**Complete AWS Security CLI Commands Reference**

**Week 1: AWS Security Services & VPC Architecture**

This comprehensive guide covers ALL CLI commands you'll need for AWS security fundamentals, MFA setup, Identity Center, breakglass scenarios, and secure EC2 deployment in VPC.

**🏗️ AWS Foundation & Account Setup**

**AWS Regions & Availability Zones**

bash

*# List all available AWS regions*

aws ec2 describe-regions --output table

*# List availability zones in current region*

aws ec2 describe-availability-zones --output table

*# List availability zones in specific region*

aws ec2 describe-availability-zones --region us-west-2 --output table

*# Check current configured region*

aws configure get region

*# Set default region*

aws configure set region us-east-1

*# Check which region you're actually using for commands*

aws sts get-caller-identity --region us-east-1

**Why this matters:** Region choice affects latency, compliance, and service availability. Some services are global (IAM) while others are regional (EC2, VPC).

**Account Identity & Verification**

bash

*# WHO AM I? (Most important command)*

aws sts get-caller-identity

*# Check current AWS CLI configuration*

aws configure list

*# Show all configured profiles*

aws configure list-profiles

*# Get account details and limits*

aws iam get-account-summary

*# Check if you're using root account (NEVER do this in production)*

aws iam get-user

*# If this returns error "cannot call GetUser operation" - you might be using root*

**🔐 MFA & Authentication Management**

**MFA Device Management**

bash

*# List MFA devices for current user*

aws iam list-mfa-devices

*# List MFA devices for specific user*

aws iam list-mfa-devices --user-name john.smith

*# Create virtual MFA device*

aws iam create-virtual-mfa-device \

--virtual-mfa-device-name MyMFA \

--outfile qr-code.png \

--bootstrap-method QRCodePNG

*# Enable MFA device (requires 2 consecutive codes)*

aws iam enable-mfa-device \

--user-name john.smith \

--serial-number arn:aws:iam::123456789012:mfa/MyMFA \

--authentication-code1 123456 \

--authentication-code2 789012

*# Disable MFA device*

aws iam deactivate-mfa-device \

--user-name john.smith \

--serial-number arn:aws:iam::123456789012:mfa/MyMFA

*# Delete MFA device*

aws iam delete-virtual-mfa-device \

--serial-number arn:aws:iam::123456789012:mfa/MyMFA

**Why 2 MFA codes?** AWS needs to verify your device generates different codes over time. First code proves it works now, second code (30+ seconds later) proves it's properly synchronized.

**MFA with CLI - Session Token Method**

bash

*# Get session token using MFA (enables CLI access with MFA)*

aws sts get-session-token \

--serial-number arn:aws:iam::123456789012:mfa/john.smith \

--token-code 123456 \

--duration-seconds 3600

*# The above returns temporary credentials - set them as environment variables:*

export AWS\_ACCESS\_KEY\_ID=ASIA...

export AWS\_SECRET\_ACCESS\_KEY=...

export AWS\_SESSION\_TOKEN=...

*# Or create a temporary profile*

aws configure set aws\_access\_key\_id ASIA... --profile mfa-session

aws configure set aws\_secret\_access\_key ... --profile mfa-session

aws configure set aws\_session\_token ... --profile mfa-session

*# Now use the MFA-authenticated session*

aws s3 ls --profile mfa-session

**Role Assumption with MFA**

bash

*# Assume role that requires MFA*

aws sts assume-role \

--role-arn arn:aws:iam::123456789012:role/AdminRole \

--role-session-name AdminSession \

--serial-number arn:aws:iam::123456789012:mfa/john.smith \

--token-code 123456

*# Example: Breakglass role assumption*

aws sts assume-role \

--role-arn arn:aws:iam::123456789012:role/BreakglassRole \

--role-session-name EmergencyAccess \

--serial-number arn:aws:iam::123456789012:mfa/breakglass-user \

--token-code 123456 \

--duration-seconds 3600

**👥 Identity Center (AWS SSO) Commands**

**Identity Center User Management**

bash

*# List all users in Identity Center*

aws identitystore list-users --identity-store-id d-1234567890

*# Get specific user details*

aws identitystore describe-user \

--identity-store-id d-1234567890 \

--user-id 1234-5678-9012

*# List groups*

aws identitystore list-groups --identity-store-id d-1234567890

*# List group memberships for user*

aws identitystore list-group-memberships-for-member \

--identity-store-id d-1234567890 \

--member-id UserId=1234-5678-9012

*# Get Identity Store ID*

aws sso-admin list-instances

**Permission Sets & Account Assignments**

bash

*# List permission sets*

aws sso-admin list-permission-sets --instance-arn arn:aws:sso:::instance/ssoins-1234567890

*# Describe permission set*

aws sso-admin describe-permission-set \

--instance-arn arn:aws:sso:::instance/ssoins-1234567890 \

--permission-set-arn arn:aws:sso:::permissionSet/ssoins-1234567890/ps-1234567890

*# List account assignments*

aws sso-admin list-account-assignments \

--instance-arn arn:aws:sso:::instance/ssoins-1234567890 \

--account-id 123456789012 \

--permission-set-arn arn:aws:sso:::permissionSet/ssoins-1234567890/ps-1234567890

**🚨 Breakglass User Management**

**Creating Breakglass Users**

bash

*# Create breakglass IAM user*

aws iam create-user \

--user-name breakglass-admin \

--tags Key=Purpose,Value=EmergencyAccess Key=Owner,Value=Security

*# Create access keys for breakglass user*

aws iam create-access-key --user-name breakglass-admin

*# Attach admin policy to breakglass user*

aws iam attach-user-policy \

--user-name breakglass-admin \

--policy-arn arn:aws:iam::aws:policy/AdministratorAccess

*# Force MFA for breakglass user (create policy that denies all actions without MFA)*

cat > breakglass-mfa-policy.json << 'EOF'

{

"Version": "2012-10-17",

"Statement": [

{

"Sid": "DenyAllExceptUnlessSignedInWithMFA",

"Effect": "Deny",

"NotAction": [

"iam:CreateVirtualMFADevice",

"iam:EnableMFADevice",

"iam:GetUser",

"iam:ListMFADevices",

"iam:ListVirtualMFADevices",

"iam:ResyncMFADevice",

"sts:GetSessionToken"

],

"Resource": "\*",

"Condition": {

"BoolIfExists": {

"aws:MultiFactorAuthPresent": "false"

}

}

}

]

}

EOF

aws iam create-policy \

--policy-name BreakglassMFARequired \

--policy-document file://breakglass-mfa-policy.json

aws iam attach-user-policy \

--user-name breakglass-admin \

--policy-arn arn:aws:iam::123456789012:policy/BreakglassMFARequired

**Breakglass Monitoring**

bash

*# Monitor breakglass user activity*

aws logs filter-log-events \

--log-group-name CloudTrail/BreakglassAccess \

--filter-pattern "{ ($.userIdentity.type = IAMUser) && ($.userIdentity.userName = breakglass-admin) }"

*# List recent API calls by breakglass user*

aws cloudtrail lookup-events \

--lookup-attributes AttributeKey=Username,AttributeValue=breakglass-admin \

--start-time 2024-01-01 \

--end-time 2024-01-31

**👤 IAM Users, Groups & Roles Management**

**User Management**

bash

*# Create IAM user*

aws iam create-user \

--user-name john.smith \

--tags Key=Department,Value=Engineering

*# Create access keys for user*

aws iam create-access-key --user-name john.smith

*# Set user password (for console access)*

aws iam create-login-profile \

--user-name john.smith \

--password TempPassword123! \

--password-reset-required

*# List all users*

aws iam list-users --output table

*# Get specific user details*

aws iam get-user --user-name john.smith

*# Delete user (cleanup process)*

aws iam delete-login-profile --user-name john.smith

aws iam delete-access-key --user-name john.smith --access-key-id AKIA...

aws iam delete-user --user-name john.smith

**Group Management**

bash

*# Create IAM group*

aws iam create-group --group-name Developers

*# Add user to group*

aws iam add-user-to-group \

--group-name Developers \

--user-name john.smith

*# Attach policy to group*

aws iam attach-group-policy \

--group-name Developers \

--policy-arn arn:aws:iam::aws:policy/PowerUserAccess

*# List groups*

aws iam list-groups

*# List users in group*

aws iam get-group --group-name Developers

*# Remove user from group*

aws iam remove-user-from-group \

--group-name Developers \

--user-name john.smith

**Role Management**

bash

*# Create trust policy for EC2 service role*

cat > ec2-trust-policy.json << 'EOF'

{

"Version": "2012-10-17",

"Statement": [

{

"Effect": "Allow",

"Principal": {

"Service": "ec2.amazonaws.com"

},

"Action": "sts:AssumeRole"

}

]

}

EOF

*# Create IAM role*

aws iam create-role \

--role-name SecureEC2Role \

--assume-role-policy-document file://ec2-trust-policy.json \

--description "Secure role for EC2 instances with minimal S3 access"

*# Create instance profile*

aws iam create-instance-profile --instance-profile-name SecureEC2Profile

*# Add role to instance profile*

aws iam add-role-to-instance-profile \

--instance-profile-name SecureEC2Profile \

--role-name SecureEC2Role

*# Create custom S3 read-only policy*

cat > s3-readonly-policy.json << 'EOF'

{

"Version": "2012-10-17",

"Statement": [

{

"Effect": "Allow",

"Action": [

"s3:GetObject",

"s3:GetObjectVersion",

"s3:ListBucket",

"s3:ListBucketVersions",

"s3:GetBucketLocation",

"s3:GetBucketVersioning"

],

"Resource": [

"arn:aws:s3:::\*",

"arn:aws:s3:::\*/\*"

]

}

]

}

EOF

*# Create and attach custom policy*

aws iam create-policy \

--policy-name S3ReadOnlyCustom \

--policy-document file://s3-readonly-policy.json

aws iam attach-role-policy \

--role-name SecureEC2Role \

--policy-arn arn:aws:iam::123456789012:policy/S3ReadOnlyCustom

*# List roles*

aws iam list-roles --query 'Roles[\*].[RoleName,CreateDate]' --output table

*# Get role details*

aws iam get-role --role-name SecureEC2Role

**🌐 VPC Network Architecture**

**VPC Creation & Configuration**

bash

*# Create custom VPC*

aws ec2 create-vpc \

--cidr-block 10.0.0.0/16 \

--tag-specifications 'ResourceType=vpc,Tags=[{Key=Name,Value=SecureArchitecture-VPC},{Key=Environment,Value=Demo}]'

*# Enable DNS hostnames and resolution*

aws ec2 modify-vpc-attribute --vpc-id vpc-xxxxxxxxx --enable-dns-hostnames

aws ec2 modify-vpc-attribute --vpc-id vpc-xxxxxxxxx --enable-dns-support

*# Create public subnet*

aws ec2 create-subnet \

--vpc-id vpc-xxxxxxxxx \

--cidr-block 10.0.1.0/24 \

--availability-zone us-east-1a \

--tag-specifications 'ResourceType=subnet,Tags=[{Key=Name,Value=PublicSubnet},{Key=Type,Value=Public}]'

*# Create private subnet*

aws ec2 create-subnet \

--vpc-id vpc-xxxxxxxxx \

--cidr-block 10.0.2.0/24 \

--availability-zone us-east-1b \

--tag-specifications 'ResourceType=subnet,Tags=[{Key=Name,Value=PrivateSubnet},{Key=Type,Value=Private}]'

*# Enable auto-assign public IP for public subnet*

aws ec2 modify-subnet-attribute \

--subnet-id subnet-xxxxxxxxx \

--map-public-ip-on-launch

**Internet Gateway & Routing**

bash

*# Create Internet Gateway*

aws ec2 create-internet-gateway \

--tag-specifications 'ResourceType=internet-gateway,Tags=[{Key=Name,Value=SecureArchitecture-IGW}]'

*# Attach Internet Gateway to VPC*

aws ec2 attach-internet-gateway \

--vpc-id vpc-xxxxxxxxx \

--internet-gateway-id igw-xxxxxxxxx

*# Create route table for public subnet*

aws ec2 create-route-table \

--vpc-id vpc-xxxxxxxxx \

--tag-specifications 'ResourceType=route-table,Tags=[{Key=Name,Value=PublicRouteTable}]'

*# Add route to Internet Gateway*

aws ec2 create-route \

--route-table-id rtb-xxxxxxxxx \

--destination-cidr-block 0.0.0.0/0 \

--gateway-id igw-xxxxxxxxx

*# Associate route table with public subnet*

aws ec2 associate-route-table \

--subnet-id subnet-xxxxxxxxx \

--route-table-id rtb-xxxxxxxxx

**VPC Verification Commands**

bash

*# List all VPCs*

aws ec2 describe-vpcs --output table

*# Show VPC details*

aws ec2 describe-vpcs --vpc-ids vpc-xxxxxxxxx

*# List subnets in VPC*

aws ec2 describe-subnets \

--filters "Name=vpc-id,Values=vpc-xxxxxxxxx" \

--query 'Subnets[\*].[SubnetId,CidrBlock,AvailabilityZone,Tags[?Key==`Name`].Value|[0]]' \

--output table

*# Show routing tables*

aws ec2 describe-route-tables \

--filters "Name=vpc-id,Values=vpc-xxxxxxxxx" \

--output table

*# Check Internet Gateway*

aws ec2 describe-internet-gateways \

--filters "Name=attachment.vpc-id,Values=vpc-xxxxxxxxx"

**🔒 Security Groups & Network ACLs**

**Security Group Management**

bash

*# Get your current public IP*

curl -s https://checkip.amazonaws.com

*# Create security group*

aws ec2 create-security-group \

--group-name SecureWebTier \

--description "Secure access for web tier instances" \

--vpc-id vpc-xxxxxxxxx

*# Add SSH access rule (your IP only)*

aws ec2 authorize-security-group-ingress \

--group-id sg-xxxxxxxxx \

--protocol tcp \

--port 22 \

--cidr $(curl -s https://checkip.amazonaws.com)/32

*# Add HTTP access (if needed)*

aws ec2 authorize-security-group-ingress \

--group-id sg-xxxxxxxxx \

--protocol tcp \

--port 80 \

--cidr 0.0.0.0/0

*# Add HTTPS access (if needed)*

aws ec2 authorize-security-group-ingress \

--group-id sg-xxxxxxxxx \

--protocol tcp \

--port 443 \

--cidr 0.0.0.0/0

*# Remove rule (if needed)*

aws ec2 revoke-security-group-ingress \

--group-id sg-xxxxxxxxx \

--protocol tcp \

--port 80 \

--cidr 0.0.0.0/0

*# View security group rules*

aws ec2 describe-security-groups --group-ids sg-xxxxxxxxx

**Security Audit Commands**

bash

*# Find security groups with wide-open SSH access (DANGEROUS)*

aws ec2 describe-security-groups \

--query 'SecurityGroups[?IpPermissions[?IpRanges[?CidrIp==`0.0.0.0/0`] && FromPort==`22`]].[GroupId,GroupName]' \

--output table

*# Find security groups with no rules*

aws ec2 describe-security-groups \

--query 'SecurityGroups[?length(IpPermissions)==`0`].[GroupId,GroupName]' \

--output table

*# Find unused security groups*

aws ec2 describe-security-groups \

--query 'SecurityGroups[?length(IpPermissions)==`0` && length(IpPermissionsEgress)==`1`].[GroupId,GroupName]' \

--output table

**Network ACL Management**

bash

*# Create custom Network ACL*

aws ec2 create-network-acl \

--vpc-id vpc-xxxxxxxxx \

--tag-specifications 'ResourceType=network-acl,Tags=[{Key=Name,Value=RestrictiveNACL}]'

*# Add inbound rule (SSH from your IP)*

aws ec2 create-network-acl-entry \

--network-acl-id acl-xxxxxxxxx \

--rule-number 100 \

--protocol tcp \

--port-range From=22,To=22 \

--cidr-block $(curl -s https://checkip.amazonaws.com)/32 \

--rule-action allow

*# Add outbound rule (return traffic)*

aws ec2 create-network-acl-entry \

--network-acl-id acl-xxxxxxxxx \

--rule-number 100 \

--protocol tcp \

--port-range From=1024,To=65535 \

--cidr-block 0.0.0.0/0 \

--rule-action allow \

--egress

*# Associate NACL with subnet*

aws ec2 replace-network-acl-association \

--association-id aclassoc-xxxxxxxxx \

--network-acl-id acl-xxxxxxxxx

*# View NACL rules*

aws ec2 describe-network-acls --network-acl-ids acl-xxxxxxxxx

**💻 EC2 Instance Management**

**SSH Key Pair Management**

bash

*# Create SSH key pair*

aws ec2 create-key-pair \

--key-name SecureArchitecture-Key \

--key-type rsa \

--key-format pem \

--query 'KeyMaterial' \

--output text > SecureArchitecture-Key.pem

*# Set proper permissions for private key*

chmod 400 SecureArchitecture-Key.pem

*# Move to secure location*

mkdir -p ~/.ssh

mv SecureArchitecture-Key.pem ~/.ssh/

*# List existing key pairs*

aws ec2 describe-key-pairs

*# Import existing public key*

aws ec2 import-key-pair \

--key-name MyExistingKey \

--public-key-material fileb://~/.ssh/id\_rsa.pub

*# Delete key pair*

aws ec2 delete-key-pair --key-name SecureArchitecture-Key

**AMI Selection**

bash

*# Find latest Amazon Linux 2 AMI*

aws ec2 describe-images \

--owners amazon \

--filters "Name=name,Values=amzn2-ami-hvm-\*-x86\_64-gp2" \

"Name=state,Values=available" \

--query 'Images | sort\_by(@, &CreationDate) | [-1].ImageId' \

--output text

*# Find latest Ubuntu 20.04 LTS AMI*

aws ec2 describe-images \

--owners 099720109477 \

--filters "Name=name,Values=ubuntu/images/hvm-ssd/ubuntu-focal-20.04-amd64-server-\*" \

"Name=state,Values=available" \

--query 'Images | sort\_by(@, &CreationDate) | [-1].ImageId' \

--output text

*# Find latest Windows Server 2019 AMI*

aws ec2 describe-images \

--owners amazon \

--filters "Name=name,Values=Windows\_Server-2019-English-Full-Base-\*" \

"Name=state,Values=available" \

--query 'Images | sort\_by(@, &CreationDate) | [-1].ImageId' \

--output text

**EC2 Instance Launch**

bash

*# Launch secure EC2 instance*

aws ec2 run-instances \

--image-id ami-xxxxxxxxx \

--instance-type t2.micro \

--key-name SecureArchitecture-Key \

--security-group-ids sg-xxxxxxxxx \

--subnet-id subnet-xxxxxxxxx \

--iam-instance-profile Name=SecureEC2Profile \

--associate-public-ip-address \

--tag-specifications 'ResourceType=instance,Tags=[{Key=Name,Value=SecureWebServer},{Key=Environment,Value=Demo}]' \

--user-data file://user-data.sh

*# Example user-data script*

cat > user-data.sh << 'EOF'

#!/bin/bash

yum update -y

yum install -y httpd

systemctl start httpd

systemctl enable httpd

echo "<h1>Secure Web Server</h1>" > /var/www/html/index.html

EOF

*# Wait for instance to be running*

aws ec2 wait instance-running --instance-ids i-xxxxxxxxx

*# Get instance details*

aws ec2 describe-instances --instance-ids i-xxxxxxxxx

*# Get instance public IP*

aws ec2 describe-instances \

--instance-ids i-xxxxxxxxx \

--query 'Reservations[0].Instances[0].PublicIpAddress' \

--output text

**Instance Connection**

bash

*# Connect via SSH*

ssh -i ~/.ssh/SecureArchitecture-Key.pem ec2-user@$(aws ec2 describe-instances \

--instance-ids i-xxxxxxxxx \

--query 'Reservations[0].Instances[0].PublicIpAddress' \

--output text)

*# Connect with verbose output (for troubleshooting)*

ssh -v -i ~/.ssh/SecureArchitecture-Key.pem ec2-user@PUBLIC-IP

*# Test connection without connecting*

ssh -o BatchMode=yes -o ConnectTimeout=5 \

-i ~/.ssh/SecureArchitecture-Key.pem ec2-user@PUBLIC-IP echo ok 2>&1

**🔍 Instance Metadata & Credential Verification**

**IMDSv2 Commands (Modern Security)**

bash

*# Get metadata token (required for IMDSv2)*

TOKEN=$(curl -X PUT "http://169.254.169.254/latest/api/token" \

-H "X-aws-ec2-metadata-token-ttl-seconds: 21600" -s)

*# Get instance metadata*

curl -H "X-aws-ec2-metadata-token: $TOKEN" -s \

http://169.254.169.254/latest/meta-data/

*# Get instance identity document*

curl -H "X-aws-ec2-metadata-token: $TOKEN" -s \

http://169.254.169.254/latest/dynamic/instance-identity/document

*# Get IAM role name*

curl -H "X-aws-ec2-metadata-token: $TOKEN" -s \

http://169.254.169.254/latest/meta-data/iam/security-credentials/

*# Get temporary credentials*

curl -H "X-aws-ec2-metadata-token: $TOKEN" -s \

http://169.254.169.254/latest/meta-data/iam/security-credentials/SecureEC2Role

*# Get instance tags*

curl -H "X-aws-ec2-metadata-token: $TOKEN" -s \

http://169.254.169.254/latest/meta-data/tags/instance/

**Security Validation Tests**

bash

*# FROM WITHIN EC2 INSTANCE - Test what should work*

aws s3 ls *# ✅ Should succeed*

aws s3api list-buckets *# ✅ Should succeed*

aws s3 cp s3://bucket/file.txt . *# ✅ Should succeed*

*# FROM WITHIN EC2 INSTANCE - Test what should fail (proving security works)*

aws ec2 describe-instances *# ❌ Should fail*

aws iam list-users *# ❌ Should fail*

aws ec2 run-instances --image-id ami-12345 --instance-type t2.micro *# ❌ Should fail*

aws s3 cp file.txt s3://bucket/ *# ❌ Should fail (read-only policy)*

*# Check that no hardcoded credentials exist*

ls -la ~/.aws/ *# Should show no files*

env | grep -i aws *# Should show no AWS variables*

cat ~/.bash\_history | grep -i aws *# Check for accidentally stored keys*

**📊 Monitoring & Auditing**

**Resource Inventory**

bash

*# List all running instances*

aws ec2 describe-instances \

--filters "Name=instance-state-name,Values=running" \

--query 'Reservations[\*].Instances[\*].[InstanceId,InstanceType,PublicIpAddress,Tags[?Key==`Name`].Value|[0]]' \

--output table

*# List all VPCs and their subnets*

aws ec2 describe-vpcs \

--query 'Vpcs[\*].[VpcId,CidrBlock,IsDefault,Tags[?Key==`Name`].Value|[0]]' \

--output table

*# List all security groups with their rules*

aws ec2 describe-security-groups \

--query 'SecurityGroups[\*].[GroupId,GroupName,Description]' \

--output table

*# List all IAM roles*

aws iam list-roles \

--query 'Roles[\*].[RoleName,CreateDate,Description]' \

--output table

**Security Compliance Checks**

bash

*# Find instances without IAM roles*

aws ec2 describe-instances \

--query 'Reservations[\*].Instances[?!IamInstanceProfile].[InstanceId,Tags[?Key==`Name`].Value|[0]]' \

--output table

*# Find unencrypted EBS volumes*

aws ec2 describe-volumes \

--query 'Volumes[?Encrypted==`false`].[VolumeId,Size,State]' \

--output table

*# Check for public S3 buckets*

aws s3api list-buckets --query 'Buckets[\*].Name' --output text | \

while read bucket; do

echo "Checking $bucket"

aws s3api get-bucket-acl --bucket $bucket 2>/dev/null || echo "Access denied"

done

*# Find IAM users with console access but no MFA*

aws iam get-account-summary | grep -A 5 -B 5 MFA

**🧹 Cleanup Commands**

**EC2 Cleanup**

bash

*# Terminate instances*

aws ec2 terminate-instances --instance-ids i-xxxxxxxxx

*# Wait for termination*

aws ec2 wait instance-terminated --instance-ids i-xxxxxxxxx

*# Delete security groups (after instances are terminated)*

aws ec2 delete-security-group --group-id sg-xxxxxxxxx

*# Delete key pairs*

aws ec2 delete-key-pair --key-name SecureArchitecture-Key

rm ~/.ssh/SecureArchitecture-Key.pem

**VPC Cleanup**

bash

*# Delete route table associations first*

aws ec2 disassociate-route-table --association-id rtbassoc-xxxxxxxxx

*# Delete custom route tables (not main)*

aws ec2 delete-route-table --route-table-id rtb-xxxxxxxxx

*# Detach and delete Internet Gateway*

aws ec2 detach-internet-gateway \

--internet-gateway-id igw-xxxxxxxxx \

--vpc-id vpc-xxxxxxxxx

aws ec2 delete-internet-gateway --internet-gateway-id igw-xxxxxxxxx

*# Delete subnets*

aws ec2 delete-subnet --subnet-id subnet-xxxxxxxxx

*# Delete VPC*

aws ec2 delete-vpc --vpc-id vpc-xxxxxxxxx

**IAM Cleanup**

bash

*# Remove role from instance profile*

aws iam remove-role-from-instance-profile \

--instance-profile-name SecureEC2Profile \

--role-name SecureEC2Role

*# Delete instance profile*

aws iam delete-instance-profile --instance-profile-name SecureEC2Profile

*# Detach policies from role*

aws iam detach-role-policy \

--role-name SecureEC2Role \

--policy-arn arn:aws:iam::123456789012:policy/S3ReadOnlyCustom

*# Delete custom policies*

aws iam delete-policy --policy-arn arn:aws:iam::123456789012:policy/S3ReadOnlyCustom

*# Delete role*

aws iam delete-role --role-name SecureEC2Role

*# Allocate Elastic IP for NAT Gateway*

aws ec2 allocate-address --domain vpc

*# Create NAT Gateway in public subnet*

aws ec2 create-nat-gateway \

--subnet-id subnet-public-xxxxxxxxx \

--allocation-id eipalloc-xxxxxxxxx \

--tag-specifications 'ResourceType=nat-gateway,Tags=[{Key=Name,Value=SecureArchitecture-NAT}]'

*# Create route table for private subnet*

aws ec2 create-route-table \

--vpc-id vpc-xxxxxxxxx \

--tag-specifications 'ResourceType=route-table,Tags=[{Key=Name,Value=PrivateRouteTable}]'

*# Add route to NAT Gateway for private subnet*

aws ec2 create-route \

--route-table-id rtb-private-xxxxxxxxx \

--destination-cidr-block 0.0.0.0/0 \

--nat-gateway-id nat-xxxxxxxxx

*# Associate private route table with private subnet*

aws ec2 associate-route-table \

--subnet-id subnet-private-xxxxxxxxx \

--route-table-id rtb-private-xxxxxxxxx

*# Apply bucket policy*

aws s3api put-bucket-policy \

--bucket company-cloudtrail-logs-$(date +%s) \

--policy file://cloudtrail-bucket-policy.json

aws iam attach-role-policy \

--role-name AWSConfigRole \

--policy-arn arn:aws:iam::aws:policy/service-role/ConfigRole

*# Create Config delivery channel*

aws configservice put-delivery-channel \

--delivery-channel name=default,s3BucketName=company-config-bucket

*# Create Config configuration recorder*

aws configservice put-configuration-recorder \

--configuration-recorder name=default,roleARN=arn:aws:iam::123456789012:role/AWSConfigRole

*# Start Config recording*

aws configservice start-configuration-recorder --configuration-recorder-name default

**🚨 Emergency & Incident Response Commands**

**Emergency Access Scenarios**

bash

*# Emergency: Identity Center is down, need breakglass access*

aws sts get-session-token \

--serial-number arn:aws:iam::123456789012:mfa/breakglass-admin \

--token-code 123456 \

--duration-seconds 3600

*# Emergency: Assume breakglass role with MFA*

aws sts assume-role \

--role-arn arn:aws:iam::123456789012:role/EmergencyAdminRole \

--role-session-name IncidentResponse \

--serial-number arn:aws:iam::123456789012:mfa/breakglass-admin \

--token-code 123456

*# Emergency: List all admin users (incident response)*

aws iam list-users \

--query 'Users[?contains(AttachedManagedPolicies[].PolicyArn, `Administrator`)].[UserName,CreateDate]'

*# Emergency: Disable a compromised user*

aws iam put-user-policy \

--user-name compromised-user \

--policy-name DenyAllAccess \

--policy-document '{

"Version": "2012-10-17",

"Statement": [

{

"Effect": "Deny",

"Action": "\*",

"Resource": "\*"

}

]

}'

*# Emergency: Rotate access keys for all users*

aws iam list-users --query 'Users[].UserName' --output text | \

while read username; do

echo "Processing $username"

aws iam list-access-keys --user-name $username

done

**Security Incident Investigation**

bash

*# Find recent failed login attempts*

aws logs filter-log-events \

--log-group-name CloudTrail/ConsoleLogin \

--start-time $(date -d '1 hour ago' +%s)000 \

--filter-pattern '{ ($.eventName = ConsoleLogin) && ($.responseElements.ConsoleLogin = Failure) }'

*# Find recent privilege escalation attempts*

aws logs filter-log-events \

--log-group-name CloudTrail/APIActivity \

--start-time $(date -d '1 hour ago' +%s)000 \

--filter-pattern '{ ($.eventName = AttachUserPolicy) || ($.eventName = AttachRolePolicy) || ($.eventName = CreateRole) }'

*# Find instances launched in last 24 hours*

aws ec2 describe-instances \

--query 'Reservations[\*].Instances[?LaunchTime>=`2024-01-01T00:00:00.000Z`].[InstanceId,LaunchTime,Tags[?Key==`Name`].Value|[0]]'

*# Check for unusual API activity*

aws cloudtrail lookup-events \

--lookup-attributes AttributeKey=EventName,AttributeValue=RunInstances \

--start-time 2024-01-01 \

--end-time 2024-01-31

**📋 Security Best Practices Validation**

**Password and Key Management**

bash

*# Check password policy*

aws iam get-account-password-policy

*# Set strong password policy*

aws iam update-account-password-policy \

--minimum-password-length 14 \

--require-symbols \

--require-numbers \

--require-uppercase-characters \

--require-lowercase-characters \

--allow-users-to-change-password \

--max-password-age 90 \

--password-reuse-prevention 12

*# List old access keys (security risk)*

aws iam list-users --query 'Users[].UserName' --output text | \

while read username; do

aws iam list-access-keys --user-name $username \

--query 'AccessKeyMetadata[?Age>`90`].[UserName,AccessKeyId,CreateDate]'

done

*# Find unused access keys*

aws iam generate-credential-report

aws iam get-credential-report --query 'Content' --output text | base64 -d > credential-report.csv

**Encryption Validation**

bash

*# Check EBS encryption by default*

aws ec2 get-ebs-encryption-by-default

*# Enable EBS encryption by default*

aws ec2 enable-ebs-encryption-by-default

*# Find unencrypted snapshots*

aws ec2 describe-snapshots \

--owner-ids self \

--query 'Snapshots[?Encrypted==`false`].[SnapshotId,Description,StartTime]'

*# Check S3 bucket encryption*

aws s3api list-buckets --query 'Buckets[].Name' --output text | \

while read bucket; do

echo "Checking encryption for $bucket"

aws s3api get-bucket-encryption --bucket $bucket 2>/dev/null || echo "Not encrypted"

done

**Network Security Validation**

bash

*# Check for overly permissive security groups*

aws ec2 describe-security-groups \

--query 'SecurityGroups[?IpPermissions[?IpRanges[?CidrIp==`0.0.0.0/0`] && (FromPort<=`22` && ToPort>=`22`)]].[GroupId,GroupName]'

*# Find security groups allowing all traffic*

aws ec2 describe-security-groups \

--query 'SecurityGroups[?IpPermissions[?IpRanges[?CidrIp==`0.0.0.0/0`] && IpProtocol==`-1`]].[GroupId,GroupName]'

*# Check default VPC usage (should be avoided)*

aws ec2 describe-instances \

--query 'Reservations[\*].Instances[?VpcId==`vpc-default`].[InstanceId,VpcId]'

*# Validate private subnet isolation*

aws ec2 describe-route-tables \

--query 'RouteTables[?Routes[?DestinationCidrBlock==`0.0.0.0/0` && GatewayId!=null && starts\_with(GatewayId, `igw-`)]]'

**🎯 Performance & Cost Optimization Commands**

**Resource Utilization**

bash

*# Find unused Elastic IPs*

aws ec2 describe-addresses \

--query 'Addresses[?AssociationId==null].[PublicIp,AllocationId]'

*# Find unattached EBS volumes*

aws ec2 describe-volumes \

--query 'Volumes[?State==`available`].[VolumeId,Size,CreateTime]'

*# Find old snapshots (potential cost optimization)*

aws ec2 describe-snapshots \

--owner-ids self \

--query 'Snapshots[?StartTime<=`2023-01-01`].[SnapshotId,Description,StartTime]'

*# Check instance types and utilization*

aws ec2 describe-instances \

--query 'Reservations[\*].Instances[\*].[InstanceId,InstanceType,State.Name,LaunchTime]' \

--output table

**Cost Monitoring**

bash

*# Get billing information (requires billing permissions)*

aws ce get-cost-and-usage \

--time-period Start=2024-01-01,End=2024-01-31 \

--granularity MONTHLY \

--metrics BlendedCost \

--group-by Type=DIMENSION,Key=SERVICE

*# List Reserved Instance recommendations*

aws ce get-reservation-purchase-recommendation \

--service EC2-Instance

**🔍 Advanced Troubleshooting Commands**

**Network Connectivity Issues**

bash

*# Test VPC Reachability Analyzer (if available)*

aws ec2 create-network-insights-path \

--source i-1234567890abcdef0 \

--destination i-0987654321fedcba0 \

--protocol tcp \

--destination-port 22

*# Start network insights analysis*

aws ec2 start-network-insights-analysis \

--network-insights-path-id nip-12345678

*# Check route propagation*

aws ec2 describe-route-tables \

--query 'RouteTables[\*].[RouteTableId,Routes[\*].[DestinationCidrBlock,GatewayId,State]]'

*# Validate DNS resolution*

aws ec2 describe-vpc-attribute \

--vpc-id vpc-xxxxxxxxx \

--attribute enableDnsHostnames

aws ec2 describe-vpc-attribute \

--vpc-id vpc-xxxxxxxxx \

--attribute enableDnsSupport

**Permission Troubleshooting**

bash

*# Simulate policy evaluation*

aws iam simulate-principal-policy \

--policy-source-arn arn:aws:iam::123456789012:user/testuser \

--action-names s3:GetObject \

--resource-arns arn:aws:s3:::testbucket/testkey

*# Check effective permissions for user*

aws iam get-user-policy --user-name testuser --policy-name TestPolicy

*# List all policies attached to user*

aws iam list-attached-user-policies --user-name testuser

aws iam list-user-policies --user-name testuser

*# Check resource-based policies*

aws s3api get-bucket-policy --bucket testbucket

**Instance Troubleshooting**

bash

*# Get system logs*

aws ec2 get-console-output --instance-id i-xxxxxxxxx

*# Check instance health*

aws ec2 describe-instance-status --instance-id i-xxxxxxxxx

*# Get instance screenshot*

aws ec2 get-console-screenshot --instance-id i-xxxxxxxxx

*# Check user data*

aws ec2 describe-instance-attribute \

--instance-id i-xxxxxxxxx \

--attribute userData \

--query 'UserData.Value' \

--output text | base64 -d

**📖 Documentation Templates**

**Security Runbook Template**

markdown

# AWS Security Incident Response Runbook

## Immediate Actions (0-15 minutes)

1. Identify compromised resource: `aws sts get-caller-identity`

2. Disable compromised user: `aws iam put-user-policy --user-name USER --policy-name DenyAll`

3. Rotate credentials: `aws iam create-access-key --user-name USER`

4. Enable CloudTrail logging: `aws cloudtrail start-logging --name TRAIL`

## Investigation (15-60 minutes)

1. Review recent activity: `aws cloudtrail lookup-events --start-time TIME`

2. Check failed logins: Review CloudWatch logs

3. Validate current permissions: `aws iam simulate-principal-policy`

4. Inventory resources: `aws ec2 describe-instances`

**Daily Practice Commands**

bash

*# Morning check routine*

aws sts get-caller-identity

aws ec2 describe-instances --query 'Reservations[\*].Instances[\*].[InstanceId,State.Name]'

aws s3 ls

aws iam list-mfa-devices

*# Security audit routine*

aws ec2 describe-security-groups --query 'SecurityGroups[?IpPermissions[?IpRanges[?CidrIp==`0.0.0.0/0`]]].[GroupId,GroupName]'

aws iam list-users --query 'Users[\*].[UserName,CreateDate,PasswordLastUsed]'

aws ec2 describe-instances --query 'Reservations[\*].Instances[?!IamInstanceProfile].[InstanceId]'

**⚡ Quick Reference Card**

**Most Used Commands**

| **Purpose** | **Command** |
| --- | --- |
| **Who am I?** | aws sts get-caller-identity |
| **Get my IP** | curl -s https://checkip.amazonaws.com |
| **List instances** | aws ec2 describe-instances --query 'Reservations[\*].Instances[\*].[InstanceId,State.Name,PublicIpAddress]' --output table |
| **Test S3 access** | aws s3 ls |
| **Check MFA** | aws iam list-mfa-devices |
| **Get metadata token** | TOKEN=$(curl -X PUT "http://169.254.169.254/latest/api/token" -H "X-aws-ec2-metadata-token-ttl-seconds: 21600" -s) |
| **Get instance role** | curl -H "X-aws-ec2-metadata-token: $TOKEN" -s http://169.254.169.254/latest/meta-data/iam/security-credentials/ |

**Essential Variables**

bash

*# Set these for easier command usage*

export AWS\_REGION=us-east-1

export VPC\_ID=vpc-xxxxxxxxx

export SUBNET\_ID=subnet-xxxxxxxxx

export SG\_ID=sg-xxxxxxxxx

export INSTANCE\_ID=i-xxxxxxxxx

export KEY\_NAME=SecureArchitecture-Key