# TCSS 562: SOFTWARE ENGINEERING FOR CLOUD COMPUTING

## AWS Overview and Demo, Cloud Enabling Technology

Wes J. Lloyd
School of Engineering and Technology
University of Washington – Tacoma

1

## OFFICE HOURS – FALL 2024

- **Tuesdays:**
  - 2:30 to 3:30 pm - CP 229
- **Fridays**
  - 1:00 pm to 2:00 pm – ONLINE via Zoom
- Or email for appointment

*> Office Hours set based on Student Demographics survey feedback*



2

## OBJECTIVES – 10/29

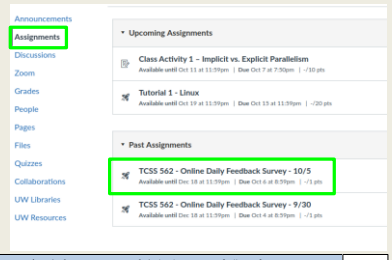
- **Questions from 10/24**
- Tutorials Questions
- Tutorial 6 – Serverless Databases
- Class Activity 2 - Horizontal Scaling in the Cloud
- AWS Overview and demo
- Tutorial 4 Demo
- Ch. 5: Cloud Enabling Technology



3

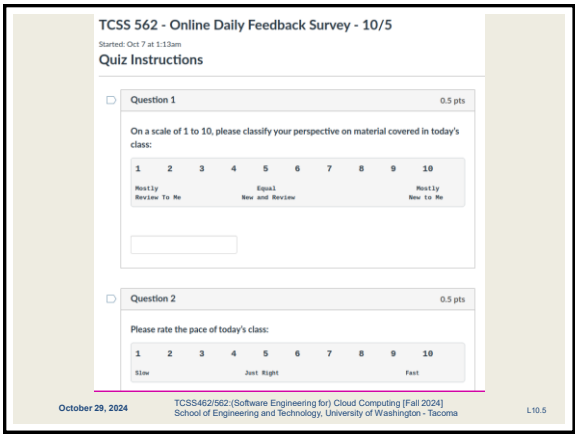
## ONLINE DAILY FEEDBACK SURVEY

- Daily Feedback Quiz in Canvas – Take After Each Class
- Extra Credit for completing






4




5

## MATERIAL / PACE

- Please classify your perspective on material covered in today's class (**44** respondents):
- 1-mostly review, 5-equal new/review, 10-mostly new
- **Average – 6.11 (↑ - *previous 6.05*)**

- Please rate the pace of today's class:
- 1-slow, 5-just right, 10-fast
- **Average – 5.41 (↓ - previous 5.74)**

- **Response rates:**
- TCSS 462: 29/42 – 69.05%
- TCSS 562: 15/20 – 75.0%



6

## FEEDBACK FROM 10/24

- ***If we use CaaS, is there a difference in how hot starts are handled compared to FaaS?***
- In AWS Fargate CaaS, you create a task.
- A task either:
  - Runs a one-time script which has a finite runtime, and is always cold
  - Hosts an always-up service endpoint accessible via TCP port(s)
- A service task is always "hot" once the service is up.
- ***Are there any best practices/tools to guarantee we're only testing hot starts and avoiding cold start scenarios?***
  - Yes, SAAF, the Serverless Application Analytics Framework, is the tool introduced in Tutorial 4 to do this.
  - Reports can be generated, and performance data is aggregated based on function state: cold (newcontainer=1) vs. warm (newcontainer=0)



7

## FEEDBACK - 2

- ***What's the best place to host a website? Would it be better than having github host a webpage, for example? Is hosting a website also what AWS is often used for, or is AWS more for situations when you need more compute power?***
- ***If I deploy microservices, or any code on AWS as part of a business application, what is it's lifetime? That is, if I keep paying my AWS bill, can that application run for 30 years without me maintaining it?***



8

## SAMPLE QUESTION 1

- CPU hyperthreads are equivalent to physical CPUs cores and provide identical performance and throughput.

A. True
B. False



9

## SAMPLE QUESTION 2

- Which taxonomy classification is most often associated with Graphical Processing Units (GPUs)?

A. SIMD
B. MIMD
C. SISD
D. MISD



10

## SAMPLE QUESTION 3

- In Tutorial 4, after creating a new AWS Lambda function in the AWS Management Console, and uploading the Java jar file, what must be done before attempting to call the function using the AWS CLI?

A. Permission must be granted to write to the CloudWatch logs
B. The function memory size and timeout needs to be adjusted
C. The function handler must be updated
D. A REST 'POST' endpoint must be created using the Amazon API Gateway
E. All of the above



11

## TERM PROJECT PROPOSALS

- 18 Total term project proposals received
- *** 2 students appear to not be in any project group*
- Status to be provided:
  - proposals accepted, or
  - revisions requested
- Application Use Cases (summary to be provided):
  - TLQ pipelines
  - image processing pipelines
  - NLP pipeline
  - Other



12

## AWS CLOUD CREDITS UPDATE

- AWS CLOUD CREDITS ARE NOW AVAILABLE FOR TCSS 462/562
- Credit codes must be securely exchanged
- Request codes by sending an email with the subject "**AWS CREDIT REQUEST**" to **wlloyd@uw.edu**
- Please provide email address and AWS account ID for tracking purposes
- Codes can also be obtained in person (or zoom), in the class, during the breaks, after class, during office hours, by appt
  - 50 credit requests fulfilled as of Oct 28 @ 11:59p
- Codes not provided using discord



13

## OBJECTIVES – 10/29

- Questions from 10/24
- **Tutorials Questions**
- Tutorial 6 - Serverless Databases
- Class Activity 2 - Horizontal Scaling in the Cloud
- AWS Overview and demo
- Tutorial 4 Demo
- Ch. 5: Cloud Enabling Technology



14

## TUTORIAL 0

- Getting Started with AWS
- https://faculty.washington.edu/wlloyd/courses/tcss562/tutorials/TCSS462_562_f2024_tutorial_0.pdf
- Create an AWS account
- Create account credentials for working with the CLI
- Install awsconfig package
- Setup awsconfig for working with the AWS CLI



15

## TUTORIAL 3 – DUE OCT 31

- Best Practices for Working with Virtual Machines on Amazon EC2
- https://faculty.washington.edu/wlloyd/courses/tcss562/tutorials/TCSS462_562_f2024_tutorial_3.pdf
- Creating a spot VM
- Creating an image from a running VM
- Persistent spot request
- Stopping (pausing) VMs
- EBS volume types
- Ephemeral disks (local disks)
- Mounting and formatting a disk
- Disk performance testing with Bonnie++
- Cost Saving Best Practices



16

## TUTORIAL 4 – DUE NOV 5

- Introduction to AWS Lambda with the Serverless Application Analytics Framework (SAAF)
- https://faculty.washington.edu/wlloyd/courses/tcss562/tutorials/TCSS462_562_f2024_tutorial_4.pdf
- Obtaining a Java development environment
- Introduction to Maven build files for Java
- Create and Deploy "hello" Java AWS Lambda Function
  - Creation of API Gateway REST endpoint
- Sequential testing of "hello" AWS Lambda Function
  - API Gateway endpoint
  - AWS CLI Function invocation
- Observing SAAF profiling output
- Parallel testing of "hello" AWS Lambda Function with faas_runner
- Performance analysis using faas_runner reports
- Two function pipeline development task



17

## TUTORIAL 5 – DUE NOV 14

- Introduction to Lambda II: Working with Files in S3 and CloudWatch Events
- https://faculty.washington.edu/wlloyd/courses/tcss562/tutorials/TCSS462_562_f2024_tutorial_5.pdf
- Customize the Request object (add getters/setters)
  - Why do this instead of HashMap ?
- Import dependencies (jar files) into project for AWS S3
- Create an S3 Bucket
- Give your Lambda function(s) permission to work with S3
- Write to the CloudWatch logs
- Use of CloudTrail to generate S3 events
- Creating CloudWatch rule to capture events from CloudTrail
- Have the CloudWatch rule trigger a target Lambda function with a static JSON input object (hard-coded filename)
- **Optional**: for the S3 PutObject event, dynamically extract the name of the file put to the S3 bucket for processing



18

## OBJECTIVES – 10/29

- Questions from 10/24
- Tutorials Questions
- **Tutorial 6 - Serverless Databases**
- Class Activity 2 - Horizontal Scaling in the Cloud
- AWS Overview and demo
- Tutorial 4 Demo
- Ch. 5: Cloud Enabling Technology



19

## OBJECTIVES – 10/29

- Questions from 10/24
- Tutorials Questions
- Tutorial 6 - Serverless Databases
- **Class Activity 2 - Horizontal Scaling in the Cloud**
- AWS Overview and demo
- Tutorial 4 Demo
- Ch. 5: Cloud Enabling Technology



20

## OBJECTIVES – 10/29

- Questions from 10/24
- Tutorials Questions
- Tutorial 6 - Serverless Databases
- Class Activity 2 - Horizontal Scaling in the Cloud
- **AWS Overview and demo**
- Tutorial 4 Demo
- Ch. 5: Cloud Enabling Technology



21

## AWS OVERVIEW AND DEMO

22

## ONLINE CLOUD TUTORIALS

- From the eScience Institute @ UW Seattle:
- https://escience.washington.edu/
- Online cloud workshops
- Introduction to AWS, Azure, and Google Cloud
- Task: Deploying a Python DJANGO web application
- Self-guided workshop materials available online:
- https://cloudmaven.github.io/documentation/
- AWS Educate provides access to many online tutorials / learning resources:
- https://aws.amazon.com/education/awseducate/



23

## LIST OF TOPICS

- AWS Management Console
- Elastic Compute Cloud (EC2)
- Instance Storage: Virtual Disks on VMs
- Elastic Block Store: Virtual Disks on VMs
- Elastic File System (EFS)
- Amazon Machine Images (AMIs)
- EC2 Paravirtualization
- EC2 Full Virtualization (hvm)
- EC2 Virtualization Evolution
- (VM) Instance Actions
- EC2 Networking
- EC2 Instance Metadata Service
- Simple Storage Service (S3)
- AWS Command Line Interface (CLI)
- Legacy / Service Specific CLIs
- AMI Tools
- Signing Certificates
- Backing up live disks
- Cost Savings Measures
- Disk images and S3



24


25

## AWS EC2

- Elastic Compute Cloud
- Instance types: https://ec2instances.info
  - **On demand Instance** – full price
  - **Reserved Instance** – contract based where customer guarantees VM rental for a fixed period of time (e.g. 1 year, 3 years, etc.)
    Deeper discounts with longer term commitments
  - **Spot Instance** – portion of cloud capacity reserved for low cost instances, when demand exceeds supply instances are randomly terminated with 2-minute warning
    - Users can make diverse VM requests using different types, zones, regions, etc. to minimize instance terminations
    - Developers can design for failure because often only 1 or 2 VMs in a cluster fail at any given time. They then need to be replaced.
  - **Dedicated host** – reserved private HW (server)
  - Instance families -
    General, compute-optimized, memory-optimized, GPU, etc.



26

## AWS EC2 - 2

- Storage types
  - Instance storage - ephemeral storage
    - Temporary disk volumes stored on disks local to the VM
    - Evolution: physical hard disk drives (HDDs)
    - Solid state drives (SSDs)
    - Non-volatile memory express (NVMe) drives (closer to DRAM speed)
  - EBS - Elastic block store
    - Remotely hosted disk volumes
  - EFS - Elastic file system
    - Shared file system based on network file system
    - VMs, Lambdas, Containers mount/interact with shared file system
    - Somewhat expensive



27

## 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
- Multi-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…***



28

## ELASTIC BLOCK STORE

- EBS provides 1 drive to 1 virtual machine **(1 : 1)** **(not shared)**
- EBS cost model is different than instance storage (uses S3)
  - ~10¢ per GB/month for General Purpose Storage (GP2)
  - ~8¢ per GB/month for General Purpose Storage (GP3)
  - 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…



29

## EBS VOLUME TYPES - 2

- Metric: I/O Operations per Second (IOPS)
- **General Purpose 2 (GP2)**
  - 3 IOPS per GB, min 100 IOPS (<34GB), max of 16,000 IOPS
  - 250MB/sec throughput per volume
- **General Purpose 3** (GP3 – new Dec 2020)
  - Max 16,000 IOPS, Default 3,000 IOPS
  - GP2 requires creating a 1TB volume to obtain 3,000 IOPS
  - GP3 all volumes start at 3000 IOPS and 125 MB/s throughput
  - 1000 additional IOPS beyond 3000 is $5/month up to 16000 IOPS
  - 125 MB/s additional throughput is $5/month up to 1000 MB/s throughput



30

## EBS VOLUME TYPES - 3

- **Provisioned IOPS (IO1)**
  - Legacy, associated with GP2
  - Allows user to create custom disk volumes where they pay for a specified IOPS and throughput
  - 32,000 IOPS, and 500 MB/sec throughput per volume MAX
- **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 90 MB/sec throughput per volume
  - 5 ¢ per GB/month



31

## ELASTIC FILE SYSTEM (EFS)

- EFS provides 1 volume to many client **(1 : n) shared storage**
- Network file system (based on NFSv4 protocol)
- Shared file system for EC2, Fargate/ECS, Lambda
- Enables mounting (sharing) the same disk "volume" for R/W access across multiple instances at the same time
- Different performance and limitations vs. EBS/Instance store
- Implementation uses abstracted EC2 instances
- ~ 30 ¢ per GB/month storage – ***default burstable throughput***
- **Throughput modes:**
- Can modify modes only once every 24 hours
- Burstable Throughput Model:
  - Baseline – 50kb/sec per GB
  - Burst – 100MB/sec pet GB (for volumes sized 10GB to 1024 GB)
  - Credits - .72 minutes/day per GB



32

## ELASTIC FILE SYSTEM (EFS) - 2

- Burstable Throughput Rates

*Information subject to revision*

  - Throughput rates: baseline vs burst
  - Credit model for bursting: maximum burst per day

| 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% |



33

## ELASTIC FILE SYSTEM (EFS) - 3

*Information subject to revision*

- **Throughput Models**
- Provisioned Throughput Model
- For applications with:
  high performance requirements, but low storage requirements
- Get high levels of performance w/o overprovisioning capacity
- $6 MB/s-Month (Virginia Region)
  - Default is 50kb/sec for 1 GB, .05 MB/s = 30 ¢ per GB/month
- If file system metered size has higher baseline rate based on size, file system follows default Amazon EFS Bursting Throughput model
  - No charges for Provisioned Throughput below file system's entitlement in Bursting Throughput mode
  - Throughput entitlement = 50kb/sec per GB



34

## ELASTIC FILE SYSTEM (EFS) - 4

*Information subject to revision*

Performance Comparison, Amazon EFS and Amazon EBS

| | Amazon EFS | Amazon EBS Provisioned IOPS |
|---|---|---|
| Per-operation latency | Low, consistent latency. | Lowest, consistent latency. |
| Throughput scale | 10+ GB per second. | Up to 2 GB per second. |

Storage Characteristics Comparison, Amazon EFS and Amazon EBS

| | Amazon EFS | Amazon EBS Provisioned IOPS |
|---|---|---|
| Availability and durability | Data is stored redundantly across multiple AZs. | Data is stored redundantly in a single AZ. |
| Access | Up to thousands of Amazon EC2 instances, from multiple AZs, can connect concurrently to a file system. | A single Amazon EC2 instance in a single AZ can connect to a file system. |
| Use cases | Big data and analytics, media processing workflows, content management, web serving, and home directories. | Boot volumes, transactional and NoSQL databases, data warehousing, and ETL. |



35

## 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



36

## EC2 VIRTUALIZATION - PARAVIRTUAL

- **1st, 2nd, 3rd, 4th generation → XEN-based**
- **5th generation instances → AWS Nitro virtualization**
- XEN - two virtualization modes
- XEN Paravirtualization "paravirtual"
  - 10GB Amazon Machine Image – base image size limit
  - Addressed poor performance of old XEN HVM mode
  - I/O performed using special XEN kernel with XEN paravirtual mode optimizations for better performance
  - Requires OS to have an available paravirtual kernel
  - PV VMs: will use common **AKI** files on AWS – ***Amazon kernel image(s)***
    - *Look for common identifiers*



37

## 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 (under /boot)
  - No AKIs (kernel images)
  - Commonly used today (*EBS-backed instances*)



38

## EC2 VIRTUALIZATION - NITRO

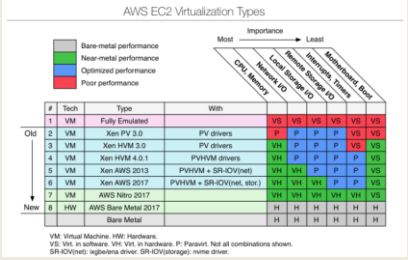
- Nitro based on Kernel-based-virtual-machines
  - Stripped down version of Linux KVM hypervisor
  - Uses KVM core kernel module
  - I/O access has a direct path to the device
- Goal: provide indistinguishable performance from bare metal



39

## EVOLUTION OF AWS VIRTUALIZATION

- From: http://www.brendangregg.com/blog/2017-11-29/aws-ec2-virtualization-2017.html




| # | Tech | Type | With | | | | | |
|---|---|---|---|---|---|---|---|---|
| 1 | VM | Fully Emulated | | VS | VS | VS | VS | VS |
| 2 | VM | Xen PV 3.0 | PV drivers | P | P | P | VS | VS |
| 3 | VM | Xen HVM 3.0 | PV drivers | VH | P | P | VS | VS |
| 4 | VM | Xen HVM 4.0.1 | PVHVM drivers | VH | P | P | P | VS |
| 5 | VM | Xen AWS 2013 | PVHVM + SR-IOV(net) | VH | VH | P | P | VS |
| 6 | VM | Xen AWS 2017 | PVHVM + SR-IOV(net, stor.) | VH | VH | VH | P | VS |
| 7 | VM | AWS Nitro 2017 | | VH | VH | VH | VH | VS |
| 8 | HW | AWS Bare Metal 2017 | | H | H | H | H | H |
| | | Bare Metal | | H | H | H | H | H |



40

## INSTANCE ACTIONS

- Stop
  - Costs of "pausing" an instance
- Terminate
- Reboot
- Image management
- Creating an image
  - EBS (snapshot)
- Bundle image
  - Instance-store



41

## 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



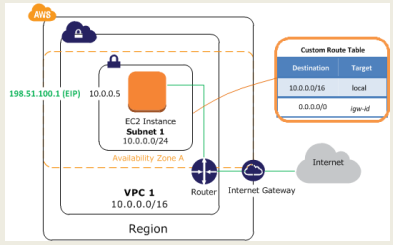
42

## WE WILL RETURN AT 4:50 PM

43

## SIMPLE VPC

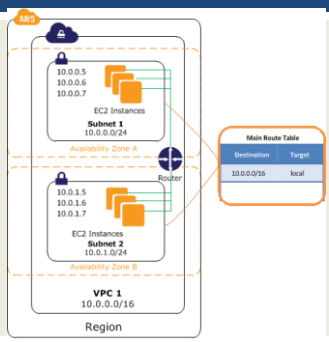
- Recommended when using Amazon EC2






44

## VPC SPANNING AVAILABILITY ZONES




45

## INSPECTING INSTANCE INFORMATION

- EC2 VMs run a local metadata service
- Can query instance metadata to self discover cloud configuration attributes

- 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 API that provides easy/formatted access to metadata



46

## 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
- Provides eventual consistency
- Can store Lambda function state for life of container.



47

## AWS CLI

- Launch Ubuntu 16.04 VM
  - Instances | Launch Instance
- Install the general AWS CLI
  - sudo apt install awscli
- Create config file

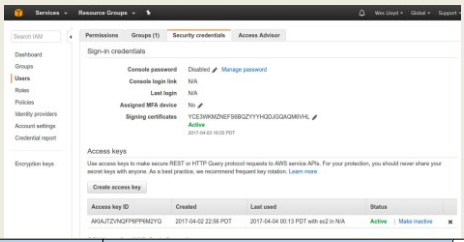
```
[default]
aws_access_key_id = <access key id>
aws_secret_access_key = <secret access key>
region = us-east-1
```



48

## AWS CLI - 2

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





49

## AWS CLI - 3

- Export the config file
  - Add to /home/ubuntu/.bashrc

```
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`



50

## LEGACY / SERVICE SPECIFIC CLI(S)

- `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 from Java:
- http://docs.aws.amazon.com/AWSJavaSDK/latest/javadoc/index.html
- Some AWS services have separate CLI installable by package



51

## AMI TOOLS

- Amazon Machine Images tools
- For working with disk volumes
- Can create live copies of any disk volume
  - Your local laptop, ec2 root volume (EBS), ec2 ephemeral disk
- Installation:
  https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ami-tools-commands.html
- AMI tools reference:
- https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ami-tools-commands.html
- Some functions may require private key & certificate files



52

## 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-certificate



53

## 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)**
- An alternative to bundling volumes and storing in S3 is to use a containerization tool such as Docker. . .
- Create image script . . .



54

## SCRIPT: CREATE A NEW INSTANCE STORE IMAGE FROM LIVE DISK VOLUME

```
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
```



55

## MAKE A DISK FROM AN IMAGE FILE

```
# ****************  ON THE LOCAL COMPUTER ********************
# create 1200 MB virtual disk = 1,258,291,200 bytes
sudo dd if=/dev/zero of=vhd.img bs=1M count=1200
# format the disk using the ext4 filesystem
sudo mkfs.ext4 vhd.img
# mount the disk at "/mnt"
sudo mount -t auto -o loop vhd.img /mnt
# check that the disk is mounted
df -h
# create a hello file (or copy data) to the new virtual disk
cd /mnt
sudo echo "hello world !" > hello.txt
ls -l
cd
# unmount the virtual disk
sudo umount /mnt
```



56

## COMPRESS IMAGE, PUSH TO S3

```
# compress the disk
bzip2 vhd.img

# push the disk image to S3
aws s3 cp vhd.img.bz2 s3://tcss562-f21-images
```



57

## RESTORE ON THE CLOUD

```
# ******************  ON THE AWS EC2 VM ********************
# with the awscli installed and configured

# download the image from S3
aws s3 cp s3://tcss562-f21-images/vhd.img.bz2 vhd.img.bz2

# uncompress the image
bzip2 -d vhd.img.bz2

# we need to calculate the number of sectors for the
partition
# disk sectors are 512 bytes each
# divide the disk size by 512 to determine sectors
# sectors = 1258291200 / 512 = 2459648

# create a disk partition for this disk that is
# 2459648 sectors in size using the ephemeral drive or
# a newly mounted EBS volume that is unformatted

sudo fdisk /dev/nvme1n1
```



58

## PARTITION THE DISK

```
Welcome to fdisk (util-linux 2.34).

Command (m for help): n
Partition type
   p   primary (0 primary, 0 extended, 4 free)
   e   extended (container for logical partitions)

Select (default p): p
Partition number (1-4, default 1): 1
First sector (2048-97656249, default 2048): 2048
Last sector, +/-sectors or +/-size{K,M,G,T,P} (2048-97656249, default
97656249): 2459648

Created a new partition 1 of type 'Linux' and of size 1.2 GiB.

Command (m for help): t
Selected partition 1
Hex code (type L to list all codes): 83
Changed type of partition 'Linux' to 'Linux'.

Command (m for help): w  (to write and exit)
```



59

## COPY DATA TO NEW DISK PARTITION

```
# now check if the partition has been created.
# it should be listed as /dev/nvme1n1p1:
ls /dev/nvme1n1*

# now copy the data to the partition
sudo dd if=vhd.img of=/dev/nvme1n1p1

# mount the disk
sudo mount /dev/nvme1n1p1 /mnt

# and check if the hello file is there
cat /mnt/hello.txt

# we were able to copy the disk image to the cloud
# and we never had to format the cloud disk
# this examples copies a filesystem from a local disk
# to the cloud disk
```



60

## FOR MORE INFORMATION

- Example script:
- https://faculty.washington.edu/wlloyd/courses/tcss562/examples/copy-disk-to-cloud.sh
- URLs:
- https://help.ubuntu.com/community/DriveImaging
- https://www.tecmint.com/create-virtual-harddisk-volume-in-linux/



61

## COST SAVINGS MEASURES

- ***From Tutorial 3:***
- #1: ALWAYS USE SPOT INSTANCES FOR COURSE/RESEARCH RELATED PROJECTS
- #2: NEVER LEAVE AN EBS VOLUME IN YOUR ACCOUNT THAT IS NOT ATTACHED TO A RUNNING VM
- #3: BE CAREFUL USING PERSISTENT REQUESTS FOR SPOT INSTANCES
- #4: TO SAVE/PERSIST DATA, USE EBS SNAPSHOTS AND THEN
- #5: DELETE EBS VOLUMES FOR TERMINATED EC2 INSTANCES.
- #6: UNUSED SNAPSHOTS AND UNUSED EBS VOLUMES SHOULD BE PROMPTLY DELETED !!
- #7: USE PERSISTENT SPOT REQUESTS AND THE "STOP" FEATURE TO PAUSE VMS DURING SHORT BREAKS



62

## OBJECTIVES – 10/29

- Questions from 10/24
- Tutorials Questions
- Tutorial 6 - Serverless Databases
- Class Activity 2 - Horizontal Scaling in the Cloud
- AWS Overview and demo
- **Tutorial 4 Demo**
- Ch. 5: Cloud Enabling Technology



63

## OBJECTIVES – 10/29

- Questions from 10/24
- Tutorials Questions
- Tutorial 6 - Serverless Databases
- Class Activity 2 - Horizontal Scaling in the Cloud
- AWS Overview and demo
- Tutorial 4 Demo
- **Ch. 5: Cloud Enabling Technology**



64

## CLOUD ENABLING TECHNOLOGY

65

## CLOUD ENABLING TECHNOLOGY

- *Adapted from Ch. 5 from Cloud Computing Concepts, Technology & Architecture*
- Broadband networks and internet architecture
- Data center technology
- Virtualization technology
- Multitenant technology
- Web/web services technology



66

## 1. BROADBAND NETWORKS AND INTERNET ARCHITECTURE

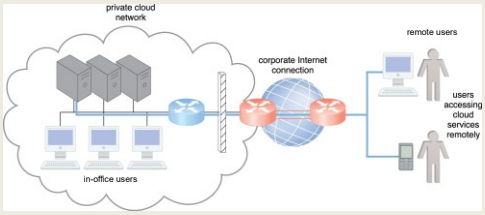
- Clouds must be connected to a network
- Inter-networking: Users' network must connect to cloud's network
- Public cloud computing relies heavily on the **Internet**



67

## PRIVATE CLOUD NETWORKING

- For institutions with in-house private clouds

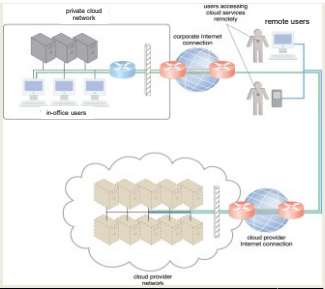





68

## PUBLIC CLOUD NETWORKING

- Resources can be extended by adding public cloud
- Places further dependency on the internet to provide connectivity






69

## INTERNETWORKING KEY POINTS

- Cloud consumers and providers typically communicate via the internet
- Decentralized provisioning and management model is not controlled by the cloud consumers or providers
- Inter-networking (internet) relies on connectionless packet switching and route-based interconnectivity
- Routers and switches support communication
- Network bandwidth and latency influence QoS, which is heavily impacted by network congestion



70

## CLOUD ENABLING TECHNOLOGY

- *Adapted from Ch. 5 from Cloud Computing Concepts, Technology & Architecture*
- Broadband networks and internet architecture
- Data center technology
- Virtualization technology
- Multitenant technology
- Web/web services technology



71

## 2. DATA CENTER TECHNOLOGY

- Grouping servers together (clusters):
- Enables power sharing
- Higher efficiency in shared IT resource usage (less duplication of effort)
- Improved accessibility and organization
- Key components:
  - Virtualized and physical server resources
  - Standardized, modular hardware
  - Automation support: enable server provisioning, configuration, patching, monitoring without supervision… ***tool/API support is desirable***



72

## CLUSTER MANAGEMENT TOOLS




73

## DATA CENTER TECHNOLOGY – KEY COMPONENTS

- Remote operation / management
- **High availability support**: **redundant everything** Includes: power supplies, cabling, environmental control systems, communication links, duplicate warm replica HW
- **Secure design**: physical and logical access control
- **Servers**: rackmount, etc.
- **Storage**: hard disk arrays (RAID)
- storage area network (SAN): disk array w/ multiple servers (individual nodes w/ disks) and a dedicated network
- network attached storage (NAS): inexpensive single node with collection of disks, provides shared filesystems, for NFS, etc.
- **Network hardware**: backbone routers (WAN to LAN connectivity), firewalls, VPN gateways, managed switches/routers



74

## CLOUD ENABLING TECHNOLOGY

- Broadband networks and internet architecture
- Data center technology
- Virtualization technology
- Multitenant technology
- Web/web services technology



75

## 3. VIRTUALIZATION TECHNOLOGY

- Convert a physical IT resource into a virtual IT resource
- Servers, storage, network, power (virtual UPSs)
- Virtualization supports:
  - Hardware independence
  - Server consolidation
  - Resource replication
  - Resource pooling
  - Elastic scalability
- Virtual servers
  - Operating-system based virtualization
  - Hardware-based virtualization



76

## VIRTUAL MACHINES

- Emulation/simulation of a computer in software
- Provides a substitute for a real computer or server
- Virtualization platforms provide functionality to run an entire operating system
- Allows running multiple different operating systems, or operating systems with different versions simultaneously on the same computer



77

## KEY VIRTUALIZATION TRADEOFF

- Tradeoff space:

**What is the "right" level of abstraction in the cloud for sharing resources with users?**

***Degree of Hardware Abstraction***


**Abstraction Concerns:**
- **Overhead**
- **Performance**
- **Isolation**
- **Security**



78

## ABSTRACTION CONCERNS

- **Overhead with too many instances w/ heavy abstractions**
  - Too many instances using a heavy abstraction can lead to hidden resource utilization and waste
  - Example: Dedicated server with 48 VMs each with separate instance of Ubuntu Linux
  - Idle VMs can reduce performance of co-resident jobs/tasks
- **"Virtualization" Overhead**
  - Cost of virtualization an OS instance
  - Overhead has dropped from ~100% to ~1% over last decade
- **Performance**
  - Impacted by weight of abstraction and virtualization overhead



79

## ABSTRACTION CONCERNS - 2

- **Isolation**
  - From others:
    What user A does should not impact user B in any noticeable way
- **Security**
  - User A and user B's data should be always separate
  - User A's actions are not perceivable by User B



80

## TYPES OF ABSTRACTION IN THE CLOUD

- **Virtual Machines** – original IaaS cloud abstraction
- **OS and Application Containers** – seen with CaaS
  - **OS Container** – replacement for VM, mimics full OS instance, heavier
  - OS containers run 100s of processes just like a VM
  - **App Container** – Docker: packages dependencies to easily transport and run an application anywhere
  - Application containers run only a few processes
- **Micro VMs** – FaaS / CaaS
  - Lighter weight alternative to full VM (KVM, XEN, VirtualBox)
  - Firecracker
- **Unikernel Operating Systems** – research mostly
  - Single process, multi-thread operating system
  - Designed for cloud, objective to reduce overhead of running too many OS instances



81

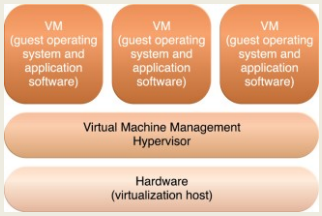
## VIRTUAL MACHINES

- **Type 1 hypervisor**
  - Typically involves a special virtualization kernel that runs directly on the system to share the underlying machine with many guest VMs
  - Paravirtualization introduced to directly share system resources with guests bypassing full emulation
  - VM becomes equal participant in sharing the network card for example
- **Type 2 hypervisor**
  - Typically involves the **Full Virtualization** of the guest, where everything is simulated/emulated
- Hardware level support (i.e. features introduced on CPUs) have made virtualization faster in all respects shrinking virtualization overhead



82

## TYPE 1 HYPERVISOR




- Host OS and VMs run atop the hypervisor
- The boot OS is the hypervisor kernel
- Xen dom0



83

## TYPE 1 HYPERVISOR

- Acts as a control program
- Miniature OS kernel that manages VMs
- Boots and runs on bare metal
- Also known as Virtual Machine Monitor (VMM)
- **Paravirtualization**: Kernel includes I/O drivers
- VM guest OSes must use special kernel to interoperate
- Paravirtualization provides hooks to the guest VMs
- Kernel traps instructions (i.e. device I/O) to implement sharing & multiplexing
- User mode instructions run directly on the CPU
- Objective: minimize virtualization overhead
- Classic example is XEN (dom0 kernel)



84

## COMMON VMMS: PARAVIRTUALIZATION

- **TYPE 1 Hypervisor**
- XEN
- Citrix Xen-server (a commercial version of XEN)
- VMWare ESXi
- KVM (virtualization support in kernel)

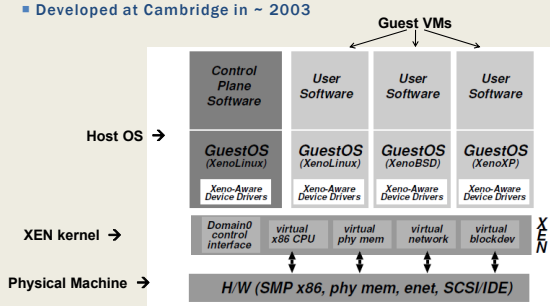
- Paravirtual I/O drivers introduced
  - XEN
  - KVM
  - Virtualbox



85

## XEN

- Developed at Cambridge in ~ 2003






86

## XEN - 2

- VMs managed as "domains"
- Domain 0 is the hypervisor domain
  - Host OS is installed to run on bare-metal, but doesn't directly facilitate virtualization (*unlike KVM*)
- Domains 1..n are guests (VMs) – not bare-metal

```
xentop - 17:53:48   Xen 3.1.2-398.el5
3 domains: 1 running, 2 blocked, 0 paused, 0 crashed, 0 dying, 0 shutdown
Mem: 8379564k total, 8377876k used, 1688k free    CPUs: 4 @ 2400MHz
      NAME  STATE    CPU(sec)  CPU(%)     MEM(k) MEM(%)  MAXMEM(k) MAXMEM(%) VCPUS
 NETS NETTX(k) NETRX(k) VBDS   VBD_OO   VBD_RD   VBD_WR SSID
    centos --b---         46     0.0     532352    6.4    1064960      12.7     1
     1    27960     885     1        0     6313    37119     0
  centos-2 --b---         17     0.0    1056640   12.6    2113536      25.2     1
     1       50       0     1        0     3981      541     0
  Domain-0 -----r       2979    19.3    6568960   78.4   no limit       n/a     4
     4  1057374  290072     0        0        0        0     0
```



87

## XEN - 3

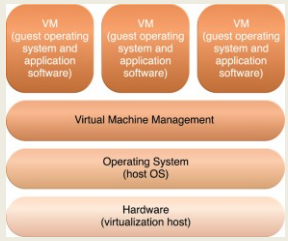
- Physical machine boots special XEN kernel
- Kernel provides paravirtual API to manage CPU & device multiplexing
- Guests require modified XEN-aware kernels
- Xen supports full-virtualization for unmodified OS guests in hvm mode
- Amazon EC2 largely based on modified version of XEN hypervisor (EC2 gens 1-4)
- XEN provides its own CPU schedulers, I/O scheduling



88

## TYPE 2 HYPERVISOR

- Adds additional layer






89

## TYPE 2 HYPERVISOR

- Problem: Original x86 CPUs could not trap special instructions
- Instructions not specially marked
- Solution: Use Full Virtualization
- Trap ALL instructions
- "Fully" simulate entire computer
- Tradeoff: Higher Overhead
- Benefit: Can virtualize any operating system without modification



90

## CHECK FOR VIRTUALIZATION SUPPORT

- See: https://cyberciti.biz/faq/linux-xen-vmware-kvm-intel-vt-amd-v-support
- # check for Intel VT CPU virtualization extensions on Linux
  `grep –color vmx /proc/cpuinfo`
- # check for AMD V CPU virtualization extensions on Linux
  `grep –color svm /proc/cpuinfo`
- Also see '`lscpu`' → "Virtualization:"
- Other Intel CPU features that help virtualization:
  `ept vpid tpr_shadow flexpriority vnmi`



91

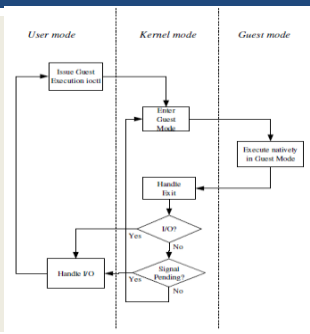
## KERNEL BASED VIRTUAL MACHINES (KVM)

- x86 HW notoriously difficult to virtualize
- Extensions added to 64-bit Intel/AMD CPUs
  - Provides hardware assisted virtualization
  - New "guest" operating mode
  - Hardware state switch
  - Exit reason reporting
  - Intel/AMD implementations different
    - Linux uses vendor specific kernel modules



92

## KVM – 2




93

## KVM – 3

- KVM has /dev/kvm device file node
  - Linux character device, with operations:
    - Create new VM
    - Allocate memory to VM
    - Read/write virtual CPU registers
    - Inject interrupts into vCPUs
    - Running vCPUs
- VMs run as Linux processes
  - Scheduled by host Linux OS
  - Can be pinned to specific cores with "taskset"



94

## KVM PARAVIRTUALIZED I/O

- KVM – Virtio
  - Custom Linux based paravirtual device drivers
  - Supersedes QEMU hardware emulation (full virt.)
  - Based on XEN paravirtualized I/O
  - Custom block device driver provides paravirtual device emulation
    - Virtual bus (memory ring buffer)
    - Requires hypercall facility
    - Direct access to memory



95

## KVM DIFFERENCES FROM XEN

- KVM requires CPU VMX support
  - Virtualization management extensions
- KVM can virtualize any OS without special kernels
  - Less invasive
- KVM was originally separate from the Linux kernel, but then integrated
- KVM is type 1 hypervisor because the machine boots Linux which has integrated support for virtualization
- Different than XEN because XEN kernel alone is not a full-fledged OS



96

## KVM ENHANCEMENTS

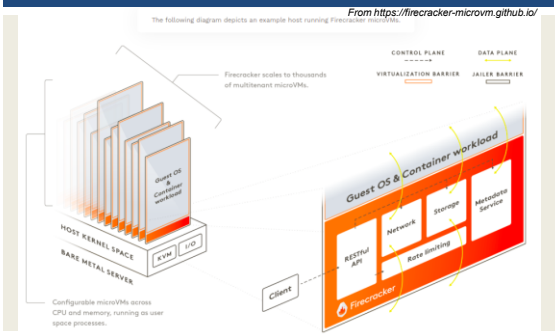
- Paravirtualized device drivers
  - Virtio
- Guest Symmetric Multiprocessor (SMP) support
  - Leverages multiple on-board CPUs
  - Supported as of Linux 2.6.23
- VM Live Migration
- Linux scheduler integration
  - Optimize scheduler with knowledge that KVM processes are virtual machines



97

## FIRECRACKER MICRO VM

From https://firecracker-microvm.github.io/




98

## FIRECRACKER MICRO VM

- Provides a virtual machine monitor (VMM) (i.e. hypervisor) using KVM to create and manage microVMs
- Has a minimalist design with goals to improve security, decreases the startup time, and increases hardware utilization
- Excludes unnecessary devices and guest functionality to reduce memory footprint and attack surface area of each microVM
- Supports boot time of <125ms, <5 MiB memory footprint
- Can run 100s of microVMs on a host, launching up to 150/sec
- Is available on 64-bit Intel, AMD, and Arm CPUs
- Used to host AWS Lambda and AWS Fargate
- Has been open sourced under the Apache 2.0 license



99

## FIRECRACKER - 2

- **Minimalistic**
- MicroVMs run as separate processes on the host
- Only 5 emulated devices are available: virtio-net, virtio-block, virtio-vsock, serial console, and a minimal keyboard controller used only to stop the microVM
- Rate limiters can be created and configured to provision resources to support bursts or specific bandwidth/operation limitations
- **Configuration**
- A RESTful API enables common actions such as configuring the number of vCPUs or launching microVMs
- A metadata service between the host and guest provides configuration information



100

## FIRECRACKER - 2

- **Security**
- Runs in user space ***(not the root user)*** on top of the Linux Kernel-based Virtual Machine (KVM) hypervisor to create microVMs
- Lambda functions, Fargate containers, or container groups can be encapsulated using Firecracker through KVM, enabling workloads from different customers to run on the same machine, without sacrificing security or efficiency
- MicroVMs are further isolated with common Linux user-space security barriers using a companion program called "jailer" which provides a second line of defense if KVM is compromised



101

## UNIKERNELS

- Lightweight alternative to containers and VMs
  - Custom Cloud Operating System
  - Single process, multiple threads, runs one program
  - Launch separately atop of hypervisor (XEN/KVM)
  - Reduce overhead, duplication of heavy weight OS
  - OSv is most well known unikernel
  - Several others exist has research projects
  - More information at: http://unikernel.org/
  - Google Trends OSv →



102

## VIRTUALIZATION MANAGEMENT

- Virtual infrastructure management (VIM) tools
- Tools that manage pools of virtual machines, resources, etc.
- Private cloud software systems can be considered as a VIM

- Considerations:
- Performance overhead
  - Paravirtualization: custom OS kernels, I/O passed directly to HW w/ special drivers
- Hardware compatibility for virtualization
- Portability: virtual resources tend to be difficult to migrate cross-clouds



103

## VIRTUAL INFRASTRUCTURE MANAGEMENT (VIM)

- Middleware to manage virtual machines and infrastructure of IaaS "clouds"

- Examples
  - OpenNebula
  - Nimbus
  - Eucalyptus
  - OpenStack



104

## VIM FEATURES

- Create/destroy VM Instances
- Image repository
  - Create/Destroy/Update images
  - Image persistence

- Contextualization of VMs
  - Networking address assignment
    - DHCP / Static IPs
  - Manage SSH keys



105

## VIM FEATURES - 2

- Virtual network configuration/management
  - Public/Private IP address assignment
  - Virtual firewall management
    - Configure/support isolated VLANs (private clusters)

- Support common virtual machine managers (VMMs)
  - XEN, KVM, VMware
  - Support via libvirt library



106

## VIM FEATURES - 3

- Shared "Elastic" block storage
  - Facility to create/update/delete VM disk volumes
    - Amazon EBS
    - Eucalyptus SC
    - OpenStack Volume Controller



107

## CONTAINER ORCHESTRATION FRAMEWORKS

- Middleware to manage Docker application container deployments across virtual clusters of Docker hosts (VMs)
- Considered Infrastructure-as-a-Service

- **Opensource**
- Kubernetes framework
- Docker swarm
- Apache Mesos/Marathon

- **Proprietary**
- Amazon Elastic Container Service



108

## CONTAINER SERVICES

- **Public cloud container cluster services**
- Azure Kubernetes Service (AKS)
- Amazon Elastic Container Service for Kubernetes (EKS)
- Google Kubernetes Engine (GKE)

- **Container-as-a-Service**
- Azure Container Instances (ACI – April 2018)
- AWS Fargate (November 2017)
- Google Kubernetes Engine Serverless Add-on (July 2018)
- Google Cloud Run (2019)
- Google Cloud Run jobs (2022)



109

## CLOUD ENABLING TECHNOLOGY

- *Adapted from Ch. 5 from Cloud Computing Concepts, Technology & Architecture*
- Broadband networks and internet architecture
- Data center technology
- Virtualization technology
- Multitenant technology
- Web/web services technology



110

## 4. MULTITENANT APPLICATIONS

- Each tenant (like in an apartment) has their own view of the application
- Tenants are unaware of their neighbors
- Tenants can only access their data, no access to data and configuration that is not their own

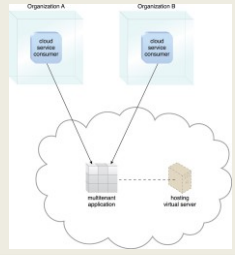
- Customizable features
  - UI, business process, data model, access control
- Application architecture
  - User isolation, data security, recovery/backup by tenant, scalability for a tenant, for tenants, metered usage, data tier isolation



111

## MULTITENANT APPS - 2

- Forms the basis for SaaS (applications)






112

## CLOUD ENABLING TECHNOLOGY

- *Adapted from Ch. 5 from Cloud Computing Concepts, Technology & Architecture*
- Broadband networks and internet architecture
- Data center technology
- Virtualization technology
- Multitenant technology
- Web/web services technology



113

## 5. WEB SERVICES/WEB

- Web services technology is a key foundation of cloud computing's "**as-a-service**" cloud delivery model
- SOAP – "Simple" object access protocol
  - First generation web services
  - WSDL – web services description language
  - UDDI – universal description discovery and integration
  - SOAP services have their own unique interfaces
- REST – instead of defining a custom technical interface REST services are built on the use of HTTP protocol
- HTTP GET, PUT, POST, DELETE



114

## HYPERTEXT TRANSPORT PROTOCOL (HTTP)

- An ASCII-based request/reply protocol for transferring information on the web
- HTTP request includes:
  - request method (GET, POST, etc.)
  - Uniform Resource Identifier (URI)
  - HTTP protocol version understood by the client
  - headers—extra info regarding transfer request
- HTTP response from server
  - Protocol version & status code →
  - Response headers
  - Response body

**HTTP status codes:**
2xx — *all is well*
3xx — *resource moved*
4xx — *access problem*
5xx — *server error*



115

## REST: REPRESENTATIONAL STATE TRANSFER

- Web services protocol
- *Supersedes SOAP* – Simple Object Access Protocol
- Access and manipulate web resources with a predefined set of stateless operations (known as web services)
- Requests are made to a URI
- Responses are most often in JSON, but can also be HTML, ASCII text, XML, no real limits as long as text-based
- HTTP verbs: GET, POST, PUT, DELETE, …



116

```
// SOAP REQUEST

POST /InStock HTTP/1.1
Host: www.bookshop.org
Content-Type: application/soap+xml; charset=utf-8
Content-Length: nnn

<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-
encoding">
<soap:Body xmlns:m="http://www.bookshop.org/prices">
  <m:GetBookPrice>
    <m:BookName>The Fleamarket</m:BookName>
  </m:GetBookPrice>
</soap:Body>
</soap:Envelope>
```



117

```
// SOAP RESPONSE
POST /InStock HTTP/1.1
Host: www.bookshop.org
Content-Type: application/soap+xml; charset=utf-8
Content-Length: nnn

<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-
encoding">
<soap:Body xmlns:m="http://www.bookshop.org/prices">
  <m:GetBookPriceResponse>
    <m: Price>10.95</m: Price>
  </m:GetBookPriceResponse>
</soap:Body>
</soap:Envelope>
```



118

```
// WSDL Service Definition
<?xml version="1.0" encoding="UTF-8"?>
<definitions  name ="DayOfWeek"
  targetNamespace="http://www.roguewave.com/soapworx/examples/DayOfWeek.wsdl"
  xmlns:tns="http://www.roguewave.com/soapworx/examples/DayOfWeek.wsdl"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns="http://schemas.xmlsoap.org/wsdl/">
  <message name="DayOfWeekInput">
    <part name="date" type="xsd:date"/>
  </message>
  <message name="DayOfWeekResponse">
    <part name="dayOfWeek" type="xsd:string"/>
  </message>
  <portType name="DayOfWeekPortType">
    <operation name="GetDayOfWeek">
      <input message="tns:DayOfWeekInput"/>
      <output message="tns:DayOfWeekResponse"/>
    </operation>
  </portType>
  <binding name="DayOfWeekBinding" type="tns:DayOfWeekPortType">
    <soap:binding style="document"
      transport="http://schemas.xmlsoap.org/soap/http"/>
    <operation name="GetDayOfWeek">
      <soap:operation soapAction="getdayofweek"/>
      <input>
        <soap:body use="encoded"
          namespace="http://www.roguewave.com/soapworx/examples"
          encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
      </input>
      <output>
        <soap:body use="encoded"
        namespace="http://www.roguewave.com/soapworx/examples"
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
      </output>
    </operation>
  </binding>
  <service name="DayOfWeekService" >
    <documentation>
      Returns the day-of-week name for a given date
    </documentation>
    <port name="DayOfWeekPort" binding="tns:DayOfWeekBinding">
      <soap:address location="http://localhost:8090/dayofweek/DayOfWeek"/>
    </port>
  </service>
</definitions>
```



119

## REST CLIMATE SERVICES EXAMPLE

- USDA Lat/Long Climate Service Demo
- Just provide a Lat/Long

```
// REST/JSON
// Request climate data for Washington

{
 "parameter": [
   {
     "name": "latitude",
     "value":47.2529
   },
   {
     "name": "longitude",
     "value":-122.4443
   }
 ]
}
```



120

## REST - 2

- App manipulates one or more types of resources.
- Everything the app does can be characterized as some kind of operation on one or more resources.
- Frequently services are CRUD operations (create/read/update/delete)
  - Create a new resource
  - Read resource(s) matching criterion
  - Update data associated with some resource
  - Destroy a particular a resource
- Resources are often implemented as objects in OO languages



121

## REST ARCHITECTURAL ADVANTAGES

- **Performance**: component interactions can be the dominant factor in user-perceived performance and network efficiency
- **Scalability**: to support large numbers of services and interactions among them
- **Simplicity**: of the Uniform Interface
- **Modifiability**: of services to meet changing needs (even while the application is running)
- **Visibility**: of communication between services
- **Portability**: of services by redeployment
- **Reliability**: resists failure at the system level as redundancy of infrastructure is easy to ensure



122

## QUESTIONS



123