AWS CLI Cheat Sheet - Getting Started

Setting up your environment (In my case, a Fedora host):

Command	Description
pip install awscli	Install AWS CLI
pip install –upgrade awscli	Upgrade AWS CLI
pip show awscli	List the version of AWS CLI
aws configure	Set up your AWS CLI environment, file output, default region, access keys
aws configure list	List your AWS CLI configuration
complete -C aws_completer aws	Set up AWS CLI auto-complete
ls ~/.aws	Location of AWS CLI credentials and configuration
export AWS_ACCESS_KEY=	Export your AWS access key
export AWS_SECRET_KEY=	Export your AWS secret

Getting Help:

Command	Description
aws help	Get general, top level help, check out “Available Services” for more subcommands to run help on.
aws ec2 help	ec2 subcommand help, check out “Available Commands” for more subommands to run help on.
aws route53 help	route53 subcommand help, check out “Available Commands” for more subommands to run help on.
aws ec2 describe-instances help	ec2 describe-instances subcommand help, check out the “Options, Filters, and Examples”.

Working with Instances:

List all the instances for the zone that you are configured to view by default:
aws ec2 describe-instances
Launch an instance:
aws ec2 run-instances --image-id <image-id> --instance-type t2.small --subnet-id <subnet-id> --block-device-mappings file://<test.json> --security-group-ids <sg-id> –key-name "<key>"
Get an instance-id:
aws ec2 describe-instances --query 'Reservations[].Instances[].[InstanceId]' --filters "Name=instance-state-name,Values=running" "Name=tag:Name,Values=<name.of.instance.*>" --output text
Add tags to an instance:
aws ec2 create-tags --resources <instance-id> --tags Key=Name,Value=<value of tag>
Launch an instance, attach an EBS volume, assign other attributes, and tag it at the same time with jq:
aws ec2 create-tags --resources `aws ec2 run-instances --image-id <image-id> --instance-type t2.small --subnet-id <subnet-id> --block-device-mappings=VirtualName=master,DeviceName=/dev/xvdb,Ebs={VolumeSize=25,DeleteOnTermination=True,VolumeType="gp2"} --security-group-ids <sg-id> --key-name "<key>" | jq -r ".Instances[0].InstanceId"` --tags "Key=Name,Value=<value>"

Cheat sheet by Scott Collier @collier_s, May 2016, v1
http://www.colliernotes.com

AWS CLI Cheat Sheet - Getting Started

Working with VPCs:

Create a VPC:
aws ec2 create-vpc --cidr-block 192.168.0.0/16
Get the VPC ID:
aws ec2 describe-vpcs --query Vpcs[].VpcId[] --filters Name=cidr,Values=192.168.0.0/16 --output text
Tag the VPC:
aws ec2 create-tags --resources <VPC-ID> --tags Key=Name,Value=<VPC_Name>
Create a subnet:
aws ec2 create-subnet --vpc-id <VPC_ID> --cidr-block <Subnet_CIDR> --availability-zone <AZ>
Query for the subnet ID:
aws ec2 describe-subnets --query Subnets[].SubnetId[] --filters Name=cidr,Values=<Subnet_ID> --output text
Tag a subnet:
aws ec2 create-tags --resources <Subnet_ID> --tags Key=Name,Value=<Subnet_Tag>
Create an Internet Gateway:
aws ec2 create-internet-gateway
Tag the Internet Gateway:
aws ec2 create-tags –resources <IG_ID> --tags Key=Name,Value=<IG_Tag>
Attach an Internet Gateway:
aws ec2 attach-internet-gateway --internet-gateway-id <IG_ID> --vpc-id <VPC_ID>
Query for the Route Table ID:
aws ec2 describe-route-tables --query RouteTables[].RouteTableId --filters Name=route.destination-cidr-block,Values=<CIDR> --output text
Create a route:
aws ec2 create-route --route-table-id <Route_Table_ID> --destination-cidr-block 0.0.0.0/0 --gateway-id <IG_ID>
Enable DNS Hostnames:
aws ec2 modify-vpc-attribute --vpc-id <VPC_ID> --enable-dns-hostnames

Working with SSH:

Create a SSH key pair, upload it, and write it out:
aws ec2 create-key-pair --key-name <Key_Name> --query 'KeyMaterial' --output text > <Key_Name.pem> > <Key_Name.pem>

Working with DHCP:

Create a DHCP options set:
aws ec2 create-dhcp-options --dhcp-configuration "Key=domain-name,Values=<DHCP_Opts>" "Key=domain-name-servers,Values=AmazonProvidedDNS" --query 'DhcpOptions.DhcpOptionsId' --output text
Associate the DHCP option set to a VPC:
aws ec2 associate-dhcp-options --dhcp-options-id <DHCP_Opt_ID> --vpc-id <VPC_ID>

Cheat sheet by Scott Collier @collier_s, May 2016, v1
http://www.colliernotes.com

AWS CLI Cheat Sheet - Getting Started

Working with IAM:

Creating an Policy:
aws iam create-policy --policy-name <IAM_Policy_Name> --policy-document file://<Policy_Doc.json> --output text
Create a role:
aws iam create-role --role-name <IAM_Role_Name> --assume-role-policy-document file://<Policy_doc.json>
Attach a role to a policy:
aws iam attach-role-policy --role-name <IAM_Role_Name> --policy-arn <Policy_Amazon_Resource_Name>
Create an instance profile:
aws iam create-instance-profile --instance-profile-name <Instance_Profile>
Add a role to an instance profile:
aws iam add-role-to-instance-profile --instance-profile-name <Instance_Profile> --role-name <IAM_Role_Name>

Working with Route53:

Create a forward DNS zone, with a unique caller reference (which is required):
aws route53 create-hosted-zone --name <Zone_Name> --caller-reference $(date "+%F-%H:%M:%S") --hosted-zone-config Comment="Comment_For_Zone"
Create a Route53 A record, Where <A_Record.json> has the information for the record:
aws route53 change-resource-record-sets --hosted-zone-id <Hosted_Zone> --change-batch file://<A_Record.json>

Working with Elastic Load Balancers:

Query instances to add to the Elastic Load Balancer:
aws ec2 describe-instances --query 'Reservations[].Instances[].[InstanceId]' --filters "Name=instance-state-name,Values=running" "Name=tag:Name,Values=<Names_Of_Nodes"
Create an Elastic Load Balancer:
aws elb create-load-balancer --load-balancer-name <ELB_Name> --listeners "Protocol=TCP,LoadBalancerPort=80,InstancePort=80" --subnets "<Subnet>" "<Subnet>"
Add the Elastic Load Balancer to a security group:
aws elb apply-security-groups-to-load-balancer --load-balancer-name <ELB_Name> --security-groups <ELB_Security_Group>
Create a health check for the Elastic Load Balancer:
aws elb configure-health-check --load-balancer-name <ELB_Name> --health-check Target=tcp:443,Interval=5,UnhealthyThreshold=2,HealthyThreshold=2,Timeout=2
Add nodes to an Elastic Load Balancer:
aws elb register-instances-with-load-balancer --load-balancer-name <ELB_Name> --instances <Instance_1> <Instance_2> <Instance_3>

Working with Security Groups:

Create a Security Group:
aws ec2 create-security-group --group-name <Security_Group_Name --description "Security Group Comment" --vpc-id <VPC_ID>
Add ingress traffic:
aws ec2 authorize-security-group-ingress --group-id <Remote_Sec_Grp> --protocol tcp --port 80 --source-group <Source_Sec_Grp>

Cheat sheet by Scott Collier @collier_s, May 2016, v1
http://www.colliernotes.com