COMMUNITY DAY

SINGAPORE

aws

COMMUNITY DAY

A Deep Dive into AWS-native Microservice Routing

Architecting for Resilience, Scalability, and Security

Ananda Dwi Rahmawati | Aug, 2nd 2025

aws COMMUNITY DAY

Ananda Dwi Rahmawati

- Cloud & DevOps Engineer, Singapore
- AWS Container Hero
- Master of Computer Science - University of Texas at Austin
- https://linktr.ee/misskecupbung

aws COMMUNITY DAY

Agenda

- The Microservice Routing Challenge
- Key Takeaways
- Service Discovery & DNS
- Load Balancing & Edge Routing
- Security & Observability Patterns
- Demo

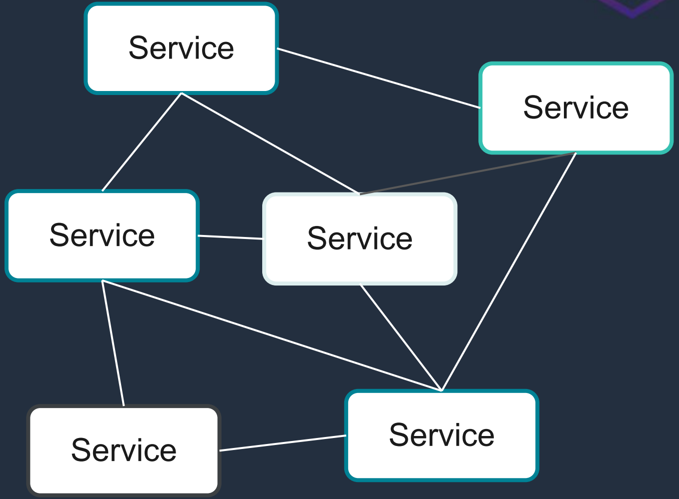
The Microservice Routing Challenge

aws COMMUNITY DAY

The Microservice Routing Challenge

In monoliths, routing was *straightforward*. In microservices, we may face challenges like:

- **Dynamic Endpoints**: How do services discover each other when instances are ephemeral and IPs change?
- **Resilience**: How do we handle failures, retries, and prevent cascading issues (circuit breaking)?
- **Traffic Management:** How do we roll out new versions (Canary/Blue-Green) and route by path, headers, or weight?
- **Security & Observability**: How do we secure inter-service traffic and trace requests across many services?



Services need to communicate with each other

aws COMMUNITY DAY

Key Takeaways

Dynamic Service Discovery with AWS Cloud Map + ECS/EKS

- Enable resilient, scalable service-to-service communication without hardcoded endpoints.

Advanced Traffic Routing with App Mesh

- Implement blue/green deployments, traffic shifting, retries, and timeouts using sidecar proxies.

API Gateway vs ALB: Ingress Decision

- Choose the right ingress based on protocol support, latency, pricing, and operational complexity.

Securing Microservice Communication

- Use IAM roles, security groups, and encryption to secure internal traffic and enforce zero-trust within your VPC and mesh.

Observability Across Microservice Routes

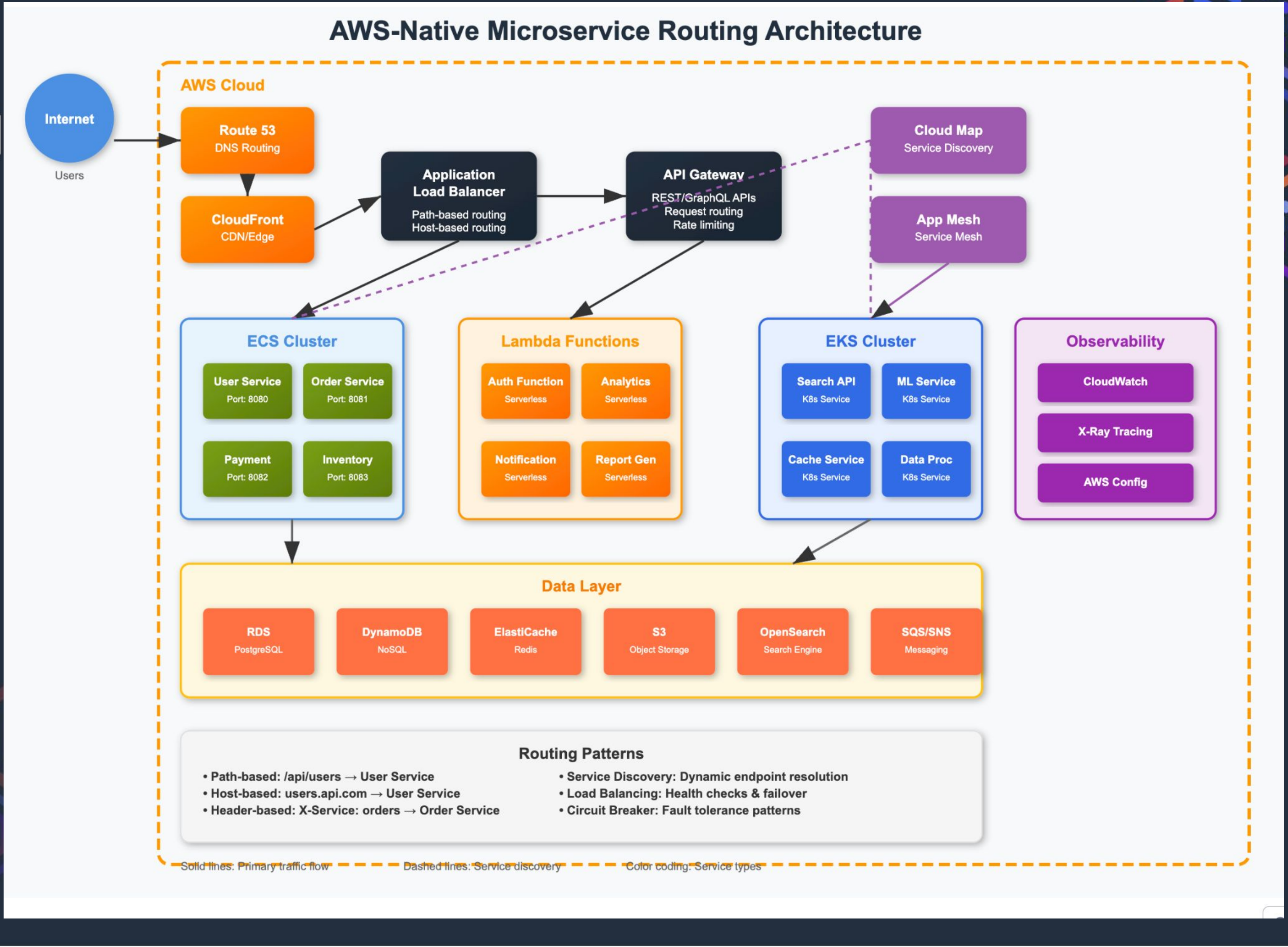
- Leverage AWS X-Ray, CloudWatch Logs to trace calls, measure performance, and troubleshoot routing issues quickly.

aws

COMMUNITY DAY

“Gain greater observability and reliability, reduce complexity, and ensure high availability and fault-tolerant communication between containerized applications using native routing patterns.”

COM



AWS-Native Microservice Routing Architecture

Routing Patterns

- Path-based: /api/users → User Service
- Host-based: users.api.com → User Service
- Header-based: X-Service: orders → Order Service
- Service Discovery: Dynamic endpoint resolution
- Load Balancing: Health checks & failover
- Circuit Breaker: Fault tolerance patterns

Solid lines: Primary traffic flow   Dashed lines: Service discovery   Color coding: Service types

Service Discovery & DNS

aws COMMUNITY DAY

Amazon Route53

Amazon Route53

Provides DNS-based **service discovery,** mapping service names to healthy endpoints. Supports both **public** and **private** DNS zones, health checks, and routing policies.

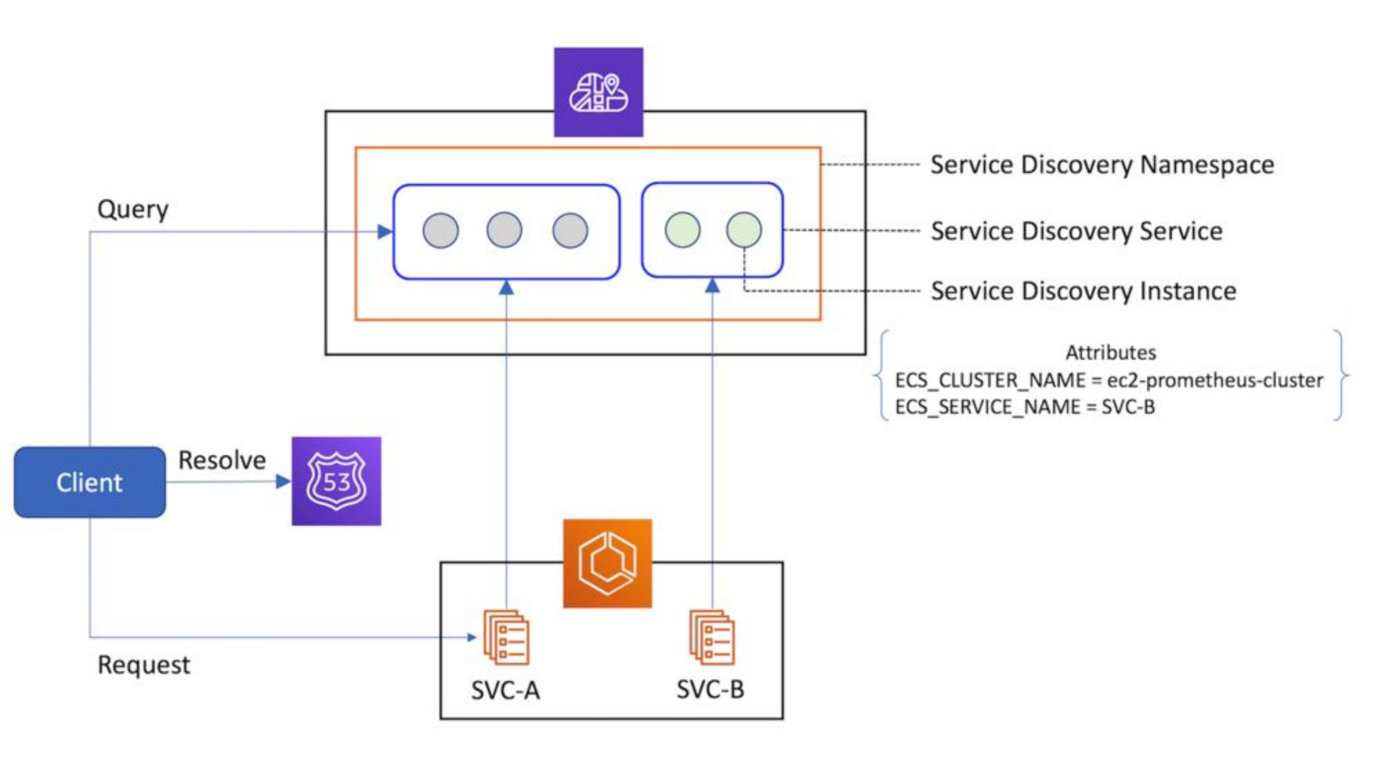
AWS Cloud Map

AWS Cloud Map

Acts as a **service registry,** allowing services to register their endpoints and attributes. Integrates with Route 53 for DNS-based lookups and supports API-based discovery.

Key Points:

- **Automates** endpoint management as services scale or change.
- Ensures **high availability** and **resilience**.
- Reduces manual configuration and operational overhead.


Query
Service Discovery Namespace
Service Discovery Service
Service Discovery Instance
Attributes
ECS_CLUSTER_NAME = ec2-prometheus-cluster
ECS_SERVICE_NAME = SVC-B
Client
Resolve
53
Request
SVC-A
SVC-B

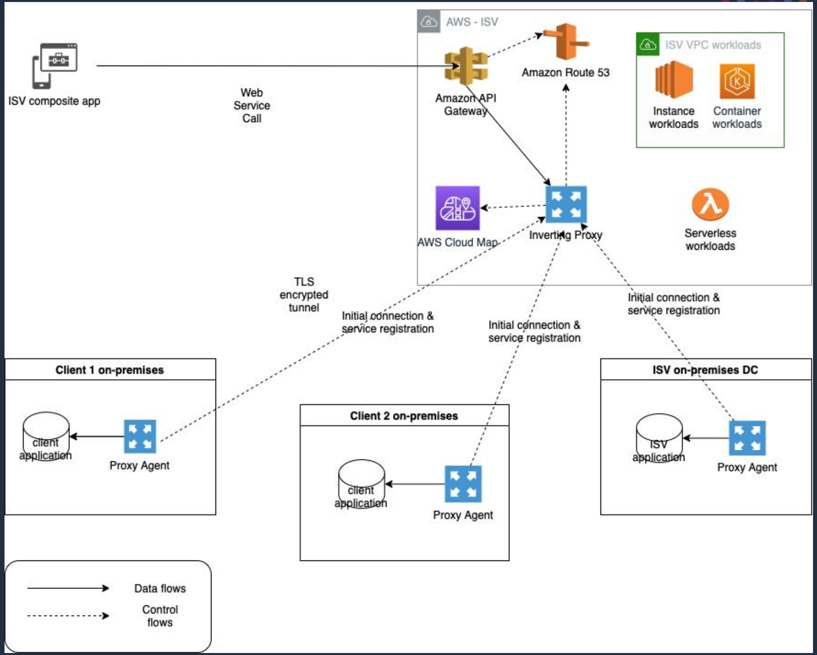
Reference:
https://aws.amazon.com/blogs/opensource/metrics-collection-from-amazon-ecs-using-amazon-managed-service-for-prometheus/

aws COMMUNITY DAY

Service Discovery - AWS Cloud Map

Problem Solved: How do services find each other's dynamic IPs?

- **Function**: A managed service registry. Services register their instances (IP, port) upon startup and deregister on shutdown.
- Discovery Modes:
  - **API Calls**: Services query the Cloud Map API to get a list of healthy endpoints. Gives most control.
  - **VPC DNS (via Route 53)**: Cloud Map automatically creates and manages Route 53 records (e.g., `users.internal.myservice.local`). Simplifies discovery for many clients.
- **Integration**: Natively integrates with ECS and EKS to automate instance registration.



Registering a service with AWS Cloud Map (using AWS CLI):

```
aws servicediscovery create-service \
  --name my-microservice \
  --dns-config 'NamespaceId=<namespace-id>,DnsRecords=[{Type="A",TTL="60"}]'
```

ECS Service Discovery integration (CloudFormation YAML):

```
ServiceRegistries:
  - RegistryArn: !GetAtt MyServiceDiscoveryService.Arn
```

Load Balancing & Edge Routing

aws COMMUNITY DAY

Load Balancing & Edge Routing

Amazon API Gateway

Fully managed service for creating, publishing, maintaining, monitoring, and securing APIs at any scale.

AWS App Mesh

Provides application-level networking to make it easy for your services to communicate with each other across multiple types of compute infrastructure.

AWS VPC Lattice

VPC Lattice is a fully managed application networking service by AWS. It simplifies service-to-service connectivity, security, and traffic management across VPCs.

ALB

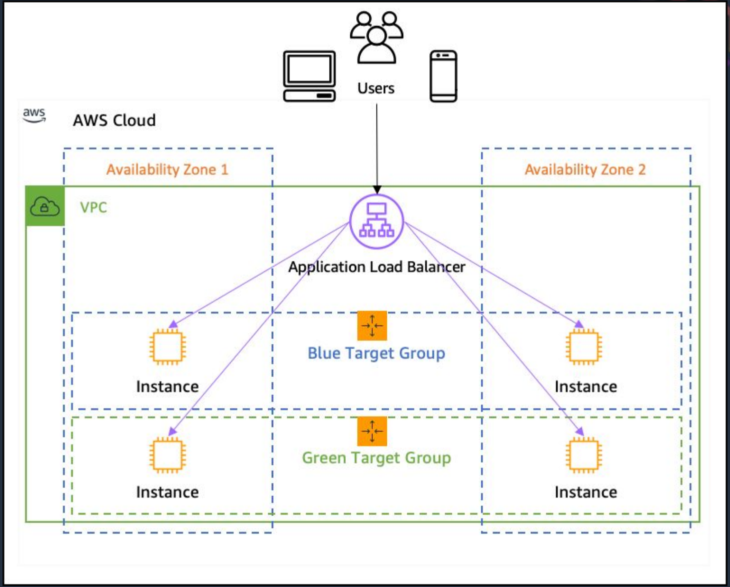
Includes Application Load Balancer (ALB) for HTTP/HTTPS (Layer 7) and Network Load Balancer (NLB) for TCP/UDP (Layer 4).

AWS COMMUNITY DAY

Internal L7 Routing - Application Load Balancer (ALB)

Problem Solved: How do we route internal HTTP/S traffic intelligently?

- Function: A managed Layer 7 (HTTP/HTTPS) load balancer.
- Key Routing Features:
  - **Path-based**: `api.example.com/users` -> users-service, `api.example.com/orders` -> orders-service.
  - **Host-based**: `users.api.example.com` -> users-service, `orders.api.example.com` -> orders-service.
  - **Header/Query String-based**: Route based on custom headers (e.g., `X-Version: v2`) for canary testing.
- **Target Groups**: Groups of backend targets (EC2, ECS, Lambda). ALBs route traffic to a target group, which manages health checks and distributes load.



Reference:
https://aws.amazon.com/blogs/aws/new-application-load-balancer-simplifies-deployment-with-weighted-target-groups/

aws

COMMUNITY DAY

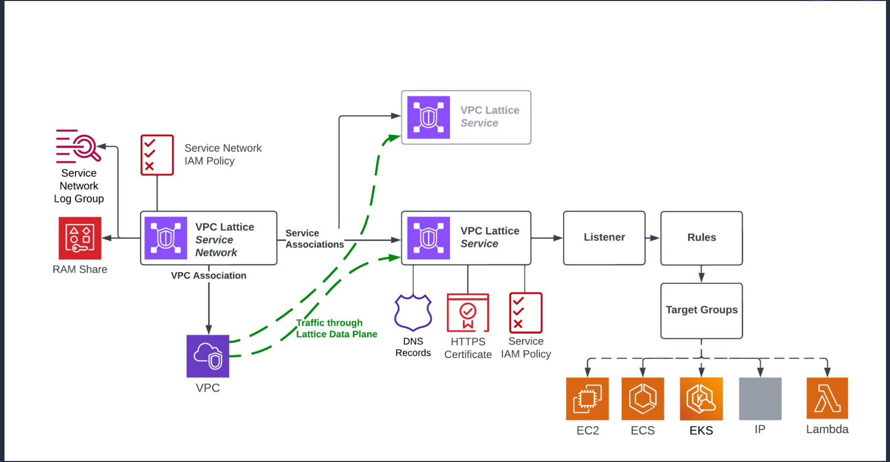
VPC Lattice

Problem Solved: How do we securely and efficiently route service-to-service HTTP/HTTPS traffic across multiple VPCs with centralized management and observability?

- **Service Discovery**: Automatically discovers services across VPCs.
- **Traffic Routing:** Flexible routing, load balancing, and path-based routing.
- **Security**: Integrated authentication, authorization, and encryption.
- **Observability**: Built-in monitoring and logging for network traffic.

Use Cases

- Microservices communication across VPCs.
- Centralized security and traffic controls.
- Hybrid and multi-account architectures.

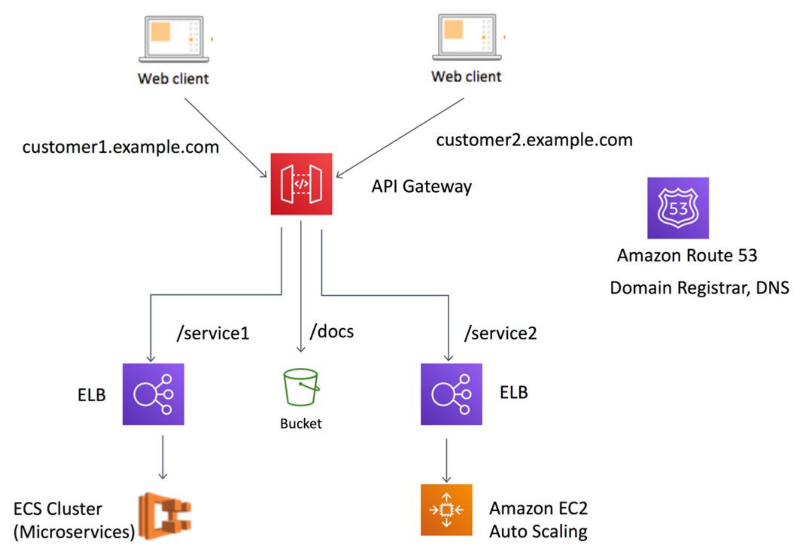


Reference: https://fourtheorem.com/vpc-lattice/

aws

COMMUNITY DAY

Edge Routing & API Management - API Gateway



Reference:
https://aws.amazon.com/blogs/architecture/using-api-gateway-as-a-single-entry-point-for-web-applications-and-api-microservices/

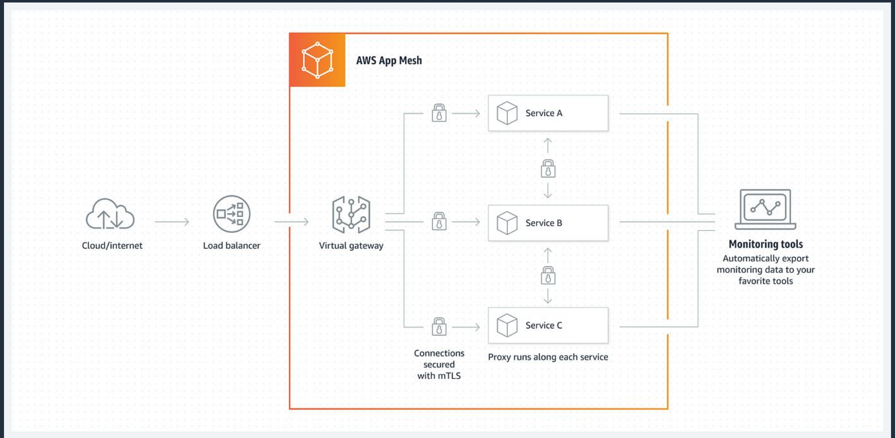
Problem Solved: How do we create a secure, managed "front door" for our APIs?

- Function: **More than a router**; a fully managed API management service.
- Key Features:
  - **Routing & Integration**: Routes requests to backend services like Lambda, ECS, Step Functions, or any HTTP endpoint.
  - **Request/Response Transformation**: Modify headers, query strings, and body content.
  - **Security**: Fine-grained authorization with IAM, Cognito User Pools, and Lambda Authorizers.
  - **Lifecycle Management:** Throttling, rate limiting, usage plans, and API keys.

Advanced Routing - AWS App Mesh

Problem Solved: How do we get fine-grained traffic control, resilience, and observability between services?

- Function: **A managed service mesh based on the Envoy proxy.**
- How it Works: App Mesh injects an Envoy proxy as a sidecar container next to each service instance. All traffic in/out of the service is routed through the proxy, which is centrally configured by App Mesh.
- Key Capabilities:
    - **Advanced Traffic Routing:** Precise weighted routing (e.g., 99% to v1, 1% to v2 for canary releases).
    - **Resilience**: Automated retries, timeouts, and circuit breakers configured centrally.
    - **Mutual TLS (mTLS)**: Enforces encrypted and authenticated communication between all services in the mesh.


AWS App Mesh
Service A
Cloud/internet
Load balancer
Virtual gateway
Service B
Monitoring tools
Automatically export monitoring data to your favorite tools
Service C
Connections secured with mTLS
Proxy runs along each service

Reference:
https://medium.com/avmconsulting-blog/application-networking-service-aws-app-mesh-e8e090c4996

Application Load Balancer (ALB) Target Group registration (Terraform):

```
resource "aws_lb_target_group" "example" {
  name     = "example-tg"
  port     = 80
  protocol = "HTTP"
  vpc_id   = "<your_vpc_id>"
}

# For EC2 instance registration
resource "aws_lb_target_group_attachment" "example" {
  target_group_arn = aws_lb_target_group.example.arn
  target_id        = "<instance_id>"
  port             = 80
}
```

aws

COMMUNITY DAY

ECS Service Discovery integration (CloudFormation YAML):

```yaml
Resources:
  MyPrivateDnsNamespace:
    Type: AWS::ServiceDiscovery::PrivateDnsNamespace
    Properties:
      Name: my-namespace.local
      Vpc: vpc-xxxxxxxx
      Description: "Private DNS namespace for ECS service discovery"

  MyServiceDiscoveryService:
    Type: AWS::ServiceDiscovery::Service
    Properties:
      Name: my-sd-service
      NamespaceId: !Ref MyPrivateDnsNamespace
      DnsConfig:
        DnsRecords:
          - Type: A
            TTL: 60
        RoutingPolicy: MULTIVALUE

  MyEcsService:
    Type: AWS::ECS::Service
    Properties:
      # ...existing ECS service properties...
      ServiceRegistries:
        - RegistryArn: !GetAtt MyServiceDiscoveryService.Arn
```

Observability Patterns

aws

COMMUNITY DAY

“Monitoring tells you whether a system is working;
Observability lets you understand why isn't working.”

Observability & Monitoring

Amazon CloudWatch

Collects metrics, logs, and events from AWS resources and applications.

AWS X-Ray

Traces requests across distributed services, identifying bottlenecks and failures.

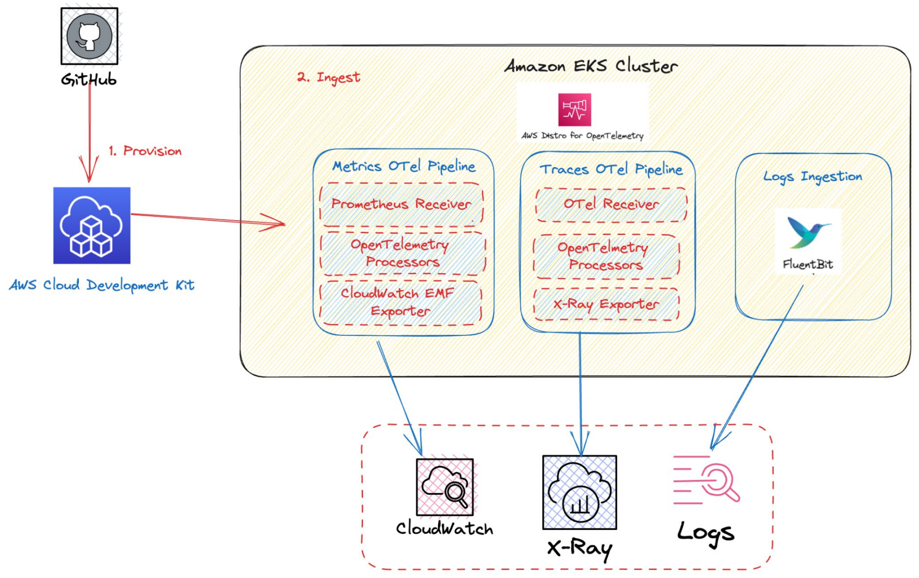
AWS Managed Services for Prometheus

Visualize key performance indicators and set up alerts for anomalies.

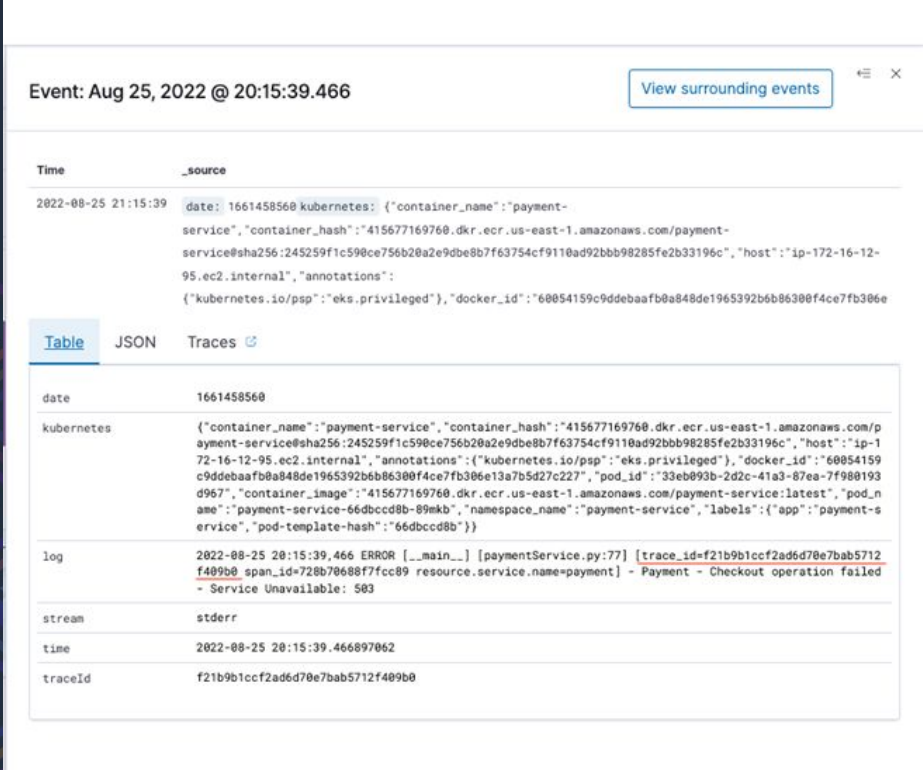
aws COMMUNITY DAY

Observability in Depth

- **CloudWatch**: Collects metrics, logs, and events from AWS resources and applications.
- **AWS X-Ray:** Traces requests across distributed services, identifying bottlenecks and failures.
- **Custom Metrics & Dashboards**: Visualize key performance indicators and set up alerts for anomalies.
- **App Mesh & Service Mesh Metrics:** Provides granular insights into service-to-service communication.



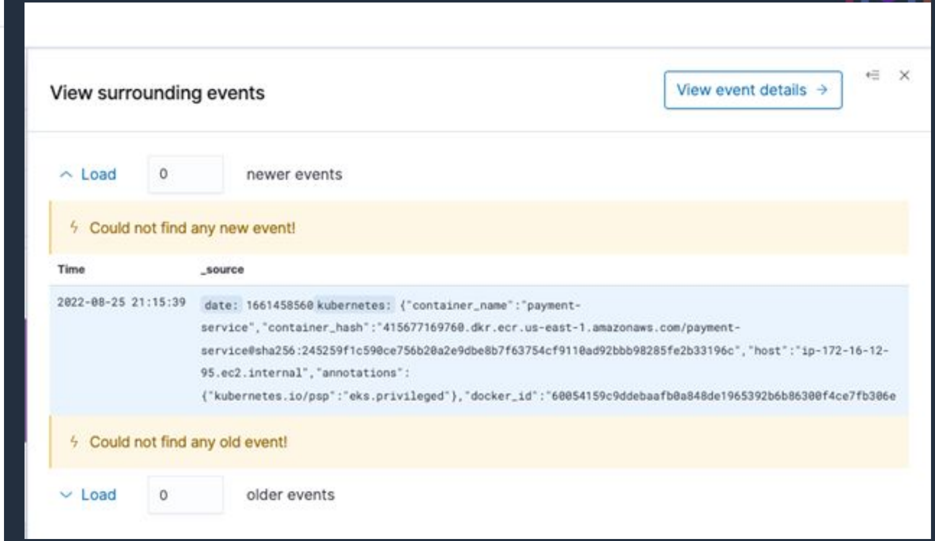
Reference:
https://aws-samples.github.io/cdk-eks-blueprints-patterns/patterns/observability/single-new-eks-awsnative-fargate-observability/



Event: Aug 25, 2022 @ 20:15:39.466

Time	_source
2022-08-25 21:15:39	date: 1661458560 kubernetes: {"container_name":"payment-service","container_hash":"415677169760.dkr.ecr.us-east-1.amazonaws.com/payment-service@sha256:245259f1c590ce756b20a2e9dbe8b7f63754cf9110ad92bbb98285fe2b33196c","host":"ip-172-16-12-95.ec2.internal","annotations":{"kubernetes.io/psp":"eks.privileged"},"docker_id":"60054159c9ddebaafb0a848de1965392b6b86300f4ce7fb306e

date	1661458560
kubernetes	{"container_name":"payment-service","container_hash":"415677169760.dkr.ecr.us-east-1.amazonaws.com/payment-service@sha256:245259f1c590ce756b20a2e9dbe8b7f63754cf9110ad92bbb98285fe2b33196c","host":"ip-172-16-12-95.ec2.internal","annotations":{"kubernetes.io/psp":"eks.privileged"},"docker_id":"60054159c9ddebaafb0a848de1965392b6b86300f4ce7fb306e13a7b5d27c227","pod_id":"33eb093b-2d2c-41a3-87ea-7f980193d967","container_image":"415677169760.dkr.ecr.us-east-1.amazonaws.com/payment-service:latest","pod_name":"payment-service-66dbccd8b-89mkb","namespace_name":"payment-service","labels":{"app":"payment-service","pod-template-hash":"66dbccd8b"}}
log	2022-08-25 20:15:39,466 ERROR [__main__] [paymentService.py:77] [trace_id=f21b9b1ccf2ad6d70e7bab5712f409b0 span_id=728b70688f7fcc89 resource.service.name=payment] - Payment - Checkout operation failed - Service Unavailable: 503
stream	stderr
time	2022-08-25 20:15:39.466897062
traceId	f21b9b1ccf2ad6d70e7bab5712f409b0



View surrounding events

ϟ Could not find any new event!

Time	_source
2022-08-25 21:15:39	date: 1661458560 kubernetes: {"container_name":"payment-service","container_hash":"415677169760.dkr.ecr.us-east-1.amazonaws.com/payment-service@sha256:245259f1c590ce756b20a2e9dbe8b7f63754cf9110ad92bbb98285fe2b33196c","host":"ip-172-16-12-95.ec2.internal","annotations":{"kubernetes.io/psp":"eks.privileged"},"docker_id":"60054159c9ddebaafb0a848de1965392b6b86300f4ce7fb306e

ϟ Could not find any old event!

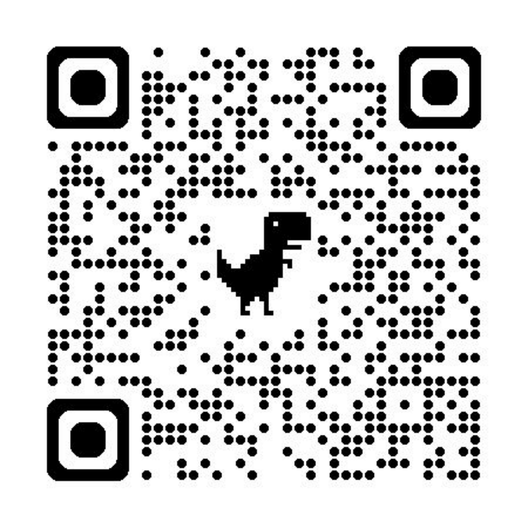
Reference:
https://aws.amazon.com/blogs/big-data/part-1-microservice-observability-with-amazon-opensearch-service-trace-and-log-correlation

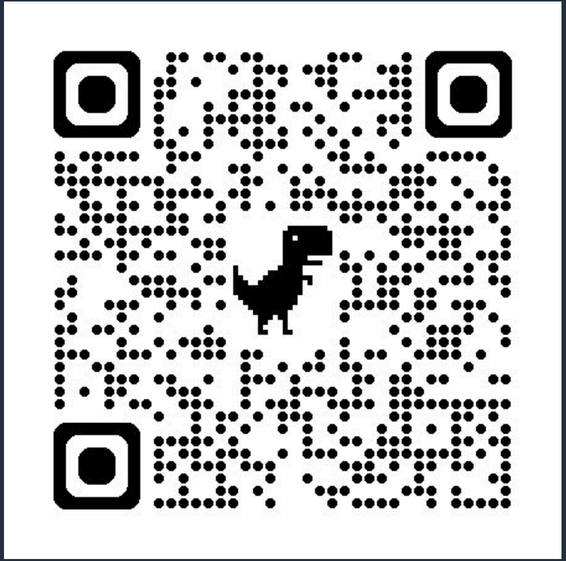
Demo

aws

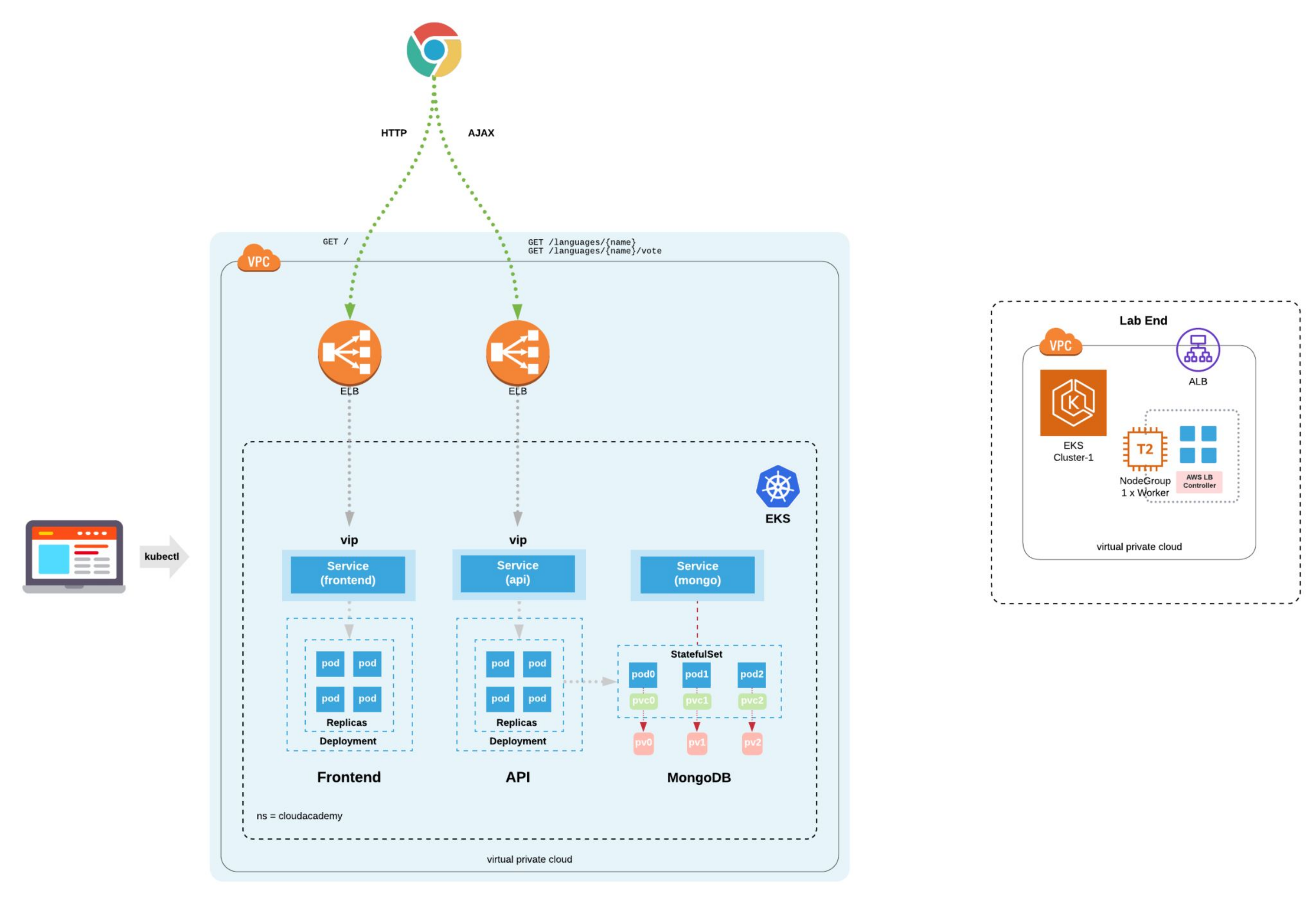
COMMUNITY DAY

Demo





co


HTTP
AJAX
GET /
GET /languages/{name}
GET /languages/{name}/vote
VPC
ELB
ELB
kubectl
EKS
vip
vip
Service (frontend)
Service (api)
Service (mongo)
pod
pod
pod
pod
Replicas
Deployment
pod
pod
pod
pod
Replicas
Deployment
StatefulSet
pod0
pod1
pod2
pvc0
pvc1
pvc2
pv0
pv1
pv2
Frontend
API
MongoDB
ns = cloudacademy
virtual private cloud
Lab End
VPC
ALB
EKS Cluster-1
T2
NodeGroup 1 x Worker
AWS LB Controller
virtual private cloud

aws COMMUNITY DAY

Thank you

@misskecupbung

SCAN ME