TCSS 562: SOFTWARE ENGINEERING FOR CLOUD COMPUTING

Fundamental Cloud Architectures

Wes J. Lloyd
Institute of Technology
University of Washington - Tacoma

OBJECTIVES

- Term project questions
- AWS Educate
- Feedback from 4/23
- Tutorial #2
- Midterm Wednesday 5/9
- Fundamental cloud architectures (Ch. 11, Thomas Erl)

FEEDBACK – 4/23

- In paravirtualization, code needs to be modified in the OS. How big is the modification?
- Modifications are in the operating system kernel to trap system calls and device I/O to route to the XEN hypervisor for scheduling.
- Modifications would not be particularly large. Just to the kernel.
- Linux kernel 4.x binary size is ~6 to 7 MB

FEEDBACK - 2

- What does horizontal scaling mean?
- Do load balancer(s) integrate a firewall?
  - Load balancers must be configured to route traffic
  - E.g. specific ports will map to specific sets of servers
  - Load balancers listen on a variety of TCP/IP and UDP ports
  - Intercept requests sent to specific ports, route and handle traffic
  - ***Load balancer is not considered a firewall however***
- Installing the Amazon AWS CLI is unclear…

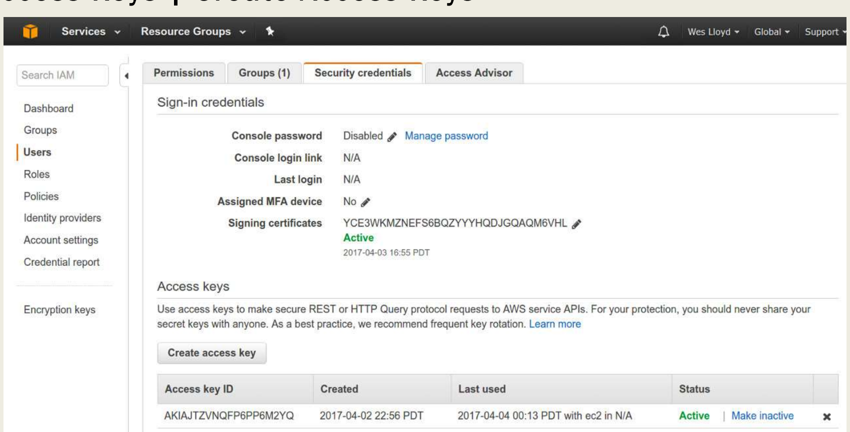
AWS CLI

- Launch Ubuntu 16.04 VM
  - **Instances | Launch Instance**
- Install the general AWS CLI
  - sudo apt install awscli
- Use "aws configure" command to configure
- *… or create a config file manually as follows:*

```
[default]
aws_access_key_id = <access key id>
aws_secret_access_key = <secret access key>
region = us-east-1
```


AWS CLI - 2

- Creating access keys: **IAM | Users | Security Credentials | Access Keys | Create Access Keys**



AWS CLI - 3

- Optionally export AWS_CONFIG_FILE variable to auto-load when logging in:
  - Add export statement to /home/ubuntu/.bashrc
  - May be required for legacy AWS CLI tools:

```
export AWS_CONFIG_FILE=$HOME/.aws/config
```

- Try some commands:
  - `aws help`
  - `aws command help`
  - `aws ec2 help`
  - `aws ec2 describes-instances --output text`
  - `aws ec2 describe-instances --output json`
  - `aws s3 ls`
  - `aws s3 ls vmscaleruw`

April 23, 2018	TCSS562: Software Engineering for Cloud Computing [Spring 2018] Institute of Technology, University of Washington - Tacoma	L7.7

AWS CLOUDWATCH CLI

- Find instance ID:
- curl http://169.254.169.254/latest/meta-data/instance-id

- Find available cloud watch metrics for instance:
- aws cloudwatch list-metrics --namespace AWS/EC2 --dimensions Name=InstanceId,Value=(instance ID)

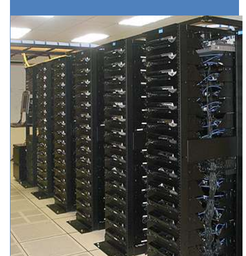
- Query metric values:
- aws cloudwatch get-metric-statistics --namespace AWS/EC2 --dimensions Name=InstanceId,Value=i-01fa0bb85354d0f74 --metric-name CPUUtilization --start-time 2018-04-25 --end-time 2018-04-26 --period 60 --statistics Average

April 25, 2018	TCSS562: Software Engineering for Cloud Computing [Spring 2018] Institute of Technology, University of Washington - Tacoma	L8.8

REQUESTED TOPICS

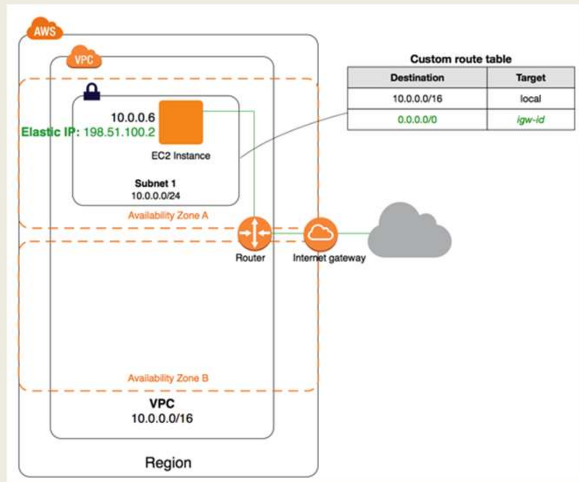
- AWS Lambda tutorial / demo
- Docker tutorial / demo
- Planned 4/30, 5/2, 5/7

TUTORIAL #2



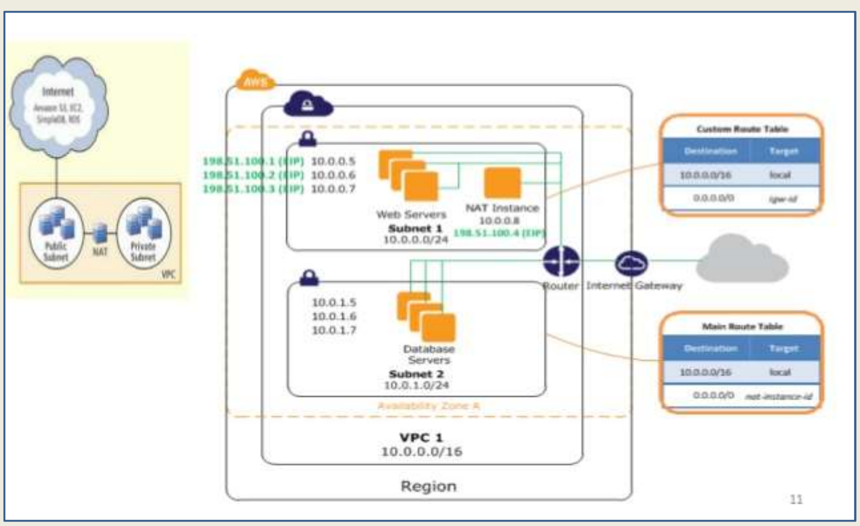
VPC: SINGLE SUBNET

- All instances require public IP address or elastic IP address for internet connectivity
- Easy to configure



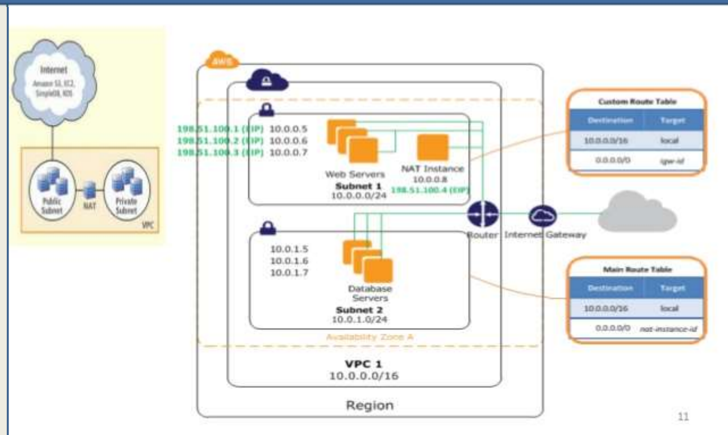
April 25, 2018	TCSS562: Software Engineering for Cloud Computing [Spring 2018] Institute of Technology, University of Washington - Tacoma	L8.11

VPC: PUBLIC / PRIVATE SUBNET



April 25, 2018	TCSS562: Software Engineering for Cloud Computing [Spring 2018] Institute of Technology, University of Washington - Tacoma	L8.12

- NAT Gateway provides internet connectivity for instances (VMs) without public addresses.
- At least once instance (VM) with a public or elastic IP provides an entry point to the VPC.



April 25, 2018	TCSS562: Software Engineering for Cloud Computing [Spring 2018] Institute of Technology, University of Washington - Tacoma	L8.13

SUBNET ADDRESSING

- Subnet rules
- Classless Inter-Domain Routing (CIDR) block:

CIDR Range	172.31.0.0/16
Netmask	255.255.0.0
Wildcard Bits	0.0.255.255
First IP	172.31.0.0
Last IP	172.31.255.255
Total Host	65536

- Try out a subnet calculator:
- http://www.calculator.net/ip-subnet-calculator.html

April 25, 2018	TCSS562: Software Engineering for Cloud Computing [Spring 2018] Institute of Technology, University of Washington - Tacoma	L8.14

FUNDAMENTAL CLOUD ARCHITECTURES

FUNDAMENTAL CLOUD ARCHITECTURES

- Common foundational cloud architectural models
- Exemplify common configurations of cloud-based application deployments
- Architectures describe cloud provisioning of: Compute, disk, and network resources

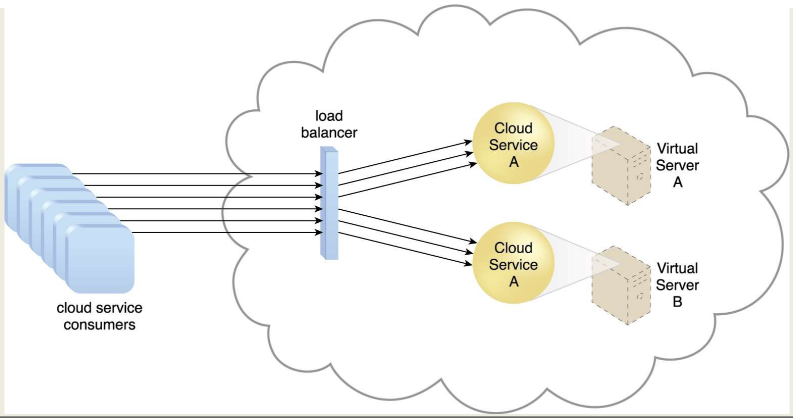
FUNDAMENTAL CLOUD ARCHITECTURES - 2

- **Workload distribution architecture**: load balancing
- **Resource pooling architecture**: resource pools
- **Dynamic scalability architecture**: auto-scaling
- **Elastic resource scalability architecture**: vertical scaling
- **Service load balancing architecture**: load balancing for cloud/web services
- **Cloud bursting architecture**: hybrid cloud
- **Elastic disk provisioning architecture**: thin vs. thick disk provisioning
- **Redundant storage architecture**: duplicate storage devices across data centers

WORKLOAD DISTRIBUTION ARCHITECTURE

- Horizontally scaled IT resources
- Add/remove resources per tier
- Load balancer distributes workload among providers
- Goal is to reduce IT resource:
  - Over-utilization
  - Under-utilization
- Sophisticated load balancing algorithms / run-time logic
  - Support resource management
  - Workload distribution

WORKLOAD DISTRIBUTION ARCHITECTURE - 2



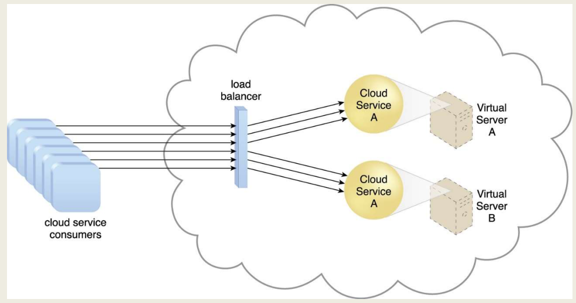
Redundant copies of the Cloud Service are implemented on both Virtual Servers. The load balancer intercepts service requests and directs them to either virtual server to ensure even workload distribution.

WORKLOAD DISTRIBUTION ARCHITECTURE - 3

- Can be applied to any IT resource
  - Virtual servers
  - Cloud storage devices
  - Cloud services
- Specializations of this architecture
  - Service load balancing (upcoming…)
  - Load balanced virtual server architecture
    balancing # of VMs per host…
  - Load balanced virtual switches architecture
    Increasing virtual network bandwidth w/ additional physical uplinks

WORKLOAD DISTRIBUTION ARCHITECTURE - 4

- Does this architecture encapsulate high availability?
  - Redundancy
  - Fault tolerant
  - Fail-over
- Is the load balancer fault tolerant?



- How could the load balancer be made fault tolerant?

HIGH AVAILABILITY LOAD BALANCING

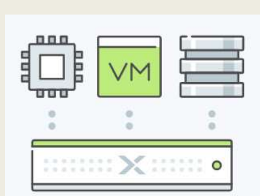
- Active / passive mode
  - Pair of load balancers are configured
  - Primary load balancer distributes traffic
  - Second load balancer operates in listening mode
  - Secondary load balancer step-ins in if primary fails
  - Achieves high availability
- Active / active mode
  - Two or more servers aggregate traffic load at the same time
  - User sessions are “locked” to one load balancer
  - Session is cached, requests are routed to same resource provider
  - If user request goes to other load balancer, it doesn’t know how to route request – would need to query other load balancer… ***slow!***
  - If one LB fails, is the other sufficient to route traffic?

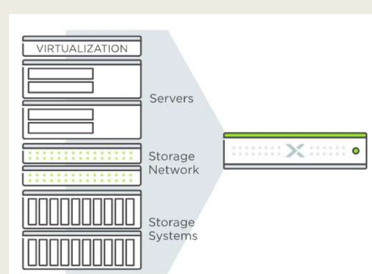
WORKLOAD DISTRIBUTION ARCHITECTURE - 5

- Other common elements of this architecture:
- **Audit monitor**: logs user requests as needed
- **Cloud usage monitor**: logs server utilization
- **Hypervisor**: virtual machines may need to be distributed
- **Logical network perimeter**: workloads distributed within
- **Resource cluster**: compute cluster resources to implement architecture
- **Resource replication**: concept of generating new resources in response to demand

RESOURCE POOLING ARCHITECTURE

- Identical IT resources are grouped and maintained
- System ensures they remained synchronized
- EXAMPLE: Hyper-converged server infrastructure
- Nutanix: https://www.nutanix.in/hyperconverged-infrastructure/



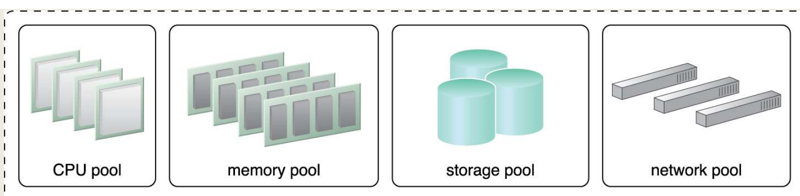


RESOURCE POOLING ARCHITECTURE - 2

- Resource Pools:
- Physical server pool / Virtual server pool
  - Preconfigured with OS/applications, ready for immediate use
- Storage pool
  - File-based, block-storage entities, with or without data, ready for use
- Network pool
  - Virtual firewall devices or network switches for redundant connectivity, load balancing, link aggregation
- CPU pool, Memory pool
  - Allocated to virtual servers

April 25, 2018	TCSS562: Software Engineering for Cloud Computing [Spring 2018] Institute of Technology, University of Washington - Tacoma	L8.25

SAMPLE RESOURCE POOL

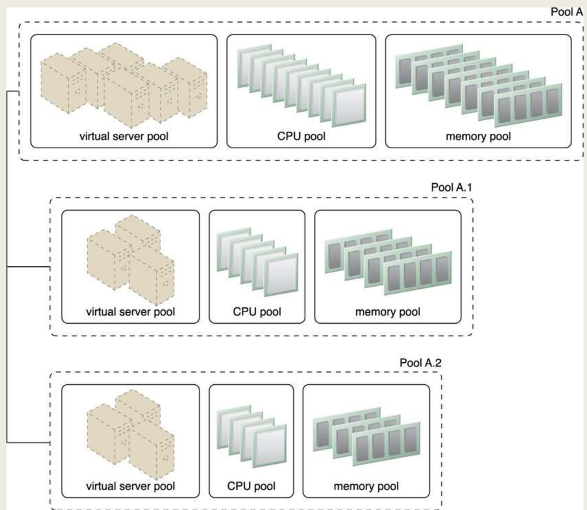


- Resources pools can be used to provide virtual devices
- Virtual server(s)
  - Consumes CPU and memory from pool
- Virtual disk(s)
  - Aggregate "just a bunch of disks" (JBoD) to provide disk(s) with required capacity, IOPS requirements, latency
- Virtual network
  - Aggregate physical network resources to provide virtual network devices which are isolated, with necessary bandwidth, and capacity

April 25, 2018	TCSS562: Software Engineering for Cloud Computing [Spring 2018] Institute of Technology, University of Washington - Tacoma	L8.26

RESOURCE POOLING ARCHITECTURE - 2

- Nested pools:
  Use same resources, but in different quantities.
- Allow rapid instantiation of resources with identical configurations



April 25, 2018	TCSS562: Software Engineering for Cloud Computing [Spring 2018] Institute of Technology, University of Washington - Tacoma	L8.27

RESOURCE POOLING MECHANISMS

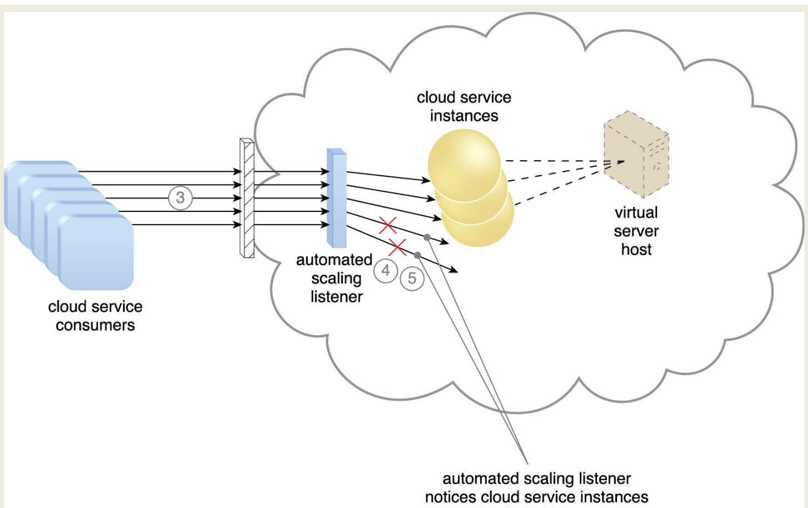
- **Audit monitor**: monitor usage to ensure legal use
- **Cloud usage monitor**: runtime tracking and synchronization to support management of resource pools
- **Pay-per-use monitor**: collects usage and billing information on how individual cloud users allocate and use resources
- **Remote administration system**: interfaces with backend systems to provide administration support
- **Resource management system**: supports administering resource pools
- **Hypervisor**, **Logical network perimeter**, **Resource replication**

April 25, 2018	TCSS562: Software Engineering for Cloud Computing [Spring 2018] Institute of Technology, University of Washington - Tacoma	L8.28

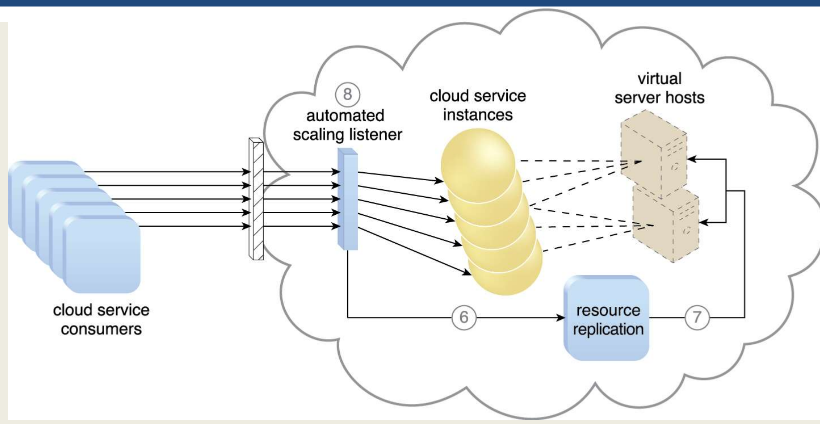
DYNAMIC SCALABILITY ARCHITECTURE

- Uses predefined scaling conditions to trigger "dynamic allocation" of IT resources from pools
- Resource allocation is adjusted dynamically based on demand
- Unnecessary resources are automatically
- Automated scaling listener
  - Monitors workload thresholds to determine when new resources should be added / removed using a scaling policy
  - Scaling policy – defines specifics of the scaling thresholds

DYNAMIC SCALABILITY ARCHITECTURE - 2



DYNAMIC SCALABILITY ARCHITECTURE - 3



Automatic scaling listener **triggers** creation of additional cloud service instances, which are added to pool for load balancing. *Automated scaling listener* resumes monitoring and adds and subtracts resources as required.

DYNAMIC SCALABILITY ARCHITECTURE - 4

- **Example**: AWS -Elastic Load Balancer (ELB)
  - **Classic load balancer**: application agnostic distribution of traffic across nodes
  - Uses cloud watch metrics …
  - **Application load balancer**: distributes traffic while considering unique content of requests enabling advanced routing capabilities
- ELB integrates with AWS auto scaling to dynamically provision +/- resources in response to demand
- Load balancer configuration automatically adjusted

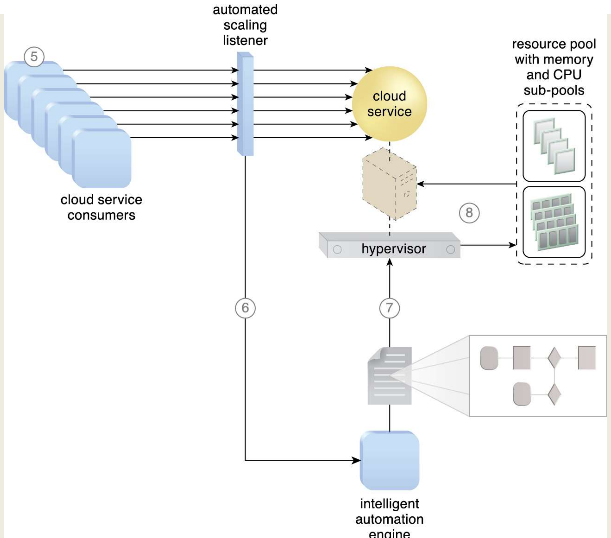
DYNAMIC SCALABILITY ARCHITECTURE QUESTIONS

- Why should load balancers / scaling listeners reroute subsequent requests for TCP sessions to the same server?
- How could "sticky" sessions impact load balancing?
- What are the advantages of classic (application agnostic) load balancing?
- For an "application load balancer" supporting "advanced routing", what features and capabilities are required of the load balancer?
- Which is more performant? Software or hardware load balancer?

ELASTIC '*RESOURCE CAPACITY*' ARCHITECTURE

- Supports dynamic provisioning of virtual servers
- Feature of public/private infrastructure-as-a-service (IaaS) clouds
- Enables reprovisioning CPUs and RAM *(*vertical scaling*)* to change the **SIZE** of a live virtual machine
  - Container platforms
- Ability to interact with the hypervisor and ***virtual infrastructure manager*** (VIM) to manage resources – **at runtime**
- Virtual server is monitored to increase capacity from a resource pool when thresholds are met.

ELASTIC RESOURCE CAPACITY ARCHITECTURE - 2



ELASTIC RESOURCE CAPACITY ARCHITECTURE - 3

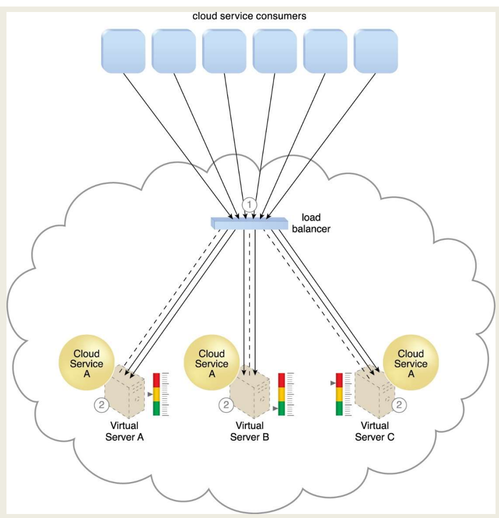
- Virtual servers may require rebooting for the changes in memory and CPU to take effect
- VIMs may automatically redistribute RAM & CPUs to VMs based on demand if rebooting is not required
- Not all Cloud VIMs or Container orchestration frameworks support/expose this feature
- Features are accessible at the hypervisor level
- Can resize # of CPUs and RAM of VMs on-the-fly by interacting directly with XEN/KVM hypervisors – via the CLI !
  - *Its preferable to recreating the VM entirely*

FUNDAMENTAL CLOUD ARCHITECTURES

- **Workload distribution architecture**: load balancing
- **Resource pooling architecture**: resource pools
- **Dynamic scalability architecture**: auto-scaling
- **Elastic resource scalability architecture**: vertical scaling
- **Service load balancing architecture**: load balancing for cloud/web services
- **Cloud bursting architecture**: hybrid cloud
- **Elastic disk provisioning architecture**: thin vs. thick disk provisioning
- **Redundant storage architecture**: duplicate storage devices across data centers

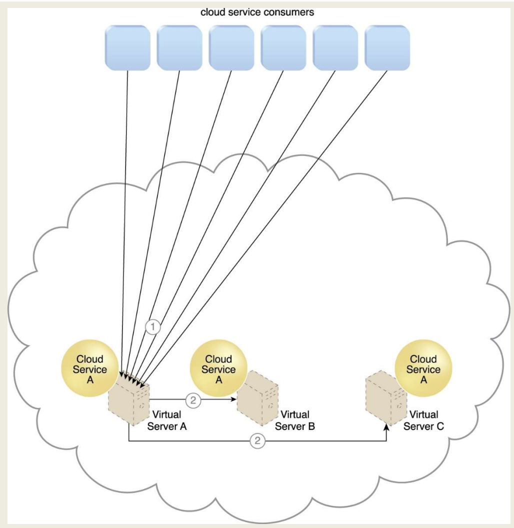
SERVICE LOAD BALANCING ARCHITECTURE

- A specialized variation of the workload distribution architecture
- Redundant deployments of cloud services are created, and load balancer distributes workloads
- The architecture we configure in tutorial #2 !
- Focuses on scaling cloud service implementations



REQUEST REDISTRIBUTION

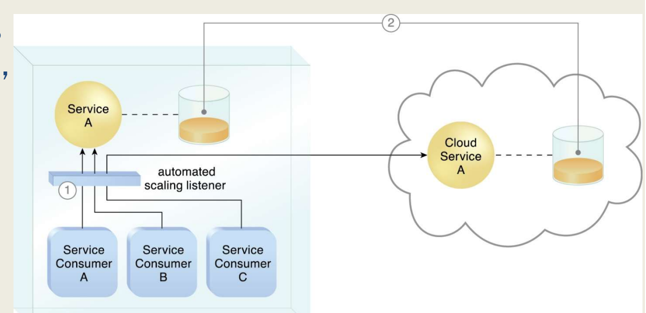
- Service redistributes request to the proper server
- "Shard" is a segment of a database hosted on a single server
- Sharding enables horizontal scaling of datasets by distributing rows across multiple servers
- Data fetch with sharding: Request is processed by application server to route request to server hosting the shard



April 25, 2018	TCSS562: Software Engineering for Cloud Computing [Spring 2018] Institute of Technology, University of Washington - Tacoma	L8.39

CLOUD BURSTING ARCHITECTURE

- Burst beyond on-premise IT resources to use public cloud when predefined capacity thresholds are surpassed
- Cloud resources are pre-deployed, but in ***inactive*** state until cloud bursting occurs
- Once cloud resources are no longer needed, they are released
- Automated scaling listener is used
- Latency to the cloud should be considered



April 25, 2018	TCSS562: Software Engineering for Cloud Computing [Spring 2018] Institute of Technology, University of Washington - Tacoma	L8.40

SCALING QUESTIONS

- When is vertical scaling preferable to horizontal scaling of cloud resources?
- Is cloud bursting vertical or horizontal scaling?
- Consider a private cloud with 5 host servers. What types of scaling is likely to be more important to the system administrator: horizontal or vertical scaling? Why?
- Can Docker container orchestration frameworks support horizontal scaling?
- Vertical scaling?

ELASTIC DISK PROVISIONING ARCHITECTURE

- Static allocations of fixed amounts of cloud disk space are expensive
- **Example:**
  Provision virtual Windows Server with 450GB disk
- Before OS is installed: 0 GB is used
- After OS is installed: <100 GB is used
- Customer is charged for: 450GB
- Elastic disk provisioning establishes a dynamic storage provisioning system to granularly bill a user for storage **actually used**...
- Based on “thin-provisioning” of storage

THIN VS. THICK PROVISIONING

- **Thin Provisioning**
- Only allocate storage space as it is used
- Increases potential for sharing the disk
- Introduces problem of *over-provisioning*: allocate more virtual disk space than actually exists

- **Thick Provisioning**
- Statically allocate all requested disk space
- A single user can provision the whole disk rendering it unusable by others !

THIN PROVISIONING - EXAMPLE

- Virtual box supports “thin provisioning” of virtual disks

- Disks have a maximize size, but only what is actually used is provisioned allowing the volume to grow.

- Eucalyptus EBS volume implementation
  - Disk volumes are thinly provisioned
  - Threat of over provisioning

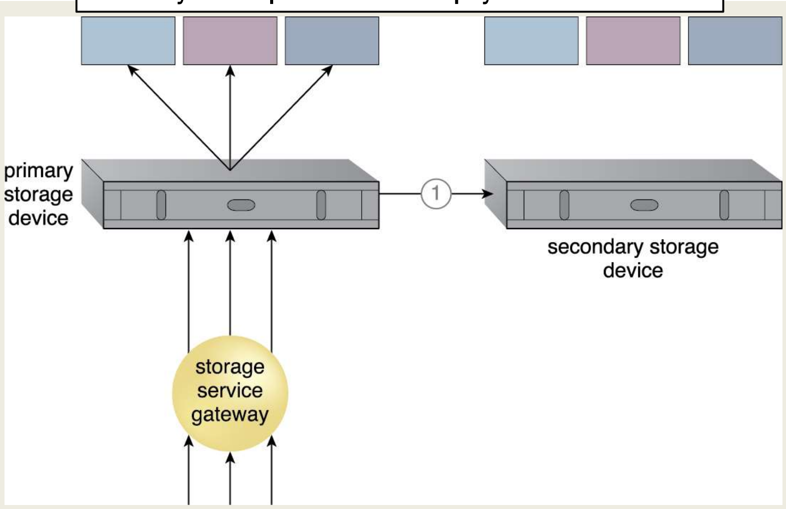
- Resizing volumes can be challenging

REDUNDANT STORAGE ARCHITECTURE

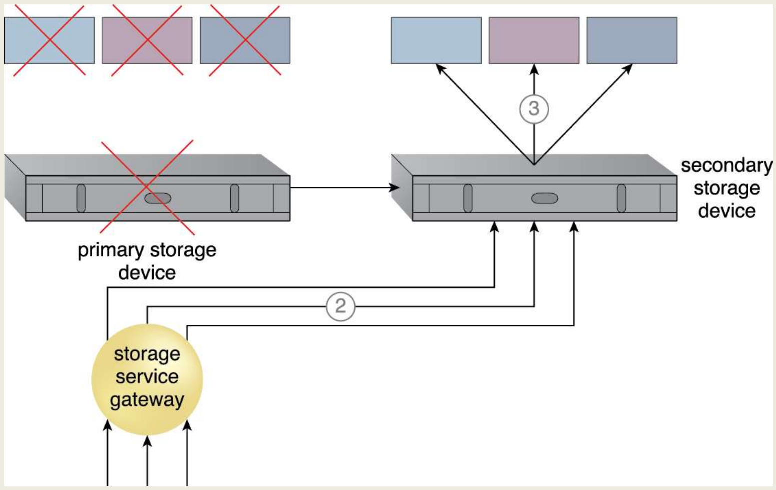
- Provide fault tolerance and improved availability of cloud storage devices
- Individual storage devices ***already*** have dual disk arrays and redundant disk controllers
- We are talking about SANs, NASs
- The idea is to replicate storage devices

REDUNDANT STORAGE ARCHITECTURE - 2

These colored blocks represent user disks. They are "Virtual" in the sense that the storage device abstracts how they are implemented with physical disks...



REDUNDANT STORAGE ARCHITECTURE - 3



REDUNDANT STORAGE ARCHITECTURE - 4

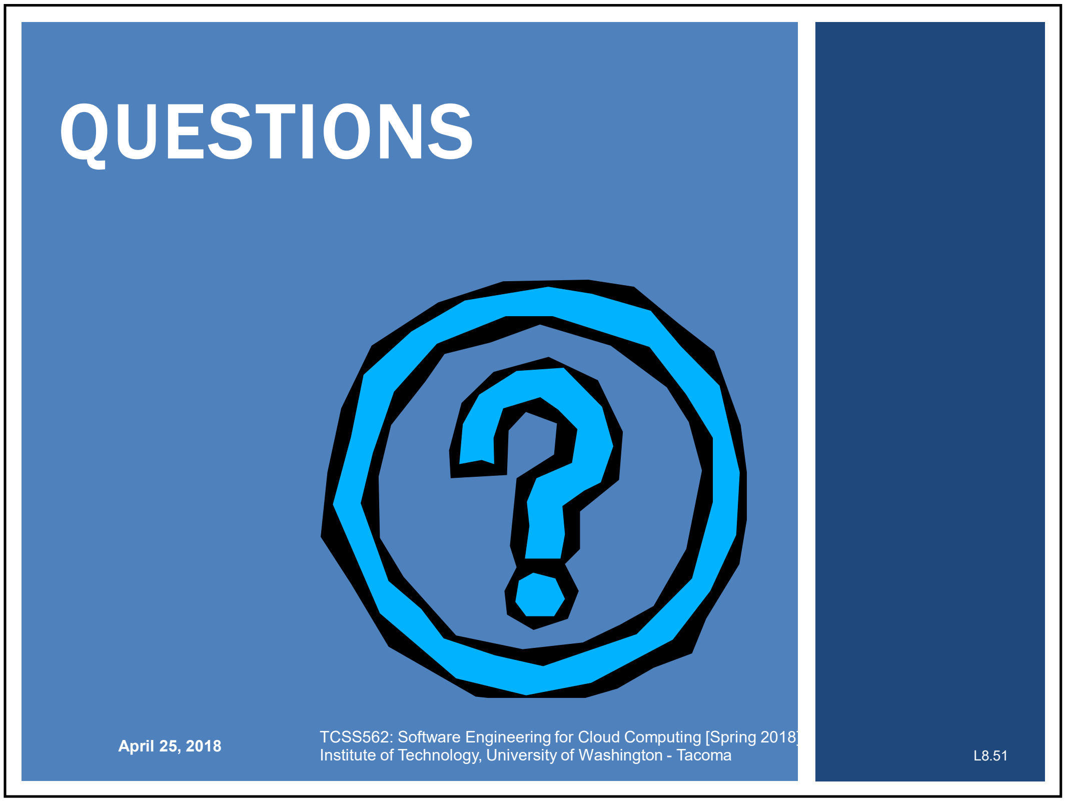
- Introduce a secondary duplicate cloud storage device that synchronizes data with the primary storage device
- Storage gateway service routes requests to second device when the primary device fails
- Secondary storage devices may be located in different physical locations

CLOUD STORAGE QUESTIONS

- If we have two identical storage devices that internally feature redundant disk arrays based on RAID 1, how many copies of the data exist?
- Besides disk space, what else does thin provisioning save?
- In addition to data redundancy, what else is gained from having multiple copies of our data?

FUNDAMENTAL CLOUD ARCHITECTURES SUMMARY

- **Workload distribution architecture**: load balancing
- **Resource pooling architecture**: resource pools
- **Dynamic scalability architecture**: auto-scaling
- **Elastic resource scalability architecture**: vertical scaling
- **Service load balancing architecture**: load balancing for cloud/web services
- **Cloud bursting architecture**: hybrid cloud
- **Elastic disk provisioning architecture**: thin vs. thick disk provisioning
- **Redundant storage architecture**: duplicate storage devices across data centers


QUESTIONS
April 25, 2018
TCSS562: Software Engineering for Cloud Computing [Spring 2018]
Institute of Technology, University of Washington - Tacoma
L8.51