CLOUD COMPUTING APPLICATIONS

Cloud Computing Glue: Introduction

Prof. Reza Farivar

Cloud Computing Glue

- Cloud Computing is all about running your compute tasks and storing your data on other computers
- At the core, it's about communication between computing sources
  - Your client + some server on the cloud, through internet
- Have we seen this problem before?
  - Inter-process communication
  - Communication on a local network
  - Communication on internet

Overview

1. Intro
2. Communication
3. Internet Protocol, HTTP and RTC on HTTP
4. Service Oriented Architecture and SOAP
5. RESTful Architecture
6. Asynchronous RPC, WebSocket
7. HTTP2 Push, Streaming Video
8. JSON, comparison with XML
9. RPC semantics and implementation
10. Protocol Buffers and Thrift

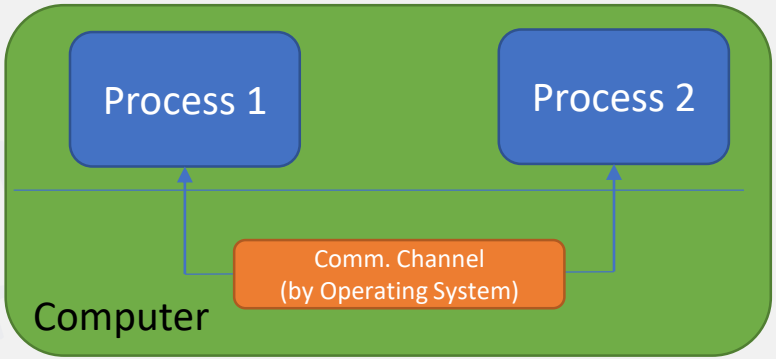
CLOUD COMPUTING APPLICATIONS

Cloud Computing Glue: Communication

Prof. Reza Farivar

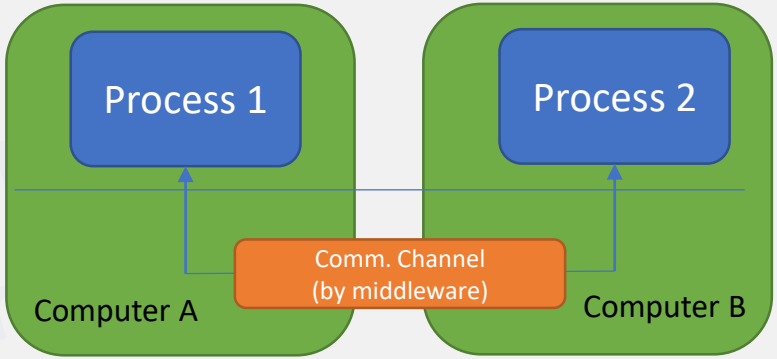
Communication in a single machine

- Communication Channel provided by the Operating System
- Shared memory block
- Shared File System
- Signal
- POSIX Socket, aka. Berkeley Socket
  - Port numbers
  - *SOCK_STREAM (compare to TCP)*
  - *SOCK_DGRAM (compare to UDP)*
- Remote Method Invocation (RMI)
  - Method invocations between objects *in different processes (processes may be on the same or different host)*
  - *From one JVM to another*
- Message Queue
- Message Passing
  - Unix Pipe
  - *Actor Model*
  - *Pi Calculus*

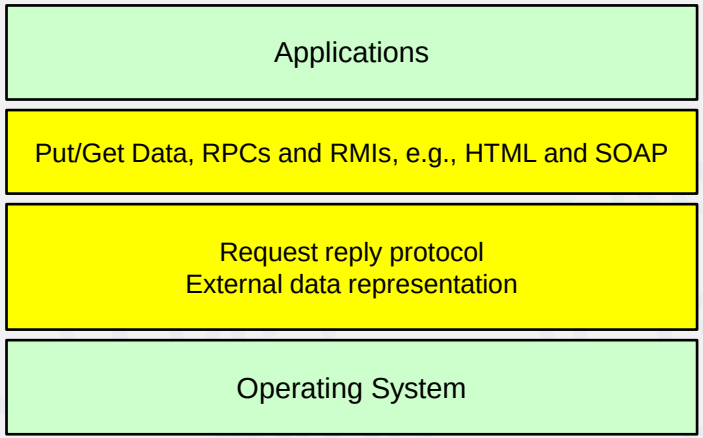


Middleware Layer Definition

- *Software that provides services to applications beyond those generally available at the operating system*
- Middleware implements functionalities that are common across many different applications
  - No need to reinvent the wheel (e.g., message parsing) every time you need to do something
- A Middleware Layer can provide the same abstractions to distributed applications!
  - Building distributed systems while maintaining our code is not very different from a single-node program



Middleware Layers



Middleware layers

Provide support to the application

Run at all servers at user level

RMI = Remote Method Invocation
CORBA = Common Object Request Brokerage Architecture
SOAP = Simple Object Access Protocol

Communication in a local network

- Scientific Computing
  - Message Passing Interface (MPI)
  - Simple model: Send() and Receive()
  - No native support for fault tolerance
  - Programming interface is complicated
    - Race, deadlock, etc.
- Business Sector
  - Remote Procedure Calls
    - RPC Semantics (behavior in presence of network failures)
    - RPC Implementation
  - Remote Method Invocation (RMI)
    - Between two JVMs on a network

Communication in Big Data Deployments

- Need scaling from the start
  - Sometimes many 10s of thousands of nodes on the network
  - Ad hoc solutions do not work
- RPC Frameworks
  - Google Protocol Buffer
    - You define the functions that will be called remotely
    - Then Compile
    - The system automatically generates interfaces functioning as communication stubs in your choice of programming language
      - Many languages are supported: C++. C#, Objective C, Java, Python, etc.
    - You import the generated header / code files into your project
    - At run-time, your program just calls the function locally. The auto-generated code takes care of serialization and marshalling of the function parameters, making the network calls (handling any network errors), and transferring the function call in the target system.
  - Apache Thrift
    - Apache HDFS, Hadoop, Spark, Storm, etc. extensively use Thrift
- Consistency
  - Paxos
  - Zookeeper

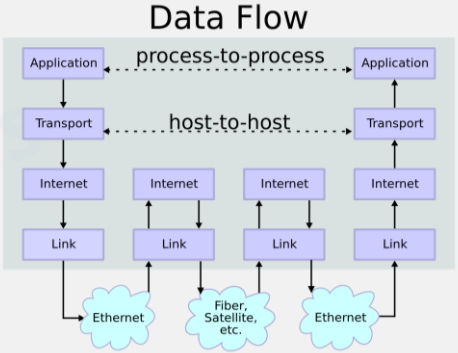
CLOUD COMPUTING APPLICATIONS

Cloud Computing Glue: Internet Protocol, HTTP and RPC on HTTP

Prof. Reza Farivar

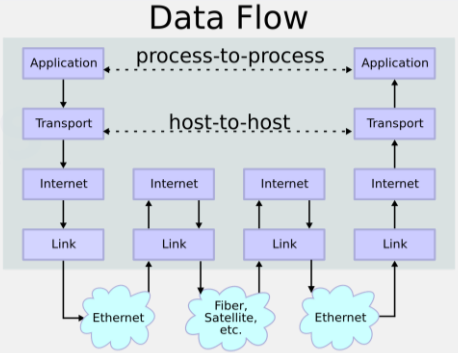
Internet Protocol Stack: Link and Internet Layers

- The second level, “internet level”, is where internet switches and routers operate
  - *Gets your message from one machine to another*
  - Responsible for addressing **host interfaces**
  - Encapsulating data into datagrams (including fragmentation and reassembly)
  - Routing datagrams across one or more IP networks
  - *IP address, IPV4, IPV6*
- EE and Network engineers live at the bottom level, the Link level
  - Out of scope our discussion



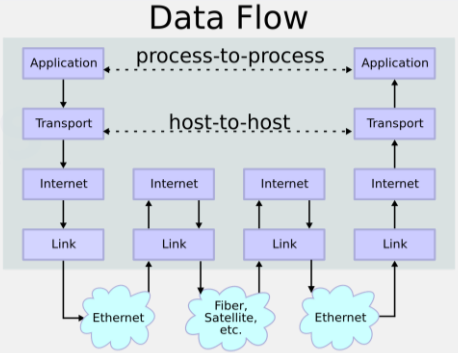
Internet Protocol Stack: Transport Layer

- The transport level handles packetizing of data
- Gets your message from a process in one machine to another process in another machine
- Typically implemented in the Operating System
- Provide port numbers (similar to Unix sockets)
- TCP
  - Keeps track of data segments, retransmission, acknowledgement
    - Handle network congestion
    - Traffic load balancing
    - Unpredictable network behavior
    - Lost, duplicated, or delivered out of order IP packets
  - Guarantees that all bytes received will be identical and in the same order as those sent
- UDP
  - “User Datagram Protocol”, aka. “Unreliable Datagram Protocol”
  - Very simple



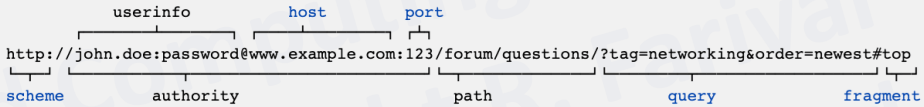
Internet Protocol Stack: Application Layer

- Application level handles “what to send”
  - HTTP, HTTPS
    - RESTful APIs
  - FTP
  - WebSocket
  - SMTP
  - IMAP
  - SSH
  - DHCP
  - DNS
  - Bit Torrent



HTTP Protocol

- Originally targeted static web pages: *Client Requests, Server Responds, Connection is closed*
- Works on top of TCP for reliable transport
- Client (user agent) can be a web browser, or any other software
- HTTP resources are identified and located on the network by Uniform Resource Locators (URLs), using the Uniform Resource Identifiers (URI's) schemes http and https



- In HTTP/1.0 a separate connection to the same server is made for every resource request
  - Establishment of TCP connections has overhead
- HTTP/1.1 can reuse a connection multiple times to download images, scripts, stylesheets, etc. after the page has been delivered
  - Persistent Sessions
  - Lower latency

HTTP Message Format

- Request “verbs”
  - GET: retrieve data
  - POST: server should accept the call parameter as a new value for the resource specified in the URL
  - PUT: server should store the enclose entity under the supplied URL
  - DELETE: server should delete the specified resource by the URL
  - PATCH: server should apply partial modification to the resource
  - …
- Request message
  - a request line (e.g., GET /dataset/inventory.htm HTTP/1.1, which requests /dataset/inventory.htm resource from the server)
  - request header fields
  - an empty line
  - an optional message body
- Response message
  - a status line which includes the status code and reason message (e.g., HTTP/1.1 200 OK, which indicates that the client's request succeeded)
    - *Informational 1XX | Successful 2XX | Redirection 3XX | Client Error 4XX | Server Error 5XX*
  - response header fields (e.g., Content-Type: text/html)
  - an empty line
  - an optional message body

Client Request

```
GET / HTTP/1.1
Host: www.example.com
```

Session

Server Response

```
HTTP/1.1 200 OK
Date: Mon, 23 May 2005 22:38:34 GMT
Content-Type: text/html; charset=UTF-8
Content-Length: 138
Last-Modified: Wed, 08 Jan 2003 23:11:55 GMT
Server: Apache/1.3.3.7 (Unix) (Red-Hat/Linux)
ETag: "3f80f-1b6-3e1cb03b"
Accept-Ranges: bytes
Connection: close

<html>
  <head>
    <title>An Example Page</title>
  </head>
  <body>
    <p>Hello World, this is a very simple HTML document.</p>
  </body>
</html>
```

RPC on HTTP

- Remote Procedure Calls built on HTTP
  - For many types of RPC, the client/server conversation model of HTTP works just fine
  - Just replace the HTML markup with an XML or JSON representation of the data
  - XML-RPC
  - JSON-RPC
    - E.g. Bitcoin server
    - Commands encoded as JSON, sent over HTTP

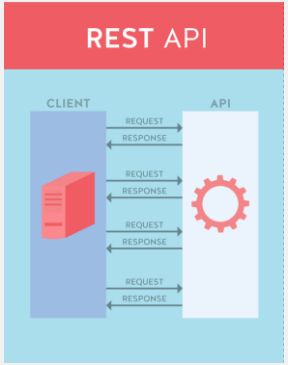
CLOUD COMPUTING APPLICATIONS

Cloud Computing Glue: RESTful Architecture

Prof. Reza Farivar

Representational State Transfer (REST)

- A style of software architecture for distributed hypermedia systems such as the World Wide Web
- Introduced in the doctoral dissertation of Roy Fielding
  - One of the principal authors of the HTTP specification
- The motivation for REST was to capture those characteristics of the Web that made the Web successful
  - URI-addressable resources
  - HTTP
  - Make a request – receive response – display response
- A collection of network architecture principles that outline how resources are defined and addressed
  - Based on HTTP methods to access resources via URL-encoded parameters and the use of JSON or XML to transmit data
  - Request/response between client and server, like a conversation
  - Something is requested, something is done, and then something is sent in return



RESTful API

- Uses HTTP verbs: GET, POST, PUT, PATCH, DELETE
  - Exploits the use of the HTTP beyond HTTP POST and HTTP GET
    - HTTP PUT and DELETE are not even supported in HTML
  - GET is safe (does not change state)
  - GET, PUT and DELETE are idempotent (you can execute them more than once and get the same state change result)
  - Example request:
    - `curl -X POST https://api.github.com/user/repos`
    - Response:

```
{
  "message": "Requires authentication",
  "documentation_url": "https://developer.github.com/v3/repos/#create"
}
```

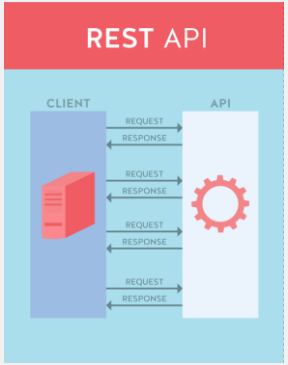


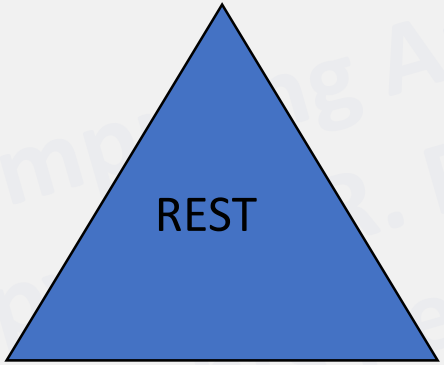
Image from https://nordicapis.com/rest-vs-streaming-apis-how-they-differ/

REST – Not a Standard

- There is no “official standard”, REST is an architectural style
  - JSR 311: JAX-RS: The Java™ API for RESTful Web Services
- But it uses several standards:
  - HTTP
  - URL
  - XML/HTML/GIF/JPEG/etc. (resource representations)
  - Text/xml, text/html, image/gif, image/jpeg, etc. (resource types, MIME types)
- Huge adoption for “Web mashup” applications, operations on When entities
- Many cloud SaaS and PaaS services
- LinkedIn, Twitter,

Main Concepts

Nouns (resources)
unconstrained
i.e., http://example.com/employees/12345



Verbs
constrained
i.e., GET

Representations
constrained
i.e., XML

Resources

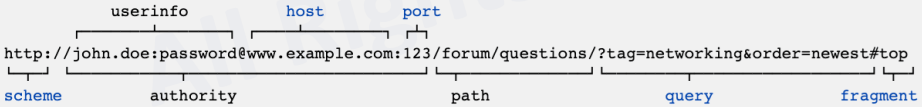
- The key abstraction of information in REST is a resource

- A resource is a conceptual mapping to a set of entities
  - Any information that can be named can be a resource: a document or image, a temporal service (e.g., "today's weather in Los Angeles"), a collection of other resources, a non-virtual object (e.g., a person), and so on

- Represented with a global identifier (URI in HTTP)
  - http://www.boeing.com/aircraft/747

Naming Resources

- REST uses URI to identify resources
  - http://localhost/books/
  - http://localhost/books/ISBN-0011
  - http://localhost/books/ISBN-0011/authors
  - http://localhost/classes
  - http://localhost/classes/cs2650
  - http://localhost/classes/cs2650/students
- As you traverse the path from more generic to more specific, you are navigating the data



Verbs

- Represent the actions to be performed on resources

- HTTP GET
- HTTP POST
- HTTP PUT
- HTTP DELETE
- HTTP PATCH

HTTP GET

- How clients ask for the information they seek
- Issuing a GET request transfers the data from the server to the client in some representation

- GET http://localhost/books
  - Retrieve all books
- GET http://localhost/books/ISBN-0011021
  - Retrieve book identified with ISBN-0011021
- GET http://localhost/books/ISBN-0011021/authors
  - Retrieve authors for book identified with ISBN-0011021

HTTP POST, HTTP PUT

- HTTP POST creates a resource
- HTTP PUT updates a resource

- POST http://localhost/books/
  - Content: {title, authors[], …}
  - Creates a new book with given properties

- PUT http://localhost/books/isbn-111
  - Content: {isbn, title, authors[], …}
  - Updates book identified by isbn-111 with submitted properties

HTTP DELETE

- Removes the resource identified by the URI

- DELETE http://localhost/books/ISBN-0011
  - Delete book identified by ISBN-0011

Representations

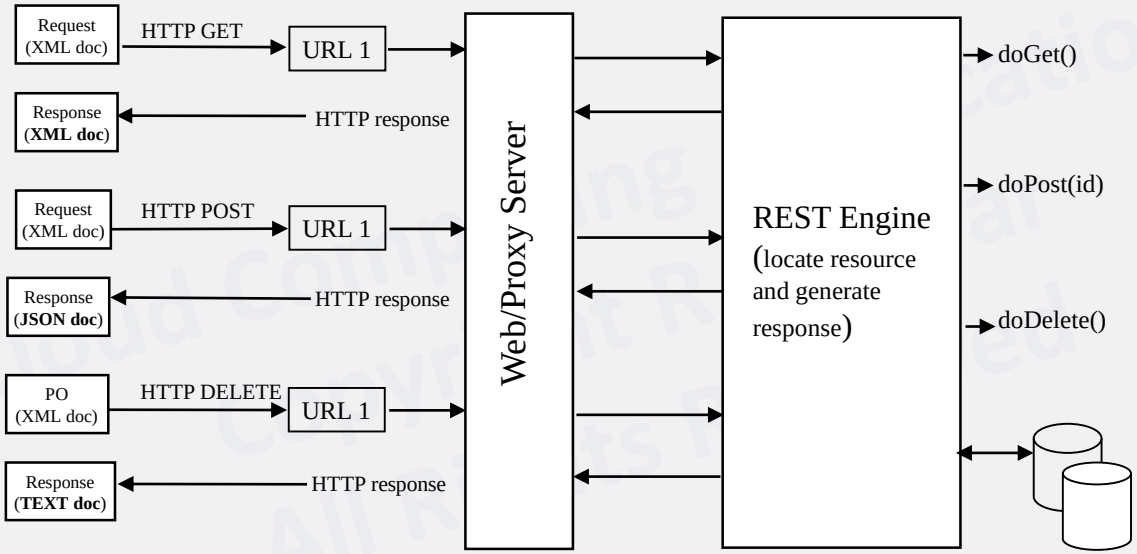
- How data is represented or returned to the client for presentation
- Two main formats:
  - JavaScript Object Notation (JSON)
  - XML
- It is common to have multiple representations of the same data
- XML

```
<COURSE>
      <ID>CS2650</ID>
      <NAME>Distributed Multimedia Software</NAME>
</COURSE>
```

- JSON

```
{course
      {id: CS2650}
      {name: Distributed Multimedia Software}
}
```

Architecture Style



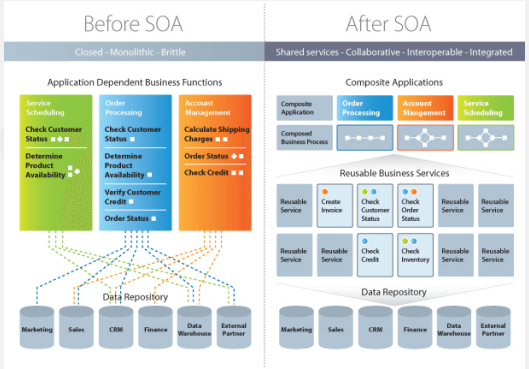
CLOUD COMPUTING APPLICATIONS

Cloud Computing Glue: Service Oriented Architecture and SOAP

Prof. Reza Farivar

Service Oriented Architecture

- Came out of the needs of the business sector, enterprise and B2B applications
  - *"SOA is the philosophy of encapsulating application logic in services with a uniformly defined interface and making these publicly available via discovery mechanisms."*
- Benefits of SOA
  - Reusable Code
  - Interaction
  - Scalability
  - Reduce Costs
- The term "Web Services" typically relates to this type of communication
  - Web Services are one option to implement SOA
  - Other options include: Java Business Integration (JBI), Windows Communication Foundation (WCF) and data distribution service (DDS)
- An example Technology was / is SOAP



Simple Object Access Protocol (SOAP)

- SOAP-based Web APIs use XML validation to ensure structural message integrity
  - XML schemas provisioned with WSDL documents
- Evolved as the successor to XML-RPC
- Characteristics:
  - Extensible
  - Neutrality (from transport layer)
    - Can run on HTTP, WebSocket, even SMTP
  - Independence

```
POST /InStock HTTP/1.1
Host: www.example.org
Content-Type: application/soap+xml; charset=utf-8
Content-Length: 299
SOAPAction: "http://www.w3.org/2003/05/soap-envelope"

<?xml version="1.0"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope" xmlns:m="http://www.example.org">
  <soap:Header>
  </soap:Header>
  <soap:Body>
    <m:GetStockPrice>
      <m:StockName>T</m:StockName>
    </m:GetStockPrice>
  </soap:Body>
</soap:Envelope>
```

Example Message

** From Wikipedia*

- Use XML validation to ensure structural message integrity
  - XML schemas provisioned with WSDL documents
- Evolved as the successor to XML-RPC
- Characteristics:
  - Extensible
  - Neutrality (from transport layer)
    - Can run on HTTP, WebSocket, even SMTP
  - Independence

Simple Object Access Protocol (SOAP)

- SOAP-based Web APIs use XML validation to ensure structural message integrity
  - XML schemas provisioned with WSDL documents
- Evolved as the successor to XML-RPC
- Characteristics:
  - Extensible
  - Neutrality (from transport layer)
    - Can run on HTTP, WebSocket, even SMTP
  - Independence

- Use XML validation to ensure structural message integrity
  - XML schemas provisioned with WSDL documents
- Evolved as the successor to XML-RPC
- Characteristics:
  - Extensible
  - Neutrality (from transport layer)
    - Can run on HTTP, WebSocket, even SMTP
  - Independence

Example WDSL

** From Wikipedia*

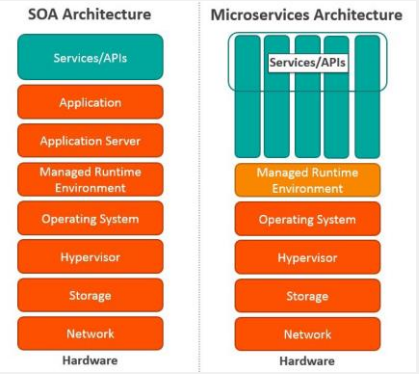
```
<!-- Abstract interfaces -->
   <interface name="Interface1">
      <fault name="Error1" element="tns:response"/>
      <operation name="Get" pattern="http://www.w3.org/ns/wsdl/in-out">
         <input messageLabel="In" element="tns:request"/>
         <output messageLabel="Out" element="tns:response"/>
      </operation>
   </interface>

<!-- Concrete Binding Over HTTP -->
   <binding name="HttpBinding" interface="tns:Interface1"
            type="http://www.w3.org/ns/wsdl/http">
      <operation ref="tns:Get" whttp:method="GET"/>
   </binding>

<!-- Concrete Binding with SOAP-->
   <binding name="SoapBinding" interface="tns:Interface1"
            type="http://www.w3.org/ns/wsdl/soap"
            wsoap:protocol="http://www.w3.org/2003/05/soap/bindings/HTTP/"
            wsoap:mepDefault="http://www.w3.org/2003/05/soap/mep/request-response">
      <operation ref="tns:Get" />
   </binding>
```

Simple Object Access Protocol (SOAP)

- It's popularity has somewhat diminished, but still very relevant in enterprise applications
  - SOAP is still used most often in the enterprise world, where communication between different services needs to *conform to a set of rules and contracts (*)*
  - Because it follows objects, rules, and constraints, SOAP is a more strict *(*)* protocol than REST
  - ** But do enterprises really conform to strict rules? Agile seems to work best in practice*
    - It might have been too rigid for its own good
- E.g. Salesforce SOAP API to create, retrieve, update or delete records, such as accounts, leads, and custom objects
- E.g. The PayPal SOAP API is based on open standards known collectively as web services, which include the Simple Object Access Protocol (SOAP), Web Services Definition Language (WSDL), and the XML Schema Definition language (XSD)

SOA and MicroServices

- MicroService Architecture is very similar, modern reincarnation of SOA
  - SOA mainly 2000~2010
  - MicroServices 2015~…
  - "*Microservices are the kind of SOA we have been talking about for the last decade. Microservices must be independently deployable, whereas SOA services are often implemented in deployment monoliths.*" - *Torsten Winterberg*
- What has changed? Adoption of:
  - Containerization
  - Asynchronous Programming
  - Distributed Computing mindset
  - CICD and Agile workflows



CLOUD COMPUTING APPLICATIONS

Cloud Computing Glue: Asynchronous RPC, WebSocket

Prof. Reza Farivar

Asynchronous RPC, aka. Streaming API

- Using old HTTP/0.9 and 1.0, what if the remote server takes a long time?
  - The client can wait, blocked, keeping the HTTP connection open (long polling)
  - It can keep polling the server periodically
  - Ultimately, these are hacks
  - HTTP/1.1 not ready for real-time web
- Web 2.0: Bidirectional client / server communication
  - AJAX
  - Comet
    - Umbrella term: Ajax Push, Reverse Ajax, Two-way-web, HTTP Streaming, and HTTP server push, …
  - Push notifications
  - XMLHttpRequest
    - XMLHttpRequest (XHR) is an API in the form of an object whose methods transfer data between a web browser and a web server.
    - The object is provided by the browser's JavaScript environment.
    - jQuery provides a nice wrapper (it also encapsulates WebSockets and server push)

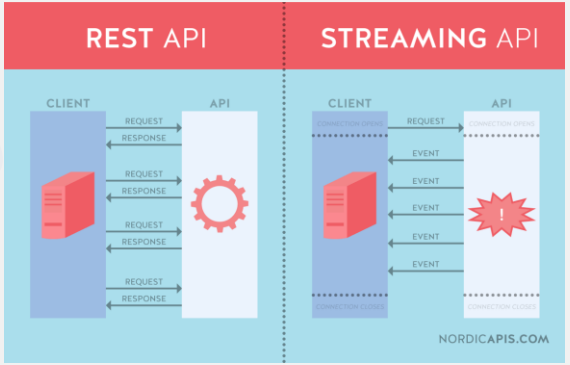


Image from https://nordicapis.com/rest-vs-streaming-apis-how-they-differ/

WebSocket

- RPC libraries can issue asynchronous methods to the server, and the server can inform them of the response later
- Streaming Architecture
  - Minimizing latency
- WebSocket
  - Part of HTML 5 standard
  - Can handle interactive sessions better than RESTful architecture
- Use cases:
  - Chat
  - Stock price update
  - Collaborative Document Editing
  - Location update (I am here now)
  - Multiplayer games

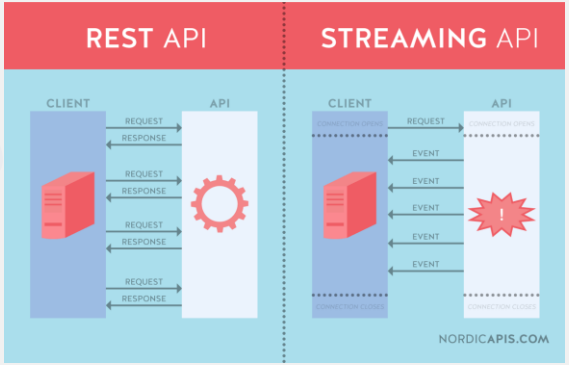


Image from https://nordicapis.com/rest-vs-streaming-apis-how-they-differ/

WebSocket Protocol

- WebSocket is an application protocol, running on top of TCP
  - It uses URIs, but not http://
  - Instead, uses ws://
- It utilizes an initial HTTP session and HTTP port numbers to process handshake phase
- The protocol has 3 phases
  - Opening Handshake
  - Data Transfer
  - Closing Handshake
- Protocol “upgrade”
- Purely event driven
  - Application code listens for events on WebSocket objects to handle incoming data and changes in connection status
- Asynchronous programming
  - Client does not need to do anything (e.g. poll) to receive data

WebSocket 3 phases

1) Opening Handshake
   - HTTP request/response to open WebSocket connection
   - Example client request

```
GET /chat HTTP/1.1
Host: example.com
Upgrade: websocket
Connection: Upgrade
Origin: http://example.com
Sec-WebSocket-Key: dGhlIHNhbXBsZSBub25jZQ==
Sec-WebSocket-Version: 13
```

   - Example server response

```
HTTP/1.1 101 Switching Protocols
         Upgrade: websocket
         Connection: Upgrade
Sec-WebSocket-Accept: s3pPLMBiTxaQ9kYGzzhZRbK+xOo=
```

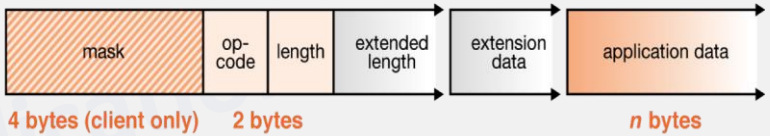
   - HTTP protocol is switched (aka. upgraded) to WebSocket

2) Data Transfer
- Bidirectional communication
- WebSocket frame
- Description of fields:
  - Op-code: Continuation, Text, Binary, Close, Ping, Pong

3) Closing handshake
   - A WebSocket frame with opcode 0x8 is sent

To keep the connection alive

WebSocket Frame



WebSocket API

- Simple Javascript W3C WebSocket API
- four different events:
  - Open - fires to establish a connection
  - Message - contains the data from the server
  - Error - fires in response to an unexpected event (failure)
  - Close - fires when the WebSocket connection is closed

- Primary methods + events
  - `WebSocket(URL, [protocols])` – Create a connection
    - `var ws = new WebSocket("ws://www.websocket.org","SOAP");//"SOAP" is optional`
  - `onOpen()` – WebSocket opened
    - `ws.onOpen = function(e){ console.log(ws.protocol);}`
  - `Send(data)` – Send data (string, Blob or Arraybuffer)
    - `ws.send("Hello WebSocket!");`
  - `onMessage()` – Message received
    - `ws.onMessage = function(e) { log("Message received: " + e.data); ws.close(); }`
  - `onClose()` – Close message received
    - `ws.onOpen = function(e){ console.log ("Disconnected: "+ e.reason);}`
  - `onError()` - Error

WebSocket

- Note that once you allow asynchronous communication in a networked environment, you should handle faults
- There are wrapper packages, encapsulating WebSockets functionality with additional features
  - Node.js supports WebSockets through plugins
  - Example: Socket.io.
  - Consisting of a Node.js server and a Javascript client library, socket.io provides reliability for handling proxies and load balancers as well as personal firewall and antivirus software and even supports binary streaming.
- Java 11 supports both HTTP as well as WebSocket protocols

WebSocket in Cloud Computing

- Many Cloud providers support WebSocket as the API of choice for interactive sessions with the Cloud service and the client
  - Amazon AWS API Gateway
  - Salesforce
  - Many cloud video vendors
    - E.g. easylive.io
  - Basis of Slack and its “Real Time Messaging API”
    - Slack has a simpler Event API based on HTTP/2 Push, but only the RTM guarantees real time delivery of messages
  - Google App Engine Channel API used WebSocket to allow server to client messaging
    - *Now deprecated, replaced by Firebase (a MBaaS solution)*
    - We can always run a Socket.io instance on Computer Engine (or EC2) instance
      - https://cloud.google.com/solutions/real-time-gaming-with-node-js-websocket
    - Or use AWS API Gateway to handle the deployment of WebSocket servers for us

CLOUD COMPUTING APPLICATIONS

Cloud Computing Glue: HTTP2 Push, Streaming Video

Prof. Reza Farivar

HTTP/2

- Published in 2015, most browsers supported it by the end of 2015
  - By 2020, about ~42% of top 10 million websites supported HTTP/2
- One of the main features of HTTP/2 is server push
  - Closely related to another feature of HTTP/2: data streaming
  - HTTP/2 Server push is being progressively implemented, for example Nginx web server implemented it in February 2018.
  - By now, all major servers and browsers support it
- It “proposed” new data in a new stream to the browser, to be stored in a Cache
- It does not send the pushed data directly to the client application itself
- To make the application aware, HTTP/2 utilizes Server-Side Events (SSE)

Amazon AWS API Gateway

- REST API
- WebSocket API
- HTTP API
  - Based on HTTP/2 push and notification

Video Streaming over the Internet

- Streaming Video Content
  - RTMP
    - Still very prevalent, but slowly being phased out
  - RTP (over UDP)
  - HLS (over HTTP)
  - MPEG-DASH
  - WebRTC

CLOUD COMPUTING APPLICATIONS

VPC: Virtual Private Cloud

Prof. Reza Farivar

Virtual Private Clouds

- Most Cloud Providers have some sort of network virtualization solution
  - Arguably the most fundamental building block
  - Amazon Virtual Private Cloud (VPC)
  - Microsoft Azure Virtual Network (VNet)
  - Google Virtual Private Cloud (VPC)
  - Oracle Virtual Cloud Networks
- They somewhat differ in detail, but the concepts are general
- In this lesson we will focus on Amazon VPC
  - Geared towards Cloud Architecture

Solving a Fundamental Problem

- Allow many different users have their own private network in the cloud
- Isolate different customers' network packets from each other
- Solution: VPC

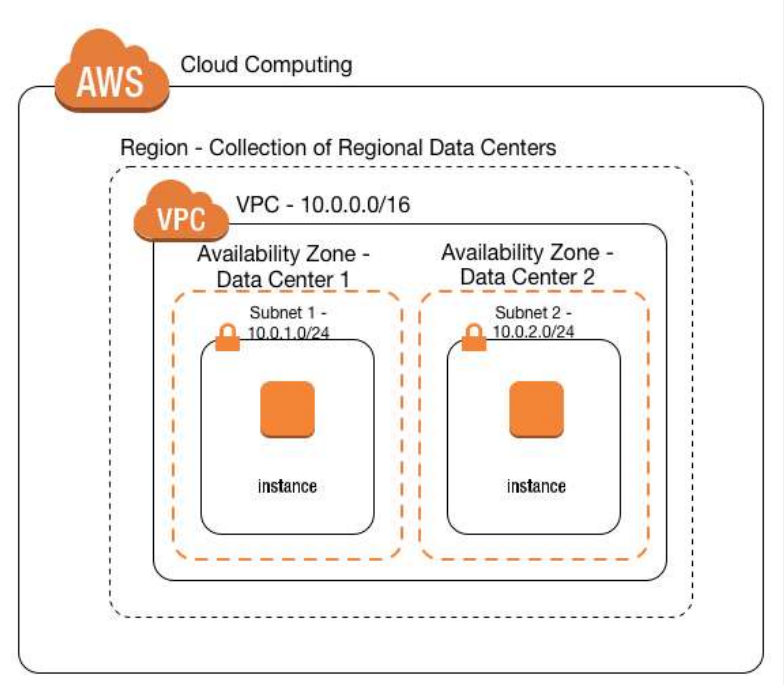
VPCs and Subnets

- You have your “own” network in the cloud
  - VPC
  - In AWS, a VPC is associated with a region, e.g. us-east-1
  - You can have more than one VPC, even in the same region
    - Default 5 quota
  - The IP address range of all the nodes in this VPC can be defined with a CIDR
- Your VPC is subdivided into logically separate segments
  - Subnet
  - Each subnet get a smaller CIDR range
  - Each subnet is associated with one Availability Zone
- You can then launch instances (EC2, RDS, etc.) in a subnet

VPCs, Subnets and Availability Zones

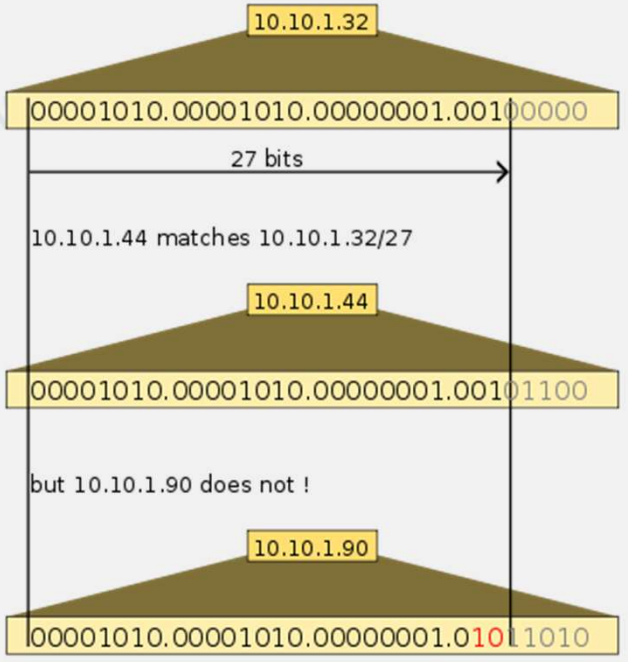
The main routing table, associated with the VPC, has the following route:

Destination	Target
10.0.0.0/16	local



Background Knowledge: CIDR

- CIDR: Classless Inter-Domain Routing
- A method of allocating IP address ranges
- In IPV4, each IP address is a 32 bit value
  - 4 bytes
  - 192.168.0.1
- CIDR notation:
  - 100.101.102.103/24
  - Take the mask (24 bits)
  - Keep the upper 24 bits the same
  - The lower bits can change ➔ range
  - 100.101.102.0 … 100.101.102.255
- IPV6, each address is 128 bits:
  - the IPv6 block *2001:db8::/48* represents the block of IPv6 addresses from *2001:db8:0:0:0:0:0:0* to *2001:db8:0:ffff:ffff:ffff:ffff:ffff*.
  - Note that in IPV6 notation, each segment is written in hex



RFC 1918: Address Allocation for Private Internets

- The Internet Assigned Numbers Authority (IANA) has reserved the following three blocks of the IP address space for private internets
- 10.0.0.0/8
  - 10.0.0.0 - 10.255.255.255
  - Number of addresses: 16,777,216
- 172.16.0.0/12
  - 172.16.0.0 - 172.31.255.255
  - Number of addresses: 1,048,576
- 192.168.0.0/16
  - 192.168.0.0 - 192.168.255.255
  - Number of addresses: 16,777,216
- Why? Because it is guaranteed no other server on the public internet has an IP address in these ranges
- Routing rules do not conflict

RFC 1918 and AWS VPC

- When you create a VPC, you must specify an IPv4 CIDR block for the VPC
- The allowed block size is between a /16 netmask (65,536 IP addresses) and /28 netmask (16 IP addresses)
  - Note that RFC 1918 would allow for 16 million distinct IP addresses in the 10.0.0.0/8, but Amazon would at most accept a /16 netmask in a VPC or subnet

RFC 1918 range	Example AWS VPC CIDR block
10.0.0.0 - 10.255.255.255 (10/8 prefix)	Your VPC must be /16 or smaller, for example, 10.0.0.0/16.
172.16.0.0 - 172.31.255.255 (172.16/12 prefix)	Your VPC must be /16 or smaller, for example, 172.31.0.0/16.
192.168.0.0 - 192.168.255.255 (192.168/16 prefix)	Your VPC can be smaller, for example 192.168.0.0/20.

Reserved IP Adresses

- The first four IP addresses and the last IP address in each subnet CIDR block are not available for you to use, and cannot be assigned to an instance; E.g. for 10.0.0.0/24:
  - 10.0.0.0: Network address
  - 10.0.0.1: The VPC router
  - 10.0.0.2: The IP address of the DNS server is the base of the VPC network range plus two
  - 10.0.0.3: Reserved for future use
  - 10.0.0.255: Network broadcast address
    - AWS does NOT support broadcast in a VPC, therefore this address is reserved

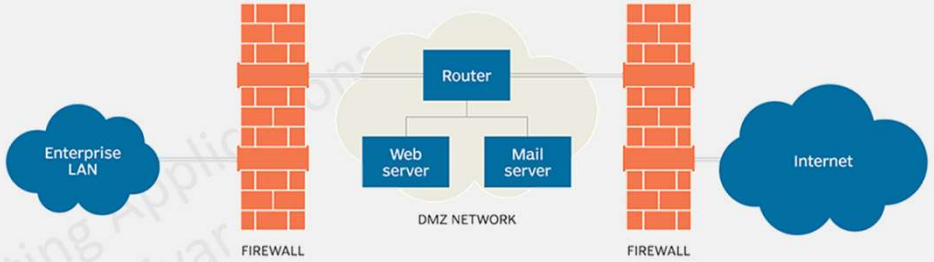
CLOUD COMPUTING APPLICATIONS

VPC: Subnets

Prof. Reza Farivar

Subnets

- Create subnets to isolate resources per the project requirement
  - DMZ/Proxy
  - Load balancer
  - web applications
  - Mail servers
  - Databases



- E.g. have a public subnet to host internet-facing resources and a private subnet for databases that accept web requests
- Create multiple subnets (public or private) in multiple AZs to host a high availability multi-AZ infrastructure and avoid a single point of failure
  - each subnet can communicate with every other subnet in the same VPC

Private Subnets

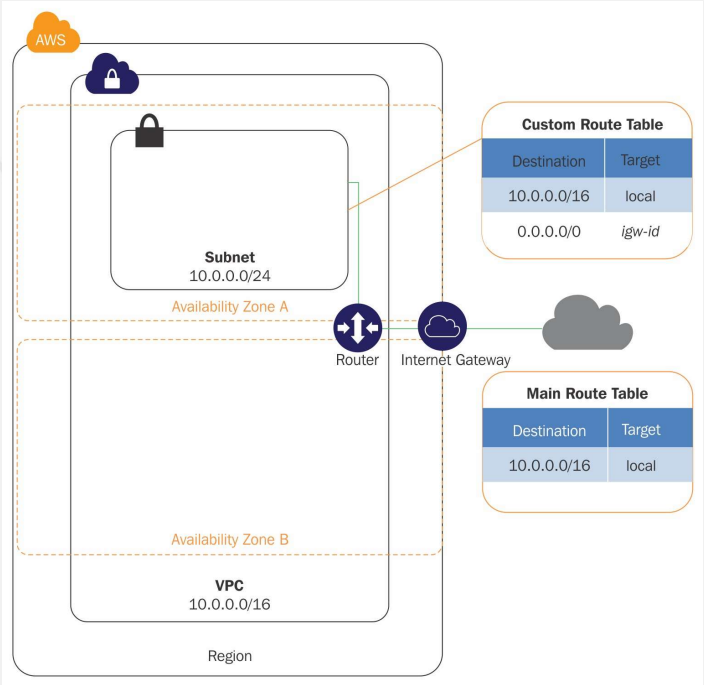
- Any incoming traffic from the internet cannot directly access the resources within a private subnet
- Outgoing traffic from a private subnet cannot directly access the internet
  - Restricted; or
  - Routed through a NAT
- Each resource (instance) gets a private IP
  - From the CIDR range associated with the subnet
- Technically, a subnet is private if there is no route in the routing table to an internet gateway

Public Subnet

- A subnet that has access to an internet gateway defined in the routing table
- Each resource in a public subnet gets a private IP within the CIDR range, AND a public IP accessible from the internet
  - the public IP can be dynamic (only remains valid while the instance is alive, and then AWS reclaims it), or
  - It can be an elastic IP, where you pay for it, and it will remain yours even if the instance shuts down
- Outgoing traffic can directly access internet
  - Unlike Private, which needs a NAT to access internet

Route Tables

- Each VPC has an associated “Main Route Table”
  - The Default VPC has a route in its “Main Route Table” to an internet gateway
  - Custom VPCs usually only have the local route in the MRT
- Subnets can have their own custom route table
  - If no custom route table is explicitly associated with a subnet, then it is associated with the VPC's main route table
- Possible route table targets
  - Local, IGW, a NAT device, A VGW, a peering connection, or a VPC endpoint (e.g. S3)
- Selection of the optimum route for network traffic is done based on the longest prefix match
  - the most specific routes that match the network traffic



CLOUD COMPUTING APPLICATIONS

VPC: Gateways

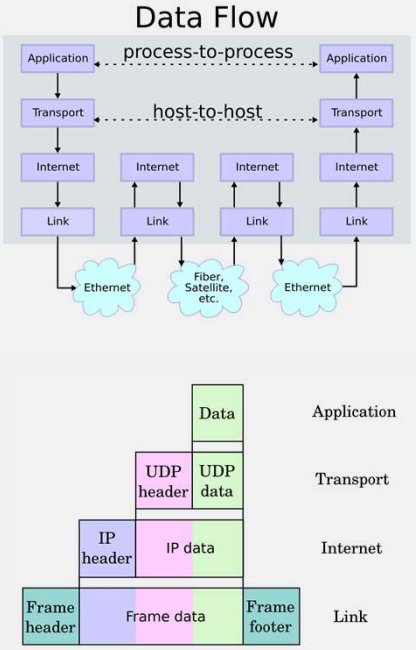
Prof. Reza Farivar

Internet Gateways

- Internet Gateway is a logical construct, not a specific instance or resource
- AWS does quite a bit of behind the scene work to allow highly available internet to all the required Availability Zones in the VPC
- Is attached to a VPC
- Highly available, redundant, and horizontally scaled
- → in the route tables, it is referred to by its name (e.g. igw-05ae7f551a8154d1a), not an IP address

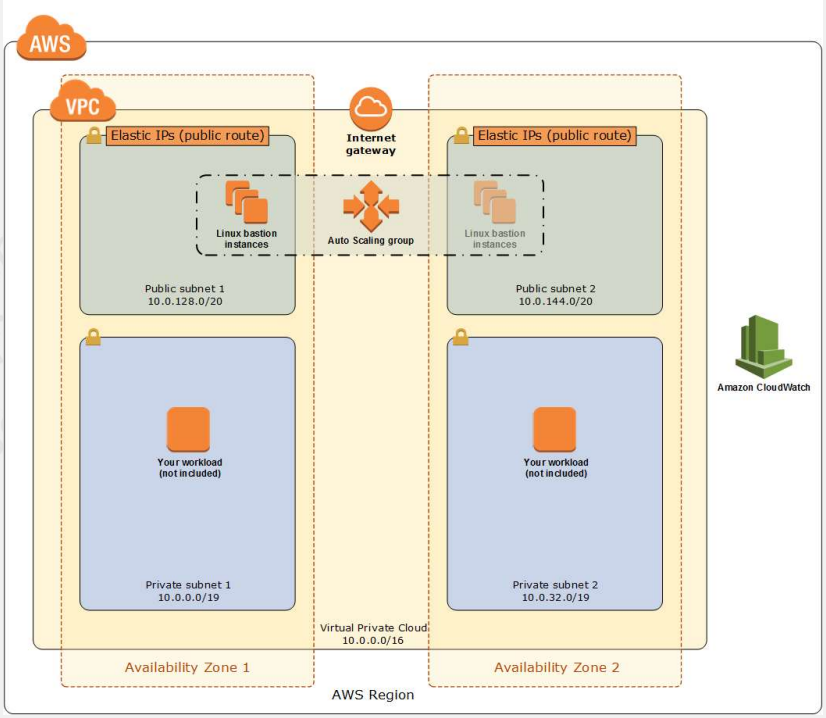
NAT Gateways

- Network Address Translation
  - Just like your home wireless router
- Virtual router or a gateway in a public subnet that enables instances in a private subnet to interact with the internet
  - IPV4 only
- Modifies the network address information in the IP header
  - It receives traffic from an EC2 instance residing in a private subnet
  - before forwarding the traffic to the internet, replaces the reply-to IPv4 address with its own public or Elastic IP address
  - When a reply is received from internet, it changes the reply-to address from its IP address to the EC2 instance private IP address
- Two types of NAT
  - NAT Instance → Runs as as EC2 instance
  - NAT Gateway → fully managed by AWS, requires elastic IP
    - Better availability and higher bandwidth



Bastion Host

- Use a bastion host to access private machines hosted in a private network in a VPC
- Bastion host: “a server whose purpose is to provide access to a private network from an external network, such as the Internet. Because of its exposure to potential attack, a bastion host must minimize the chances of penetration”



Further reading: https://cloudacademy.com/blog/aws-bastion-host-nat-instances-vpc-peering-security/

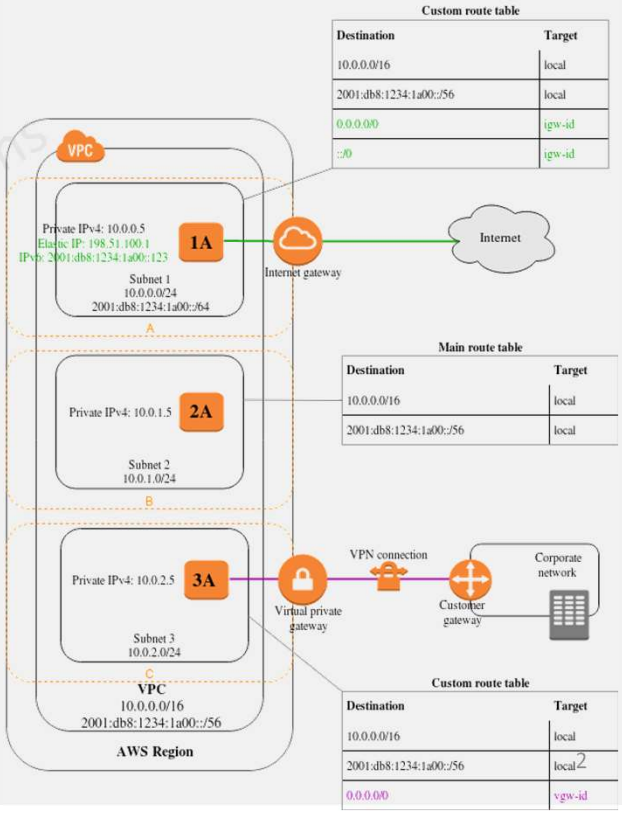
CLOUD COMPUTING APPLICATIONS

VPC: Advanced VPC

Prof. Reza Farivar

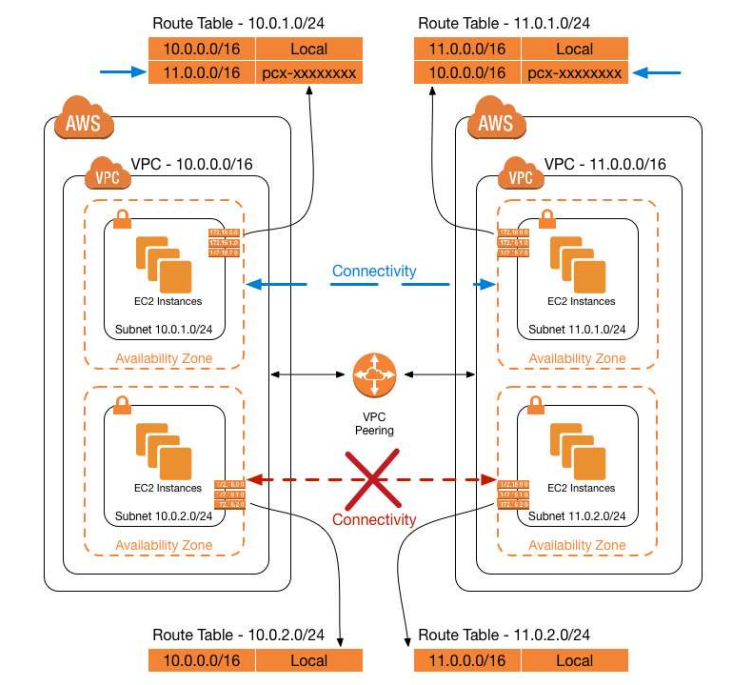
Virtual Private Gateway

- If a subnet doesn't have a route to the internet gateway, but has its traffic routed to a virtual private gateway for a Site-to-Site VPN connection, the subnet is known as a *VPN-only subnet*
- In this diagram, subnet 3 is a VPN-only subnet
- Concepts
  - **VPN connection**: A secure connection between your on-premises equipment and your VPCs.
  - **VPN tunnel**: An encrypted link where data can pass from the customer network to or from AWS.
    - Each VPN connection includes two VPN tunnels which you can simultaneously use for high availability.
  - **Customer gateway**: An AWS resource which provides information to AWS about your customer gateway device.
  - **Customer gateway device**: A physical device or software application on your side of the Site-to-Site VPN connection.
  - **Virtual private gateway**: The VPN concentrator on the Amazon side of the Site-to-Site VPN connection. You use a virtual private gateway or a transit gateway as the gateway for the Amazon side of the Site-to-Site VPN connection.
  - **Transit gateway**: A transit hub that can be used to interconnect your VPCs and on-premises networks. You use a transit gateway or virtual private gateway as the gateway for the Amazon side of the Site-to-Site VPN connection.
- As of 2020, VPN connections into AWS are IPV4 only
- It is recommended that you use non-overlapping CIDR blocks for your networks



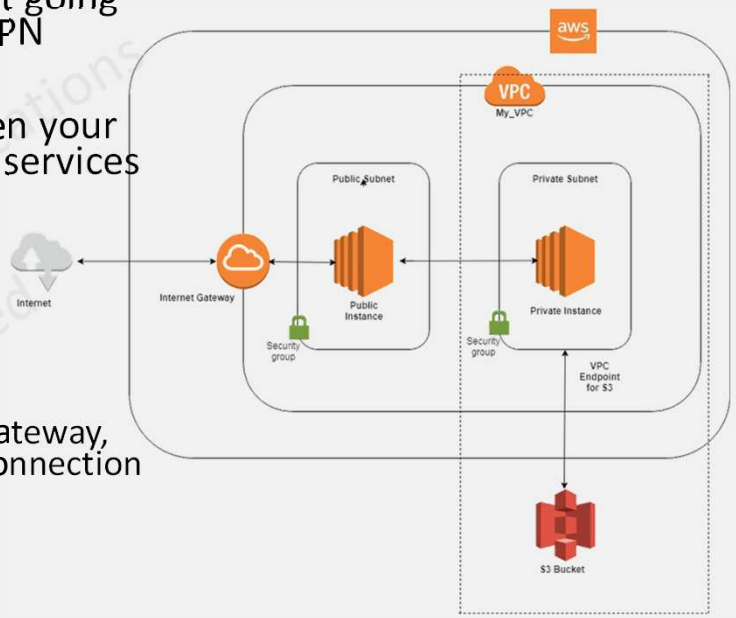
VPC Peering

- VPC peering can be used to make communication between VPCs within the same account, different AWS accounts, or any two VPCs within the same region or different regions
- Initially, VPC peering was supported only within the same region, but later AWS added support for VPC peering across regions
- The two VPCs cannot have CIDR blocks that overlap with each other.



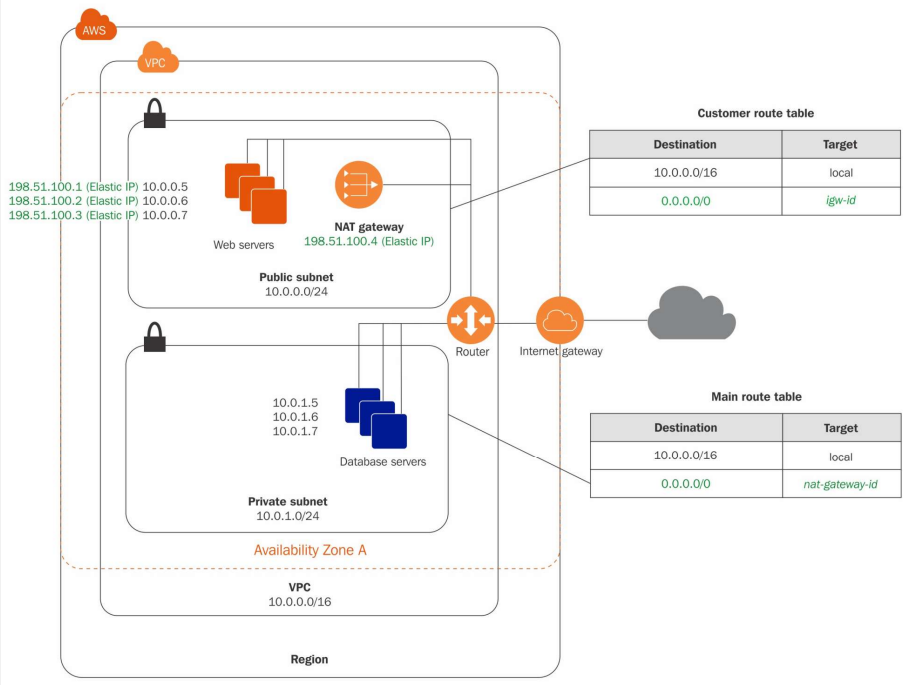
VPC Endpoints

- Generally, AWS services are different entities and do not allow direct communication with each other without going through either an IGW, a NAT gateway/instance, a VPN connection, or AWS Direct Connect
- A VPC endpoint enables private connections between your VPC and supported AWS services and VPC endpoint services
  - S3
  - DynamoDb
- AWS PrivateLink
  - Private IP addresses
  - Traffic does not leave the Amazon network
  - Does not require an internet gateway, virtual private gateway, NAT device, VPN connection, or AWS Direct Connect connection



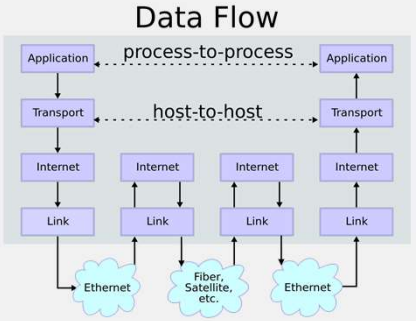
** Interesting reading: https://www.bluematador.com/blog/s3-endpoint-connectivity-in-aws-vpc*

VPC with Private and Public Subnets



Routing in VPC vs. Physical Network

- Physical Ethernet Network
  - Link Layer
    - Lowest layer in the Internet Protocol
      - Layer 2 in OSI model
  - In a physical traditional network, this layer uses MAC address and ARP messaging (to discover unknown MAC addresses)
- VPC Network
  - Amazon backend intercepts any MAC ARP request
  - Looks up routing tables, and returns the destination without implementing ARP



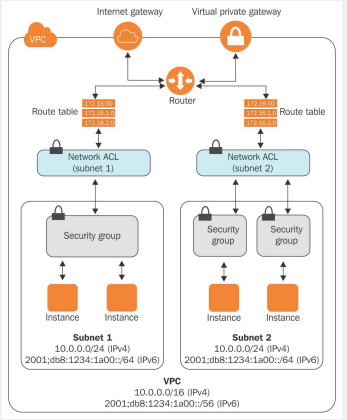
CLOUD COMPUTING APPLICATIONS

VPC: Security and Firewalls

Prof. Reza Farivar

Security and Firewalls

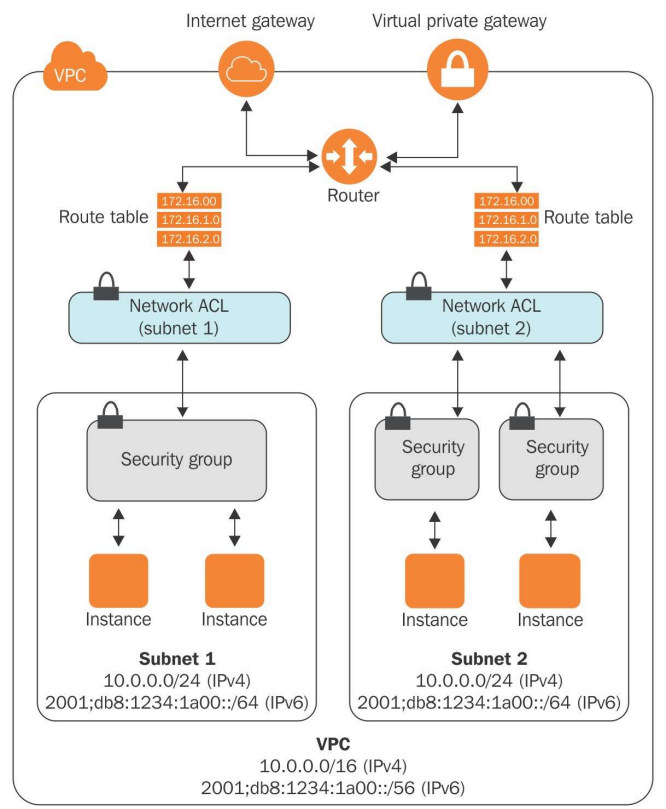
- Security
  - Security Groups
    - EC2 instance-level firewall
  - Network Access Control Lists (NACL)
    - Subnet firewall
- Monitoring
  - Flow Logs
  - Enable VPC flow logs for audit purposes
  - Study flow logs from time to time
    - highlights unauthorized attempts to access the resources



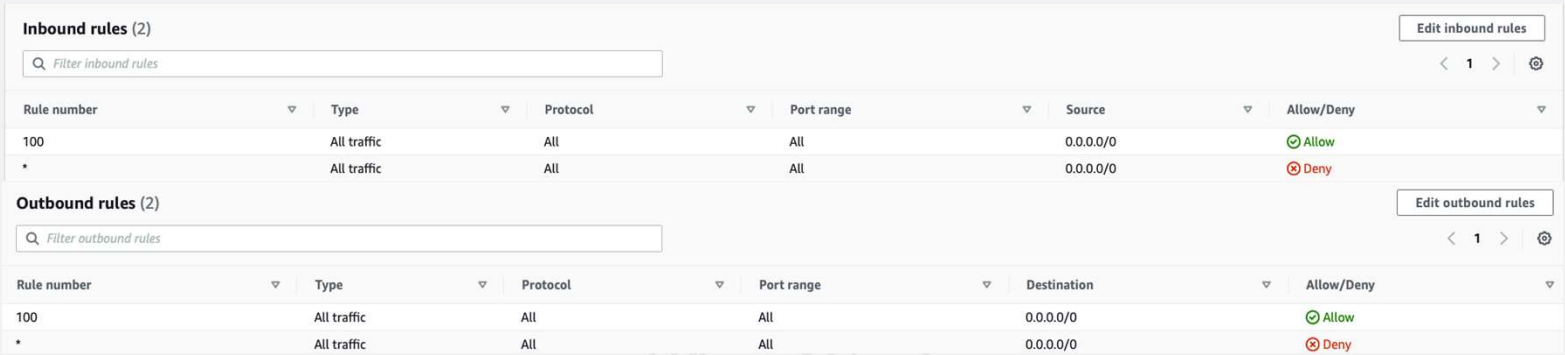
Security

- Make sure that only required ports and protocols from trusted sources can access AWS resources using security groups and NACLs
- Make sure that unwanted outgoing ports are not open in security groups
  - A security group for a web application does not need to open incoming mail server ports

Security and Firewalls



Network Access Control List



Inbound rules (2)

Edit inbound rules

Filter inbound rules

1

Rule number	Type	Protocol	Port range	Source	Allow/Deny
100	All traffic	All	All	0.0.0.0/0	Allow
*	All traffic	All	All	0.0.0.0/0	Deny

Outbound rules (2)

Edit outbound rules

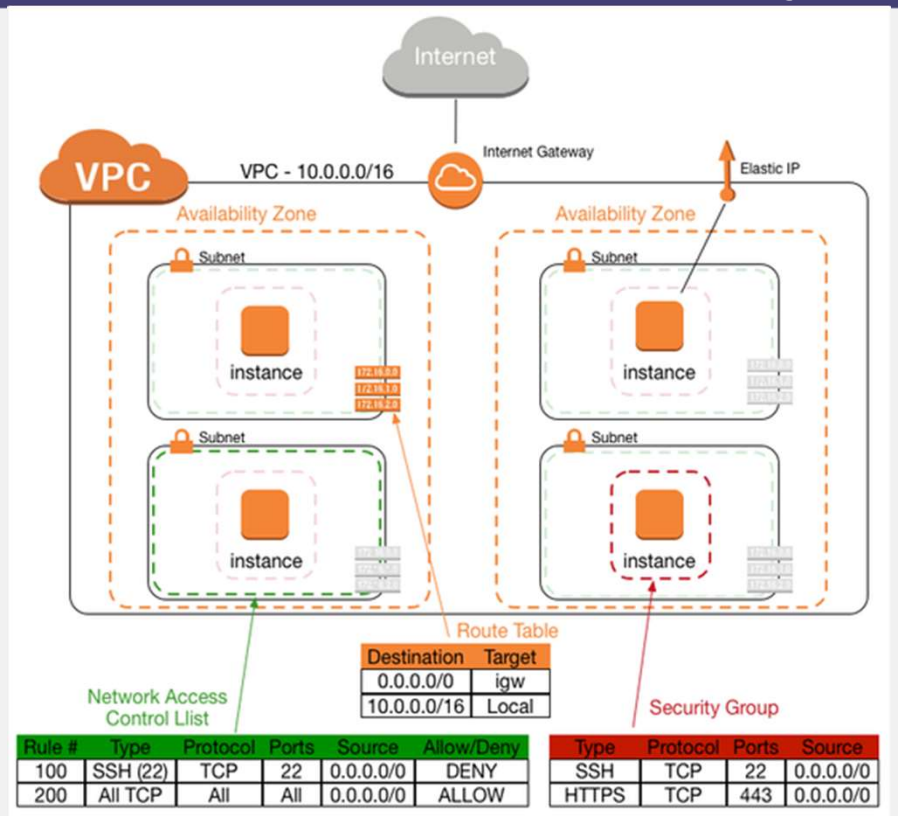
Filter outbound rules

1

Rule number	Type	Protocol	Port range	Destination	Allow/Deny
100	All traffic	All	All	0.0.0.0/0	Allow
*	All traffic	All	All	0.0.0.0/0	Deny

- NACL acts as a virtual firewall at the subnet level
- Every VPC has a default NACL
- Every subnet, whether it is private or public in a VPC, must be associated to one NACL
- One NACL can be associated with one or more subnets; but each subnet can have ONE NACL associated with it
- NACL rules are evaluated based on its rule numbers. It evaluates the rule starting from the lowest number to the highest number
- NACL is stateless:
  - Separate rules to allow or deny can be created for inbound and outbound traffic
  - If a port is open for allowing inbound traffic, it does not automatically allow outbound traffic
- The default NACL for any VPC contains a rule numbered as * in both inbound and outbound rules
  - This rule appears and executes last

Anatomy of a VPC with Route Table, Network ACL and Security Group



Security Group

Inbound rules — Edit inbound rules

Type	Protocol	Port range	Source	Description - optional
HTTP	TCP	80	0.0.0.0/0	-
HTTP	TCP	80	::/0	-
Custom TCP	TCP	8080	0.0.0.0/0	-
Custom TCP	TCP	8080	::/0	-
SSH	TCP	22	0.0.0.0/0	-
SSH	TCP	22	::/0	-
HTTPS	TCP	443	0.0.0.0/0	-
HTTPS	TCP	443	::/0	-

Outbound rules — Edit outbound rules

Type	Protocol	Port range	Destination	Description - optional
All traffic	All	All	0.0.0.0/0	-

- Firewall at the instance level
- One or more security groups can be associated with each EC2 instance
- A security group can be attached to many EC2 instances
- Each SG contains rules allowing inbound and outbound traffic
- Using CIDR notation, a source IP can be fixed to a particular IP, such as 10.108.20.107/32
- Any source IP can be allowed by a 0.0.0.0/0

Security Group as Source IP

Inbound rules Edit inbound rules

Type	Protocol	Port range	Source	Description - optional
Custom TCP	TCP	4003 - 65535	sg-00fbbeba74e062d16 (nodes.dev.k8s.mp3-k8.in)	-
Custom TCP	TCP	2382 - 4000	sg-00fbbeba74e062d16 (nodes.dev.k8s.mp3-k8.in)	-
All traffic	All	All	sg-003e7c9121913dca3 (masters.dev.k8s.mp3-k8.in)	-
SSH	TCP	22	0.0.0.0/0	-
Custom UDP	UDP	1 - 65535	sg-00fbbeba74e062d16 (nodes.dev.k8s.mp3-k8.in)	-
Custom TCP	TCP	1 - 2379	sg-00fbbeba74e062d16 (nodes.dev.k8s.mp3-k8.in)	-
HTTPS	TCP	443	0.0.0.0/0	-

- A security group ID can be specified as a source IP to allow communication from all the instances that are attached to that security group
- For example, in the case of autoscaling, the number of EC2 instances and their IP addresses keeps changing.
- In such situations, it is best practice to attach a security group to such EC2 instances with the help of an autoscaling template and place a security group ID as a source IP in another security group.