Risks

- Shadow IT
  - The key advantage of the cloud over the traditional data center model is the ease of provisioning services
  - This provides an easy pathway for organizational units and departments to engage cloud providers without going through the traditional IT procurement process
- Expenses are operational vs. capital
  - Making it more difficult to track
- Inexperienced buyers
  - Consumers may not have the technical knowledge or experience to understand the nature of services provided
  - May buy more than they need (reducing the cloud cost benefits)
- Cloud services should be treated as high risk
  - Requiring controls to monitor and manage service level agreements
  - Deploying needed process controls and monitoring systems
- Cloud services require more rather than less governance from organizations

Treacherous 12

- CSA (Cloud Security Alliance) top 12 cloud computing threats organizations face
  1. Data breaches
  2. Insufficient identity, credential, and access management
  3. Insecure interfaces and APIs
  4. System vulnerabilities
  5. Account hijacking
  6. Malicious insiders
  7. Advanced persistent threats (APTs)
  8. Data loss
  9. Insufficient due diligence
  10. Abuse and nefarious use of cloud services
  11. DoS
  12. Shared technology vulnerabilities



https://downloads.cloudsecurityalliance.org/assets/research/top-threats/Treacherous-12_Cloud-Computing_Top-Threats.pdf

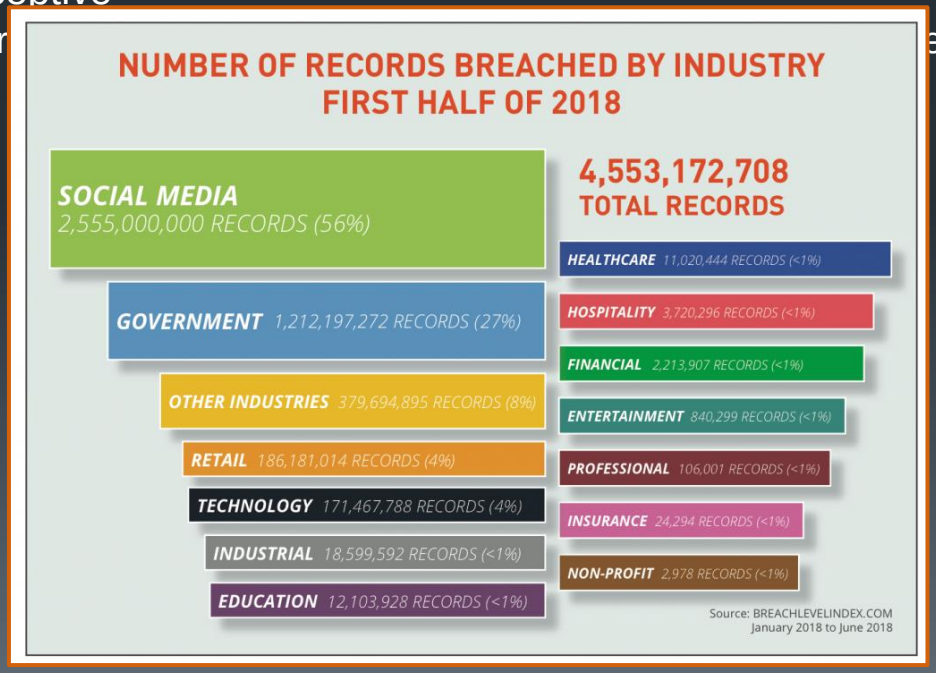
Data in the Cloud

- Classifying data from a protection standpoint:
  - Sensitivity
  - Regulatory/compliance restrictions
    - E.g. must it stay within your national boundary?
- Requires standards for:
  - Data classification must be easy for all originators of data to use
    - E.g. ISO/US-CERT Traffic Light Protocol (TLP): https://www.us-cert.gov/tlp
  - Standardized metadata identifying what security needs to be applied to each item of data
    - Cloud data resources (block storage volumes, object storage buckets, etc.) should be tagged (with TLP or similar)
- With an understanding of what security you need to apply to your data, you're in a position to decide:
  - What data and processes can be moved to the cloud
  - Which cloud models and vendors are suited to your needs
- Security in depth practices almost always require data to be encrypted at rest and in transit

		Attacker			
		Provider (or its employees)		Hacker	
		Plausibility (Motivation)	Success probability	Plausibility (Motivation)	Success probability
Approach	1. Local, on-premises storage of encrypted data	Low	Low	Moderate	Significant
	2. Cloud storage without client-side encryption	Moderate	Large	Large	Moderate
	3. Cloud storage with client-side, end-to-end encryption	Low	Low	Moderate	Moderate

Data Risk Example

- Dropbox password authentication was accidentally disabled for four hours in June of 2011
  - This allowed anyone to log into any Dropbox account with any password and access any data on the platform
  - According to Dropbox, "less than 1 percent" of its 45 million users logged in during that time
    - Therefore approximately 450,000 users had illicit access to other users' data
- This issue was made public by security researcher Christopher Soghoian, who received a tip from a Dropbox user
  - The company was either unaware or unwilling to make the matter public
  - Soghoian filed a complaint against Dropbox with the Federal Trade Commission alleging the company's security claims had been deceptive
    - Dropbox had advertised that "All files stor… inaccessible without your account password."
    - Later the company altered its claims to make it clear that Dropbox, rather than the account holder, controls the file encryption keys, thereby enabling Dropbox to provide access to account holders' files when presented with lawful demands from authorities
- If files were encrypted by the client before they were sent to the cloud the platform would be inherently more secure



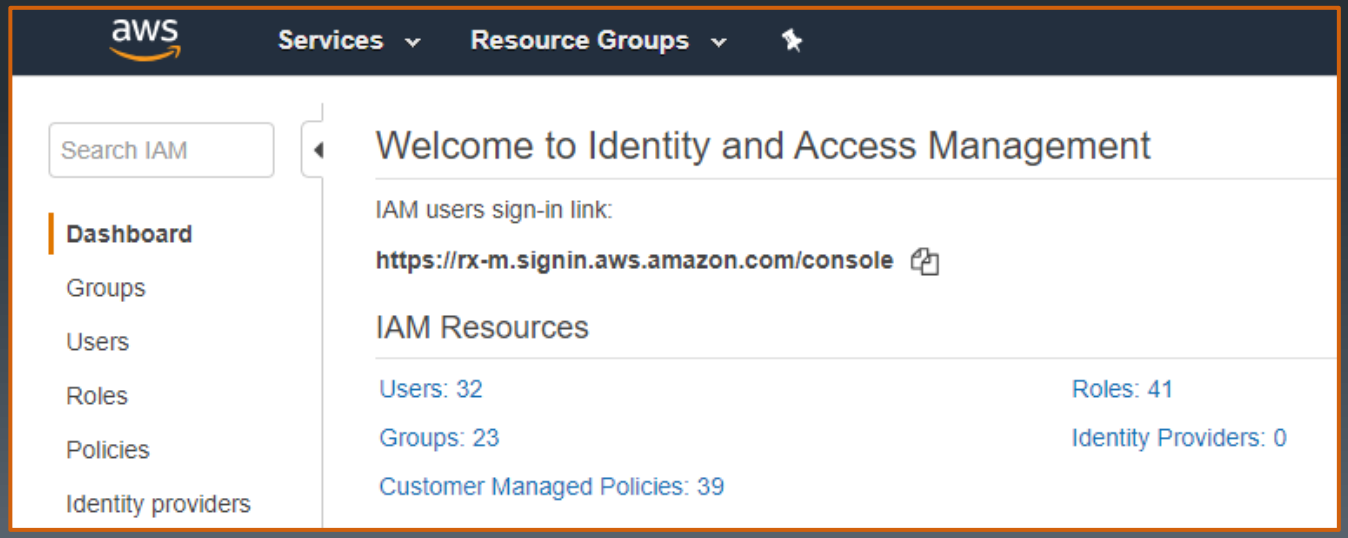
CASB

- Cloud Access Security Broker
  - Sometimes pronounced “cas-bah”
- On-premises or cloud based software that sits between cloud service users and cloud services
- Adopted by many enterprises to manage governance risk
- A CASB can offer a variety of services:
  - Monitoring user activity
  - Warning administrators about potentially hazardous actions
  - Enforcing security policy compliance
  - Automatically preventing malware

Figure 1. Magic Quadrant for Cloud Access Security Brokers

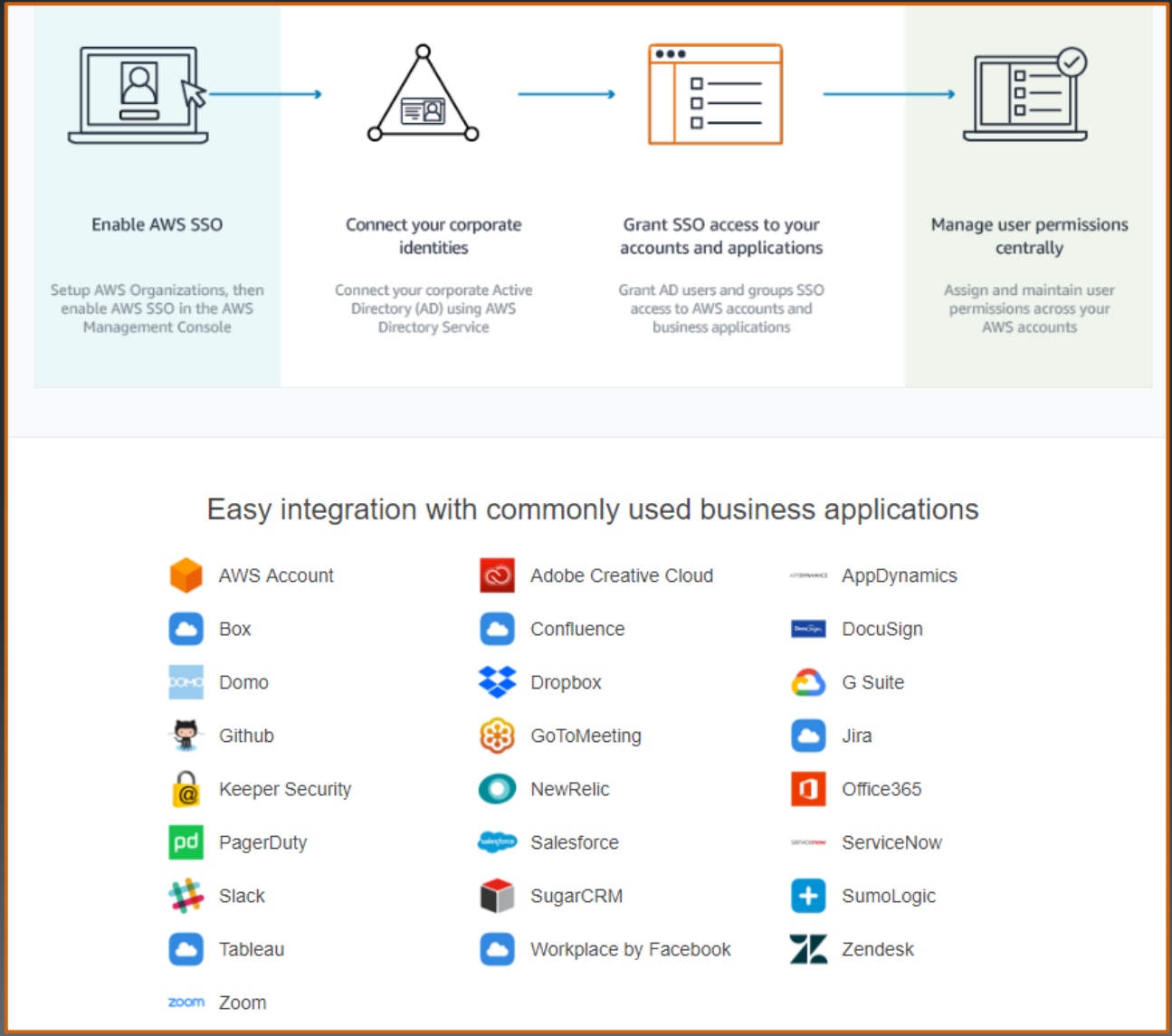
Identity and Access Management

- Management of cloud identity and access is among the more challenging problems for enterprises [Auth]
  - Ensuring that an individual is who they say they are [Authentication/AuthN]
  - Ensuring that individuals only do what they are allowed to do [Authorization/AuthZ]
- Cloud Auth services control how an enterprise's users and systems access cloud-based applications and services
- Cloud Security Alliance (CSA) core functions of cloud identity and access:
  - Identity provisioning & deprovisioning
  - Authentication & federation
  - Authorization & user profile management
  - Support for compliance



Auth Federation

- Federation enables single sign-on (SSO) access to cloud resources using credentials from a corporate directory
  - LDAP (Lightweight Directory Access Protocol)
  - Active Directory
- Federation uses open standards:
  - Security Assertion Markup Language 2.0 (SAML)
  - OpenID Connect
- Many clouds offer SSO to manage access to multiple accounts and business applications centrally

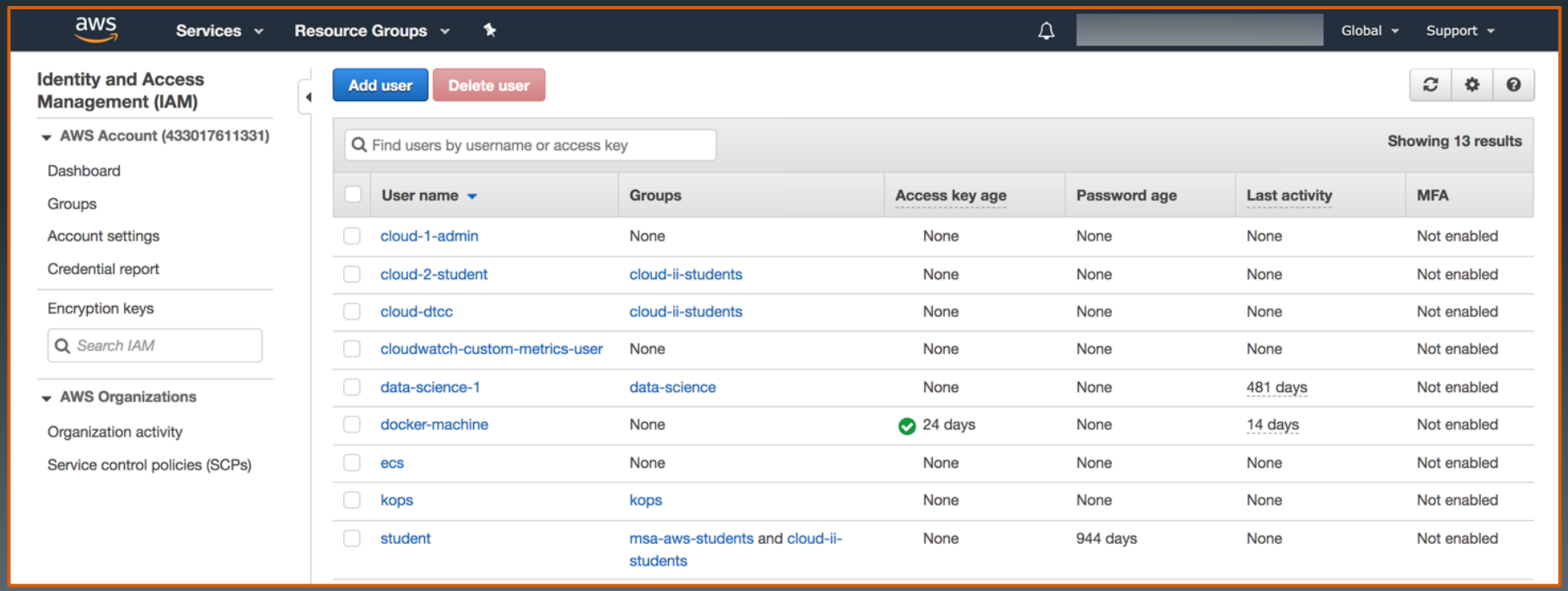


A case study

AWS Identity and Access Management (IAM)

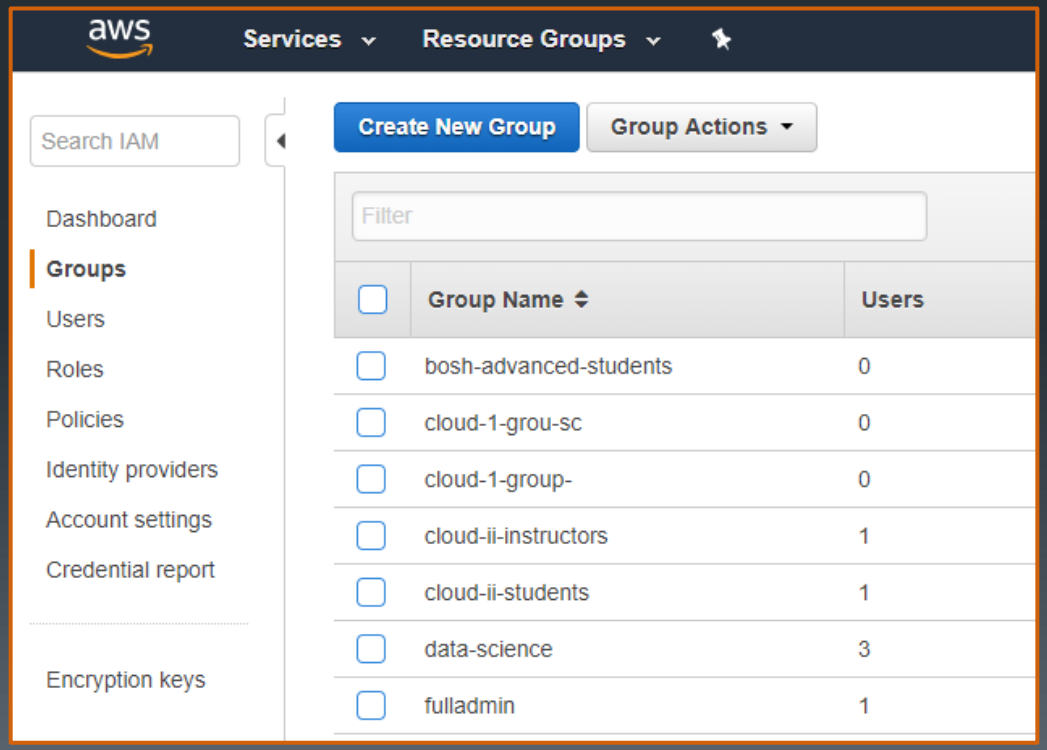
Users

- Users are principals that can gain access to the system through authentication
  - Natural persons
    - Human users
  - Software programs or computers
    - This type of user is represented by a “service account” in many environments



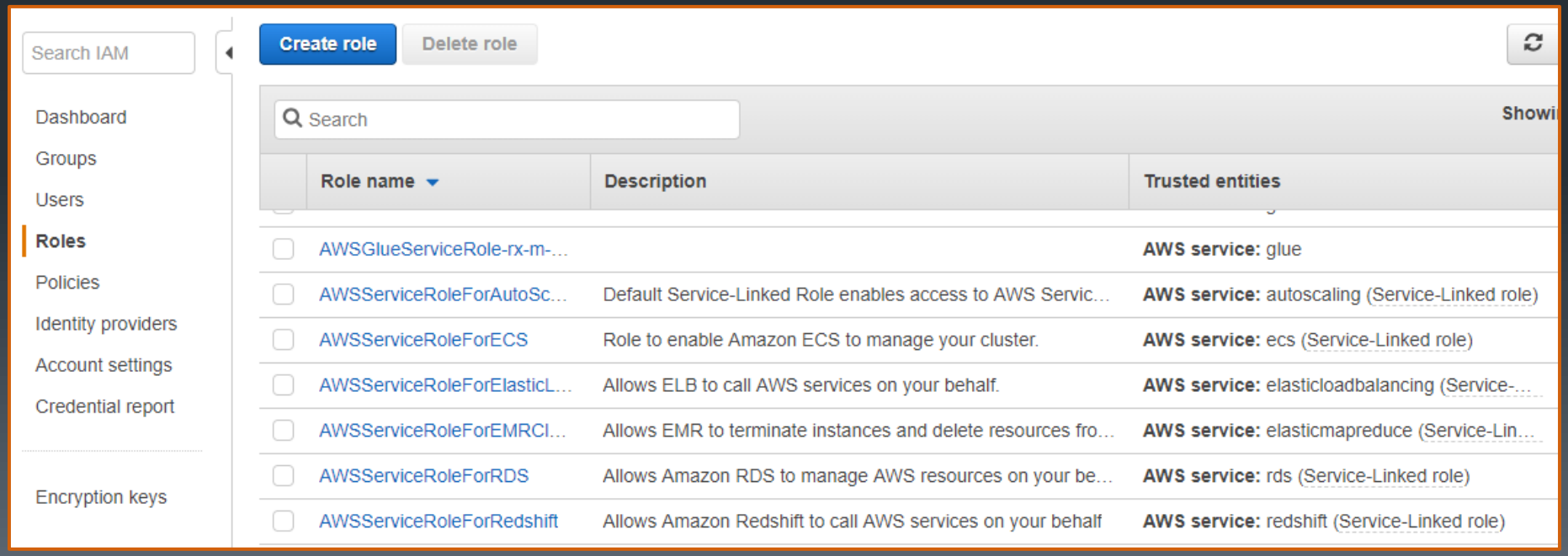
Groups

- Groups simplify administration of permissions
  - Permissions are assigned to groups
    - Rather than every single user explicitly
  - Users are added/removed from groups
    - Allows the user to make use of the permissions for the groups they are a member of



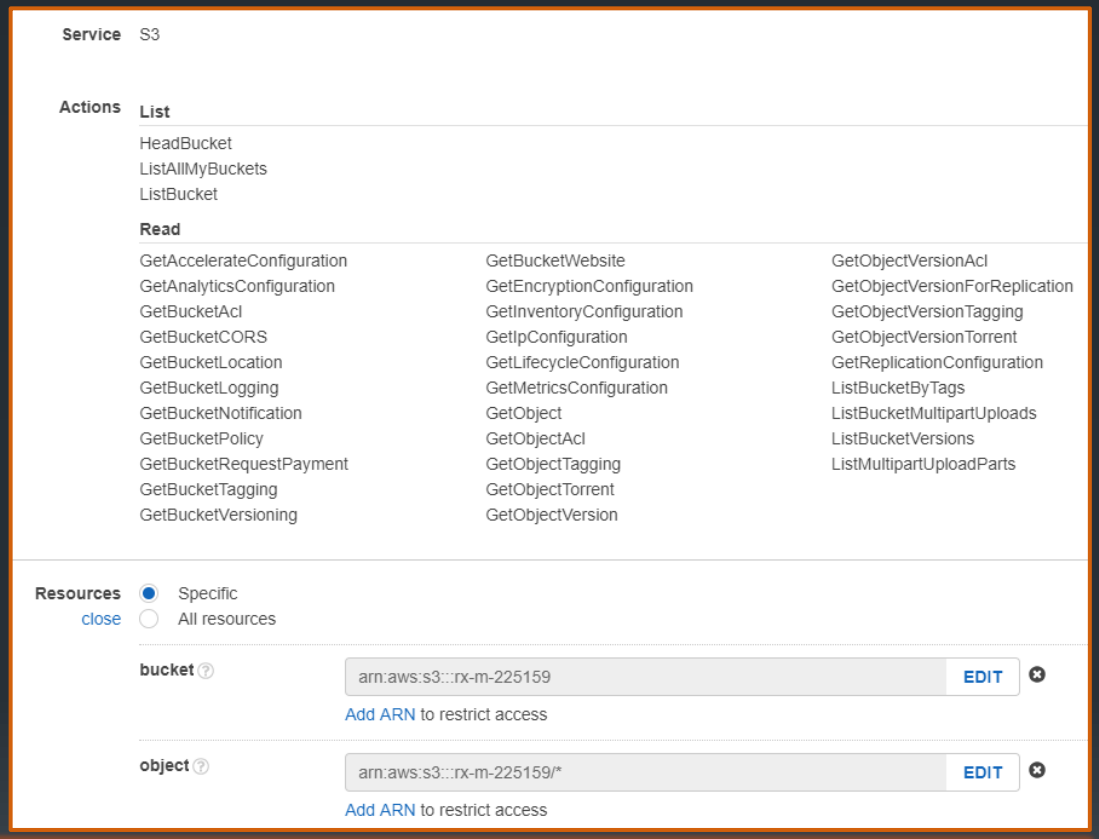
Roles

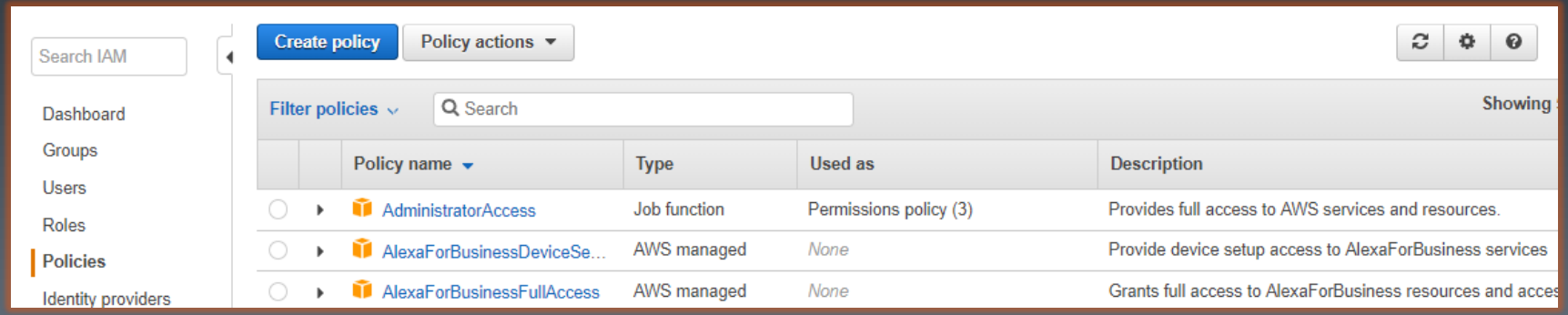
- Roles issue keys that are valid for short durations, making them a secure way to grant access to the platform
- Role can be assigned to:
  - Users in other accounts/organizations
  - Application code running on cloud instances
  - Services that need to act on resources
  - Users from a corporate directory identified through federation



Policies

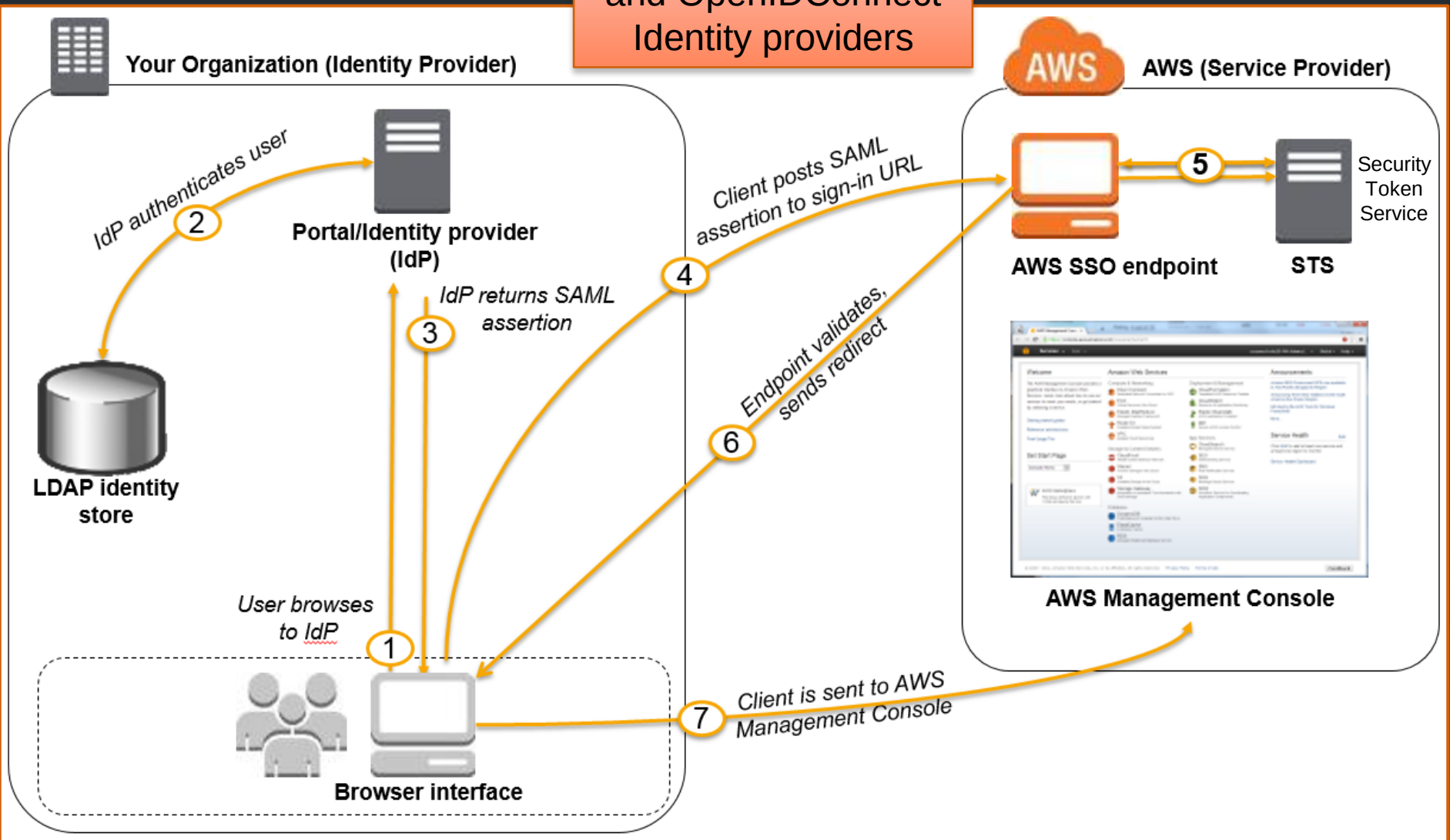
- A policy defines the permissions assigned to a user, group, or role





AWS Single Sign on with SAML

AWS supports SAML and OpenIDConnect Identity providers



Summary

- Clouds can be deployed over many governance domains
  - Private data centers
  - Public cloud data centers
  - Hybrids of the two
- Federation allows authentication to span all sides of hybrid clouds
- A wide range of governance and security concerns are specific to operations in the cloud

Lab 3

- AWS CLI introduction and Identity and Access Management (IAM)
  - Install and Configure the AWS Command Line
  - Use the AWS Command Line to explore Identity and Access Management (IAM)

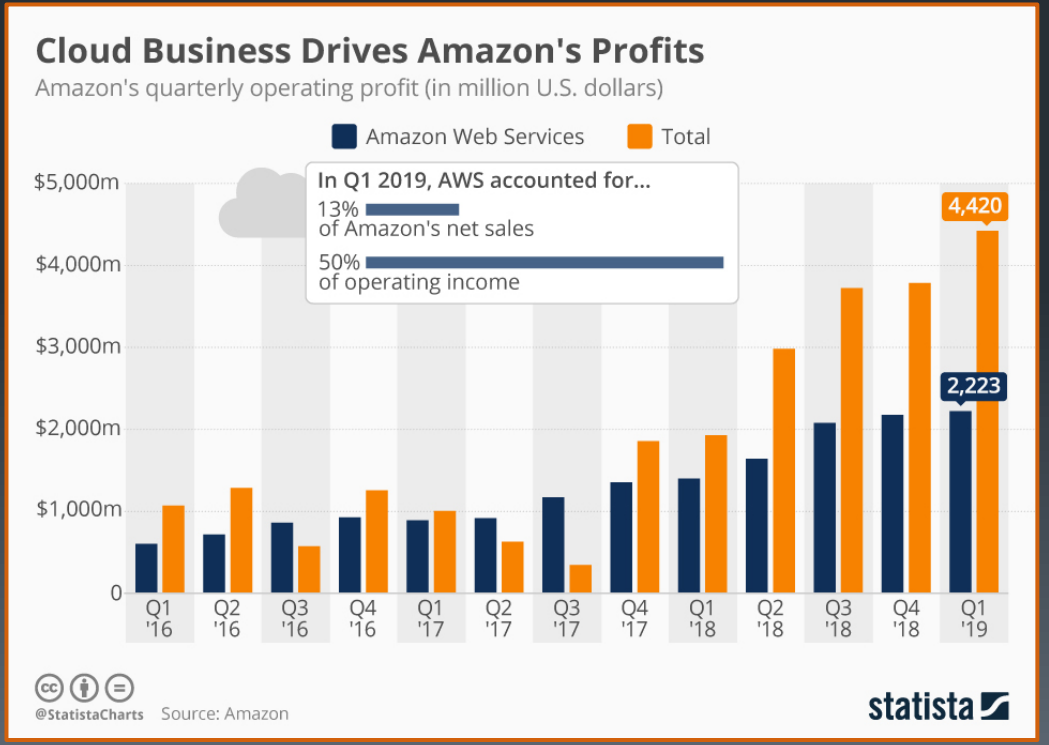
4: AWS case study

Objectives

- Explore the AWS IaaS services
  - Compute
    - Elastic Compute 2 (EC2)
  - Networking
    - Virtual Private Cloud (VPC)
  - Storage
    - Simple Storage Service (S3)
    - Elastic Block Storage (EBS)

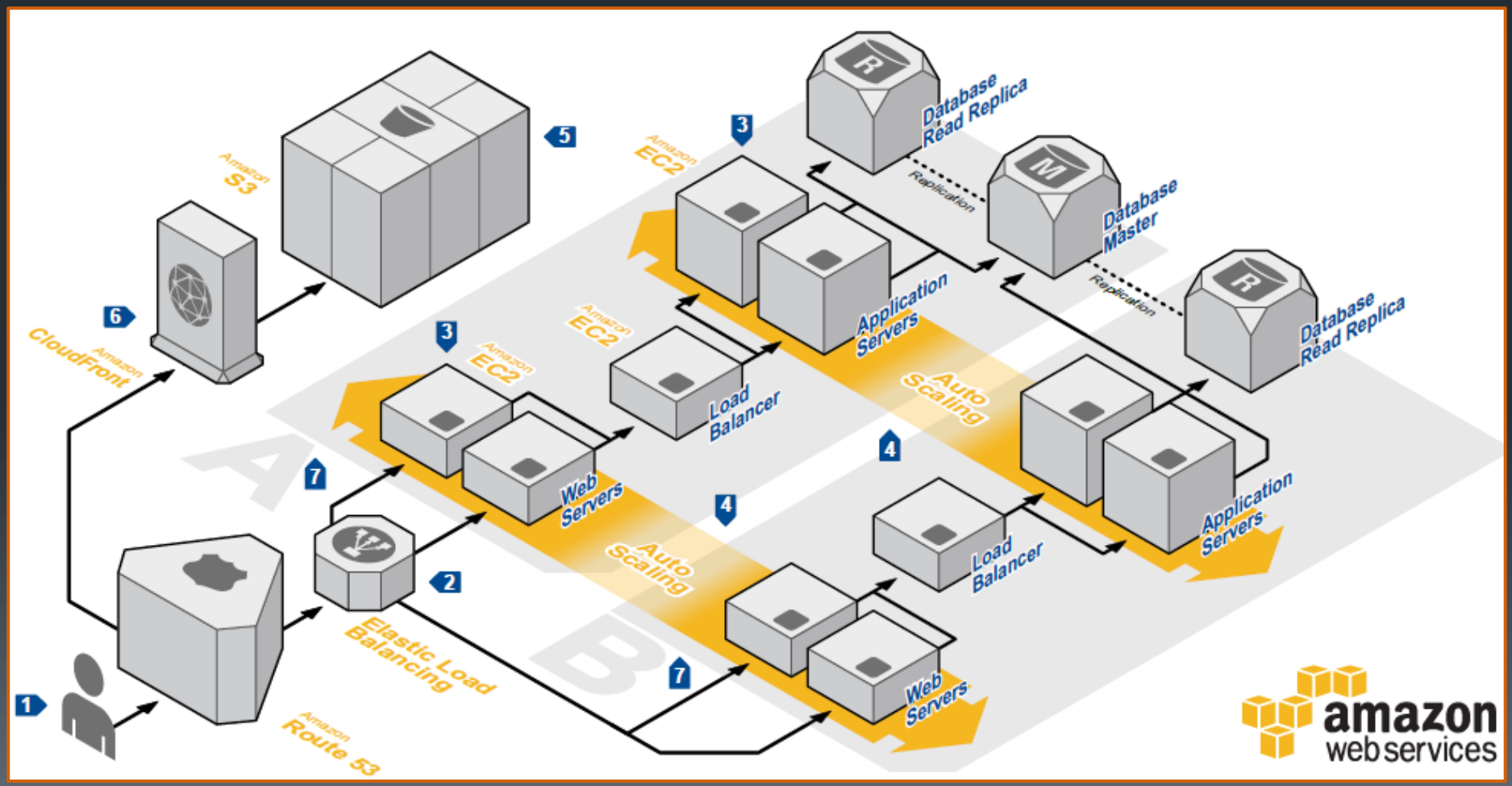
AWS

- Amazon Web Services (AWS)
  - A subsidiary of Amazon.com
  - Provides on-demand cloud computing platforms to individuals, companies and governments
- Fees are based on a combination of usage metrics
  - Compute instances (VMs)
  - Network bytes sent/received
  - Data stored
  - Etc.
- Most services are not exposed directly to end users, instead functionality is offered through APIs for developers to use in their applications
  - Amazon Web Services' offerings are accessed over HTTP using the REST architectural style



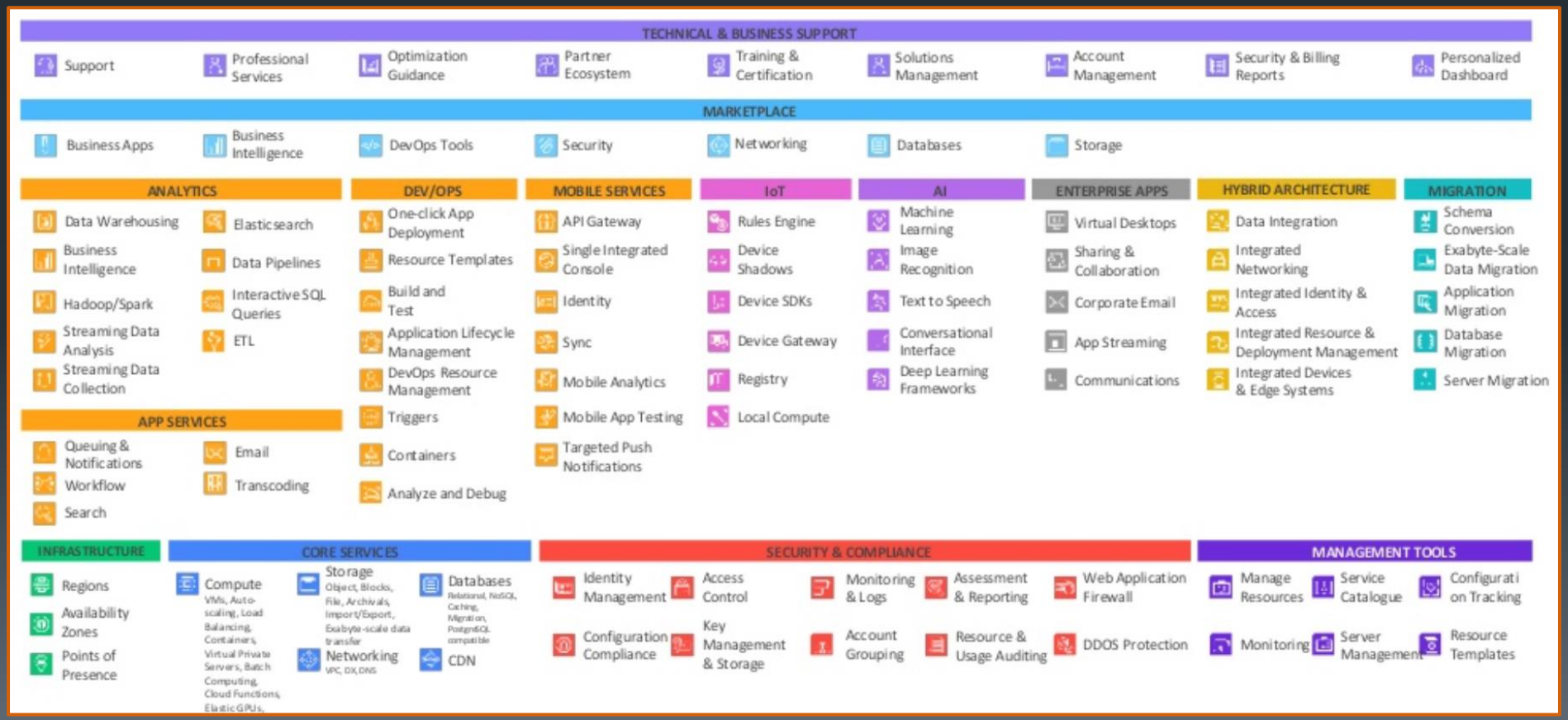
Key AWS Services

- Traditional 3 tier architectures are directly supported by AWS IaaS services



AWS Services

- AWS offers more than 200 services
- Services include computing, storage, networking, database, analytics, application services, deployment, management, mobile, developer tools, and tools for the Internet of Things



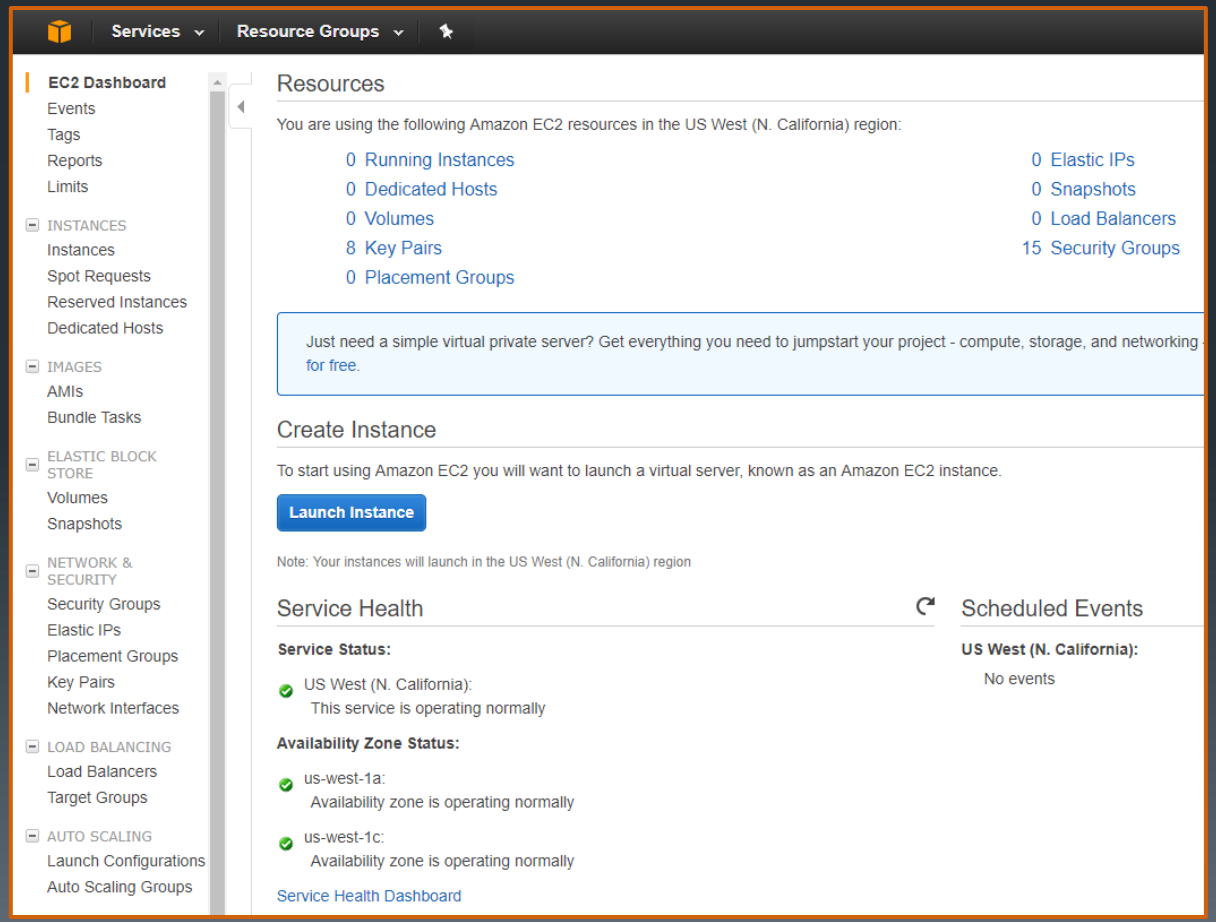
AWS Regions and AZs

- Amazon EC2 is hosted in multiple locations world-wide
- AWS data centers are organized into Regions and Availability Zones
  - Region - a separate geographic area
    - Communications between regions are generally billed
    - Resources are *not replicated* across regions unless configured to do so
  - Availability Zone – an isolated location within a region
    - Communications across AZs are generally free
    - Resources can be replicated across AZs economically
- Failures can occur that affect the availability of instances that are in the same location
  - Either AZ or region
  - Regional failures are generally less frequent than AZ failures
- If you host all resources in a single location (AZ or region) affected by a failure, none of your resources would be available during the outage

Region Name	Code
US East (Ohio)	us-east-2
US East (N. Virginia)	us-east-1
US West (N. California)	us-west-1
US West (Oregon)	us-west-2
Africa (Cape Town)	af-south-1
Asia Pacific (Hong Kong)	ap-east-1
Asia Pacific (Mumbai)	ap-south-1
Asia Pacific (Osaka-Local)	ap-northeast-3
Asia Pacific (Seoul)	ap-northeast-2
Asia Pacific (Singapore)	ap-southeast-1
Asia Pacific (Sydney)	ap-southeast-2
Asia Pacific (Tokyo)	ap-northeast-1
Canada (Central)	ca-central-1
China (Beijing)	cn-north-1
China (Ningxia)	cn-northwest-1
Europe (Frankfurt)	eu-central-1
Europe (Ireland)	eu-west-1
Europe (London)	eu-west-2
Europe (Milan)	eu-south-1
Europe (Paris)	eu-west-3
Europe (Stockholm)	eu-north-1
Middle East (Bahrain)	me-south-1
South America (São Paulo)	sa-east-1

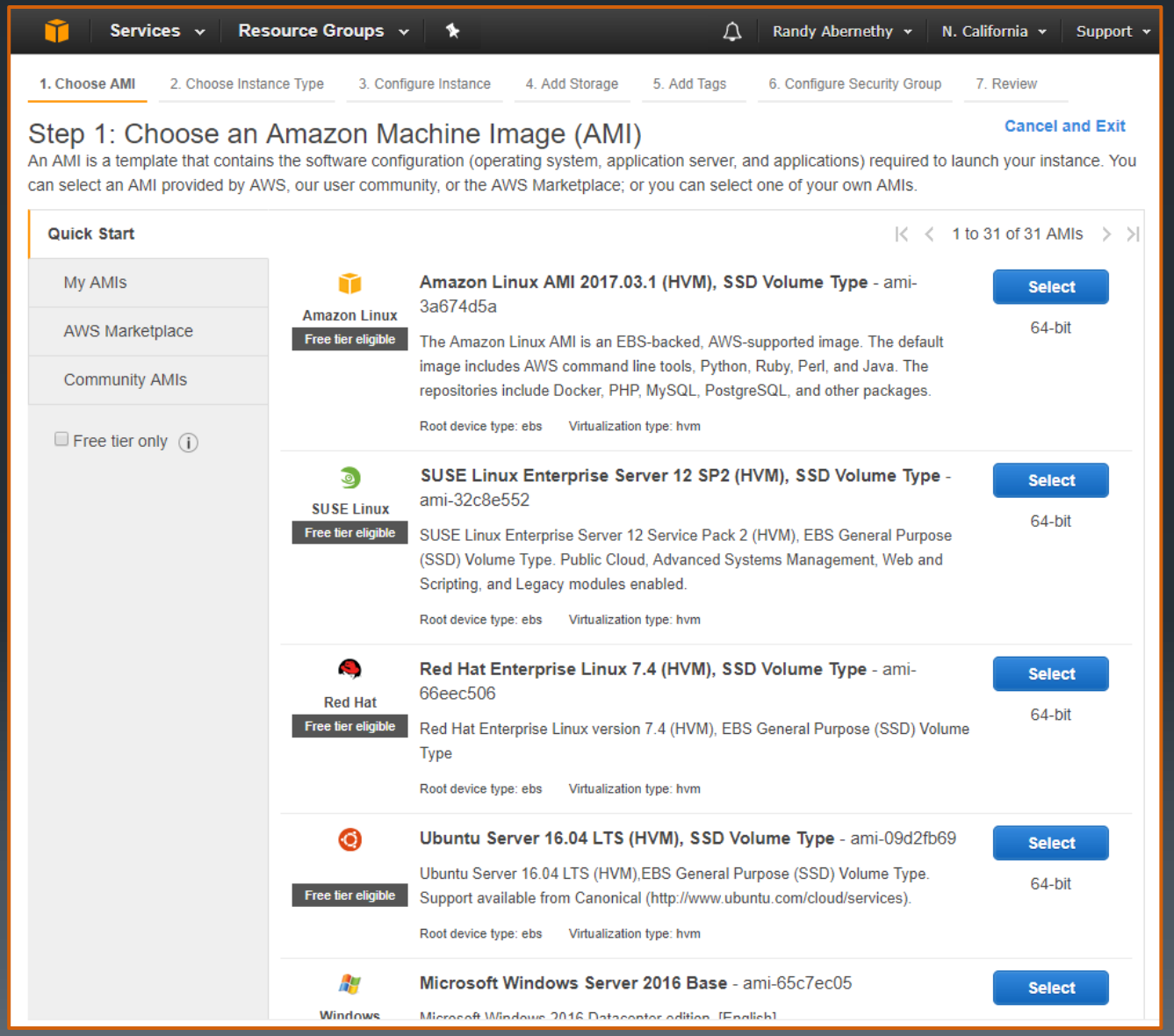
Elastic Compute 2 (EC2)

- Amazon Elastic Compute Cloud (Amazon EC2)
  - A web service that provides secure, resizable compute capacity in the cloud in the form of virtual machines (VMs)
- Designed to make web-scale cloud computing easier for developers
- EC2 offers a simple web service interface as well as support in the AWS console
- EC2 server instances boot in minutes
- Users pay for EC2 instances by the hour



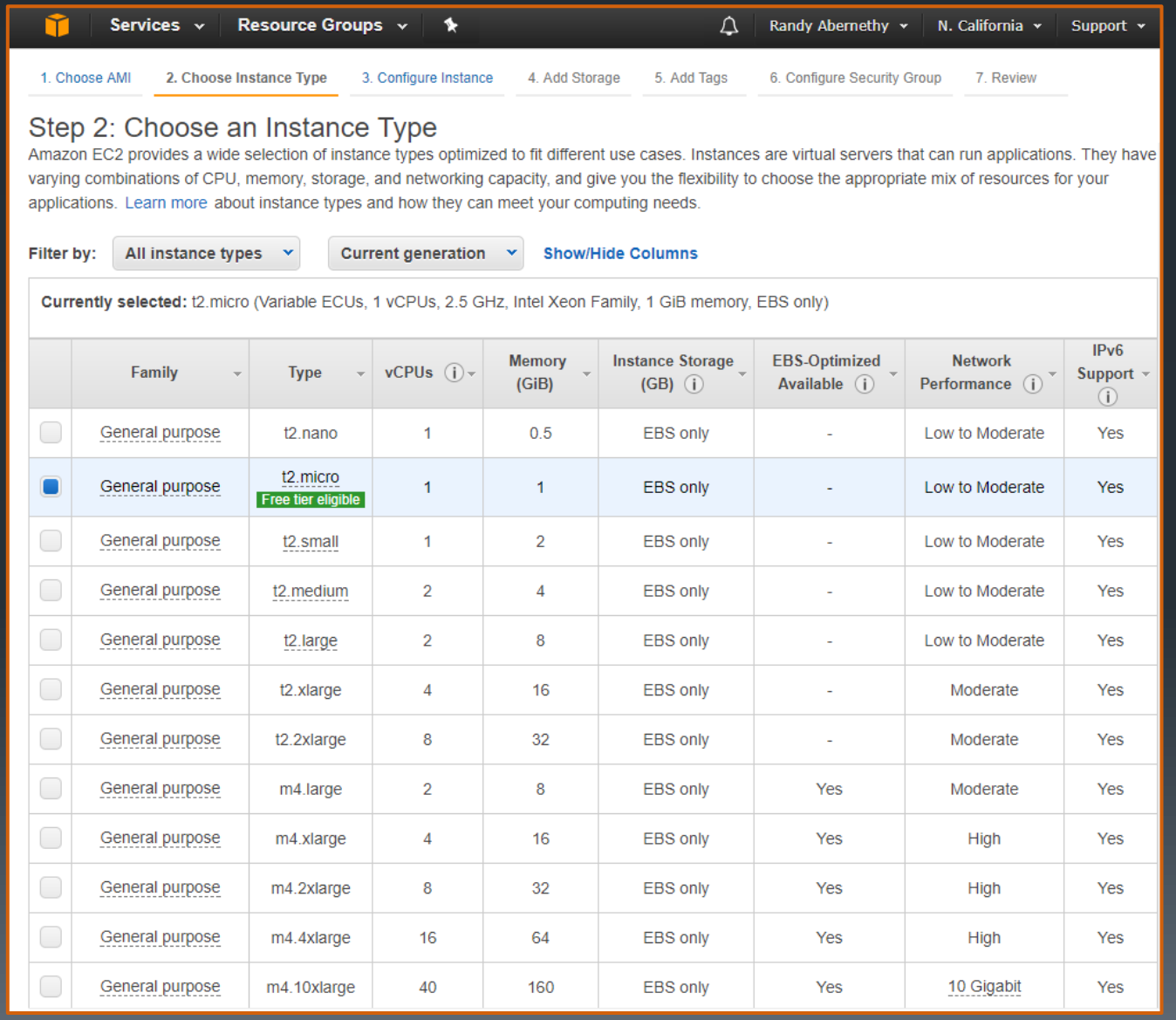
AMIs

- EC2 VMs require an image to boot
- EC2 can boot a wide range of systems stored in Amazon Machine Image (AMI) format
- Some images are free
  - Amazon Linux
  - Ubuntu
  - Suse
  - Centos
- Some images require a license
  - Windows
  - Red Hat EL
- The AWS Marketplace offers additional commercial images
- The Community AMI repository offers a wide range of additional free AMIs



Instance Types

- Instance types (called Flavors on some systems) define the size of the Virtual Machine deployed
- There are several type families
  - General Purpose
  - Compute Optimized
  - GPU Instances
  - Memory Optimized
  - Storage Optimized



Services | Resource Groups | Randy Abernethy | N. California | Support

1. Choose AMI | 2. Choose Instance Type | 3. Configure Instance | 4. Add Storage | 5. Add Tags | 6. Configure Security Group | 7. Review

Step 2: Choose an Instance Type

Amazon EC2 provides a wide selection of instance types optimized to fit different use cases. Instances are virtual servers that can run applications. They have varying combinations of CPU, memory, storage, and networking capacity, and give you the flexibility to choose the appropriate mix of resources for your applications. Learn more about instance types and how they can meet your computing needs.

Filter by: All instance types | Current generation | Show/Hide Columns

Currently selected: t2.micro (Variable ECUs, 1 vCPUs, 2.5 GHz, Intel Xeon Family, 1 GiB memory, EBS only)

	Family	Type	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance	IPv6 Support
☐	General purpose	t2.nano	1	0.5	EBS only	-	Low to Moderate	Yes
☑	General purpose	t2.micro Free tier eligible	1	1	EBS only	-	Low to Moderate	Yes
☐	General purpose	t2.small	1	2	EBS only	-	Low to Moderate	Yes
☐	General purpose	t2.medium	2	4	EBS only	-	Low to Moderate	Yes
☐	General purpose	t2.large	2	8	EBS only	-	Low to Moderate	Yes
☐	General purpose	t2.xlarge	4	16	EBS only	-	Moderate	Yes
☐	General purpose	t2.2xlarge	8	32	EBS only	-	Moderate	Yes
☐	General purpose	m4.large	2	8	EBS only	Yes	Moderate	Yes
☐	General purpose	m4.xlarge	4	16	EBS only	Yes	High	Yes
☐	General purpose	m4.2xlarge	8	32	EBS only	Yes	High	Yes
☐	General purpose	m4.4xlarge	16	64	EBS only	Yes	High	Yes
☐	General purpose	m4.10xlarge	40	160	EBS only	Yes	10 Gigabit	Yes

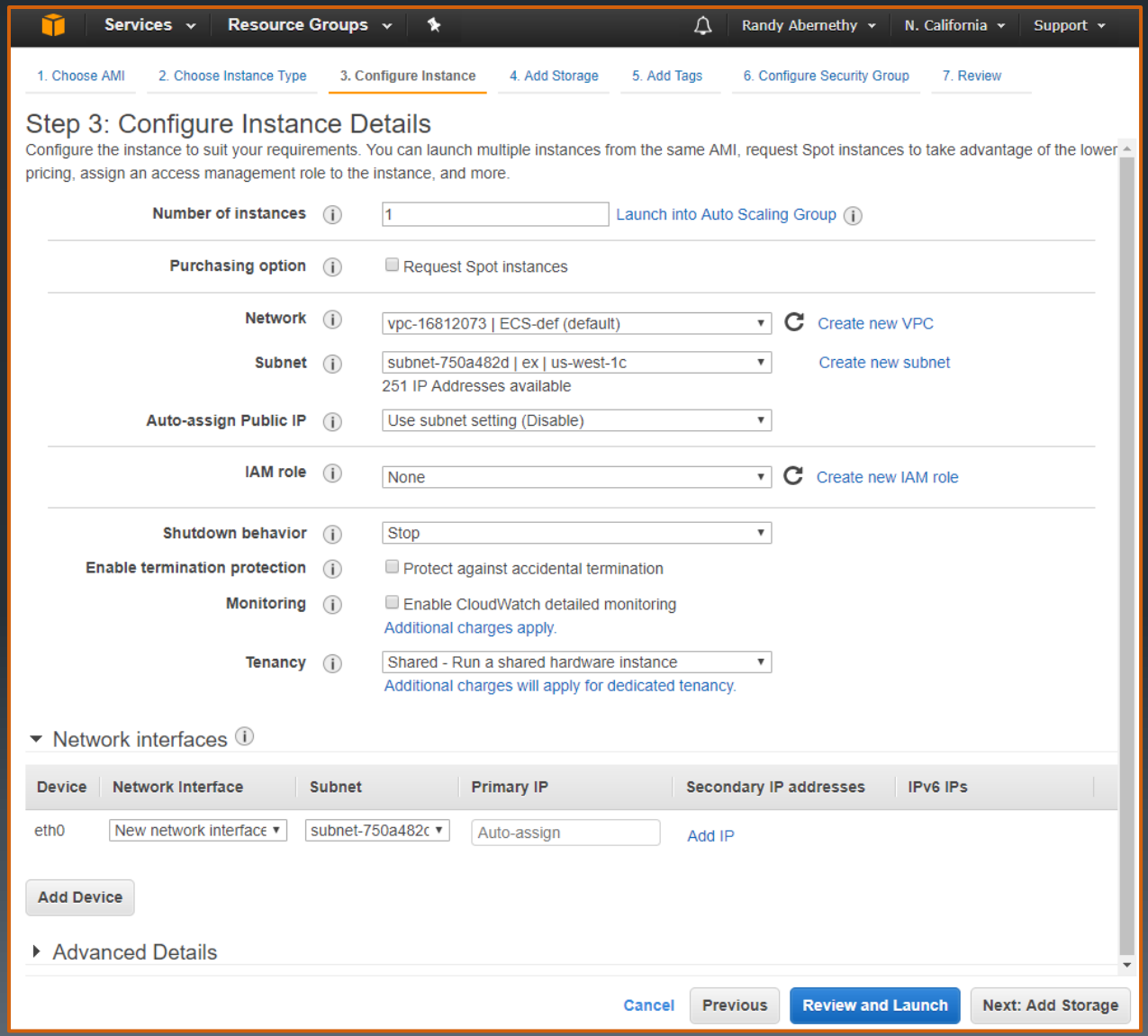
Instance Pricing

- EC2 instances are billed per hour based on type and OS
  - This is called On-Demand pricing
  - A typical 8 vCPU EC2 instance will cost < $10 per day
- Spot pricing allows you to bid on spare EC2 capacity for up to 90% off the On-Demand price
  - Failed bids do not create instances
  - Only the highest bidder wins in a spot auction
- Reserved instances are also much cheaper than On-Demand instances (up to 75% off)
  - Reservations must be paid for whether used or not
  - Reservations require a time commitment (months or years)

	vCPU	ECU	Memory (GiB)	Instance Storage (GB)	Linux/UNIX Usage
General Purpose - Current Generation					
t3.nano	2	Variable	0.5 GiB	EBS Only	$0.0062 per Hour
t3.micro	2	Variable	1 GiB	EBS Only	$0.0124 per Hour
t3.small	2	Variable	2 GiB	EBS Only	$0.0248 per Hour
t3.medium	2	Variable	4 GiB	EBS Only	$0.0496 per Hour
t3.large	2	Variable	8 GiB	EBS Only	$0.0992 per Hour
t3.xlarge	4	Variable	16 GiB	EBS Only	$0.1984 per Hour
t3.2xlarge	8	Variable	32 GiB	EBS Only	$0.3968 per Hour
t3a.nano	2	Variable	0.5 GiB	EBS Only	$0.0056 per Hour
t3a.micro	2	Variable	1 GiB	EBS Only	$0.0112 per Hour
t3a.small	2	Variable	2 GiB	EBS Only	$0.0223 per Hour
t3a.medium	2	Variable	4 GiB	EBS Only	$0.0446 per Hour
t3a.large	2	Variable	8 GiB	EBS Only	$0.0893 per Hour
t3a.xlarge	4	Variable	16 GiB	EBS Only	$0.1786 per Hour
t3a.2xlarge	8	Variable	32 GiB	EBS Only	$0.3571 per Hour
t2.nano	1	Variable	0.5 GiB	EBS Only	$0.0069 per Hour

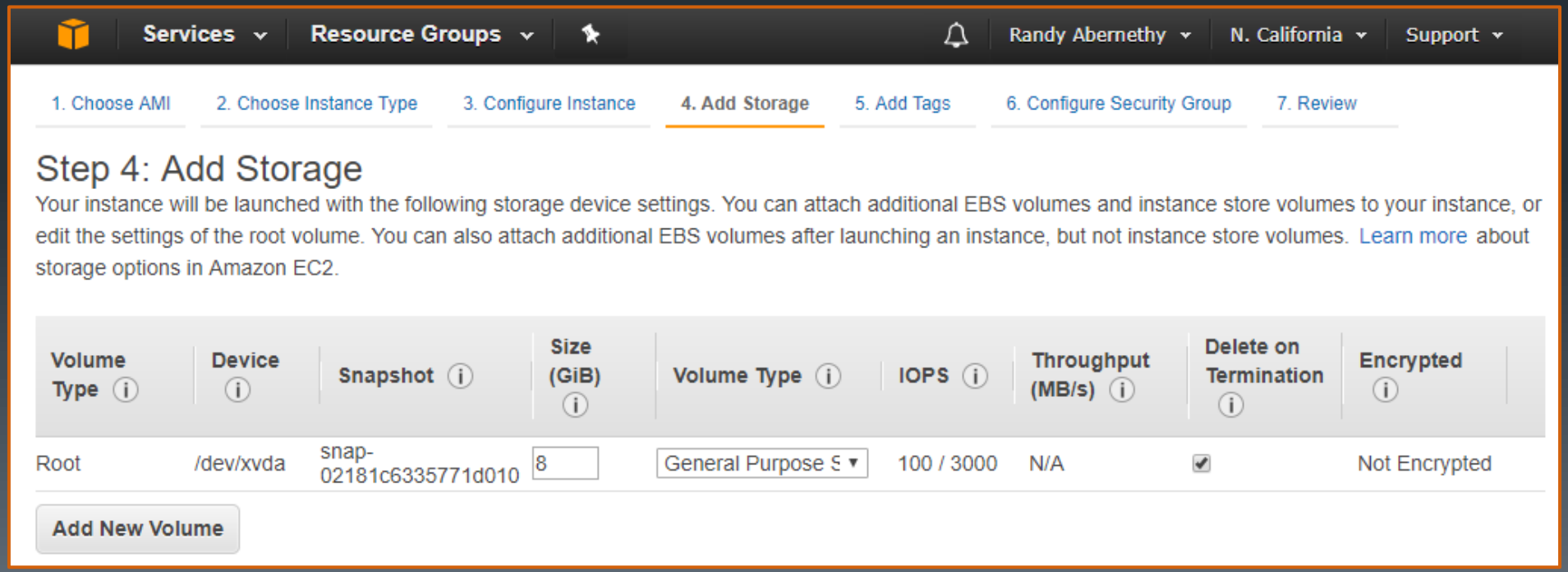
Instance Configuration

- EC2 instances can be created in any of the user's VPCs
  - Subnets and IPs can be assigned
  - Additional network interfaces can be defined
- IAM roles can be assigned to the host, giving it implicit access to other services
- Dedicated instances can be allocated to ensure only instances controlled by the user run on the underlying hardware



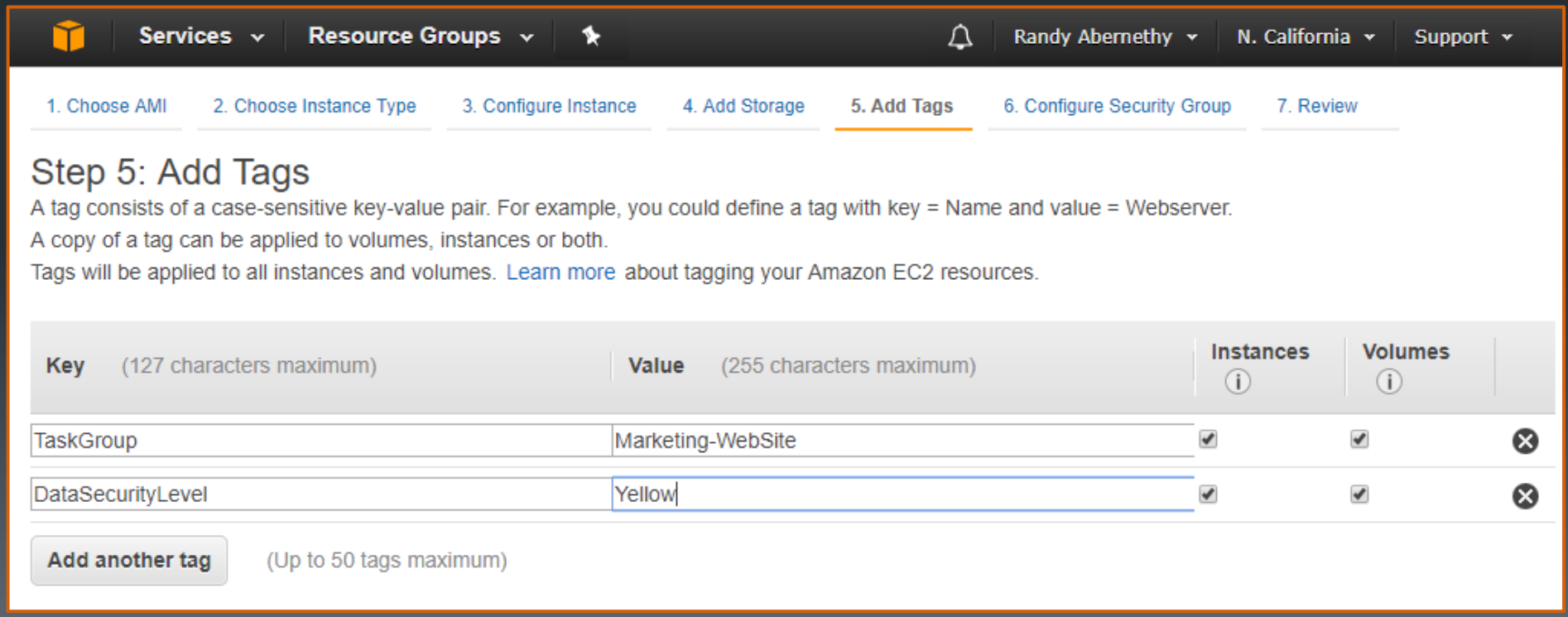
Instance Storage

- Compute instances must be assigned disk volumes
- Volume Types
  - Root – hosts the bootable operating system copied from the AMI
    - Some AMIs are backed by Amazon EC2 instance store (S3 based)
      - These root volumes are ephemeral (local) and copied from an S3 AMI template
    - Some AMIs are backed by Amazon EBS (snapshot based)
      - These are recommend because they launch faster and use persistent storage
  - Additional instance (ephemeral) and EBS volumes can be added



Tags

- Tags enable you to categorize your AWS resources
  - By purpose, owner, environment, criticality, etc.
- You identify a specific resource based on the tags you've assigned
- Tags consist of a key and an optional value that you define
- You should devise a set of tags that meet your needs for each resource type
  - Using consistent tags makes it easier to manage resources
  - Billing can be broken down by resource tags
  - You can search and filter resources based on their tags

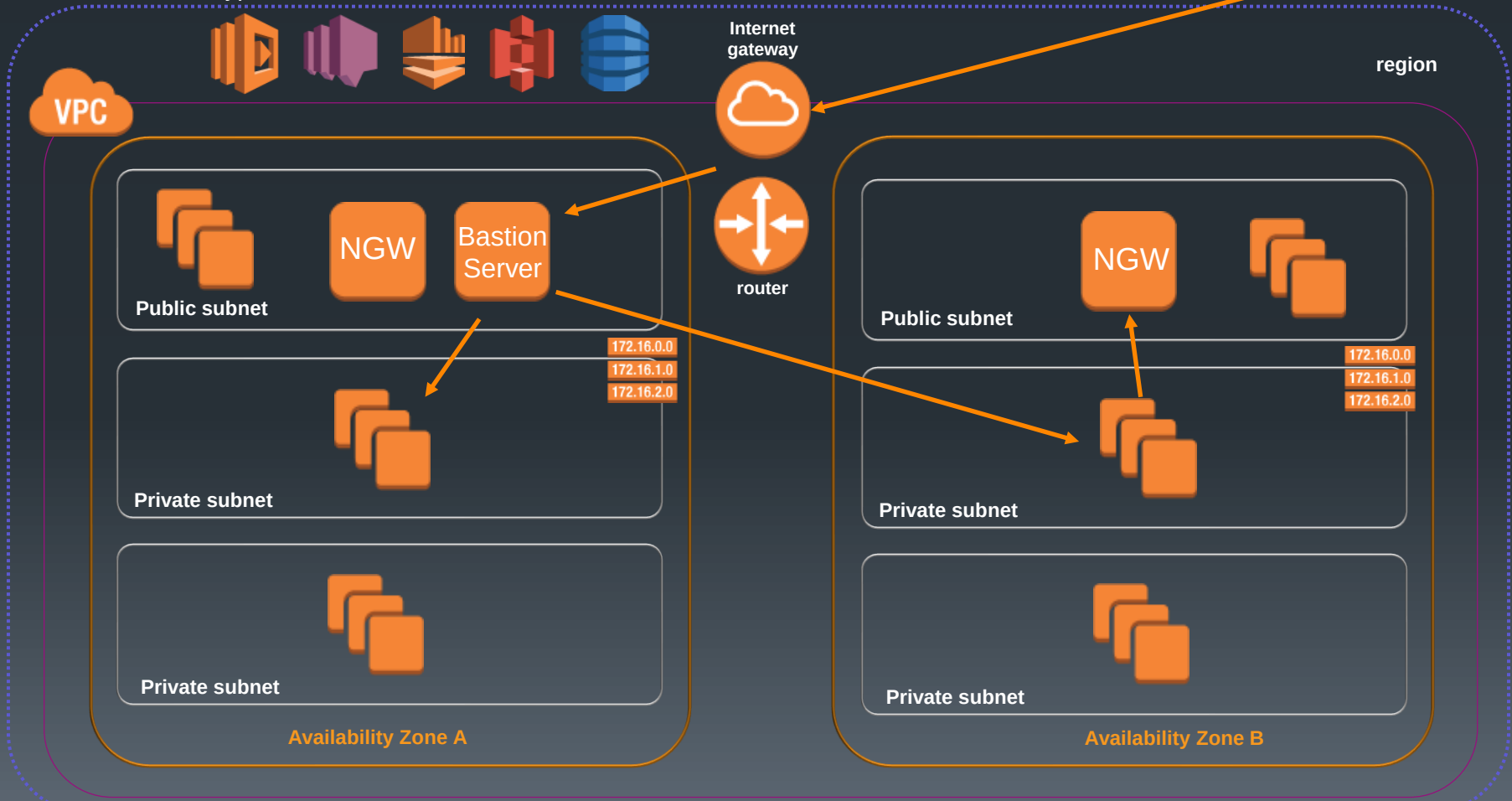


Amazon VPC

- Virtual Private Clouds (VPCs)
  - Allow customers to cut out a private section of the cloud to launch services in
- VPCs use a virtual network topology that is similar to a traditional network
  - Specify IP address ranges for private and public subnets
  - Administering inbound and outbound access using network access control lists
  - Attaching multiple virtual network interfaces
  - Bridging the VPC with on-site IT infrastructure using a VPN to extend existing security and management policies
  - Storing data in the Amazon S3 storage service and setting access permissions

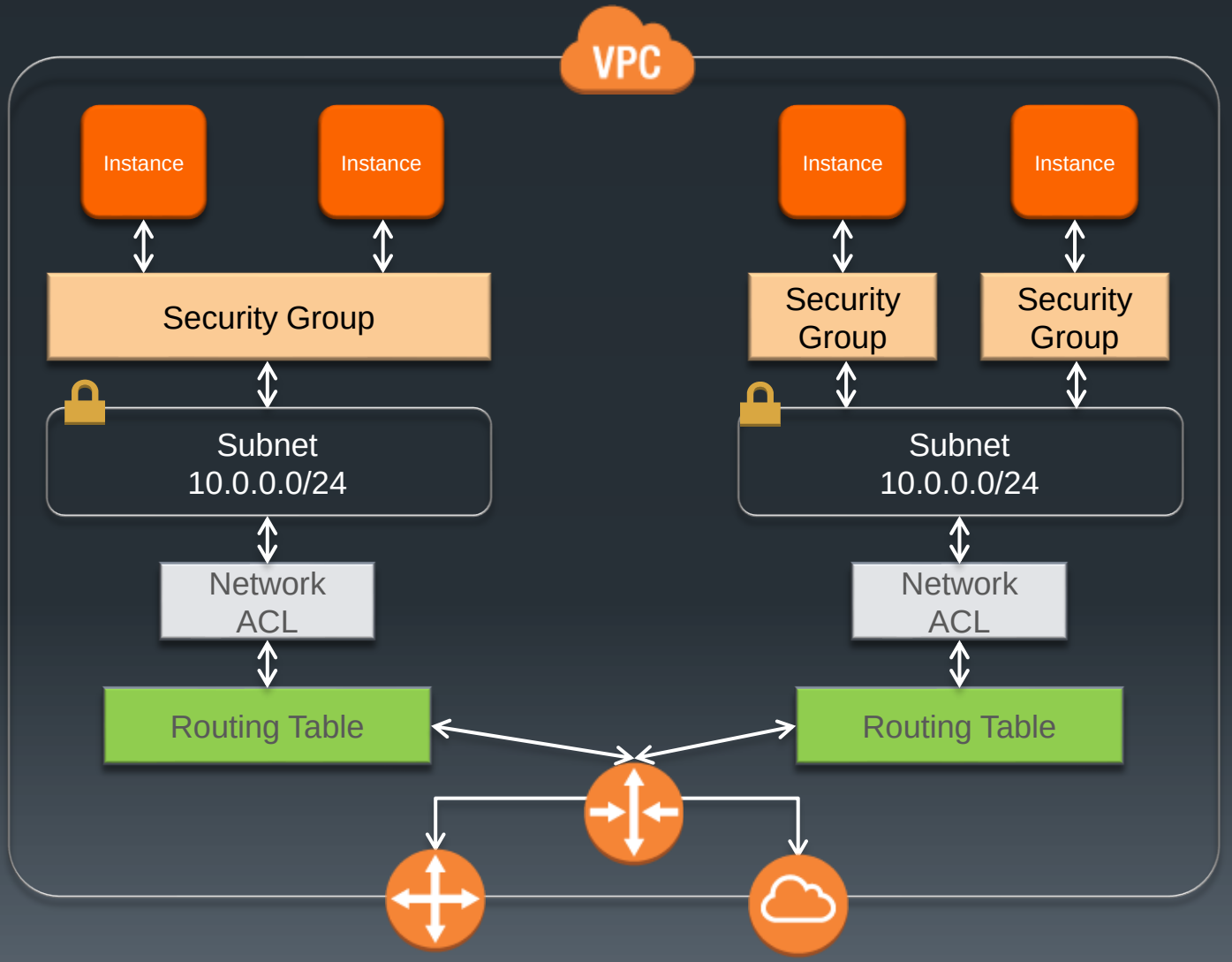
Amazon VPC

- A variety of connectivity options exist for Amazon VPC
  - Public subnets – launch instances into a publicly accessible subnet where they can send and receive traffic from the Internet
  - Private subnets – used for instances that you do not want to be directly addressable from the Internet
  - Network Address Translation – can access the Internet without exposing their private IP address by routing their traffic through a Network Address Translation (NAT) instance in a public subnet
  - VPN – traffic to/from instances in the VPC is routed to the corporate data center over a standard, encrypted IPsec hardware VPN connection



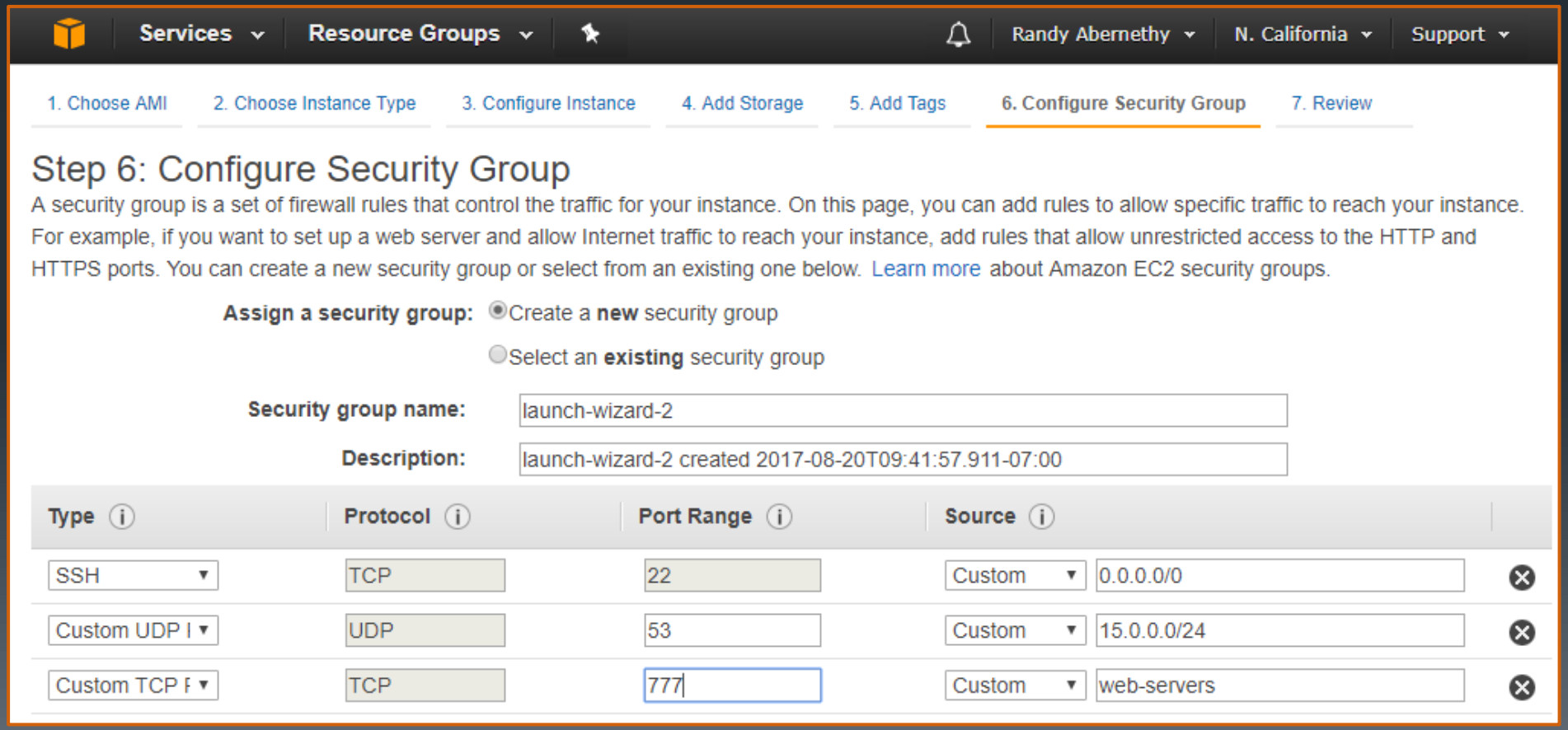
Amazon VPC SGs

- Amazon VPC provides advanced security features such as security groups (SGs)
- Filtering at the instance level
- Filtering at the subnet level
- Filtering at the Amazon S3 level (e.g. only accessible from instances in your VPC)



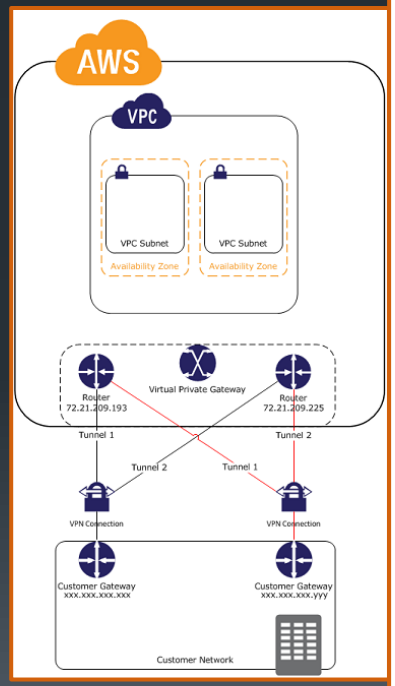
Security Groups

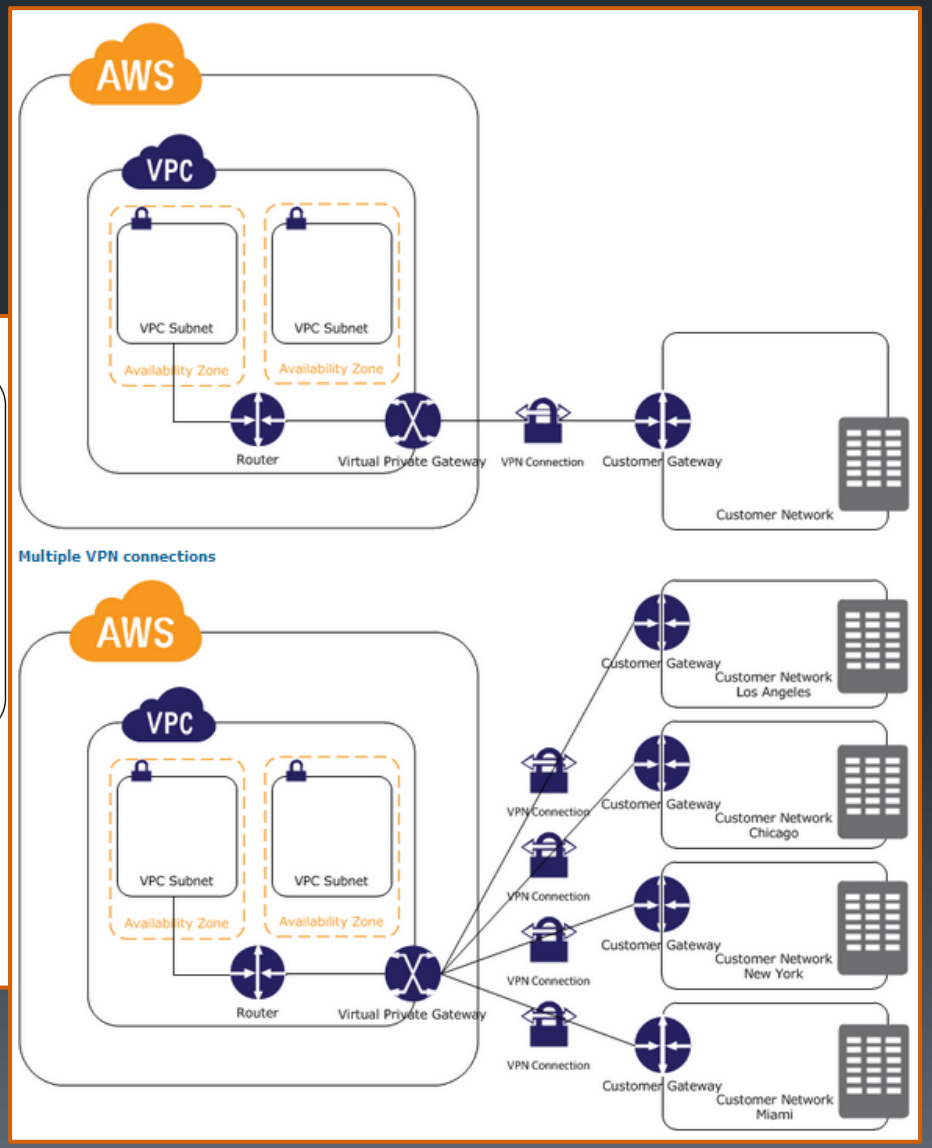
- Security groups define firewall rules for the compute instance
- You can assign one or more instances to a single security group
- Security group rules control which hosts can access which ports of the instance
  - Sources can be limited by IP, subnet or security group



Securely Bridging to a VPC

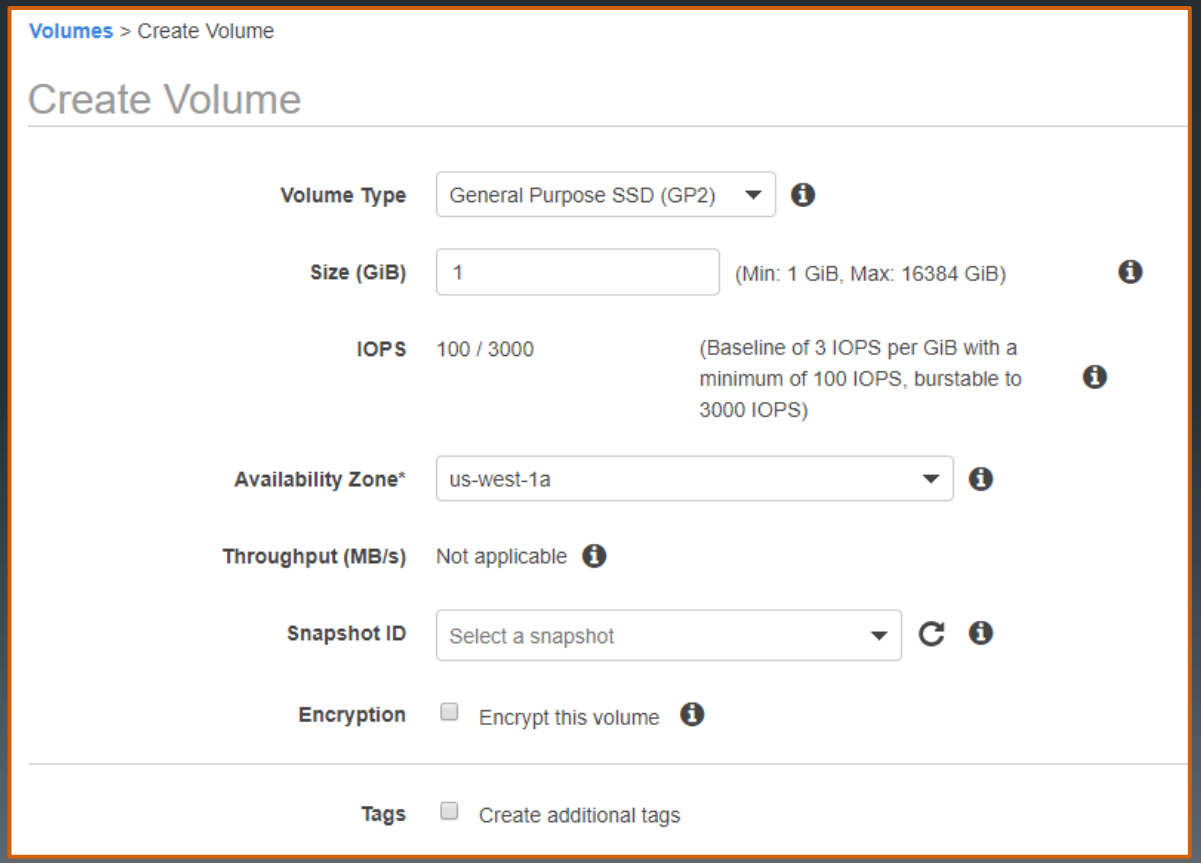
- VPCs are typically integrated with on premises systems via VPN
  - In many cases dedicated WAN links can also be arranged
- A VPN connection can typically be configured in one of two ways:
  - Border Gateway Protocol (BGP) with dynamic routing
  - Static routing
- A second VPN connection can be configured for redundancy
  - Using BGP and Hot Standby Router Protocol (HSRP) enables automatic failover and enables maintenance without losing connectivity
- All routers involved should be secured as always
  - Only allow dynamic BGP traffic between peer, etc.





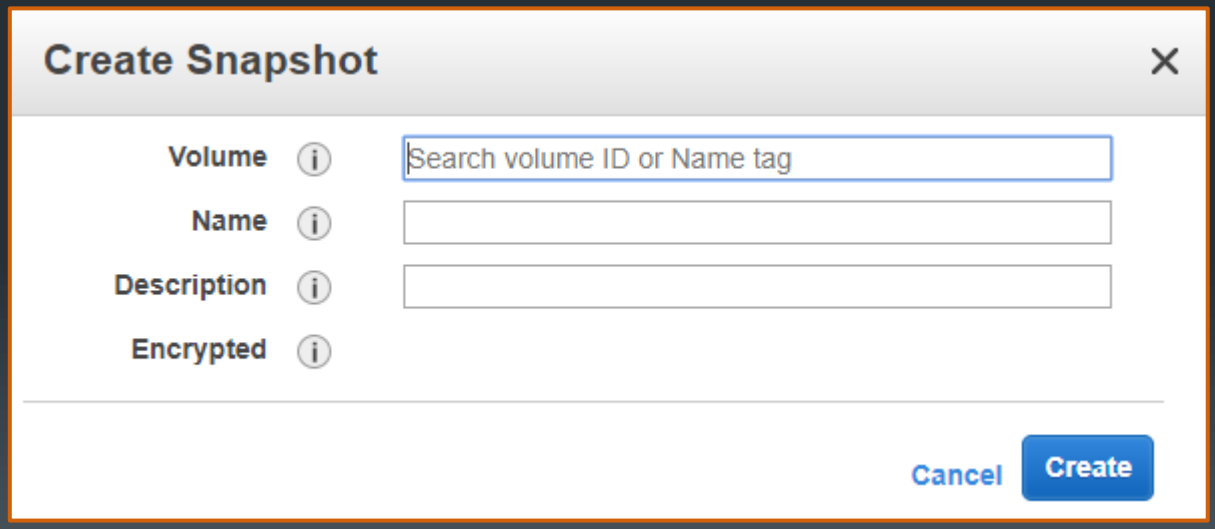
Elastic Block Storage (EBS)

- Amazon Elastic Block Store (Amazon EBS)
  - Provides persistent block storage volumes for use with Amazon EC2
- Each Amazon EBS volume is automatically replicated within its Availability Zone
- EBS volumes offer consistent low-latency performance
- EBS volumes can be created independent of EC2 instances
  - Mounted and unmounted from various instances as required



Snapshots

- EBS snapshots allow you to capture the state of an EBS volume as a point in time
- Snapshots are backed by S3 (11 9's of reliability!!)
  - EBS SLA is only 99.95%



Summary

- AWS offers perhaps the most fully developed range of IaaS services in the public cloud space
  - EC2 supports compute
  - S3 and EBS support storage
  - VPC supports networking

Lab 4

- AWS VPC setup and EC2 application deployment
  - Use the AWS CLI to set up a Virtual Private Cloud
  - Deploy an application to instances hosted on AWS Elastic Compute Cloud (EC2)

Day 2 – Cloud Native Solutions

5. Cloud native systems overview
6. CaaS solutions
7. Cloud data stores
8. API gateways

5: Cloud native systems overview

Objectives

- Examine how cloud native technologies are changing application design patterns
- Present the benefits and challenges of microservice-based applications
- Examine container technology and clarify the differences between containers and VMs
- Explain how standards bodies enable transformations to the cloud

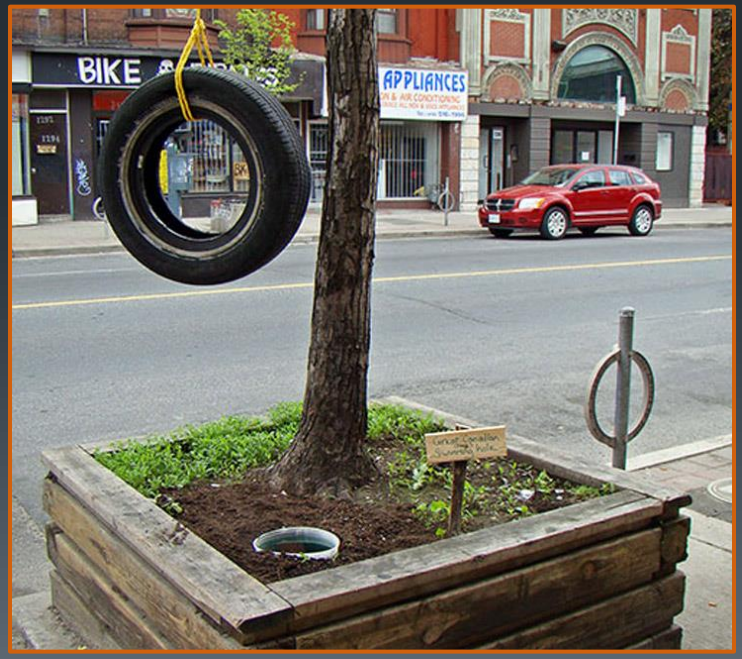
What is Cloud Native?

- Cloud native applications are:
  - Microservice Oriented
    - Applications are decomposed into microservices
  - Container Packaged
    - Each service is packaged within its own container
  - Dynamically Orchestrated
    - Containers are dynamically orchestrating to optimize resource utilization and maximize automation
- Cloud native computing uses an open source software stack, often combined with IaaS, PaaS and SaaS services to run cloud native applications

Why Cloud Native?

- Key motivation:
  - For enterprise: speed of innovation
  - For hyperscale: scalability
- To realize the vision of Agile
  - Just in time delivery of reliable quality
  - The unit of deployment must match the team work product to avoid excess coordination overhead
  - This means applications must be decomposed into team sized services
- No one cares about the operating system
  - Companies want to develop and deploy new innovative applications rapidly
  - The focus is on building and deploying applications not operating systems
  - Infrastructure is commoditized and consumed as a product in the cloud era
    - DevOps of old is dead

-- or --

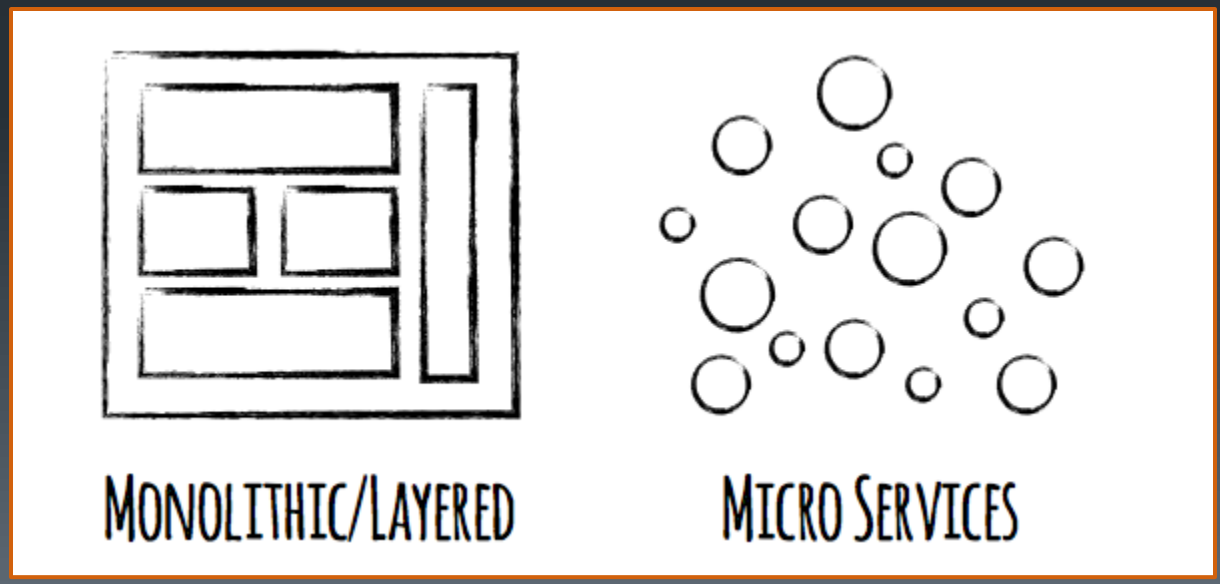


Cloud Native Enablers

- Agile methodologies
- Test automation tooling and methodologies
- Continuous integration and continuous delivery (CI/CD) technology
- Public cloud
  - Pay as you go, self service, infrastructure as code, virtually infinite scale, opaque infrastructure
- Cluster native databases, aka. NoSQL, Cassandra, Couchbase, …
- Cluster native messaging, Kafka, NATS, …
- High speed networks (40 and 100gb are current data center standards)
- Container technology, Docker, Containerd, OCI, …
- Software defined networking (SDN), CNI, …
  - Policy based security
  - Young but workable for many
- Orchestration technology, Kubernetes, Mesos/Marathon, …
  - Young but workable for many
- Software defined storage (SDS)
  - Work in progress, commercial solutions and bridges from the traditional virtualization space are the mainstays at present

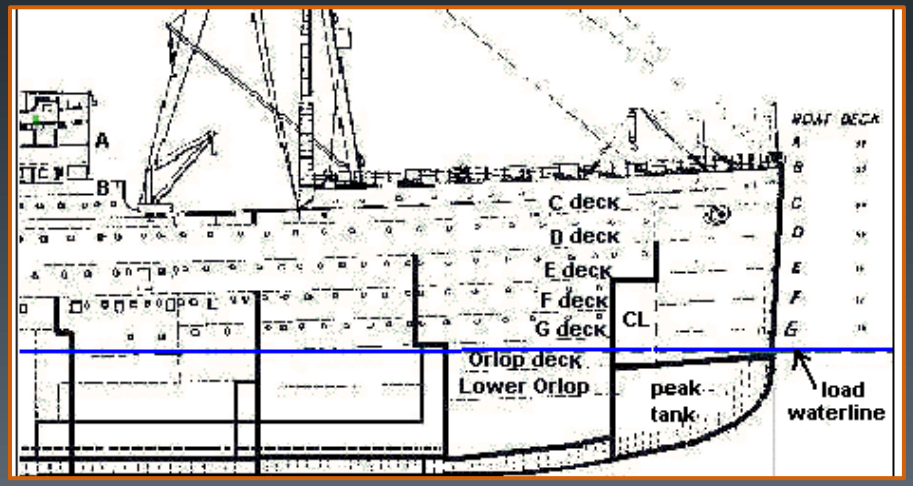
Microservice Attributes

- Small crisply defined services
- Decomposed by business/problem domain, not technology
- Well defined interfaces, completely abstract implementations
- Light-weight, technology-agnostic inter-service communications
- Polyglot friendly
- Changes are single service and incremental
- Services are atomically deployable and designed to be replaced
- Services are not changed they are replaced (and can be rolled back)
- Services are loosely coupled (discovering each other dynamically if need be)
- Single responsibility (high cohesion)
- Persistent state is encapsulated behind service interfaces and managed using cluster friendly network distributed schemes
- State is decomposed by service not centralized or shared



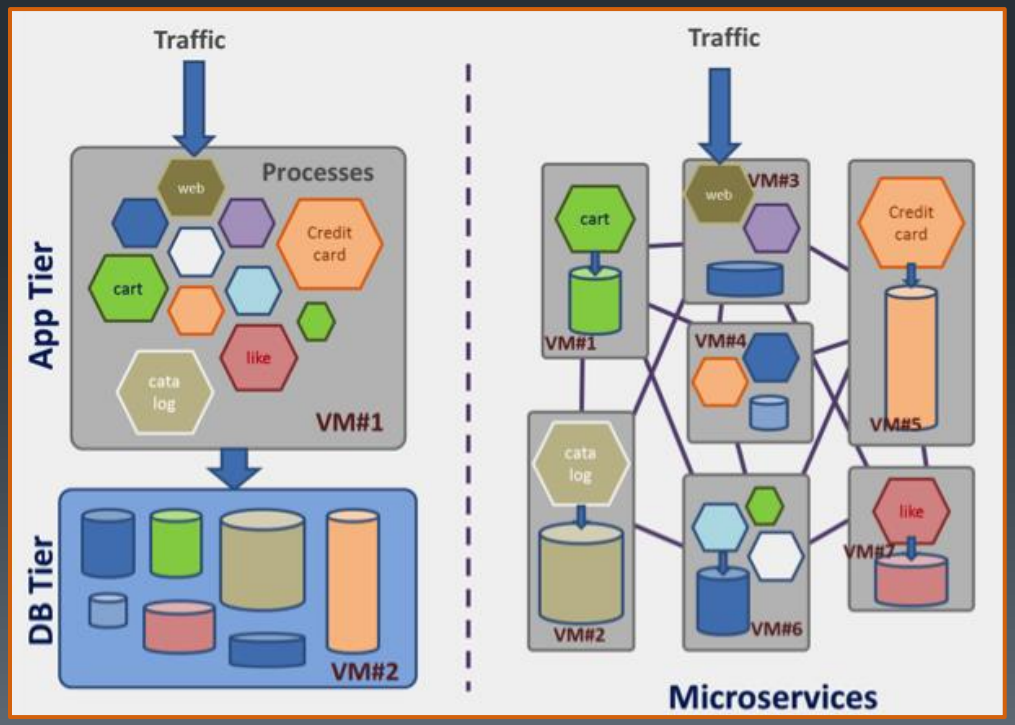
Microservices Benefits

- Deliver software faster
  - A change to a microservice typically requires only the deployment of the microservice
  - Monolithic applications require large deployments for small changes
- Fine grained scaling
  - Microservices allow you to scale only the parts that need to be scaled
- Flexibility to embrace newer technologies
  - Risk is a key barrier to adopting new technologies
  - With a monolithic application a new programming language, database, or framework will impact a large amount of the system
  - With microservices this can be a small experiment and rolled back easily and without large cost if it fails
- Organizational alignment
  - Each team can own one or more services, reducing coordination overhead
- Respond faster to change
  - Microservices are easier to rewrite wholesale, making it possible to try different ideas and track changes in business structure and patterns
- Composability
  - Microservices are atomic enough to be easily consumed by a range of clients for different purposes
- Resilience
  - When a single microservice fails only a small part of the application is inoperable
  - Enables microservice applications to operate more effectively in the face of partial failure
    - Resilience engineering bulkhead concept



Microservice Challenges

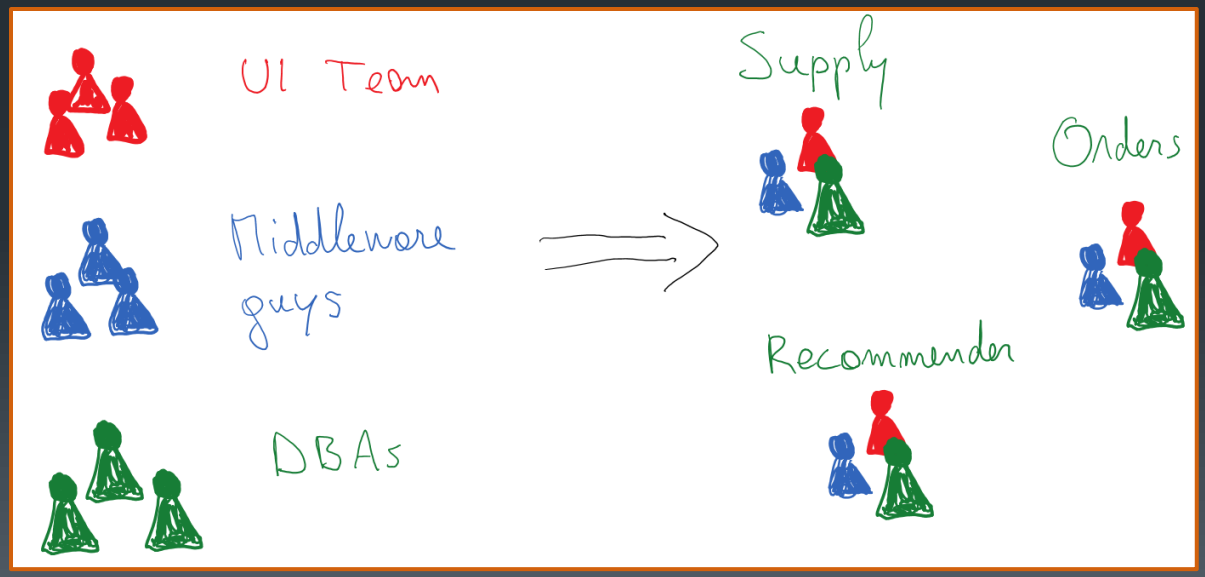
- Explosion in the number of processes to manage
  - Requires reliable deployment solutions and automated dynamic orchestration to manage
- New and not well understood by many teams
  - Easy to slip into old habits making things worse not better
- Heavy network utilization and increased latency
  - Microservices communicate through out-of-process network interfaces
  - Slower, perhaps much slower, than in-process function calls, though mitigated by async communications in some cases
- Small to medium applications may be harder to deploy and manage
  - No transactions, no single integration database
- Integration is no longer anyone in development's problem
  - Isolating teams within a single process may lead to poor integration practices in development making life harder for test and production



Changing the Teams is Harder than Changing the Tech

- Evolution at 3 major tech firms:
- eBay
  - 5th generation today
    - Monolithic Perl
    - Monolithic C++
    - Java
    - Microservices
- Twitter
  - 3rd generation today
    - Monolithic Rails
    - JS / Rails / Scala
    - Microservices
- Amazon
  - Nth generation today
    - Monolithic C++
    - Perl / C++
    - Java / Scala
    - Microservices

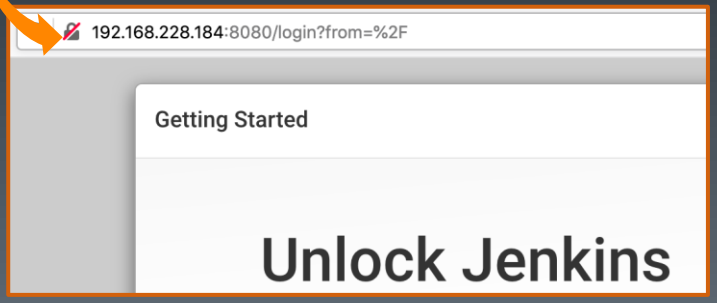
Very small autonomous teams achieve great things



What is a Container?

- Lightweight Linux environment
  - More recently, lightweight Windows environments as well
- Encapsulated, deployable, runnable
  - The new way to package applications
- Microservice-centric
  - one atomic service per container
- Made widely popular by Docker
  - Docker, Inc. provides a container platform including systems for publishing and sharing containers
  - Docker has been the dominant mover in this space but competition is growing and standards are evolving
  - Container technology predates and enables Docker
  - Containers are now standardized through the Open Container Initiative (OCI)
- Containers rely on integral features of the Linux Kernel
  - CGroups
  - Namespaces
  - Linux Bridge/IPTables/Capabilities/etc.

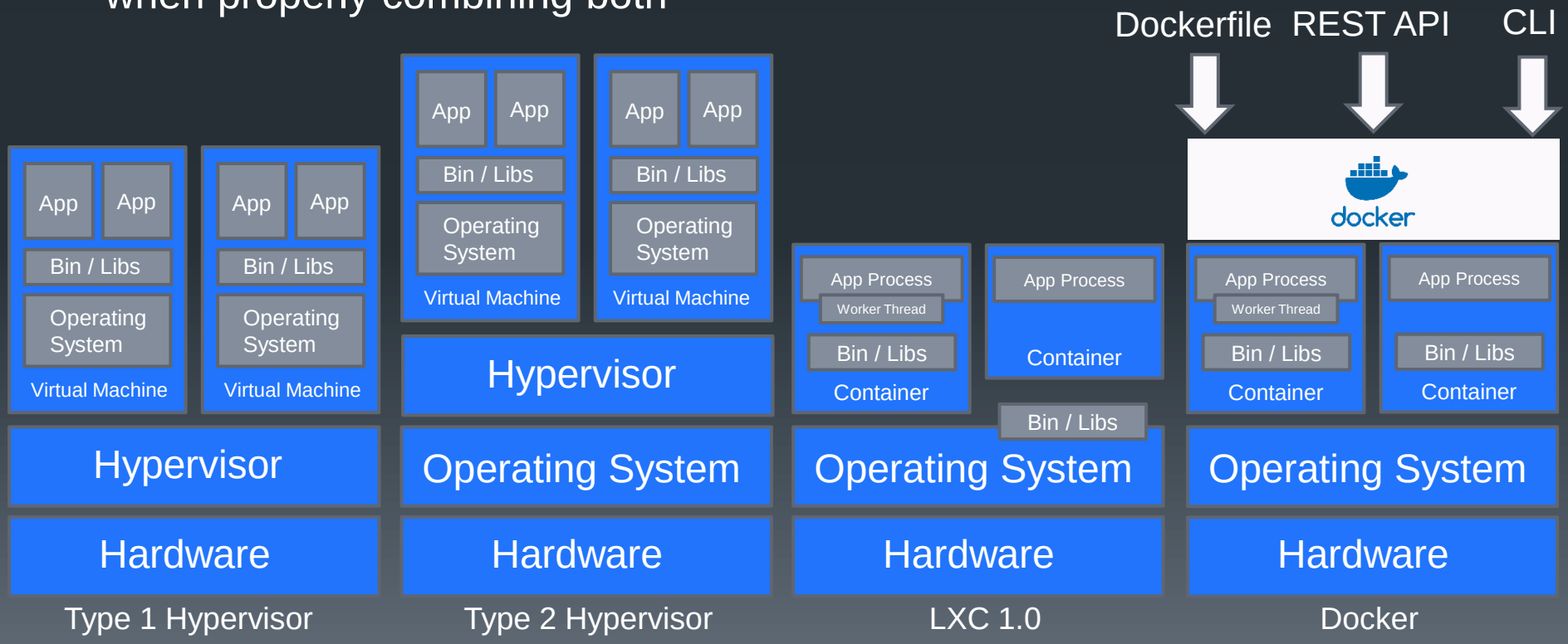
```
$ ip a | grep "global ens"
    inet 192.168.228.184/24 brd
192.168.228.255 scope global ens33

$ time docker run -p 8080:8080 -d jenkins
327ed4e2bf0269ec1ab41c9487d25435a700b0907
dc2b3bae2c74b607b411ae6

real        0m0.517s
user        0m0.028s
sys         0m0.020s
```



Machine vs. OS virtualization

- VM
  - A virtual machine for operating systems
- Container
  - A virtual operating system for applications
- These are not mutually exclusive and many environments become optimal when properly combining both

Containers supply only the executables and library interfaces necessary to mimic the application dependent aspects of a Linux distribution (Ubuntu, RHEL, SUSE, etc.), leveraging the fact that all Linux systems use the same underlying kernel



VM/Container Tech Convergence

- Intel working hard to empower container technology on Intel CPUs through Linux
  - Much like the VTx extensions support modern hypervisors
  - ClearLinux and Clear Containers
  - Joined the Kata Containers project
    - Intel will continue to maintain Intel Clear Containers 3.0 through the transition

- VMware VIC
  - vSphere Integrated Containers make vSphere look like Docker
  - Each container is run in a lightweight VM on Photon (VMware's lightweight Linux, about 60MB)

- RunV
  - Hypervisor-based runtime for OCI
  - Used with KVM and Xen

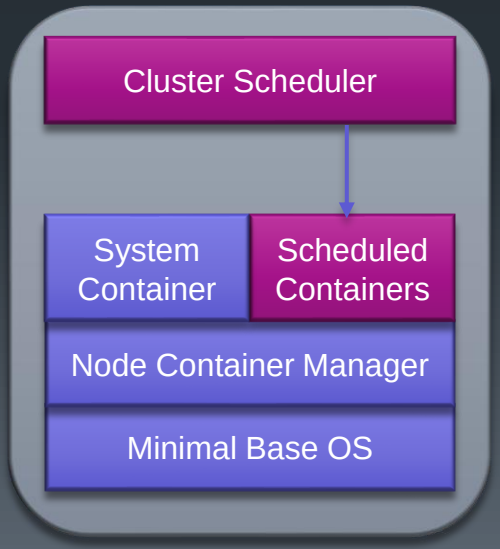
- LinuxKit
  - Suite of tools from Docker used to build small custom Linux kernels
  - Unikernel tech offshoot and integration options

- Kata Containers
  - Managed by OpenStack Foundation
  - Combines technology from Intel Clear Containers and Hyper runV
  - Designed to be hardware agnostic and compatible with the Open Container Initiative (OCI) specification

Package Management

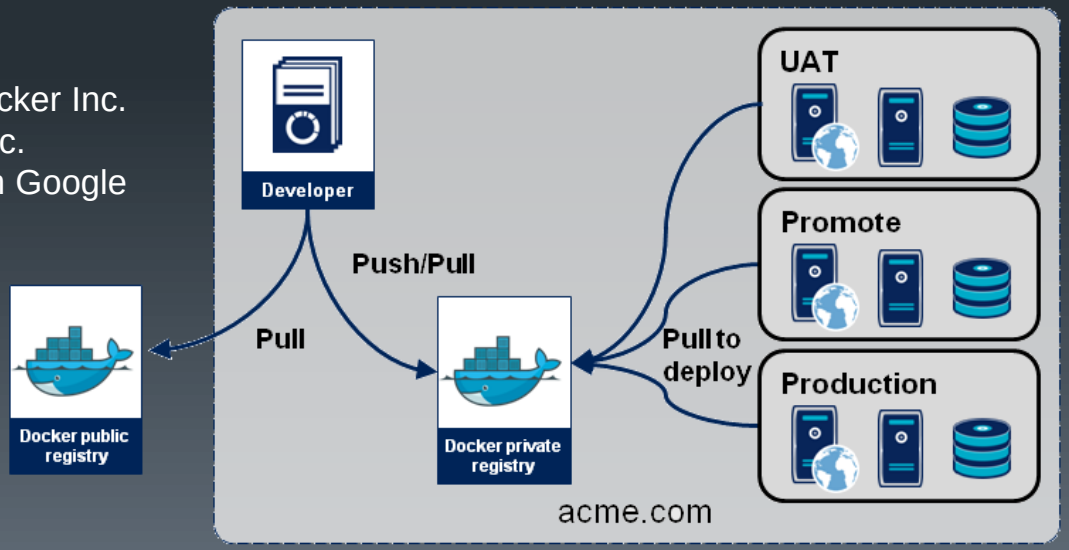
- The OLD: RPMs, DEBs and MSIs
  - Mutate the host
    - $ yum install
    - $ apt install
    - $ zypper install
  - Makes reliable deployment and dependency management nearly impossible
  - Are host OS specific
  - CM tools were an early effort to stabilize this but ultimately failed to eliminate some of the most damaging externalities
    - Puppet, Chef, Ansible, Salt

- The NEW: Containers
  - Docker images are immutable
  - Host OS agnostic
  - When no longer applicable they are disposed of and replaced wholesale
  - Require only basic Linux kernel features (3.10)
    - Causing an explosion in stripped down kernel offerings
      - Ubuntu Core
      - Red Hat/CentOS Atomic
      - CoreOS Container OS
      - VMware Photon
      - Windows Nano



Container Registries

- Registries are services that allow you to store and retrieve container images
  - Platform independent (unlike RPMs and DEBs)
  - Docker Hub is the central public registry
  - Companies can deploy their own registries
    - Using open source and commercial solutions
      - Docker Trusted Registry (formerly known as: Docker Hub Enterprise) from Docker Inc. (commercial product)
      - Quay Enterprise part of the Tectonic platform from CoreOS (commercial product)
      - Docker Registry from Docker Inc. (free open source) (https://github.com/docker/docker-registry)
      - Artifactory 3.4+ (http://www.jfrog.com/open-source/#os-arti)
      - Nexus 3 Milestone 5 (in preview) (https://issues.sonatype.org/browse/NEXUS-6166)
      - Harbor open source enterprise-class container registry (https://github.com/vmware/harbor)
    - Using hosted cloud solutions
      - Docker Hub (hub.docker.com) from Docker Inc.
      - Quay (quay.io), acquired by CoreOS Inc.
      - Google Container Registry (gcr.io) from Google
      - Amazon EC2 Container Registry (ECR) integrated with AWS EC2 Container Service (ECS)
      - Azure Container Registry (ACR) integrated with Azure Container Service (ACS)



Container Standardization

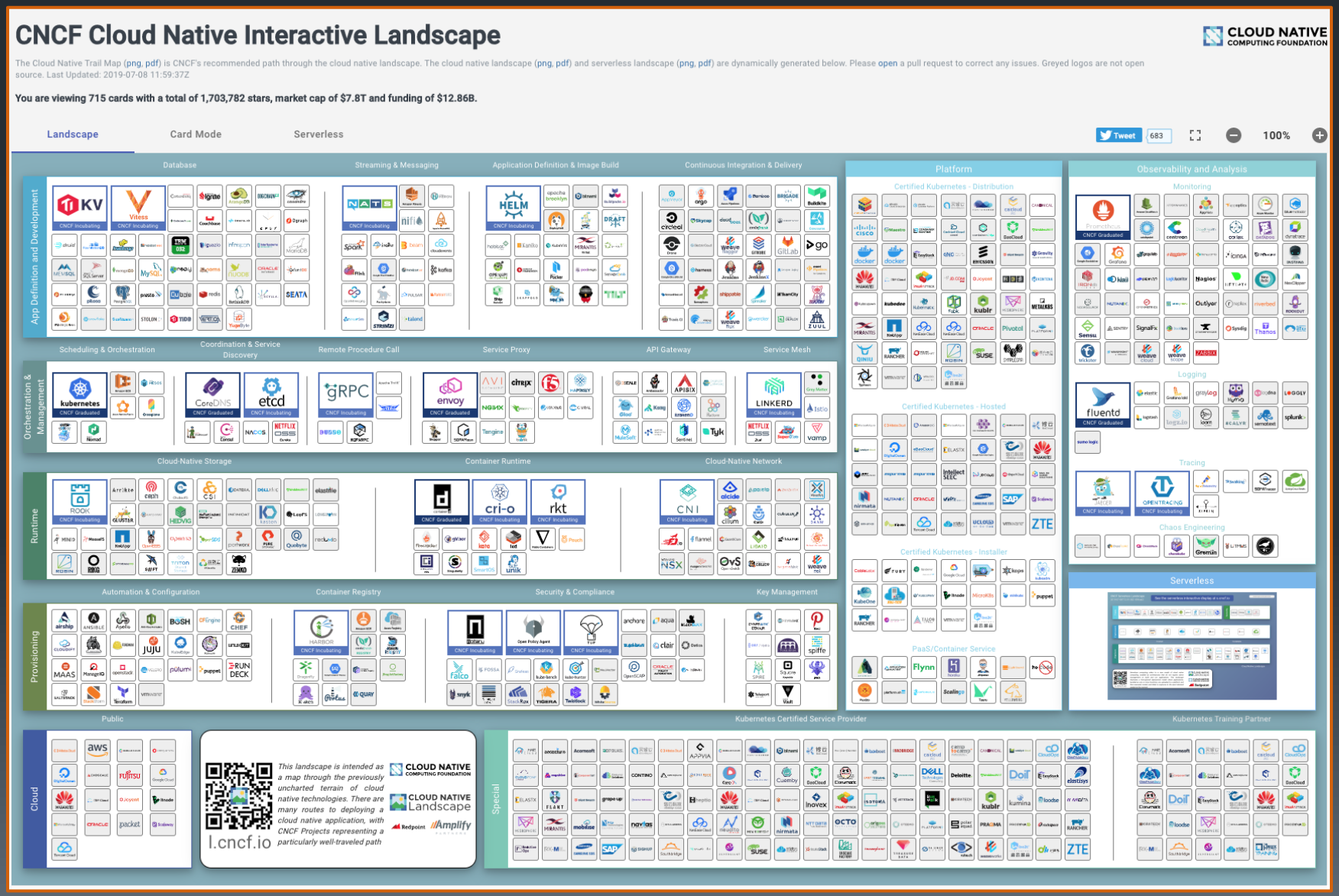
- Open Container Initiative (OCI) [circa 6/2015]
  - A standard that ensures any container can run on any machine or cloud computing service
- Run by the Linux Foundation
- Three specifications:
  - Runtime Specification (runtime-spec)
    - Specifies the configuration, execution environment, and lifecycle of a container
    - Released v1.0 July 19, 2017
  - Image Specification (image-spec)
    - An OCI implementation downloads an OCI Image, unpacks it, then runs it
    - Released v1.0 July 19, 2017
  - Distribution specification (distribution-spec)
    - Standardize container image distribution based on the specification for the Docker Registry HTTP API V2 protocol
    - Released v1.0 RC 0 February 14, 2019
- Docker donated the container format, runtime code (libcontainer) and specifications
  - runc was created as a libcontianer client, representing the lowest level tool for spawning a containerized process

The Open Container Initiative (OCI) is a lightweight, open governance structure (project), formed under the auspices of the Linux Foundation, for the express purpose of creating open industry standards around container formats and runtime. The OCI was launched on June 22nd 2015 by Docker, CoreOS and other leaders in the container industry

Cloud Native Computing Foundation

- Non-profit, foundation under the Linux Foundation (circa 2015)
- A response to the realization that all companies need to be software companies and that cloud native approaches allow IT and software to move faster, yet information and solutions are scarce commodities
- The CNCF seeks to foster an environment within which active discussion and evangelization of cloud native approaches can take place
- The CNCF also acts as a hosting body for cloud native open source projects
  - Providing governance, guidance, CI and community support
- The key CNCF conference is CNCFCon/KubeCon
  - Taking place in North America, Europe, and Asia annually
- Current projects:
  - Kubernetes, Prometheus, Fluentd, CoreDNS, Containerd, Rocket, CNI

Cloud Native Landscape



Summary

- Cloud native systems are:
  - Microservice oriented
  - Container packaged
  - Dynamically orchestrated
- Cloud native systems provide a “last mile” model that finally allows the realization of Agile
  - Microservices break applications into team sized chunks
  - Containers package them for reliable testing and deployment
  - Orchestration allows microservices to be deployed and scaled automatically and dynamically
- Cloud native systems can operate on top of IaaS VMs but do not require them
  - E.g. Kubernetes, Mesos, Swarm

Lab 5

- Cloud Native Systems Overview
  - Explore containerization with Docker
  - Show how Docker interacts with cloud services using AWS Elastic Container Service (ECS)

6: CaaS solutions

Objectives

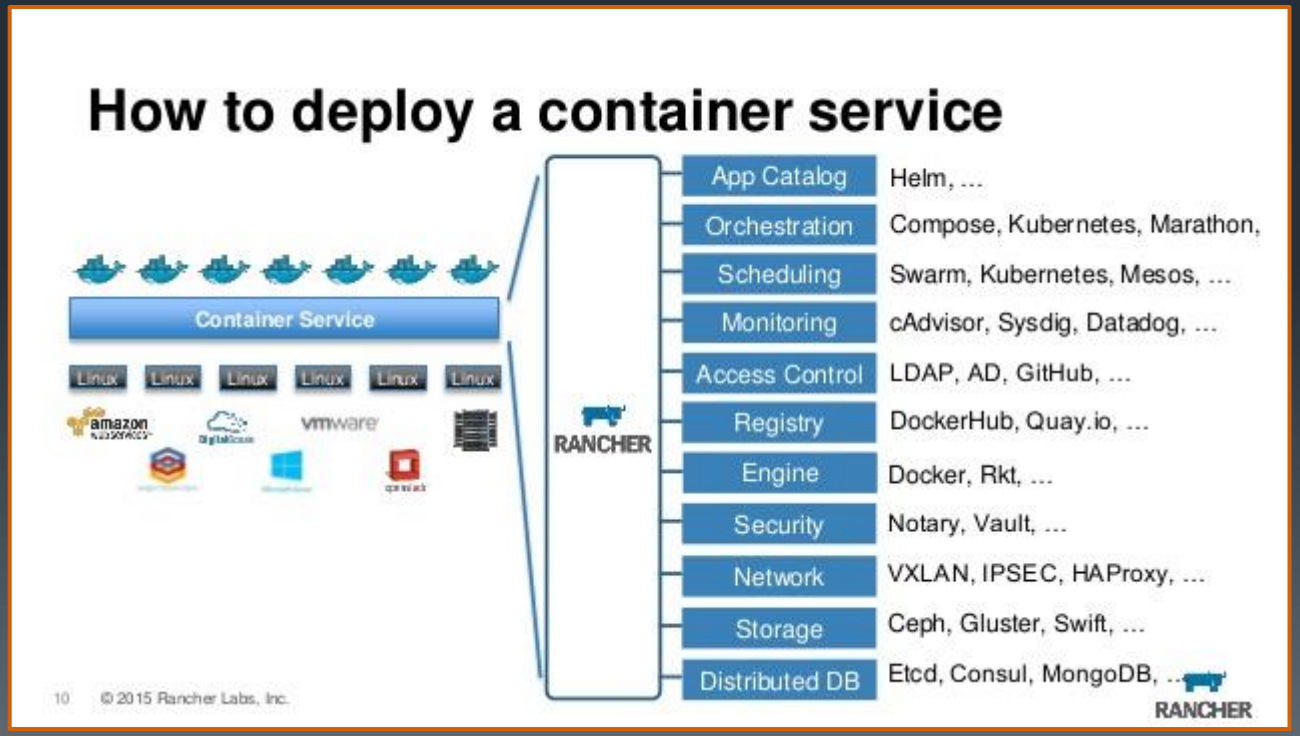
- Explain what functionality orchestration platforms provide to manage containers and their usage trends
- Show how container-based applications are distributed to CaaS clusters
- Examine AWS Elastic Container Service (ECS) and its components

After Containers, what's left?

- Orchestration!
  - Real cloud native applications are delivered in 10s of images and require 100s - 1000s of running containers
  - Registries manage image distribution
- Orchestration features:
  - Container Scheduling
    - Distributing containers to appropriate hosts
    - Host resource leveling
    - Availability zone diversity
    - Related container packaging and co-deployment
    - Affinity/Anit-affinity
  - Container Management
    - Monitoring and recovery
    - Image upgrade rollout
    - Scaling
    - Logging
  - Service Endpoints
    - Discovery/Name-Resolution
    - High Availability
    - Load Balancing
    - Traffic Routing
  - External Services
    - Container Runtimes
      - CRI <--> OCI
    - Network configuration and management
      - CNI
    - Durable volume management
      - CSI

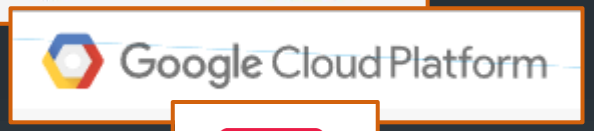
CaaS

- Containers as a Service
  - Container engines, orchestration and underlying compute resources are delivered to users as a service from a cloud provider
- An evolution of Platform as a Service (PaaS) catering to container deployment rather than code deployment
  - No need for build packs!
  - Developers can package software written in any language and with any framework/ library in a portable container



Containers as a Service (CaaS) in the Cloud

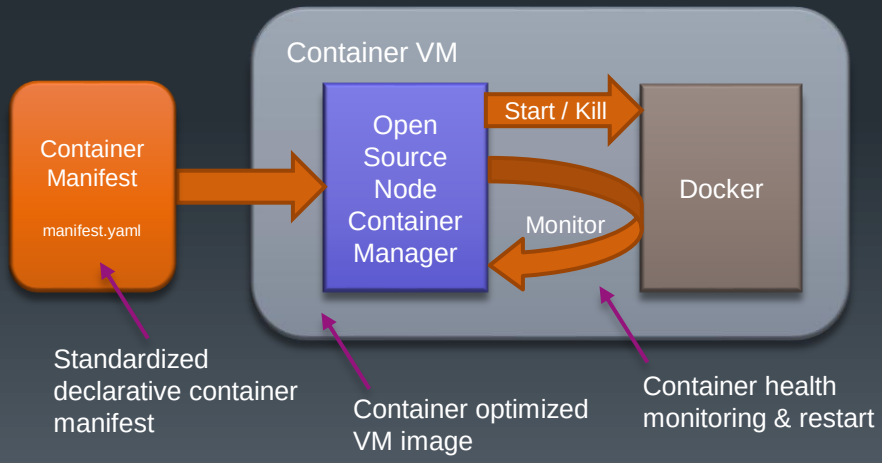
- Proprietary
  - Amazon EC2 Container Service (ECS)
    - AWS EC2 Docker containers
    - ECR (EC2 Container Registry) and AWS integration (IAM, etc.)
  - Docker Datacenter / Docker Cloud
    - Hosted Swarm or Kubernetes and Dashboard (formerly Tutum)
  - Azure Container Service (ACS)
    - Production ready Kubernetes, DC/OS, or Docker Swarm cluster
    - In August of 2017 Microsoft was the first cloud vendor to offer pricing based on running containers – not VMs (!)
- Agnostic
  - Google Container Engine (GKE)
    - Optimized VMs with Kubernetes preinstalled
  - OpenStack Magnum Container Cluster as a Service
    - Swarm, Mesos and Kubernetes deployments available
    - IBM Cloud, Rackspace Carina and other OS clouds
  - Azure Kubernetes Service (AKS) (formerly Deis)
    - Kubernetes as a service created by Engine Yard, bought by Microsoft in 2017
  - IBM Cloud Kubernetes Service (IKS)
    - Hosted Kubernetes
  - Amazon EC2 Kubernetes Service (EKS)
    - Hosted Kubernetes
  - Oracle Container Engine for Kubernetes (OKE)
    - Certified as conformant by the CNCF
- Open Source CaaS solutions (run on Cloud infra w/o lock-in)
  - Apache Mesos
  - CNCF Kubernetes
  - Moby Project Docker Swarm Mode
  - Hashicorp Nomad
  - Rancher



Containers at Scale

- Containers - a key enabler of PaaS cloud environments
  - Google App Engine is one of the most visible container based cloud PaaS systems
- Google Container Engine is a native Kubernetes Container as a Service based cloud
  - Pokémon GO was the largest Kubernetes deployment, 30,000 – 50,000 containers running
  - Its scale and throughput identified a multitude of bugs which were fixed and merged into the open source k8s project
- Everything at Google, from Search to Gmail, is packaged and run in a Linux container
  - Google's Borg system (now replaced by its successor Omega) is a cluster manager that runs hundreds of thousands of jobs, from many thousands of different containerized applications, across a number of clusters each with up to tens of thousands of machines (Borg is the basis for Kubernetes)
  - Over 2 billion containers are started per week at Google (over 3,000 per second on average)
- lmctfy ("Let Me Contain That For You") open source version of Google's container stack
  - Core lmctfy concepts and abstractions have now been ported to OCI libcontainer

Docker containers in Google Kubernetes Engine (GKE)

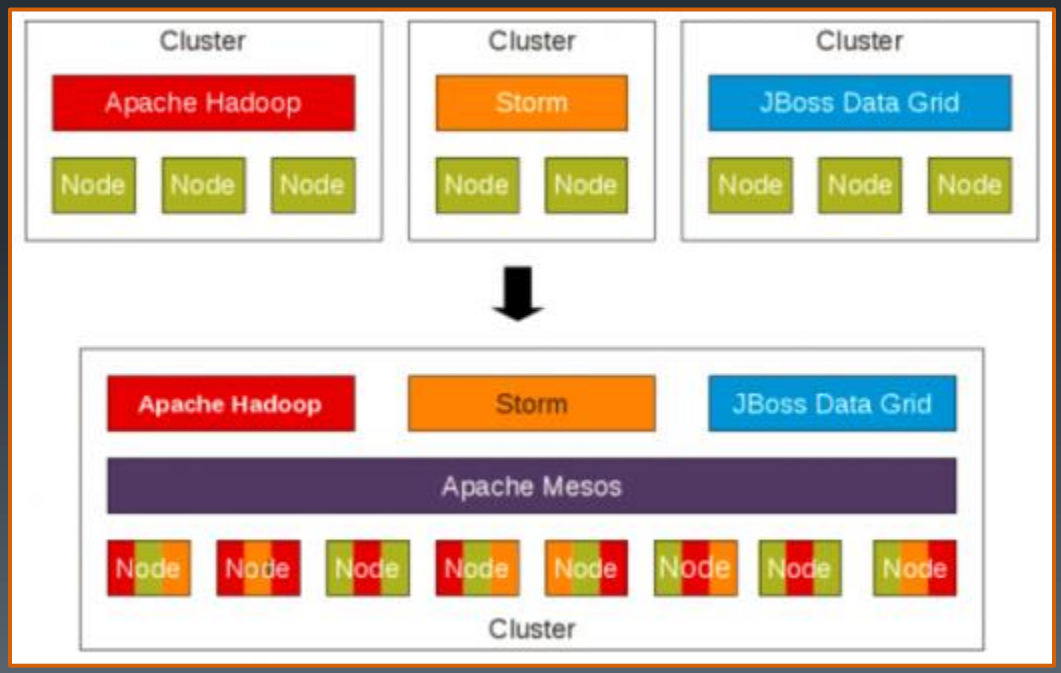


Containers At Scale by
Joe Beda
Published May 22, 2014
in Technology

Borg Paper: https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/43438.pdf
Omega Paper: http://static.googleusercontent.com/media/research.google.com/en//pubs/archive/41684.pdf
https://cloudplatform.googleblog.com/2016/09/bringing-Pokemon-GO-to-life-on-Google-Cloud.html

Mesos

- A top level Apache Project
  - Apache 2.0 license
  - Written in C++
  - Originally developed at the UC Berkeley RAD Lab
- A cluster manager that provides efficient resource isolation and sharing for distributed applications on a cluster of dynamically shared nodes
  - The Mesos “agent” runs on every machine and provides applications with APIs for resource management and scheduling across an entire data center and/or cloud
    - Uses isolation features of the Linux kernel to segregate applications on nodes
  - Linear scalability
  - High availability
  - Support for Docker containers
- Native Mesos applications are called Frameworks
  - Frameworks are available for Hadoop, MPI, Hypertable, Spark, Storm, Kafka, Elasticsearch, Jboss Data Grid and other distributed applications
  - Supports containers through
    - Mesosphere Marathon
    - Apache Aurora
- Provides two level scheduling



Kubernetes

- Kubernetes builds upon a decade and a half of experience that Google has with running production workloads at scale (Borg), combined with best-of-breed ideas and practices from the community
  - The 2015 Borg white paper discusses the architecture and usage of Borg within Google
    - "8.3 Conclusion: Virtually all of Google's cluster workloads have switched to use Borg over the past decade. We continue to evolve it, and have applied the lessons we learned from it to Kubernetes."

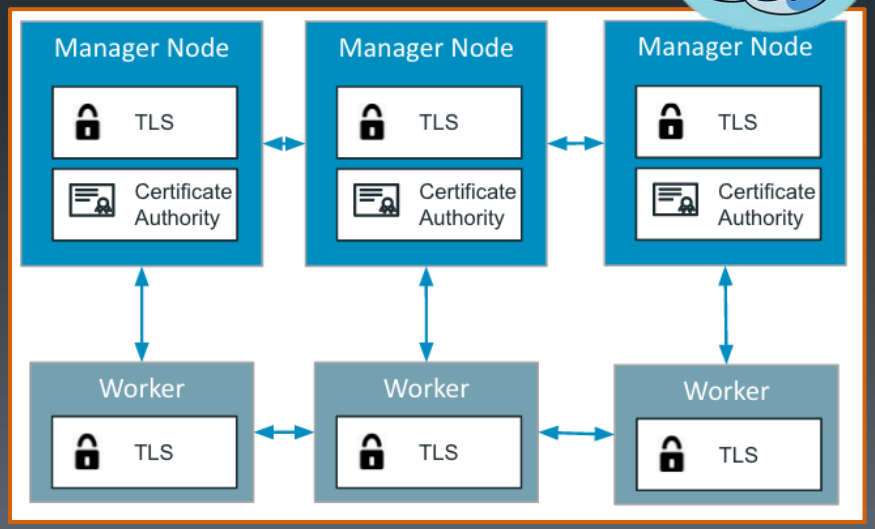
- Kubernetes is an open-source platform for automating deployment, scaling, and operations of application containers across clusters of hosts
- Responds quickly to user demand:
  - Scaling applications on the fly
  - Seamlessly rollout new features
  - Optimized use of hardware using only the resources needed

Kubernetes

- Greek word (κυβερνήτης) meaning helmsman or captain
- Derived from the Greek word kubernan
- Derivations in other languages include the English cybernetics

Docker Swarm Mode

- Docker Engine v1.12.0+ added support for “Swarm Mode”
  - Mode for natively managing a cluster of Docker Engines
  - Allows you to use the Docker CLI to create a swarm, deploy application services to a swarm and manage swarm behavior
- Features
  - Integrated with Docker Engine - no additional orchestration software needed
  - Decentralized design - you can deploy managers and workers using the Docker Engine
  - Declarative service model - lets you define the desired state of services
  - Scaling - you can declare the instance count for each service
  - State reconciliation - the Swarm manager node monitors the cluster state and reconciles any differences between the actual state and your expressed desired state
  - Multi-host networking - you can specify an overlay network for your services and your Swarm manager will then automatically assign addresses to the containers on the overlay network when it initializes or updates them
  - Service discovery - the Swarm manager nodes assign each service a unique DNS name
  - Load balancing - expose the ports for services to an external load balancer
  - Secure by default - each node enforces TLS mutual authentication and encryption
  - Rolling updates - apply service updates to nodes incrementally, Swarm manager lets you control the delay between service deployment to different sets of nodes

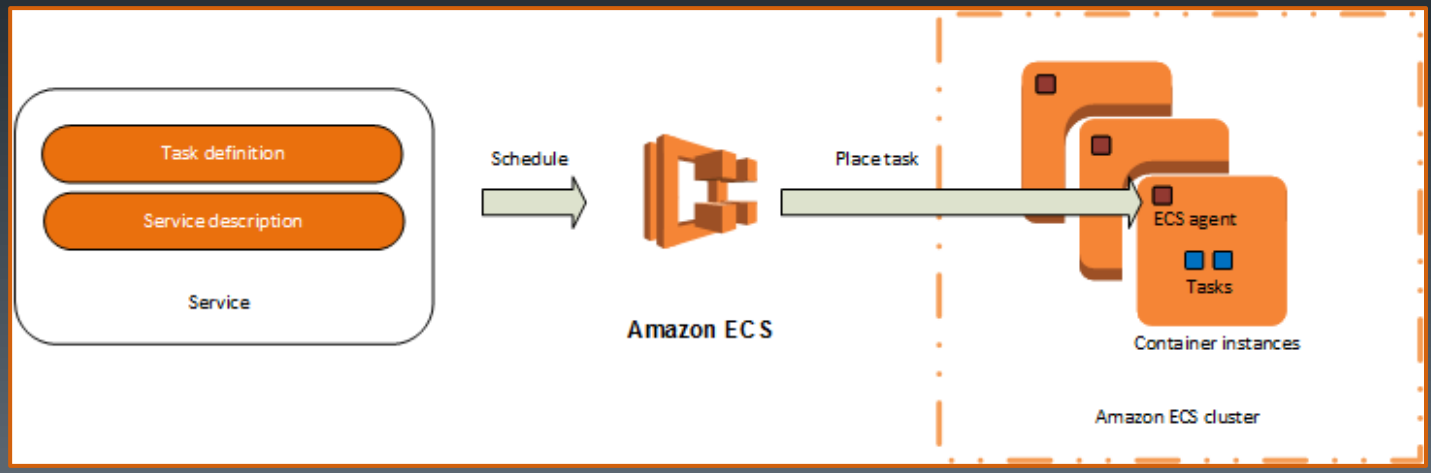


Case Study

ECS

EC2 Container Service

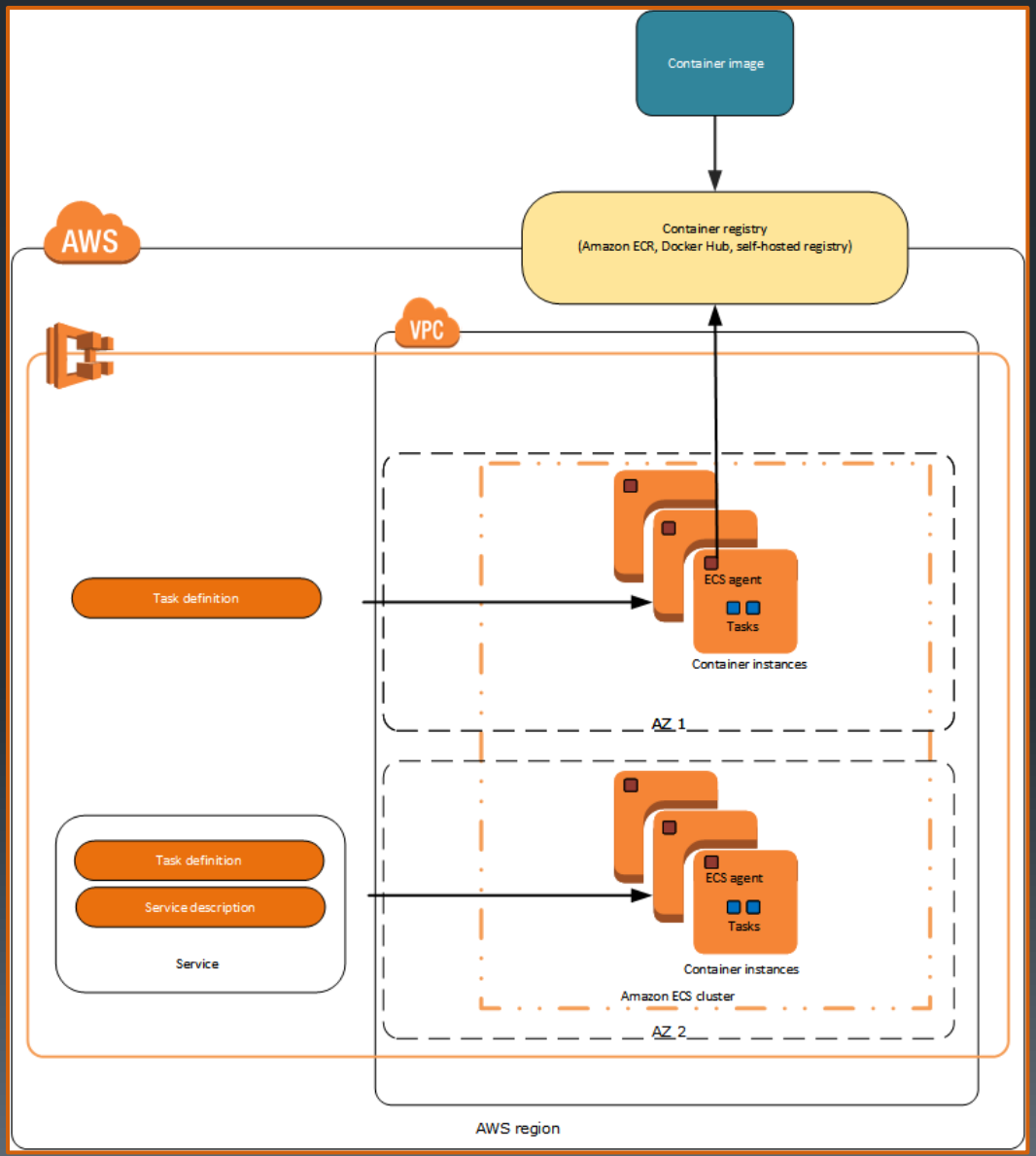
- Amazon EC2 Container Service (Amazon ECS)
- Two modes: EC2 launch type and Fargate launch type
  - EC2 launch type – allows you to run, stop, scale and manage Docker containers on a cluster of Amazon EC2 instances
  - Fargate launch type – specify the CPU and memory requirements, define networking and IAM policies, and launch the container from a pre-created image
- Schedules the placement of containers based on resource needs, isolation policies, and availability requirements
- A regional service that can run containers across multiple AZs
- Eliminates the need to operate and configure a cluster
- Leverages Docker's awslogs plugin for CloudWatch Logs integration
- AWS Elastic Beanstalk can be used to deploy Docker containers to ECS
  - Using ECS directly provides more fine-grained control



ECS Tasks

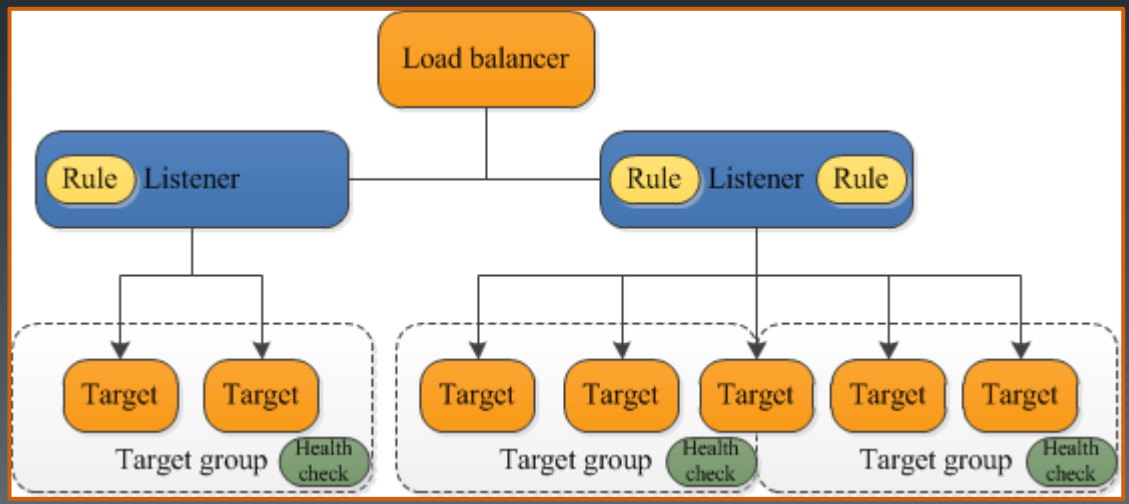
- Clusters
  - Logical grouping of EC2 instances
- VPC
  - You create ECS clusters within a VPC
- Images
  - ECS only runs container images
  - Images can be stored and pulled from any container registry
- Tasks
  - A text file in JSON format that describes one or more containers that form your application
  - Specifies various parameters, such as which images to use, which ports should be mapped and what data volumes should be used
- Container Agent
  - Runs on each instance within the cluster
  - Sends info about the instance to ECS

```
{
      "family": "webserver",
      "containerDefinitions": [
      {
            "name": "web",
            "image": "nginx",
            "cpu": 99,
            "memory": 100,
            "portMappings": [{
                  "containerPort": 80,
                  "hostPort": 80
            }]
      }]
}
```



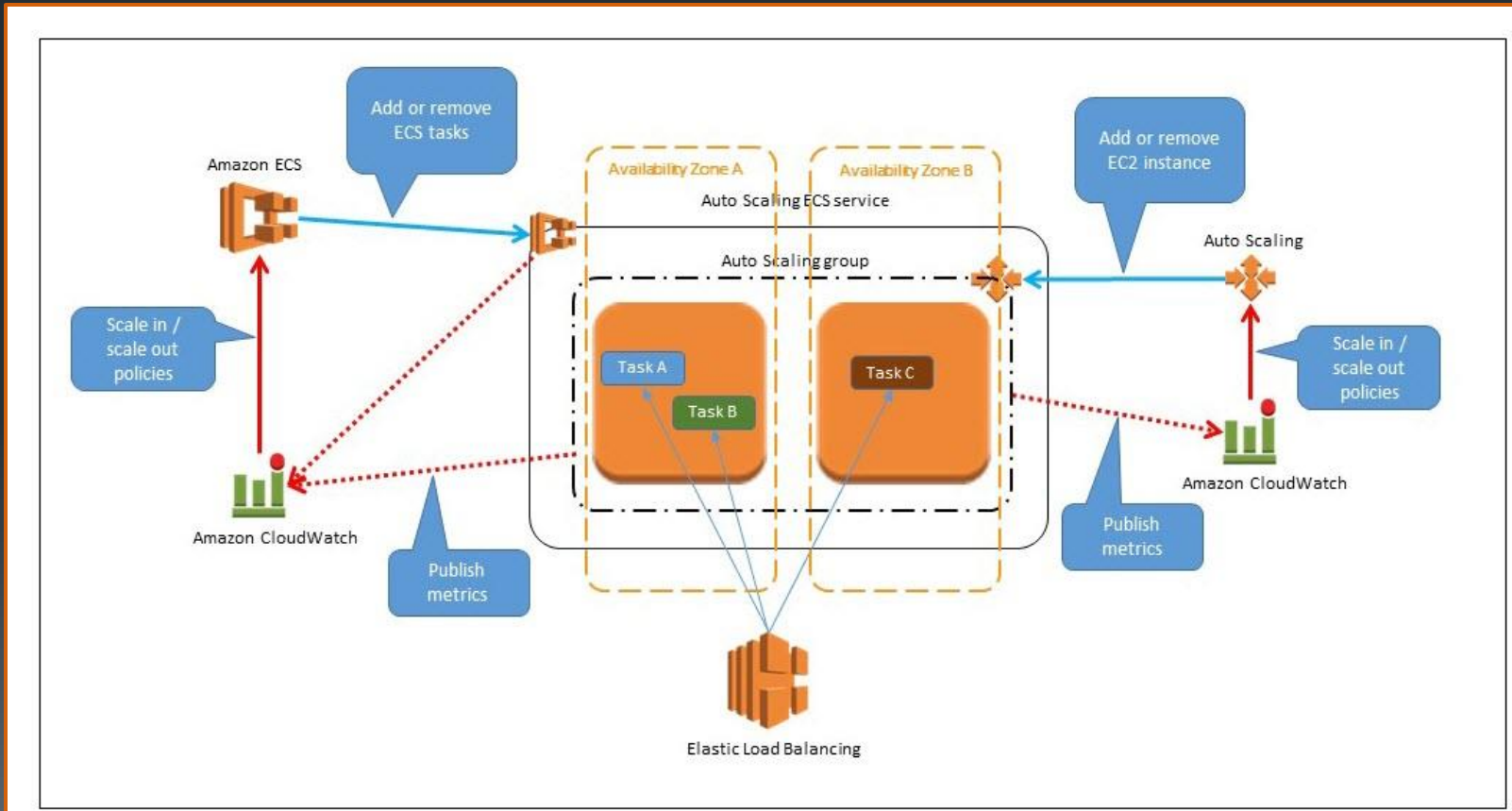
ECS Services

- ECS services run a specified number of task replicas
- If a task fails or stops ECS launches a replacement
- When the service scheduler launches new tasks it attempts to balance them across the Availability Zones, processes:
  - Determine which instances in the cluster can support the service's task definition
  - Sort the possible Zones by fewest number of running tasks for this service
  - Favor instances with the fewest number of running tasks for this service
- Services can use a special ELB implementation known as an Application Load Balancer
  - Allow containers to use dynamic host port mapping
    - Multiple tasks from the same service are allowed per instance
  - Application Load Balancers support path-based routing and priority rules
    - Multiple services can use the same listener port on a single Application Load Balancer



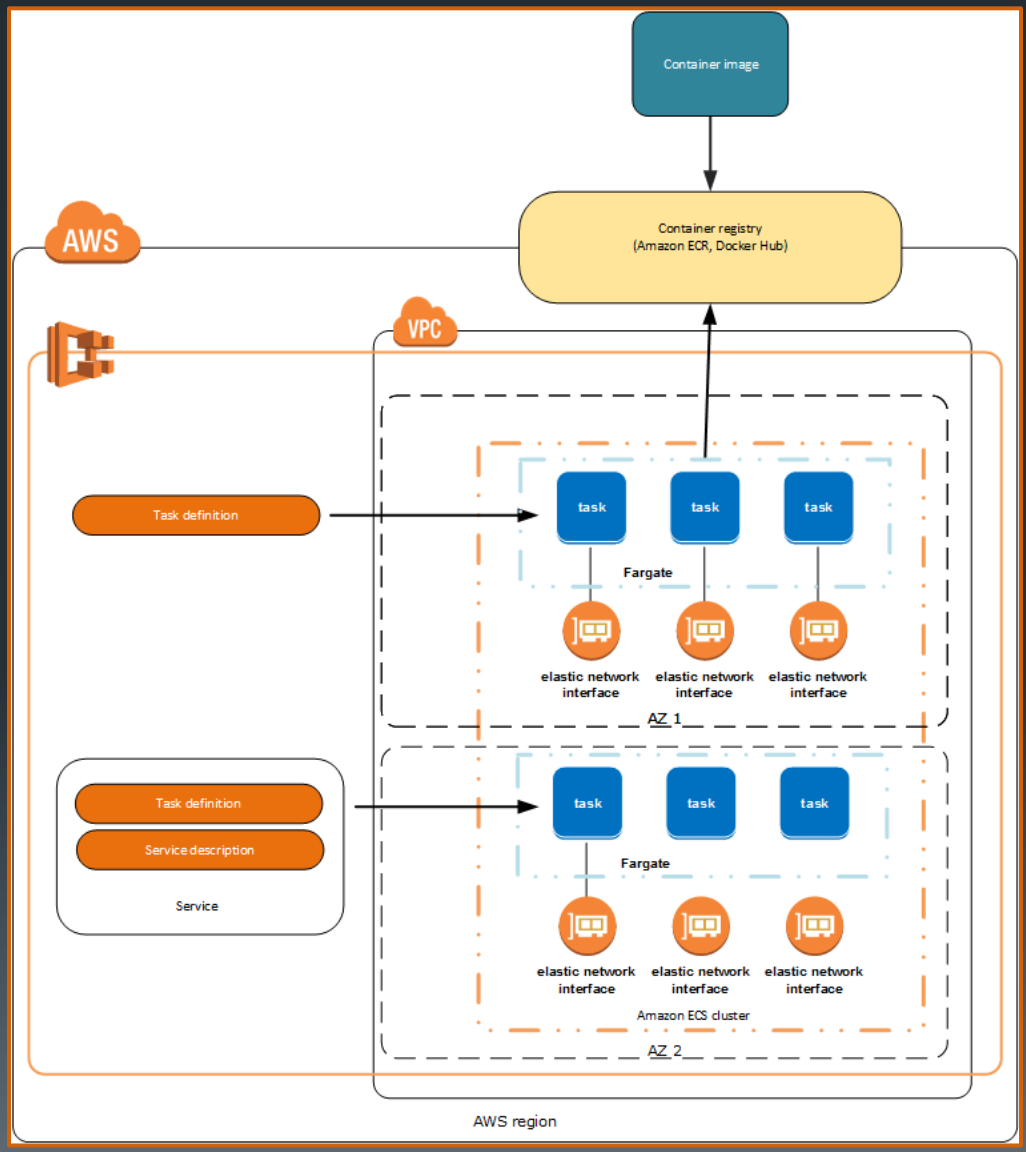
Amazon Auto Scaling

- Auto Scaling can be performed at multiple levels
  - Infrastructure level: EC2 Auto Scaling Groups
  - Application level: ECS tasks



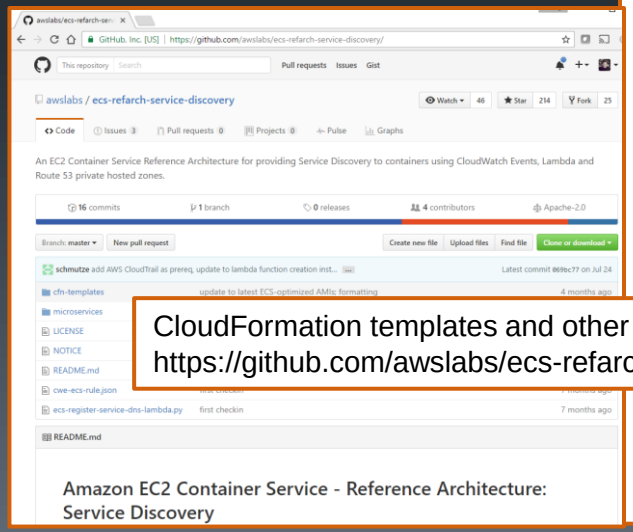
AWS Fargate

- Run containers without managing servers or clusters
- Removes the need to choose server types, decide when to scale your clusters, or optimize cluster packing
- Manages scaling and infrastructure needed to run your containers in an HA manner
- Uses the same ECS task definitions, CPU and memory specifications, networking and IAM policies as those defined in EC2 launch type

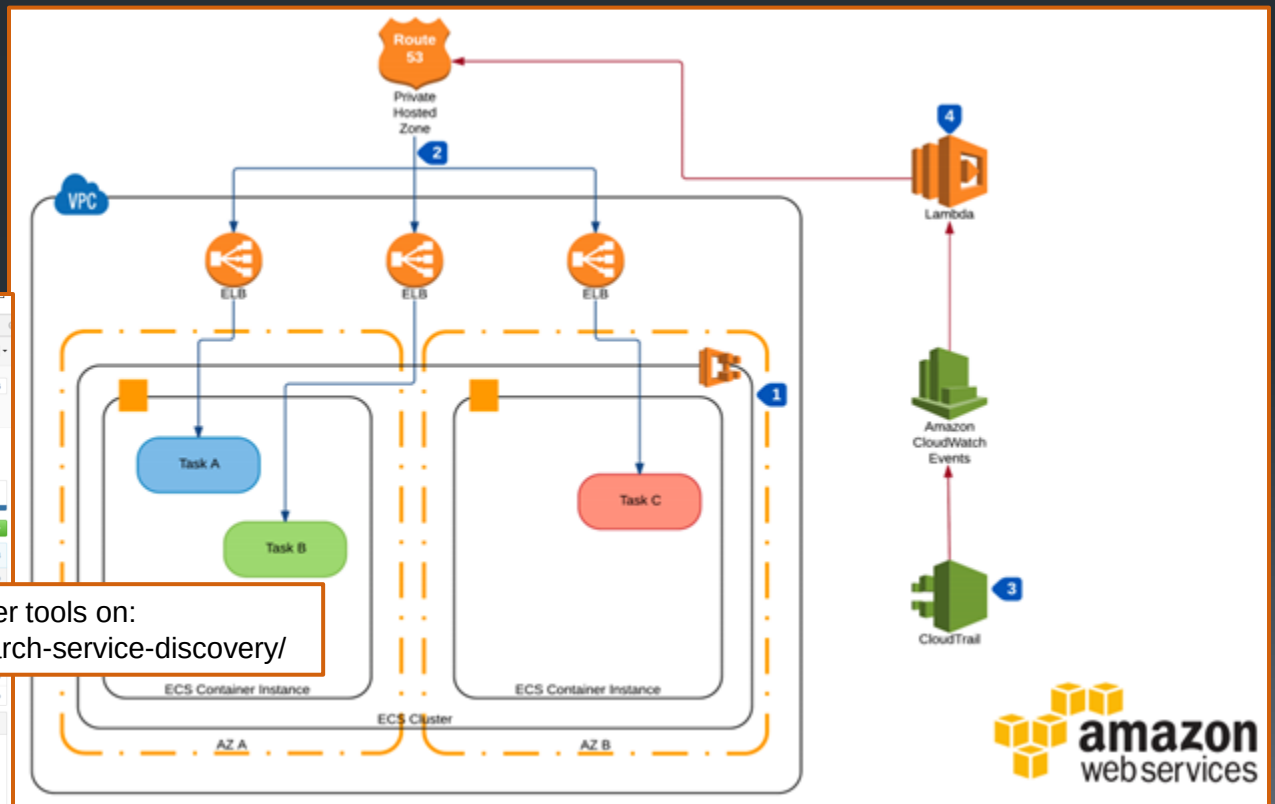


ECS Service Discovery

- Amazon has created a reference architecture to demonstrate DNS/ELB service discovery
  - A CloudWatch Events filter listens to all ECS service creation messages (1) from AWS CloudTrail (3) and triggers an Amazon Lambda function (4)
  - The Lambda function identifies which Elastic Load Balancing load balancer is used by the new service and inserts a DNS resource record (CNAME) pointing to it, using Amazon Route 53 (2)
  - The Lambda function also handles service deletion to make sure that the DNS records reflect the current state of applications running
  - Because Route 53 allows hosted zones per VPC and ECS lets you segment clusters per VPC, you can isolate different environments (dev, test, prod) while sharing the same service names
- Alternatives include:
  - Directly passing Elastic Load Balancing names as environment variables
  - Setting up vendor or open source solutions (Eureka, Consul, …)



CloudFormation templates and other tools on:
https://github.com/awslabs/ecs-refarch-service-discovery/



amazon web services

Summary

- Containers as a Service solutions allow organizations to deploy containers directly to the cloud
- CaaS enables the agile last mile
  - Time to market is the most important CaaS motivator for most firms
- Older PaaS solutions (like Elastic Beanstalk, Cloud Foundry and OpenShift) have rapidly added container support
- Open source CaaS systems have grown rapidly
  - Apache Mesos
  - CNCF Kubernetes
  - Docker Swarm
- AWS ECS is Amazon's proprietary CaaS platform

Lab 6

- Deploying Containerized Apps on GKE
  - Use Google Kubernetes Engine (GKE) to explore container orchestration in the cloud

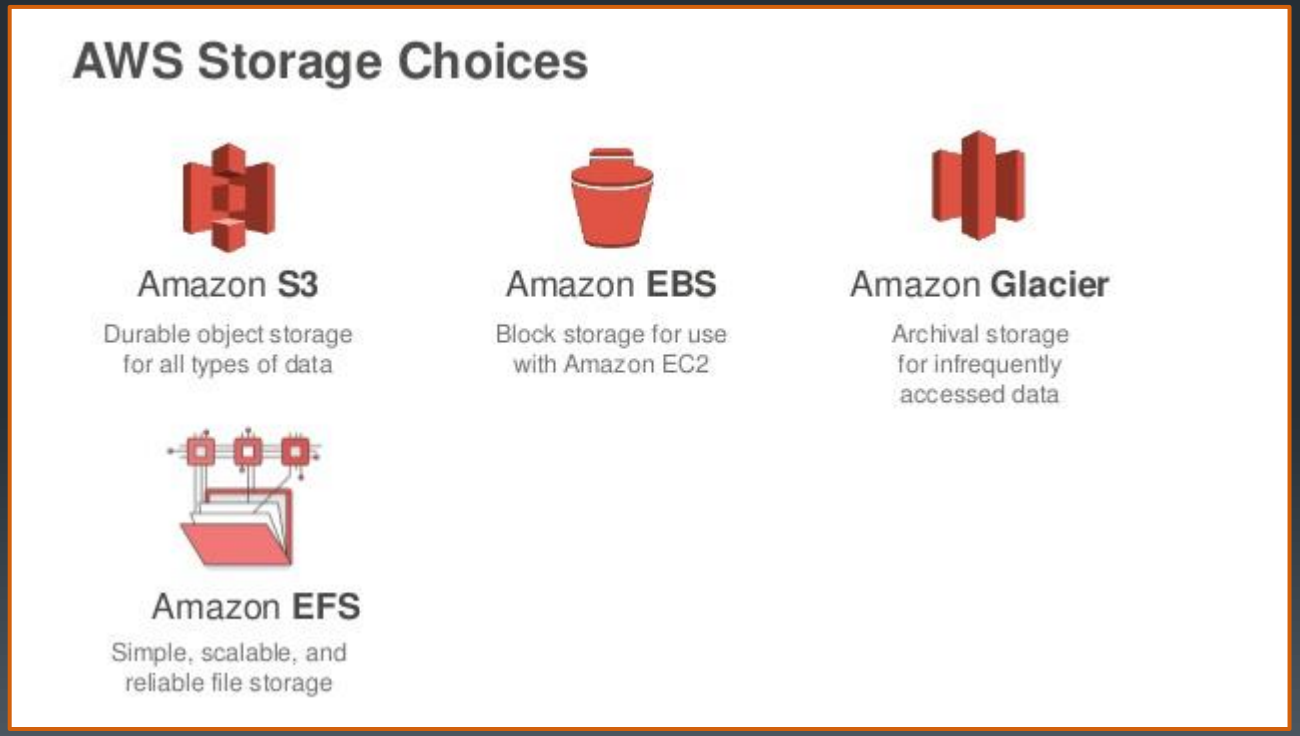
7: Cloud data stores

Objectives

- Define block, object and file storage
- Introduce Relational Database hosting services and AWS RDS
- Examine NoSQL database models and software
  - Key-Value
  - Document
  - Column
  - Graph

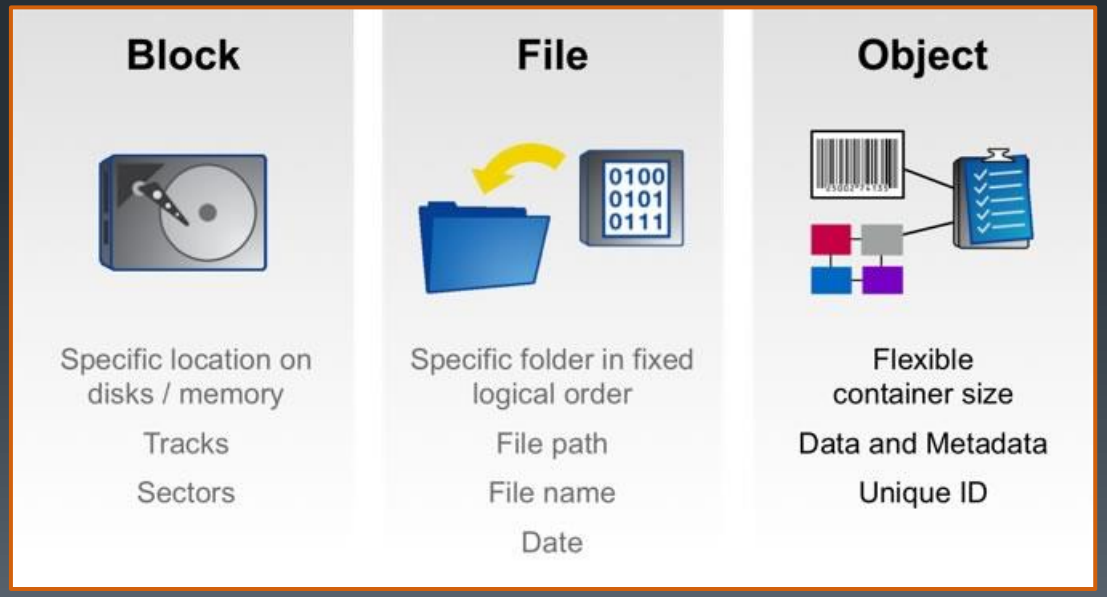
Cloud-based Storage

- Cloud systems offer a range of generic storage options
  - Archival Storage
  - Object Storage
  - Block Storage
  - File Storage
- Cloud systems also offer a range of database solutions
  - Many applications require nothing more than a scalable database



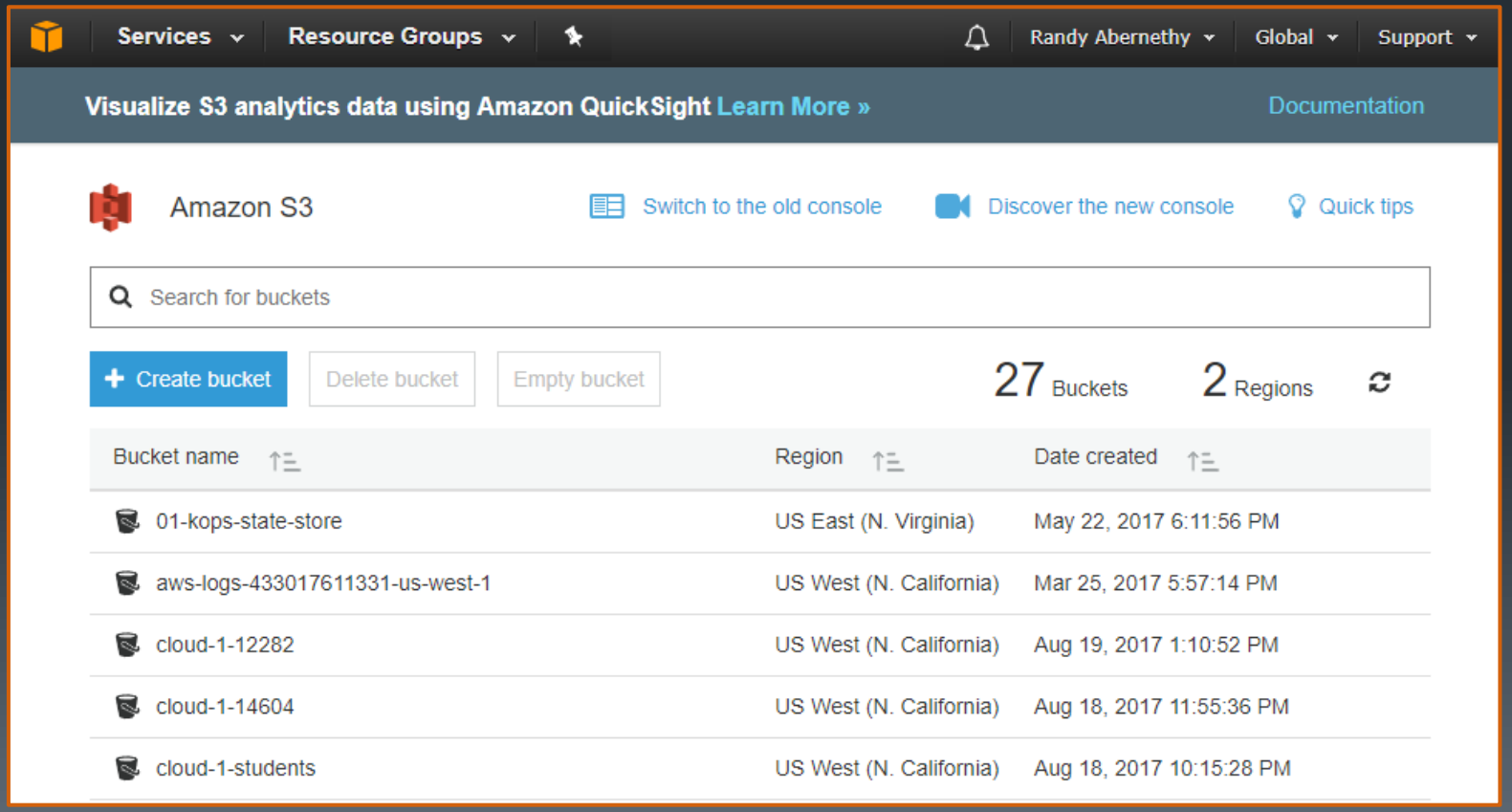
Cloud Object Storage

- Object storage manages data as objects
- Object stores typically save URL accessible blobs
- Object stores typically support a variable amount of metadata associated with each object
- Object stores expose namespaces that can span multiple servers
  - Data management functions typically replicate and distribute object data for reliability
- Object storage systems allow retention of massive amounts of unstructured data (terabytes, petabytes or more)
- Object storage is used for purposes such as storing photos on Facebook, songs on Spotify, or files in online collaboration services, such as Dropbox



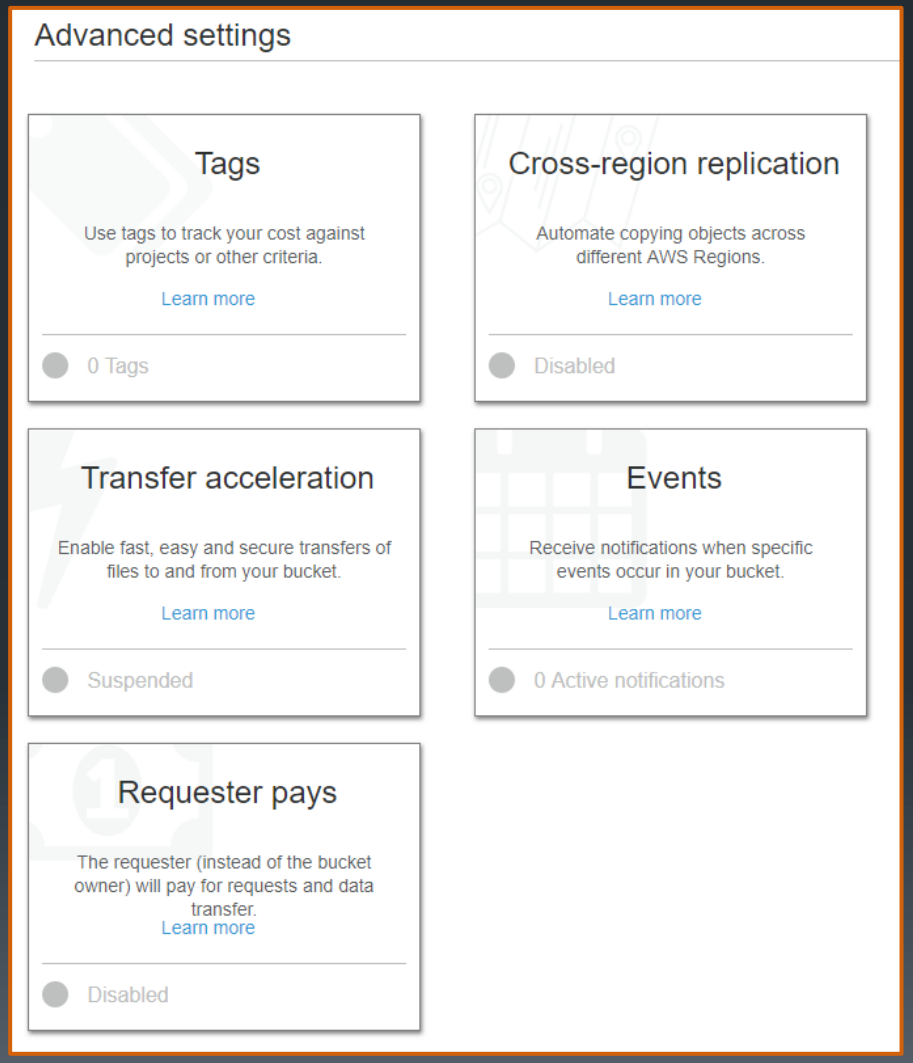
Simple Storage Service (S3)

- Amazon S3 is an object store
  - Maps URLs to blobs stored and (optionally) replicated on AWS infrastructure
- Objects are stored in buckets
  - Buckets must have AWS wide unique names (used to compose URL)



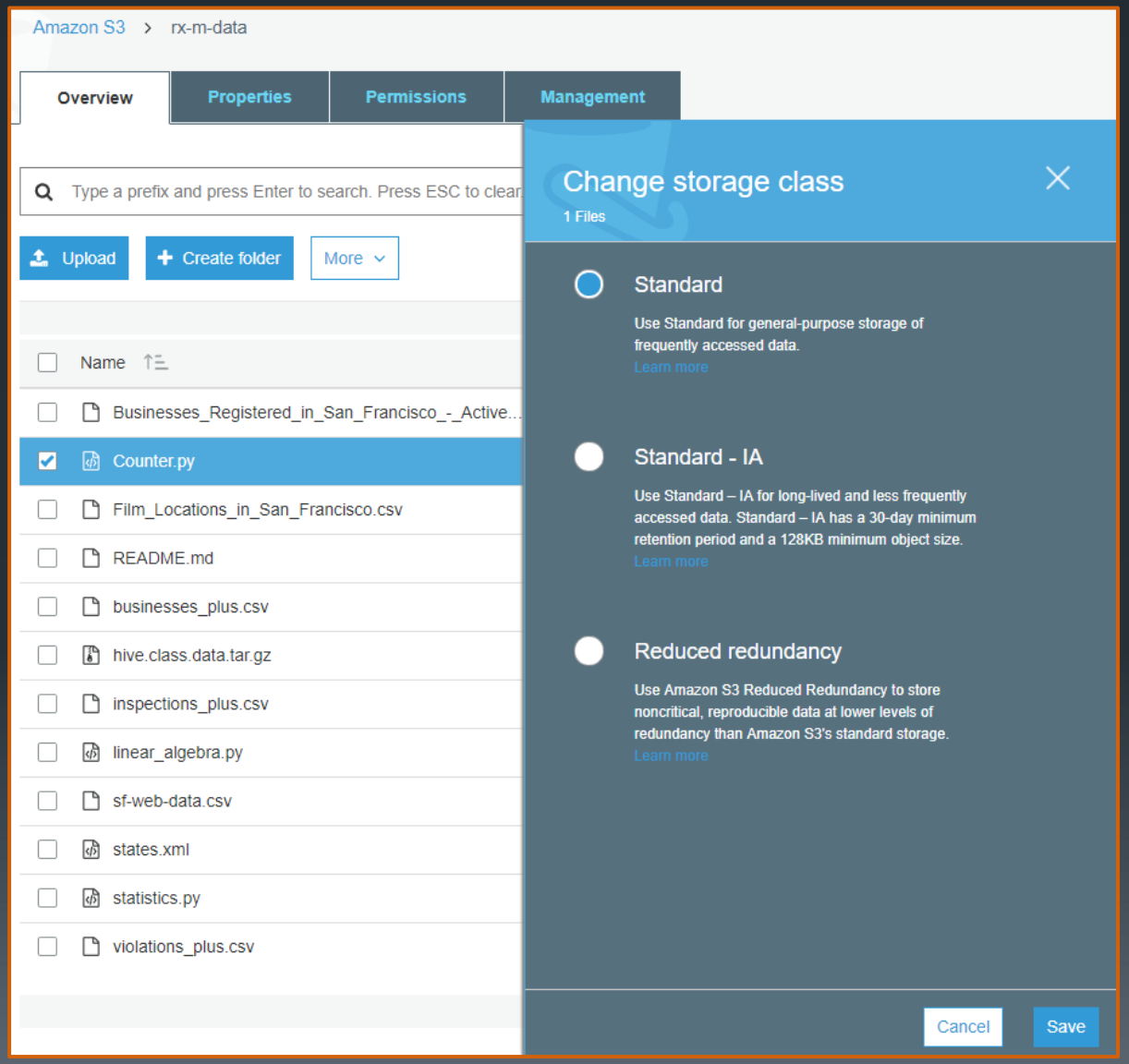
S3 Bucket Features

- Advanced Features
  - Tags
  - Transfer acceleration
  - Requester pays
  - Cross-region replication
  - Events
- Tiered Storage
  - Lifecycle policies can automatically move objects to Glacier (an offline but much cheaper storage option)
- Object Expiration
  - Expiration policies can automatically remove old objects



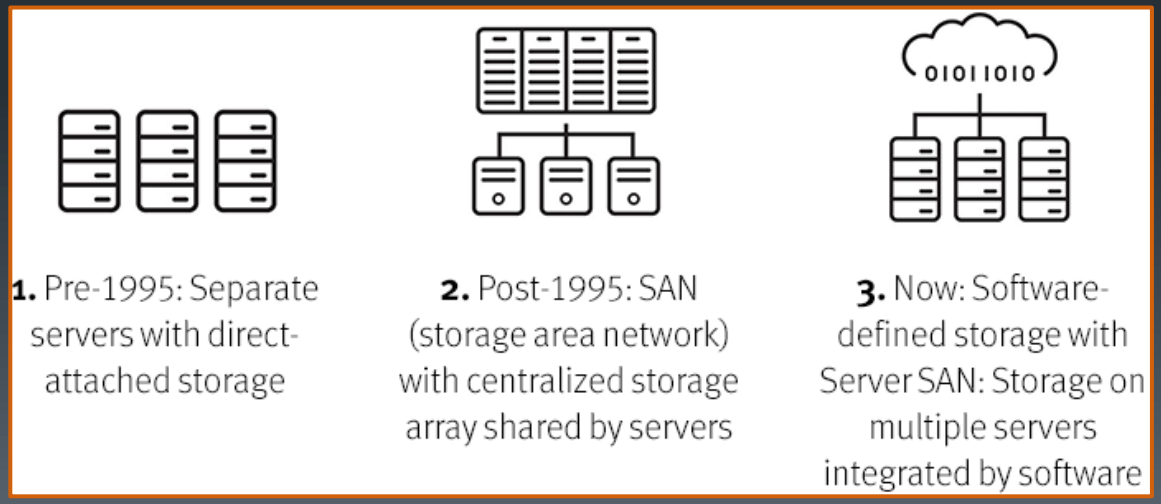
Storage Class

- Individual objects have an adjustable storage class
  - Standard
    - Low latency/high throughput
    - 99.999999999% object durability
    - 99.99% availability over a given year
    - Designed to sustain the concurrent loss of data in two facilities
  - Standard – IA
    - Infrequent Access
    - Similar to Standard but trades some performance for lower cost
  - Reduced Redundancy
    - Reduces replication and eliminates durability guarantee for lower cost
    - Only use on data you can recreate!
- Lifecycle processes can be configured to change storage class of objects reaching a certain age



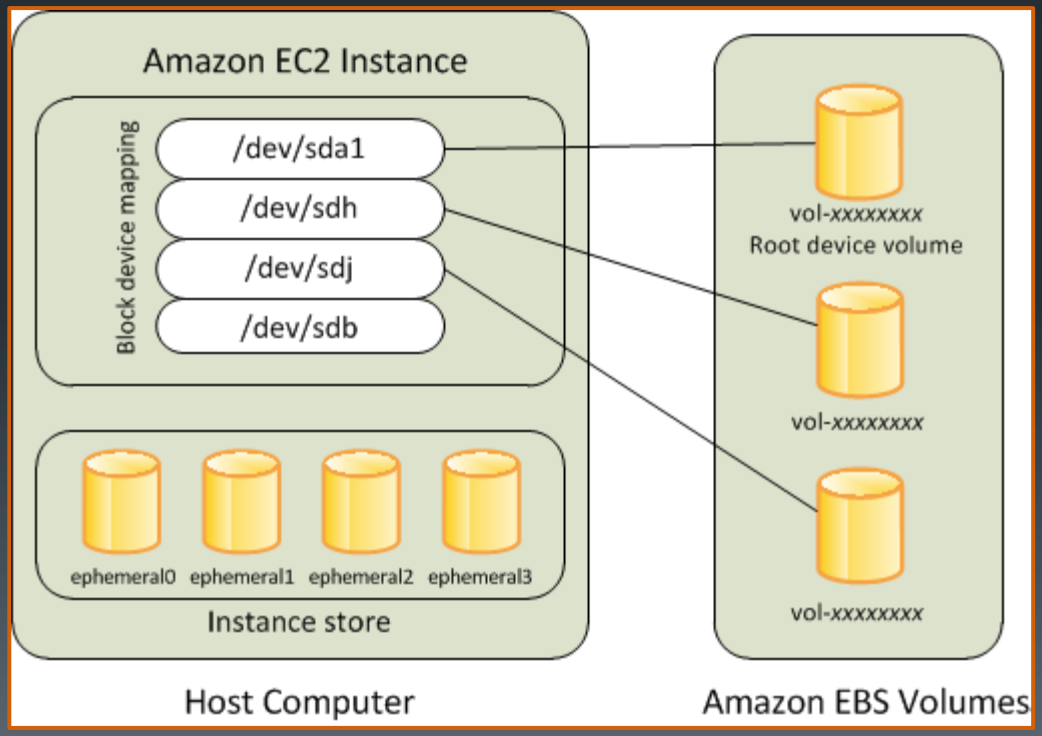
Cloud Block Storage

- A block is a sequence of bytes representing a unit of storage, usually on a disk device
  - Blocked data is normally read or written as a unit
- Most file systems are based on a block device
  - Any block storage device can therefore host such a file system
  - The block size in file systems may be a multiple of the physical block size
- Some systems use block devices directly without a file system
  - Database management systems (DBMS) in particular
- Distant block devices are typically accessed via a storage area network (SAN) using a protocol such as iSCSI or FC
  - iSCSI (Internet Small Computer Systems Interface) is an IP based block storage networking standard
    - EBS is iSCSI based
  - FC (Fiber Channel) is a high speed network technology (16, 32, or 128 Gb/sec) primarily used to connect computer data storage to servers



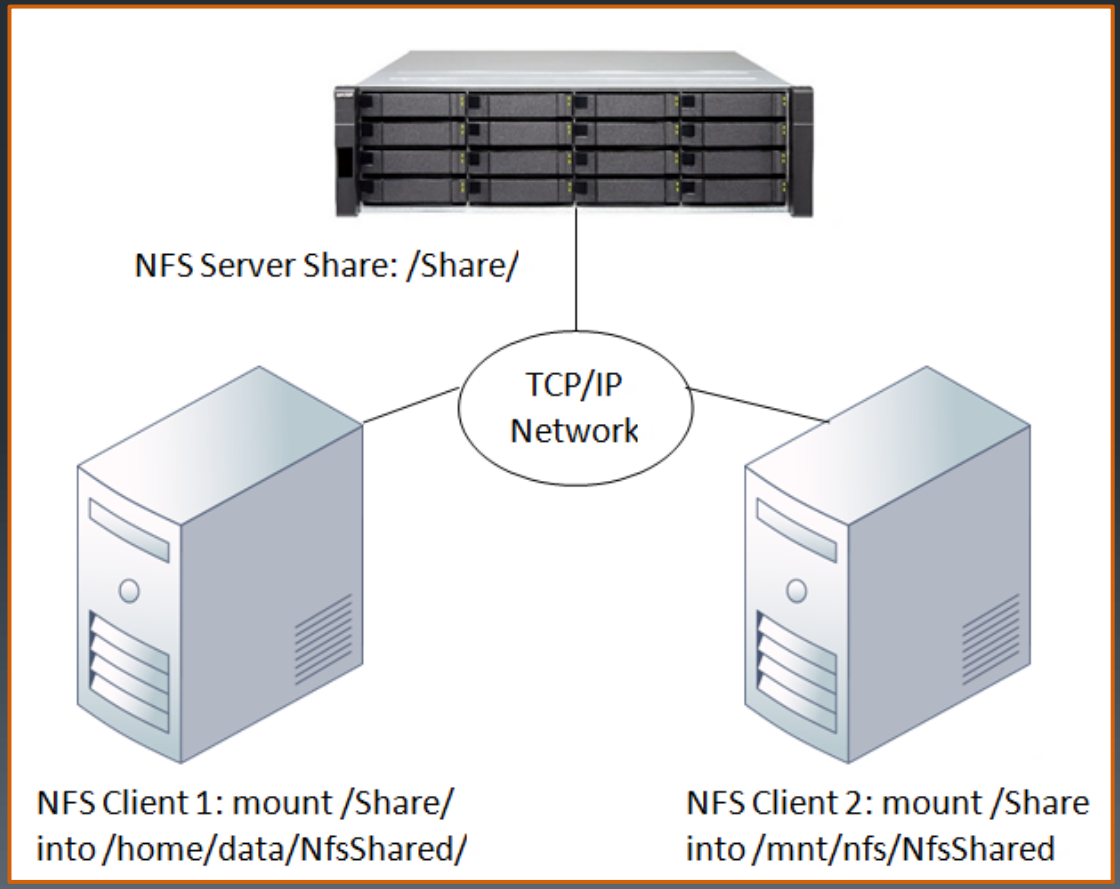
Cloud Block Storage Solutions

- All major cloud providers offer block storage services
  - AWS EBS
  - Google Persistent Disks
  - Azure Disks
  - OpenStack Cinder
    - IBM Cloud
    - Rackspace
    - Others



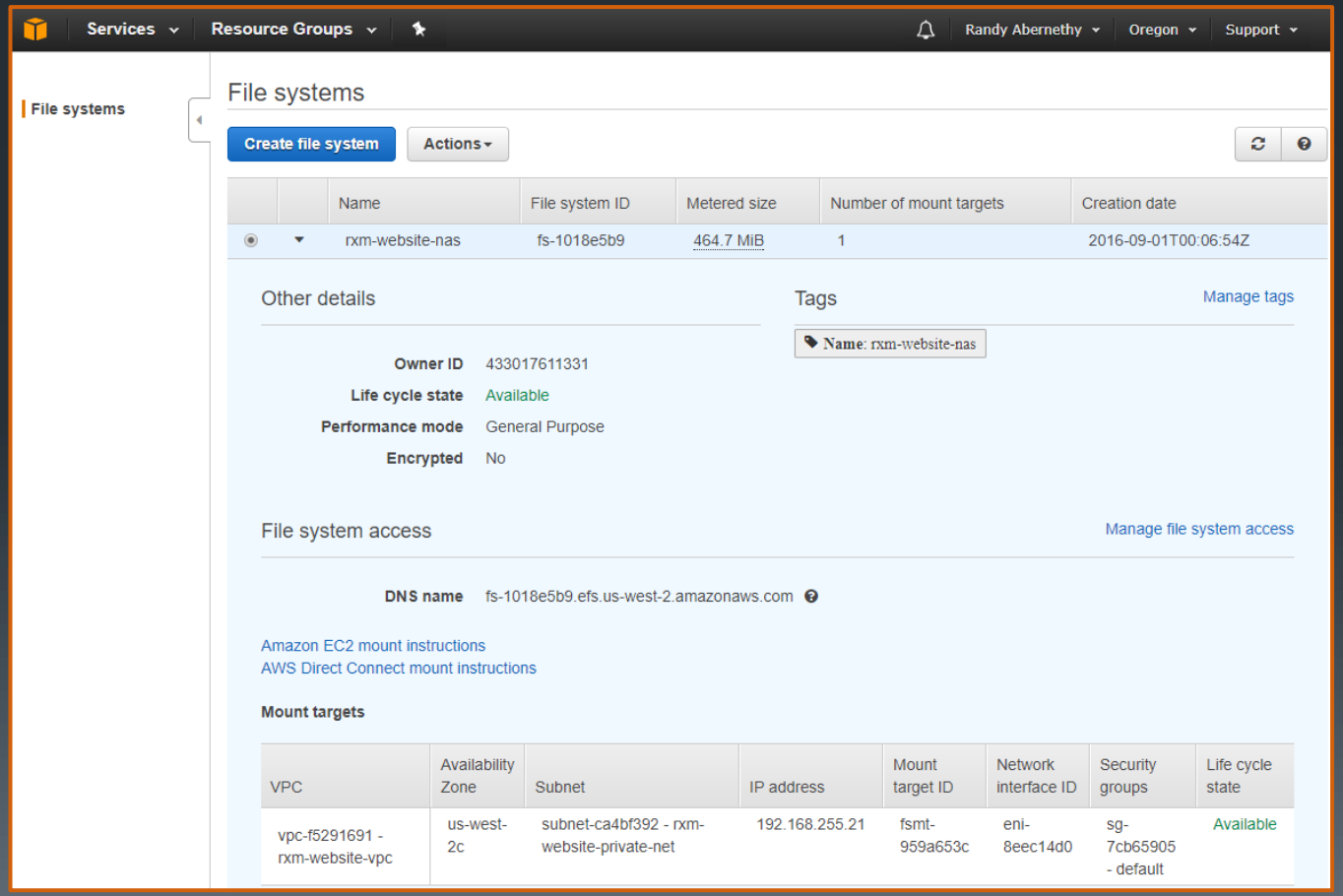
Cloud File Storage

- Many public cloud vendors offer file storage as a service
  - AWS Elastic File System (EFS)
  - Azure File Storage
  - OpenStack Manila
    - IBM Cloud
    - Others
- Such systems offer hosted file system storage
- Other open source systems can be deployed manually
  - NFS
  - GlusterFS
  - CephFS



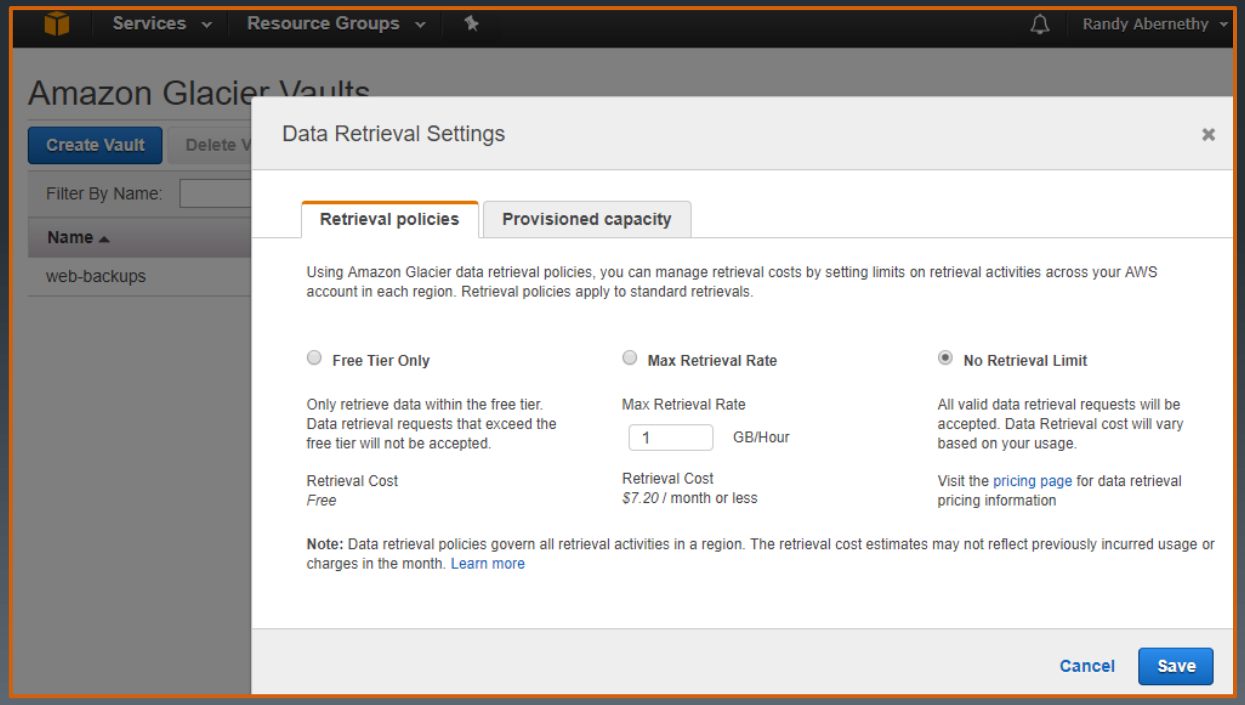
AWS EFS

- Amazon Elastic File System (EFS) provides simple, scalable file storage for use with Amazon EC2
- EFS storage capacity is elastic, growing and shrinking automatically as you add and remove files
- EFS provides standard file system interfaces and access semantics
  - Multiple EC2 instances can access an EFS file system at the same time
- EFS file systems can be mounted from on-premises data center servers
  - Requires Amazon VPC with AWS Direct Connect
- EFS is designed for performance, high availability and durability
- A spectrum of use cases, e.g.:
  - Web serving
  - Media processing
  - Container storage
  - Big Data
  - Analytics



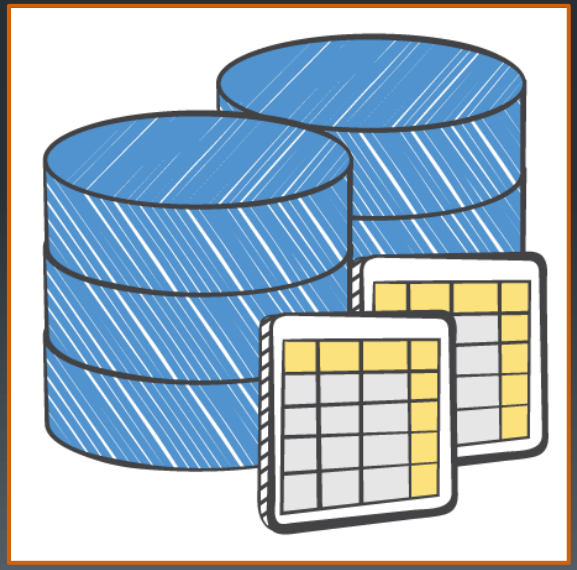
Amazon Glacier

- Amazon Glacier stores data in "archives"
- An archive can be any data file
  - Photo, video, document, etc.
- To archive multiple files you can compress them into a single file
  - zip, tar, etc.
- A single archive can be as large as 40 terabytes
  - You can store an unlimited number of archives
- Each archive is assigned a unique archive ID at the time of creation
- The content of the archive is immutable
- Glacier uses "vaults" as containers to store archives
- Under a single AWS account, you can have up to 1000 vaults
- Vaults are accessible via the AWS Management Console and API
- Glacier retrieval incurs fees



DB Hosting

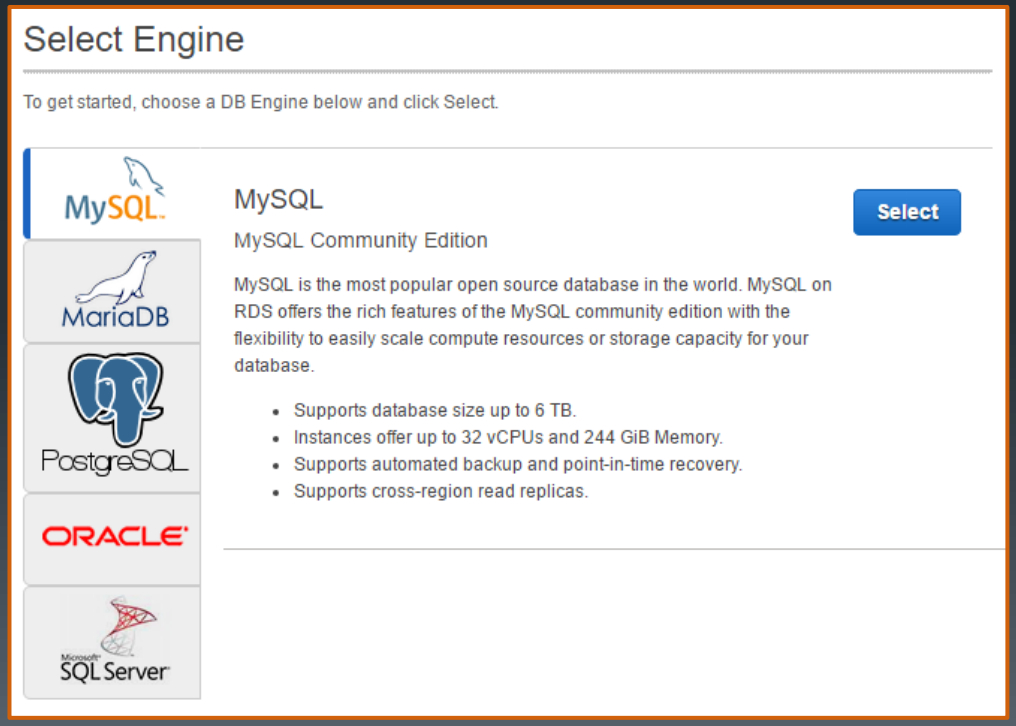
- Databases have traditionally been installed and managed by enterprise IT departments and similar organizations
- Cloud systems offer databases as a service
  - AWS RDS, DynamoDB, etc.
  - Google Cloud SQL, Cloud BigTable, etc.
  - Azure MySQL, PostgreSQL, Cosmos DB, etc.
  - OpenStack Trove
    - IBM Cloud
    - Others
- These services typically offer:
  - Optimal deployment and tuning
  - Extreme scalability
  - Automated replication
  - Automated backups



AWS RDS

- Amazon Relational Database Service (Amazon RDS)
- Easy to set up, operate, and scale relational database
- Multi-AZ deployment with high availability and built-in automated fail-over from primary to synchronously replicated secondary
- Read Replicas can scale beyond capacity of single database deployment
- Resizable capacity
- Amazon Aurora
  - MySQL-compatible
  - Combines the speed and availability of high-end commercial databases with the simplicity and cost-effectiveness of open source databases
  - Up to five times better performance than MySQL

Amazon Aurora



NoSQL Solutions

- Most cloud systems offer NoSQL solutions
- NoSQL systems process aggregates not tuples and are typically designed for clustering and unstructured data
- NoSQL storage engines address many of the shortcomings of traditional relational database management systems (RDBMS)
  - RDBMS <> Application Impedance mismatch
    - The difference between the relational model and app in-memory data structures is often substantial
  - SOA services use richer data structures with nested records and lists
    - Represented as documents in XML, JSON, etc., reducing server round trips makes a rich structure desirable
  - Joins and atomicity, consistency, isolation, and durability (ACID) requirements make relational systems difficult to scale horizontally
    - Join performance and highly structured data is also ill-suited for big data environments
  - A cluster of small machines can use commodity hardware
    - Scaling out
    - Cheaper
    - More resilient
- SQL is a powerful and well understood language
  - Requires significant complexity in the underlying platform
  - Not present in most NoSQL platforms
    - However SQL like solutions are present on some NoSQL solutions
      - Hive
      - Cassandra CQL
      - CockroachDB

NoSQL Data Models

NoSQL Attributes:

- Not using the relational model
- Running well on clusters
- Open-source
- Built for the 21st century web estates
- Schemaless

Key-Value

- Redis
- Memcached
- Riak
- BerkeleyDB
- Hazelcast
- LevelDB

Document

- AWS DynamoDB
- MongoDB
- CouchDB
- Couchbase
- MarkLogic
- OrientDB
- ArangoDB
- RavenDB

Column

- Cassandra
- AWS SimpleDB
- HBase
- Accumulo
- Hypertable

Graph

- Neo4J
- FlockDB
- Virtuoso
- Giraph
- Titan (can use DynamoDB as backend)
- OrientDB
- ArangoDB

AWS solutions in orange

Key-Value Model

- Each aggregate has a key or ID that is used to get the data
- The aggregate (value) is opaque to the database
  - Usually seen as a blob
  - Aggregates can be anything
  - Usually size limited
- Some DBs support metadata to define relationships, expiration times, etc.
- These solutions are very simple, and thus tend to be very fast
- Sharding is based on key hash

Use:
- Storing session Info
- User profiles/preferences
- Shopping carts

Don't Use:
- Data relationships
- Multioperation transactions
- Query by data
- Operations on sets

Memcached

- A general-purpose distributed memory K/V caching system
- Used to speed up dynamic database/API driven websites
- Runs on Unix, Linux, Windows and Mac OS X
- Applications using Memcached typically fall back to a database request on cache miss
- The client-side library knows all servers
  - Servers do not communicate with each other
- When the table is full, subsequent inserts cause older data to be purged in least recently used (LRU) order
- Originally developed by Danga Interactive for LiveJournal, now used by YouTube, Reddit, Zynga, Facebook, Twitter, Tumblr, Wikipedia, etc.
- Engine Yard and Jelastic are using Memcached as the part of their Platform as a Service technology stack
- Heroku offers a managed Memcached service built on Couchbase Server as part of their Platform as a Service
- Google App Engine, AppScale, Windows Azure and Amazon Web Services also offer Memcached

```
function get_foo(int userid) {
   /* first try the cache */
   data = memcached_fetch("userrow:" + userid);
   if (!data) {
      /* not found : request database */
      data = db_select("SELECT * FROM users WHERE userid = ?", userid);
      /* then store in cache until next get */
      memcached_add("userrow:" + userid, data);
   }
   return data;
}
```

Document Model

- Each aggregate has a key or ID that's used to get the data
- The aggregate is visible to the database
  - Usually a JSON or XML document
  - Aggregates can have any structure supported by the document type
- Document DBs support limited searches on document data
- These solutions are simple, fast but supply useful aggregate search features
  - You can look up data by something other than aggregate key
- Sharding is based on key hash
- In practice the distinction between Key-Value and Document DBs blurs heavily in some implementations

Use:
- Event logging
- CMS/blogging
- Web analytics
- E-commerce

Don't Use:
- Complex transactions
- Queries against varying aggregate structure

DynamoDB

- A fully managed NoSQL cloud database
- Consistent, single-digit millisecond latency at scale
- Supports both document and key-value store models
- Secondary indexes can be created to allow queries on non-key fields
- Good fit for mobile, web, gaming, ad tech, IoT, and many other applications
- Cross-region replication support
- Local version of DynamoDB available for development
  - Saves on provisioned throughput, data storage and transfer fees
  - No need for an Internet connection
  - Only minor changes to code to switch to DynamoDB web service
- DynamoDB offers triggers to execute AWS Lambda functions
  - Can automatically execute a lambda when item levels change for example
- The DynamoDB Logstash Plugin allows Elasticsearch to search DynamoDB messages, locations, tags, and keywords
- Can act as a storage backend for the Titan Graph Database

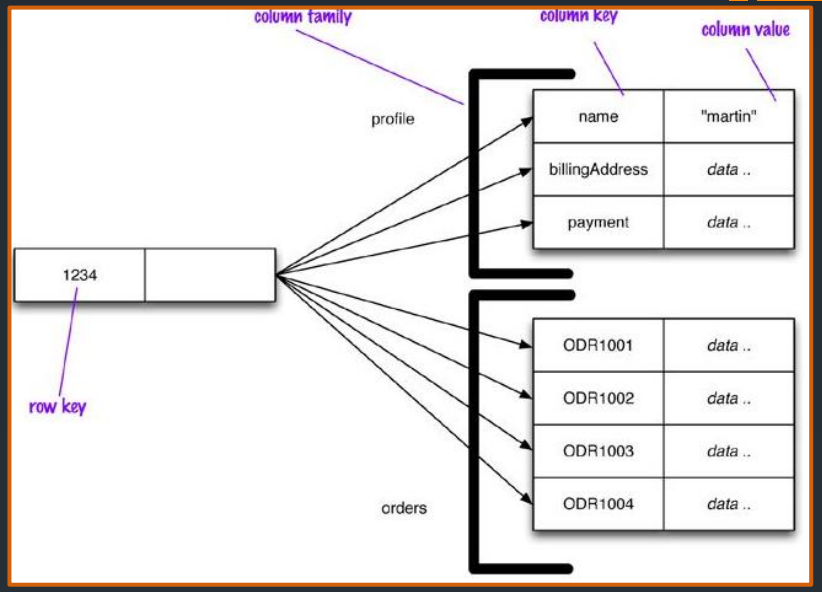
```
FROM openjdk:8u111-jre-alpine
RUN /usr/bin/curl -L http://dynamodb-local.s3-website-us-west-2.amazonaws.com/dynamodb_local_latest | /bin/tar xz
ENTRYPOINT ["/opt/jdk/bin/java", "-Djava.library.path=./DynamoDBLocal_lib", "-jar", "DynamoDBLocal.jar"]

---

$ docker run -i -t -p 7777:7777 dynamodb-local -inMemory -port 7777
```

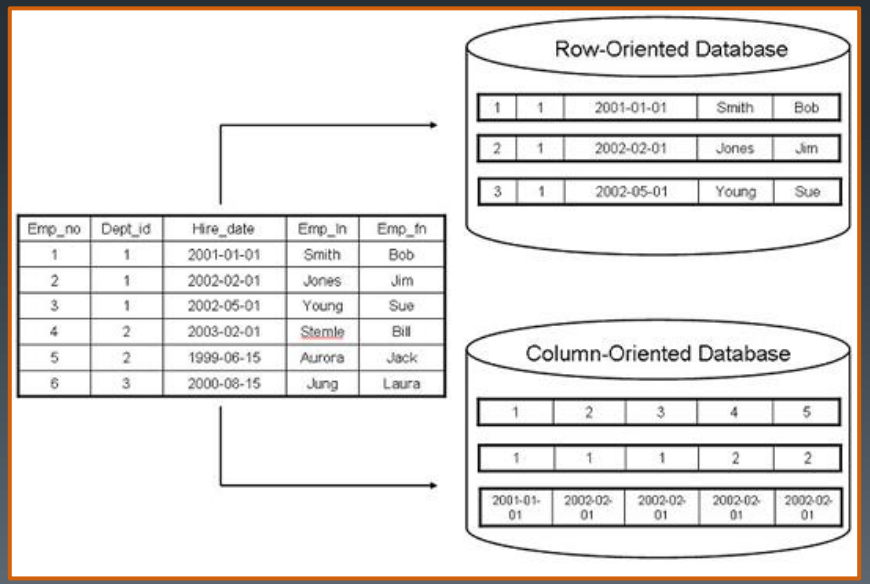
Column

- Google documented Bigtable in a famous white paper [Chang etc.]
  - Their core data store at the time (previously used for search)
  - Uses sparse columns and no schema
  - A two-level map
  - Influenced HBase and Cassandra
- Column Families
  - Stores groups of columns together
  - Each column has to be part of a single column family
  - Columns are the unit for access
  - Column stores make summing columns very fast due to column layout on disc versus row layout
  - Column families can be treated like tables
- Columns can be added freely
- Skinny Rows
  - Few columns
  - Same columns used across the many rows
  - The column family defines a record type
  - Each row is a record, and each column is a field
- Wide Rows
  - Have many columns (perhaps thousands), with rows having very different columns
  - A wide column family models a list, with each column being one element in that list
  - A consequence of wide column families is that a column family may define a sort order for its columns

Use:
- Event logging
- CMS/blogging
- Counters
- Expiring usage (columns)

Don't Use:
- Situations where the column families change frequently



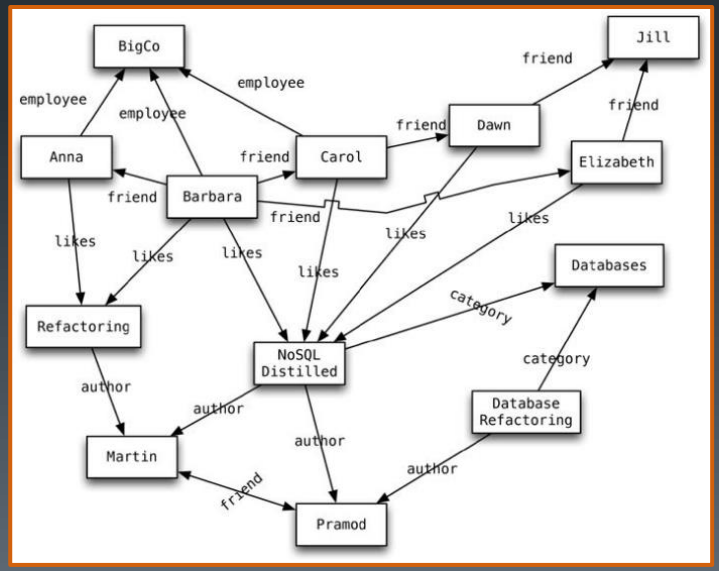


Graph

- Graph databases are motivated by a different frustration with relational databases and thus have an opposite model
  - Small records with complex interconnections
- Nodes & Edges (aka. Arcs)
- Queries like:
  - Select books in the Databases category written by someone whom a friend of mine likes
- Edge traversal is cheap
  - Graph databases shift most of the work of navigating relationships from query time to insert time
  - Good only if querying performance is more important than insert speed
- Usually single server based
- Examples
  - FlockDB - nodes and edges with no mechanism for additional attributes
  - Neo4J - attach Java objects as properties to nodes and edges
  - Infinite Graph - stores Java objects, which are subclasses of its built-in types, as nodes and edges

Use:
- Connected data (social graphs)
- Routing/dispatch (location based systems)
- Recommendation engines

Don't Use:
- Situations where multiple nodes must be updated



Summary

- Almost all cloud systems offer a wide range of storage solutions
- Infrastructure level solutions include:
  - Archival
  - Object
  - Block
  - File
- Application Platform level solutions include:
  - Relational Databases
  - NoSQL Databases
    - Key-Value
    - Document
    - Column
    - Graph

Lab 7

- Cloud Data Stores
  - Explore data storage in the cloud using Amazon Simple Storage Service (S3)
  - Work with DynamoDB to explore the nature of a cloud native datastore

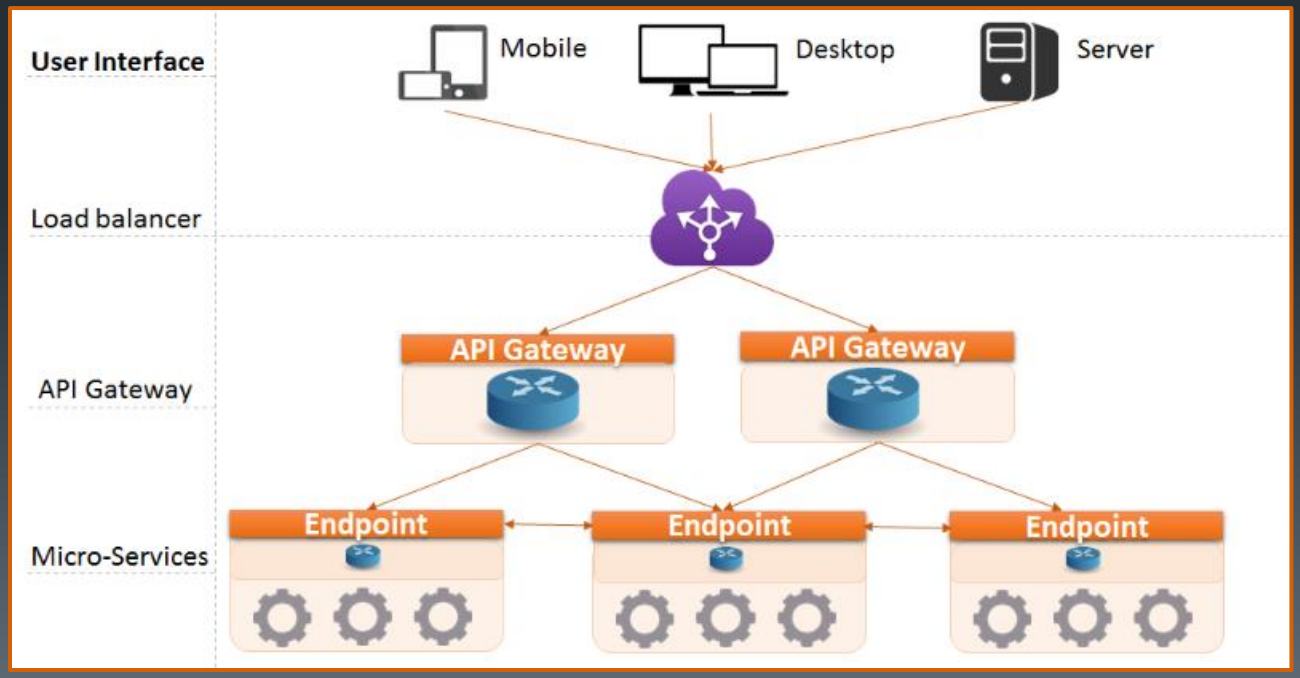
8: API gateways & FaaS

Objectives

- Define the role of the application gateway
  - Explain gateways and FaaS
- Discuss open source gateways and service mesh software
- Examine AWS API Gateway and AWS Lambda
- Introduce Google Functions, Azure Functions, Apache OpenWhisk and Serverless Frameworks

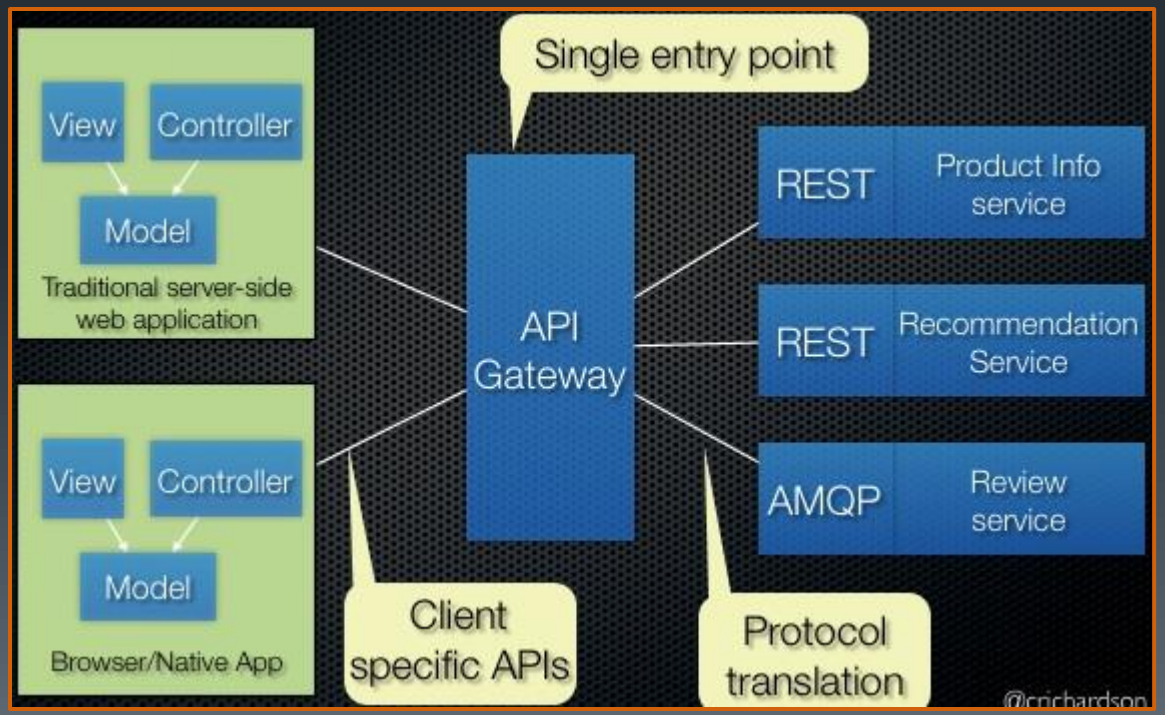
The Role of Gateways

- Microservice based applications must define explicit means for clients to interact with the application
- With a monolithic application there is just one set of (typically replicated, load-balanced) endpoints
- In a microservices architecture each microservice exposes a set of fine-grained endpoints
- API Gateways expose a perimeter API isolating the microservices within the bounded context of the application
- Clients use the API Gateway perimeter API to consume services



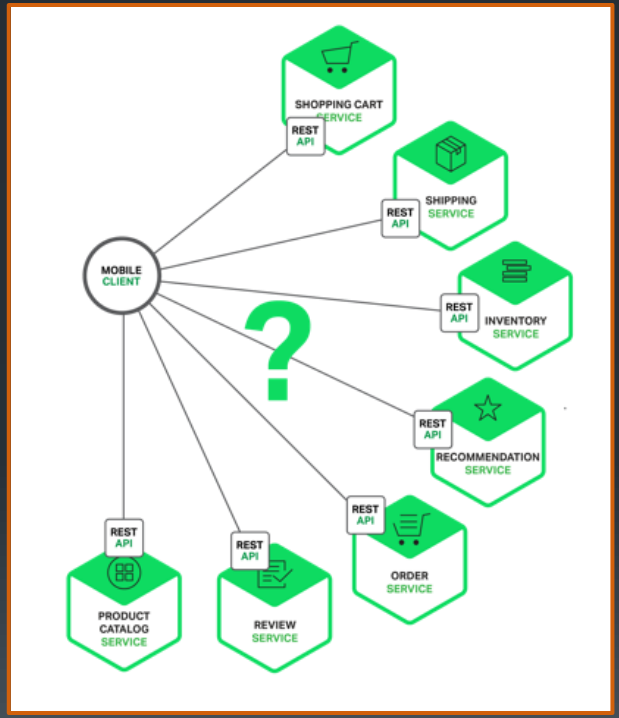
Gateway Features

- API Gateways should allow systems to rapidly change behavior in order to react to changing circumstances
- Key functions:
  - Authentication and Security - identifying authentication requirements for each resource and rejecting requests that do not satisfy them
  - Insights and Monitoring - tracking meaningful data and statistics at the edge in order to give an accurate view of production
  - Dynamic Routing - dynamically routing requests to different backend clusters as needed
  - Stress Testing - gradually increasing the traffic to a cluster in order to gauge performance
  - Circuit Breakers - allocating capacity for each type of request and dropping requests that go over the limit
  - Caching - building some responses directly at the edge instead of forwarding them to an internal cluster
  - Multiregion Resiliency – routing requests across regions in order to diversify usage and move the edge closer to clients



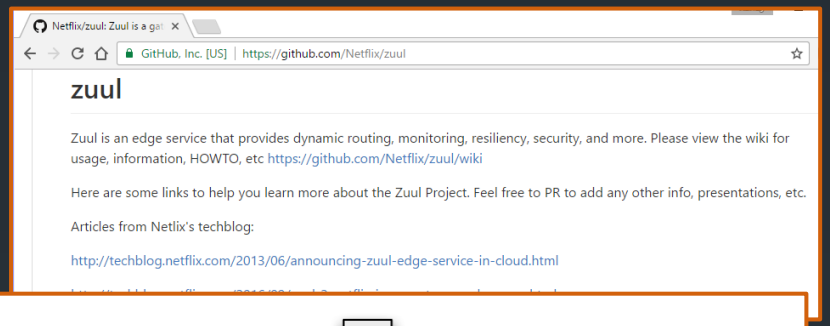
Ingress Integration Anti-patterns

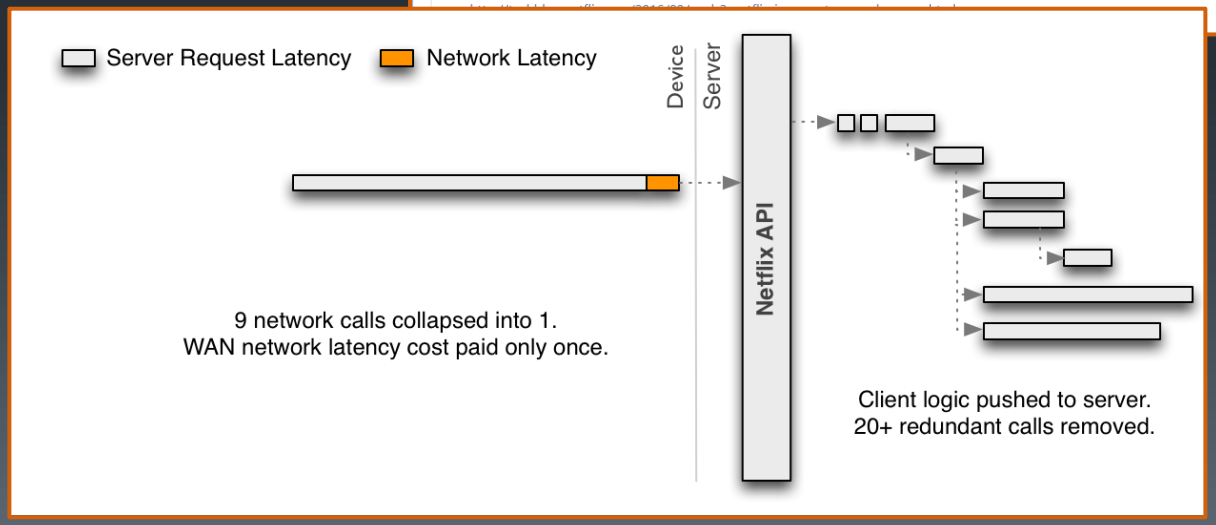
- Direct Client-to-Microservice Communication anti-pattern
  - A client could make requests to each microservice directly
  - Each microservice would have a public endpoint (https://serviceName.api.company.name) mapped to the microservice's load balancer
- Challenges and limitations with this option:
  - The mismatch between the needs of the client and the fine-grained APIs exposed by each of the microservices
    - The client has to make many separate requests to perform an operation
  - Chatty services in a 10Gb data center may work well, but they almost always produce problems over the slower, less reliable Internet
  - Microservices may use protocols that are not web-friendly (Thrift, AMQP messaging protocol, etc.)
- Huge drawback with this approach is that it makes it difficult to refactor the microservices
  - How the system is partitioned is now the client's concern (!)
  - Information about the implementation of services should never escape the enclosing bounded context
- It rarely makes sense for clients to talk directly to microservices



Netflix Gateway Example

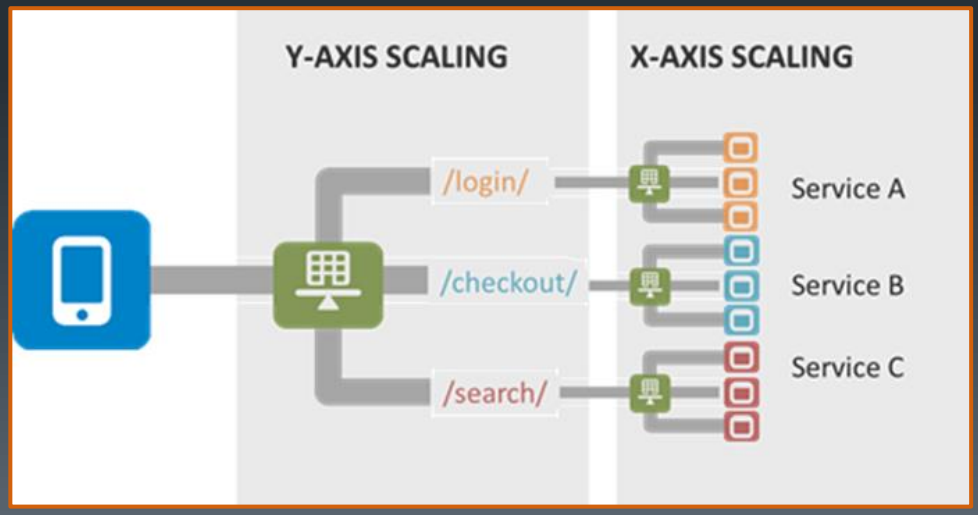
- API Gateways are responsible for:
  - Request routing
  - Composition
  - Protocol translation
- Typically requests from external clients go through an API Gateway
  - Often requests invoke multiple microservices requiring data aggregation
- An example API Gateway is the Netflix API Gateway Zuul
  - Netflix streaming service is available on hundreds of different kinds of devices including televisions, set-top boxes, tablets, phones, etc.
    - Handles billions of requests per day
  - Netflix attempted to provide a one-size-fits-all API for their streaming service, discovered it didn't work because device diversity and their unique needs
  - The Zuul API Gateway provides an API tailored for each device by running device-specific adapter code
  - An adapter typically handles each request by invoking on average six to seven backend services





Service Constructs and Endpoints

- Services are conceptual in cloud native systems
  - Containers implement services
  - Orchestration systems start and stop additional containers for a service as load dictates
  - Load balancing mechanisms distribute clients across the available containers
  - Endpoints define the connection target for clients
    - Endpoints are manifested by real or virtual Load Balancers
      - VIP
      - Per Node Ports
      - Netflix Ribbon
      - AWS Elastic/Application Load Balancer
      - Google Cloud Load Balancer
      - Nginx
      - HA Proxy
      - Traefik
      - Etc.

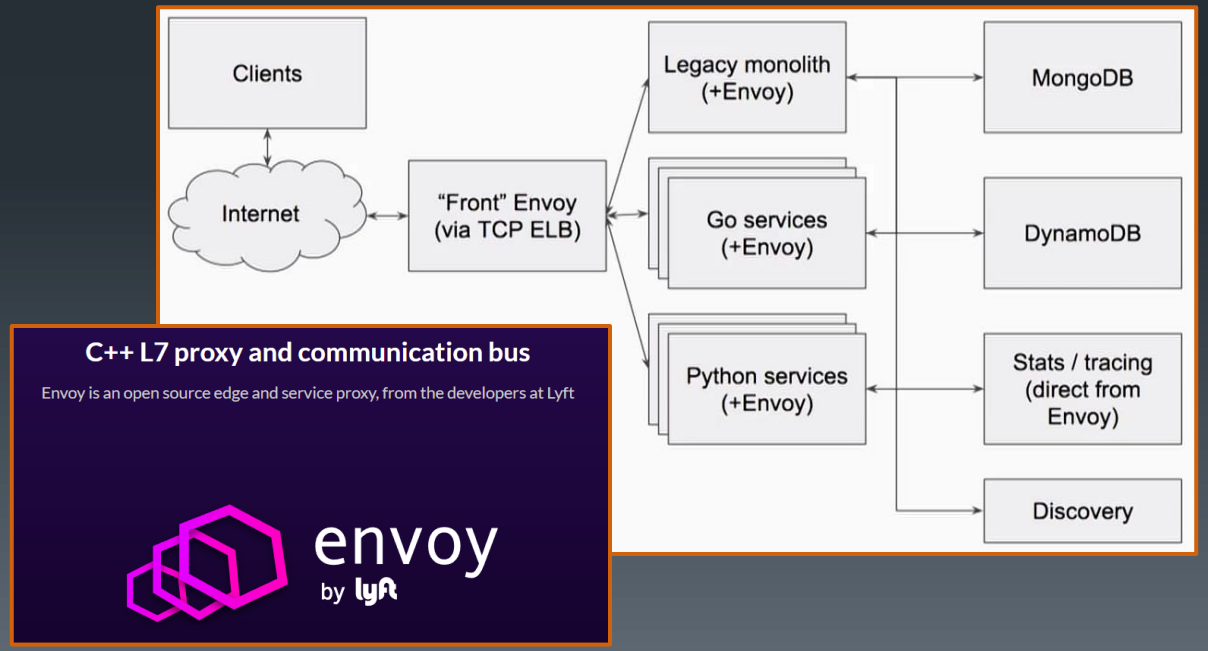


Envoy

- A self contained OSS network service proxy

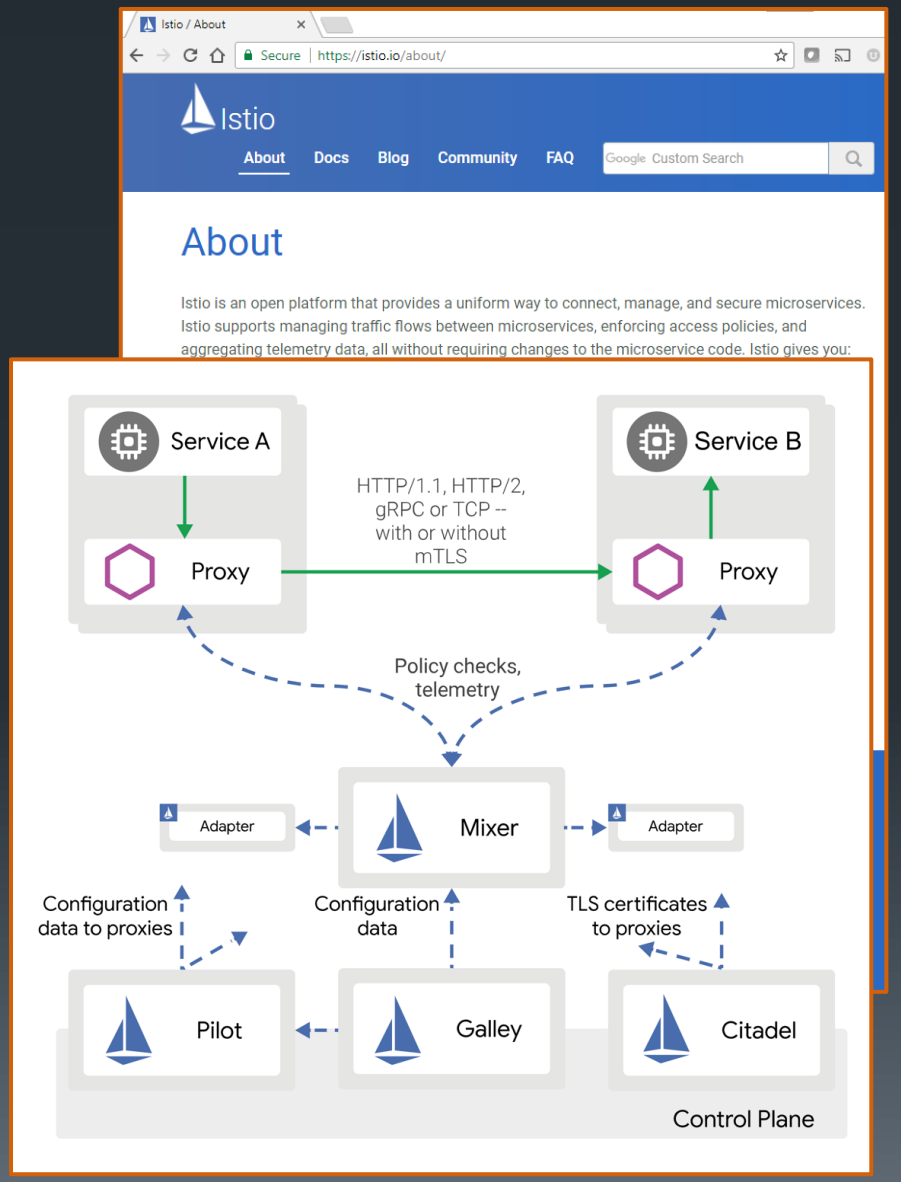
- HTTP/2
  - First class in/out support for HTTP/2 and transparent HTTP/1.1 to HTTP/2
- Load balancing
  - Automatic retries
  - Circuit breaking
  - Global rate limiting
  - Request shadowing (copy requests to QA)
  - Zone local LB (avoids cross zone fees)
- Filtering
  - Envoy allows HTTP/TCP/IP filtering
  - Filters can be chained
  - New filters can be written
- Coded in C++11
- gRPC support
- HTTP routing
  - Redirection, virtual hosts, virtual clusters, matching on different request parameters
- TLS support
  - Termination and initiation, client certificate verification, and certificate pinning
- Observability
  - Envoy exposes rich statistics and distributed tracing via third party providers (e.g. light step, zipkin)

- MongoDB
  - Envoy contains a MongoDB wire format parser used to gather statistics about database connections
- DynamoDB
  - Envoy contains a DynamoDB API parser that is used to gather statistics about database requests and responses
- Service discovery
  - Envoy supports asynchronous DNS resolution as well as integration with an external service discovery service
- Health checking
  - Envoy is capable of active health checking of backend servers
  - Active health checking along with service discovery yields eventually consistent and extremely resilient load balancing



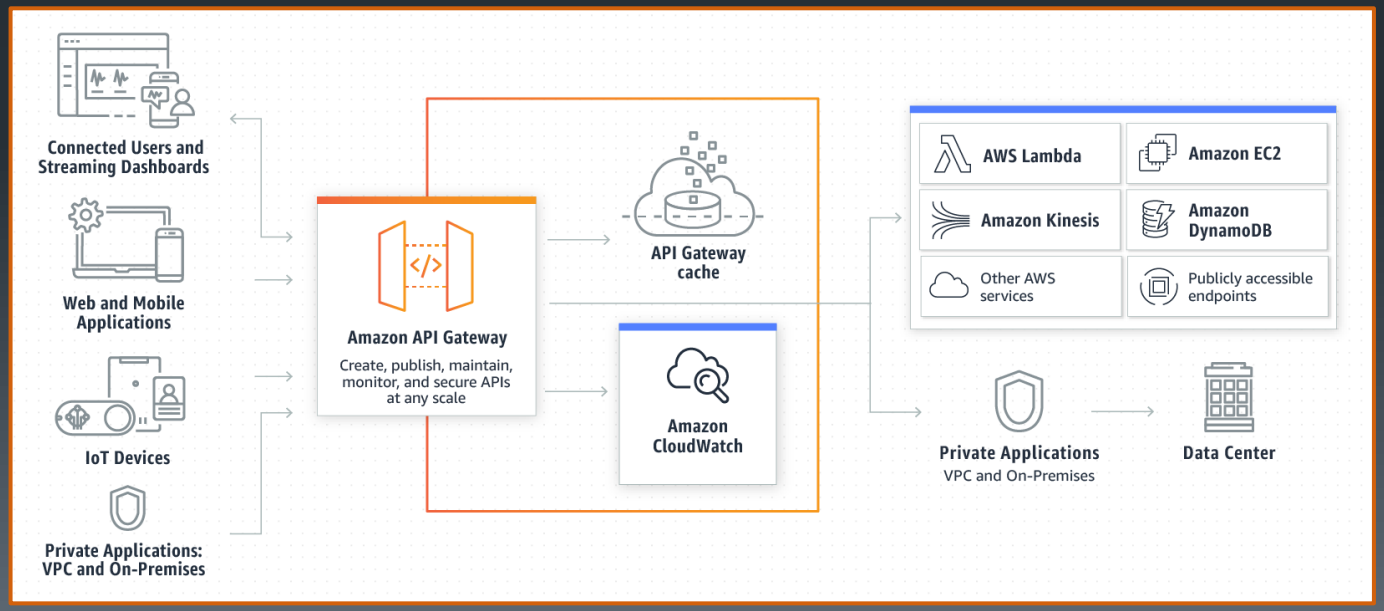
Istio

- In a phrase: Kubernetes integrated Envoy
- Istio supports policy based traffic management between microservices
- Istio generates telemetry data
- Istio is transparent to the actual microservices it intermediates
- Features:
  - Automatic load balancing for HTTP, gRPC, and TCP traffic
  - Fine-grained control of traffic behavior with rich routing rules, retries, failovers, and fault injection
  - A pluggable policy layer and configuration API supporting access controls, rate limits and quotas
  - Automatic metrics, logs, and traces for all traffic within a cluster, including cluster ingress and egress
  - Secure service-to-service authentication with strong identity assertions between services in a cluster
- Supports Kubernetes, Nomad and Consul
  - Support is planned for Cloud Foundry, Mesos, and bare metal
- Components:
  - Envoy service proxy
  - Mixer collects telemetry and enforces service mesh policy
  - Pilot is an interface for users to configure Istio
  - Citadel provides service-to-service and end-user authentication using mutual TLS
  - Galley is Istio's configuration validation, ingestion, processing and distribution component
    - Responsible for insulating the rest of the Istio components from the details of obtaining user configuration from the underlying platform (e.g. Kubernetes)



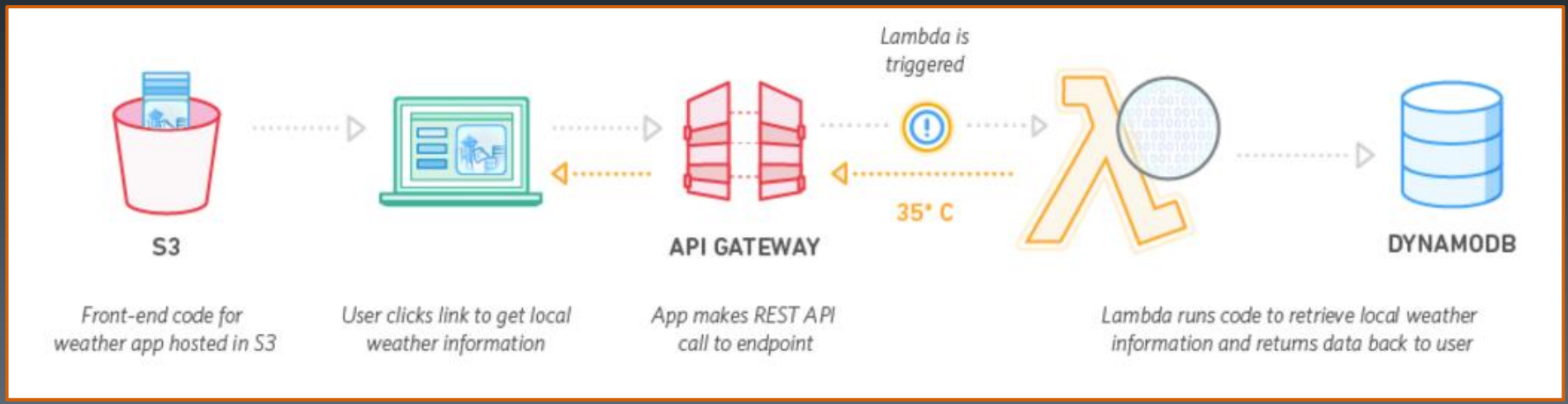
AWS API Gateway

- A managed service that makes it easy for developers to create, publish, maintain, monitor, and secure APIs at scale
- In Amazon API Gateway, an API refers to a collection of resources and methods that can be invoked through HTTPS endpoints
- Support for:
  - Amazon Elastic Compute Cloud (Amazon EC2)
  - AWS Lambda
- Handles:
  - Traffic management
  - Authorization
  - Access control
  - Monitoring
  - API version management
- Charges for the API calls received and data transferred out



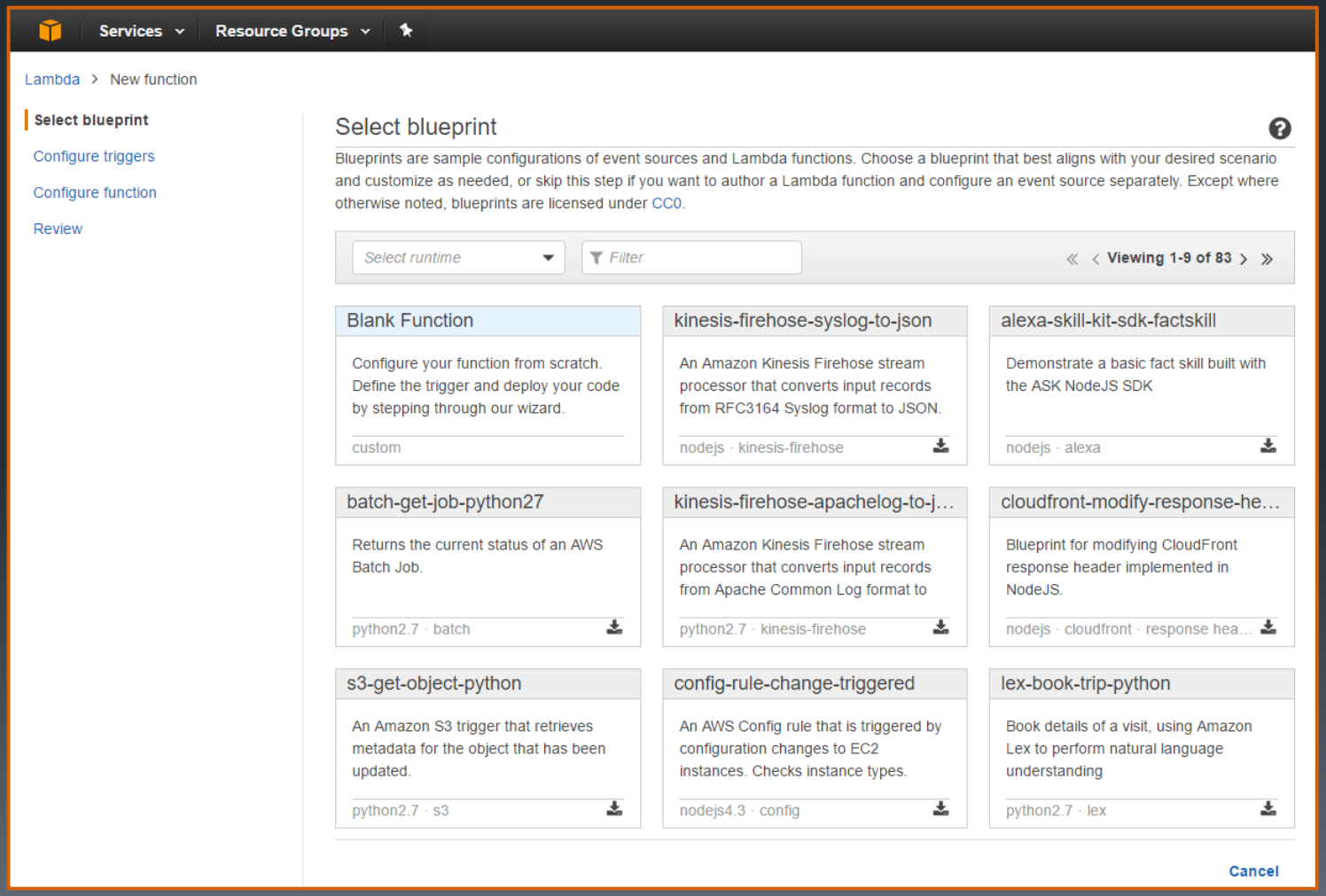
AWS Lambda

- AWS Lambda lets you run code without provisioning or managing servers
- Pay for the compute time you consume
- Lambda can run code for virtually any type of application or backend service with zero administration
- Upload code and Lambda takes care of everything required to run and scale the code with high availability
- You can set up code to automatically trigger from other AWS services or call it directly from any web or mobile app



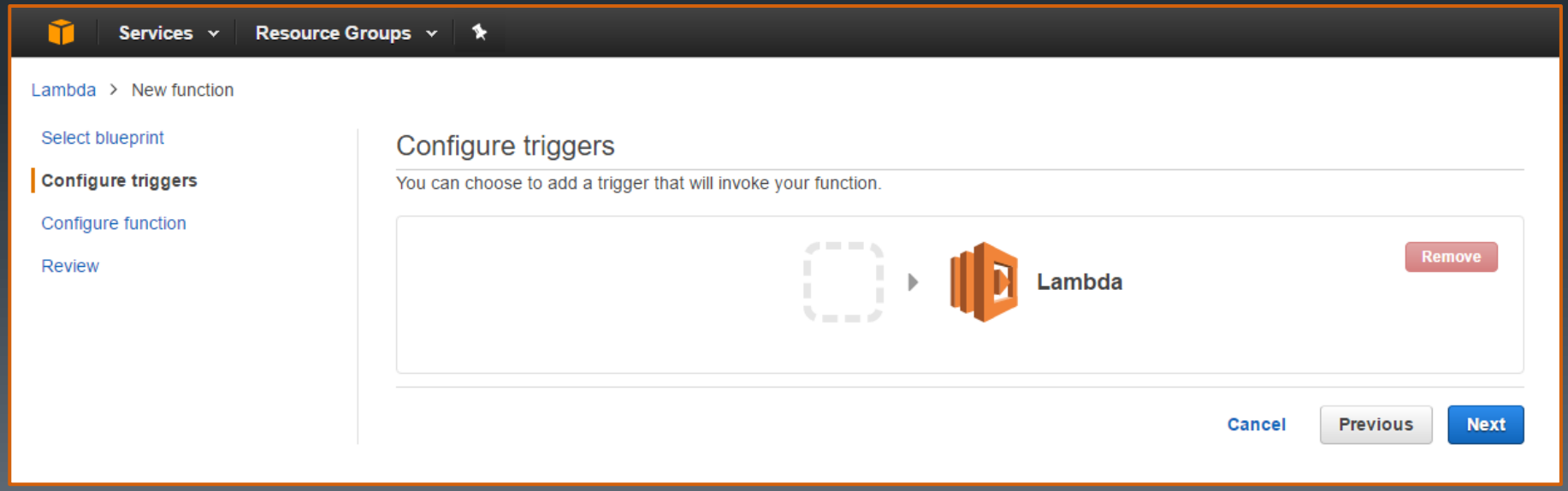
AWS Blueprints

- AWS offers over 80 Lambda blueprints to help you quickly wire functions to other AWS services



AWS Triggers

- Lamdas can be invoked by one or many events within AWS
  - Amazon S3
  - Amazon DynamoDB
  - Amazon Kinesis Streams
  - Amazon Simple Notification Service
  - Amazon Simple Email Service
  - Amazon Cognito
  - AWS CloudFormation
  - Amazon CloudWatch Logs & Events
  - AWS CodeCommit
  - Scheduled Events (powered by Amazon CloudWatch Events)
  - AWS Config
  - Amazon Echo
  - Amazon Lex
  - Amazon API Gateway
  - Invoked On Demand

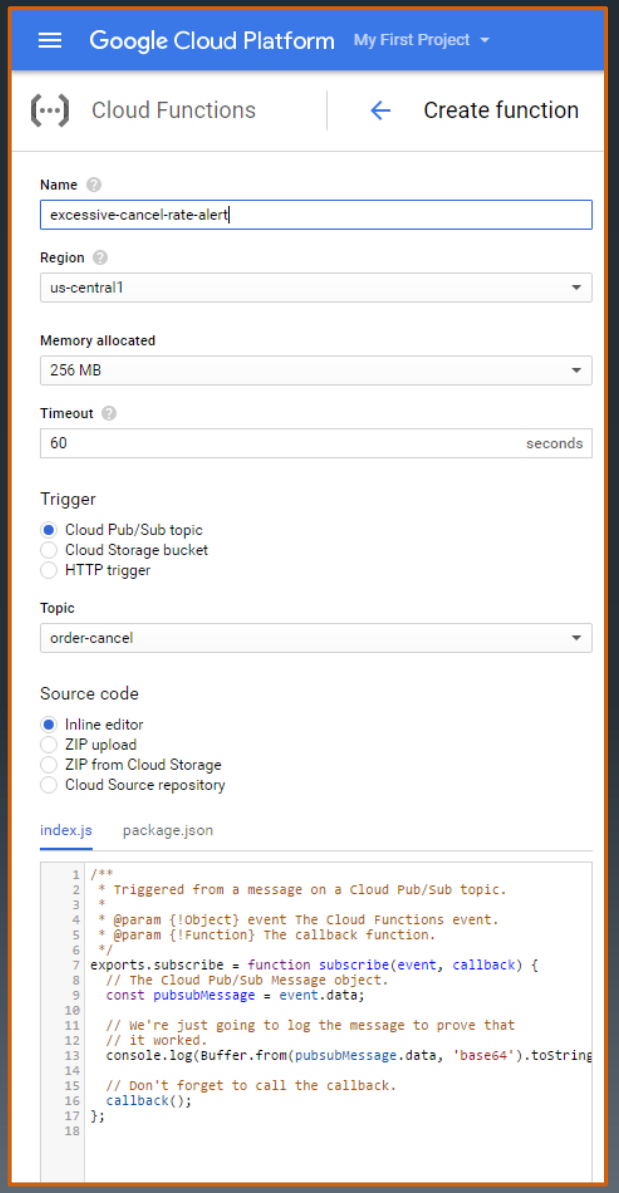


Google Cloud Functions

- Lightweight compute solution for developers to create single-purpose, stand-alone functions that respond to cloud events without the need to manage a server or runtime environment
- You can invoke Cloud Functions with an HTTP request using the POST, PUT, GET, DELETE, and OPTIONS HTTP methods
  - To create an HTTP endpoint for your function, you specify `--trigger-http` as the trigger type when deploying your function
  - From the caller's perspective, HTTP invocations are synchronous, meaning that the result of the function execution will be returned in the response to the HTTP request

- `gcloud beta functions deploy helloHttp \ --stage-bucket cloud-functions --trigger-http`

- `curl -X POST https://<YOUR_REGION>-<YOUR_PROJECT_ID>.cloudfunctions.net/helloHttp -H "Content-Type:application/json" --data '{"name":"Keyboard Cat"}'`



Economics of FaaS

- Pay for use while not idle!
  - 1 GB-second is 1 second of wallclock time with 1GB of memory provisioned
  - 1 GHz-second is 1 second of wallclock time with a 1GHz CPU provisioned

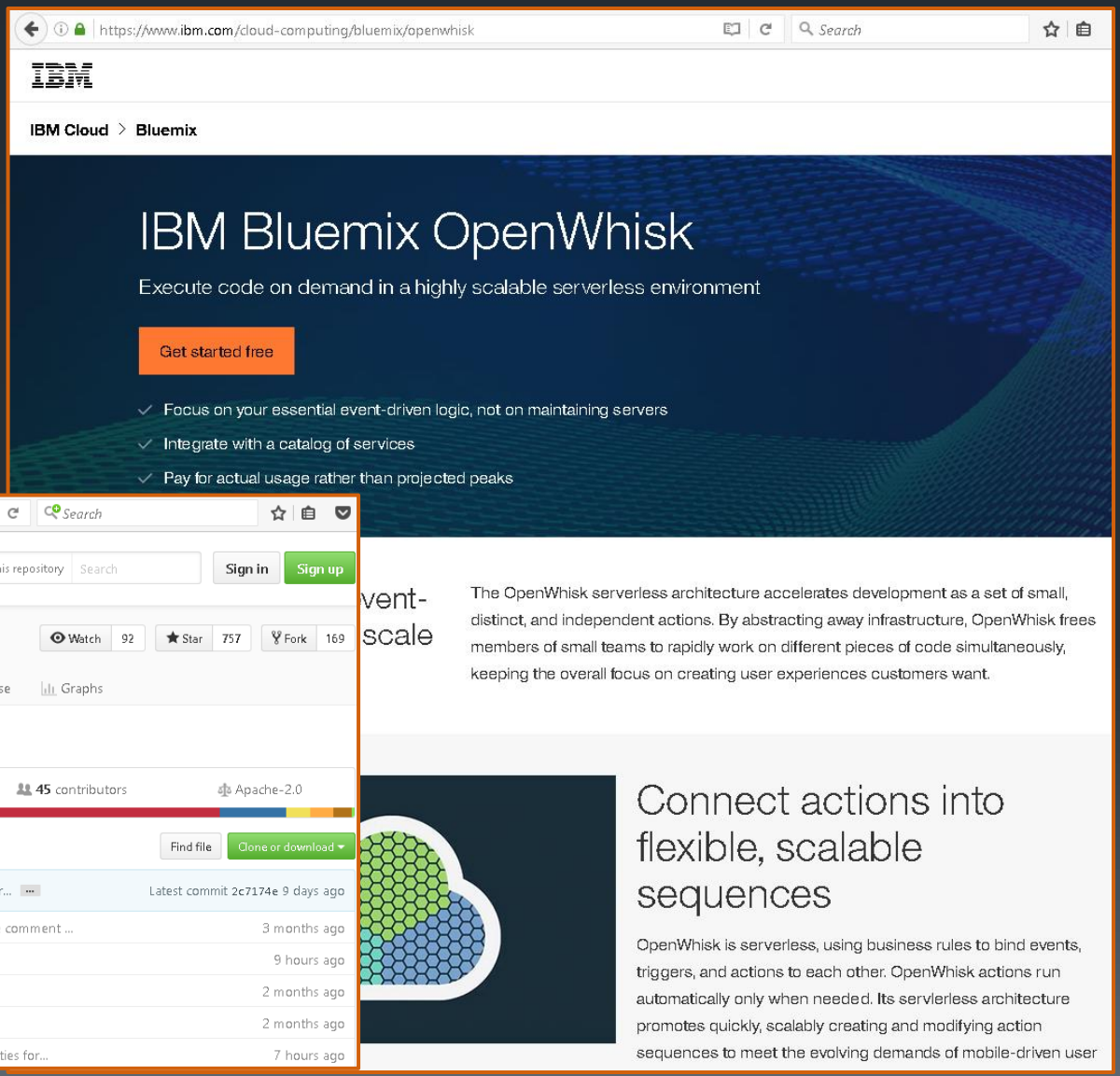
CLOUD FUNCTIONS PRICING

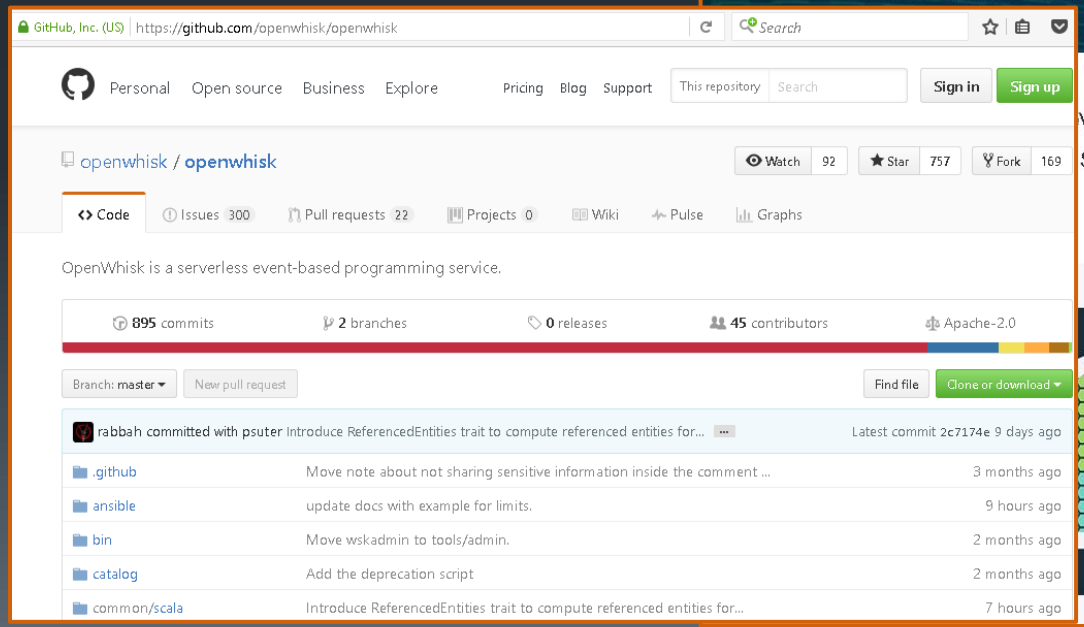
Google Cloud Functions charges for invocations, compute time, and outbound data. Inbound data, and outbound data to other Google APIs in the same region is free. For detailed pricing information, please view the pricing guide.

	FREE *LIMIT PER MONTH*	PRICE *ABOVE FREE LIMIT (PER UNIT)*	PRICE UNIT
Invocations *	2 million invocations	$0.40	per million invocations
Compute Time	400,000 GB-seconds	$0.0000025	per GB-Second
	200,000 GHz seconds	$0.0000100	per GHz-Second
Outbound Data (Egress)	5GB	$0.12	per GB
Inbound Data (Ingress)	Unlimited	Free	per GB
Outbound Data to Google APIs in same region	Unlimited	Free	per GB

Apache OpenWhisk

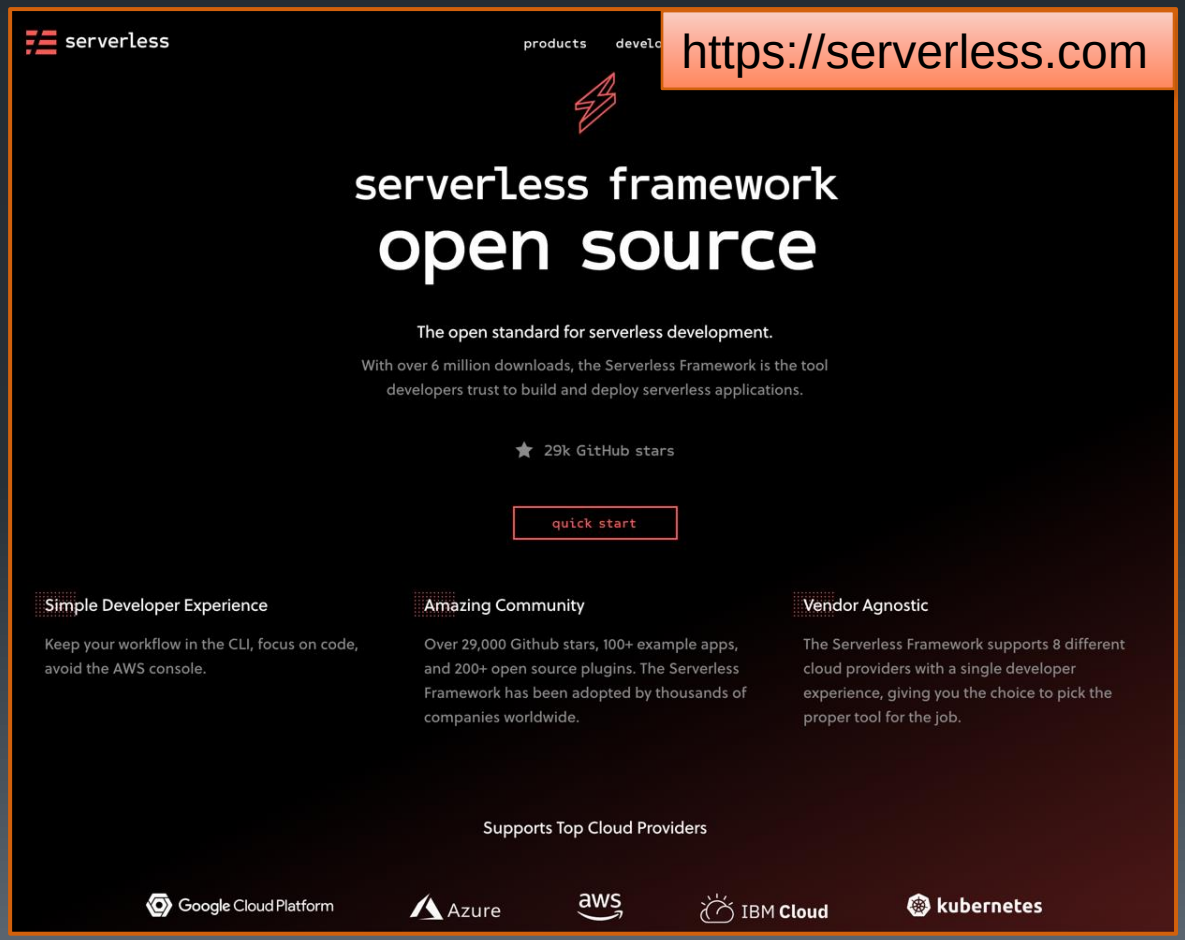
- An open source lambda platform
  - Was IBM OpenWhisk
- Runs on IBM Cloud, OpenStack, AWS, Vagrant etc.
- Can execute containers
  - No language restrictions





Serverless Framework

- Helps you develop and deploy your functions, along with the infrastructure resources they require
- A CLI that offers structure, automation and best practices
- The Serverless Framework is different than other application frameworks because it manages your code as well as your infrastructure, configuring and wiring events to functions
- Cloud Agnostic – share cross-cloud functions and events with AWS Lambda, Microsoft Azure, IBM OpenWhisk and Google Cloud Platform



Summary

- Cloud Gateways offer hosted perimeter ingress services
- Gateways can perform a wide array of important services
  - Authentication
  - Call routing
  - Monitoring
  - Etc.
- FaaS solutions offer functions as a service
- FaaS combined with gateway services allow users to create serverless applications over the Internet

Lab 8

- Building application gateways with an open source API Gateway
  - Explore interoperability between two cloud hosted services with API Gateways

Thank you for your time!