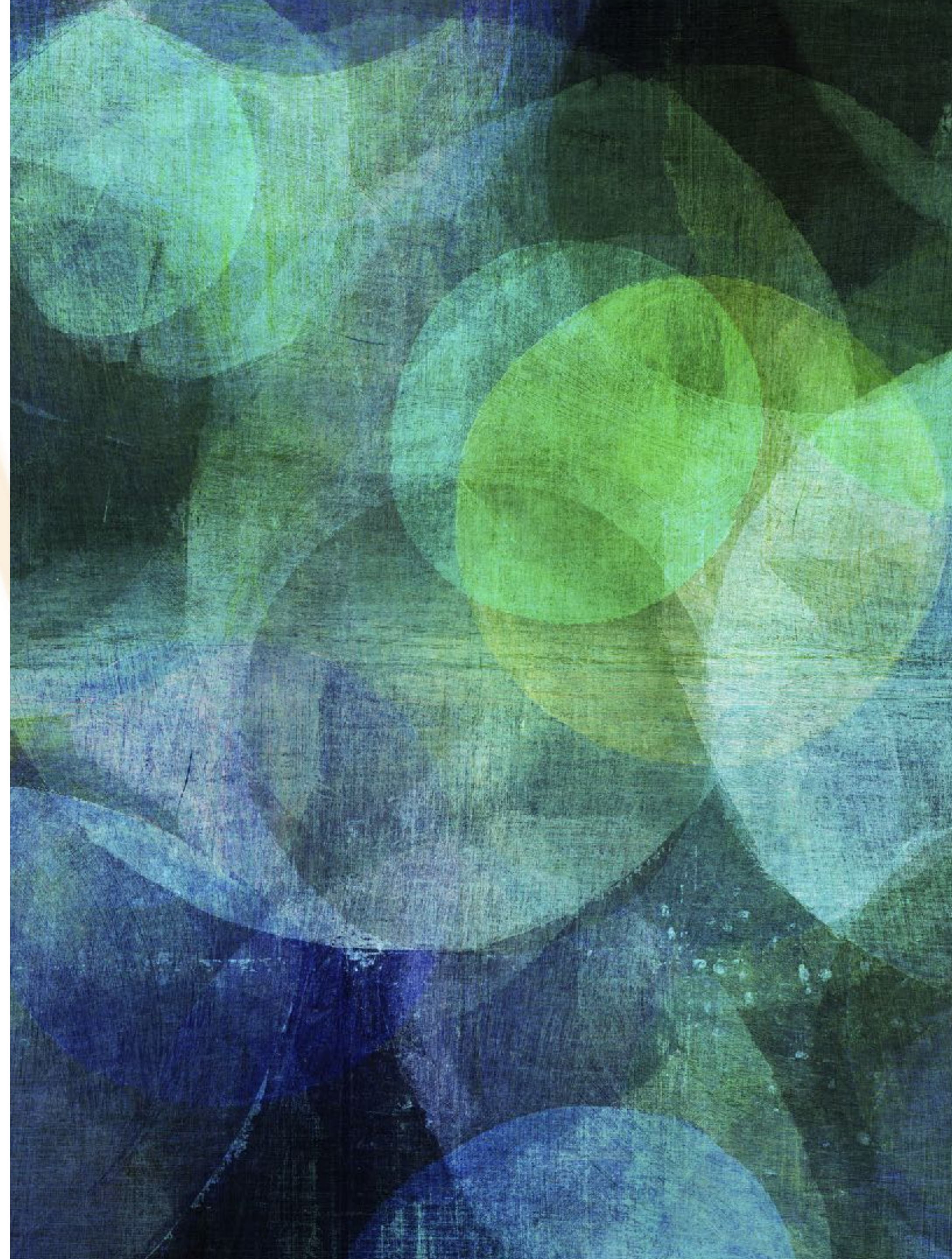


# BIG DATA ON AWS 01

---

*Nan Dun*  
*nan.dun@acm.org*





# COPYRIGHT POLICY 版权声明

---

*All content included on the Site or third-party platforms as part of the class, such as text, graphics, logos, button icons, images, audio clips, video clips, live streams, digital downloads, data compilations, and software, is the property of BitTiger or its content suppliers and protected by copyright laws.*

*Any attempt to redistribute or resell BitTiger content will result in the appropriate legal action being taken.*

*We thank you in advance for respecting our copyrighted content.*

*For more info see <https://www.bittiger.io/termsfuse> and <https://www.bittiger.io/termservice>*

所有太阁官方网站以及在第三方平台课程中所产生的课程内容，如文本，图形，徽标，按钮图标，图像，音频剪辑，视频剪辑，直播流，数字下载，数据编辑和软件均属于太阁所有并受版权法保护。

对于任何尝试散播或转售BitTiger的所属资料的行为，太阁将采取适当的法律行动。

我们非常感谢您尊重我们的版权内容。

有关详情，请参阅

<https://www.bittiger.io/termsfuse>

<https://www.bittiger.io/termservice>



# DISCLAIMER

---

- I. *“All data, information, and opinions expressed in this presentation is for informational purposes only. I do not guarantee the accuracy or reliability of the information provided herein. This is a personal presentation. The opinions expressed here represent my own and not those of my employer.”*
- II. *“The copyright of photos, icons, charts, trademarks presented here belong to their authors.”*
- III. *“I could be wrong.”*



BIT TIGER

**GET READY**



# CONTENTS MARKS

---



*Important, Must Know*



*Homework, Todo*



*Information, Read later*

# DO THIS FIRST



- Register AWS account at <https://aws.amazon.com/account/>
- Register Bitbucket account at <https://bitbucket.org>
- Register Github account at <https://github.com/>
- Send an email to [nan.dun@acm.org](mailto:nan.dun@acm.org), including following information
  - Your name, email for mailing group
  - Your AWS account ID, a 12-digit number found at [AWS account home](#)
  - Your Bitbucket ID or email
- Once you received notification from [bittiger-aws@googlegroups.com](mailto:bittiger-aws@googlegroups.com), send a brief introduction of yourself to the group
- Create a billing alert: <http://docs.aws.amazon.com/awsaccountbilling/latest/aboutv2/free-tier-alarms.html>

# MAIN PROJECT

---

- **Main Project: 1.2 Billion NYC Taxi Trip Data Analysis**
  - Build a data pipeline to analysis 300GB, 1.2B taxi trip data and visualization
  - Stack: EC2, S3, ECS, SQS, DynamoDB, Terraform, Ansible, Docker, Python, Bokeh
  - Learning by full code examples: <https://bitbucket.org/bittiger-aws/aws>
- Evaluation
  - Project report
  - Project extensions:
    - Monitoring using CloudWatch, or Datadog
    - Bug fixes or other improvements via PRs



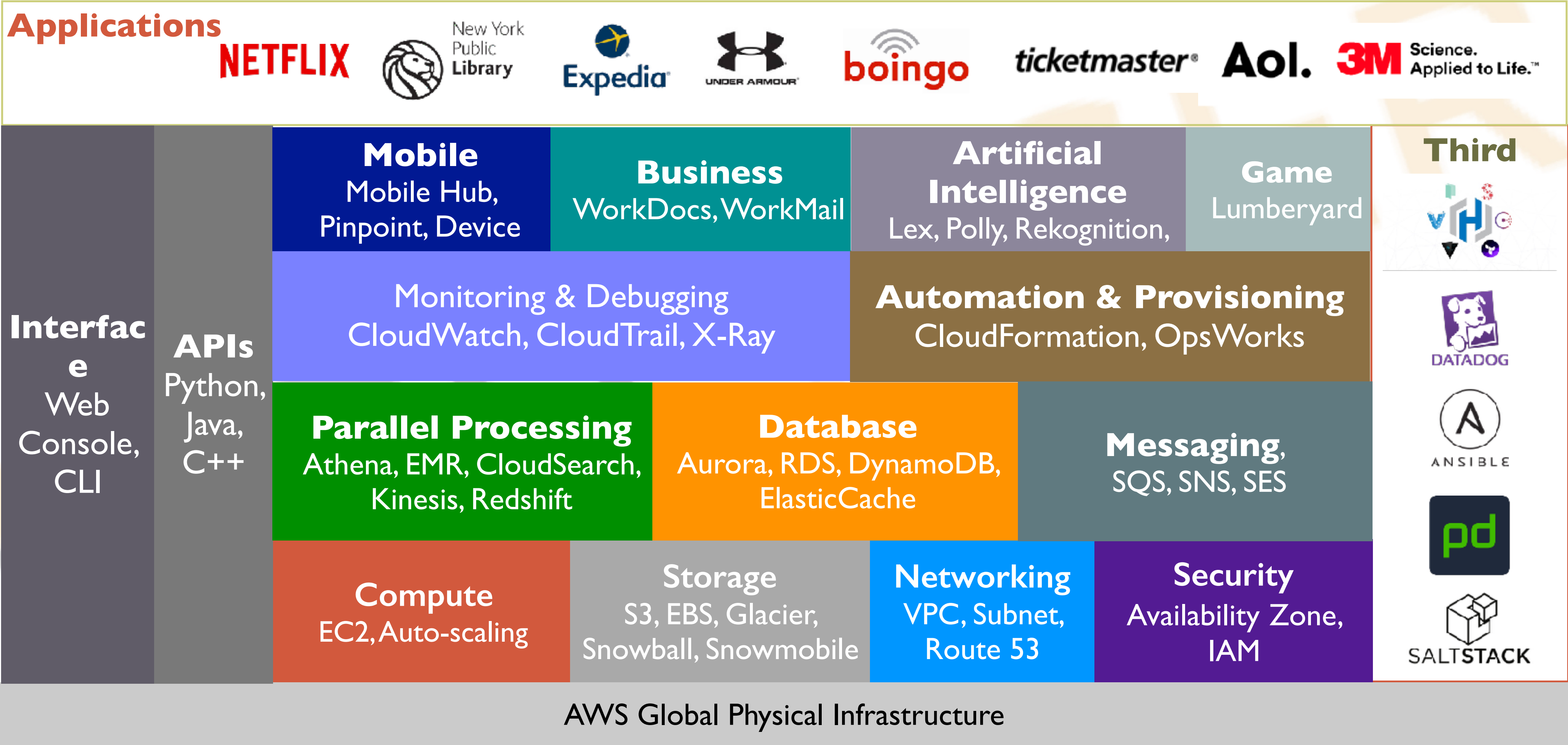
# OPEN PROJECT

---

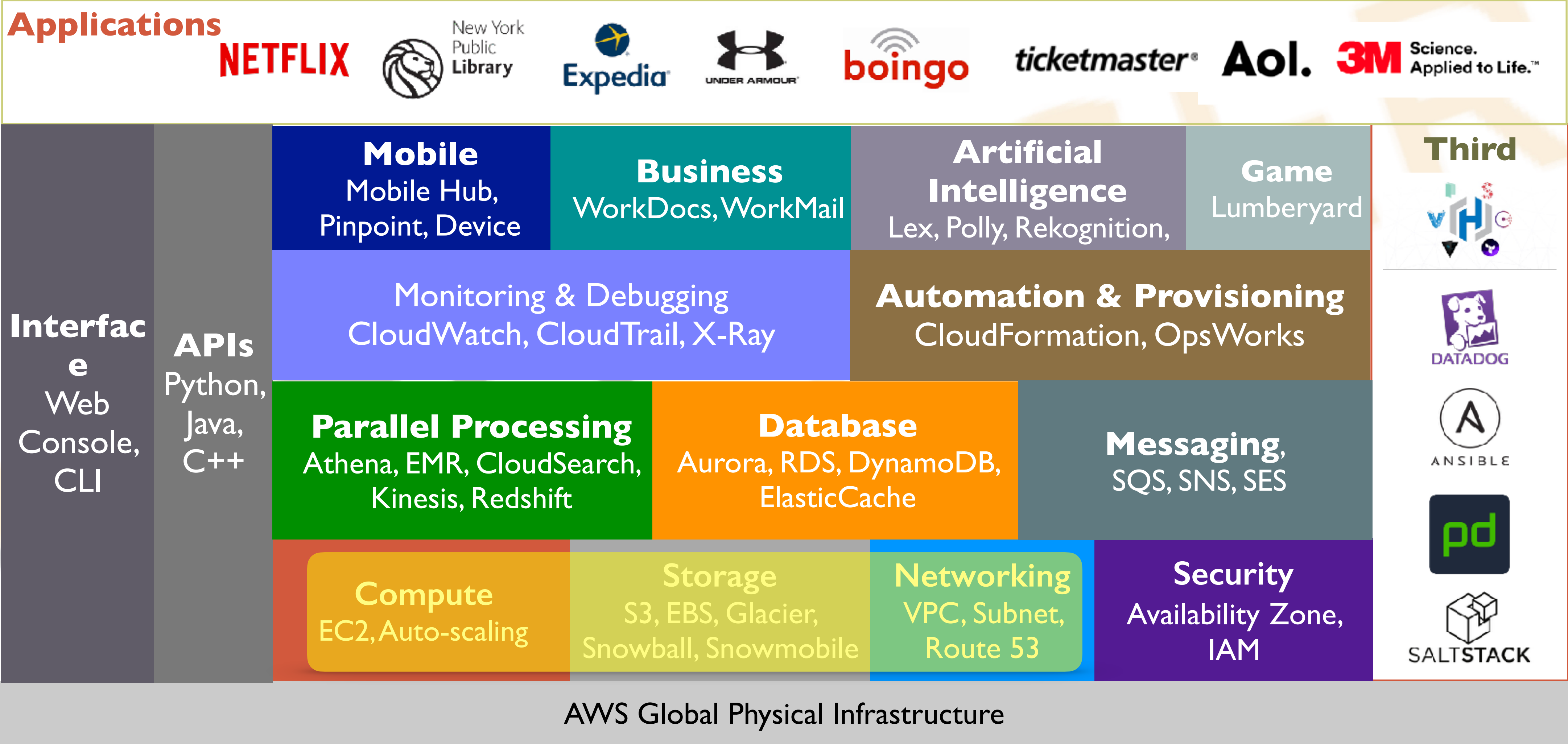
- **Mining Open Datasets using Clustering Techniques**
  - Must use one of clustering techniques, such as K-means
  - Free to choose your data such as [Awesome Public Datasets](#) or [AWS Public Dataset](#)
  - **Clear, tangible goals and design**, focusing on problem and system integration
  - Make it work, and hopefully get some surprising results
  - Hosted on Github by group and shared with lecturer
- Evaluation
  - Critical thinking and problem-solving oriented! (**Absolutely NO tech show-off**)
  - Technical challenges, systematic design
  - Performance and cost analysis, trade-off
  - Coding skills and elegance
  - Presentation and communication
  - Teamwork and help each other



# TODAY'S TOPIC



# TODAY'S TOPIC







EC2

# EC2 – OUTLINE

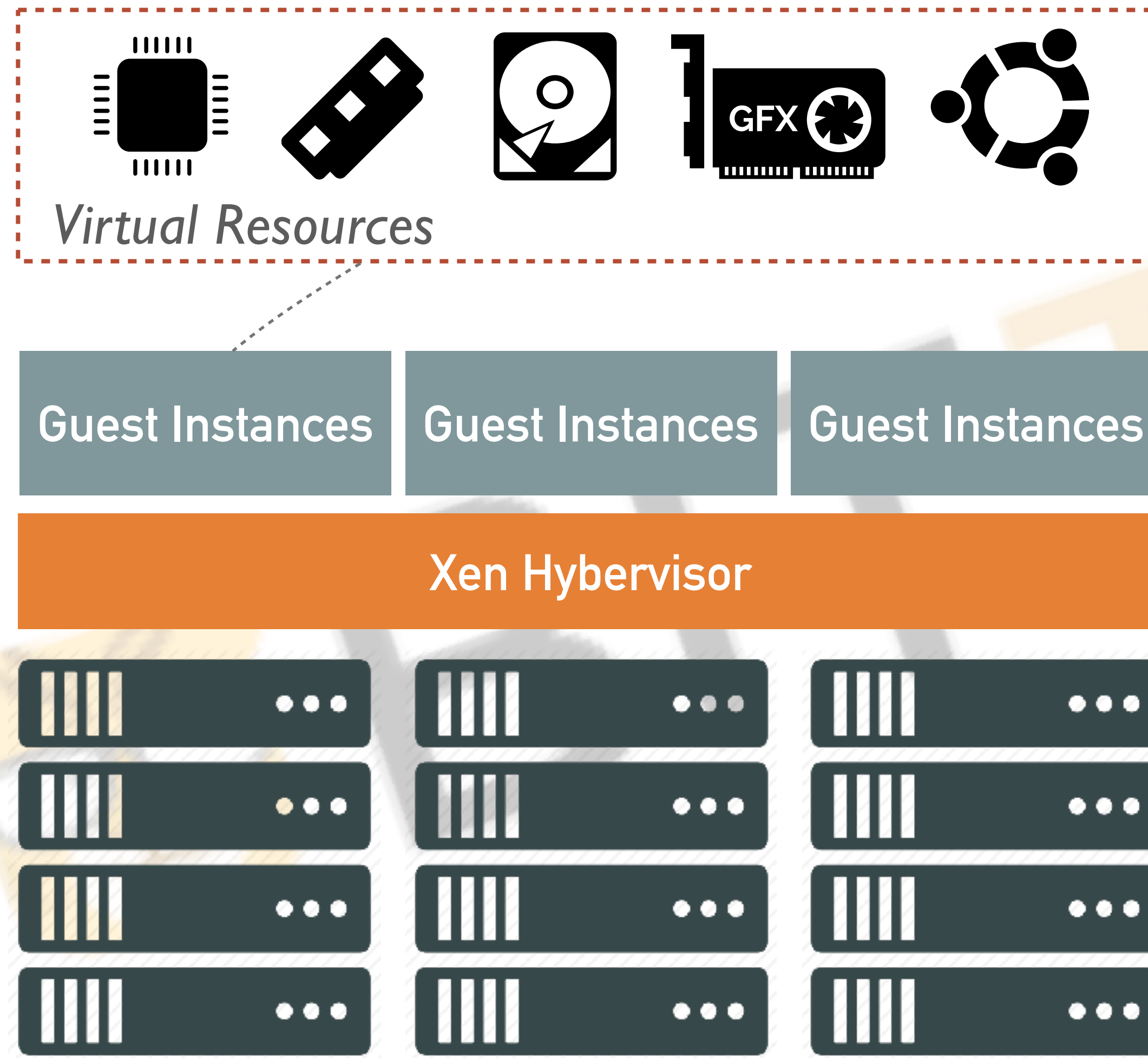
---

- **Elastic Compute Cloud (EC2)**
  - Instance Types
  - Networking
  - Storage Types
  - Elastic Block Storage (EBS)





# EC2 INSTANCE



- **What is Xen?**
  - <https://en.wikipedia.org/wiki/Xen>
- **What is HVM and PV? difference?**
  - [http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/virtualization\\_types.html](http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/virtualization_types.html)
  - <http://cloudacademy.com/blog/aws-ami-hvm-vs-pv-paravirtual-amazon/>

# INSTANCE TYPE



General Purpose: T, M

Compute Optimized: C

Memory Optimized: X, R

GPU Accelerated: P, G, F

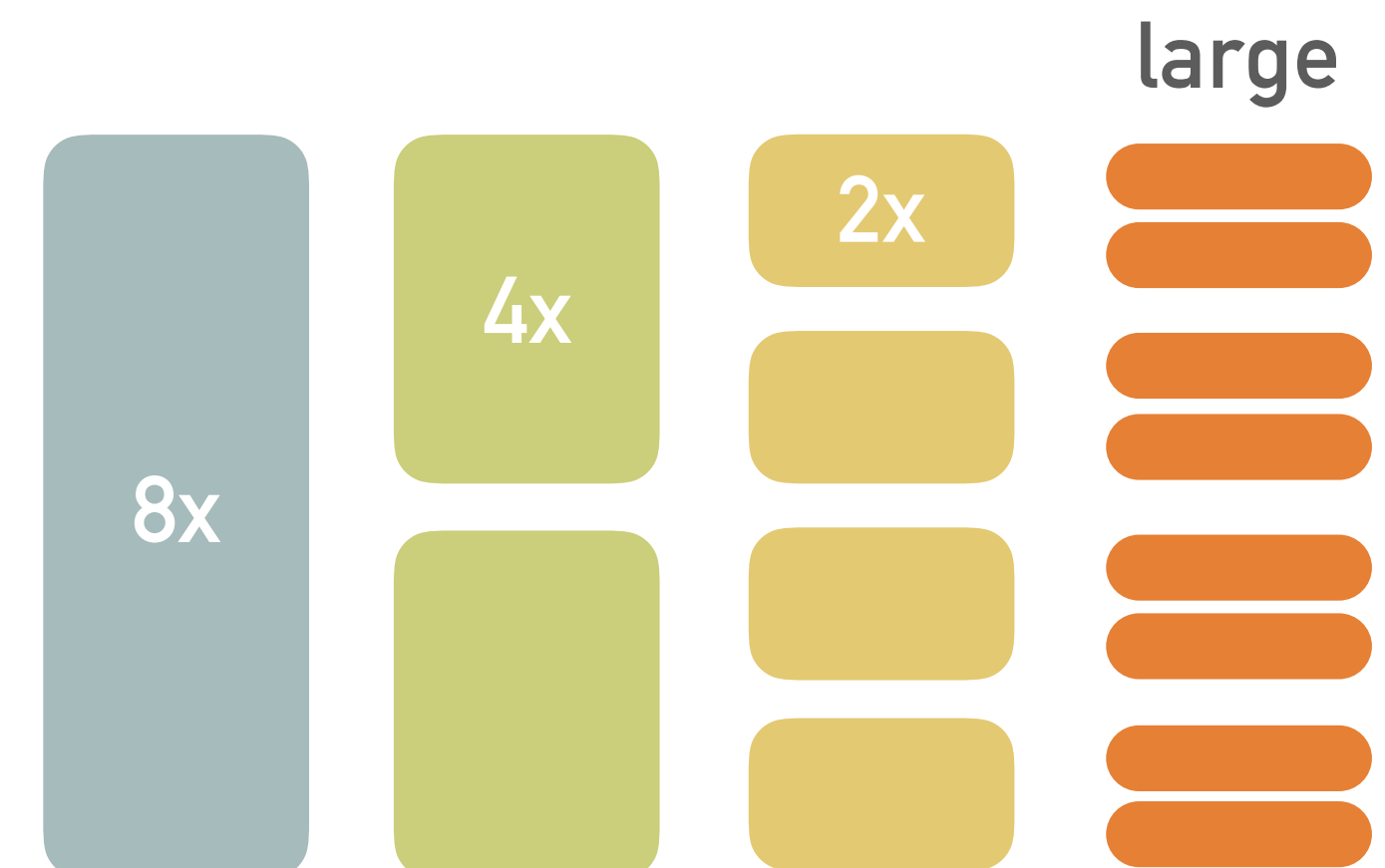
Storage Optimized: I, D

Instance Generation

Instance Family

Instance Size

c4.2xlarge





# EC2 – CPU

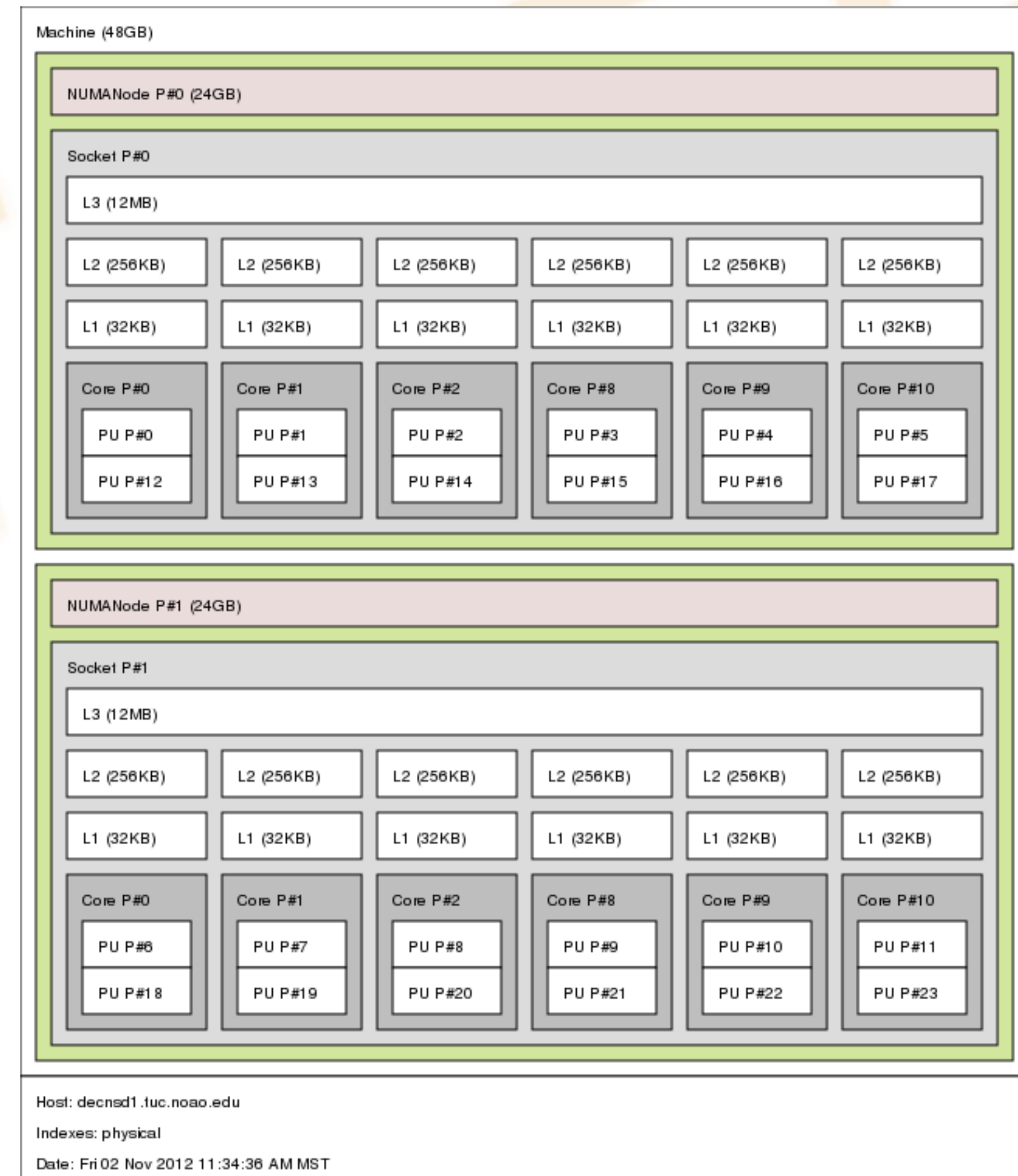


- **vCPU = A hyper-threaded physical core**
  - What is hyper threading?
  - Why do we need hyper threading?
  - How to show the vCPU topology of EC2 instances?
  - Why do we need to disable hyper threading sometimes?
  - How to disable hyper-threading?
  - <https://aws.amazon.com/ec2/virtualcores/>
- **Dedicated Host**
  - <https://aws.amazon.com/ec2/dedicated-hosts/>

# EC2 - CPU



- Homework: CPU layout and Enable/Disable Hyper-threading
  - Launch a instance with c4.8xlarge
  - `lscpu`
  - `lstopo -.txt`
  - `/sys/devices/system/*`
  - Disable hyper-threading





# CREDIT MODEL: T2 INSTANCE

---

Instance Type	Initial CPU Credit	Credit Earned/Hour	Baseline	Max. Balance
t2.nano	30	3	5%	72
t2.micro	30	6	10%	144
t2.small	30	12	20%	288
t2.medium	60	24	40%	576
t2.large	60	36	60%	864
t2.xlarge	120	54	90%	1296
t2.2xlarge	240	81	135%	1944

*\*1 CPU Credit = Full CPU core Usage for one minute*

# HIGH-END: X1 INSTANCE

---

Instance Type	vCPU	Memory	Storage	Network	EBS Bandwidth
x1.32xlarge	128	1952	2x1920 SSD	20 Gbps	10 Gbps
x1.16xlarge	64	976	1x1920 SSD	10 Gbps	5 Gbps



# PARALLEL PROGRAMMING ON HIGH-END INSTANCES

---



- NUMA Architecture and Programming
  - [https://en.wikipedia.org/wiki/Non-uniform\\_memory\\_access](https://en.wikipedia.org/wiki/Non-uniform_memory_access)
  - <http://cseweb.ucsd.edu/classes/fa12/cse260-b/Lectures/Lec17.pdf>
- Shared Memory Programming with Pthread and OpenMP
  - [https://people.eecs.berkeley.edu/~demmel/cs267\\_Spr11/Lectures/lecture06\\_sharedmem\\_jwdkay11.ppt](https://people.eecs.berkeley.edu/~demmel/cs267_Spr11/Lectures/lecture06_sharedmem_jwdkay11.ppt)
- NUMA Balancing in Kernel
  - [www.linux-kvm.org/images/7/75/01x07b-NumaAutobalancing.pdf](http://www.linux-kvm.org/images/7/75/01x07b-NumaAutobalancing.pdf)

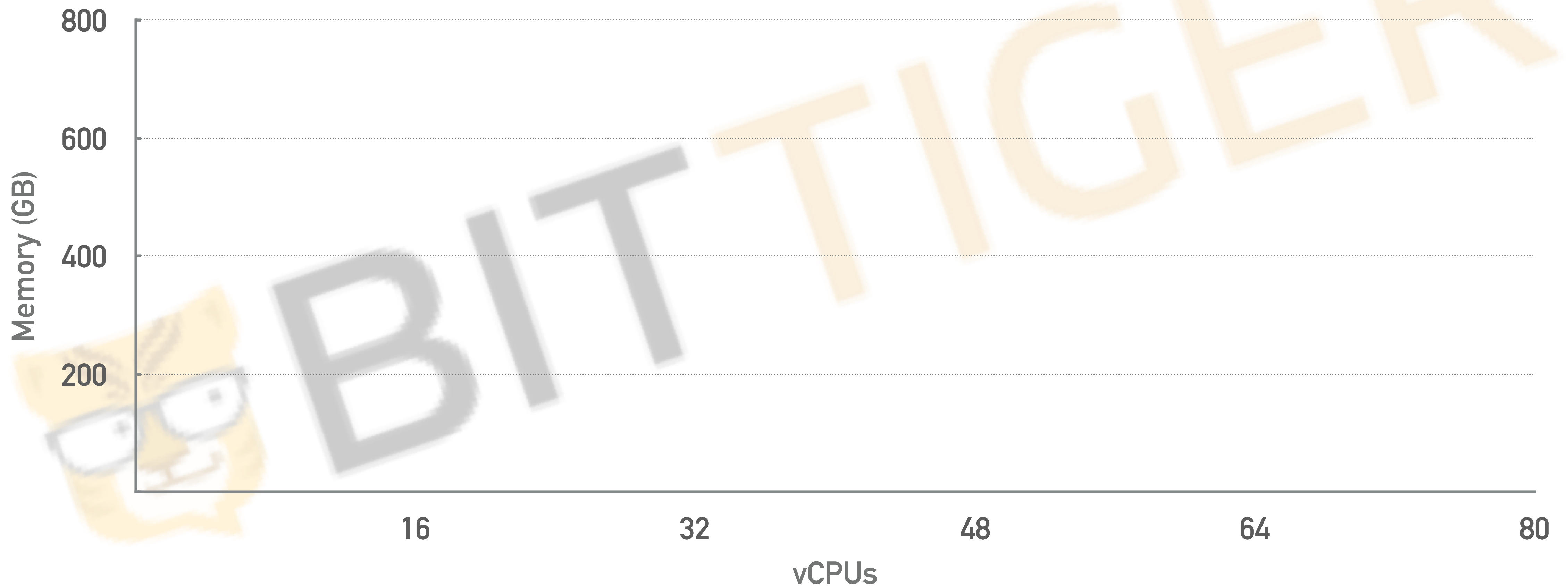
# EC2 - COST

---



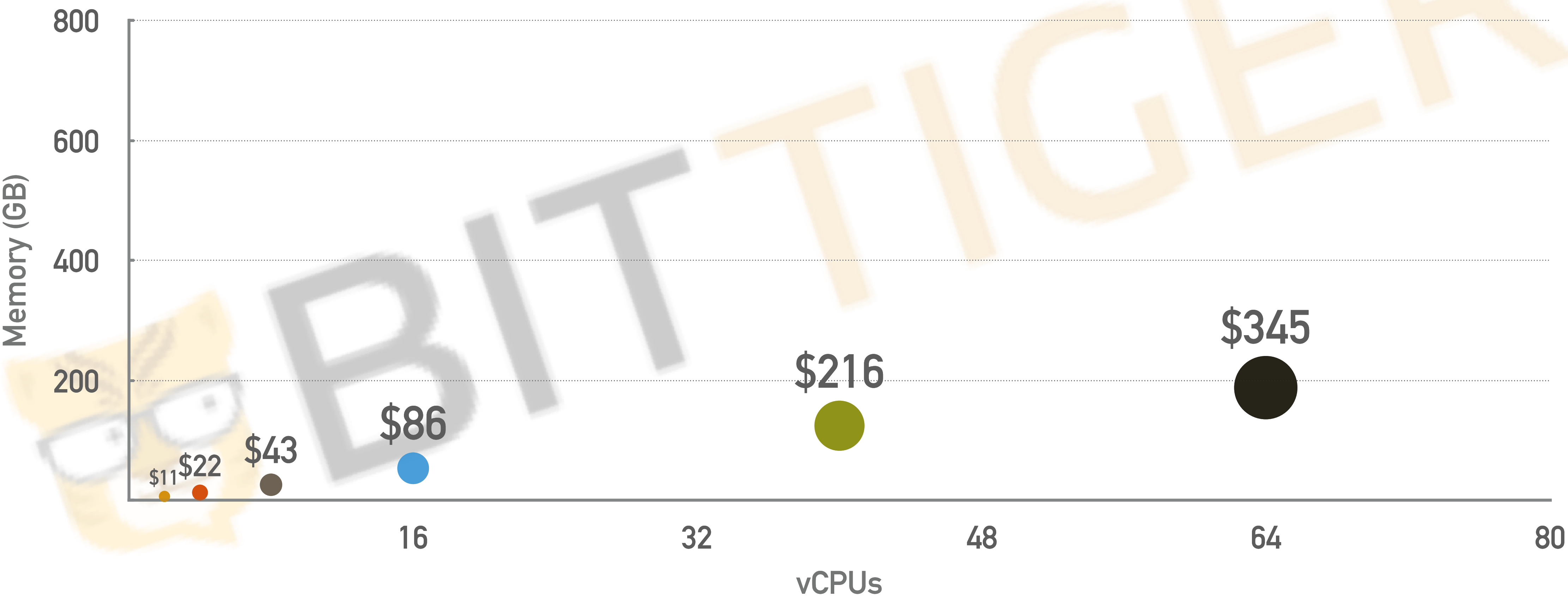
BITTIGER





# EC2 - COST

EC2 Instance Types vs. Cost (US\$/100 hours)

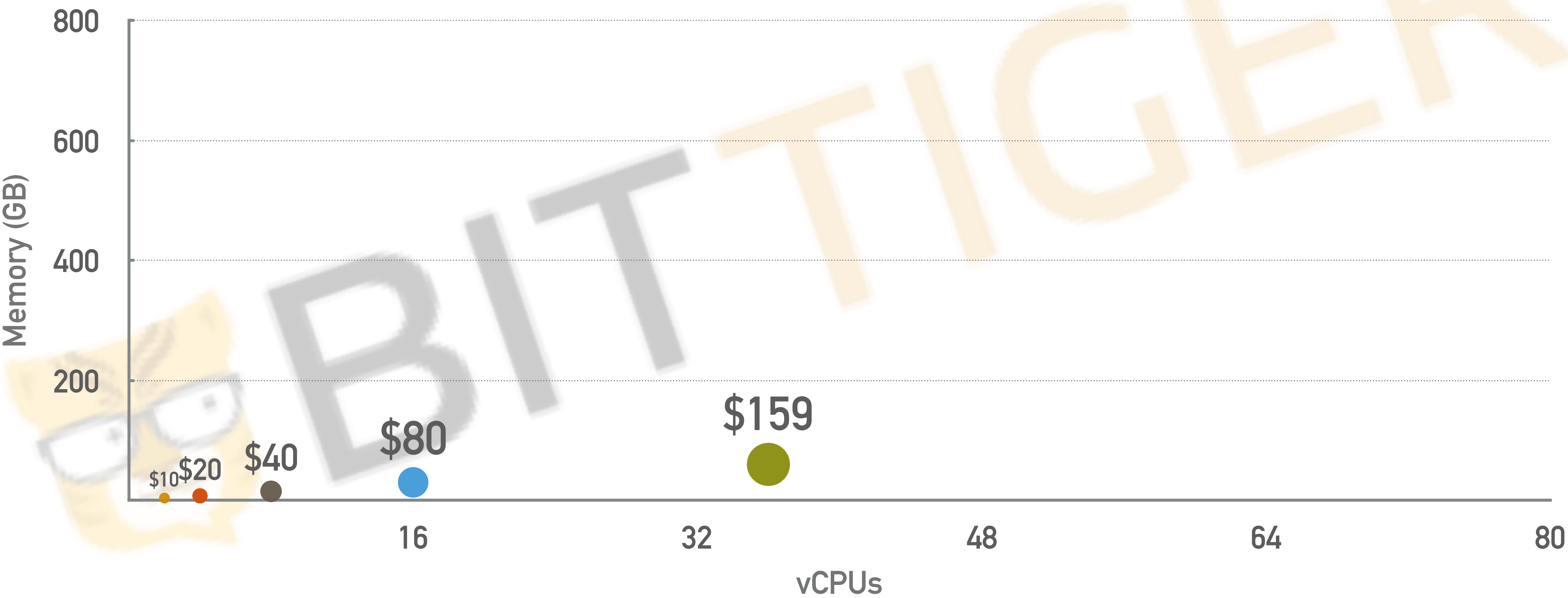


General Purpose (m4)



# EC2 - COST

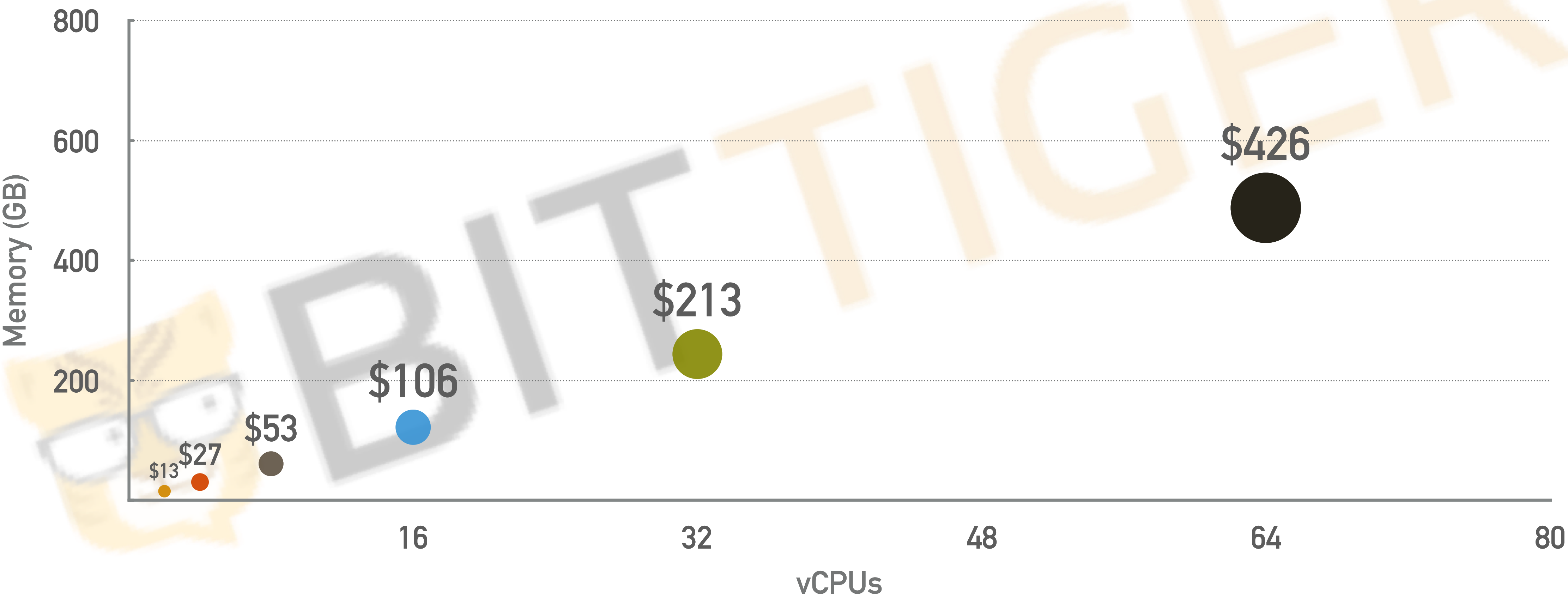
EC2 Instance Types vs. Cost (US\$/100 hours)



Compute Optimized (c4)

# EC2 - COST

EC2 Instance Types vs. Cost (US\$/100 hours)

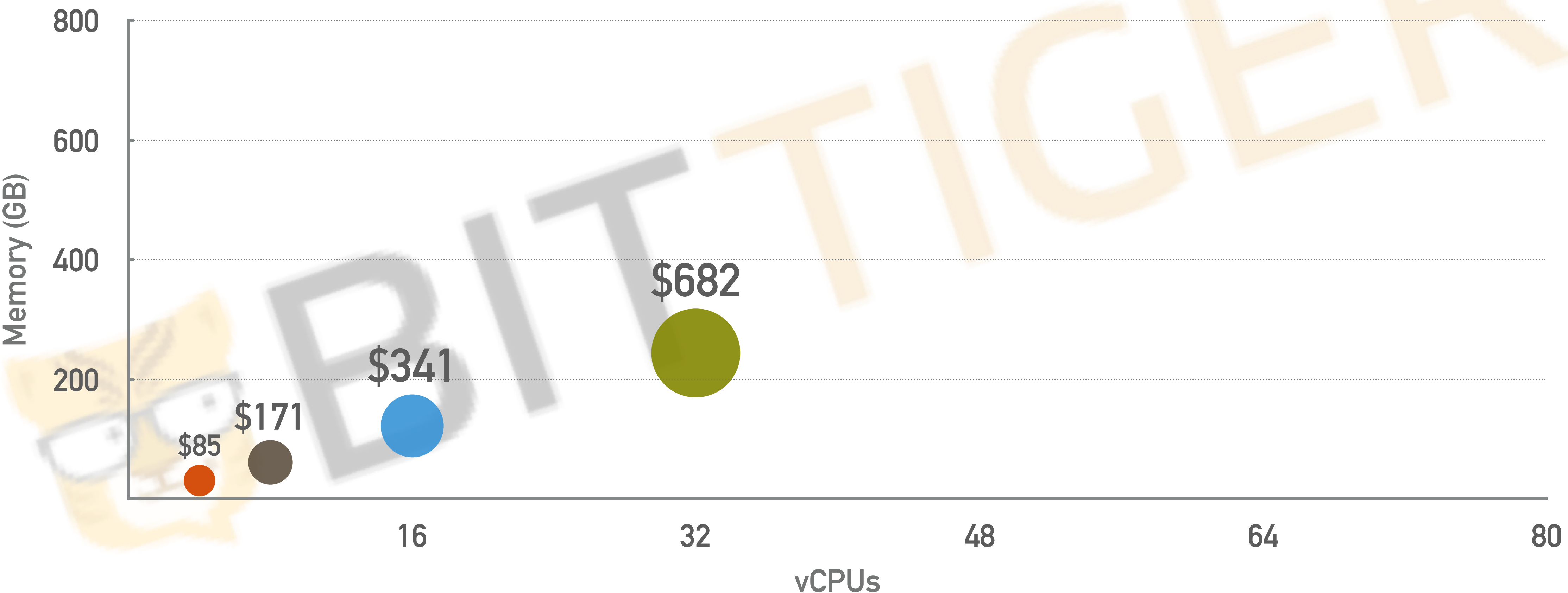


Memory Optimized (r4)



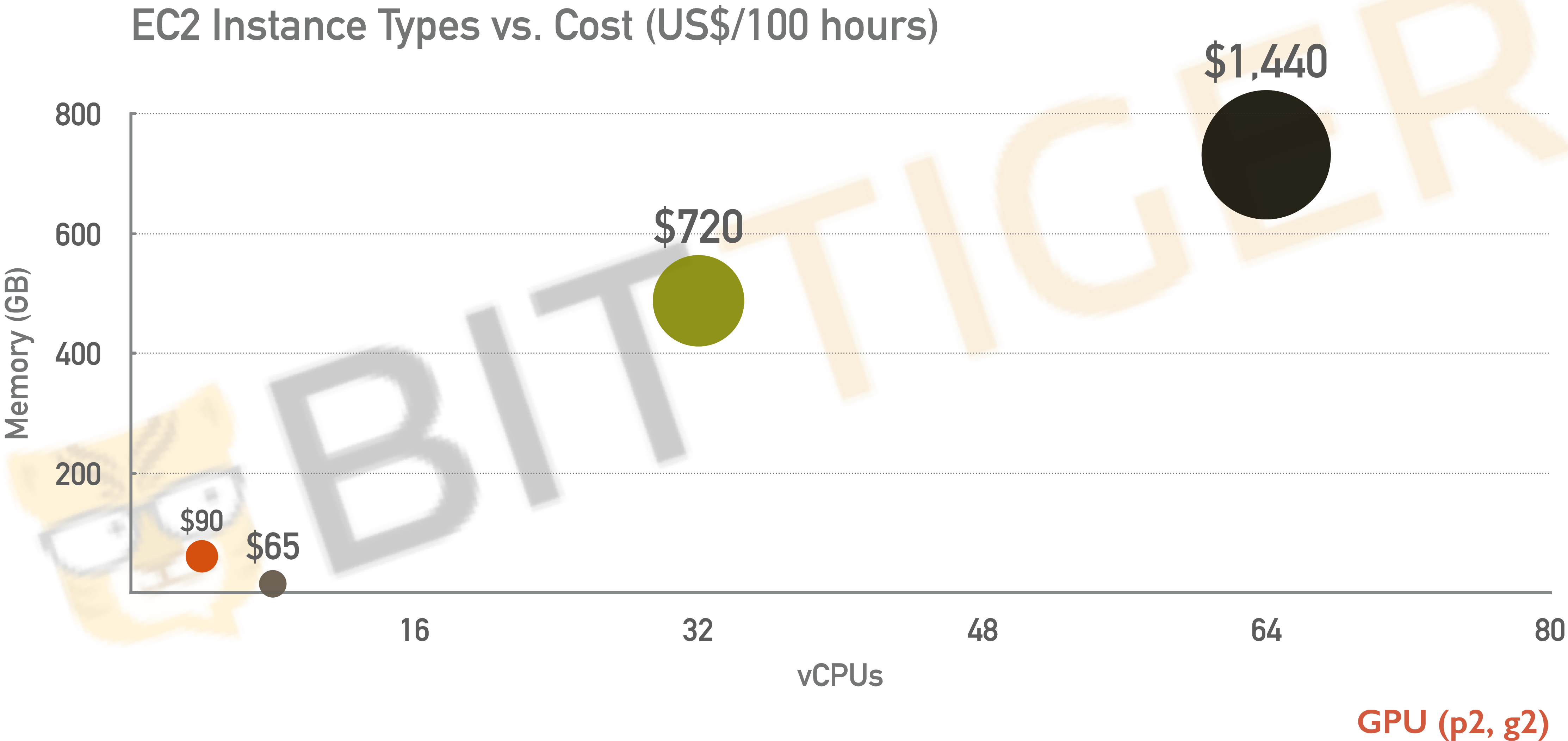
# EC2 - COST

EC2 Instance Types vs. Cost (US\$/100 hours)



Storage Optimized (i2)

# EC2 - COST





# EC2 – TIMESOURCE

---



- TSC vs. Xen
  - `gettimeofday()`, `clock_gettime()`, ...
  - TSC (Time Stamp Counter)
    - Userspace accessible CPU counter
  - How to find and choose time source?
    - `/sys/devices/system/clocksource`
- Homework
  - Write a Python program (less than 10 LoC) to measure the performance difference by using different time source on CentOS7

# EC2 – CPU P-STATE VS. C-STATE

---



- P-State
  - A all-power-on performance state, not necessary running
- C-State
  - A CPU state
  - Low-power mode
  - Higher-latency to exit
  - Limit by adding “intel\_idle.max\_cstate=1” in /boot/grub/grub.conf
- <http://haypo.github.io/intel-cpus.html>
- <https://software.intel.com/en-us/articles/power-management-states-p-states-c-states-and-package-c-states>

# NETWORK PERFORMANCE

---



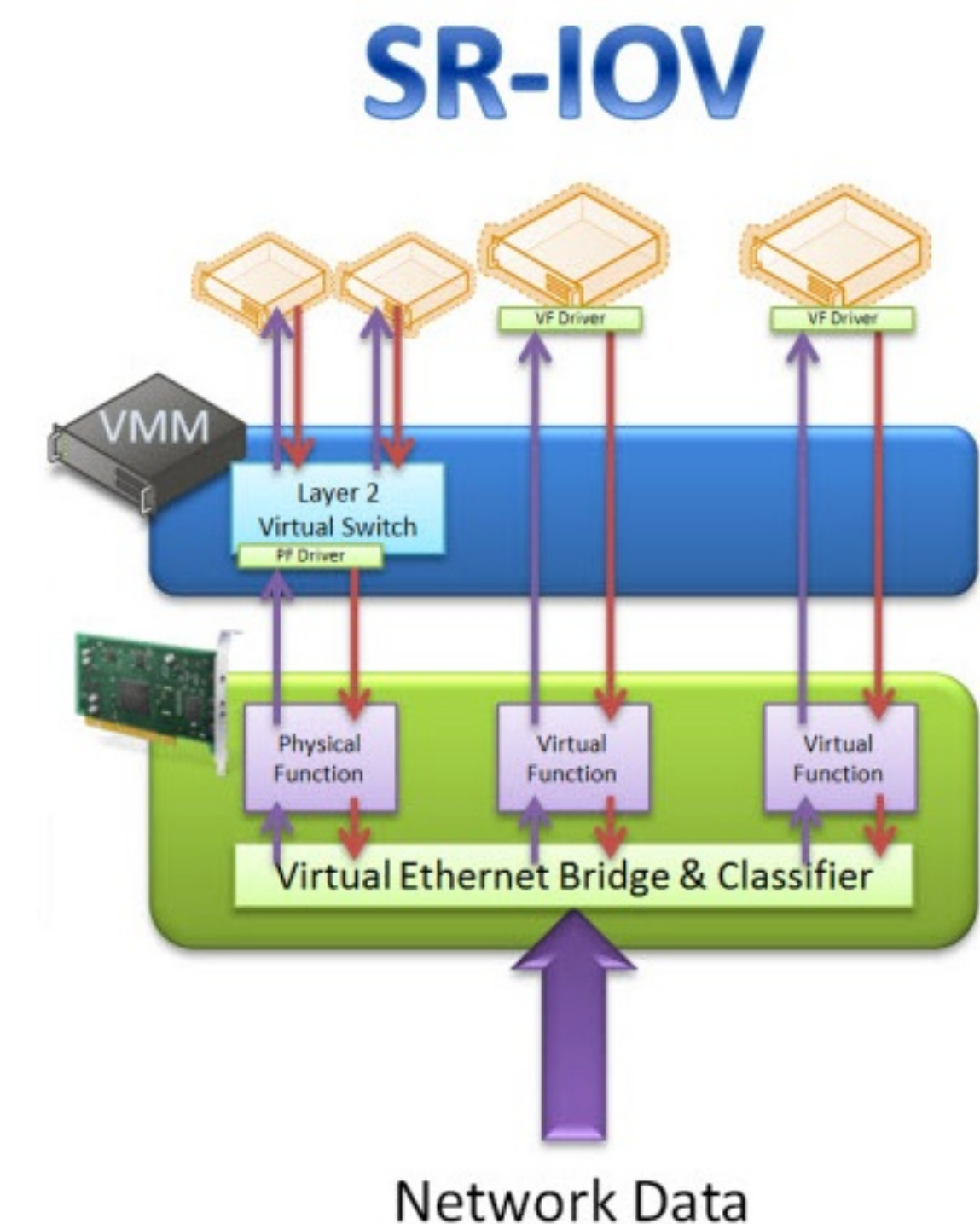
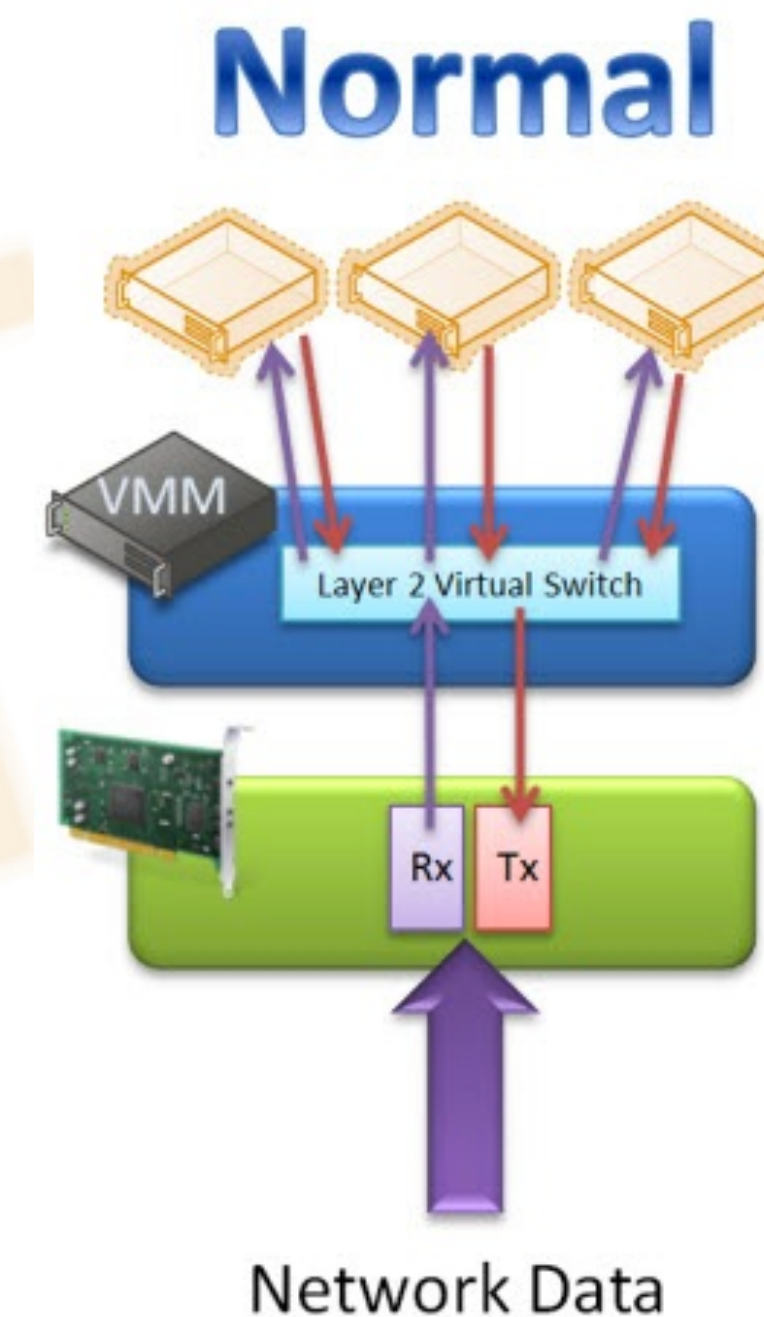
- Instance to instance: 20Gbps and 10Gbps
  - One-way bandwidth
- High, moderate, low?
  - Roughly proportional to instance size and EBS optimization
  - Measure it before use
- Use placement groups, not all instance type
- Homework
  - Measure inter-instance bandwidth with different types, in different AZ, in placement group
- ~5Gbps when outbound from EC2, including S3



# ENHANCED NETWORKING



- Single-root input/output virtualization (SR-IOV)
  - What is SR-IOV? <http://blog.scottlowe.org/2009/12/02/what-is-sr-iov/>
  - Bypass hypervisor for PERFORMANCE reason
  - CentOS 7
    - **Disable Consistent Networking!**
      - <http://stackoverflow.com/questions/30970695/amazon-ec2-how-to-install-ixgbevf-on-a-centos-7-instance>
  - Comes with Amazon Linux AML: <http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/enhanced-networking.html>



# STORAGE TYPES

---

- **Block**

- EC2 Instance Store
- Elastic Block Store (EBS), SSD-Backed
- Elastic Block Store, HDD-backed

- **File**

- Elastic File System (EFS)

- **Object**

- Simple Storage Service (S3)



# EC2 INSTANCE STORE (EPHEMERAL)

---



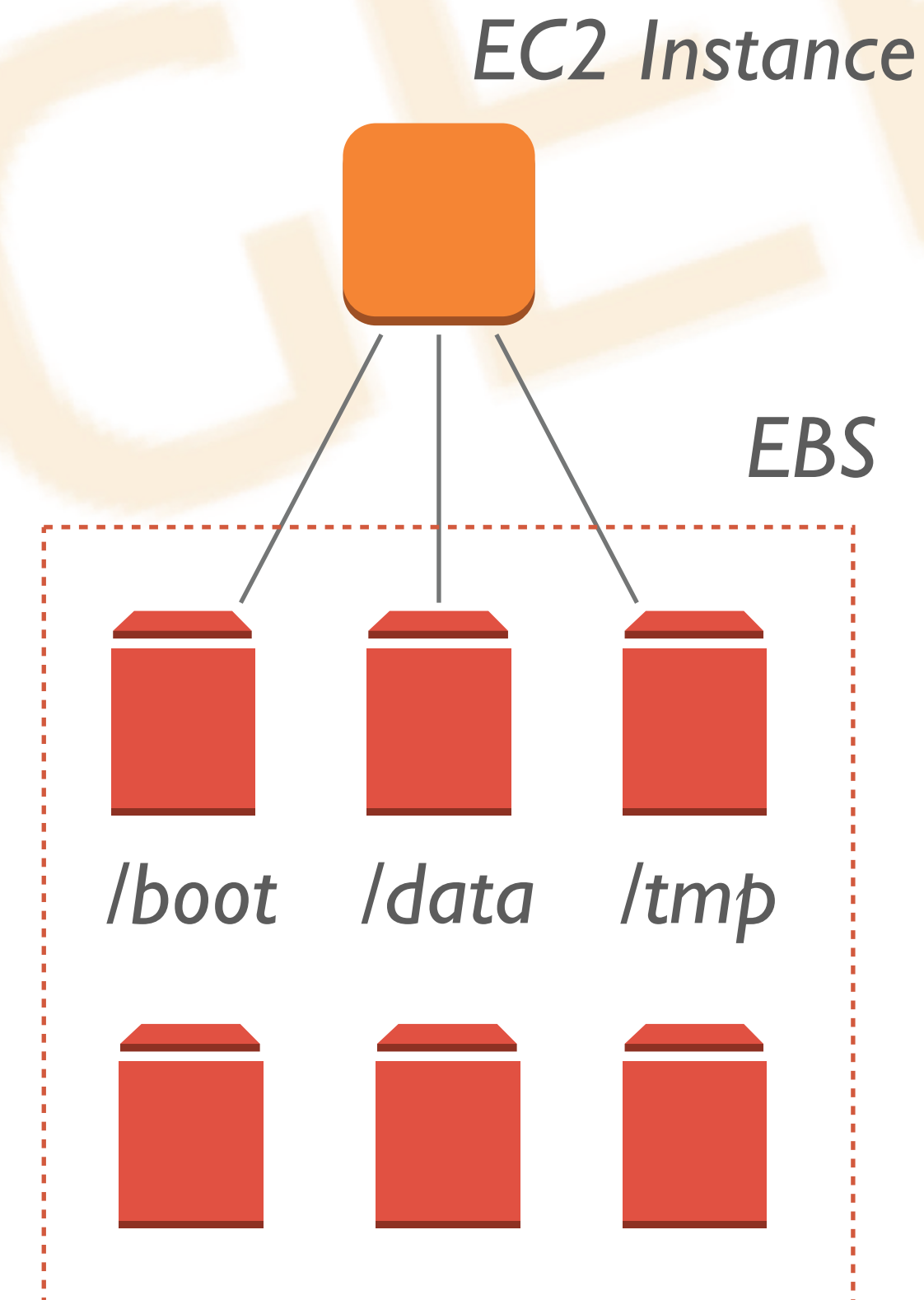
- Lifetime
  - Temporary
  - Lost on disk fails, instance stop/terminate
  - Cannot be detach and reattach
- Volumes
  - <http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/InstanceStorage.html>
- NVMe SSD Volumes (i3 instance)
  - Needs kernel tuning, work with some CentOS kernels
  - Amazon Linux support NVMe by default



# EBS



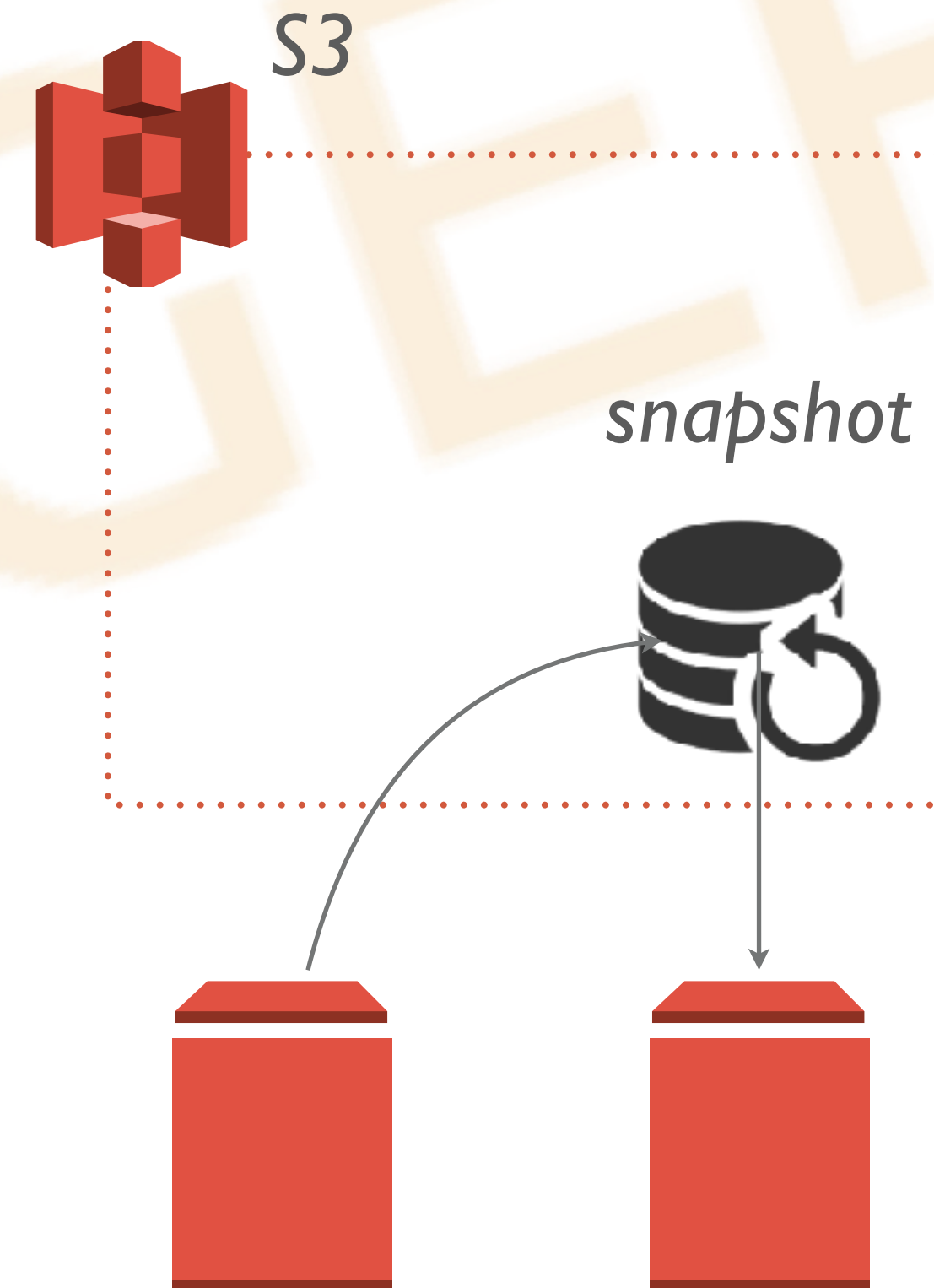
- Similar as network drive or NAS with backups
  - Homework: mount/umount a EBS online
  - `sudo fdisk -l`  
`sudo mkfs -t ext4 /dev/xvdf`  
`mkdir /mnt`  
`sudo mount /dev/xvdf /mnt`  
`mount -l`
- Lifetime
  - Independent from instance
  - Terminated with instance (on/off)
  - AFR: 0.1% to 0.2%
- Requirement
  - Volume and instance must be in the same AZ
- Instance can have multiple EBS volumes
- EBS volume can be attached to only one instance at a time



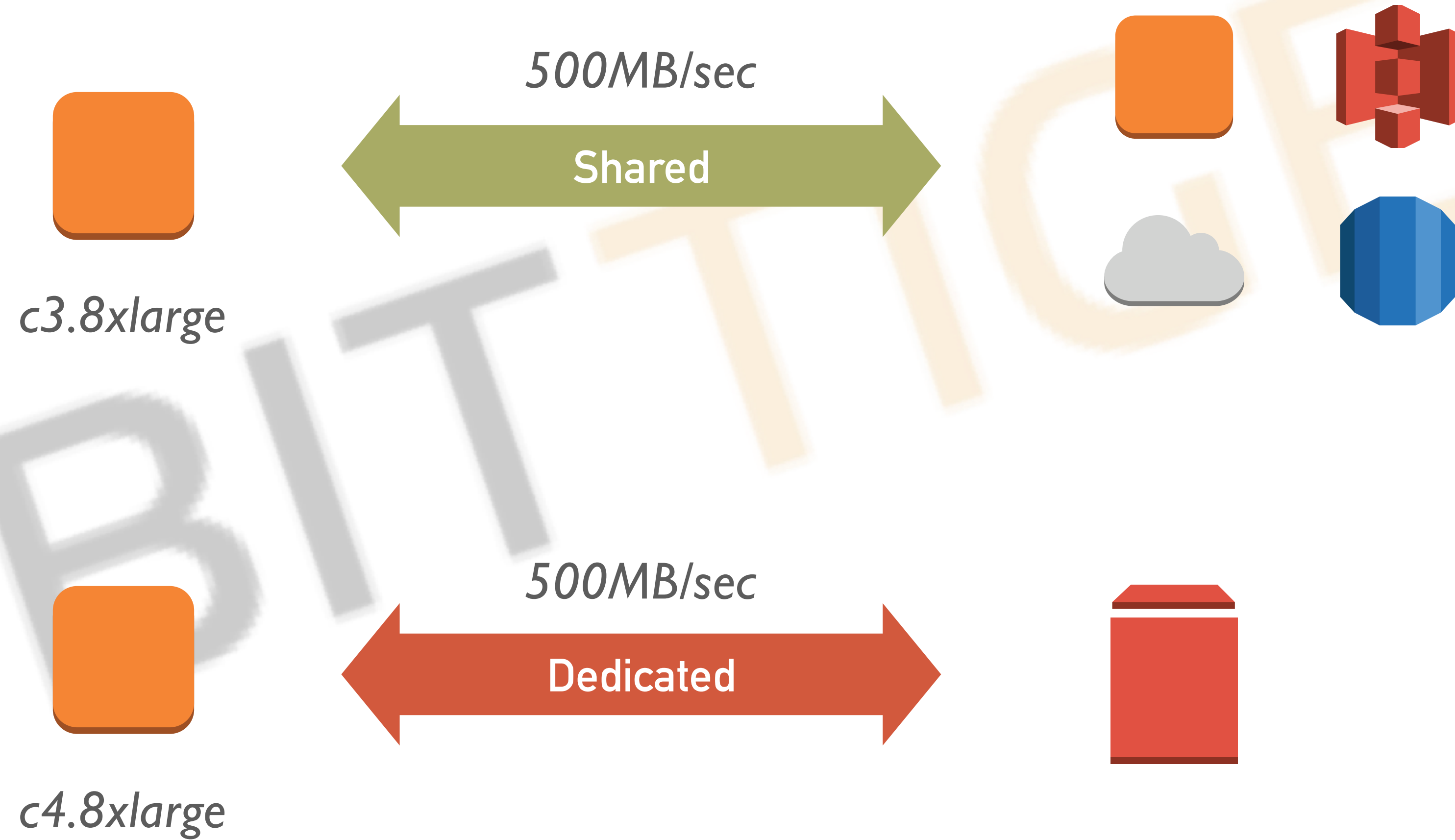
# EBS SNAPSHOT



- S3-backed EBS volume duplicate
  - Create AMI
  - Accessible across availability zones
  - Incrementally backed up
- Public datasets: <https://aws.amazon.com/public-datasets/>



# EBS OPTIMIZED INSTANCE





# EBS ENCRPTION



- EBS Encryption
  - <http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/EBSEncryption.html>
- Root Volume Encryption
  - <https://aws.amazon.com/blogs/aws/new-encrypted-ebs-boot-volumes/>
- It will be easier to encryption your data instead of volume

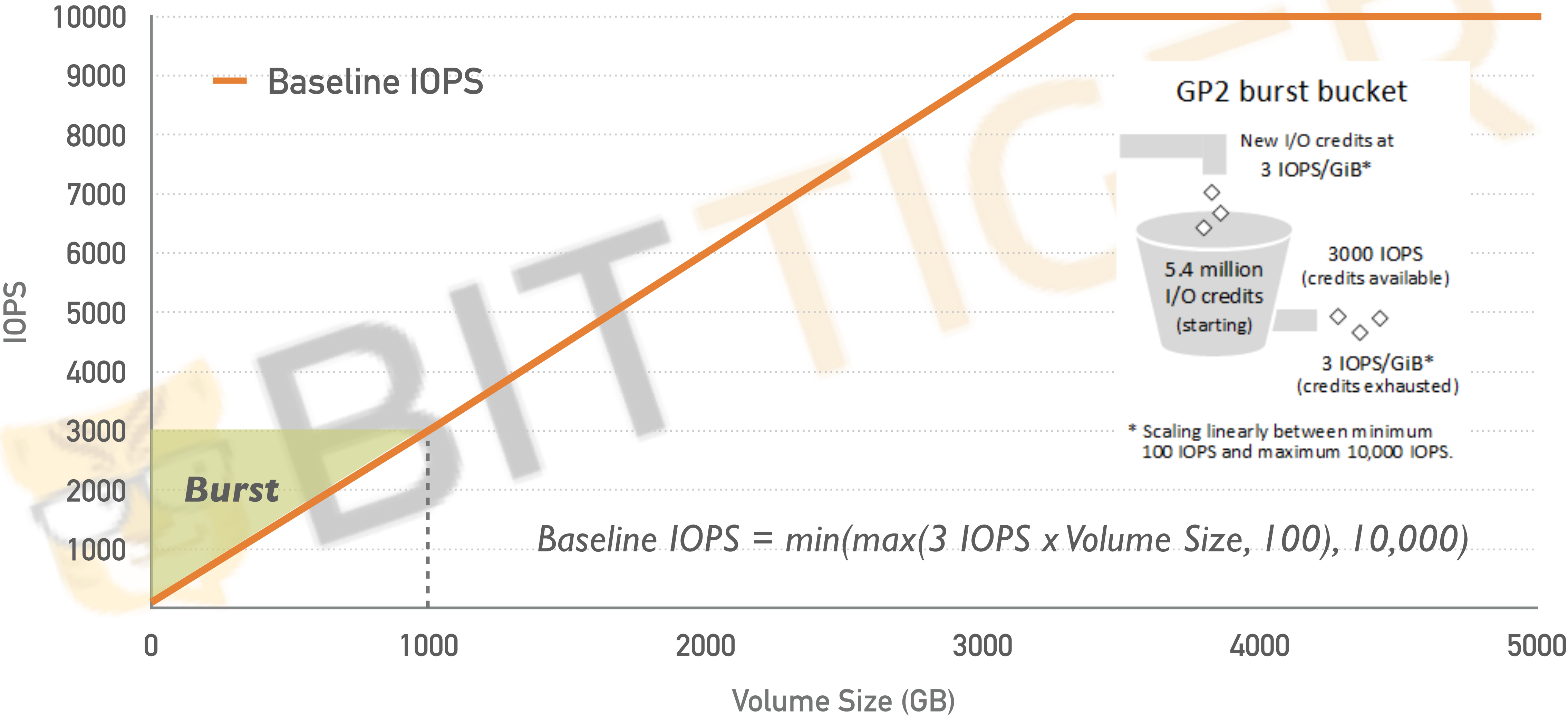


# EBS VOLUME TYPES



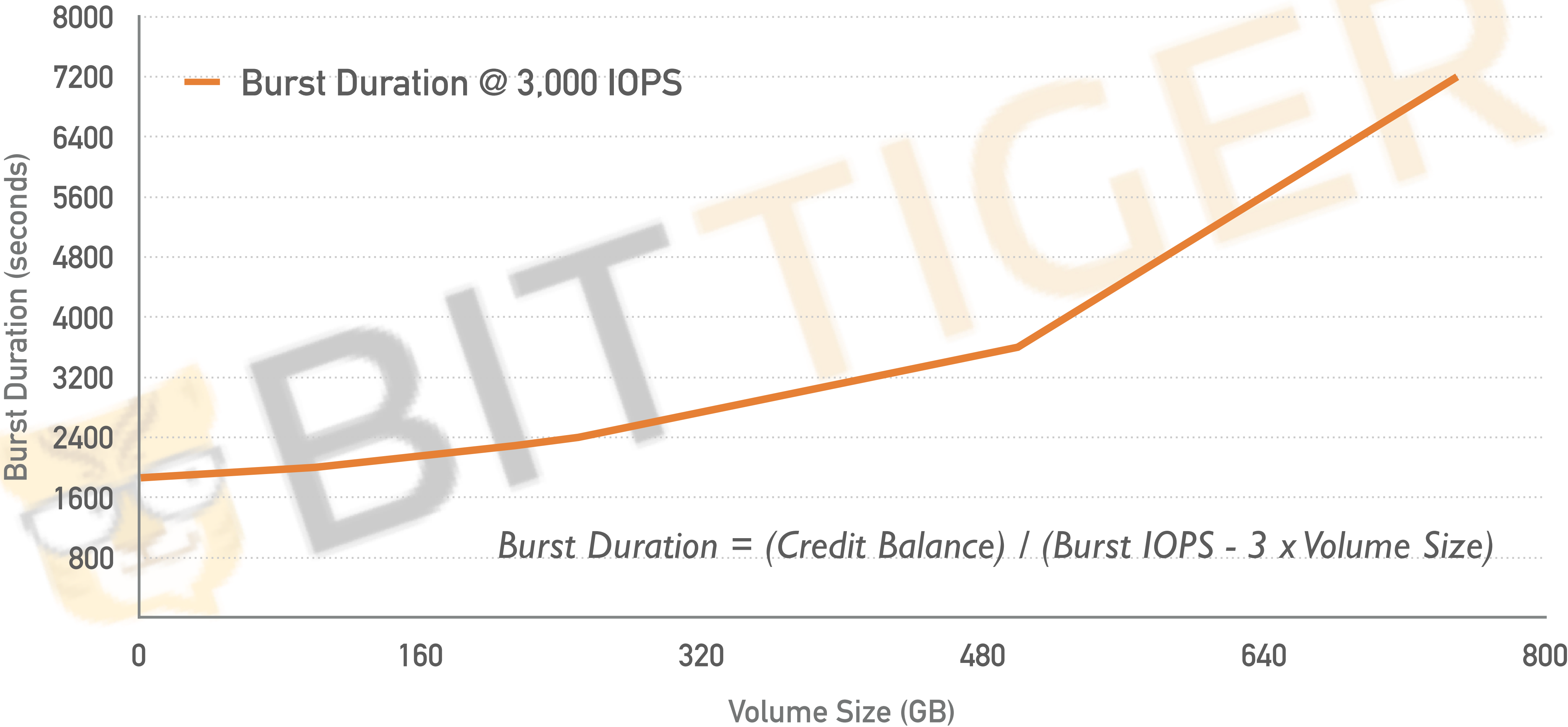
Performance/ Types	gp2	io1	st1	sc1
Baseline	3 IOPS/GB up to 10,000 IOPS	100 ~ 20,000 IOPS	40 MB/s per TB up to 500 MB/s	12MB/s per TB up to 192 MB/s
Burst	3,000 IOPS	Provisioned	250 MB/s per TB up to 500 MB/s	80 MB/s per TB up to 250 MB/s
Throughput	160 MB/s	320 MB/s		
Latency	<10 ms	<10 ms	<20ms	<20ms
Capacity	1 GB ~ 16 TB	4 GB ~ 16 TB	500 GB ~ 16 TB	500 GB ~ 16 TB

# GP2 IOPS

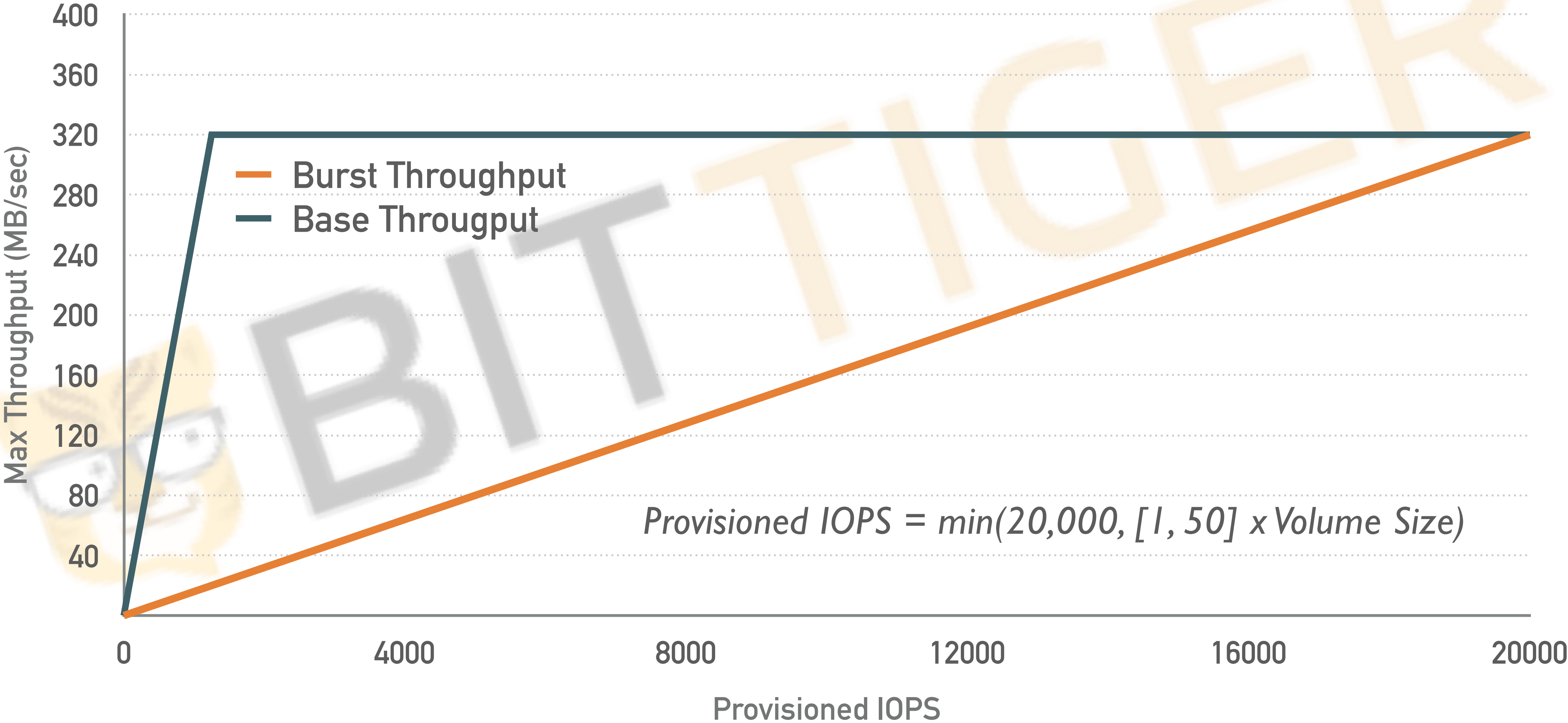




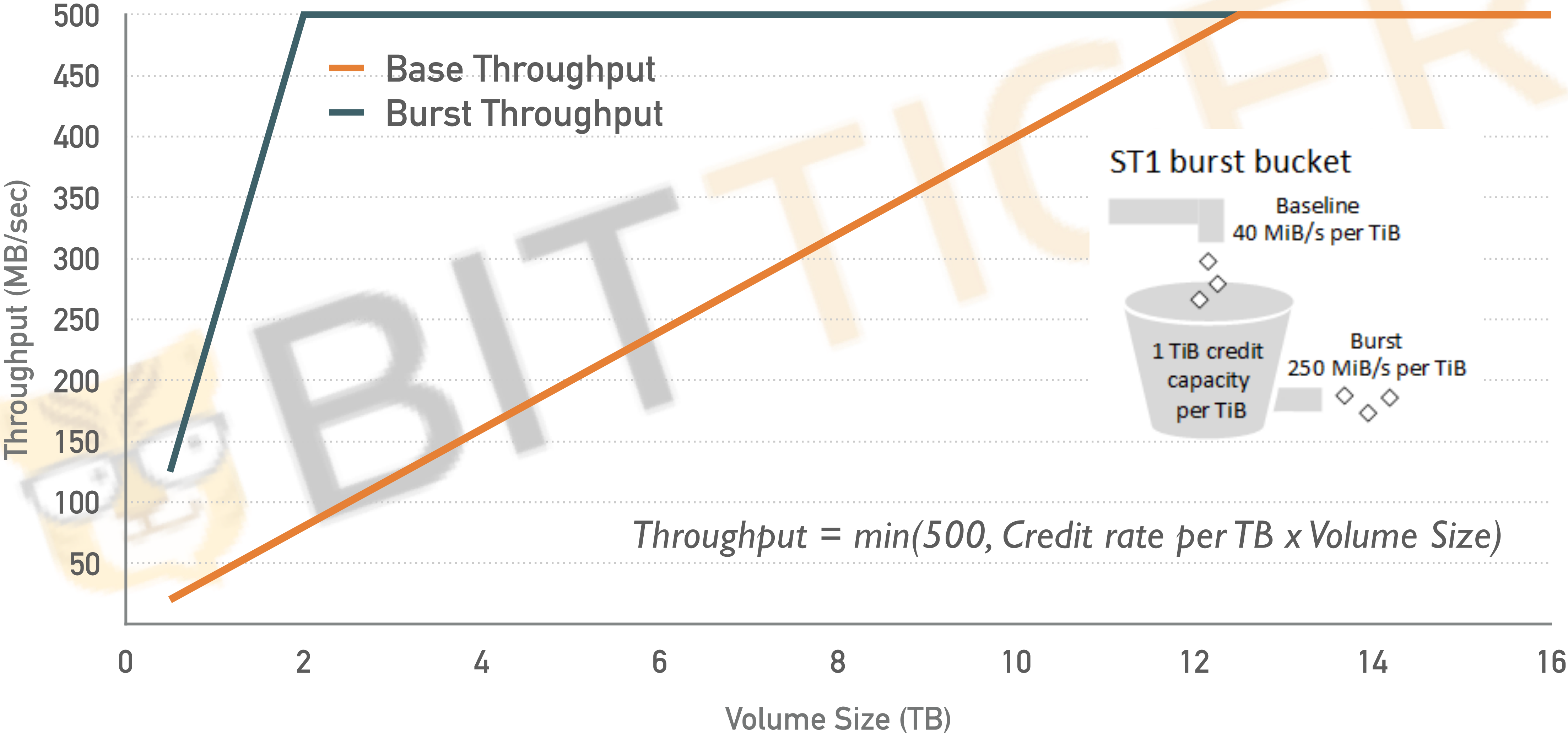
# GP2 BURST DURATION



# IO1 THROUGHPUT

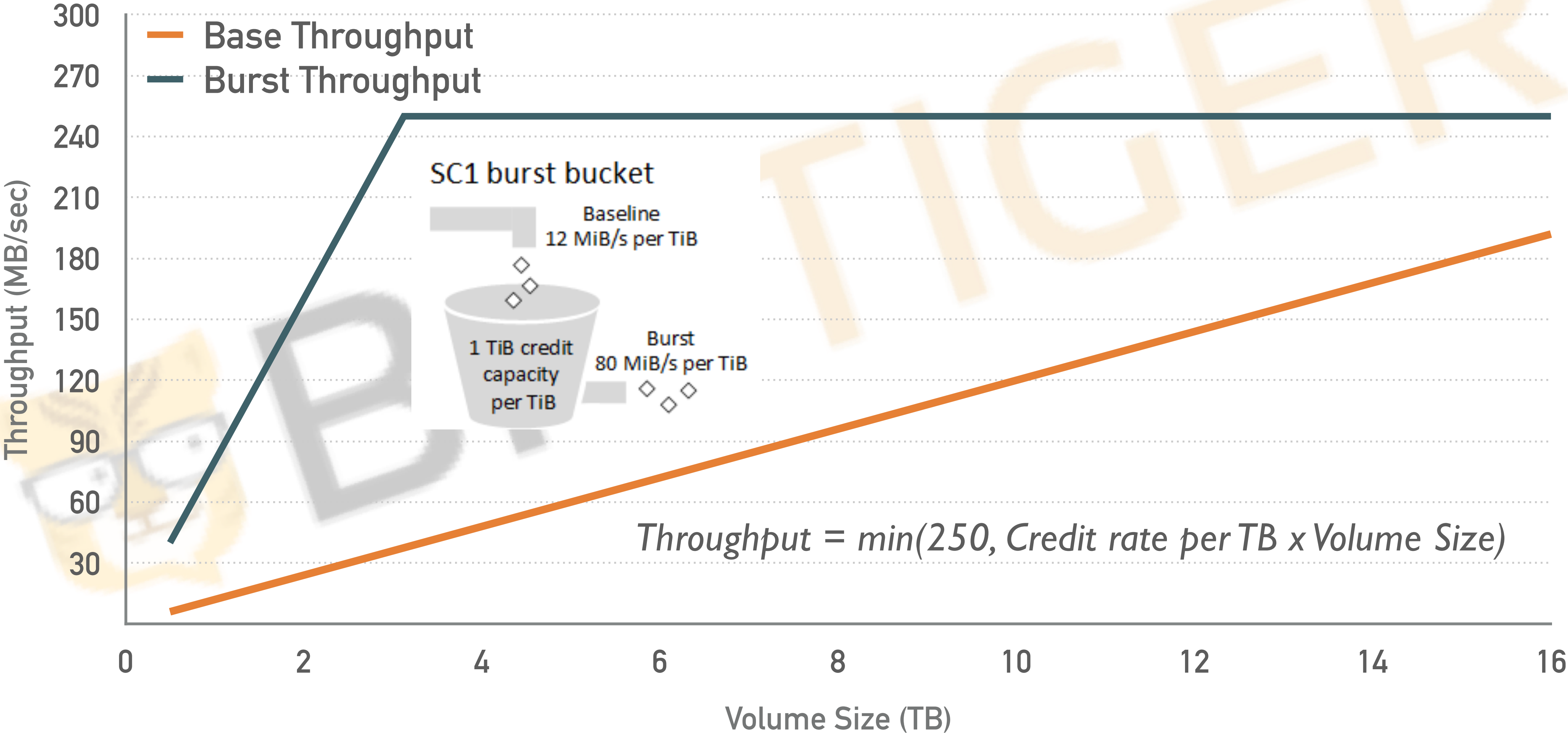


# ST1 THROUGHPUT

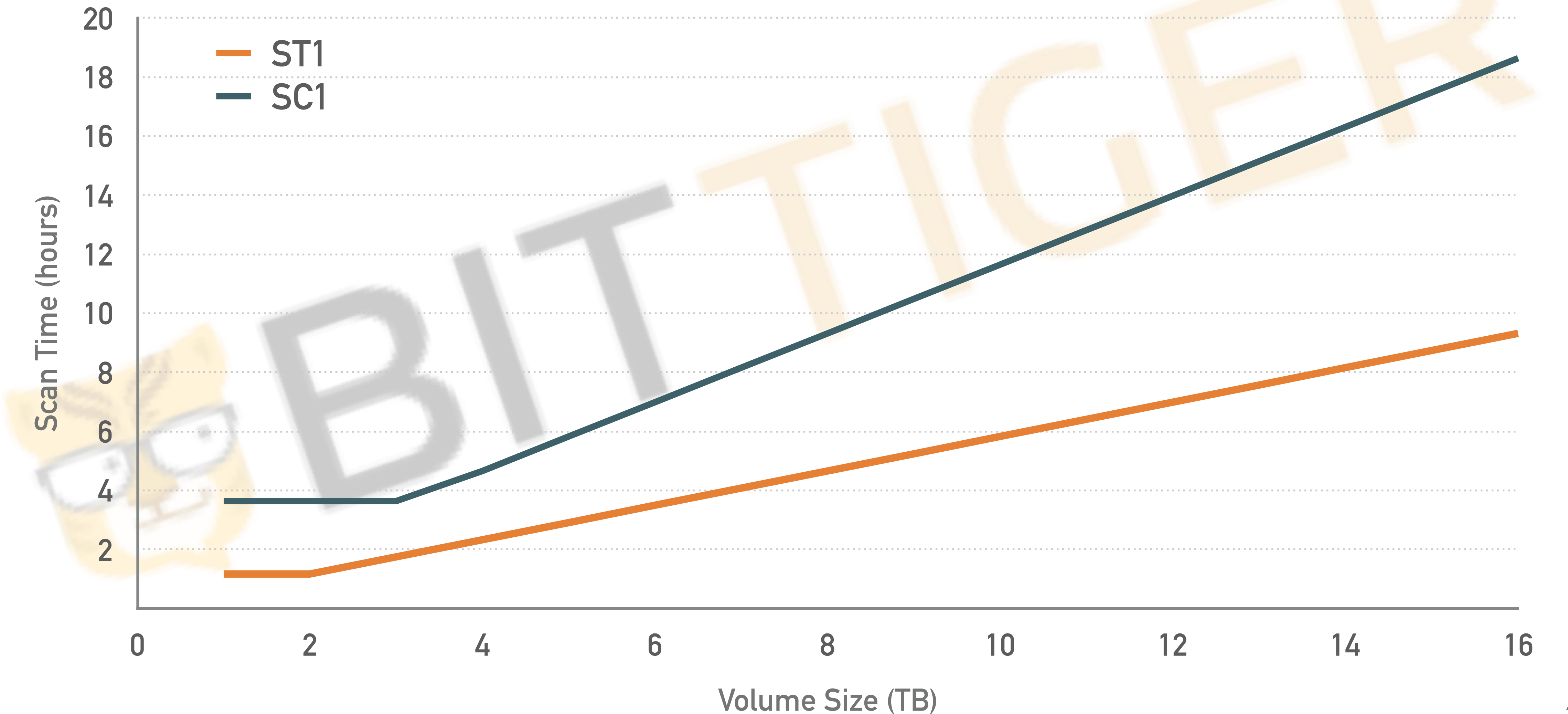




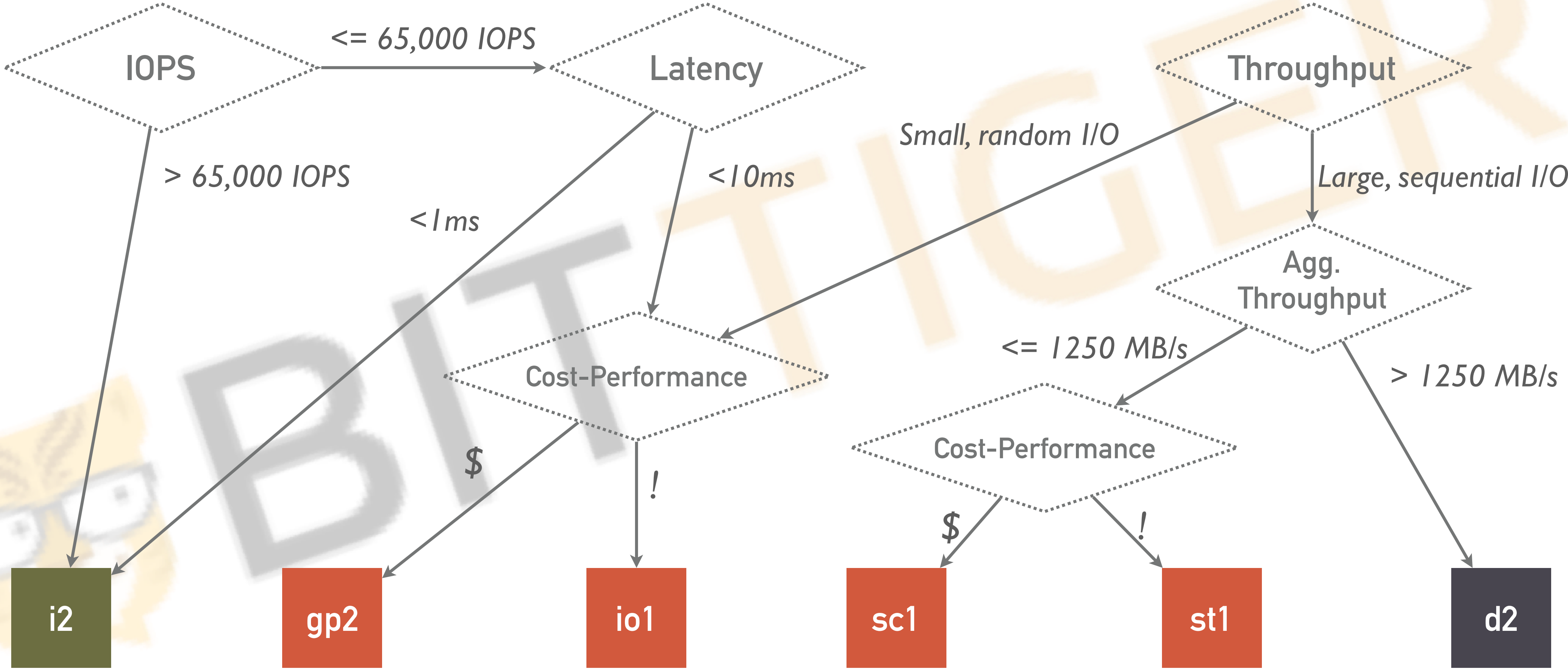
# SC1 THROUGHPUT



# VOLUME SCAN TIME: ST1 VS. SC1



# EBS TRADE-OFF





“

Whenever you find yourself on the side of majority, it is time to pause and reflect.

*-Mark Twain*



S3

# S3 – OUTLINE

---


- **Architecture**
- **Basic Concepts**
- **Performance Optimizations**
- **Management**

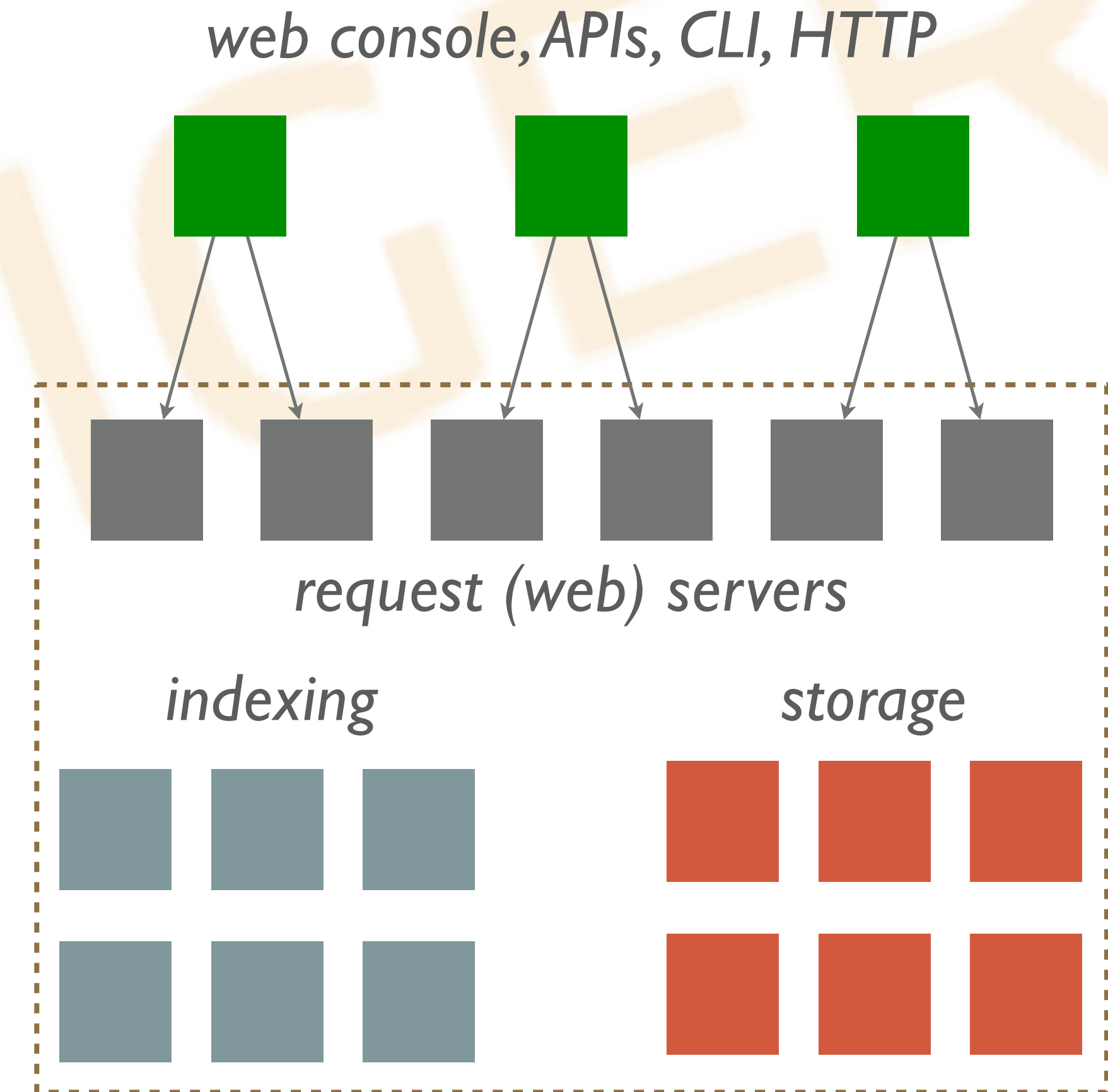


BITTIGER




# S3 ARCHITECTURE

- An object store system, not a file system
- Write once read many
- Eventually consistency 
  - If no new updates are made to a given data item, eventually all accesses to that item will return the last updated value



# S3 – BUCKET



- Naming
  - Global unique name in the whole S3 system
  - DNS-compliant name but (.) not recommended
- Region
  - Cross-region traffic costs \$\$
- Access Control (IAM, Week 2)
- Website hosting 
  - `<bucket-name>.s3-website-<AWS-region>.amazonaws.com`
- Charge for both storage and requests



# S3 – OBJECT

---

- Key/Name
  - UTF-8 encoding, at most 1024 bytes
  - Alphanumeric characters [0-9a-zA-Z]
  - Special characters !, -, \_, ., \*, ', (, and )
- Object size is up to 5TB
- Max 5GB in one single upload request





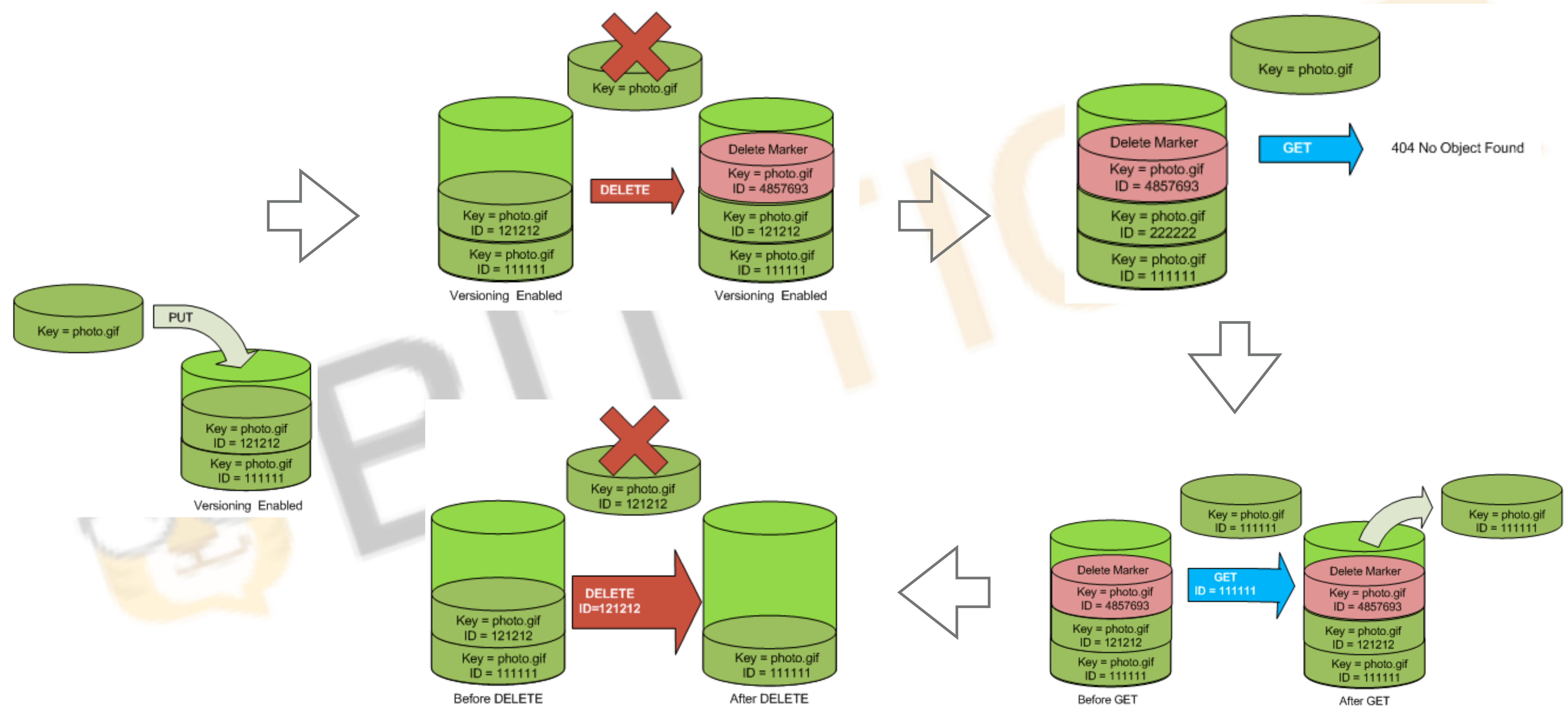
# S3 – STORAGE CLASS

---

Storage Classes	Usage	Use Cases
Standard	Active data	Big Data, Content Distribution, Web Hosting
Infrequent Access	Infrequent Access Data	Backup, Recovery, File sync and Share
Reduced Redundancy	Active data (non-critical, reproducible)	(as standard)
Glacier	Archive	Long-term archives

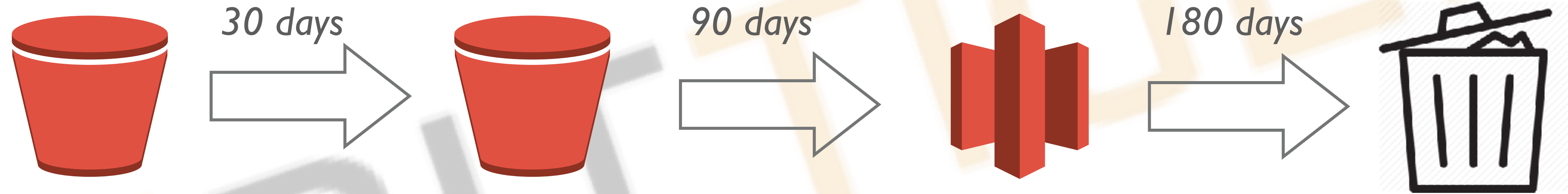
Price calculator: <http://calculator.s3.amazonaws.com/index.html>

# S3 - OBJECT VERSIONING



# S3 - LIFECYCLE

---



S3

S3 IA

Glacier

*Tradition Actions*

*Expiration Actions*



# S3 – LOAD BALANCING



- Key distribution and Partition
  - Use a well-distributed key prefix, e.g., **hash, UUID, md5, reverse epoch time**

service\_log.2012-02-27-23.hostname1.mydomain.com

service\_log.2012-02-27-23.hostname2.mydomain.com

service\_log.2012-02-27-23.hostname3.mydomain.com

service\_log.2012-02-27-23.hostname4.mydomain.com

service\_log.2012-02-27-23.john.myotherdomain.com

service\_log.2012-02-27-23.paul.myotherdomain.com

service\_log.2012-02-27-23.george.myotherdomain.com

service\_log.2012-02-27-23.ringo.myotherdomain.com

service\_log.2012-02-27-23.pete.myotherdomain.com

c/service\_log.2012-02-27-23.com.mydomain.hostname1

4/service\_log.2012-02-27-23.com.mydomain.hostname2

9/service\_log.2012-02-27-23.com.mydomain.hostname3

2/service\_log.2012-02-27-23.com.mydomain.hostname4

b/service\_log.2012-02-27-23.com.myotherdomain.john

7/service\_log.2012-02-27-23.com.myotherdomain.paul

2/service\_log.2012-02-27-23.com.myotherdomain.george

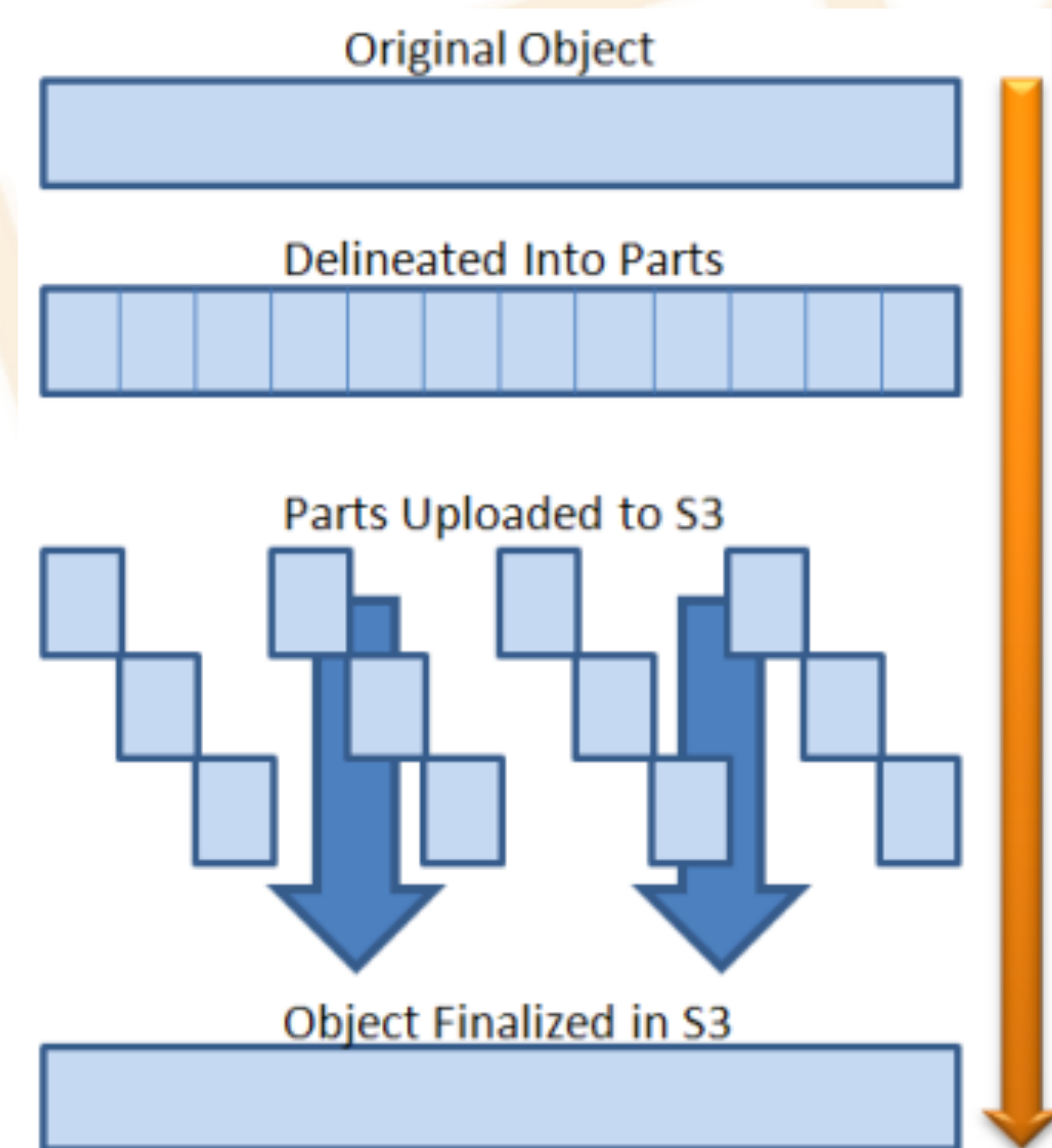
0/service\_log.2012-02-27-23.com.myotherdomain.ringo

d/service\_log.2012-02-27-23.com.myotherdomain.pete

# S3 – MULTIPART UPLOAD



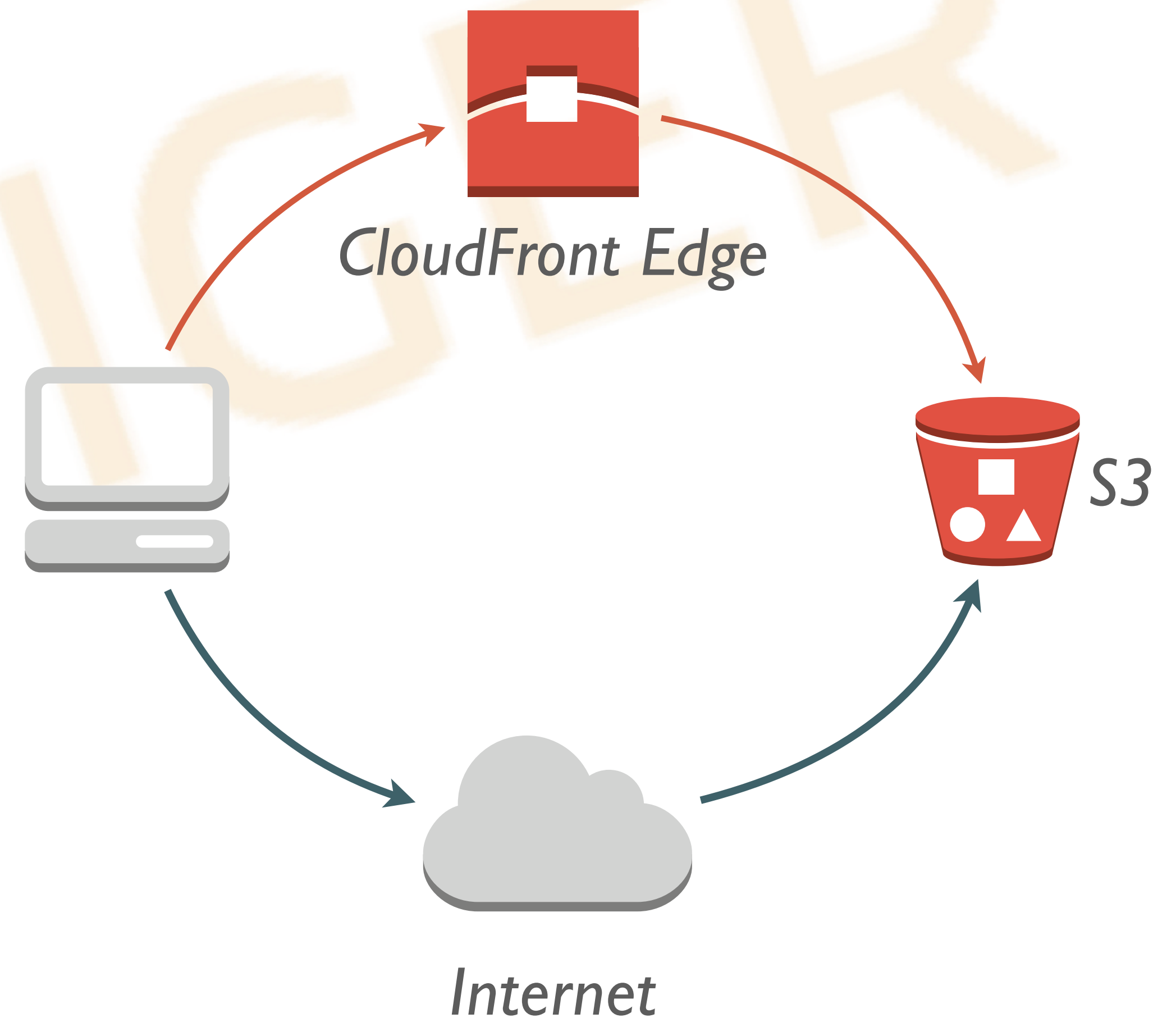
- Increase throughput by parallelizing PUTs and utilizing instance bandwidth
- Reduce failures for large files, fewer retries on failures
- Optimal part size
  - 20-50MB on high speed networks
  - 10MB on mobile networks
- Balance part size and # of parts
  - Small parts: connection overhead
  - Large parts: no parallel throughput
- Use AES-256 for Encryption



# S3 TRANSFER ACCELERATION

---

- Use CloudFront endpoints to speedup the transfer
  - Fast link
  - Short path
- Cost: \$\$
- Not always fast, test before use it: <http://s3speedtest.com>





# S3 – FAST OBJECTS LIST

---



- By LIST Requests
  - AWS CLI and Boto3 listing
  - \$0.005 per 1,000 requests
  - Use Prefix to narrow your listing or do parallel listing
- S3 Inventory
  - List objects **on S3 side, thus save significant round-trip time!**
  - CSV file output to S3 bucket
  - Daily or weekly bases
  - Half price of list API

# S3 – FAST OBJECT GETS

---



- Range based GETs
  - Read part of large file in parallel
  - But you need to align the data
- Out S3
  - Use CloudFront (CDN) for distributing
    - Cache objects (WORM)
    - Low latency due to geo locality



# S3 – LARGE-SCALE THROUGHPUT

---



- How to achieve 100GB/sec read?
  - Key and partition
  - Parallel uploads/downloads
    - around 3,000 parallel uploads
  - DNS lookup performance
    - S3 uses DNS to choose S3 endpoints
    - 10,000 QPS
    - Use Amazon Linux AMI
    - Demo: how AWS extensively use DNS for load balance? (Also by Netflix)



# S3 MANAGEMENT – OBJECT TAGGING

---

- Up to 10 tags per object
- (key, value) tags
- Access control based on tags (IAM, Week 2)
- Lifecycle policy based on tags
- Storage metrics and analytics (CloudWatch, Week 3)
- Put objects with tags and add tags to existing objects
- \$0.01 per 10,000 tags per month, i.e, \$1 for 1 million objects per tag per month

# S3 MANAGEMENT – AUDIT AND MONITORING

---

- Data Events in CloudTrail (CloudTrail, Week 3)
  - Audit data integrity
  - Capture both object-level and bucket-level requests
  - Access logs in S3 bucket
  - \$0.01 per 100,000 data events
- Performance and Operation Monitoring (CloudWatch, Week 3)
  - 1-minute metrics
  - Alerts on metrics
  - \$0.30 per metric per month



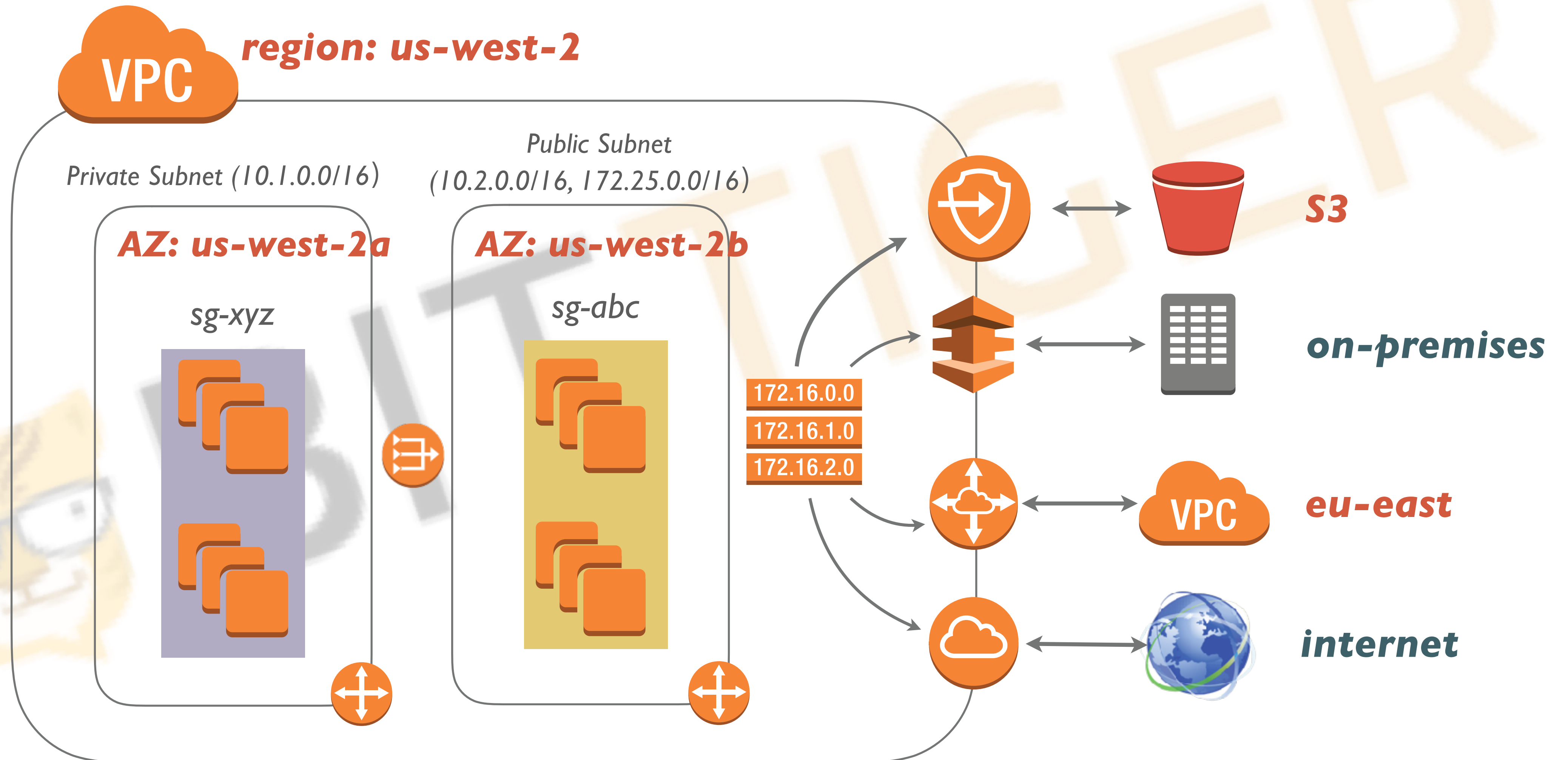
VPC



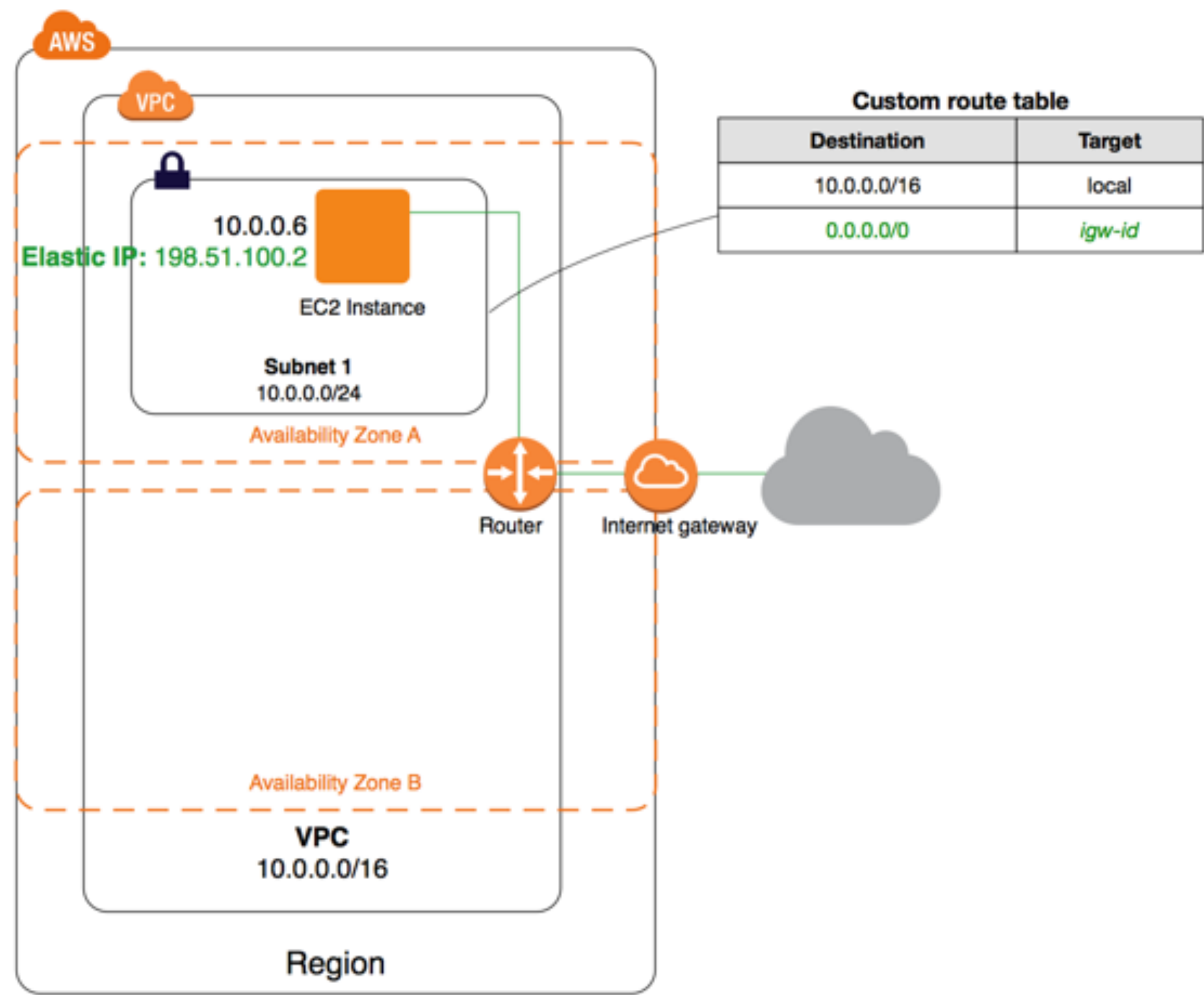
# NETWORKING COMPONENTS

---

- Availability zone
- VPC
- Subnet
- Network interfaces
- Route Tables
- Internet Gateways
- Egret-Only Internet Gateways
- DNS
- Elastic IP Addresses
- VPC Endpoints
- S3 Endpoints
- NAT
- VPC Peering
- Direct link
- Security Groups
- Network ACLs
- VPC Flow Logs

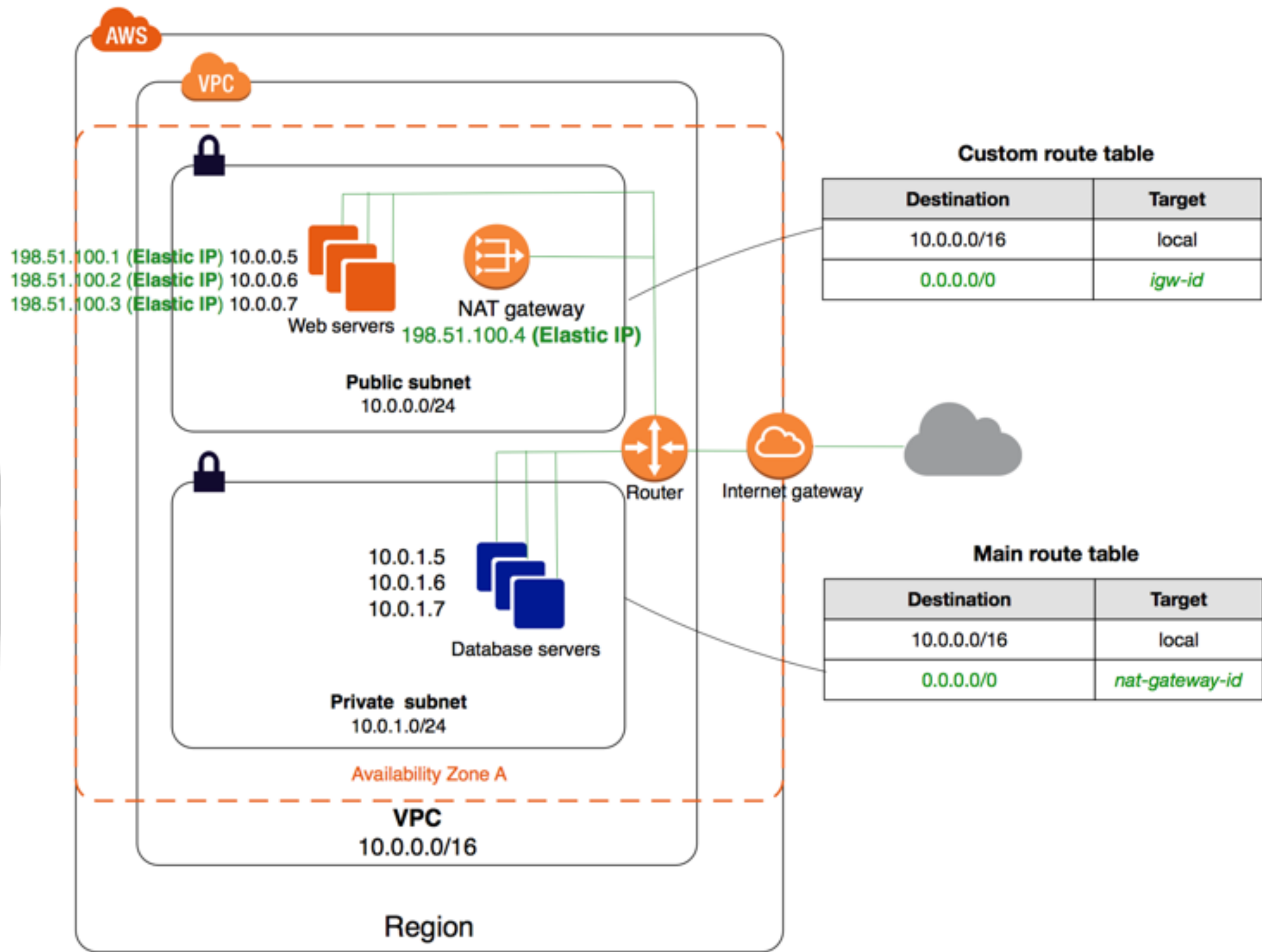


# VPC WITH SINGLE PUBLIC SUBNET

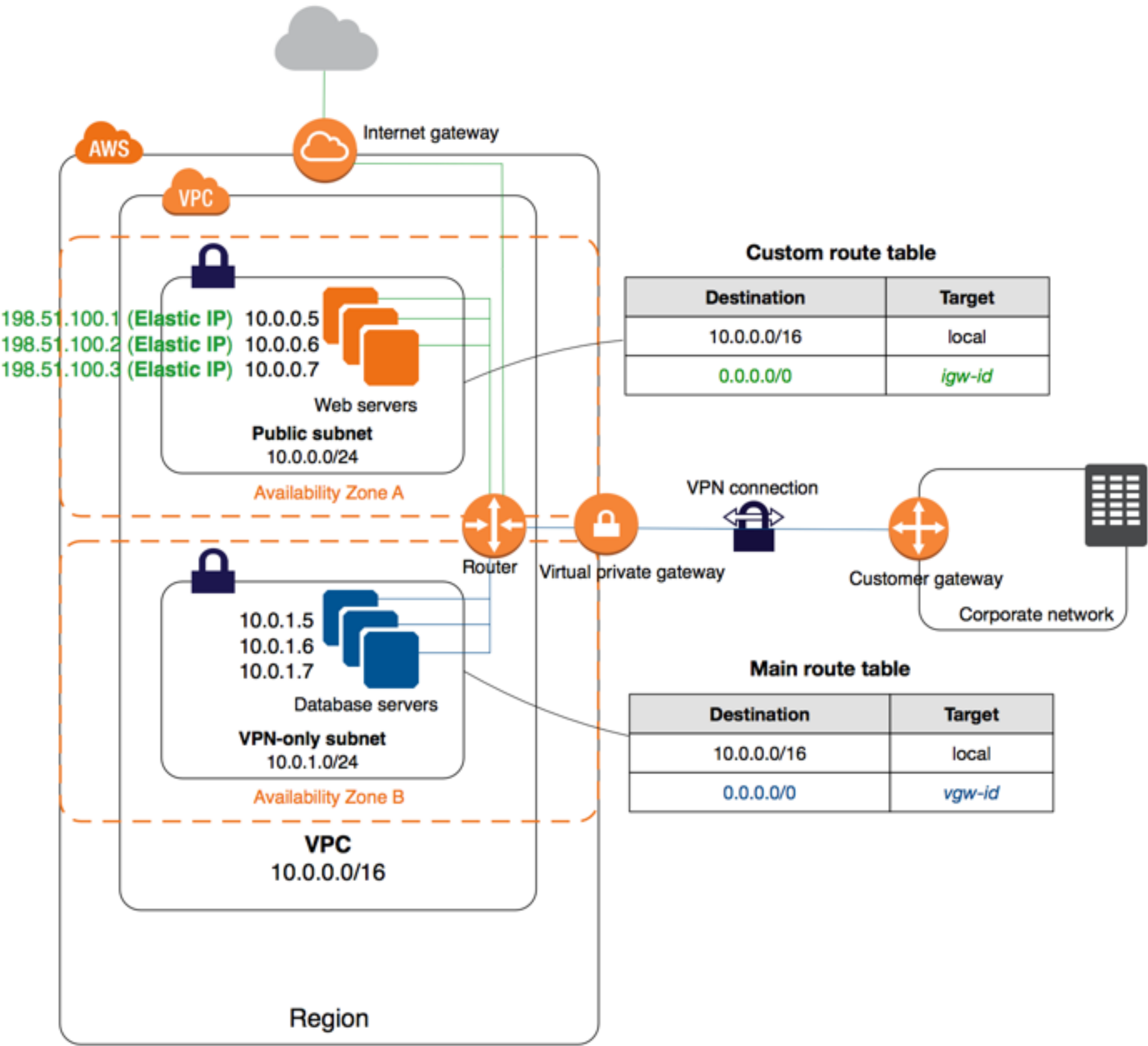




# VPC WITH PUBLIC AND PRIVATE SUBNET



# VPC WITH PUBLIC AND PRIVATE SUBNETS AND HARDWARE VPN ACCESS



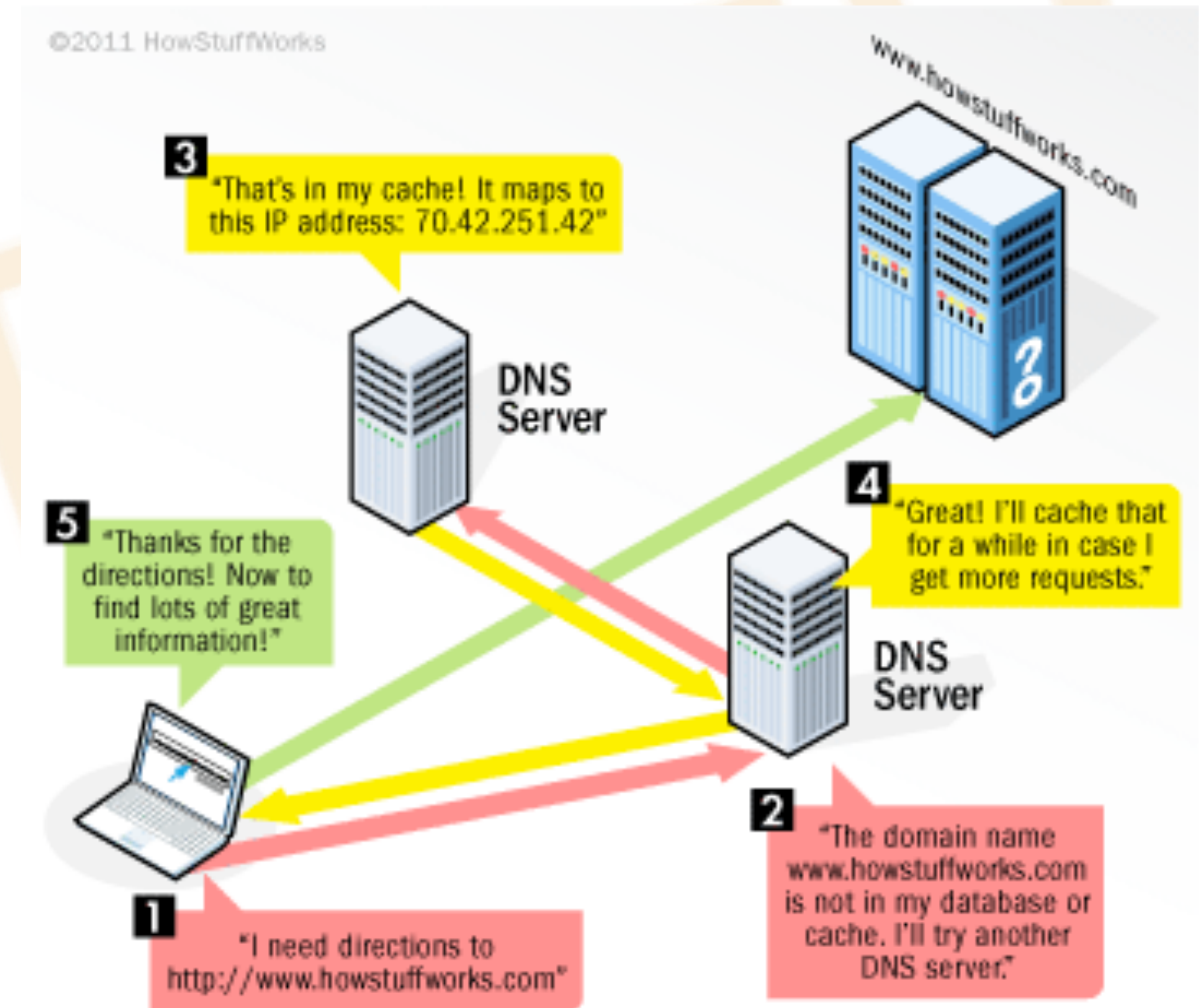
# ROUTE 53





# DNS

- Records Type
  - A: address record
  - NS: name server
  - CNAME: Canonical name record
  - Route53 also support Alias
- Difference
  - <https://support.dnssimple.com/articles/differences-between-a-cname-alias-url/>

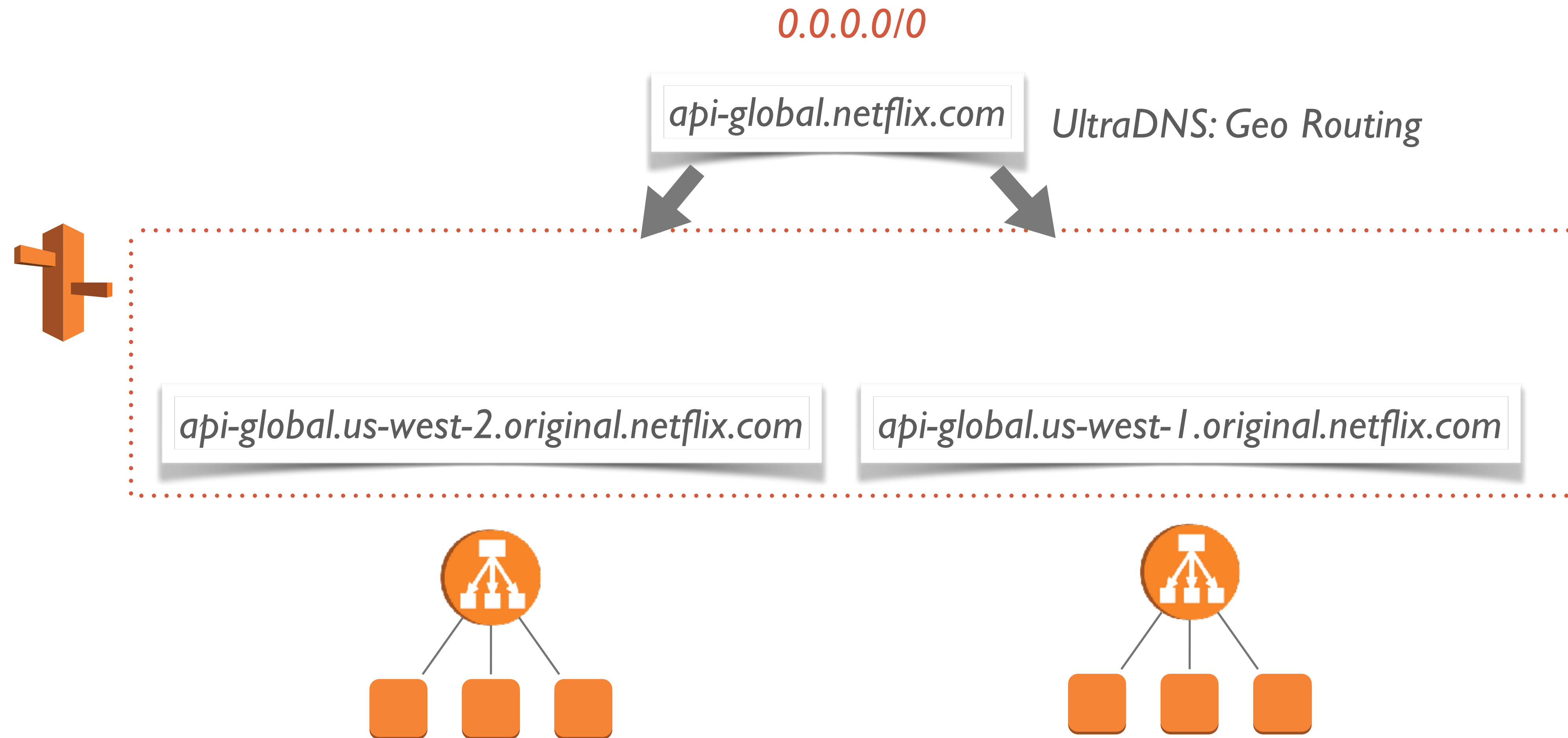


# ROUTE 53

- Amazon DNS service
  - Hosted Zones
  - Traffic Flow
  - Queries
- Homework
  - Start a web server
    - `sudo yum install httpd mod_ssl`  
`sudo /usr/sbin/apachectl start`
  - Create a A record for server
  - Create a CNAME

Cost/Millin Queries	< 1 Billion	>= 1 Billion
Standard	\$0.4	\$0.2
Latency	\$0.6	\$0.3
Geo	\$0.7	\$0.35

# NETFLIX DNS RESILIENCY

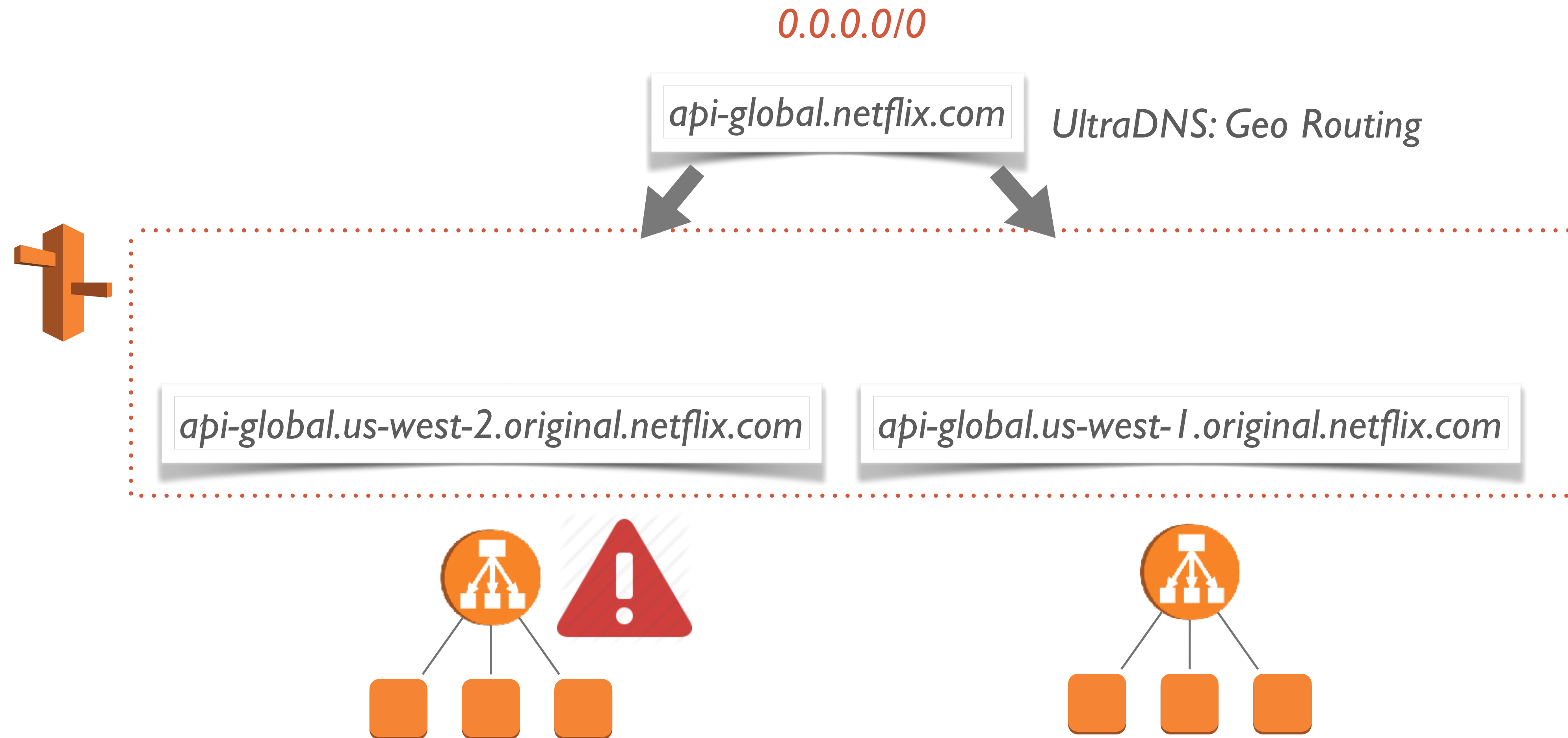


***Multi-Regional Resilient DNS for 100 million***

Source: <http://amzn.to/2iFvHA9>



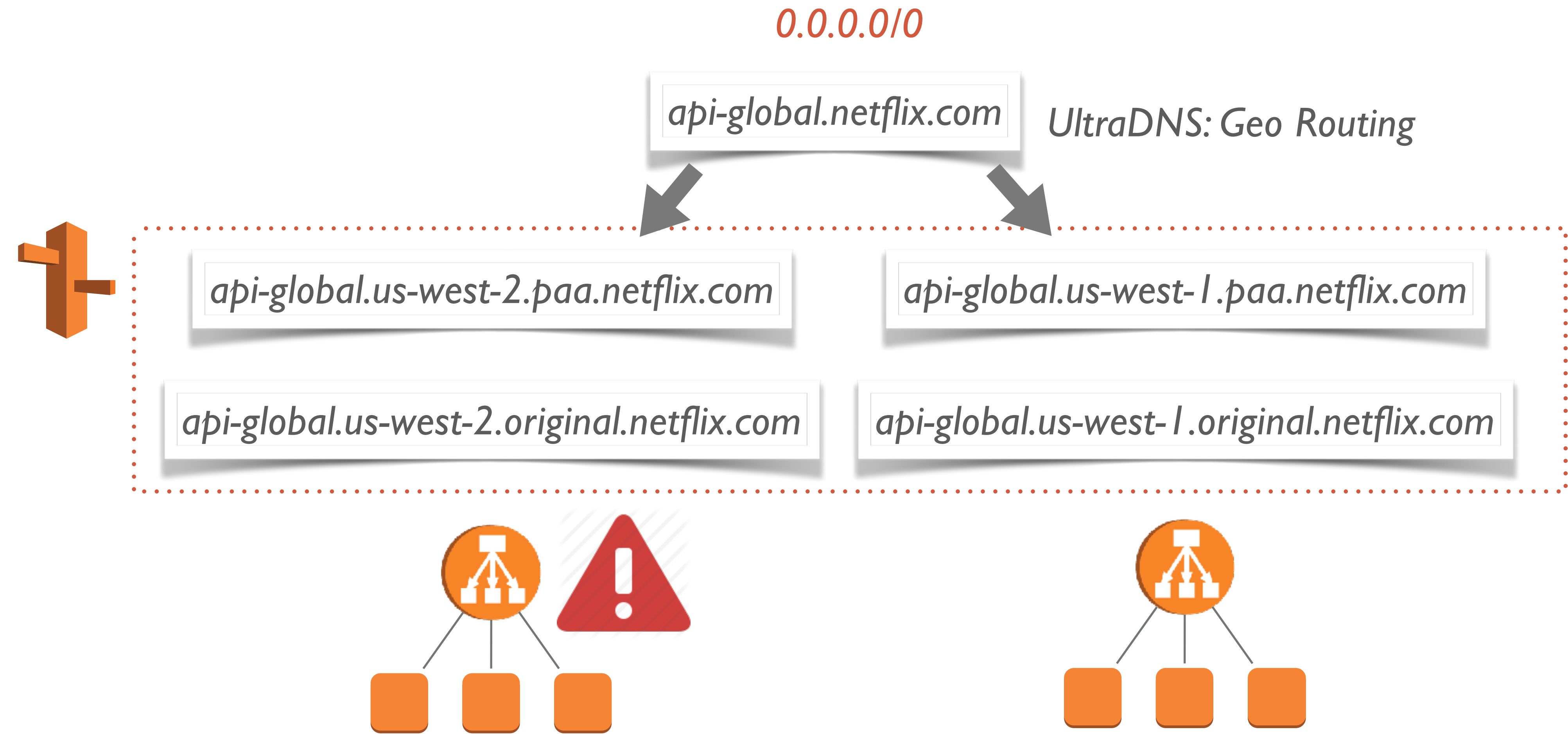
# NETFLIX DNS RESILIENCY



***Multi-Regional Resilient DNS for 100 million***

Source: <http://amzn.to/2iFvHA9>

# NETFLIX DNS RESILIENCY



**Multi-Regional Resilient DNS for 100 million**

Source: <http://amzn.to/2iFvHA9>



BITTIGER

# AWS CLI



# AWS ACCOUNT SETUP

---

- Bookmark following URLs
  - <http://www.ec2instances.info/>
  - <http://calculator.s3.amazonaws.com/index.html>
  - <http://s3speedtest.com>
- AWS Account Setup
  - Credit card information
  - Set billing alarm in CloudWatch
  - Inbound SSH rule for default VPC security group
  - Create main user and **save your key pair**
- Customize your console

# ENVIRONMENT SETUP

---

- Python > 2.7.9
  - Use brew or macports to install Python
- Git > 2.10
  - If you are using Mac, it should come with Xcode
- Unix/Linux Command Line Tools
  - <http://osxdaily.com/2014/02/12/install-command-line-tools-mac-os-x/>
- Checkout the repository: <https://bitbucket.org/bittiger-aws/aws>
  - Fork the repository
  - `$ make prepare`  
`$ make configure`

# AWS CLI CONFIGURATION

---

- Examples: scripts/configure.sh
- ~/.aws/credentials
- ~/.aws/config
- Profile
  - default
  - dev



BITTIGER



# AWS CLI EXAMPLES

---

- `aws COMMAND SUBCOMMAND [OPTIONS]`
- Github: <https://github.com/aws/aws-cli>
- Examples: `scripts/awsccli.sh`



BITTIGER

# AWS SHELL

- Auto completion
- Pop document
- Server side auto completion
- Execute shell command

```
aws> ec2 describe-instances --instance-ids
accept-reserved-instances-exchange-quote
accept-vpc-peering-connection
allocate-address
allocate-hosts
assign-ipv6-addresses
assign-private-ip-addresses
associate-address
associate-dhcp-options
associate-route-table
associate-subnet-cidr-block
associate-vpc-cidr-block
attach-classic-link-vpc
attach-internet-gateway
attach-network-interface
attach-volume
attach-vpn-gateway
```

Amazon Elastic Compute Cloud (Amazon EC2) provides resizable computing capacity in the Amazon Web Services (AWS) cloud. Using Amazon EC2 eliminates your need to invest in hardware up front, so you can develop and deploy applications faster.

AVAILABLE COMMANDS

```
* accept-reserved-instances-exchange-quote
* accept-vpc-peering-connection
* allocate-address
```

[F2] Fuzzy: ON [F3] Keys: Vi [F4] Single Column [F5] Help: ON [F10] Exit



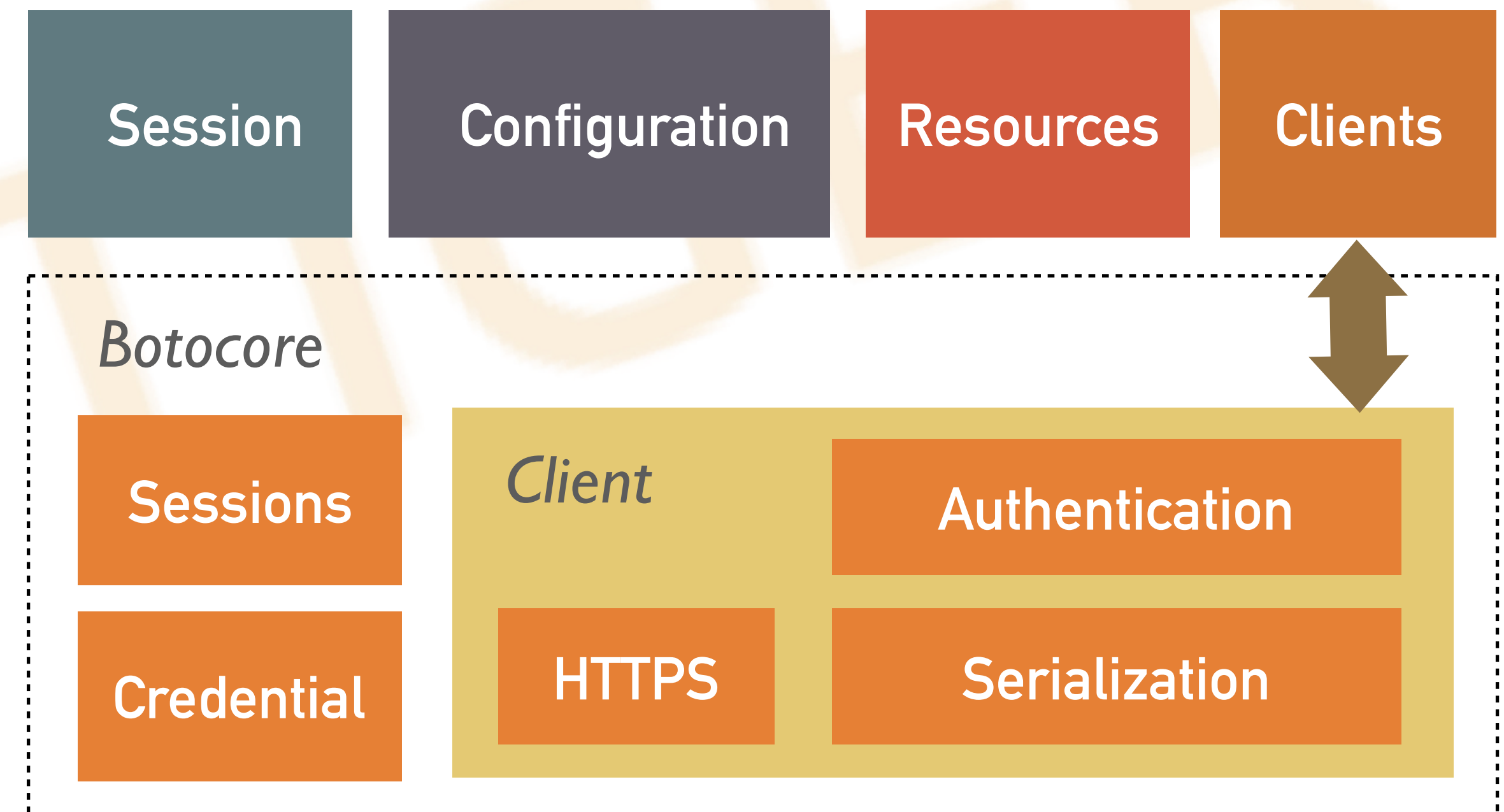
BIT TIGER

**BOT03**



# BOTO3

- AWS SDK for Python
- boto2
  - Early project
  - Community contributed code
- boto3
  - New version with consistent interfaces and up-to-date API support
- botocore
  - Low-level service and configurations
- <https://github.com/boto/boto3>
- <https://boto3.readthedocs.io/en/latest/>



# CODING EC2



# EC2 – EXAMPLES

---

- Launch EC2 instance by CLI and Boto3
- Evaluation
  - CPU
  - EBS Bandwidth
    - fio
  - Inter-Instance Bandwidth
    - iPerf
  - Enhanced networking SR-IOV
    - Check if SR-IOV
    - Bandwidth comparison





BITTIGER

# CODING S3

# S3 – EXAMPLES

---

- Create/empty/delete bucket
- List/upload/download/copy from/to bucket
- Selective copy
- Examples
  - scripts/s3.sh
  - scripts/s3cp.py
  - taxi/raw2s3.py





BITTIGER

PROJECT



# PROJECT NYC TAXI DATA

---

- Source: [http://www.nyc.gov/html/tlc/html/about/trip\\_record\\_data.shtml](http://www.nyc.gov/html/tlc/html/about/trip_record_data.shtml)
  - Stored in S3
    - [https://s3.amazonaws.com/nyc-tlc/trip+data/yellow\\_tripdata\\_2016-01.csv](https://s3.amazonaws.com/nyc-tlc/trip+data/yellow_tripdata_2016-01.csv)
- Format
  - vendor\_id, tpep\_pickup\_datetime, tpep\_dropoff\_datetime, passenger\_count, trip\_distance, pickup\_longitude, pickup\_latitude, rate\_code\_id, store\_and\_fwd\_flag, dropoff\_longitude, dropoff\_latitude, payment\_type, fare\_amount, extra, mta\_tax, tip\_amount, tolls\_amount, improvement\_surcharge, total\_amount
  - Not aligned...

# PROJECT – DATA PREPROCESSING

---

1. Clean: remove empty lines and invalid data
2. Select interested fields
  - pickup\_datetime, dropoff\_datetime
  - pickup\_longitude, pickup\_latitude, dropoff\_longitude, dropoff\_latitude
  - trip\_distance, total\_amount
3. Compact data (160 bytes/record vs. 80 bytes/record)
  - Change datetime to offset since 2009/1/1, e.g., 2009/1/1 12:01 => 60
  - Round coordinates to 6 digits
  - Round distance and fare to 2 digits
  - Padding to 80 bytes

# DATA PIPELINE



vendor\_id, tpep\_pickup\_datetime, tpep\_dropoff\_datetime, passenger\_count, trip\_distance, pickup\_longitude, pickup\_latitude, rate\_code\_id, store\_and\_fwd\_flag, dropoff\_longitude, dropoff\_latitude, payment\_type, fare\_amount, extra, mta\_tax, tip\_amount, tolls\_amount, improvement\_surcharge, total\_amount

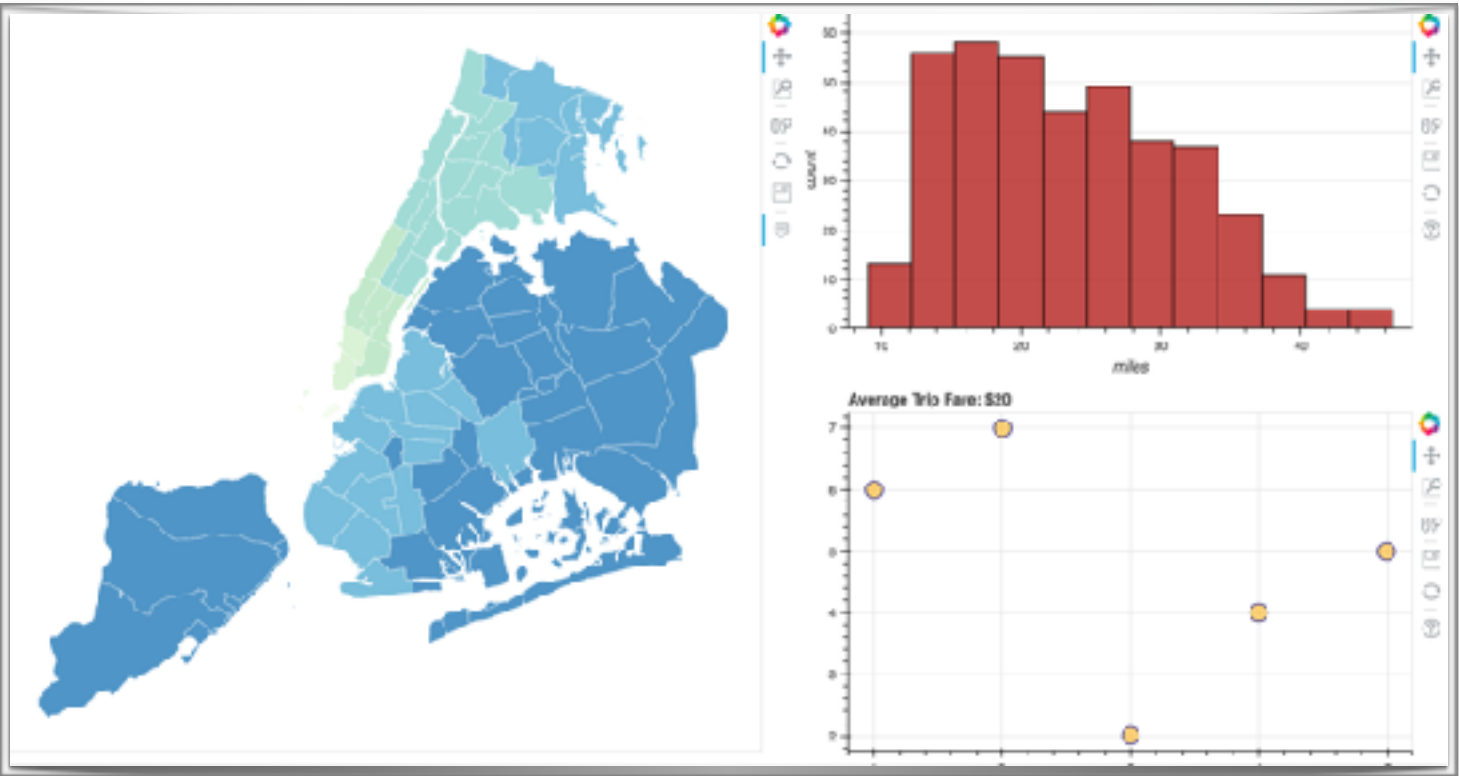


pickup\_datetime, dropoff\_datetime, pickup\_longitude, pickup\_latitude, dropoff\_longitude, dropoff\_latitude, trip\_distance, total\_amount

## Parallel Processing

pickup\_datetime, dropoff\_datetime, pickup\_district, dropoff\_district, trip\_distance, total\_amount

district, total\_pickup, total\_dropoff, total\_trip\_distance, total\_amount





# HOMEWORK



- Understand and explore EC2 hyper-threading (Pg. 9)
- Familiar with AWS console
  - Create, tagging, login, and terminate instance
  - Create bucket, upload and download a file, and delete object and bucket
- Measure enhanced networking by iPerf
- Measure EBS volume bandwidth: [http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/benchmark\\_procedures.html](http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/benchmark_procedures.html)

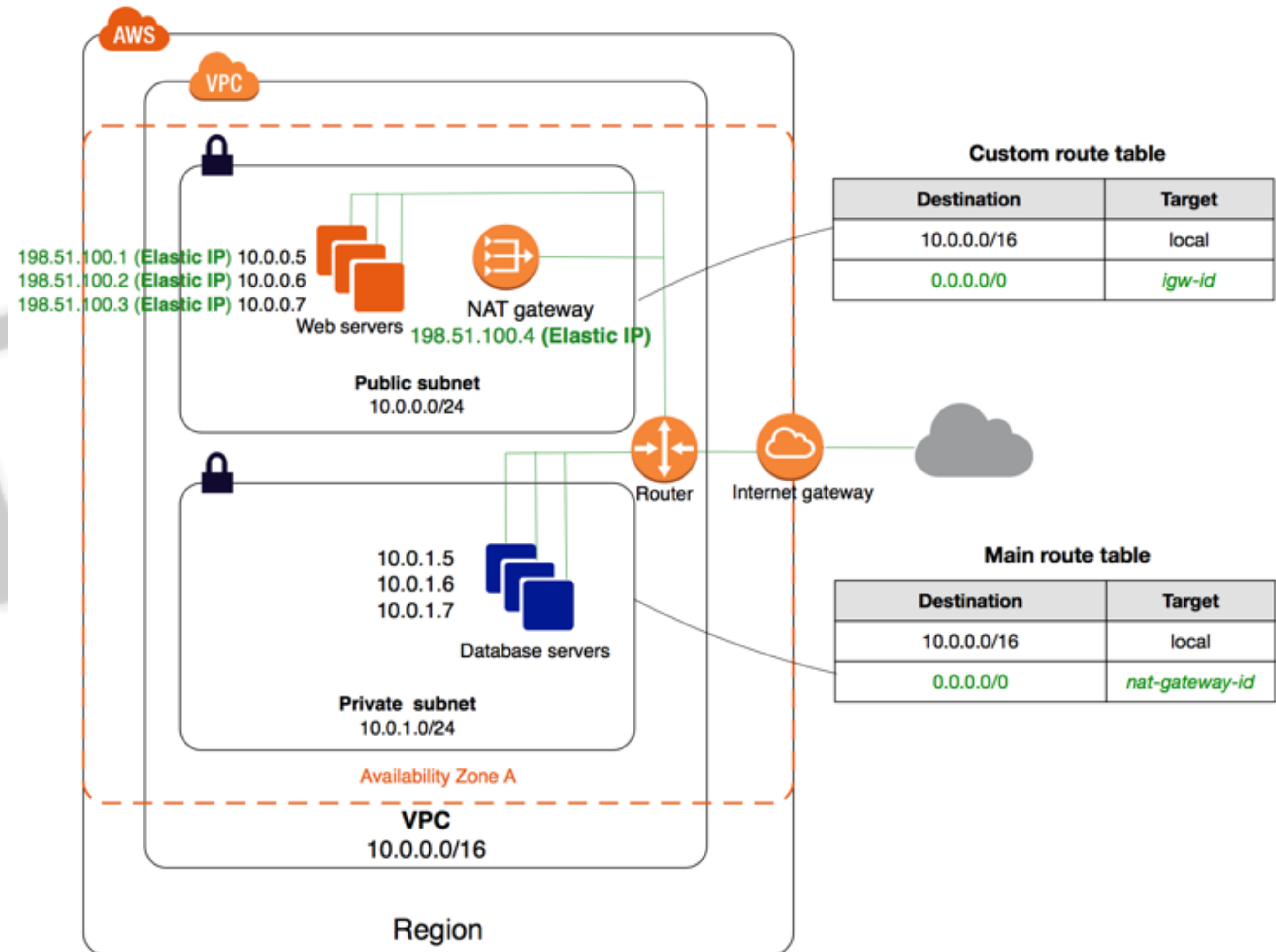
# HOMEWORK

---

- Create a VPC
- Create a public subnet
- Create a private subnet
- **Create a internet gateway and attach to VPC**
- Check ACL and security group
  - Add inbound SSH rule
- Make sure you can login to you instance in public subnet

# HOMEWORK

- Create a VPC with both public and private subnet
- Write a boto3 program to launch two instances in each subnet
- Verify the one in public subnet can be accessed from internet and another cannot
- Measure the performance of these two instances
- Use `taxi/raw2aws.py` to upload part of data to `s3://aws-nyc-taxi-data` and explain how you make the transfer efficient (parallel)





# QUESTIONS

---

- [bittiger-aws@googlegroups.com](mailto:bittiger-aws@googlegroups.com)



**BITTIGER**

Copyright 2017, Nan Dun, all rights reserved