TCSS 562: SOFTWARE ENGINEERING FOR CLOUD COMPUTING

Cloud Computing:
Cloud Delivery Models II
AWS Demo

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

FEEDBACK FROM 10/22

- **Can we have microservice written in different languages for ETL pipeline project?**
  - YES
  - May need to customize performance testing code
  - May need to recast examples provided in Java
- **Can we bypass API Gateway when calling AWS Lambda Microservice to avoid extra billing for API Gateway?**
  - YES
  - But no REST URL
  - First 1,000,000 calls free, 9¢/GB data storage
  - $3.50 for next 1,000,000 calls

FEEDBACK - 2

- **Is FaaS a promotion / update / improvement of PaaS?**
- **What is Container-as-a-Service?**
- Lambda pricing obfuscation example:
- **Is the breakeven point dependent on the # calls and workload?**
- **Breakeven point:**
- Making 2 client calls every second is only ~5 million calls/month (< $1)
- Breakeven point ***primarily*** depends on runtime and memory reservation size of the Lambda function

OBJECTIVES

- **From: Cloud Computing Concepts, Technology & Architecture:**
- Cloud Computing Concepts and Models
  - Roles and boundaries
  - Cloud characteristics
  - Cloud delivery models
  - Cloud deployment models
- AWS Demo

OBJECTIVES

- **From: Cloud Computing Concepts, Technology & Architecture:**
- Cloud Computing Concepts and Models
  - Roles and boundaries
  - Cloud characteristics
  - **Cloud delivery models**
  - Cloud deployment models
- AWS Demo

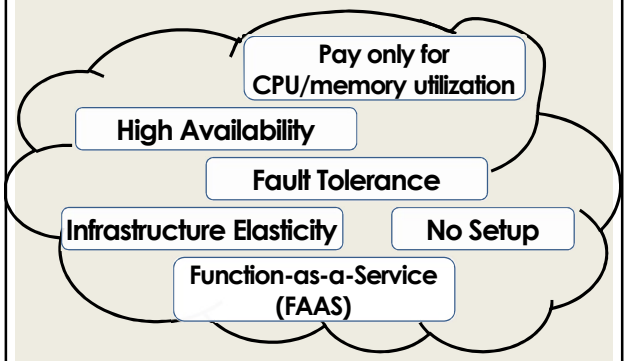
CLOUD DELIVERY MODELS

- What is the appropriate level of **abstraction**?
- How should applications be deployed?
  - IaaS, PaaS, SaaS, DbaaS, FaaS
- How do we ensure Quality-of-Service?
  - Performance, Availability, Responsiveness, Fault Tolerance
- How is ***scalability*** provided?
- How do we minimize hosting costs?
  - How do we estimate hosting costs?

CLOUD DELIVERY MODELS

- Infrastructure-as-a-Service (IaaS)
- Platform-as-a-Service (PaaS)
- Software-as-a-Service (SaaS)
- Function-as-a-Service (FaaS)
- Container-as-a-Service (CaaS)
- Other: as-a-Service offerings

SERVERLESS COMPUTING



SERVERLESS VS. FAAS

- **Serverless Computing**
- Refers to the avoidance of managing servers
- Can pertain to a number of "as-a-service" cloud offerings:
- Function-as-a-Service (FaaS)
  - Developers write small code snippets (microservices) which are deployed separately
- Database-as-a-Service (DBaaS)
- Container-as-a-Service (CaaS)
- Others…

- Serverless is a buzzword
- This space is evolving…

IAAS BILLING MODELS

- Virtual machines as-a-service at ¢ per hour
- No premium to scale:

```
      1000 computers    @     1 hour
  =      1 computer     @  1000 hours
```

- Illusion of infinite scalability to cloud user
- As many computers as you can afford
- Billing models are becoming increasingly granular
  - By the minute, second, 1/10th sec
- Auction-based instances: Spot instances →

FAAS COMPUTING BILLING MODELS

- **AWS Lambda Pricing**

- **FREE TIER:**
  first 1,000,000 function calls/month → FREE
  first 400,000 GB-sec/month → FREE

- Afterwards: *obfuscated pricing (AWS Lambda):*
  $0.0000002 per request
  $0.000000208 to rent 128MB / 100-ms
  $0.00001667 GB-second

WEBSERVICE HOSTING EXAMPLE

- **ON AWS Lambda**
- Each service call: 100% of 1 CPU-core
  100% of 4GB of memory
- Workload: 2 continuous client threads
- Duration: 1 month (30 days)

- **ON AWS EC2:**
- Amazon EC2 c4.large 2-vCPU VM
- Hosting cost: $72/month
  c4.large: 10¢/hour, 24 hrs/day x 30 days

- **How much would hosting this workload cost on AWS Lambda?**

PRICING OBFUSCATION

- Workload: 20,736,000 GB-sec
- FREE: - 400,000 GB-sec
- Ch
- Me 0
- In
- FR
- Ch
- Calls: $.84
- **Total:** $339.84
- **BREAK-EVEN POINT = ~4,320,000 GB-sec-month**

Worst-case scenario = ~4.72x !

AWS EC2:	$72.00
AWS Lambda:	$339.84

FAAS PRICING

- Break-even point is the point where renting VMs or deploying to a serverless platform (e.g. Lambda) is exactly the same.
- Our example is for one month
- Could also consider one day, one hour, one minute
- **What factors influence the break-even point for an application running on AWS Lambda?**

FACTORS IMPACTING PERFORMANCE OF FAAS COMPUTING PLATFORMS

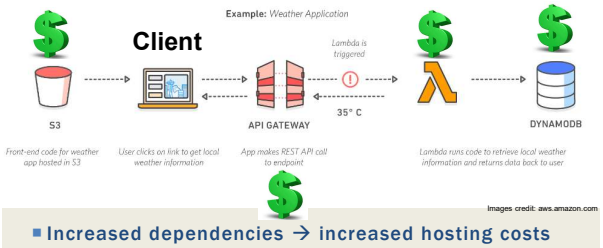
- Infrastructure elasticity
- Load balancing
- Provisioning variation
- Infrastructure retention: COLD vs. WARM
  - Infrastructure freeze/thaw cycle
- Memory reservation
- Service composition

FAAS CHALLENGES

- Vendor architectural lock-in – how to migrate?
- Pricing obfuscation – is it cost effective?
- Memory reservation – how much to reserve?
- Service composition – how to compose software?
- Infrastructure freeze/thaw cycle – how to avoid?

VENDOR ARCHITECTURAL LOCK-IN

- Cloud native (FaaS) software architecture requires external services/components



- Increased dependencies → increased hosting costs

PRICING OBFUSCATION

- **VM pricing:** hourly rental pricing, billed to nearest second is intuitive…
- **FaaS pricing:**

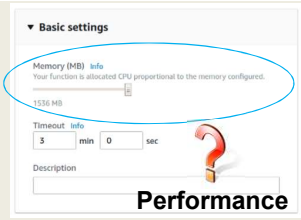
AWS Lambda Pricing

FREE TIER: first 1,000,000 function calls/month → FREE
first 400 GB-sec/month → FREE

- Afterwards: $0.0000002 per request
  $0.000000208 to rent 128MB / 100-ms

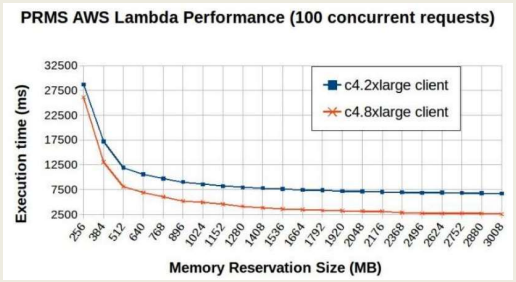
MEMORY RESERVATION QUEST

- Lambda memory reserved for functions
- UI provides "slider bar" to set function's memory allocation
- Resource capacity (CPU, disk, network) coupled to slider bar:
  "*every **doubling** of memory, **doubles** CPU…*"

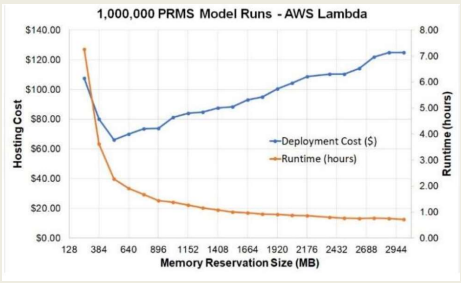


- **But how much memory do model services require?**

LAMBDA: PERFORMANCE VS MEMORY

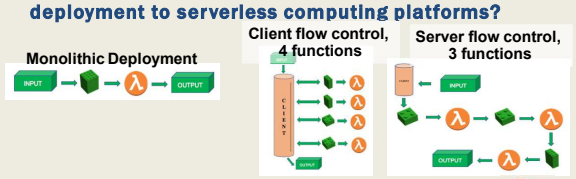


LAMBDA: OPTIMIZING COST OF 1,000,000 CALLS



SERVICE COMPOSITION

- **How should application code be composed for deployment to serverless computing platforms?**



- Recommended practice: Decompose into many microservices
- Platform limits: code + libraries ~250MB
- **How does composition impact the number of function invocations, and memory utilization?**

Performance

INFRASTRUCTURE FREEZE/THAW CYCLE

- Unused infrastructure is deprecated
  - *But after how long?*
- Infrastructure: VMs, "containers"
- **Provider-COLD / VM-COLD**
  - "Container" images - built/transferred to VMs
- **Container-COLD**
  - Image cached on VM
- **Container-WARM**
  - "Container" running on VM

Performance

Image from: Denver7 – The Denver Channel News

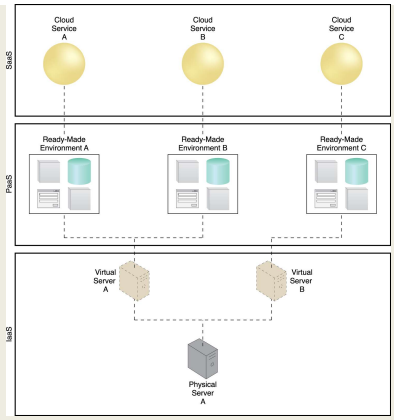
FUNCTION-AS-A-SERVICE

AWS Lambda Demo

24

SOFTWARE-AS-A-SERVICE

- Software applications as shared cloud service
- Nearly all server infrastructure management is abstracted away from the user
- Software is generally configurable
- SaaS can be a complete GUI/UI based environment
- Or UI-free (database-as-a-service)
- SaaS offerings
  - Google Docs
  - Office 365
  - Cloud9 Integrated Development Environment
  - Salesforce



CONTAINER-AS-A-SERVICE

- Cloud service model for deploying application containers (e.g. Docker) to the cloud
- Deploy containers without worrying about managing infrastructure:
  - Servers
  - Or container orchestration platforms
  - Container platform examples: Kubernetes, Docker swarm, Apache Mesos/Marathon, Amazon Elastic Container Service
  - Container platforms support creation of container clusters on the using cloud hosted VMs
- CaaS Examples: no VMs or clusters to manage
  - AWS Fargate
  - Azure Container Instances
  - Google KNative

OTHER CLOUD SERVICE MODELS

- IaaS
  - Storage-as-a-Service
- PaaS
  - Integration-as-a-Service
- SaaS
  - Database-as-a-Service
  - Testing-as-a-Service
  - Model-as-a-Service
- ?
  - Security-as-a-Service

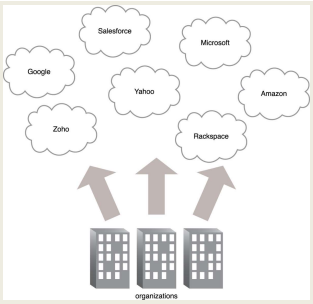
OBJECTIVES

- Cloud Computing Concepts and Models
  - Roles and boundaries
  - Cloud characteristics
  - Cloud delivery models
  - **Cloud deployment models**
- **AWS Demo**

CLOUD DEPLOYMENT MODELS

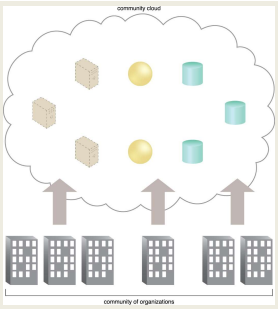
- Distinguished by ownership, size, access
- Four common models
  - Public cloud
  - Community cloud
  - Hybrid cloud
  - Private cloud

PUBLIC CLOUDS



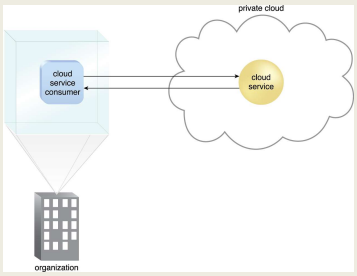
COMMUNITY CLOUD

- Specialized cloud built and shared by a particular community
- Leverage economies of scale within a community
- Research oriented clouds
- Examples:
  - Bionimbus - bioinformatics
  - Chameleon
  - CloudLab



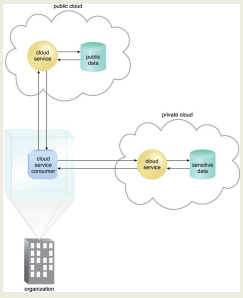
PRIVATE CLOUD

- Compute clusters configured as IaaS cloud
- Open source software
- Eucalyptus
- Openstack
- Apache Cloudstack
- Nimbus
- Virtualization: XEN, KVM, …



HYBRID CLOUD

- Extend private cloud typically with public or community cloud resources
- Cloud bursting: Scale beyond one cloud when resource requirements exceed local limitations
- Some resources can remain local for security reasons



OTHER CLOUDS

- Federated cloud
  - Simply means to aggregate two or more clouds together
  - Hybrid is typically private-public
  - Federated can be public-public, private-private, etc.
  - Also called inter-cloud
- Virtual private cloud
  - Google and Microsoft simply call these virtual networks
  - Ability to interconnect multiple independent subnets of cloud resources together
  - Resources allocated private IPs from individual network subnets can communicate with each other (10.0.1.0/24) and (10.0.2.0/24)
  - Subnets can span multiple availability zones within an AWS region

AWS DEMO

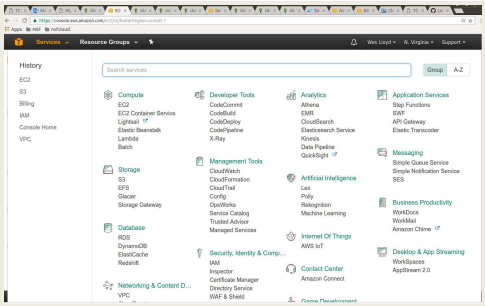
CLOUD 101 WORKSHOP

- From the eScience Institute @ UW Seattle:
- https://escience.washington.edu/
- Offers 1-day cloud workshops

- Introduction to AWS, Azure, and Google Cloud
- Task: Deploying a Python DJANGO web application
- Workshop materials available online:

- https://cloudmaven.github.io/documentation/rc_cloud101_immersion.html

AWS MANAGEMENT CONSOLE



AWS EC2

- Elastic Compute Cloud
- Instance types: https://ec2instances.info
  - On demand instance – full price
  - Reserved instance – contract based
  - Spot instance – auction based, terminates with 2 minute warning
  - Dedicated/reserved host – reserved HW
  - Reserved host
  - Instance families:
    General, compute-optimized, memory-optimized, GPU, etc.
- Storage types
  - Instance storage - ephemeral storage
  - EBS - Elastic block store
  - EFS - 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 (EFS)

- Network file system (based on NFSv4 protocol)
- Shared file system for EC2 instances
- 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

ELASTIC FILE SYSTEM (EFS) - 2

- Burstable Throughput Rates
  - 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%

ELASTIC FILE SYSTEM (EFS) - 3

- **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

ELASTIC FILE SYSTEM (EFS) - 4

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.

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 → 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*

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*)

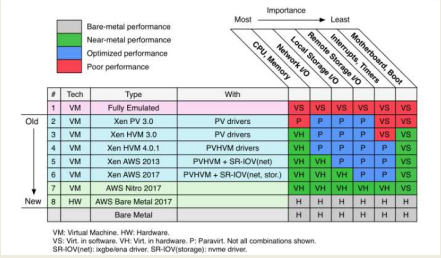
EC2 VIRTUALIZATION - NITRO

- 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

EVOLUTION OF AWS VIRTUALIZATION

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

AWS EC2 Virtualization Types



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

VM: Virtual Machine. HW: Hardware.
VS: Virt. in software. VH: Virt. in hardware. P: Paravirt. Not all combinations shown.
SR-IOV(net): ixgbe/ena driver. SR-IOV(storage): nvme driver.

INSTANCE ACTIONS

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

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

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)

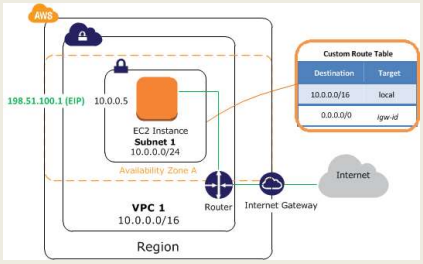
- Security groups
  - E.g. firewall

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

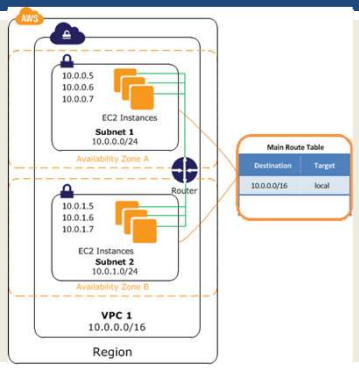
- VPC / Subnet / Internet Gateway / Router
- NAT-Gateway

SIMPLE VPC

- Recommended when using Amazon EC2



VPC SPANNING AVAILABILITY ZONES



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.

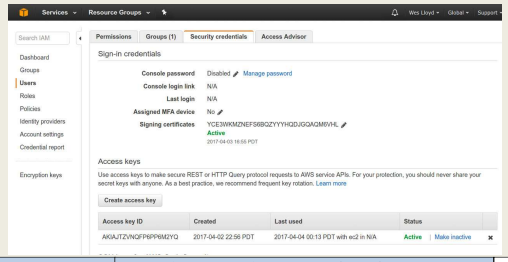
AWS CLI

- 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
```


AWS CLI - 2

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



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

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 from Java:
- http://docs.aws.amazon.com/AWSJavaSDK/latest/javadoc/index.html

INSPECTING INSTANCE INFORMATION

- 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 (??)

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

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 . . .

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
```


QUESTIONS