

System Engineering Assignment

Requirement: write timestamp to DB when POST methods is issued to /app endpoint

```
curl -X POST http://avi-app-alb-638864777.eu-west-1.elb.amazonaws.com//app
curl -X POST http://52.18.161.254/app
curl -X POST http://52.17.154.97/app
```

A Records (IPs) for the service URL is highlighted below

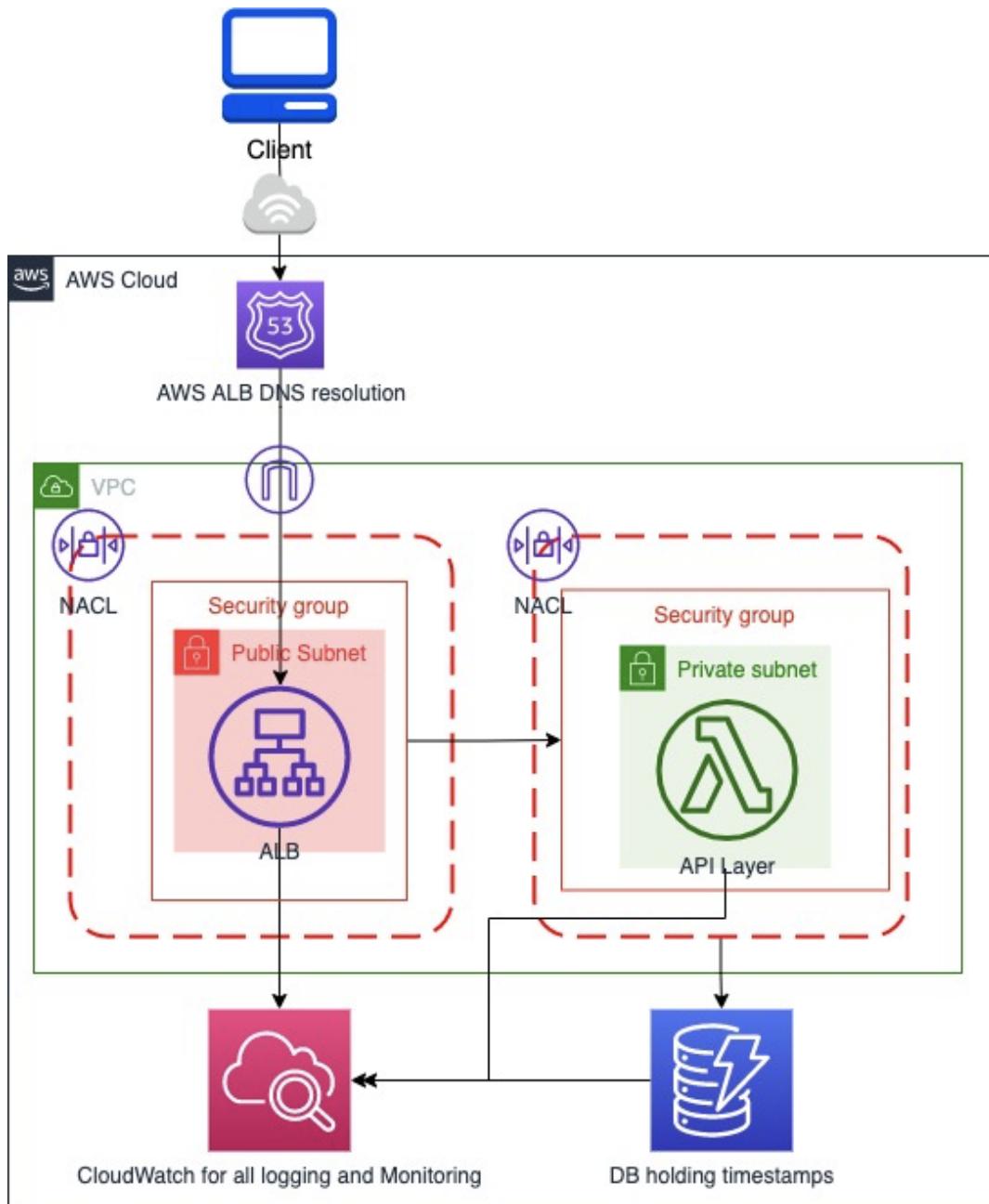
```
avi-app-alb-638864777.eu-west-1.elb.amazonaws.com. 60 IN A 52.18.161.254
avi-app-alb-638864777.eu-west-1.elb.amazonaws.com. 60 IN A 52.17.154.97
```

Index:

- High-level design
- Low-level design
- Deployment
- Technology consideration
- Few clarifications as asked
- Tech Debt
- Reference

Code base: <https://github.com/asrivastava-github/writeDBAPI>

High-Level design (As illustrated and explained below)



The solution consists of two major components:

1. API layer which is being served by AWS Application Load Balancer with AWS Lambda as its target.
2. DB layer which is being served by AWS DynamoDB

When a client (Terminal, Browser etc) sends the request to API using AWS ALB external DNS or Public IP:

- The internet facing AWS Application Load Balancer receives it and based on the endpoint (path - /app in this case) passes it on to the respective target.
- Lambda is connected to target groups of ALB ready to serve the requests.
- Based on request Lambda executes the business logic (writing timestamp to DB in this case) and sends the response back to ALB which forwards it further to client.

Low-level design

An attempt to explain each component/function in detail covering below points

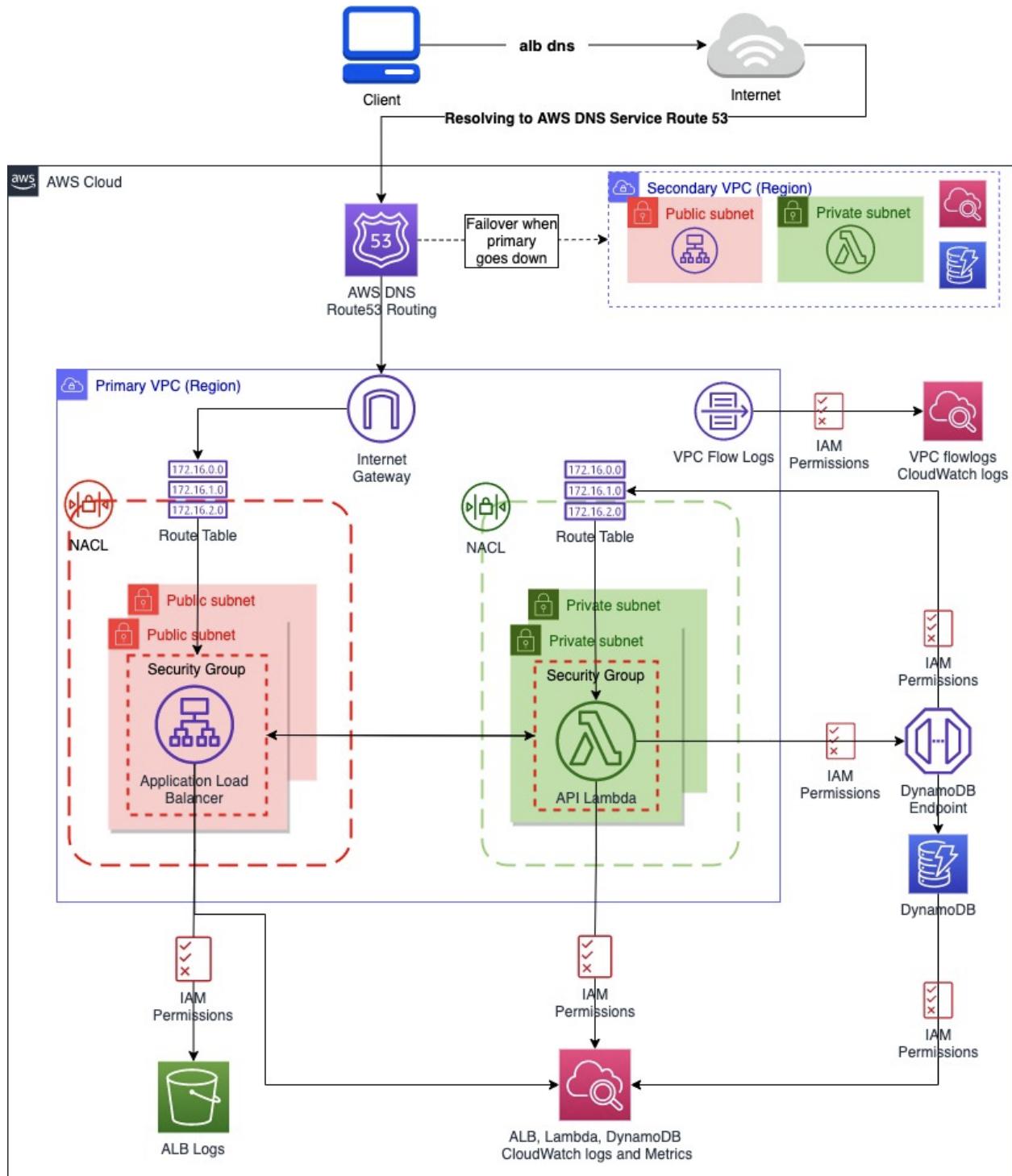
- Network
- Security
 - IAM
 - Network Security
 - Network Access Control Lists
 - Security Group
 - VPC endpoint
 - TLS
 - User Authentication
- API Layer
- DB Layer
- Logging and Monitoring

-
1. Request goes to the internet looking for Address or IP mention.
 2. ISP eventually finds where the requested DNS or IP needs to land, in this case Amazon Web Service, where Route 53 is ready to receive it.
 3. Route53 routes the request to the Internet gateway (IGW) of VPC where solution (ALB + Lambda) has been deployed.
 4. Now if connectivity and routing has been properly sorted, the request will be received by ALB as ALB sits in a Public Subnet which has routing enabled for internet → Internet gateway and local → Public Subnet. Public Subnet is attached to a NACL which allows

traffic from the internet on HTTP and HTTPS port. It allows outbound traffic on ephemeral ports as well (**Required for ALB to respond back**). Another layer of security on ALB, it's own security group allowing traffic only on HTTP and HTTPS ports.

5. ALB will analyse the request and based on the endpoint (path) mentioned in the request it will route it further to one of its Targets, which is nothing but Lambda. Lambda resides inside a Private subnet which has NACL allowing internal/local traffic on port 80 and 443, **along with Amazon's DynamoDB Service IP list for eu-west-*region on ephemeral ports required for DynamoDB to connect with Lambda which is deployed inside VPC.**
6. Lambda is a serverless compute ready to execute the code/business logic based on the request sent to it in the form of an Event. Lambda also carries its own security group rules to accept HTTP and HTTPS traffic from ALB security group only.
7. Post understanding the request, Lambda sends the response back to ALB in a specific format which ALB can understand and serve it further to the client, again if connectivity is properly sorted for outbound as well.
 - a. Now it's worth explaining how the actual requirement is being served here. When Lambda receives the request where the **endpoint** (Path) is “/app” and the method is **POST**, it tries to connect to DB layer which is being served by AWS DynamoDB.
 - b. Since Lambda is deployed under my/customer’s VPC (NOT AWS own VPC), it cannot connect to AWS Managed Service DynamoDB straight away. Lambda will need a VPC endpoint created for DynamoDB which AWS facilitates free of cost but making it work is tricky, so ultimately a Secured Serverless compute has a way to connect to Serverless DynamoDB securely and internally (within AWS network, without going to internet)
 - c. Post executing the business logic lambda responds to ALB and ALB takes care further.
8. HTTPS connectivity can be enabled by creating a certificate based on DNS and attaching it to the ALB HTTPS listener. This has not been covered here.
9. Additionally Lambda can easily be integrated with Cognito (AWS managed user authentication Server or IDP needs to be explored, Same is the case with DynamoDB).
10. Logging and Monitoring has been enabled at each level.
 - a. Starting with vpc flow logs to capture all the requests entering to VPC
 - b. ALB logs being stored to s3 and AWS provided metrics
 - c. Cloudwatch logs have been enabled for Lambda.
 - d. Cloudwatch metric for DynamoDB

Below is the low level design touching each component.



Deployment

Checkout below README for in depth detail

<https://github.com/asrivastava-github/writeDBAPI/blob/main/README.md>

- Directory Structure
 - Deployer file --> deploy_from_local.py
 - Business logic/API layer/Application -> ./application/write_timestamp.py
 - An attempt to design the application structure so that deployment can be controlled via config file --> application_structure.json
 - Infrastructure builder --> main.tf
 - Create security group rules
--> ./infrastructure/modules/security_group_rules/sg_rules.tf
 - Create Security groups
--> ./infrastructure/modules/security_groups/sg.tf
 - Configure the infrastructure of API layer e.g. Lambda and target groups etc.
--> ./infrastructure/modules/target/lambda_target.tf

```
|Avinashs-MBP:writeDBAPI avinashsrivastava$ tree
.
├── README.md
├── application
│   └── write_timestamp.py
├── application_structure.json
├── deploy_from_local.py
└── infrastructure
    ├── modules
    │   ├── security_group_rules
    │   │   └── sg_rules.tf
    │   ├── security_groups
    │   │   └── sg.tf
    │   └── target
    │       └── lambda_target.tf
    └── policy
        ├── lambda_iam_policy.json
        └── vpc_flow_logs_policy.json
└── main.tf

7 directories, 10 files
Avinashs-MBP:writeDBAPI avinashsrivastava$
```

- Prerequisite:

- Terraform version v0.13 or above must be installed. This project is build/tested using v0.14.2/0.15.4
- Python version 3 (3.5 above) should be installed, This project is build/tested using Python 3.8.2
 - python3 -m pip install boto3 (pip Install boto3)
 - python3 -m pip install requests (pip Install requests)

Avoiding file clutter else a requirements.txt will also be okay to define the required python packages. [Quickstart — Boto 3 Docs 1.9.42 documentation](#)

- AWS Configure

- Create an Admin account on AWS which has access to provision the resources like ALB, Lambda, DynamoDB, CloudWatch, IAM etc
- Install AWS CLI on your machine, Make sure you are able to connect to AWS post configuring AWS with "aws configure"
- Default region used is eu-west-1

- Worth checking: Login to AWS console --> go to VPC --> Managed prefix lists, It should exist for s3 and DynamoDB or for either of them. If not, don't worry terraform will create.
- Clone Project repository writeDBAPI

```
git clone https://github.com/asrivastava-github/writeDBAPI.git
```

- Deployment: plan --> apply --> visual test --> planDestroy --> destroy

```
cd writeDBAPI
```

Currently I am owning an avi-assignment-api-service bucket so if you are testing/deploying the solution, and getting a bucket exists error, let me know I will delete. Alternatively you can change the bucket name in application_structure.json as s3 bucket names are unique.

- Run below to plan the system infrastructure
 - python3 deploy_from_local.py -a plan -e poc
- Run below to provision the system infrastructure
 - python3 deploy_from_local.py -a apply -e poc
- Starting visual testing/Using the system
 1. You can hit `http://<ALB DNS printed as output>` in the browser to see the welcome page.
 2. Run the below curl command which will create the timestamp entry inside DynamoDB

```
curl -X POST http://<ALB DNS printed as output>/app
```

3. Hit link <http://<ALB DNS printed as output>/app> in browser to see all the time stamps recorded.

```
Apply complete! Resources: 51 added, 0 changed, 0 destroyed.

Outputs:

albDNS = "avi-app-alb-1874219681.eu-west-1.elb.amazonaws.com"
[Avinashs-MBP:writeDBAPI avinashsrivastava$ curl -X POST http://avi-app-alb-1874219681.eu-west-1.elb.amazonaws.com/app
2021-05-23 22:15:06.172057
[Avinashs-MBP:writeDBAPI avinashsrivastava$ curl -X POST http://avi-app-alb-1874219681.eu-west-1.elb.amazonaws.com/app
2021-05-23 22:15:07.468924
[Avinashs-MBP:writeDBAPI avinashsrivastava$ curl -X POST http://avi-app-alb-1874219681.eu-west-1.elb.amazonaws.com/app
2021-05-23 22:15:08.567516
[Avinashs-MBP:writeDBAPI avinashsrivastava$ curl -X POST http://avi-app-alb-1874219681.eu-west-1.elb.amazonaws.com/app
2021-05-23 22:15:09.383439
[Avinashs-MBP:writeDBAPI avinashsrivastava$ curl -X POST http://avi-app-alb-1874219681.eu-west-1.elb.amazonaws.com/app
2021-05-23 22:15:10.173715
[Avinashs-MBP:writeDBAPI avinashsrivastava$ curl -X POST http://avi-app-alb-1874219681.eu-west-1.elb.amazonaws.com/app
2021-05-23 22:15:11.442211
[Avinashs-MBP:writeDBAPI avinashsrivastava$ curl -X POST http://avi-app-alb-1874219681.eu-west-1.elb.amazonaws.com/app
2021-05-23 22:15:12.266034
[Avinashs-MBP:writeDBAPI avinashsrivastava$ curl -X POST http://avi-app-alb-1874219681.eu-west-1.elb.amazonaws.com/app
2021-05-23 22:15:13.069396
[Avinashs-MBP:writeDBAPI avinashsrivastava$ curl -X POST http://avi-app-alb-1874219681.eu-west-1.elb.amazonaws.com/app
2021-05-23 22:15:13.962218
[Avinashs-MBP:writeDBAPI avinashsrivastava$
```

```
← → C ⚠ Not Secure | avi-app-alb-1874219681.eu-west-1.elb.amazonaws.com/app

{'uniqueId': 'Root-1-60aad3ea-525c293248abaf1a44b59cf092021-05-2322:15:06.172057', 'clientIP': '81.140.213.55', 'timeStamp': '2021-05-23 22:15:06.172057', 'protocol': 'http'}, {'uniqueId': 'Root-1-60aad3eb-04c38a4c3955d391206976632021-05-2322:15:07.468924', 'clientIP': '81.140.213.55', 'timeStamp': '2021-05-23 22:15:07.468924', 'protocol': 'http'}, {'uniqueId': 'Root-1-60aad3ec-5a7d36284713d39a6f1967f8c2021-05-2322:15:08.567516', 'clientIP': '81.140.213.55', 'timeStamp': '2021-05-23 22:15:08.567516', 'protocol': 'http'}, {'uniqueId': 'Root-1-60aad3ed-497b16227e01fc267e7cf72021-05-2322:15:09.383439', 'clientIP': '81.140.213.55', 'timeStamp': '2021-05-23 22:15:09.383439', 'protocol': 'http'}, {'uniqueId': 'Root-1-60aad3ef-04f47710011c131c496582021-05-2322:15:10.173715', 'clientIP': '81.140.213.55', 'timeStamp': '2021-05-23 22:15:10.173715', 'protocol': 'http'}, {'uniqueId': 'Root-1-60aad3ef-04f474f2433b709624991092021-05-2323:15:11.442211', 'clientIP': '81.140.213.55', 'timeStamp': '2021-05-23 22:15:11.442211', 'protocol': 'http'}, {'uniqueId': 'Root-1-60aad3f0-170aa26500cdab8e6950f79f2021-05-2322:15:12.266034', 'clientIP': '81.140.213.55', 'timeStamp': '2021-05-23 22:15:12.266034', 'protocol': 'http'}, {'uniqueId': 'Root-1-60aad3f1-507400c229047e3e29133d5d2021-05-2322:15:13.069396', 'clientIP': '81.140.213.55', 'timeStamp': '2021-05-23 22:15:13.069396', 'protocol': 'http'}, {'uniqueId': 'Root-1-60aad3f1-5eba385e73e8f4df55e8eca82021-05-2322:15:13.962218', 'clientIP': '81.140.213.55', 'timeStamp': '2021-05-23 22:15:13.962218', 'protocol': 'http'}
```

- Destroy the system

1. Check what's being destroyed

```
python3 deploy_from_local.py -a planDestroy -e poc
```

2. Finally destroy

```
python3 deploy_from_local.py -a destroy -e poc
```

Deployment with AzureDevOps (CI/CD)

Pipeline:

<https://github.com/asrivastava-github/writeDBAPI/blob/release/v2/pipeline.yaml>

Which uses python deployer script:

<https://github.com/asrivastava-github/writeDBAPI/blob/release/v2/deployer.py>

Preparation:

1. Create an IAM User in AWS for terraform running on Azure DevOps to connect

The screenshot shows the AWS IAM User summary page for a user named 'adoterraform'. The top navigation bar shows 'Users > adoterraform'. The main section is titled 'Summary' and displays the following details:

User ARN	arn:aws:iam::123456789012:User/adoterraform
Path	/
Creation time	2020-01-01T12:00:00Z

Below the summary, there are three tabs: 'Permissions' (selected), 'Groups (1)', and 'Tags (2)'. The 'Permissions' tab shows a list of attached policies:

- Permissions policies (8 policies)
- Add permissions button

The 'Attached from group' section lists the following policies:

- AmazonEC2FullAccess
- IAMFullAccess
- ElasticLoadBalancingFullAccess
- AmazonS3FullAccess
- CloudWatchFullAccess
- AmazonDynamoDBFullAccess
- AmazonVPCFullAccess
- AWSLambda_FullAccess

2. Either you can create a service connection or create a variable group to store the credential as secret (a sensitive variable which ADO will not expose). You can use Azure vault as well.

The screenshot shows the 'Variables' section of the 'aws-terraform' variable group in the Azure DevOps Library. It lists four variables:

Name	Value
AWS_ACCESS_KEY_ID	*****
AWS_SECRET_ACCESS_KEY	*****
BASE_PATH	terraform/modules
TERRAFORM_VERSION	0.13.4

This is all We need to connect from Azure DevOps to AWS over the internet. It will be recommended to have a self hosted Azure agent (installed inside your premises cloud or DC) and have connectivity to AWS cloud.

Now coming to deployment, It has 3 major steps to perform:

1. Scan terraform code using checkov

Jobs in run #20210606.1	
asrivastava-github.writeDBAPI	
Terraform Scan	
✓ Scan network_layer	25s
✓ Initialize job	<1s
✓ Checkout asrivastava-...	2s
✓ Install Checkov	15s
✓ Scan terraform DSL	6s
✓ Post-job: Checkout a...	<1s
✓ Finalize Job	<1s
Terraform Action	
✓ Terraform plan	22s
✓ Initialize job	<1s
✓ Checkout asrivastava-...	2s
✓ Install boto3	5s
✓ Configure AWS poc	4s
✓ plan network_layer	9s
✓ Post-job: Checkout a...	<1s
✓ Finalize Job	<1s
Load Test	
Load Test on network_layer	
Scan terraform DSL	
16	_ _ _ _ _
17	/ _ \ / _ \ / _ \ / _ \ / _ \ / _ \ /
18	() _ / (_ < () \ V /
19	\ _ \ _ \ \ _ \ \ _ \ \ _ \ / \ /
20	
21	By bridgecrew.io version: 2.0.178
22	
23	terraform scan results:
24	
25	Passed checks: 33, Failed checks: 0, Skipped checks: 0
26	
27	Check: CKV_AWS_66: "Ensure cloudwatch log groups specify retention days"
28	PASSED for resource: aws_cloudwatch_log_group.vpc-logs
29	File: /main.tf:50-53
30	Guide: https://docs.bridgecrew.io/docs/logging_13
31	
32	Check: CKV_AWS_60: "Ensure IAM role allows only specific services or principals to assume it"
33	PASSED for resource: aws_iam_role.vpc-logs-role
34	File: /main.tf:56-68
35	Guide: https://docs.bridgecrew.io/docs/bc_aws_iam_44
36	
37	Check: CKV_AWS_61: "Ensure IAM role allows only specific principals in account to assume it"
38	PASSED for resource: aws_iam_role.vpc-logs-role
39	File: /main.tf:56-68
40	Guide: https://docs.bridgecrew.io/docs/bc_aws_iam_45
41	
42	Check: CKV_AWS_62: "Ensure IAM policies that allow full \"*-*\" administrative privileges are not created"
43	PASSED for resource: aws_iam_policy.vpc_flow_logs_policy
44	File: /main.tf:70-73
45	Guide: https://docs.bridgecrew.io/docs/iam_47
46	
47	Check: CKV_AWS_63: "Ensure no IAM policies documents allow \"*\" as a statement's actions" ↗
48	PASSED for resource: aws_iam_policy.vpc_flow_logs_policy
49	File: /main.tf:70-73
50	Guide: https://docs.bridgecrew.io/docs/iam_48
51	
52	Check: CKV_AWS_40: "Ensure IAM policies are attached only to groups or roles (Reducing access management risk)" ↗
53	PASSED for resource: aws_iam_policy_attachment.attach_vpc_flow_log_policy
54	File: /main.tf:75-90

2. Perform Terraform deployment

ersrivastava / DevOps / Pipelines / asrivastava-github.writeD... / 20210606.1

Jobs in run #20210606.1	plan network_layer
Terrafrom Scan	
✓ Scan network_layer 25s	
✓ Initialize job <1s	
✓ Checkout asrivastava-... 2s	
✓ Install Checkov 15s	
✓ Scan terraform DSL 6s	
✓ Post-job: Checkout a... <1s	
✓ Finalize Job <1s	
Terrafrom Action	
✓ Terraform plan 22s	
✓ Initialize job <1s	
✓ Checkout asrivastava-... 2s	
✓ Install boto3 5s	
✓ Configure AWS poc 4s	
✓ plan network_layer 9s	
✓ Post-job: Checkout a... <1s	
✓ Finalize Job <1s	
Load Test	

```
+ id                      = (known after apply)
+ protocol                = "tcp"
+ security_group_id        = (known after apply)
+ self                     = false
+ source_security_group_id = (known after apply)
+ to_port                  = 443
+ type                     = "egress"
}

# module.lambda_in[0].aws_security_group_rule.sg_rule[0] will be created
+ resource "aws_security_group_rule" "sg_rule" {
  + description              = "Lambda inbound"
  + from_port                = 80
  + id                       = (known after apply)
  + protocol                 = "tcp"
  + security_group_id        = (known after apply)
  + self                     = false
  + source_security_group_id = (known after apply)
  + to_port                  = 80
  + type                     = "ingress"
}

# module.lambda_in[1].aws_security_group_rule.sg_rule[0] will be created
+ resource "aws_security_group_rule" "sg_rule" {
  + description              = "Lambda inbound"
  + from_port                = 443
  + id                       = (known after apply)
  + protocol                 = "tcp"
  + security_group_id        = (known after apply)
  + self                     = false
  + source_security_group_id = (known after apply)
  + to_port                  = 443
  + type                     = "ingress"
}

Plan: 42 to add, 0 to change, 0 to destroy.
```

← Jobs in run #20210606.2 asrivastava-github.writeDBAPI		✓ apply network_layer
Terrafrom Scan		
> ✓ Scan network_layer 26s		
Terrafrom Action		
✓ ✓ Terraform apply 2m 32s		
✓ Initialize job <1s		
✓ Checkout asrivastava-... 2s		
✓ Install boto3 7s		
✓ Configure AWS poc 2s		
✓ apply network_la... 2m 18s		
✓ Post-job: Checkout a... <1s		
✓ Finalize Job <1s		
Load Test		
⌚ Load Test on network_layer		
<pre> 913 aws_iam_policy_attachment.attach_vpc_flow_log_policy: Creation complete after 2s 914 module.alb_in[0].aws_security_group_rule.cidr_sg_rule[0]: Creation complete after 2s 915 module.alb_out_std[0].aws_security_group_rule.sg_rule[0]: Creation complete after 2s 916 module.alb_out_std[1].aws_security_group_rule.sg_rule[0]: Creation complete after 2s 917 aws_dynamodb_table.app_db: Creation complete after 6s [id=avi-app-dynamo] 918 aws_vpc_endpoint.dynamoDB: Creation complete after 6s [id=vpce-066559c943b5ae9d] 919 data.aws_prefix_list.private_dynamoDB: Reading... 920 aws_network_acl_rule.private_nacl_rules_in_dynamoDB_EP[3]: Creating... 921 aws_network_acl_rule.private_nacl_rules_in_dynamoDB_EP[1]: Creating... 922 aws_network_acl_rule.private_nacl_rules_in_dynamoDB_EP[0]: Creating... 923 aws_network_acl_rule.private_nacl_rules_in_dynamoDB_EP[2]: Creating... 924 data.aws_prefix_list.private_dynamoDB: Read complete after 0s [id=pl-b3a742da] 925 aws_security_group_rule.lambda_out_endpoint: Creating... 926 aws_network_acl_rule.private_nacl_rules_in_dynamoDB_EP[1]: Creation complete after 2s 927 aws_network_acl_rule.private_nacl_rules_in_dynamoDB_EP[0]: Creation complete after 2s 928 aws_network_acl_rule.private_nacl_rules_in_dynamoDB_EP[2]: Creation complete after 2s 929 aws_network_acl_rule.private_nacl_rules_in_dynamoDB_EP[3]: Creation complete after 2s 930 aws_security_group_rule.lambda_out_endpoint: Creation complete after 0s [id=sgr] 931 aws_lb.lb: Still creating... [10s elapsed] 932 aws_lb.lb: Still creating... [20s elapsed] 933 aws_lb.lb: Still creating... [30s elapsed] 934 aws_lb.lb: Still creating... [40s elapsed] 935 aws_lb.lb: Still creating... [50s elapsed] 936 aws_lb.lb: Still creating... [1m0s elapsed] 937 aws_lb.lb: Still creating... [1m10s elapsed] 938 aws_lb.lb: Still creating... [1m20s elapsed] 939 aws_lb.lb: Still creating... [1m30s elapsed] 940 aws_lb.lb: Still creating... [1m40s elapsed] 941 aws_lb.lb: Still creating... [1m50s elapsed] 942 aws_lb.lb: Still creating... [2m0s elapsed] 943 aws_lb.lb: Creation complete after 2m1s [id=arn:aws:elasticloadbalancing:eu-west-2:123456789012:loadbalancer/app/12345678901234567890123456789012] 944 945 Apply complete! Resources: 42 added, 0 changed, 0 destroyed. 946 </pre>		

ersrivastava / DevOps / Pipelines / asrivastava-github.writeDBAPI / 20210606.7

← Jobs in run #20210606.7 asrivastava-github.writeDBAPI		✓ apply api_layer
Terraform Scan		
> ✓ Scan api_layer	17s	457 module.createLambda.data.aws_lb.lb: Reading... 458 module.createLambda.data.aws_security_groups.private_sg: Reading... 459 module.createLambda.data.aws_vpcs.avi_vpc: Reading... 460 module.createLambda.data.aws_subnet.private_subnet_1: Read complete after 0s [id=subnet-01a2d4317b74309a2] 461 module.createLambda.data.aws_vpcs.avi_vpc: Read complete after 0s [id=eu-west-2] 462 module.createLambda.data.aws_security_groups.private_sg: Read complete after 0s [id=eu-west-2] 463 module.createLambda.aws_lb_target_group.tg: Creating... 464 module.createLambda.data.aws_subnet.private_subnet_0: Read complete after 0s [id=subnet-027debd27e33f6efe] 465 module.createLambda.aws_lambda_function.write_api: Creating... 466 module.createLambda.data.aws_elasticloadbalancing.eu-west-2:377219 467 module.createLambda.aws_lb_target_group.tg: Creation complete after 0s [id=arn:aws:elasticloadbalancing:eu-west-2: 468 module.createLambda.aws_lb_listener.lb_listener: Creating... 469 module.createLambda.aws_lb_listener.lb_listener: Creation complete after 1s [id=arn:aws:elasticloadbalancing: 470 module.createLambda.aws_alb_listener_rule.listener_rule: Creating... 471 module.createLambda.aws_alb_listener_rule.listener_rule: Creation complete after 0s [id=arn:aws:elasticloadbalanc 472 aws_iam_policy_attachment.attach_lambda_policy: Creation complete after 1s [id=lambdaPolicyAttach] 473 module.createLambda.aws_lambda_function.write_api: Still creating... [10s elapsed] 474 module.createLambda.aws_lambda_function.write_api: Still creating... [20s elapsed] 475 module.createLambda.aws_lambda_function.write_api: Still creating... [30s elapsed] 476 module.createLambda.aws_lambda_function.write_api: Still creating... [40s elapsed] ↵ 477 module.createLambda.aws_lambda_function.write_api: Still creating... [50s elapsed] 478 module.createLambda.aws_lambda_function.write_api: Still creating... [1m0s elapsed] 479 module.createLambda.aws_lambda_function.write_api: Still creating... [1m10s elapsed] 480 module.createLambda.aws_lambda_function.write_api: Still creating... [1m20s elapsed] 481 module.createLambda.aws_lambda_function.write_api: Still creating... [1m30s elapsed] 482 module.createLambda.aws_lambda_function.write_api: Still creating... [1m40s elapsed] 483 module.createLambda.aws_lambda_function.write_api: Still creating... [1m50s elapsed] 484 module.createLambda.aws_lambda_function.write_api: Still creating... [2m0s elapsed] 485 module.createLambda.aws_lambda_function.write_api: Still creating... [2m10s elapsed] 486 module.createLambda.aws_lambda_function.write_api: Still creating... [2m20s elapsed] 487 module.createLambda.aws_lambda_function.write_api: Creation complete after 2m26s [id=avi-lambda-app-api] 488 module.createLambda.aws_lambda_permission.alb-permission: Creating... 489 module.createLambda.aws_lambda_permission.alb-permission: Creation complete after 0s [id=AllowExecutionFromA 490 module.createLambda.aws_lb_target_group_attachment.attach-lambda: Creating... 491 module.createLambda.aws_lb_target_group_attachment.attach-lambda: Creation complete after 0s [id=arn:aws:ela 492 493 Apply complete! Resources: 9 added, 0 changed, 0 destroyed. 494 Finishing: apply api_layer
Terraform Action		
> ✓ Terraform apply	3m 6s	
✓ Initialize job	1s	
✓ Checkout asrivastava-...	7s	
✓ Install boto3	7s	
✓ Configure AWS poc	6s	
✓ apply api_layer	2m 43s	
✓ Post-job: Checkout a...	<1s	
✓ Finalize Job	<1s	
Load Test		
> ✓ Load Test on api_layer	56s	
✓ Initialize job	1s	
✓ Checkout asrivastava-...	2s	
✓ Install locust, boto3 f...	11s	
✓ Configure AWS poc	2s	
✓ Running locust load ...	38s	
✓ Post-job: Checkout a...	<1s	

3. Perform load test using locust in case the api layer is deployed. (Need to work on DB connectivity) will complete the refactoring post discussion.

ersrivastava / DevOps / Pipelines / asrivastava-github.writeDBAPI / 20210606.7

← Jobs in run #20210606.7
asrivastava-github.writeDBAPI

Terraform Scan

> ✓ Scan api_layer 17s

Terraform Action

> ✓ Terraform apply 3m 6s

Load Test

✓ Load Test on api_layer 56s

- ✓ Initialize job 1s
- ✓ Checkout asrivastava-... 2s
- ✓ Install locous, boto3 f... 11s
- ✓ Configure AWS poc 2s
- ✓ Running locust load ... 38s
- ✓ Post-job: Checkout a... <1s
- ✓ Finalize Job <1s

Running locust load test

```

228 INFO:load_test:avi-app-alb-498259348.eu-west-2.elb.amazonaws.com
229 INFO:load_test:
230 Executing: cd api_layer && locust -f locustfile.py --host=http://avi-app-alb-498259348.eu-west-2.elb.amazonaws.com --run-time=30s --user-count=10 --spawn-rate=10
231
232
233
234 [2021-06-06 10:24:18,681] fv-az414-488/INFO/locust.main: Run time limit set to 20 seconds
235 [2021-06-06 10:24:18,681] fv-az414-488/INFO/locust.main: Starting Locust 1.5.3
236 [2021-06-06 10:24:18,682] fv-az414-488/INFO/locust.runners: Spawning 10 users at the rate 10 users/s (0 users already running)
237 [2021-06-06 10:24:18,682] fv-az414-488/INFO/root: Terminal was not a tty. Keyboard input disabled
238 Name # reqs # fails | Avg Min Max Median
239 -----
240 Aggregated 0 0(0.00%) | 0 0 0 0 0
241
242 [2021-06-06 10:24:19,593] fv-az414-488/INFO/locust.runners: All users spawned: QuickstartUser: 10 (10 total running)
243 Name # reqs # fails | Avg Min Max Median
244 -----
245 Aggregated 0 0(0.00%) | 0 0 0 0 0
246
247 [2021-06-06 10:24:19,593] fv-az414-488/INFO/locust.runners: All users spawned: QuickstartUser: 10 (10 total running)
248 Name # reqs # fails | Avg Min Max Median
249 -----
250 Aggregated 0 0(0.00%) | 0 0 0 0 0
251
252 Aggregated 0 0(0.00%) | 0 0 0 0 0
253

```

ersrivastava / DevOps / Pipelines / awsAppDeploy / 20210605.88

← Jobs in run #20210605.88
awsAppDeploy

Terraform Scan

> ✓ Scan api_layer 19s

Terraform Action

> ✓ Terraform apply 1m 15s

Load Test

✗ Load Test on api_layer 52s

- ✓ Initialize job 1s
- ✓ Checkout awsAppDepl... 1s
- ✓ Install locous, boto3 f... 11s
- ✓ Configure AWS poc 2s
- ✗ Running locust load ... 36s
- ✓ Post-job: Checkout a... <1s
- ✓ Finalize Job <1s

Running locust load test

```

251 [2021-06-05 19:35:25,361] fv-az201-729/INFO/locust.main: Time limit reached. Stopping Locust.
252 [2021-06-05 19:35:25,362] fv-az201-729/INFO/locust.runners: Stopping 10 users
253 [2021-06-05 19:35:25,363] fv-az201-729/INFO/locust.runners: 10 Users have been stopped, 0 still running
254 [2021-06-05 19:35:25,363] fv-az201-729/INFO/locust.main: Running teardowns...
255 [2021-06-05 19:35:25,363] fv-az201-729/INFO/locust.main: Shutting down (exit code 1), bye.
256 [2021-06-05 19:35:25,363] fv-az201-729/INFO/locust.main: Cleaning up runner...
257 Name # reqs # fails | Avg Min Max Median | req/s failures/s
258 -----
259 GET //app 33 33(100.00%) | 23 14 34 22 | 1.71 1.71
260 POST //app 77 77(100.00%) | 23 13 58 23 | 3.99 3.99
261 Aggregated 110 110(100.00%) | 23 13 50 22 | 5.70 5.70
262
263 Response time percentiles (approximated)
264 Type Name 50% 66% 75% 80% 90% 95% 98% 99% 99.9% 99.99% 100% # reqs
265 -----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
266 GET //app 22 24 25 25 27 35 35 35 35 35 35 33
267 POST //app 23 25 26 27 32 38 41 50 50 50 50 77
268
269 None Aggregated 23 24 26 26 31 35 40 41 50 50 50 110
270
271 Error report
272 # occurrences Error
273 77 POST //app: HTTPError('502 Server Error: Bad Gateway for url: http://avi-app-alb-1689568087.eu-west-2.elb.amazonaws.com/app')
274
275 77 GET //app: HTTPError('502 Server Error: Bad Gateway for url: http://avi-app-alb-1689568087.eu-west-2.elb.amazonaws.com/app')
276 33
277

```

Technology consideration

Tools:

1. Terraform as Infrastructure as Code (Iac)
2. Git for version control (github)
3. Python for automation

AWS Platform:

- Route 53
- Internet Gateway
- Virtual Private Cloud (VPC) and Subnets
- Route Tables, NACL and Security Groups.
- VPC Endpoint for DynamoDB Service
- Application Load Balancer
- Lambda
- DynamoDB
- CloudWatch
- S3
- And other services like IAM, ALB listeners, Target groups etc

Amazon Web Service's Serverless platforms is suitable for such requirement's solution, because:

1. I can avoid platform maintenance overhead in terms of
 - a. Patching
 - b. Scanning
 - c. Upgrade and updates
 - d. Life cycle management
 - e. From pay as you go to pay when it runs (in milliseconds.)
2. Highly scalable and AWS managed service.
3. Flexibility in terms of scalability and elasticity.

Choosing NoSQL DB for this requirement because it's not asking to follow a strict schema to maintain and We are looking to perform any relational activity among tables hence NoSQL is suitable as it can scale.

Few clarifications as asked:

1. Provide details of your web API that will be used to submit a curl command (curl -X POST <http://<someip>/app>) that will insert the DateTime stamp into your database. The api code should be well documented.

```
curl -X POST http://avi-app-alb-859748445.eu-west-1.elb.amazonaws.com/app
```

Or

```
curl -X POST http://34.240.221.237/app
```

Or

```
curl -X POST http://108.128.135.54/app
```

A Records (IPs) for the service URL is highlighted below

```
avi-app-alb-859748445.eu-west-1.elb.amazonaws.com. 60 IN A 34.240.221.237
avi-app-alb-859748445.eu-west-1.elb.amazonaws.com. 60 IN A 108.128.135.54
```

```
Outputs:

albDNS = "avi-app-alb-859748445.eu-west-1.elb.amazonaws.com"
[Avinashs-MBP:writeDBAPI avinashsrivastava$ dig avi-app-alb-859748445.eu-west-1.elb.amazonaws.com
; <>> DiG 9.10.6 <>> avi-app-alb-859748445.eu-west-1.elb.amazonaws.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 64355
;; flags: qr rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 1
;; OPT PSEUDOSECTION:
;; EDNS: version: 0, flags:; udp: 512
;; QUESTION SECTION:
;avi-app-alb-859748445.eu-west-1.elb.amazonaws.com. IN A
;; ANSWER SECTION:
avi-app-alb-859748445.eu-west-1.elb.amazonaws.com. 60 IN A 34.240.221.237
avi-app-alb-859748445.eu-west-1.elb.amazonaws.com. 60 IN A 108.128.135.54
;; Query time: 32 msec
;; SERVER: 192.168.1.254#53(192.168.1.254)
;; WHEN: Wed May 26 07:18:21 BST 2021
;; MSG SIZE  rcvd: 110

[Avinashs-MBP:writeDBAPI avinashsrivastava$ curl -X POST http://avi-app-alb-859748445.eu-west-1.elb.amazonaws.com/app
2021-05-26 06:21:45.660429
[Avinashs-MBP:writeDBAPI avinashsrivastava$ curl -X POST http://34.240.221.237/app
2021-05-26 06:22:08.074466
[Avinashs-MBP:writeDBAPI avinashsrivastava$ curl -X POST http://108.128.135.54/app
2021-05-26 06:22:17.135785
Avinashs-MBP:writeDBAPI avinashsrivastava$ ]
```

2. In addition, provide any automation for your backup/restore process you would implement.

With AWS DynamoDB, which is AWS managed services We get feature of replication and backup as well, It can be easily enabled (Commented out in my code base at <https://github.com/asrivastava-github/writeDBAPI/blob/main/main.tf#L298-L303>).

Additionally with the help of this feature in case of primary region failure once route 53 will start routing request to secondary region (Which is not part of code because of absence of hosted zone and domain for now) the secondary region API layer will be able to consume the replica/backup DynamoDB (We have chosen same region for DynamoDB replica and route 53 failover region for API layer)

- Provide details of your Persistence Layer, with details of your cluster setup and configuration.

AWS provides great compatibility between Lambda and DynamoDB. At the API layer (Lambda) I need to just mention the DynamoDB table name and DynamoDB endpoint while creating the client (Needed when Lambda is inside VPC and I would recommend it for security reasons) and voila a secure connectivity is established. There is AWS resource level trust defined between Lambda and DynamoDB using IAM policy for authentication, Hence I don't need to manage any credential (another plus). You can access DynamoDB tables with login to the AWS management console.

Table and entries inside DynamoDB screen shots:

The screenshot shows the AWS DynamoDB console. In the top navigation bar, there is a search bar with placeholder text 'Search for services, features, marketplace products, and docs [Option+S]' and a user dropdown for 'Avinash'. Below the search bar, there are tabs for Overview, Items, Metrics, Alarms, Capacity, Indexes, Global Tables, Backups, Contributor Insights, Triggers, and Access control. The 'Items' tab is selected. On the left, there is a sidebar with buttons for 'Create table' and 'Delete table', and a filter section for 'Filter by table name' with a dropdown set to 'Choose a table ...' and an 'Actions' dropdown. The main content area displays a table titled 'Scan: [Table] avi-app-dynamo: uniqueId, timeStamp'. The table has columns: uniqueId, timeStamp, clientIP, and protocol. The data shows four rows of results from a scan operation:

	uniqueId	timeStamp	clientIP	protocol
<input type="checkbox"/>	Root=1-60ade8f9-67e0c7b9738af5492e8dd1d62021-05-2606:21:45.660429	2021-05-26 06:21:45.660429	81.140.213.55	http
<input type="checkbox"/>	Root=1-60ade910-1b4cf9e821725df8542cb59f2021-05-2606:22:08.074466	2021-05-26 06:22:08.074466	81.140.213.55	http
<input type="checkbox"/>	Root=1-60ade919-18dd96151a66f60d12b5eb6e2021-05-2606:22:17.135785	2021-05-26 06:22:17.135785	81.140.213.55	http

Additionally you can have External authentication enabled using IDP but that is not applicable in this case.

Identity Provider Screenshot:

4. Please explain how elastic your service is, what would be trigger points and how the scale up or down would actually work.

- At API layer, for an Initial burst of traffic, Lambda function's concurrency (number of requests being served at any given time) can reach an initial level of between 500 to 3000 (varies per region)
- After the initial burst, your function's **concurrency** can scale by an **additional 500 instances each minute**. This continues until there are enough instances to serve all the requests, or a concurrency limit is reached.

We can opt for Reserved Concurrency (guarantees the maximum number of concurrent instances) or Provisioned Concurrency (prepares number of execution environment for all requests)

Limitations

Serverless compute can not be suitable for all kinds of systems. The system design pattern largely varies based on use case.

If system has not be thoroughly understood and not designed adequately then Serverless Systems can face challenges like:

- Lambda throttling (If there is not enough concurrency to server all the requests, then additional requests will be throttled and will be retried)

- Function timeouts, dependent AWS resources/services might face throttling.
- Detailed Telemetry.

Useful link to refer:

<https://docs.aws.amazon.com/lambda/latest/dg/gettingstarted-limits.html>

Technical Debt

1. Segregation of Standard Network terraform (e.g. VPC, Endpoints, Route Tables, NACLs, ALB etc) and Application specific Infrastructure (ALB Target Groups, Lambda, DynamoDB etc)
2. Flexibility to deploy the additional endpoint which is dependent on the above point.
3. Creation of a pipeline on Azure DevOps for portability.
4. Demonstration of additional regional deployment which will require a hosted zone with the specific domain.
5. Test Automation

References

- <https://docs.aws.amazon.com/vpc/latest/userguide/managed-prefix-lists.html>
- <https://docs.aws.amazon.com/general/latest/gr/rande.html>
- <https://docs.aws.amazon.com/lambda/latest/dg/configuration-vpc.html>
- <https://aws.amazon.com/blogs/aws/new-vpc-endpoints-for-dynamodb/>
- https://docs.amazonaws.cn/en_us/amazondynamodb/latest/developerguide/vpc-endpoints-dynamodb.html
- <https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/vpc-endpoints-dynamodb-tutorial.html>
- <https://aws.amazon.com/blogs/networking-and-content-delivery/lambda-functions-as-targets-for-application-load-balancers/>
- <https://docs.aws.amazon.com/lambda/latest/dg/monitoring-functions-access-metrics.html>
- <https://docs.aws.amazon.com/lambda/latest/dg/runtimes-extensions-api.html>
- <https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/GettingStarted.Python.03.html>

- <https://boto3.amazonaws.com/v1/documentation/api/latest/reference/services/dynamodb.html#service-resource>
- <https://docs.aws.amazon.com/general/latest/gr/aws-ip-ranges.html>