**AWS ARCHITECT ASSOCIATE PRACTICE SCENARIOS.**

* **EC2 – Elastic Compute Cloud**

1. Create instance in N.Virginia (us-east-1) region using Free tier AMI with t2.micro instance type of windows operating system.
2. Create and Attach Elastic IP to the instance created above.
3. Allow port 3389 for your machine IP in security group for RDP access.
4. Install IIS on it and allow 80, 443 in security group for site access using Elastic IP.
5. Add tags to windows instance with key values Name, product, Role, environment.
6. Observe/get the instance screenshot.
7. Change the termination protection of the instance created.
8. Change the security group of your instance created.
9. Attach an additional private IP to the instance created.
10. Create additional new volume with GP2 type and attach to instance and make it online on server.
11. Create a snapshot of root volume for the windows instance created.
12. Extend additional new volume created online.
13. Create a snapshot of new volume created.
14. Create a volume by using snapshot of new volume.
15. Create AMI with above created windows instance with additional 5 GiB volume.
16. Delete the snapshot of Root volume created above.
17. Detach the additional volume created above.
18. Change instance type from t2.micro to t3.medium. (Hints: stop instance change instance type and start instance). Experience the compute resource changes.
19. Detach and release the Elastic IP address attached to above instance.
20. Stop the windows instance.
21. Terminate the windows instance. (Hint: check instance termination protection is enabled or disabled)
22. Deregister AMI created by above windows instance.
23. Create Linux instance in us-east-1a region with a1.medium instance type.
24. Create Linux Instance in us-east-1b region with t2.micro instance type and install apache server using user data.
25. Attach an additional volume to Linux instance and mount it as /tomcat on server.
26. Extend the additional volume and remount /tomcat on Linux server created.

* **IAM - Identity and Access Management:**

1. Create an IAM User with console password.
2. Set-up MFA using google authenticator (downloaded on Mobile) for the user created above and experience the IAM user console login with MFA.
3. Create an IAM User with Programmatic access for the AWS API, CLI, SDK, and other development tools. Experience the download of Access and secret keys of the same user.
4. Activate and Inactivate keys created for above programmatic IAM user.
5. Create a simple custom IAM policy with visual editor to allow list and read actions of EC2 service.
6. Create an IAM role with trusted entity of EC2 for instance profile and attach IAM Policy created above.
7. Create an IAM Group and allow/attach “AdministratorAccess” AWS managed policy to it.
8. Attach IAM Group to the IAM User created above (point 1).
9. Experience IAM password policy on Account settings tab.
10. Download Credential report and experience column header and values inside report.

* **Virtual Private Cloud (VPC)**

1. Create different scenarios of VPC’s using “Launch VPC Wizard” from VPC Dashboard and observe respective resources created i.e. Subnets, internet gateways, NAT Gateway etc.

2. Create VPC with CIDR of 10.176.0.0/16 and slice it to 2 subnets with /24 CIDR.

3. Create 2 Route tables and associate each to above created subnets

4. Identify/Differentiate public subnet and private subnets using Route table.

5. Create Internet Gateway (IGW) and convert private subnet to public subnet and perform vice versa by creating NAT gateway i.e. convert public subnet to private subnet.

6. Observe Elastic IP creation in process of NAT Gateway creation process.

7. Create VPC Peering connection between 2 regions or within the region.

8. Create Security group in specific VPC and allow traffic ports 22, 80, 443, 3389 from your machine IP address only.

9. Create a “Network ACL” for additional security and associate to public subnet created above.

10. Modify “inbound rule” on “Network ACL” created to allow only port 80 from anywhere.

11. Create Customer Gateway (CGW) for VPN Connection creation.

12. Create Virtual private gateway (VGW) to assign to VPN connection.

13. Create VPN connection by attaching above created CGW and VGW’s.

14. Delete the VPC and observe the related resource deletion.

* **S3 (Simple Storage Service)**

1. Create a bucket in US N.virginia Region (us-east-1) and observe bucket name is globally unique.
2. Try to create folder inside the bucket created and upload a file from your local machine.
3. Try to create folder (prefix) with encryption enabled and explore “default encryption” option which is under “properties” section.
4. Experience the download, rename, get size and finally delete options from Actions menu of an object.
5. Enable versioning of the bucket and experience by differentiating of a same object uploaded to same bucker and folder along with Versions Hide and Show options.
6. Try to add lifecycle rule for objects under Management section to move objects to Glacier.
7. Observe the storage class of an objects and understand the difference of different storage classes (i.e. Standard, Glacier).
8. Experience cross region replication by adding new rule.
9. Experience a simple .txt object access using the link provided under object metadata (Object URL) without public access and with public access.
10. Experience assigning permissions public access on bucket level and object level.
11. Try to change storage type of an objects.
12. Enable static website hosting by uploading simple index.html page and try to access the site by using endpoint given.
13. Explore option Inventory under management section to get your total object inventory on daily or weekly basis.
14. Try to modify the tags of a bucket created.
15. Try to disable versioning of a bucket and finally delete the bucket.

* **RDS (Relational Database Service)**

1. Create DB instance using MySQL (or your choice) engine in N.virginia Region using below scenario.
   1. Use only Free tier.
   2. Instance type: db.t2.micro
   3. Allocated storage: 30
   4. Publicly accessible: No
   5. Enable monitoring
   6. Backup: 7 days.
2. Create DB instance using MySQL (or your choice) engine in N.virginia Region using below scenario.
   1. Instance type: db.t2.micro
   2. Allocated storage: 20
   3. Publicly accessible: Yes
   4. Multi - AZ: enable (Yes)
   5. Backup for 10 days every day at 00:30 UTC (select window).
   6. Disable auto-minor version upgrade.
   7. Enable delete protection.
3. Experience stop, start and reboot of your DB instance.
4. Experience the option modify by changing storage type and backup parameters of your DB instance.
5. Take a snapshot of your DB instance and verify once it is completed at Snapshots.
6. Check out for number of automated backups (system snapshots) – after 7 days of DB Instance creation - usually you might see 7 backups as you have selected for 7 days (scenario 1).
7. Explore the option “Restore point in time” by selecting Custom date and time.
8. Create a CloudWatch alarm for your DB instance if your storage free space is less than 10 GiB and send notification.
9. Experience view/downloading logs of your DB instance.
10. Edit Tags on your DB Instance.

* **ELB (Elastic Load Balancer)**

1. Create Classic Load Balancer inside your VPC below scenarios’.
   1. Load Balancer should be accessed internally
   2. Listers port 80
   3. Select default security group
   4. Add a tag Value "ClassicLB-internal"
2. Create Classic Load Balancer inside your VPC below scenarios’.
   1. Load Balancer should be internet access
   2. Listers port both 80 and 443
   3. Create a new security group to access your local machine public ip (ref: whatismyip) with 80 and 443 ingress rules.
   4. Add a tag Value "ClassicLB-public"
3. Explore health checks tab of your load balancer and understand the protocol, port, path and advanced details importance, understand the calculation of timeouts and intervals.
4. Experience cross-zone load balancing by selecting minimum subnets of different zones.
5. Install/Enable IIS on a windows server and register it with "ClassicLB-internal" load balancer. Make sure your load balancer serves IIS page inside VPC using LB DNS.
   1. Make sure instance is healthy at health checks of load balancer.
   2. Required ports are allowed at Security group level.
6. Install apache-tomcat with port 8080 on a Linux instance and register it to "ClassicLB-public" load balancer. Make sure your load balancer serves apache-tomcat page from your local machine using LB DNS.
   1. Make sure instance is healthy at health checks of load balancer.
   2. Required ports are allowed at Security group level.
7. Once Load balancers are created explore options of editing subnets, security groups, Instances and listens from Action Tab.
8. Create "Application Load Balancer (ALB)" inside your VPC with below scenarios’.
   1. Scheme with internet-facing.
   2. Add 443 to Listers port
   3. Select default security group make sure your local machine public ip (ref: whatismyip) with 80 and 443 ingress rules added.
   4. Add a tag Value "ApplicationLB-internet"
   5. Create new target group with port 8080.
   6. Enable stickiness
   7. Register your Linux instance (apache-tomcat installed instance)
   8. Make sure instance is healthy and able to access apache-tomcat page over the internet using load balancer DNS
9. Create a "Target Group" with name "AppLBTG-windows" and port 80, register your window server (IIS Installed instance) with health check interval 20 sec and add success code 301 as well.
10. Create "Application Load Balancer" inside your VPC with below scenarios’.
    1. Scheme with internal.
    2. Listers port 80
    3. Create new Security group and allow required ports.
    4. Add a tag Value "ApplicationLB-internet"
    5. Select existing target group - "AppLBTG-windows".
11. Create another instance in different zone (Linux or Windows) and register to classic/application load balancer (which is available in your practice) to understand round robin concept with multiple instance at load balancer.
12. Observe/Understand instance connection initial, healthy and draining (de-register instances) at Instances (Classic) and Targets (ALB).
13. Create alarm for minimum HTTP 5xx's count reached to 10 on monitoring tab
14. Create alarm whenever minimum unhealthy host count is less than are equal to 1 (one) - Make sure you have at least 2 instances registered and one of the instance is unhealthy.

* **CF (Cloud Formation)**

1. Understand the importance of different section (Parameters, Mapping, Conditions, Resources, Output)
2. Download and validate/understand sample templates from AWS to understand parameter types, Input validations and functions (Fn:) (https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/sample-templates-services-us-west-2.html#w2ab1c23c48c13b7)
3. Launch a LAMP stack using sample template by providing the required inputs, observe the resources created.
4. Create a simple JSON template to launch a windows server in US-east-1 region with free tier AMI.
5. Delete the launched stack and observe the resources deletion associated to stack.

* **Route53**

1. Understanding of domain registration and transfer of domain from one registrar to Route53.
2. Create Hosted Zone and observe default set of records created.
3. Understand the type of record sets like A, CNAME, MX, AAAA, TXT, SPF, etc. and difference between alias / without alias.
4. Create Alias "A" record with your load balancer DNS Name.
5. Create CNAME record set for hosted zone with Failover policy - Primary to your load balancer DNS Name, Secondary to S3 static web hosting page – Verify Failover by making unhealthy instance at load balancer.

* **Auto Scaling**

1. Create a Launch configuration "test-launchConfig-WebServer" using below scenarios
   1. Select Free tier Linux AMI with t2.micro instance type.
   2. Choose IAM Role which has S3 bucket access
   3. Root drive Storage of 10 GiB
   4. select existing security group
2. Create another launch configuration with your customized AMI by coping existing Launch configuration like "test-launchConfig-WebServerCopy".
3. Create Auto-scaling group (ASG) "Test-AutoScaling-WebServer" with in VPC using below scenarios’
   1. Use existing Launch configuration - "test-launchConfig-WebServer"
   2. Group size with 1 instance
   3. Default configuration policies
   4. Choose multiple availability zones
   5. Tag Keys "Name" and "Role" with Values "Linux-WebServer" and "Apache" respectively
   6. Observe "Activity History" of server launch, instance detail at "Instance" tab of ASG and login to server to start services.
4. Edit Auto-scaling group (ASG) "Test-AutoScaling-WebServer" to meet below configuration.
   1. Change Desired capacity, Min and max parameters to launch 2 instances.
   2. Modify Launch configuration to "test-launchConfig-WebServerCopy"
   3. Associate Classic load balancer or target group (Based on ELB created/available) to ASG (make sure instance attached and healthy at ELB)
   4. switch health check Type from EC2 to ELB
   5. Stop required service to make one instance unhealthy at ELB and observe the actions at "Activity History" of ASG
5. Experience adding policy to ASG "Test-AutoScaling-WebServer", Launch one more instance if average CPU usage is more than 80%.
6. Create notification for ASG whenever Instances launch, terminate, fail to launch.
7. Delete Auto-scaling group(ASG), Launch configurations and identify the resources deleted.