AWS CLI

RED/BLUE TEAM	RECON/ADMIN	CLOUD

The AWS Command Line Interface is a unified tool to manage your AWS services.

aws [options] <command> <subcommand> [parameters]

Command displays help for available top-level commands:
```
aws help
```

Command displays the available EC2 (Amazon EC2) specific commands:
```
aws ec2 help
```

Command displays detailed help for EC2 DescribeInstances operation.
```
aws ec2 describe-instances help
```

Cloudtrail - Logging and Auditing

List all trails
```
aws cloudtrail describe-trails
```

List all S3 buckets
```
aws s3 ls
```

Create a new trail
```
aws cloudtrail create-subscription --name awslog --s3-new-bucket
awslog2020
```

List the names of all trails
```
aws cloudtrail describe-trails --output text | cut -f 8
```

Get the status of a trail
```
aws cloudtrail get-trail-status --name awslog
```

Delete a trail
```
aws cloudtrail delete-trail --name awslog
```

Delete the S3 bucket of a trail
```
aws s3 rb s3://awslog2020 --force
```

Add tags to a trail, up to 10 tags allowed

```
aws cloudtrail add-tags --resource-id awslog --tags-list "Key=log-
type,Value=all"
```

List the tags of a trail

```
aws cloudtrail list-tags --resource-id-list
```

Remove a tag from a trail

```
aws cloudtrail remove-tags --resource-id awslog --tags-list
"Key=log-type,Value=all"
```

IAM USERS

***Limits = 5000 users, 100 group, 250 roles, 2 access keys per user*

List all user's info

```
aws iam list-users
```

List all user's usernames

```
aws iam list-users --output text | cut -f 6
```

List current user's info

```
aws iam get-user
```

List current user's access keys

```
aws iam list-access-keys
```

Create new user

```
aws iam create-user --user-name aws-admin2
```

Create multiple new users from file

```
allUsers=$(cat ./user-names.txt)
for userName in $allUsers; do
    aws iam create-user --user-name $userName
done
```

List all users

```
aws iam list-users --no-paginate
```

Get a specific user's info

```
aws iam get-user --user-name aws-admin2
```

Delete one user

```
aws iam delete-user --user-name aws-admin2
```

Delete all users

```
allUsers=$(aws iam list-users --output text | cut -f 6);
```

```
allUsers=$(cat ./user-names.txt)
for userName in $allUsers; do
    aws iam delete-user --user-name $userName
done
```

IAM PASSWORD POLICY

List password policy

```
aws iam get-account-password-policy
```

Set password policy

```
aws iam update-account-password-policy \
       --minimum-password-length 12 \
       --require-symbols \
       --require-numbers \
       --require-uppercase-characters \
       --require-lowercase-characters \
       --allow-users-to-change-password
```

Delete password policy

```
aws iam delete-account-password-policy
```

IAM ACCESS KEYS

List all access keys

```
aws iam list-access-keys
```

List access keys of a specific user

```
aws iam list-access-keys --user-name aws-admin2
```

Create a new access key

```
aws iam create-access-key --user-name aws-admin2 --output text |
tee aws-admin2.txt
```

List last access time of an access key

```
aws iam get-access-key-last-used --access-key-id
AKIAINA6AJZY4EXAMPLE
```

Deactivate an access key

```
aws iam update-access-key --access-key-id AKIAI44QH8DHBEXAMPLE --
status Inactive --user-name aws-admin2
```

Delete an access key

```
aws iam delete-access-key --access-key-id AKIAI44QH8DHBEXAMPLE --
user-name aws-admin2
```

IAM GROUPS, POLICIES, MANAGED POLICIES

List all groups

```
aws iam list-groups
```

Create a group

```
aws iam create-group --group-name FullAdmins
```

Delete a group

```
aws iam delete-group --group-name FullAdmins
```

List all policies

```
aws iam list-policies
```

Get a specific policy

```
aws iam get-policy --policy-arn <value>
```

List all users, groups, and roles, for a given policy

```
aws iam list-entities-for-policy --policy-arn <value>
```

List policies, for a given group

```
aws iam list-attached-group-policies --group-name FullAdmins
```

Add a policy to a group

```
aws iam attach-group-policy --group-name FullAdmins --policy-arn
arn:aws:iam::aws:policy/AdministratorAccess
```

Add a user to a group

```
aws iam add-user-to-group --group-name FullAdmins --user-name aws-
admin2
```

List users, for a given group

```
aws iam get-group --group-name FullAdmins
```

List groups, for a given user

```
aws iam list-groups-for-user --user-name aws-admin2
```

Remove a user from a group

```
aws iam remove-user-from-group --group-name FullAdmins --user-name
aws-admin2
```

Remove a policy from a group

```
aws iam detach-group-policy --group-name FullAdmins --policy-arn
arn:aws:iam::aws:policy/AdministratorAccess
```

Delete a group

```
aws iam delete-group --group-name FullAdmins
```

S3 BUCKETS

List existing S3 buckets

```
aws s3 ls
```

Create a public facing bucket

```
aws s3api create-bucket --acl "public-read-write" --bucket
bucket_name
```

Verify bucket was created

```
aws s3 ls | grep bucket_name
```

Check for public facing s3 buckets

```
aws s3api list-buckets --query 'Buckets[*].[Name]' --output text |
xargs -I {} bash -c 'if [[ $(aws s3api get-bucket-acl --bucket {} -
-query
'"'"'Grants[?Grantee.URI==`http://acs.amazonaws.com/groups/global/A
llUsers` && Permission==`READ`]'"'"' --output text) ]]; then echo
{} ; fi'
```

Check for public facing s3 buckets & update them to be private

```
aws s3api list-buckets --query 'Buckets[*].[Name]' --output text |
xargs -I {} bash -c 'if [[ $(aws s3api get-bucket-acl --bucket {} -
-query
'"'"'Grants[?Grantee.URI==`http://acs.amazonaws.com/groups/global/A
llUsers` && Permission==`READ`]'"'"' --output text) ]]; then aws
s3api put-bucket-acl --acl "private" --bucket {} ; fi'
```

EC2 KEYPAIRS

List all keypairs

```
aws ec2 describe-key-pairs
```

Create a keypair

```
aws ec2 create-key-pair --key-name <value> --output text
```

Create a new local private / public keypair, using RSA 4096-bit

```
ssh-keygen -t rsa -b 4096
```

Import an existing keypair

```
aws ec2 import-key-pair --key-name keyname_test --public-key-
material file:///home/user/id_rsa.pub
```

Delete a keypair

```
aws ec2 delete-key-pair --key-name <value>
```

SECURITY GROUPS

List all security groups

```
aws ec2 describe-security-groups
```

Create a security group

```
aws ec2 create-security-group --vpc-id vpc-1a2b3c4d --group-name
web-access --description "web access"
```

List details about a security group

```
aws ec2 describe-security-groups --group-id sg-0000000
```

Open port 80, for all users

```
aws ec2 authorize-security-group-ingress --group-id sg-0000000 --
protocol tcp --port 80 --cidr 0.0.0.0/24
```

```
Open port 22, just for "my IP"
aws ec2 authorize-security-group-ingress --group-id sg-0000000 --
protocol tcp --port 80 --cidr <my_ip>/32
```

Remove a firewall rule from a group

```
aws ec2 revoke-security-group-ingress --group-id sg-0000000 --
protocol tcp --port 80 --cidr 0.0.0.0/24
```

Delete a security group

```
aws ec2 delete-security-group --group-id sg-00000000
```

IMAGES

List all private AMI's, ImageId and Name tags

```
aws ec2 describe-images --filter "Name=is-public,Values=false" --
query 'Images[].[ImageId, Name]' --output text | sort -k2
```

Delete an AMI, by ImageId

```
aws ec2 deregister-image --image-id ami-00000000
```

INSTANCES

List all instances (running, and not running)

```
aws ec2 describe-instances
```

List all instances running

```
aws ec2 describe-instances --filters Name=instance-state-
name,Values=running
```

Create a new instance

```
aws ec2 run-instances --image-id ami-f0e7d19a --instance-type
t2.micro --security-group-ids sg-00000000 --dry-run
```

Stop an instance

```
aws ec2 terminate-instances --instance-ids <instance_id>
```

List status of all instances

```
aws ec2 describe-instance-status
```

List status of a specific instance

```
aws ec2 describe-instance-status --instance-ids <instance_id>
```

List all running instance, Name tag and Public IP Address

```
aws ec2 describe-instances --filters Name=instance-state-
name,Values=running --query
'Reservations[].Instances[].[PublicIpAddress,
Tags[?Key==`Name`].Value | [0] ]' --output text | sort -k2
```

INSTANCES TAGS

List the tags of an instance

```
aws ec2 describe-tags
```

Add a tag to an instance

```
aws ec2 create-tags --resources "ami-1a2b3c4d" --tags
Key=name,Value=debian
```

Delete a tag on an instance

```
aws ec2 delete-tags --resources "ami-1a2b3c4d" --tags
Key=Name,Value=
```

CLOUDWATCH LOG GROUPS

Create a group

```
aws logs create-log-group --log-group-name "DefaultGroup"
```

List all log groups

```
aws logs describe-log-groups
```

```
aws logs describe-log-groups --log-group-name-prefix "Default"
```

Delete a group

```
aws logs delete-log-group --log-group-name "DefaultGroup"
```

CLOUDWATCH LOG STREAMS

Create a log stream

```
aws logs create-log-stream --log-group-name "DefaultGroup" --log-
stream-name "syslog"
```

List details on a log stream

```
aws logs describe-log-streams --log-group-name "syslog"
```

```
aws logs describe-log-streams --log-stream-name-prefix "syslog"
```

Delete a log stream

```
aws logs delete-log-stream --log-group-name "DefaultGroup" --log-
stream-name "Default Stream"
```

LAMBDA

Get Lambda function config

```
aws lambda get-function-configuration --function-name
<CUSTOM_FUNCTION_NAME> --profile <PROFILE_NAME>
```

SNS

Get Simple Notification Service configurations

```
aws sns list-topics --profile <PROFILE_NAME>
aws sns get-topic-attributes --topic-arn "arn:aws:sns:us-east-
1:945109781822:<custom_suffix>" --profile <PROFILE_NAME>
aws sns list-subscriptions --profile <PROFILE_NAME>
aws sns get-subscription-attributes --subscription-arn
"arn:aws:sns:us-east-1:945109781822:<custom_part>:6d92f5d3-f299-
485d-b6fb-1aca6d9a497c" --profile <PROFILE_NAME>
```

RDS

Get database instances

```
aws rds describe-db-security-groups --db-security-group-name
<DB_SG_NAME> --profile <PROFILE_NAME>
aws rds describe-db-instances --db-instance-identifier
<DB_INSTANCE_ID> --profile <PROFILE_NAME>
```

REFERENCE:
https://github.com/aws/aws-cli
https://docs.aws.amazon.com/cli/latest/userguide/cli-chap-welcome.html
https://gist.github.com/apolloclark/b3f60c1f68aa972d324b

CLOUDTRAIL MONITORING

Successful Logins

Example search below returns successful authentications without multi-factor authentication. It can help detect suspicious logins or accounts on which MFA is not enforced.

```
sourcetype="aws:cloudtrail" eventName="ConsoleLogin"
"responseElements.ConsoleLogin"=Success
"additionalEventData.MFAUsed"=No
```

Failed Logins by Source

Example search returns a table of failed authentication, including the source IP, country, city and the reason why the authentication failed.

```
sourcetype="aws:cloudtrail" eventName="ConsoleLogin"
"responseElements.ConsoleLogin"=Failure
| iplocation sourceIPAddress
| stats count by userName, userIdentity.accountId, eventSource,
sourceIPAddress, Country, City, errorMessage
| sort - count
```

CryptoMining GPU Instance Abuse

Example of Splunk search to identify GPU instances that have been started.

```
sourcetype="aws:cloudtrail" eventSource="ec2.amazonaws.com"
eventName="RunInstances"
| spath output=instanceType path=requestParameters.instanceType
| spath output=minCount
path=requestParameters.instancesSet{}.items{}.minCount
| search instanceType IN ("p3.2xlarge", "p3.8xlarge",
"p3.16xlarge", "p3dn.24xlarge", "p2.xlarge", "p2.8xlarge",
"p2.16xlarge", "g3s.xlarge", "g3.4xlarge", "g3.8xlarge",
"g3.16xlarge")
| stats count by eventSource, eventName, awsRegion, userName,
userIdentity.accountId, sourceIPAddress, userIdentity.type,
requestParameters.instanceType,
responseElements.instancesSet.items{}.instanceId,
responseElements.instancesSet.items{}.networkInterfaceSet.items{}.p
rivateIpAddress, minCount
| fields - count
```

Security Group Configurations

Example search below looks for rules allowing inbound traffic on port 22 from any IPs. Then we look for the associated instance IDs and append them to the list.

```
sourcetype="aws:cloudtrail" eventSource="ec2.amazonaws.com"
eventName="AuthorizeSecurityGroupIngress"
| spath output=fromPort
path=requestParameters.ipPermissions.items{}.fromPort
| spath output=toPort
path=requestParameters.ipPermissions.items{}.toPort
| spath output=cidrIp
path=requestParameters.ipPermissions.items{}.ipRanges.items{}.cidrI
p
| spath output=groupId path=requestParameters.groupId
| spath output=accountId path=userIdentity.accountId
| spath output=type path=userIdentity.type
| search fromPort=22 toPort=22 AND cidrIp="0.0.0.0/0"
| spath output=ipPermissions
path=requestParameters.ipPermissions.items{}
| mvexpand ipPermissions
| fields - fromPort, toPort, cidrIp
| spath input=ipPermissions
| spath output=cidrIp path=ipRanges.items{}.cidrIp
input=ipPermissions
| join groupId
    [ search index=aws eventName=RunInstances earliest=-7d
    | fields
"responseElements.instancesSet.items{}.groupSet.items{}.groupId",
"responseElements.instancesSet.items{}.instanceId"
    | rename
responseElements.instancesSet.items{}.groupSet.items{}.groupId as
groupId, "responseElements.instancesSet.items{}.instanceId" as
instanceId]
| stats values(instanceId) by groupId, userName, accountId, type,
sourceIPAddress, cidrIp, fromPort, toPort, ipProtocol
```

Network ACL Creation

Example below searches for creation of Network ACL rules allowing inbound connections from any sources.

```
sourcetype="aws:cloudtrail" eventSource="ec2.amazonaws.com"
eventName=CreateNetworkAclEntry
| spath output=cidrBlock path=requestParameters.cidrBlock
| spath output=ruleAction path=requestParameters.ruleAction
| search cidrBlock=0.0.0.0/0 ruleAction=Allow
```

Detect Public S3 Buckets

Eample search looking for the PutBucketAcl event name where the grantee URI is AllUsers we can identify and report the open buckets.

```
sourcetype=aws:cloudtrail AllUsers eventName=PutBucketAcl
errorCode=Success
| spath output=userIdentityArn path=userIdentity.arn
| spath output=bucketName path=requestParameters.bucketName
| spath output=aclControlList
path=requestParameters.AccessControlPolicy.AccessControlList
| spath input=aclControlList output=grantee path=Grant{}
| mvexpand grantee
| spath input=grantee
| search Grantee.URI=*AllUsers
| rename userIdentityArn as user
| table _time, src,awsRegion Permission, Grantee.URI, bucketName,
user
```

VPC Traffic Mirroring

Capture & Inspect Network Traffic

```
aws ec2 create-traffic-mirror-filter --description "TCP Filter"
```

REFERENCE:
https://0x00sec.org/t/a-blue-team-guide-to-aws-cloudtrail-monitoring/15086
https://docs.aws.amazon.com/vpc/latest/mirroring/traffic-mirroring-filter.html#create-traffic-mirroring-filter

A A

AWS_Exploit

RED TEAM	EXPLOITATION	CLOUD

NIMBOSTRATUS

Install

```
git clone git@github.com:andresriancho/nimbostratus.git
cd nimbostratus
pip install -r requirements.txt
```

Prerequisites

Amazon AWS User account
Access Key
Boto Python 2.7 library

Insert VULN_URL into the utils/mangle.py file. Run dump-metada:

```
nimbostratus -v dump-ec2-metadata --mangle-
function=core.utils.mangle.mangle
```

Enumerate meta-data service of target using mangle function & retrieve any access key credentials found on the meta-data server:

```
nimbostratus -v dump-credentials --mangle-
function=core.utils.mangle.mangle
```

Dump all permissions for the provided credentials. Use right after dump-credentials to know which permissions are available:

```
nimbostratus  dump-permissions --access-key=**************PXXQ --
secret-key=*****************************SUW --token
*****************************************JFE
```

Create a new user. Assigns a random name to the created user and attaches a policy which looks like this:

```
    {
        "Version": "2012-10-17",
        "Statement": [
            {
                "Effect": "Allow",
                "Action": "*",
                "Resource": "*"
            }
        ]
    }
```

Execute:

```
nimbostratus -v create-iam-user --access-key **************UFUA --
secret-key *************************************DDxSZ --token
****************************************tecaoI
```

Create RDS database snapshot:

```
nimbostratus -v snapshot-rds --access-key ********AUFUA --secret-
key ****************************yDDxSZ --token
***********************************************K2g2QU= --rds-name
<DB_NAME> --password ********* --region us-west-2
```

PACU

Install

```
git clone https://github.com/RhinoSecurityLabs/pacu
cd pacu
bash install.sh
python3 pacu.py
```

Starting Pacu

python3 pacu.py

```
>set_keys
```

#Key alias - Used internally within Pacu and is associated with a AWS key pair. Has no bearing on AWS permissions.
#Access Key - Generated from an AWS User
#Secret Key - Secret key associated with access key. Omitted in image.
#(Optional) Session Key - serves as a temporary access key to access AWS services.
***provide a session name, after which you can add your compromised credentials with the set_keys command and begin running modules*

Running Modules

```
#list out modules
```

```
> ls
```

```
SYNTAX:> run <module_name> [--keyword-arguments]
```

PACU MODULES

iam__enum_assume_role
Enumerates existing roles in other AWS accounts to try and gain access via misconfigurations.

iam__enum_users
Enumerates IAM users in a separate AWS account, given the account ID.

s3__bucket_finder
Enumerates/bruteforces S3 buckets based on different parameters.

aws__enum_account
Enumerates data About the account itself.

aws__enum_spend
Enumerates account spend by service.

codebuild__enum
Enumerates CodeBuild builds and projects while looking for sensitive data

ebs__enum_volumes_snapshots
Enumerates EBS volumes and snapshots and logs any without encryption.

ec2__check_termination_protection
Collects a list of EC2 instances without termination protection.

ec2__download_userdata
Downloads User Data from EC2 instances.

ec2__enum
Enumerates a ton of relevant EC2 info.

glue__enum
Enumerates Glue connections, crawlers, databases, development endpoints, and jobs.

iam__enum_permissions
Tries to get a confirmed list of permissions for the current (or all) user(s).

iam__enum_users_roles_policies_groups

Enumerates users, roles, customer-managed policies, and groups.

iam__get_credential_report
Generates and downloads an IAM credential report.

inspector__get_reports
Captures vulnerabilties found when running a preconfigured inspector report.

lambda__enum
Enumerates data from AWS Lambda.

lightsail__enum
Captures common data associated with Lightsail

iam__privesc_scan
An IAM privilege escalation path finder and abuser.
**WARNING: Due to the implementation in IAM policies, this module has a difficult time parsing "NotActions". LATERAL_MOVE

cloudtrail__csv_injection
Inject malicious formulas/data into CloudTrail event history.

vpc__enum_lateral_movement
Looks for Network Plane lateral movement opportunities.

api_gateway__create_api_keys
Attempts to create an API Gateway key for any/all REST APIs that are defined.

ebs__explore_snapshots
Restores and attaches EBS volumes/snapshots to an EC2 instance of your choice.

ec2__startup_shell_script
Stops and restarts EC2 instances to execute code.

lightsail__download_ssh_keys
Downloads Lightsails default SSH key pairs.

lightsail__generate_ssh_keys
Creates SSH keys for available regions in AWS Lightsail.

lightsail__generate_temp_access
Creates temporary SSH keys for available instances in AWS Lightsail.

systemsmanager__rce_ec2
Tries to execute code as root/SYSTEM on EC2 instances.
**NOTE: Linux targets will run the command using their default shell (bash/etc.) and Windows hosts will run the command using

PowerShell, so be weary of that when trying to run the same command against both operating systems.Sometimes Systems Manager Run **Command can delay the results of a call by a random amount. Experienced 15 minute delays before command was executed on the target.

ec2__backdoor_ec2_sec_groups
Adds backdoor rules to EC2 security groups.

iam__backdoor_assume_role
Creates assume-role trust relationships between users and roles.

iam__backdoor_users_keys
Adds API keys to other users.

iam__backdoor_users_password
Adds a password to users without one.

s3__download_bucket
Enumerate and dumps files from S3 buckets.

cloudtrail__download_event_history
Downloads CloudTrail event history to JSON files to ./sessions/[current_session_name]/downloads/cloudtrail_[region]_event_history_[timestamp].json.
**NOTE: This module can take a very long time to complete. A rough estimate is about 10000 events retrieved per five minutes.

cloudwatch__download_logs
Captures CloudWatch logs and downloads them to the session downloads folder

detection__disruption
Disables, deletes, or minimizes various logging/monitoring services.

detection__enum_services
Detects monitoring and logging capabilities.

elb__enum_logging
Collects a list of Elastic Load Balancers without access logging and write a list of ELBs with logging disabled to ./sessions/[current_session_name]/downloads/elbs_no_logs_[timestamp].csv.

guardduty__whitelist_ip
Adds an IP address to the list of trusted IPs in GuardDuty.
**NOTE: This will not erase any existing GuardDuty findings, it will only prevent future findings related to the included IP addresses.

**WARNING: Only one list of trusted IP addresses is allowed per GuardDuty detector. This module will prompt you to delete an existing list if you would like, but doing so could have unintended bad consequences on the target AWS environment.

waf__enum
Detects rules and rule groups for WAF.

REFERENCE:
https://andresriancho.github.io/nimbostratus/
https://www.cloudsecops.com/post-exploitation-in-aws/
https://github.com/RhinoSecurityLabs/pacu
https://github.com/puresec/awesome-serverless-security/
https://zoph.me/posts/2019-12-16-aws-security-toolbox/
https://know.bishopfox.com/research/privilege-escalation-in-aws
https://github.com/BishopFox/smogcloud
https://github.com/bishopfox/dufflebag
https://rhinosecuritylabs.com/aws/abusing-vpc-traffic-mirroring-in-aws/

A **A**

AWS_Hardening

BLUE TEAM	CONFIGURATION	CLOUD

AWS Best Practices Rules
https://www.cloudconformity.com/knowledge-base/aws/

A **A**

AWS_Terms

ALL	GENERAL	CLOUD

AWS IoT: AWS IoT is a managed cloud service that lets connected devices easily and securely interact with cloud applications and other devices.
Certificate Manager: AWS Certificate Manager easily provision, manage, and deploy Secure Sockets Layer/Transport Layer Security (SSL/TLS) certificates for use with AWS services.
CloudFormation: AWS CloudFormation lets you create and update a collection of related AWS resources in a predictable fashion.
CloudFront: Amazon CloudFront provides a way to distribute content to end-users with low latency and high data transfer speeds.
CloudSearch: AWS CloudSearch is a fully managed search service for websites and apps.
CloudTrail: AWS CloudTrail provides increased visibility into user activity by recording API calls made on your account.

Data Pipeline: AWS Data Pipeline is a lightweight orchestration service for periodic, data-driven workflows.
DMS: AWS Database Migration Service (DMS) helps you migrate databases to the cloud easily and securely while minimizing downtime.
DynamoDB: Amazon DynamoDB is a scalable NoSQL data store that manages distributed replicas of your data for high availability.
EC2: Amazon Elastic Compute Cloud (EC2) provides resizable compute capacity in the cloud.
EC2 Container Service: Amazon ECS allows you to easily run and manage Docker containers across a cluster of Amazon EC2 instances.
Elastic Beanstalk: AWS Elastic Beanstalk is an application container for deploying and managing applications.
ElastiCache: Amazon ElastiCache improves application performance by allowing you to retrieve information from an in-memory caching system.
Elastic File System: Amazon Elastic File System (Amazon EFS) is a file storage service for Amazon Elastic Compute Cloud (Amazon EC2) instances.
Elasticsearch Service: Amazon Elasticsearch Service is a managed service that makes it easy to deploy, operate, and scale Elasticsearch, a popular open-source search and analytics engine.
Elastic Transcoder: Amazon Elastic Transcoder lets you convert your media files in the cloud easily, at low cost, and at scale
EMR: Amazon Elastic MapReduce lets you perform big data tasks such as web indexing, data mining, and log file analysis.
Glacier: Amazon Glacier is a low-cost storage service that provides secure and durable storage for data archiving and backup.
IAM: AWS Identity and Access Management (IAM) lets you securely control access to AWS services and resources.
Inspector: Amazon Inspector enables you to analyze the behavior of the applications you run in AWS and helps you to identify potential security issues.
Kinesis: Amazon Kinesis services make it easy to work with real-time streaming data in the AWS cloud.
Lambda: AWS Lambda is a compute service that runs your code in response to events and automatically manages the compute resources for you.
Machine Learning: Amazon Machine Learning is a service that enables you to easily build smart applications.
OpsWorks: AWS OpsWorks is a DevOps platform for managing applications of any scale or complexity on the AWS cloud.
RDS: Amazon Relational Database Service (RDS) makes it easy to set up, operate, and scale familiar relational databases in the cloud.
Redshift: Amazon Redshift is a fast, fully managed, petabyte--scale data warehouse that makes it cost-effective to analyze all your data using your existing business intelligence tools.

Route 53: Amazon Route 53 is a scalable and highly available Domain Name System (DNS) and Domain Name Registration service.
SES: Amazon Simple Email Service (SES) enables you to send and receive email.
SNS: Amazon Simple Notification Service (SNS) lets you publish messages to subscribers or other applications.
Storage Gateway: AWS Storage Gateway securely integrates on-premises IT environments with cloud storage for backup and disaster recovery.
SQS: Amazon Simple Queue Service (SQS) offers a reliable, highly scalable, hosted queue for storing messages.
SWF: Amazon Simple Workflow (SWF) coordinates all of the processing steps within an application.
S3: Amazon Simple Storage Service (S3) can be used to store and retrieve any amount of data.
VPC: Amazon Virtual Private Cloud (VPC) lets you launch AWS resources in a private, isolated cloud.

REFERENCE:
https://www.northeastern.edu/graduate/blog/aws-terminology/

AWS_Tricks

ALL	MISC	CLOUD

SUBNETS

Creating A Subnet

```
aws ec2 create-subnet --vpc-id <vpc_id> --cidr-block <cidr_block> --availability-zone <availability_zone> --region <region>
```

Auto Assigning Public IPs To Instances In A Public Subnet

```
aws ec2 modify-subnet-attribute --subnet-id <subnet_id> --map-public-ip-on-launch --region <region>
```

VPC

Creating A VPC

```
aws ec2 create-vpc --cidr-block <cidr_block> --regiosn <region>
```

Allowing DNS hostnames

```
aws ec2 modify-vpc-attribute --vpc-id <vpc_id> --enable-dns-hostnames "{\"Value\":true}" --region <region>
```

NAT

Setting Up A NAT Gateway

#Allocate Elastic IP

```
aws ec2 allocate-address --domain vpc --region <region>
```

#AllocationId to create the NAT Gateway for the public zone

```
aws ec2 create-nat-gateway --subnet-id <subnet_id> --allocation-id
<allocation_id> --region <region>
```

S3 API

Listing Only Bucket Names

```
aws s3api list-buckets --query 'Buckets[].Name'
```

Getting a Bucket Region

```
aws s3api get-bucket-location --bucket <bucket_name>
```

Syncing a Local Folder with a Bucket

```
aws s3 sync <local_path> s3://<bucket_name>
```

Copying Folders

```
aws s3 cp <folder_name>/ s3://<bucket_name>/ --recursive
```

To exclude files from copying

```
aws s3 cp <folder_name>/ s3://<bucket_name>/ --recursive --exclude
"<file_name_or_a_wildcard_extension>"
```

To exclude a folder from copying

```
aws s3 cp example.com/ s3://example-backup/ --recursive --exclude
".git/*"
```

Removing a File from a Bucket

```
aws s3 rm s3://<bucket_name>/<file_name>
```

Deleting a Bucket

```
aws s3 rb s3://<bucket_name> --force
```

Emptying a Bucket

```
aws s3 rm s3://<bucket_name>/<key_name> --recursive
```

EC2 Instance

Creating AMI Without Rebooting the Machine

```
aws ec2 create-image --instance-id <instance_id> --name "image-
$(date +'%Y-%m-%d_%H-%M-%S')" --description "image-$(date
+'%Y-%m-%d_%H-%M-%S')" --no-reboot
```

LAMBDA

Using AWS Lambda with Scheduled Events

```
sid=Sid$(date +%Y%m%d%H%M%S); aws lambda add-permission --
statement-id $sid --action 'lambda:InvokeFunction' --principal
events.amazonaws.com --source-arn
arn:aws:events:<region>:<arn>:rule/AWSLambdaBasicExecutionRole --
function-name function:<awsents> --region <region>
```

Deleting Unused Volumes

```
for x in $(aws ec2 describe-volumes --filters
Name=status,Values=available  --profile <your_profile_name>|grep
VolumeId|awk '{print $2}' | tr ',|"' ' '); do aws ec2 delete-volume
--region <region> --volume-id $x; done
```

With "profile":

```
for x in $(aws ec2 describe-volumes --filters
Name=status,Values=available  --profile <your_profile_name>|grep
VolumeId|awk '{print $2}' | tr ',|"' ' '); do aws ec2 delete-volume
--region <region> --volume-id $x --profile <your_profile_name>;
done
```

REFERENCE:
https://github.com/eon01/AWS-CheatSheet

A **A**

AZURE CLI

RED/BLUE TEAM	RECON/ADMIN	CLOUD

Azure command-line interface (Azure CLI) is an environment to create and manage Azure resources.

Login in CLI

```
az login -u myemail@address.com
```

List accounts

```
az account list
```

Set subscription

```
az account set --subscription "xxx"
```

List all locations

```
az account list-locations
```

List all resource groups

```
az resource list
```