List of resources provisioned through Cloudformation templates:

Parameters given as user input

1) Environment tag
2) Virtual private cloud id (list will be displayed)
3) Private Subnet ids (list will be displayed)

Aurora:

4) Aurora DB port
5) Aurora KMS key id
6) Subnet IDs for opensearch cluster (list will be displayed)

Opensearch:

7) Opensearch KMS key id
8) Opensearch instance count
9) Opensearch instance type
10) Opensearch Volume type
11) Opensearch Volume size
12) Opensearch availability zone count

Elasticache-memcached

13) Elasticache node type
14) Elasticache parameter group name
15) Elasticache cluster name
16) Number of nodes

S3 bucket

17) Bucket name
18) Logging bucket name
19) Retention policy for Glacier class
20) Retention policy for Deep archive class
21) KMS key id

SNS

22) KMS key id

Backups

23) KMS key id

Elasticache-Redis

24) KMS key id

Roles created:

1) EFS
2) Aurora
3) EKS
4) Fargate profile
5) Lambda execution
6) Role for aurora IAM based authentication
7) Role for enhanced monitoring
8) Backups

Security groups created:

1) Cluster
2) EFS
3) Aurora
4) Elasticache-memached
5) Elasticache-redis

1) Elastic Kubernetes Service cluster

Description : An EKS cluster with fargate profile

Parameters:

Version:1.27 (latest)
Subnets: Private subnets in vpc (Parameter inputs for Cloud formation)
Logging: API and audit logs enabled (as per recommendation)
Endpoint private access: true (as per ppt)
Endpoint public access: false
Metrics: CPU and memory (enabled by default)

Integration:

Created a security group for eks cluster with ingress and egress rule
This security group is added is ingress reference in EFS, Aurora, Memcached and Redis
It is also added as a reference security group for Opensearch cluster

Roles:

A specific role created (EKSClusterRole) with policy references for:

1) EKS cluster policy
2) EKS service policy
3) ECR Read only

Template:

a) Resource

```
Cluster:
  Type: AWS::EKS::Cluster
  Properties:
    Name: Dev-eks-cluster
    RoleArn: !GetAtt EKSClusterRole.Arn
    Version: '1.27'
    Logging:
      ClusterLogging:
        EnabledTypes:
        - Type: api
        - Type: audit
    ResourcesVpcConfig:
      SecurityGroupIds:
      - !Ref ClusterSecurityGroup
      SubnetIds: !Ref PrivateSubnetIds
      EndpointPrivateAccess: true
      EndpointPublicAccess: false
    Tags:
    - Key: "Environment"
      Value: !Ref Environment
```

b) Role

```yaml
EKSClusterRole:
  Type: AWS::IAM::Role
  Properties:
    RoleName: EKS-ClusterRole
    AssumeRolePolicyDocument:
      Version: '2012-10-17'
      Statement:
      - Effect: Allow
        Principal:
          Service:
          - eks.amazonaws.com
        Action: sts:AssumeRole
    ManagedPolicyArns:
    - arn:aws:iam::aws:policy/AmazonEKSClusterPolicy
    - arn:aws:iam::aws:policy/AmazonEKSServicePolicy
    - arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryReadOnly
    - arn:aws:iam::942417914032:policy/devbursa-IAMRDSpolicy-1W6EGI8BKBVQN
    Tags:
    - Key: "Environment"
      Value: !Ref Environment
```

2) Fargate profile

To run containers without actually having to manage EC2 instances

Parameters:

Depends on EKS cluster creation

Pod execution role: the components that run on the Fargate infrastructure must make calls to AWS APIs

Subnets: private subnets

Namespaces: default, kube-system, dev

Integration:

Added pod execution role policy to user created for EFS. (required for mounting EFS to EKS)

Role:

A role created which allows privileges to eks-fargate-pods.amazonaws.com service, with AWS Fargate pod execution policy as reference

Template:

a) Resource

```
FargateProfile:
  Type: AWS::EKS::FargateProfile
  DependsOn: Cluster
  Properties:
    ClusterName: !Ref Cluster
    FargateProfileName: Dev-fargate-profile
    PodExecutionRoleArn: !GetAtt FargateProfileRole.Arn
    Subnets: !Ref PrivateSubnetIds
    Selectors:
    - Namespace: default
    - Namespace: kube-system
    - Namespace: dev
    Tags:
    - Key: "Environment"
      Value: !Ref Environment
```

b) Role

```
  FargateProfileRole:
    Type: AWS::IAM::Role
    Properties:
      RoleName: FargateProfile-Role
      AssumeRolePolicyDocument:
        Version: '2012-10-17'
        Statement:
        - Effect: Allow
          Principal:
            Service:
            - eks-fargate-pods.amazonaws.com
          Action: sts:AssumeRole
      ManagedPolicyArns:
      - arn:aws:iam::aws:policy/AmazonEKSFargatePodExecutionRolePolicy
      Tags:
      - Key: "Environment"
        Value: !Ref Environment
```

3) Elastic File system

Description: EFS creation for providing storage to eks

Parameters:

Performance mode: General purpose
Throughput mode: Bursting
Encryption enabled: true
KMS Key: user input
Lifecycle policy : hardcoded

Integration:

Added two mount targets with reference to the above file system ids and private subnets with separate security group

Role:

Created a custom role allowing privileges to elasticfilesystem.amazonaws.com service, with managed policies for KMS and Fargate pod execution role policy

Template:

a) Resource

```
FileSystem:
  Type: AWS::EFS::FileSystem
  Properties:
    FileSystemTags:
    - Key: Name
      Value: "DevEFS"
    PerformanceMode: "generalPurpose"
    ThroughputMode: "bursting"
    Encrypted: true
    KmsKeyId: !Ref EFSKmsKey
    LifecyclePolicies:
    - TransitionToIA: AFTER_14_DAYS
```

b) Mount targets

```yaml
MountTargetResource2:
  Type: AWS::EFS::MountTarget
  Properties:
    FileSystemId: !Ref FileSystem
    SubnetId: !Select [1, !Ref PrivateSubnetIds]
    SecurityGroups:
    - !Ref EfsSecurityGroup

MountTargetResource3:
  Type: AWS::EFS::MountTarget
  Properties:
    FileSystemId: !Ref FileSystem
    SubnetId: !Select [2, !Ref PrivateSubnetIds]
    SecurityGroups:
    - !Ref EfsSecurityGroup
```

c) Role

```yaml
DevEFSRole:
  Type: AWS::IAM::Role
  Properties:
    RoleName: DevEFSRole
    AssumeRolePolicyDocument:
      Version: '2012-10-17'
      Statement:
      - Effect: Allow
        Principal:
          Service:
          - elasticfilesystem.amazonaws.com
        Action: sts:AssumeRole
    ManagedPolicyArns:
    - arn:aws:iam::aws:policy/AWSKeyManagementServicePowerUser
    - arn:aws:iam::aws:policy/AmazonEKSFargatePodExecutionRolePolicy
    Tags:
    - Key: "Environment"
      Value: !Ref Environment
```

4) Relational Database Service

Description: Aurora PostgreSQL Database cluster with one instance

Parameters:

Engine: Aurora Postgresql
Version: 15.3
Subnet group: Created a subnet group for private subnets
Password: Using a secret from secrets manager
Port: User input
Backup retention period: 7 days
Encryption: enabled
KMS key id : user input
IAM based Database authentication : enabled
Security group : Aurora security group created separately
Class: db.t3.large
Auto minor version upgrade : enabled
Publicly accessible: false
Performance insights: enabled
Enhanced monitoring : configured
Cloudwatch : metrics and logs configured

Integration:

Resources related to the secret (attachment, secrets policy, Lambda function for rotation, Rotation function, permission and schedule)
Policy for allowing RDS database authentication based on IAM roles
Parameter group
Log group (for cloudwatch)
Log stream
EKS cluster security group added as ingress

Roles and policies:

Lambda execution role (for rotating password)
Role and policy for allowing RDS database authentication based on IAM roles
Role for enhanced monitoring

Template:

a) Resource

```yaml
AuroraCluster:
  Type: AWS::RDS::DBCluster
  Properties:
    Engine: aurora-postgresql
    EngineVersion: '15.3'
    DBClusterIdentifier: Dev-aurora-cluster
    DBSubnetGroupName: !Ref DBSubnetGroup
    DatabaseName: mydatabase
    MasterUsername: !Join ['', ['{{resolve:secretsmanager:', !Ref DevAuroraDBSecret, ':SecretString:username}}']]
    MasterUserPassword: !Join ['', ['{{resolve:secretsmanager:', !Ref DevAuroraDBSecret, ':SecretString:password}}']
    StorageEncrypted: true
    Port: !Ref AuroraDBPort
    BackupRetentionPeriod: 7
    PreferredBackupWindow: '00:00-00:30'
    KmsKeyId: !Ref AuroraDBKmsKeyID
    EnableIAMDatabaseAuthentication: true

    AssociatedRoles:
    - RoleArn: !GetAtt DevAuroraRole.Arn
      FeatureName: s3Import
    VpcSecurityGroupIds:
    - !Ref AuroraSecurityGroup # Attach AuroraSecurityGroup
    Tags:
    - Key: "Environment"
      Value: !Ref Environment
```

b) Database instance

```yaml
DBInstance:
  Type: AWS::RDS::DBInstance
  Properties:
    Engine: aurora-postgresql
    EngineVersion: '15.3'
    DBInstanceIdentifier: Dev-Aurora-DB-instance
    DBInstanceClass: db.t3.large
    DBClusterIdentifier: !Ref AuroraCluster
    AutoMinorVersionUpgrade: true
    PubliclyAccessible: false
    StorageType: aurora
    EnablePerformanceInsights: true
    PerformanceInsightsRetentionPeriod: 7
    DBParameterGroupName: !Ref DBParameterGroup
    MonitoringInterval: 60
    MonitoringRoleArn: !GetAtt ['EnhancedMonitoringRole', 'Arn']
    Tags:
    - Key: "Environment"
      Value: !Ref Environment
```

c) RDS role for IAM authentication

```yaml
IAMRDSRole:
  Type: AWS::IAM::Role
  Properties:
    RoleName: IAMRDSRole
    AssumeRolePolicyDocument:
      Version: '2012-10-17'
      Statement:
        - Effect: Allow
          Principal:
            Service: rds.amazonaws.com
          Action: sts:AssumeRole
    ManagedPolicyArns:
      - !Ref IAMRDSpolicy
```

d) Enhanced monitoring role

```yaml
EnhancedMonitoringRole:
  Type: AWS::IAM::Role
  Properties:
    AssumeRolePolicyDocument:
      Version: '2012-10-17'
      Statement:
      - Sid: ''
        Effect: Allow
        Principal:
          Service: monitoring.rds.amazonaws.com
        Action: sts:AssumeRole
    ManagedPolicyArns:
    - arn:aws:iam::aws:policy/service-role/AmazonRDSEnhancedMonitoringRole
    Path: "/"
```

5) Opensearch

Description: Opensearch domain with dedicated master nodes disabled and 2 data nodes

Parameters:

Instance count : user input
Instance type : user input
Dedicated master: disabled
Multi AZ standby : disabled
EBS : enabled
Subnets: user input (private)
Encryption at rest: enabled
KMS key id : user input
Logging : application, index slow and search slow logs enabled

Integration:

Log groups for application, index and search logs

Roles and policies:

A custom policy created for sending logs to different log groups (cloudwatch)

Template:

a) Resource

```
DevOpenSearchCluster:
  Type: 'AWS::OpenSearchService::Domain'
  Properties:
    ClusterConfig:
      InstanceCount: !Ref InstanceCount
      InstanceType: !Ref InstanceType
      DedicatedMasterEnabled: false
      ZoneAwarenessEnabled: true
      ZoneAwarenessConfig:
        AvailabilityZoneCount: !Ref AvailabilityZoneCount
      MultiAZWithStandbyEnabled: false
    EBSOptions:
      EBSEnabled: true
      VolumeType: !Ref VolumeType
      VolumeSize: !Ref VolumeSize
    VPCOptions:
      SubnetIds: !Ref SubnetIdsForOpenSearch
      SecurityGroupIds:
      - !Ref ClusterSecurityGroup
    EncryptionAtRestOptions:
      Enabled: true
      KmsKeyId: !Ref OpenSearchKMSKeyID
    Tags:
    - Key: "Environment"
      Value: !Ref Environment
    LogPublishingOptions:
      ES_APPLICATION_LOGS:
        CloudWatchLogsLogGroupArn: !GetAtt   ApplicationLogsCloudWatchLogGroup.Arn
        Enabled: true
      INDEX_SLOW_LOGS:
        CloudWatchLogsLogGroupArn: !GetAtt IndexSlowLogsCloudWatchLogGroup.Arn
        Enabled: true
      SEARCH_SLOW_LOGS:
        CloudWatchLogsLogGroupArn: !GetAtt SearchSlowLogsCloudWatchLogGroup.Arn
        Enabled: true
```

b) Policy

```
OpenSearchCloudWatchPolicy:
  Type: AWS::Logs::ResourcePolicy
  Properties:
    PolicyName: OpenSearchCloudWatchPolicy
    PolicyDocument: !Sub |
      {
        "Version": "2012-10-17",
        "Statement": [
          {
            "Sid": "AllowOpenSearchToPublishLogs",
            "Effect": "Allow",
            "Principal": {
              "Service": "es.amazonaws.com"
            },
            "Action": [
              "logs:PutLogEvents",
              "logs:CreateLogStream"
            ],
            "Resource": [
              "arn:aws:logs:${AWS::Region}:${AWS::AccountId}:log-group:/aws/opensearch/${Environment}/application-l
              "arn:aws:logs:${AWS::Region}:${AWS::AccountId}:log-group:/aws/opensearch/${Environment}/index-slow-lo
              "arn:aws:logs:${AWS::Region}:${AWS::AccountId}:log-group:/aws/opensearch/${Environment}/search-slow-l

            ]
          }
        ]
      }
```

6) Elasticache - Memcached

Description: Memcached cluster with a single node

Parameters:

Node type: user input
Parameter group name : user input (default.memcached1.6)
Engine: memcached
Version: 1.6.17
Number of nodes: user input
Subnet group: created
Security group: separate and created

Integration:

Subnet group : mandatory parameter for elasticache
Security group : Port 11211 in ingress and EKS security group

Template:

a) Resource

```
CacheCluster:
  Type: AWS::ElastiCache::CacheCluster
  Properties:
    CacheNodeType: !Ref CacheNodeType
    CacheParameterGroupName: !Ref CacheParameterGroupName
    ClusterName: !Ref MemCacheClusterName
    Engine: memcached
    EngineVersion: 1.6.17
    NumCacheNodes: !Ref NumMemCacheNodes
    TransitEncryptionEnabled: true
    CacheSubnetGroupName: !Ref CacheSubnetGroup
    VpcSecurityGroupIds:
    - !Ref CacheSecurityGroup
    Tags:
    - Key: "Environment"
      Value: !Ref Environment
```

b) Subnet group

```
CacheSubnetGroup:
  Type: AWS::ElastiCache::SubnetGroup
  Properties:
    CacheSubnetGroupName: MyCacheSubnetGroup
    Description: Subnet group for ElastiCache
    SubnetIds: !Ref PrivateSubnetIds
    Tags:
    - Key: "Environment"
      Value: !Ref Environment
```

c) Security group

```
CacheSecurityGroup:
  Type: AWS::EC2::SecurityGroup
  Properties:
    GroupDescription: Security group for the Memcache cluster
    VpcId: !Ref VPCId
    SecurityGroupIngress:
    - IpProtocol: tcp
      FromPort: 11211
      ToPort: 11211
      SourceSecurityGroupId: !Ref ClusterSecurityGroup
```

7) Elasticache-Redis

Description: A redis replication group with 1 shard

Parameters:

Engine: redis
Cache node type: cache.t4g.micro
No. of shards: 1
Subnet group name: created
Engine : 7.0
AutomaticFailoverEnabled: false
TransitEncryptionEnabled: true
AtRestEncryptionEnabled: true
AutoMinorVersionUpgrade: true
Log configuration : configured slow and engine logs

Integration:

Slow and engine log groups
Subnet group which includes private subnets given through user inputs
Security group with ingress 6379 and EKS cluster security group

Template:

a) Resource

```yaml
RedisReplicationGroup:
  Type: AWS::ElastiCache::ReplicationGroup
  Properties:
    ReplicationGroupDescription: Redis Replication Group
    Engine: redis
    CacheNodeType: cache.t4g.micro
    NumNodeGroups: 1 # Number of shards (primary and replica)
    CacheSubnetGroupName: !Ref RedisCacheSubnetGroup
    EngineVersion: 7.0
    AutomaticFailoverEnabled: false
    TransitEncryptionEnabled: true # Enable encryption in transit
    AtRestEncryptionEnabled: true # Enable encryption at rest
    AutoMinorVersionUpgrade: true
    SecurityGroupIds:
    - !Ref RedisSecurityGroup
    Tags:
    - Key: "Environment"
      Value: !Ref Environment
    LogDeliveryConfigurations:
    - LogType: slow-log
      DestinationType: cloudwatch-logs
      LogFormat: json
      DestinationDetails:
        CloudWatchLogsDetails:
          LogGroup: !Sub "/aws/elasticache/${AWS::StackName}/redis-slow-logs"
    - LogType: engine-log
      DestinationType: cloudwatch-logs
      LogFormat: json # Specify the log format here (e.g., json or text)
```

b) Subnet group

```yaml
RedisCacheSubnetGroup:
  Type: AWS::ElastiCache::SubnetGroup
  Properties:
    CacheSubnetGroupName: RedisCacheSubnetGroup
    Description: Subnet group for RedisCache
    SubnetIds: !Ref PrivateSubnetIds
    Tags:
    - Key: "Environment"
      Value: !Ref Environment
```

c) Security group

```yaml
RedisSecurityGroup:
  Type: AWS::EC2::SecurityGroup
  Properties:
    GroupDescription: Security group for the Redis cluster
    VpcId: !Ref VPCId
    SecurityGroupIngress:
    - IpProtocol: tcp
      FromPort: 6379
      ToPort: 6379
      SourceSecurityGroupId: !Ref ClusterSecurityGroup
```

d) Log groups

```yaml
CloudWatchSlowLogGroup:
  Type: AWS::Logs::LogGroup
  Properties:
    LogGroupName: !Sub "/aws/elasticache/${AWS::StackName}/redis-slow-logs"
    RetentionInDays: 14

CloudWatchEngineLogGroup:
  Type: AWS::Logs::LogGroup
  Properties:
    LogGroupName: !Sub "/aws/elasticache/${AWS::StackName}/redis-engine-logs"
    RetentionInDays: 14
```

8) S3 bucket

Description: A private S3 bucket with access logging enabled

Parameters:

Access control : private

Versioning: enabled

Logging configuration : configured access logs to fall in another s3 bucket

Lifecycle policy : Glacier transition rule with user inputs for deep archive and glacier transitions (number of days)

Bucket encryption: Configured to accept user given kms key id

Integration:

Configured another bucket for obtaining access logs of the source bucket

Roles and policies:

Bucket policy which only accepts SSL based connections only

Logging bucket policy

Template

a) Resource

```
S3Bucket:
  Type: 'AWS::S3::Bucket'
  Properties:
    BucketName: !Ref BucketName
    AccessControl: Private
    VersioningConfiguration:
      Status: Enabled
    LoggingConfiguration:
      DestinationBucketName: !Ref LogBucketName
      LogFilePrefix: dev-logs
    LifecycleConfiguration:
      Rules:
      - Id: GlacierTransitionRule
        Status: Enabled
        Transitions:
        - StorageClass: DEEP_ARCHIVE
          TransitionInDays: !Ref DeepArchiveRetentionPolicyDays
        - StorageClass: GLACIER
          TransitionInDays: !Ref RetentionPolicyDays
    BucketEncryption:
      ServerSideEncryptionConfiguration:
      - ServerSideEncryptionByDefault:
          SSEAlgorithm: 'aws:kms'
          KMSMasterKeyID: !Ref S3KMSKeyID
    Tags:
    - Key: "Environment"
      Value: !Ref Environment
```

b) Bucket policy

```yaml
S3SourceBucketPolicy:
  Type: "AWS::S3::BucketPolicy"
  Properties:
    Bucket: !Ref BucketName
    PolicyDocument:
      Version: "2012-10-17"
      Statement:
      - Sid: "EnforceSSLOnly"
        Effect: "Deny"
        Principal: "*"
        Action: "s3:*"
        Resource: !Join ["", ["arn:aws:s3:::", !Ref S3Bucket, "/*"]]
        Condition:
          Bool:
            "aws:SecureTransport": "false"
```

c) Logging bucket policy

```yaml
S3BucketPolicy:
  Type: 'AWS::S3::BucketPolicy'
  Properties:
    Bucket: !Ref LogBucketName
    PolicyDocument:
      Version: 2012-10-17
      Statement:
      - Action:
        - 's3:PutObject'
        Effect: Allow
        Principal:
          Service: logging.s3.amazonaws.com
        Resource: !Join
        - ''
        - - 'arn:aws:s3:::'
          - !Ref LogBucketName
          - /*
        Condition:
          ArnLike:
            'aws:SourceArn': !GetAtt
            - S3Bucket
            - Arn
          StringEquals:
            'aws:SourceAccount': !Sub '${AWS::AccountId}'
```

9) Backups

Description: Roles, vaults, plans and selection for backing up EFS and RDS

Parameters:

a) Backup plans (configured as per requirement)
   Rules
   Cron expression
b) Backup vault
   Encryption key: user input
c) Backup selection: configured target resources (file systems and RDS instances)

Roles and policies:

Created a separate role which allows privileges to backup.amazonaws.com service

Integrations:

Linking rules to backup plan
Target backup vault in backup plan
Attaching resources to backup plan using backup selection

Template:

a) Backup vault

```
EFSTargetBackupVault:
  Type: "AWS::Backup::BackupVault"
  Properties:
    BackupVaultName: "EFSBackupVault1"
    EncryptionKeyArn:
      Fn::Join:
      - ""
      - - "arn:aws:kms:"
        - !Ref "AWS::Region"
        - ":"
        - !Ref "AWS::AccountId"
        - ":key/"
        - !Ref BackupsKMSKeyID
```

b) Backup selection

```
BackupSelection:
  Type: AWS::Backup::BackupSelection
  Properties:
    BackupPlanId: !Ref EFSBackupPlan
    BackupSelection:
      SelectionName: EFSBackupselection
      IamRoleArn: !GetAtt BackupRole.Arn
      Resources:
      - !Sub "arn:aws:s3:::${FileSystem}"
```

c) Role:

```
BackupRole:
  Type: AWS::IAM::Role
  Properties:
    AssumeRolePolicyDocument:
      Version: '2012-10-17'
      Statement:
      - Effect: Allow
        Principal:
          Service: backup.amazonaws.com
        Action: sts:AssumeRole
```

d) Plan:

```
EFSBackupPlan:
  Type: "AWS::Backup::BackupPlan"
  Properties:
    BackupPlan:
      BackupPlanName: "EFSBackupPlan"
      BackupPlanRule:

      - RuleName: "EFSDailyBackupRule"
        TargetBackupVault: !Ref EFSTargetBackupVault
        ScheduleExpression: "cron(0 0 ? * * *)" # Daily at midnight
        Lifecycle:
          DeleteAfterDays: 35
```

Similar plan, role, selection and vault for Aurora RDS