TCSS 562: SOFTWARE ENGINEERING FOR CLOUD COMPUTING

April 23

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

OBJECTIVES

- Project Teams, Term project questions
- AWS Educate
- Feedback from 4/11
- Class presentations: Technology Sharing…
- Tutorial #1
- Tutorial #2 - Wednesday
- Review: AWS Demo
- Review: Cloud Enabling Technology (Ch. 5 Erl book)
- Fundamental cloud architectures (Ch. 11, Thomas Erl)

FEEDBACK – 4/11

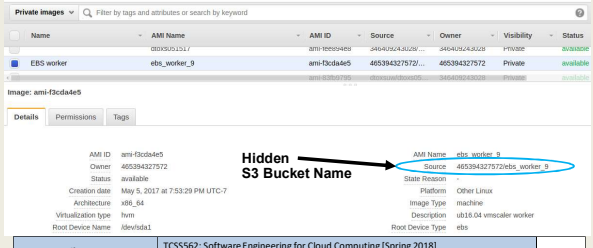
- How to verify that I successfully registered for the course through Amazon WS Educate
- I followed instructions but when I log into my AWS Educate account, I do not see access link to the course anywhere.

FEEDBACK - 2

- **Can you go over the process of "burning an image" again?**
- **Why are the snapshots stored as .img?**
  - AMIs are stored in raw format. "Burning an image" via the ec2_bundle_vol command compresses and encrypts the image files
- **Is there any other format it could be saved to, to utilize less space?**
  - Storing the image in RAW format decouples the compression mechanism from the image file format.
  - Decoupling allows ***any*** Linux-based compression tool to be used to compress/uncompress the image
  - Other virtualization image formats exist (for example: QCOW2 for KVM) that are sparse, where unused sections are not included
- Description of common cloud image formats:
- https://docs.openstack.org/image-guide/introduction.html

FEEDBACK - 3

- **If I take a snapshot and delete the EBS volume, what if one of the chunks are lost? Or gets corrupted?**
  - EBS snapshots are stored in S3 but not in a user-visible bucket.



FEEDBACK - 4

- **How do I create an EBS volume from the snapshot again?**
- From Volumes tab:
  - Create Volume button
  - Select a Snapshot ID
- From Snapshots tab:
  - Select a snapshot
  - Actions button
    - Create Volume

PROJECT PROPOSAL SUMMARY

- 10 Teams
  - Serverless Computing Services Composition– 2
    - Team 1, Team 3
  - PaaS Hosting Platform Comparison– 1
    - Team 2
  - Serverless Computing Platform Comparison– 2
    - Team 4, Team 5
  - In-memory key value services Comparison– 1
    - Team 6
  - NoSQL Database Services Comparison- 1
    - Team 7
  - Integration: Lambda+RDBMS- 1
    - Team 8
  - Open source serverless platform comparison- 2
    - Team 9, Team 10

PROJECT TECHNOLOGIES

- 18 Technologies
  - PAAS
    - Elastic Beanstalk, Google App Engine, Apache Tomcat
  - Serverless Platforms
    - AWS Lambda, Azure Functions, Google Cloud Functions, IBM Cloud Functions
  - In-memory key-value stores
    - Amazon ElastiCache, Azure Redis Cache, Redis
  - NoSQL Databases
    - DynamoDB, Azure Tables, Google Big Table, MongoDB
  - Relational Databases
    - Amazon Aurora, MySQL
  - Opensource Serverless Platforms
    - Oracle Fn, Apache OpenWhisk

PROJECT PROPOSALS

- **Team 1**
- Jason Eckstein (Team Leader), Timothy Yang, Arshdeep Singh
- Topic: Serverless Computing Services Composition (1)

- **Team 2**
- Ibrahim Diabate (Team Leader), Ming Hoi Lam, Manish KC, Swetha Reddy Nathala
- Topic: PaaS Hosting Platform Comparison

- **Team 3**
- Anisha Agarwal, Chhaya Choudhary, Sanchya Bhagat
- Topic: Serverless Computing Services Composition (2)

PROJECT PROPOSALS - 2

- **Team 4**
- Khushboo Baheti, Siri Sadashiva, Kiruthiga Gunasekaran, Suganya Jeyaraman (Team Leader)
- Topic: Serverless Computing Platform Evaluation (1)

- **Team 5**
- Yuxiao Guo, Ziyu Gao, Kaixuan Gao, Baojia Zhang
- Topic: Serverless Computing Platform Evaluation (2)

- **Team 6**
- Zhixiong Cai, Ningwei Chu, Edward Han, Xumeng Lyu
- Topic: Key value store services comparison (1)

PROJECT PROPOSAL - 3

- **Team 7**
- Priyanka Konduru, Resham Ahluwalia (Team Lead), Savita Rana, Sriharshitha Somaraju
- Topic: NoSQL database services Comparison (1)

- **Team 8**
- Raaghavi Sivaguru, Ramya Kumar (Team Lead), Sindhuja Chandran, Sujanasree Ratakonda
- Topic: Lambda + Relational Databases (MySQL, Amazon Aurora) (1)

PROJECT PROPOSAL - 4

- **Team 9**
- Bryan Sands
- Lan Ly
- Topic: Opensource Serverless Computing Platform Evaluation (1)

- **Team 10**
- Navid Heydari (Team Lead)
- Topic: Opensource Serverless Computing Platform Evaluation (2)

UPCOMING CONFERENCE OPPORTUNITIES

- **2018 IEEE CloudCom** – Cyprus
- Full papers: 8 pages: submission deadline ~June 15 (possibly will be extended to early-July)
- Short papers or poster: 4 pages, printed in proceedings submission deadline TBD
  - Short paper of good project(s) could be very achievable …
- **2018 ACM/IEEE Utility and Cloud Computing (UCC)** – Zurich, Switzerland
- Call for Papers - August 1st
- **2019 IEEE Cloud Engineering Conference (IC2E)** – Prague, CR
- Call for Papers – Late September

CLASS PRESENTATIONS

- Each team should make one presentation
- Teams 9 and 10 can combine to form one team
- Teams will choose to offer either:
  - **Technology Sharing Talk** – Limit 6 total for class
  - Week 7: May 7, May 9
  - Week 8: May 14
  - **Research Paper Presentation**
  - Week 9: May 23
  - Week 10: May 30
- Presentations must be unique – no duplicate topics
- Each group member participates in talk
- Submission of talk proposals will open on Wed April 25th
- Preferences are first come, first serve

TECHNOLOGY SHARING TALK

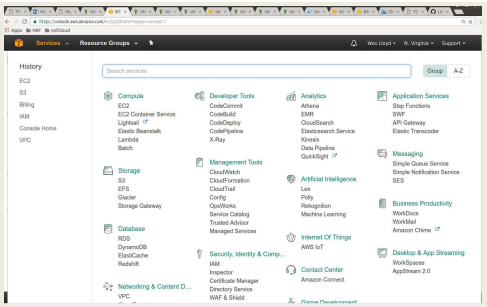
- Technology sharing talks must include at least 2 of 3
  - Demonstration: How to use cloud service using GUI
  - Demonstration: How to use cloud service using CLI
  - Demonstration: How to use cloud service using programming API
- Structure:
  - Slide presentation – technology overview 15-20 slides (20 minutes)
  - Demonstration – (10 minutes)
  - Question and Answer with class

RESEARCH PAPER TALK

- Groups present review and critique of a high quality research publication related to TCSS 562 group project
- Groups work with professor to identify and select paper based on the project
- Structure:
  - Overview of the paper
  - Summary of primary research contributions
  - Overview of related work
  - Presentation of the paper's findings: *What technology is proposed? What is the evaluation approach? What are the results?*
  - Critique of the paper: *Identify strengths and weaknesses*
  - Question and answer with class

AWS DEMO

AWS MANAGEMENT CONSOLE



AWS EC2

- Elastic Compute Cloud
- Instance types
  - On demand instance – full price
  - Reserved instance – contract based
  - Spot instance – auction based, terminates with 2 minute warning
  - Dedicated/resreved host – reserved HW
  - Reserved host
  - Instance families:
    General, compute-optimized, memory-optimized, GPU, etc.
- Storage types
  - Instance storage - ephemeral storage
  - Elastic block store
  - Elastic file system

INSTANCE STORAGE

- Also called ephemeral storage
- Persisted using images saved to S3 (simple storage service)
  - ~2.3¢ per GB/month on S3
  - 5GB of free tier storage space on S3
- Requires "burning" an image
- Mutli-step process:
  - Create image files
  - Upload chunks to S3
  - Register image
- Launching a VM
  - Requires downloading image components from S3, reassembling them… is potentially slow
- VMs with instance store backed root volumes not pause-able
- Historically root volume limited to 10-GB max– ***faster imaging…***

ELASTIC BLOCK STORE

- EBS cost model is different than instance storage (uses S3)
  - ~10¢ per GB/month
  - 30GB of free tier storage space
- EBS provides "live" mountable volumes
  - Listed under volumes
  - **Data volumes**: can be mounted/unmounted to any VM, dynamically at any time
  - **Root volumes**: hosts OS files and acts as a boot device for VM
  - In Linux drives are linked to a mount point "directory"
- Snapshots back up EBS volume data to S3
  - Enables replication (required for horizontal scaling)
  - EBS volumes not actively used should be snapshotted, and deleted to save EBS costs…

EBS VOLUME TYPES - 2

- Metric: I/O Operations per Second (IOPS)
- General Purpose 2 (GP2)
  - 3 IOPS per GB, Max 10,000 IOPS, 160MB/sec per volume
- Provisioned IOPS (IO1)
  - 32,000 IOPS, and 500 MB/sec throughput per volume
- Throughput Optimized HDD (ST1)
  - Up to 500 MB/sec throughput
  - 4.5 ¢ per GB/month
- Cold HDD (SC1)
  - Up to 250 MB/sec throughput
  - 2.5 ¢ per GB/month
- Magnetic
  - Up to 800 MB/sec throughput
  - 5 ¢ per GB/month

ELASTIC FILE SYSTEM

- Network file system (NFSv4 protocol) for EC2 instances
- Hosted by EC2 instances
- ~ 30 ¢ per GB/month
- Enables mounting (sharing) the same disk "volume" for R/W access across multiple instances at the same time
- Performance scales based on size of deployment

EFS PERFORMANCE

File System Size (GiB)	Baseline Aggregate Throughput (MiB/s)	Burst Aggregate Throughput (MiB/s)	Maximum Burst Duration (Min/Day)	% of Time File System Can Burst (Per Day)
10	0.5	100	7.2	0.5%
256	12.5	100	180	12.5%
512	25.0	100	360	25.0%
1024	50.0	100	720	50.0%
1536	75.0	150	720	50.0%
2048	100.0	200	720	50.0%
3072	150.0	300	720	50.0%
4096	200.0	400	720	50.0%

- From: Hornacek, M., et al., Geospatial Analytics in the Large for Monitoring Depth of Cover for Buried Pipeline Infrastructure, IC2E 2018.

AMAZON MACHINE IMAGES

- AMIs
- Unique for the operating system (root device image)
- Two types
  - Instance store
  - Elastic block store (EBS)
- Deleting requires multiple steps
  - Deregister AMI
  - Delete associated data - (*files in S3*)
- Forgetting both steps leads to costly "orphaned" data
  - No way to instantiate a VM from deregistered AMIs
  - Data still in S3 resulting in charges

EC2 VIRTUALIZATION - PARAVIRTUAL

- **1st, 2nd, 3rd, 4th generation → XEN-based**
- **5th generation instances → KVM (full virtualization)**

- XEN - two virtualization modes
- XEN Paravirtualization "paravirtual"
  - 2008-2012: required because of poor performance of HVM mode
  - I/O performed in kernel mode for better performance
  - Requires special OS paravirtual kernel
  - Notice use of common **AKI** files on AWS – ***Amazon kernel Image(s)***

EC2 VIRTUALIZATION - HVM

- XEN HVM mode
  - Full virtualization – no special OS kernel required
  - Computer entirely simulated
  - MS Windows runs in "hvm" mode
  - Allows work around: 10GB instance store root volume limit
  - Kernel is on the root volume
  - No AKIs (kernel images)
  - Commonly used today (*EBS-backed instances*)

INSTANCE ACTIONS

- Stop
  - Costs of "pausing" an instance
- Terminate
- Reboot

- Image management
- Creating an image
  - EBS (snapshot)
- Bundle image
  - Instance-store

BURNING AN IMAGE (AMI)

- Paravirtual / Instance Store backed root volume
  - CLI only
  - Images saved to user defined S3 buckets
  - User must manage deletion, etc.
  - CLI APIs: ec2_bundle_vol, ec2_upload_bundle, ec2_register
- HVM / Instance Store backed root volume
  - GUI and CLI
  - Images saved on hidden S3 buckets
- HVM / EBS-backed root volume
  - GUI and CLI
  - Images saved on hidden S3 buckets

EC2 INSTANCE: NETWORK ACCESS

- Public IP address
- Elastic IPs
  - Costs: in-use FREE, not in-use ~12 ¢/day
  - Not in-use (e.g. "paused" EBS-backed instances)

- Security groups
  - E.g. firewall

- Identity access management (IAM)
  - AWS accounts, groups

- VPC / Subnet / Internet Gateway / Router
- NAT-Gateway

SIMPLE STORAGE SERVICE (S3)

- Key-value blob storage
- What is the difference vs. key-value stores (NoSQL DB)?
- Can mount an S3 bucket as a volume in Linux
  - Supports common file-system operations
- Eventual consistency

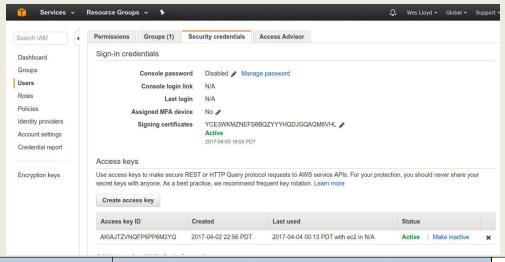
AWS CLI

- 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`

ALTERNATIVE CLI

- `sudo apt install ec2-api-tools`
- Provides more concise output
- Additional functionality
- Define variables in .bashrc or another sourced script:
- `export AWS_ACCESS_KEY={your access key}`
- `export AWS_SECRET_KEY={your secret key}`
- `ec2-describe-instances`
- `ec2-run-instances`
- `ec2-request-spot-instances`
- EC2 management libraries for Java:
- `http://docs.aws.amazon.com/AWSJavaSDK/latest/javadoc/index.html`

INSPECTING INSTANCE INFORMATION

- Explore instance metadata using http GET uri
- Each instance locally responds to these requests
- Example: find your instance ID:

```
curl http://169.254.169.254/
curl http://169.254.169.254/latest/
curl http://169.254.169.254/latest/meta-data/
curl http://169.254.169.254/latest/meta-data/instance-id
; echo
```

- ec2-get-info command (??)
  - Python ec2 command to query meta-data
  - Same as
  - What is the "aws" CLI equivalent?

PRIVATE KEY AND CERTIFICATE FILE

- Some EC2 APIs require additional authentication
  - Private key and certificate file
- Install openssl package on VM

```
# generate private key file
$openssl genrsa 2048 > mykey.pk

# generate signing certificate file
$openssl req -new -x509 -nodes -sha256 -days 36500 -key
mykey.pk -outform PEM -out signing.cert
```

- Add signing.cert to IAM | Users | Security Credentials | *- - new signing certificate - -*
- From: http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/set-up-ami-tools.html?icmpid=docs_iam_console#ami-tools-create-certificateE

PRIVATE KEY, CERTIFICATE FILE

- These files, combined with your AWS_ACCESS_KEY and AWS_SECRET_KEY and AWS_ACCOUNT_ID enable you to publish new images from the CLI
- Objective:
  1. Configure VM with software stack
  2. Burn new image for VM replication **(horizontal scaling)**
- Some folks may just install Docker. . .
- Create image script . . .

AMI TOOLS

- AMI Tools API:
- https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ami-tools-commands.html
- **ec2-bundle-vol**
- Creates instance store-backed Linux AMI by ***compressing***, ***encrypting***, and ***signing*** copy of live/running root device volume of an instance

CREATE A NEW INSTANCE STORE IMAGE SCRIPT

```
image=$1
echo "Burn image $image"
echo "$image" > image.id
mkdir /mnt/tmp
AWS_KEY_DIR=/home/ubuntu/.aws
export EC2_URL=http://ec2.amazonaws.com
export S3_URL=https://s3.amazonaws.com
export EC2_PRIVATE_KEY=${AWS_KEY_DIR}/mykey.pk
export EC2_CERT=${AWS_KEY_DIR}/signing.cert
export AWS_USER_ID={your account id}
export AWS_ACCESS_KEY={your aws access key}
export AWS_SECRET_KEY={your aws secret key}
ec2-bundle-vol -s 5000 -u ${AWS_USER_ID} -c ${EC2_CERT} -k ${EC2_PRIVATE_KEY}
--ec2cert /etc/ec2/amitools/cert-ec2.pem --no-inherit -r x86_64 -p $image -i
/etc/ec2/amitools/cert-ec2.pem
cd /tmp
ec2-upload-bundle -b tcss562 -m $image.manifest.xml -a ${AWS_ACCESS_KEY} -s
${AWS_SECRET_KEY}  --url http://s3.amazonaws.com --location US
ec2-register tcss562/$image.manifest.xml --region us-east-1 --kernel aki-
88aa75e1
```


FUNDAMENTAL CLOUD ARCHITECTURES

April 23, 2018

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

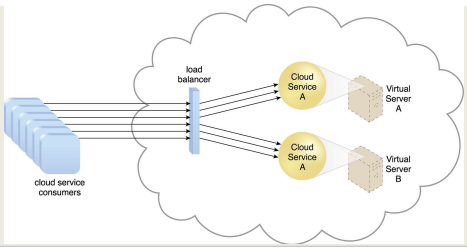
FUNDAMENTAL CLOUD ARCHITECTURES - 2

- **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



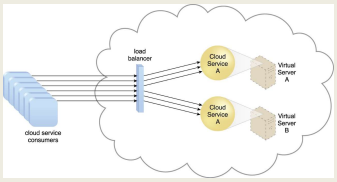
Redundant copies of the Cloud Service are implemented on both Virtual Servers. The load balancer intercepts service requests and directs them to either virtual server to ensure even workload distribution.

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

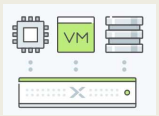
- Active / passive mode
  - Pair of load balancers are configured
  - Primary load balancer distributes traffic
  - Second load balancer operates in listening mode
  - Secondary load balancer step-ins in if primary fails
  - Achieves high availability
- Active / active mode
  - Two or more servers aggregate 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?

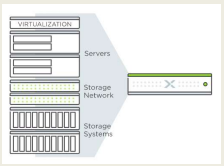
WORKLOAD DISTRIBUTION ARCHITECTURE - 5

- 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/

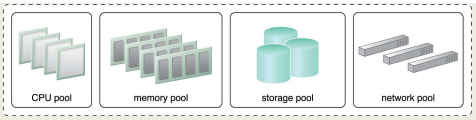




RESOURCE POOLING ARCHITECTURE - 2

- 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

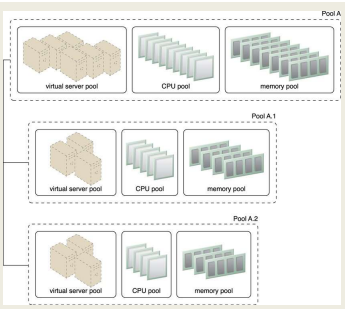
SAMPLE RESOURCE POOL



- 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

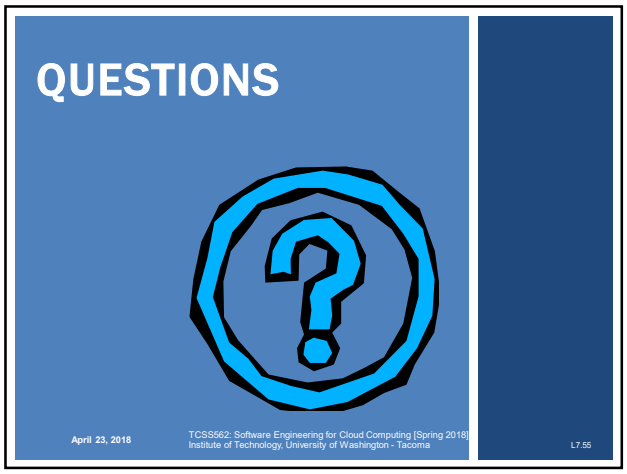
RESOURCE POOLING ARCHITECTURE - 2

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



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**


QUESTIONS
April 23, 2018
TCSS562: Software Engineering for Cloud Computing [Spring 2018]
Institute of Technology, University of Washington - Tacoma
L7.55