Amazon Web Services

Lab assignments

Login to AWS Console and switch to us-east-1 region and finish the following tasks.

1. Create a public EC2 instance with CentOS7 image available in the AWS marketplace and login using pem file. Add the following tags to the instance. “Name=WebServer, Environment=QA, Project=Lab, Node=1”.
   1. Change the SSH authentication to accept passwords also.
   2. Change the centos user password to default123
   3. Disable selinux
   4. Login to this instance from a new terminal using the password.
2. Create a public EC2 instance with windows 2008/2012 R2 and explore how to login to this instance using pem file. Please restrict the access to this server to allow connections only from your public IP address
3. Create an internal EC2 instance **without** any public IP address. Use the WebServer created in step1 to connect to this server. Also see how to upload/download the server from internal server. “Name=mariadb, Environment=QA, Tier=DB”
4. Extend the existing disk size to 12GB of the WebServer instance created in step1. Add the following tags to the volume. “Name=data1, Environment=QA, Tier=web”
5. Add a new volume to the database server with 10GB and mount it under /var/lib/mysql. Add the following tags to the new volume. “Name=mariadb,Environment=QA,Tier=DB”
6. Install mariadb on the internal server and create user to connect from the application.
   1. Install mariadb-server package and set root password to ‘default123’.
   2. Create database ‘sample’ on this database server.
   3. Create db user ‘appclient’ with password ‘app1234’ and allow access from the webserver to connect to the sample database.
   4. Update the security group attached to the database instance to access the mysql connections from web server (Webserver internal IP address should be allowed to access 3306 port ).
7. Configure php-registration application on WebServer and change the documentroot /mnt/data1.
   1. Install apache and php
   2. Update apache DocumentRoot to /mnt/data1.
   3. Download php-registration zip file from <https://github.c.om/satheeshchalla/sources>
   4. Extract content from php-registration.zip to /mnt/data1
   5. Update db.properties file with database IP, username and password (created in the previous step).
8. Verify the application URL.
   1. In browser type in http://<public IP of WebServer>/
   2. Type in username and email and submit.
   3. If the application setup is successful you should be able to see the entries you have submitted in the bottom of the page.
   4. Verify the web server logs if there are any errors.
   5. Login to the database and verify the entries in the sample database users table.

1. Create an AMI from WebServer and add the following tags to the AMI. “Name=web-ami-v1, version=1”
2. Create second webserver in a different availability zone.
   1. Use the AMI created in the previous step to create this instance.
   2. Choose a different subnet while creating the instance.
   3. Add following tags. “Name=WebServer, Environment=QA, Project=Lab, Node=2”
   4. Please add the internal IP of this server in the database server security group to accept the connections.
3. Create an ELB with the following requirement.
   1. Security group name “web”. Anyone should be able to access http.
   2. Listeners. http/80 => http/80
   3. Internet-facing
   4. Tags “Name=elb, Environment=QA,”
   5. Attach both webservers to this ELB. Verify the load distribution to the EC2 instances.
4. Create a self-signed certificate and apply on the ELB
   1. Enable https in the elb security group to allow traffic from everyone.
   2. Add Listener https/443 => http/80
5. Migrate the database server to Amazon RDS.
   1. Create RDS Without multi-AZ
   2. Security group “sg-db” and open 3306 port to internal subnet only.
   3. Tags “Name=DbRDS, Environment=QA,Project=Lab”
   4. Export the data from the DB server and import into RDS.
   5. Stop the database server instance.
6. Replace the database IP address in both web server nodes with the RDS database endpoint and also update the username and password needed to connect to the database if needed.
7. Verify the application URL and make sure all the user entries you created previously are available.
8. Create database snapshot.
   1. Tags “Name=database-v1”
9. Create a new database from the snapshot created with the following requirements.
   1. With multi-AZ enabled
   2. Tags “Name=db-server-v2, Environment=LAB”
10. Migrate the whole application to another region.
    1. Copy web-ami-v1 to us-west-2
    2. Copy RDS snapshot to us-west-2
    3. Create EC2 and RDS instances and test your application functionality.

(You may remove all the resources created in the previous exercises except the AMI ‘web-ami-v1’)

1. Create an EC2 instance and enable following cloudwatch metrics.
   1. cpu utilization >=60
   2. memory utilization >=60
2. Create SNS topic “alerts” and subscribe your email Id to the topic to receive any messages from the Topic.

1. Update cloudwatch to send notifications to alerts topic.
2. Run stress on ec2 instance and increase the CPU utilization more than 60 and verify if you are receiving alerts.
3. IAM
   1. Create IAM user “john” and set the password “default123” for this user.
   2. Create group ‘support’ and the user john to support group.
   3. Create a custom IAM policy “polyglot-iam-v1” with full access to EC2, ELB, S3, Glacier, CloudFront, SNS, CloudWatch, autoscaling and Route53 and attach this policy to support group.
   4. Login with john account to complete the next exercises.
4. Create an auto-scaling group for a web application which has dynamic traffic pattern.
   1. Create RDS instance.
   2. Create your AMI with application configuration bundled
   3. Create a ELB
   4. Create auto-scaling group
5. Create an s3 bucket “demo-<name>-<date>” example : demo-satheesh-20160624
   1. unzip html-website.zip and upload all the content to the bucket created.
   2. Enable website configure index.html as Index Document and error.html as Error Document.
   3. Enable versioning
6. Create another bucket for event logging
   1. Create an s3 bucket “events-satheesh-20160624”
   2. Enable event logging on the first bucket and configure “events-satheesh-20160624” as destination bucket.
   3. Also write lifecycle policies on “events-satheesh-20160624” to archive the data if the files are older than 15 days to Glacier and purge the files older than 30 days.
7. Create a CloudFront distribution
   1. Configure the s3 bucket created in previous step as origin to this distribution.
   2. Disable direct access to s3 bucket using s3 policies
   3. Add domain alias ‘static.example.com’ to this bucket.
8. Create a hosted zone on Route53 for a domain example.com
   1. Create a resource record static.example.com and point to the CloudFront distribution end point.
   2. Create another resource record [www.example.com](http://www.example.com) and point to ELB endpoint.
9. Create OpenVPN server from marketplace and try to connect to EC2 servers using private IP addresses.
10. Create EC2 server with an IAM role to create EC2, RDS, S3 and auto-scaling resources using CLI without giving any user credentials. Try all the above exercises using AWS Command Line Interface.
11. Find ways to recover the EC2 server access when you lost the key pair.
12. Add EC2 servers to newrelic monitoring when they are created using user data.
13. Create an application stack including Security group, Autoscaling group, ELB using CloudFormation template. Apply all the tags similar to examples above.
14. AWS reporting and cost optimization using CloudHealth or Cloudability or Cloudyn
15. gliffy / lucid chart
16. Symian Army
17. Basecamp for team project management, collaboration.