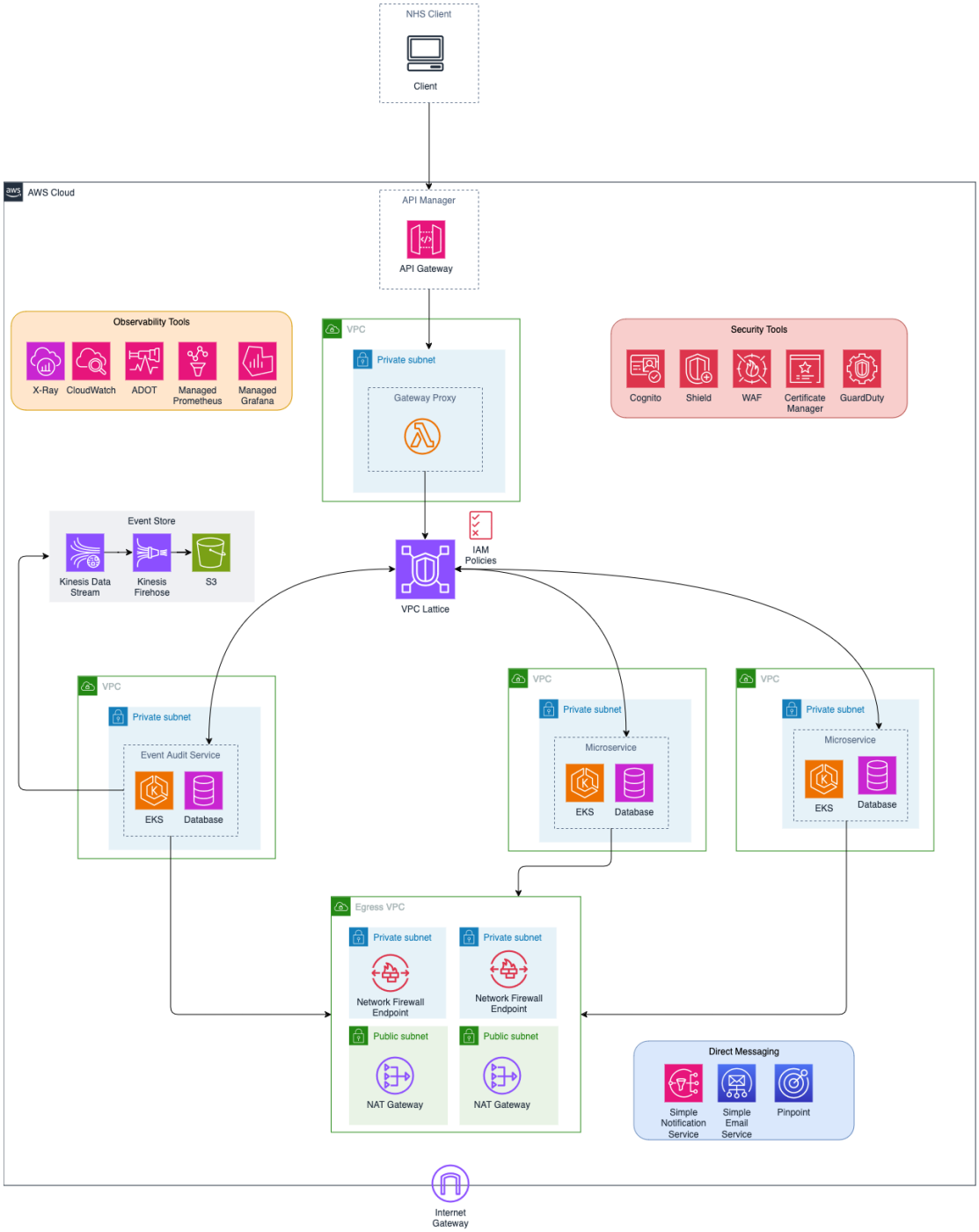
DPDP AWS Design Patterns

Below is high level description and illustration of design patterns that can be used in the design of the DPDP on AWS Cloud. These patterns have been categorised into various sections that look at Networking, Microservice design, API Management, Event Audit, Security, High Availability and Disaster Recovery, and External communication. Below is a diagram putting these design patterns together



Networking and service discovery

During our conversations one of the things that was pointed out is how do we utilise as much managed services as possible and allow the developers to focus on building DPDP services and apps. One of the key areas to accelerate developer productivity is to minimise the time spent on networking and connectivity between services.

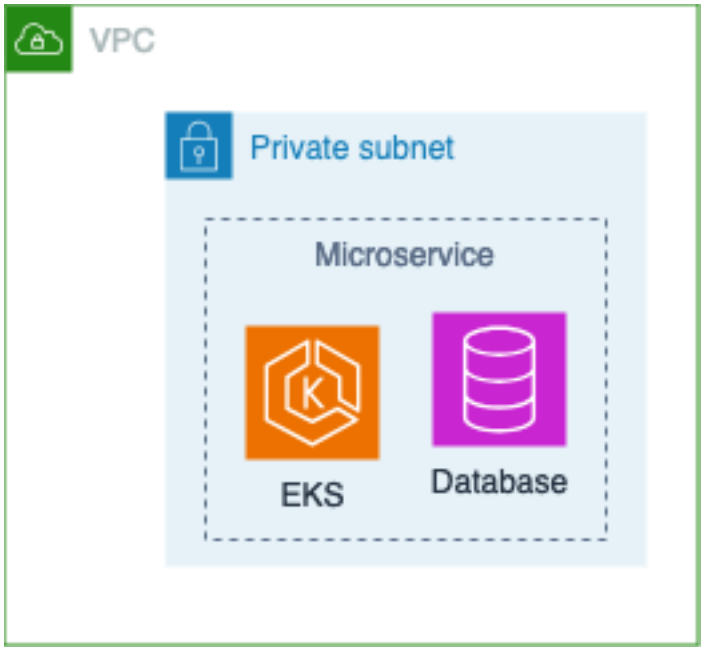
Utilising Amazon VPC Lattice for service-to-service networking will mean you will be able to automatically manage network connectivity and application layer routing between services across different VPCs and AWS accounts. No need to manage the underlying network connectivity or frontend load balancers. You can also start implement some zero trust principals between services as VPC Lattice integrates with IAM and provides you authentication and authorization capabilities between your services. Ensuring service can communicate with services that they need to. Below is a link to find out more about Amazon VPC Lattice
https://aws.amazon.com/vpc/lattice/features/

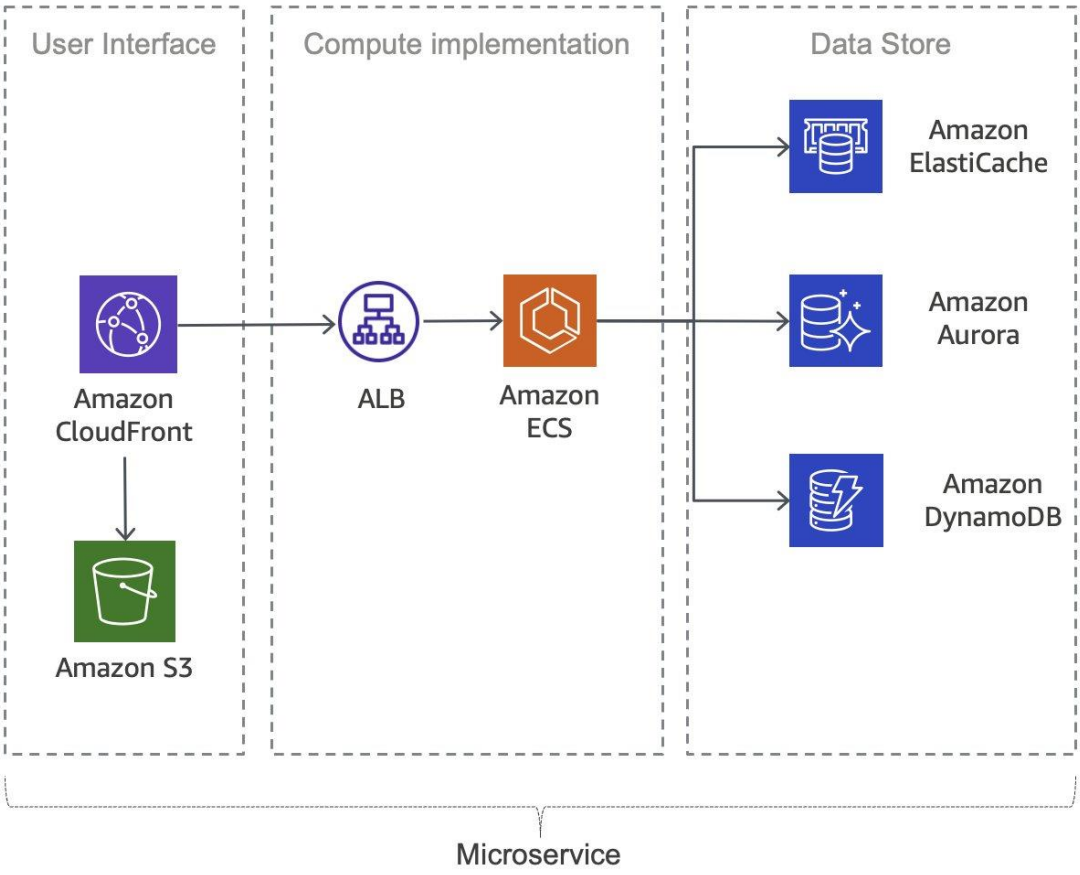
Also below is link to some reference architectures in implementing Amazon VPC Lattice
https://docs.aws.amazon.com/architecture-diagrams/latest/amazon-vpc-lattice-use-cases/amazon-vpc-lattice-use-cases.html

Services Design Pattern

Microservice Design

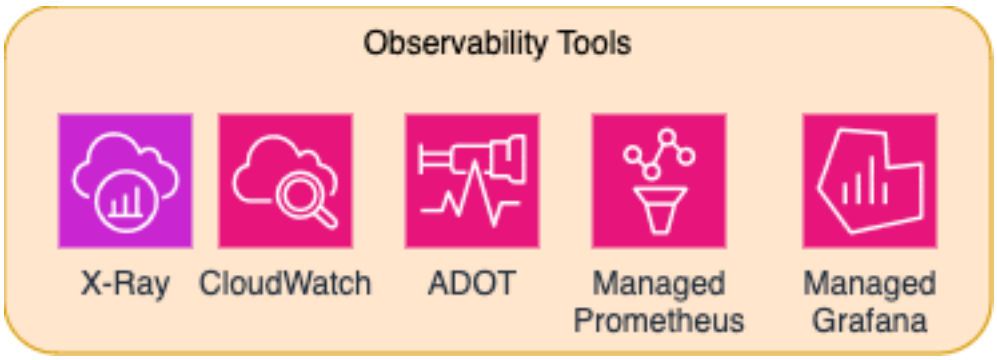
Given that services will be containerised, you can leverage either Amazon Elastic Container Service (Amazon ECS) or Amazon Elastic Kubernetes Service (Amazon EKS) for container orchestration. Although ECS is an easier to use container orchestration service than EKS, given the wider adoption and familiarity of Kubernetes, it would probably be best to stick with EKS. I would also recommend the use of a purpose-built AWS managed database service rather than hosting on Kubernetes for any persistent data store, as a managed database service such as DynamoDB (for NoSQL key-value datastore) or Amazon Aurora (RDMS) will provide you with less operational overhead such as database backups and patching. The choice of database will depend on each services need.


VPC
Private subnet
Microservice
EKS
Database


User Interface
Compute implementation
Data Store
Amazon CloudFront
Amazon S3
ALB
Amazon ECS
Amazon ElastiCache
Amazon Aurora
Amazon DynamoDB
Microservice

Observability and Telemetry

To enable observability within services and across requests, AWS provides a number of tools to instrument, collect, analyse and visualise metrics events, logs and traces (MELT) from microservices.



Starting with logs and metrics, Amazon CloudWatch can meet most logging and monitoring requirements, and provides a reliable, scalable, and flexible solution. Many AWS services automatically provide CloudWatch metrics, in addition to CloudWatch logging integration for monitoring and analysis. CloudWatch also provides agents and log drivers to support a variety of compute options such as servers (both in the cloud and on premises), containers, and serverless computing. These CloudWatch agents run on your containers and will be able to collect system and application logs and route them to Cloudwatch. Alternative to the CloudWatch agent, you also have AWS FireLens Container which runs alongside your containers in a sidecar config and can route your logs to CloudWatch, S3, OpenSearch or 3rd Party Logging tools (E.g. Datadog, Splunk). So for flexibility of log destination and preprocessing/filtering, you can utilise the Firelens container which utilises Fluent Bit or Fluentd. This is a bit more complex to manage than the CloudWatch agent and comes with additional cost implications as it's an additional container to host.

For application tracing, AWS X-Ray helps developers analyse and debug production, distributed applications, such as those built using a microservices architecture. With X-Ray, you can understand how your application and its underlying services are performing to identify and troubleshoot the root cause of performance issues and errors. X-Ray provides an end-to-end view of requests as they travel through your application, and shows a map of your application's underlying components. With X-Ray you can create service maps, identify errors and bugs and build your own analysis and visualization apps.

For Events, Amazon EventBridge is a service that uses events to connect application components together, making it easier for you to build event-driven applications. You can use EventBridge to route events from your microservices, AWS services, and third-party software to consumer applications across your Infrastructure. EventBridge provides simple and consistent ways to ingest, filter, transform, and deliver events so you can build applications quickly. There is another approach for managing events in the Events Audit Service which I highlight later on in the document

For Amazon EKS, an often-preferred choice is Prometheus for providing comprehensive monitoring and alerting capabilities. It's typically coupled with Grafana for intuitive metrics visualization. Amazon Managed Service for Prometheus (AMP) offers a monitoring service fully compatible with Prometheus, letting you oversee containerized applications effortlessly. Additionally, Amazon Managed Grafana (AMG) simplifies the analysis and visualization of your metrics, eliminating the need for managing underlying infrastructure. I believe Prometheus and Grafana are currently being used on the NDP so this could serve as an alternative to Cloudwatch logs.

There is also the option of utilising AWS Distro for OpenTelemetry (ADOT) which is a secure, production-ready, AWS-supported distribution of the OpenTelemetry project. With AWS Distro for OpenTelemetry, you can:

- Instrument your applications just once to send correlated metrics and traces to multiple monitoring solutions and use auto-instrumentation agents to collect traces without changing your code.
- Collect metadata from your AWS resources and managed services, so you can correlate application performance data with underlying infrastructure data, reducing the mean time to problem resolution.
- Use AWS Distro for OpenTelemetry to observe your applications running on Amazon Elastic Compute Cloud (EC2), Amazon Elastic Container Service (ECS), and Amazon Elastic Kubernetes Service (EKS) on EC2, and AWS Fargate, as well as on-premises.

Security

There are a number of AWS security services that can be used within DPDP to improve the security posture of the platform.
Starting with Amazon Cognito which is an identity platform for web and mobile apps. It’s a user directory, an authentication server, and an authorization service for OAuth 2.0 access tokens and AWS credentials. With Amazon Cognito, you can authenticate and authorize users from the built-in user directory, from your enterprise directory, and from consumer identity providers like Google and Facebook. As an alternative to using IAM roles and policies or Lambda authorizers (mentioned in the next section), you can use an Amazon Cognito user pool to control who can access your API in Amazon API Gateway.

To protect your publicly accessible REST APIs, AWS WAF can be enabled. AWS WAF is a web application firewall that helps protect web applications and APIs from attacks. It enables you to configure a set of rules called a web access control list (web ACL) that allow, block, or count web requests based on customizable web security rules and conditions that you define. You can use AWS WAF to protect your API Gateway REST API from common web exploits, such as SQL injection and cross-site scripting (XSS) attacks.

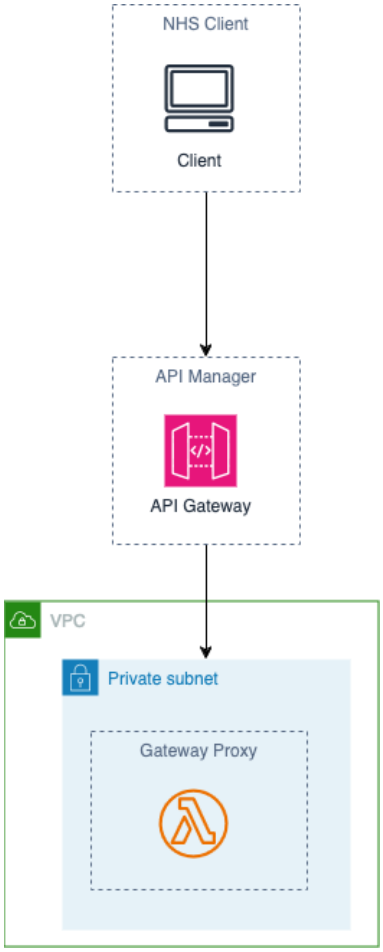
To further protect your APIs from DDoS attacks you can use AWS Shield and Shield Advanced to improve your security posture with DDoS detection, mitigation and response capabilities.

You can stipulate withing VPC Lattice policies that traffic between services are encrypted and use AWS Certificate manager to handle the complexity of creating, storing, and renewing public and private SSL/TLS X.509 certificates and keys that protect these microservices.

Lastly, we can enable Amazon GuardDuty which is a threat detection service that continuously monitors, analyses, and processes AWS data sources and logs in your AWS environment. GuardDuty uses threat intelligence feeds, such as lists of malicious IP addresses and domains, file hashes, and machine learning (ML) models to identify suspicious and potentially malicious activity in your AWS environment.

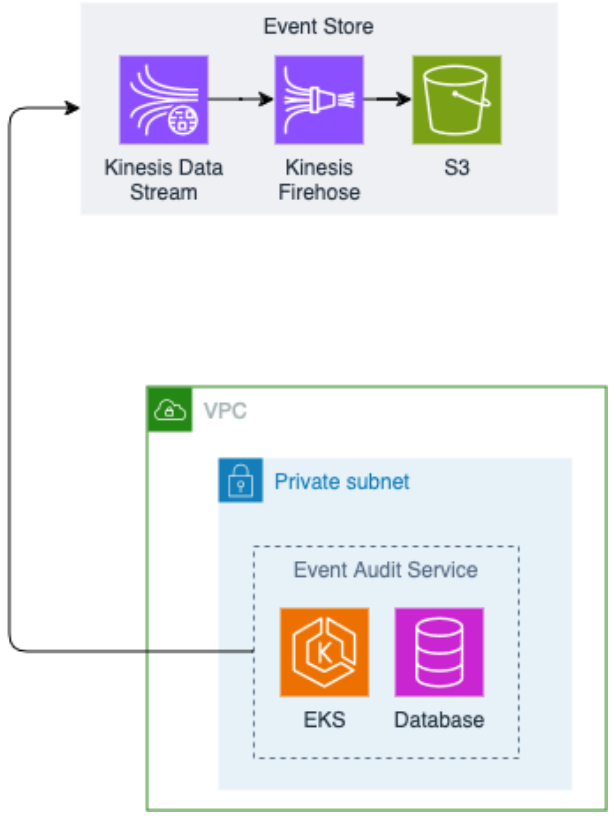
API Manager and Authentication and Authorisation

As the DPDP application will require multiple client applications connecting to a large microservices-based backend, it is recommended to use the API gateway pattern. The pattern provides a reverse proxy to redirect or route requests to your internal microservices endpoints. An API gateway provides a single endpoint or URL for the client applications, and it internally maps the requests to internal microservices. A layer of abstraction is provided by hiding certain implementation details (for example, the Lambda function name and version), and additional functionality can also be added on top of the backend service, such as response and request transformations, endpoint access authorization, or tracing. The suggested design below utilises Amazon API Gateway service for creating, publishing, maintaining, monitoring, and securing REST, HTTP, and WebSocket APIs at any scale. This is used in combination with AWS Lambda as a Lambda proxy. The Lambda proxy is situated within a private VPC with connectivity to other microservices (Via VPC Lattice) and therefore can be used to call these microservices. The lambda can also act as a Lambda Authorizer in which it can be used to control access to your API, providing you flexibility within your OAuth flows.



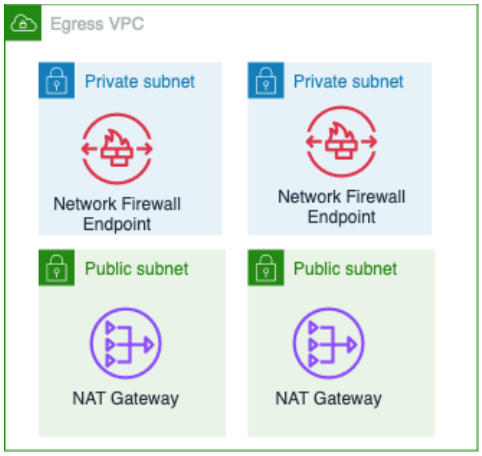
Audit Service

A common approach in managing changes across microservices is event sourcing. Every change in the application is recorded as an event, creating a timeline of the system's state. This approach not only helps debug and audit but also allows different parts of an application to react to the same events.
Event sourcing often works hand-in-hand with the Command Query Responsibility Segregation (CQRS) pattern, which separates data modification and data querying into different modules for better performance and security.
On AWS, you can implement these patterns using a combination of services. Amazon Kinesis Data Streams can serve as your central event store, while Amazon S3
provides a durable storage for all event records. The Event Audit Microservice Service (EKS with DynamoDB or Aurora) can handle and process these events sent by other microservices. The data store within the microservice can be used for query operations as these databases can be optimised for specific query requirements.



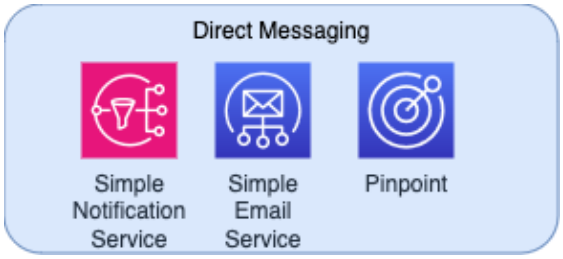
External Communication and Direct messaging

As DPDP will require external communications with services and users outside the environment, we can centralise these communications via an Egress VPC which will contain NAT Gateway devices. AWS recommends that you use NAT gateways because they provide better availability and bandwidth and require less effort on your part to administer. If you want to inspect and filter your outbound traffic, you can incorporate AWS Network Firewall with NAT gateway in your centralized egress architecture. AWS Network Firewall is a managed service that makes it easy to deploy essential network protections for all of your VPCs. It provides control and visibility to Layer 3-7 network traffic for your entire VPC. You can do URL/domain name, IP address, and content-based outbound traffic filtering to stop possible data loss, help meet compliance requirements, and block known malware communications.



https://docs.aws.amazon.com/whitepapers/latest/building-scalable-secure-multi-vpc-network-infrastructure/using-nat-gateway-with-firewall.html

For Direct Messaging and communication with service users, AWS provides a number of services to enable Email communication, Text, and mobile app notifications.
With Amazon SNS, you have the ability to send push notification messages directly to apps on mobile devices. Push notification messages sent to a mobile endpoint can appear in the mobile app as message alerts, badge updates, or even sound alerts. SNS also has capabilities to send Email and SMS messages but for more advance controls such as two-way messaging and HTML format emails, you have the option to use Amazon Pinpoint and Simple Email Service (SES) respectively



HA and DR

High Availability
Amazon EKS ensures high availability by running Kubernetes control and data plane instances across multiple Availability Zones. It automatically detects and replaces unhealthy control plane instances and provides automated version upgrades and patching.
Amazon ECR uses Amazon Simple Storage Service (Amazon S3) for storage to make your container images highly available and accessible. It works with Amazon EKS, Amazon ECS, and AWS Lambda, simplifying development to production workflow.
AWS Lambda operates in multiple Availability Zones, ensuring availability during service interruptions in a single zone. If connecting your function to a VPC, specify subnets in multiple Availability Zones for high availability.

Disaster Recovery
Microservices applications often follow the Twelve-Factor Application patterns, where processes are stateless, and persistent data is stored in stateful backing services like databases. This simplifies disaster recovery (DR) because if a service fails, it's easy to launch new instances to restore functionality.
Disaster recovery strategies for microservices should focus on downstream services that maintain the application's state, such as file systems, databases, or queues. You should plan for recovery time objective (RTO) and recovery point objective (RPO). In the existing architecture, the main components that store state are the databases and storage.
Depending on the choice of database, AWS has a number of backup and replication strategies to meet various RTOs and RPOs. The list of these can be seen in the link below

https://docs.aws.amazon.com/prescriptive-guidance/latest/strategy-database-disaster-recovery/choosing-database.html

https://docs.aws.amazon.com/whitepapers/latest/disaster-recovery-workloads-on-aws/disaster-recovery-options-in-the-cloud.html

Amazon S3 offers several features to help support your data resiliency and backup needs.

Lifecycle configuration

A lifecycle configuration is a set of rules that define actions that Amazon S3 applies to a group of objects. With lifecycle configuration rules, you can tell Amazon S3 to transition objects to less expensive storage classes, archive them, or delete them. For more information, see Managing your storage lifecycle.

Versioning

Versioning is a means of keeping multiple variants of an object in the same bucket. You can use versioning to preserve, retrieve, and restore every version of every object stored in your Amazon S3 bucket. With versioning, you can easily recover from both unintended user actions and application failures. For more information, see Using versioning in S3 buckets.

S3 Object Lock

You can use S3 Object Lock to store objects using a write once, read many (WORM) model. Using S3 Object Lock, you can prevent an object from being deleted or overwritten for a fixed amount of time or indefinitely. S3 Object Lock enables you to meet regulatory requirements that require WORM storage or simply to add an additional layer of protection against object changes and deletion. For more information, see Using S3 Object Lock.

Storage classes

Amazon S3 offers a range of storage classes to choose from depending on the requirements of your workload. The S3 Standard-IA and S3 One Zone-IA storage classes are designed for data you access about once a month and need milliseconds access. The S3 Glacier Instant Retrieval storage class is designed for long-lived archive data accessed with milliseconds access that you access about once a quarter. For archive data that does not require immediate access, such as backups, you can use the S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive storage classes. For more information, see Using Amazon S3 storage classes.