AWS re:Invent

SEC347-R

DNS across a multi-account environment

Paul Bayer

Senior Consultant, Cloud Infrastructure Architecture

Amazon Web Services

Agenda

Problem statement

Current solutions

Design Points to Consider

Previous solutions

Best Practices

Problem statement

Problems to *{re}*solve

- How to prevent data exfiltration
  - https://dejandayoff.com/using-dns-to-break-out-of-isolated-networks-in-a-aws-cloud-environment/
- How to consistently resolve names when servers aren’t using **the same** DNS servers?
- What name should I use to refer to a given machine?
- Where should machines *register* their hostname?
  - To their Active Directory on-prem?
  - To their cloud-native private zone?
  - Do we really need to register names at all?

DNS Overview

- DNS (Domain Name Service)
  - Critical fundamental service to all IP communications
  - Resolves name to IP (forward)
  - Resolves IP to name (reverse)
  - Easy to unknowingly get wrong
  - Critical for use with TLS/certificates
- **DO NOT USE HOSTS FILES**

DNS zones/namespaces we’ll discuss

- Amazon Virtual Private Cloud (Amazon VPC) native
  - “compute-1.amazonaws.com” (Amazon EMR)
  - “compute.internal” / “ec2.internal” (Amazon EC2)
- Custom-named zones
  - “cloud.example.com”
  - Must manually add names to the hosted zone*
- On-premises zones
  - “on-prem.example.com”
  - Can be automatically registered to Active Directory’s DNS

- **Yes, you can write a lambda to do this*

Private Hosted Zone

- Private Hosted Zone – abbreviated “PHZ”
- Owned by a single Account – but shared/associated with 1 or more VPCs.
  - See code snippet for how to associate
- There is no charge to associate this PHZ with multiple VPCs
- PHZ is a global service, so you may need to create multiple PHZs and share them with appropriate VPCs as necessary.

DNS Enhancements

- EDNS(0)
  - Wikipedia: https://en.wikipedia.org/wiki/Extension_mechanisms_for_DNS
  - Enables DNS to support larger than 512 byte packets within UDP
  - Opens up additional capacity for flags and features
- DNSSEC
  - Wikipedia: https://en.wikipedia.org/wiki/Domain_Name_System_Security_Extensions
  - Method to ensure a chain of integrity starting with the DNS root
  - Requires every zone to ***sign*** their records, and update the signatures periodically
  - If the chain is broken, resolution will not work
  - If DNSSEC isn’t supported at the root, then it won’t be requested further down

Current Solution

{Re}solution (Question 1)

- How to prevent DNS queries from exfiltrating data?
  - Allow connections to KNOWN endpoints only
  - Don’t let them REQUEST unknown websites
  - Log all data sent via DNS queries *(already done on-prem)*

{Re}solution (Question 2)

- How to consistently resolve names when servers aren’t using **the same** DNS servers?
  - There should always be only one source of truth for DNS. There may be many roads to that source, but only one authority.
  - Split DNS works for some situations, but does require additional care.

{Re}solution (Question 3)

- What name should I use to refer to a given machine?
  - Let’s use something that won’t change just because you scaled, updated, released, etc.
  - Let’s talk about which machines you’re going to name…
- Good things to name:
  - Load Balancers
  - Amazon EMR master servers
  - Individual machines (fault tolerance?)
  - On-prem resources should resolve *on-prem*
  - Cloud resources should resolve *in the cloud*

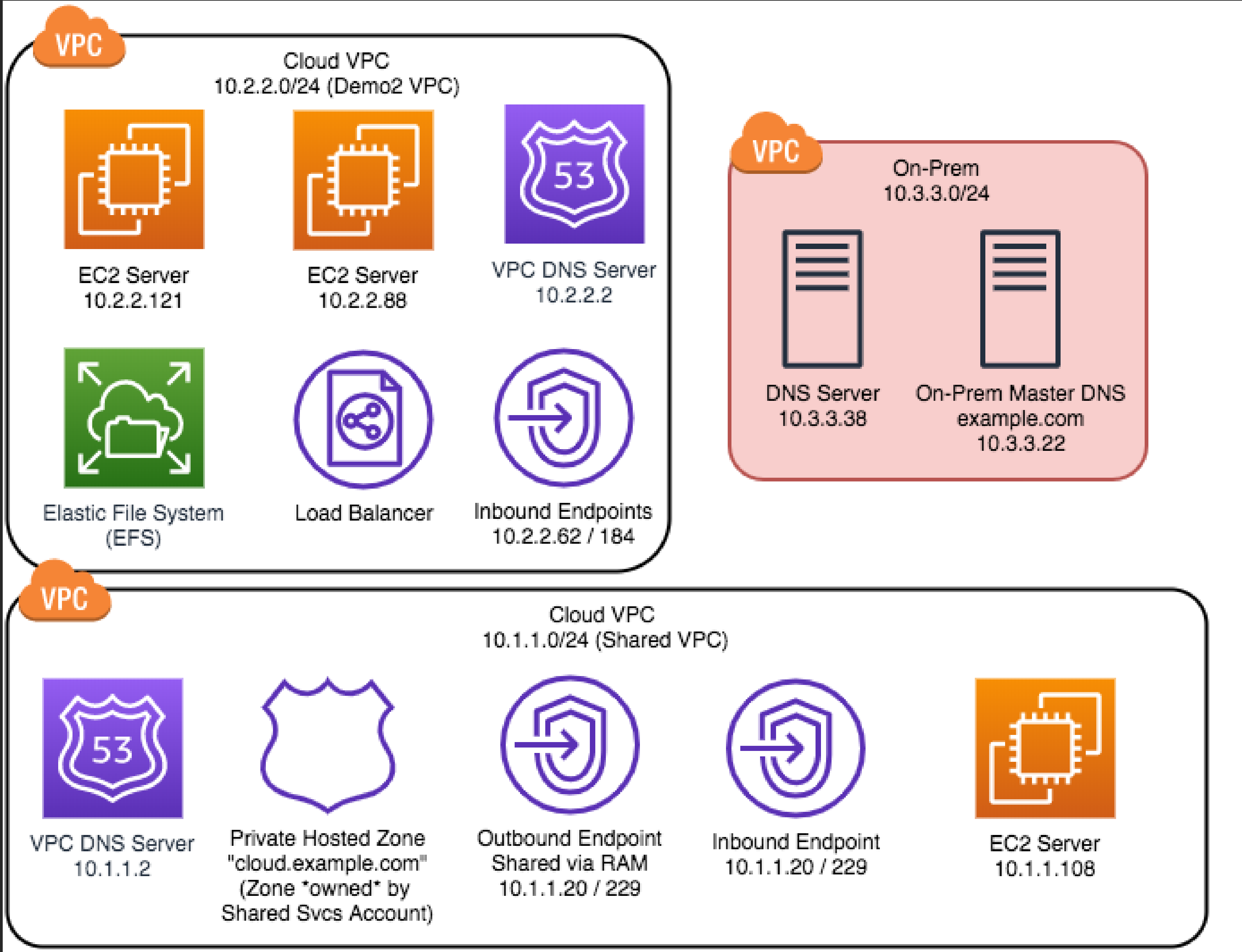
{Re}solution (Question 4)

- Where should machines register their hostname?
  - To the Active Directory on-prem?
    - *If you're going to use AD authentication, you may need to do that*
  - To their cloud-native private zone?
    - *Can be done on instance-creation with AWS Lambda functions*
  - Do we really need to register names at all?
    - *"Cattle, not pets" – consider the ephemerality of your instances*
- Registering your host to AD doesn't mean you must use that server as your **primary** DNS, only that the AD domain is always resolveable.

Demo of Use Cases

Overall Environment

- Source
  - App VPC
- Shared Services
  - Central VPC
- On-Prem
  - On-Premises replica
- Private Hosted Zone
- On-Prem Zone
- Amazon Elastic File System (Amazon EFS) Mount
- Load-Balancer URL



Private Hosted Zone (cloud.example.com)

www.cloud.example.com - ALIAS record to external LB
www-int.cloud.example.com - ALIAS record to internal LB
myserver.cloud.example.com - A - 10.1.1.200
shareddns.cloud.example.com - A - 10.1.1.108

On-Prem Zone (example.com)

www.example.com - A - 10.3.3.100
mail.example.com - A - 10.3.3.101
ftp.example.com - CNAME - www.example.com
dns.example.com - A - 10.3.3.22

Use Cases

1. Cloud resource to resolve a local AWS-native name
2. Cloud resource to resolve a PHZ name
3. On-prem resource to resolve an AWS-native name (*EFS*)
4. Cloud resource to resolve an on-prem name
5. Cloud resource to resolve ANYTHING else

Diagram - use case 1

(Cloud resource to resolve a local AWS-native name)

- Query from “121”
- ➢ dig ip-10-2-2-86.ec2.internal

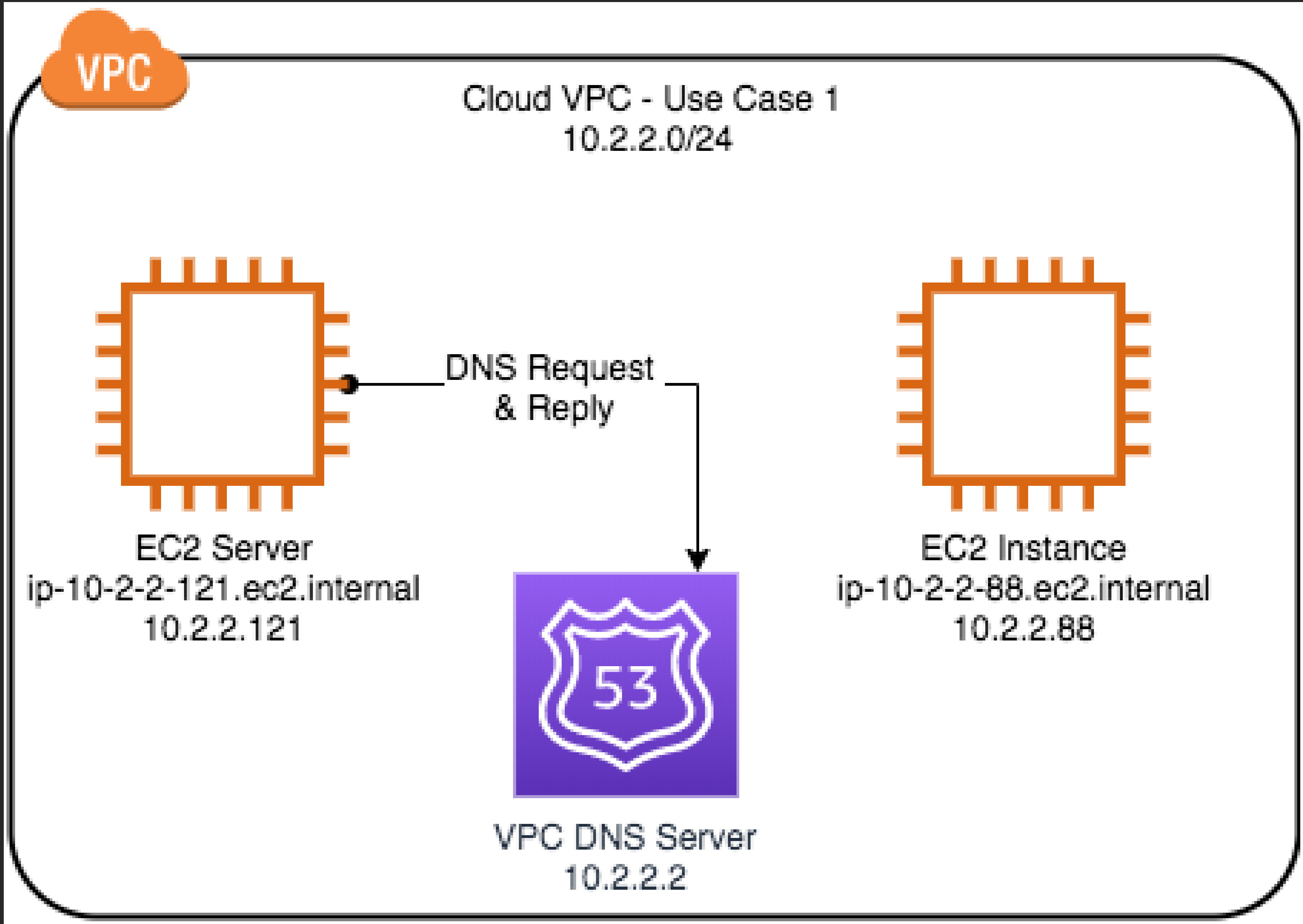


Diagram - use case 2

(Cloud resource to resolve a PHZ name)

- Query from "121"

➢ Dig www-cloud.example.com

➢ dig www-int.cloud.example.com

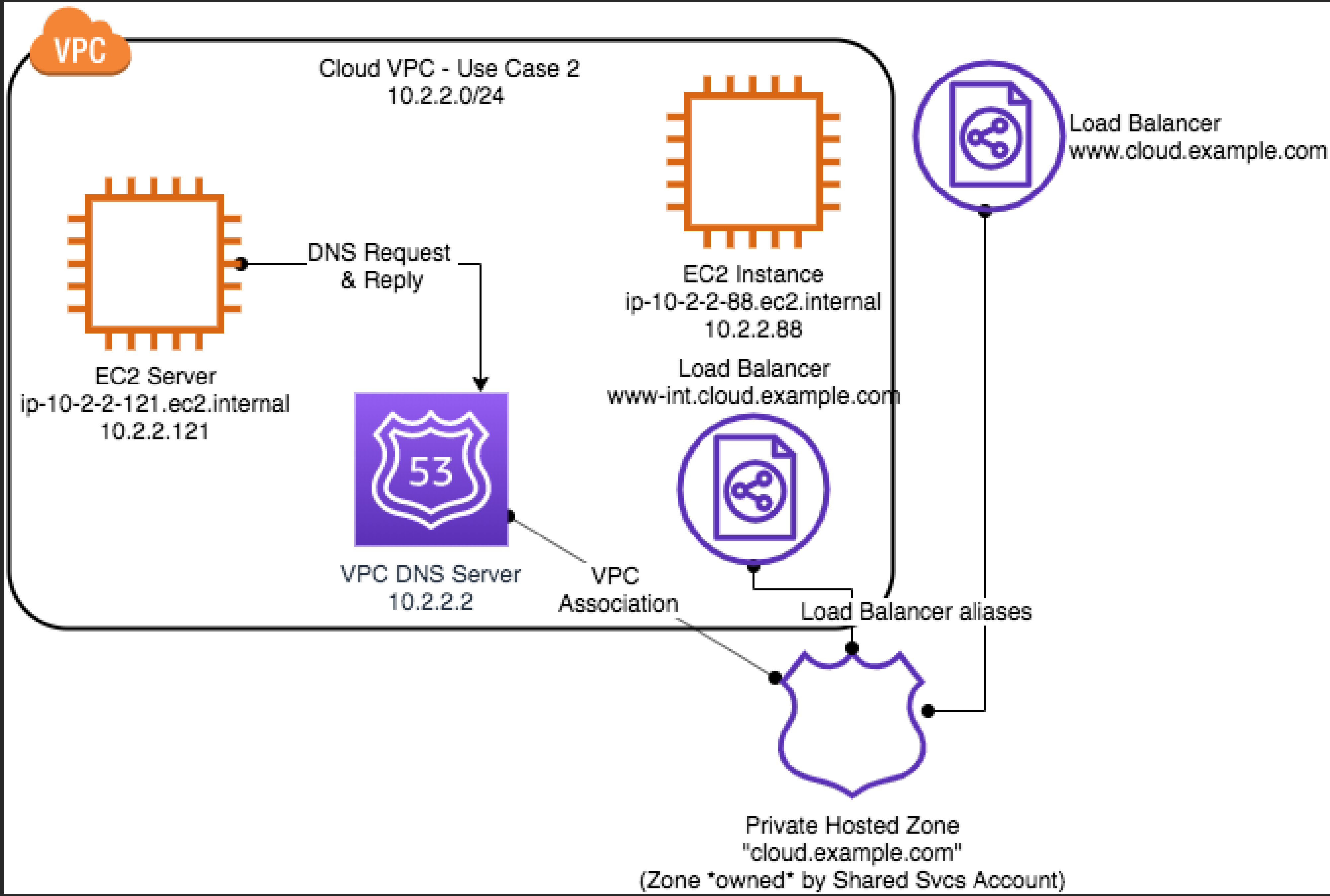


Diagram - use case 3
(On-prem resource to resolve an AWS-native name (EFS))

- Query from "on-prem"
  - ➢ dig fs-53bec…

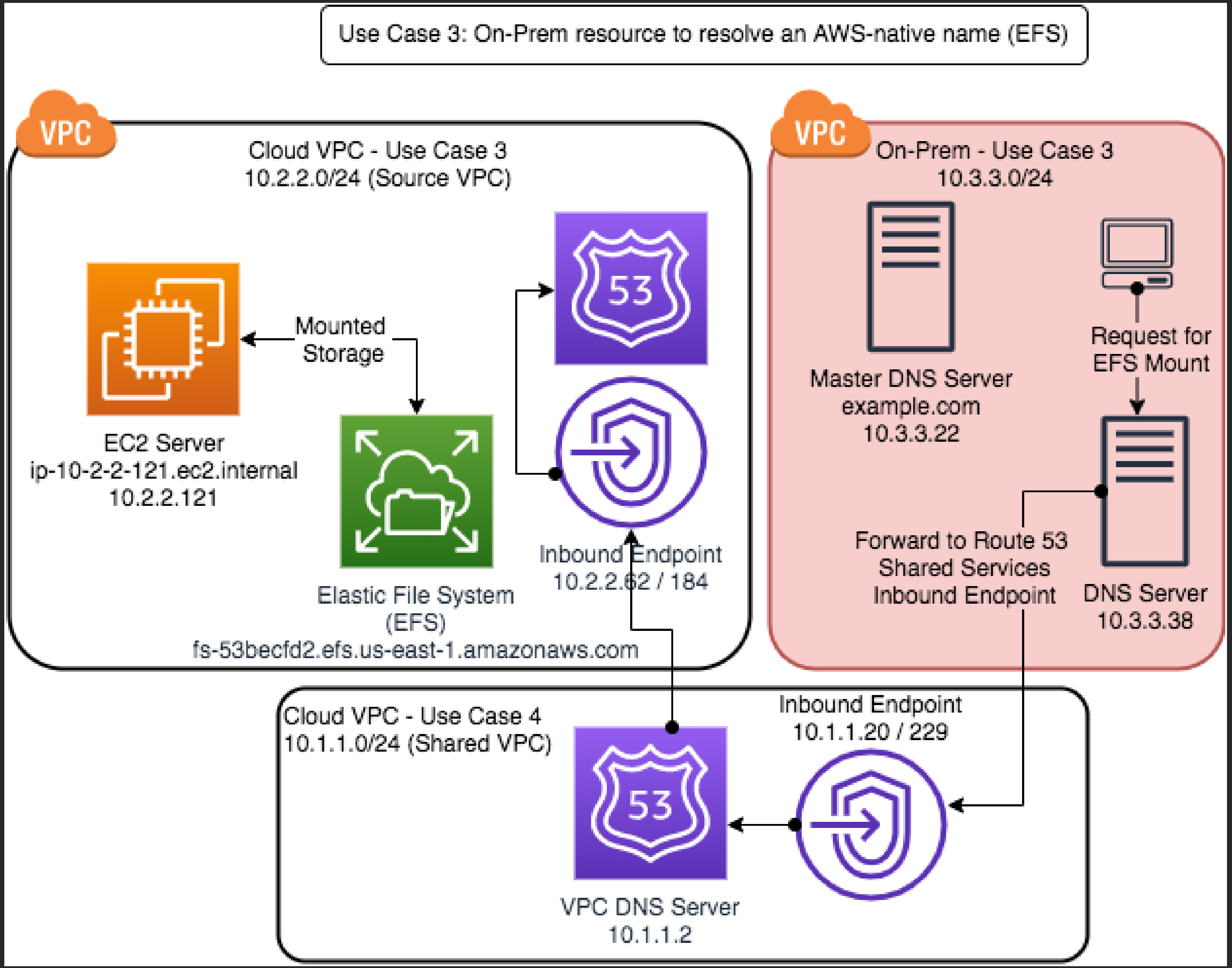


Diagram - use case 3

(Cloud resource to resolve an AWS-native name in another VPC (EKS))

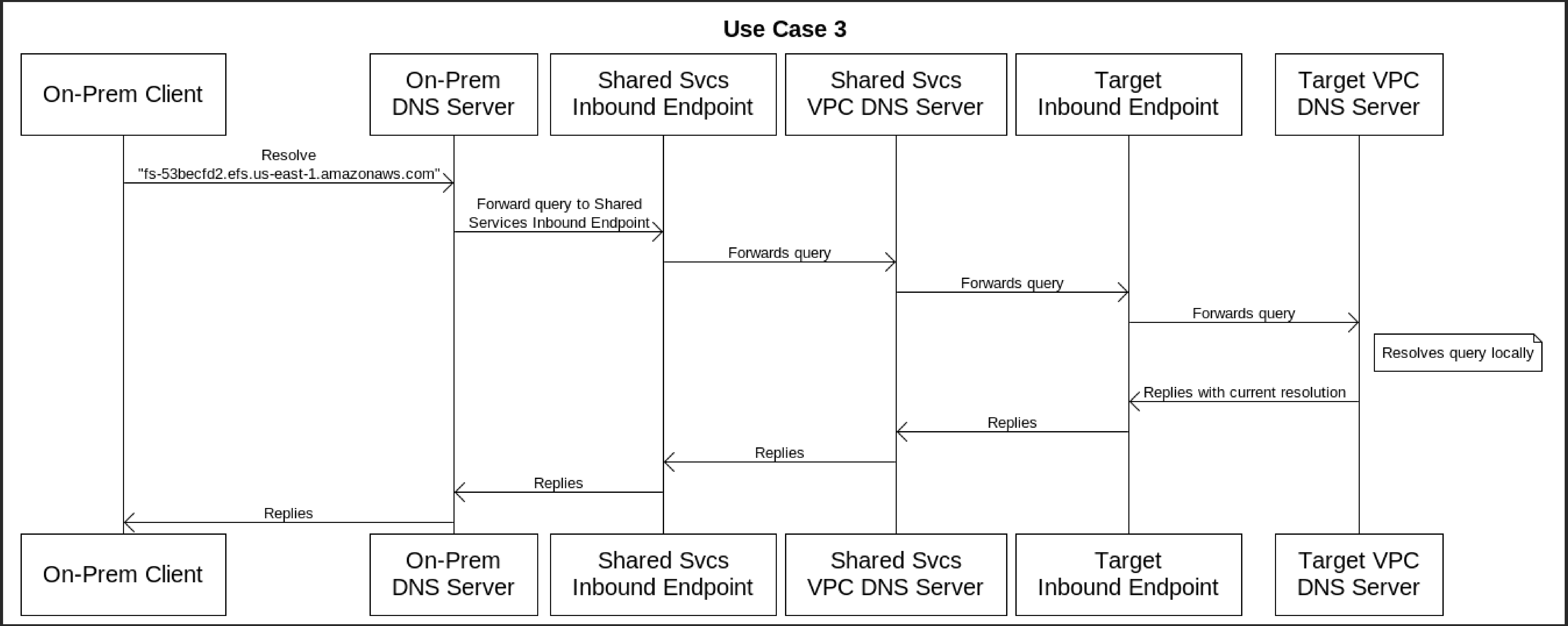


Diagram - use case 4

(Cloud resource to resolve an on-prem name)

- Query from cloud

➢ dig www.example.com

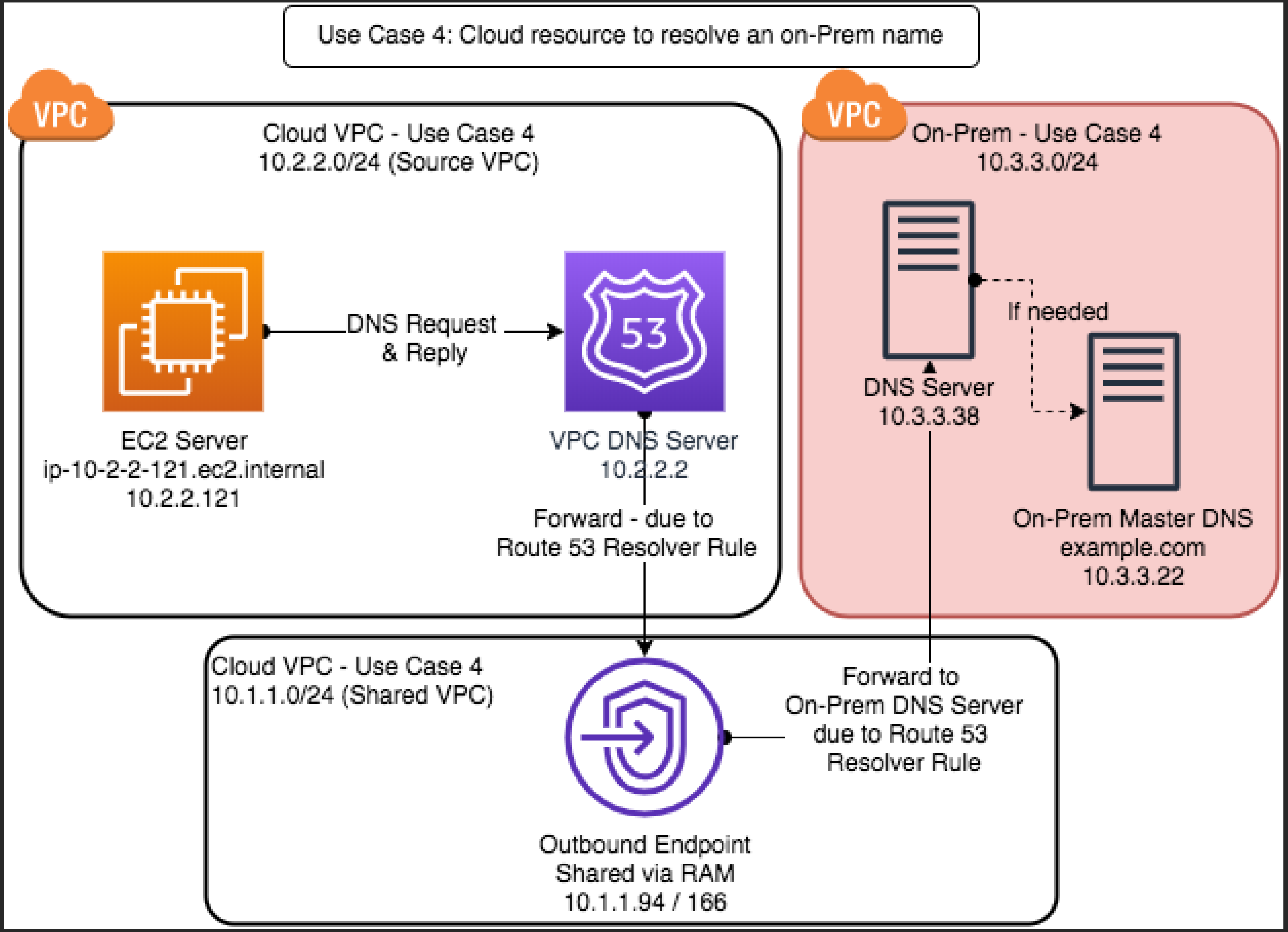


Diagram - use case 4

(Cloud resource to resolve an on-prem name)

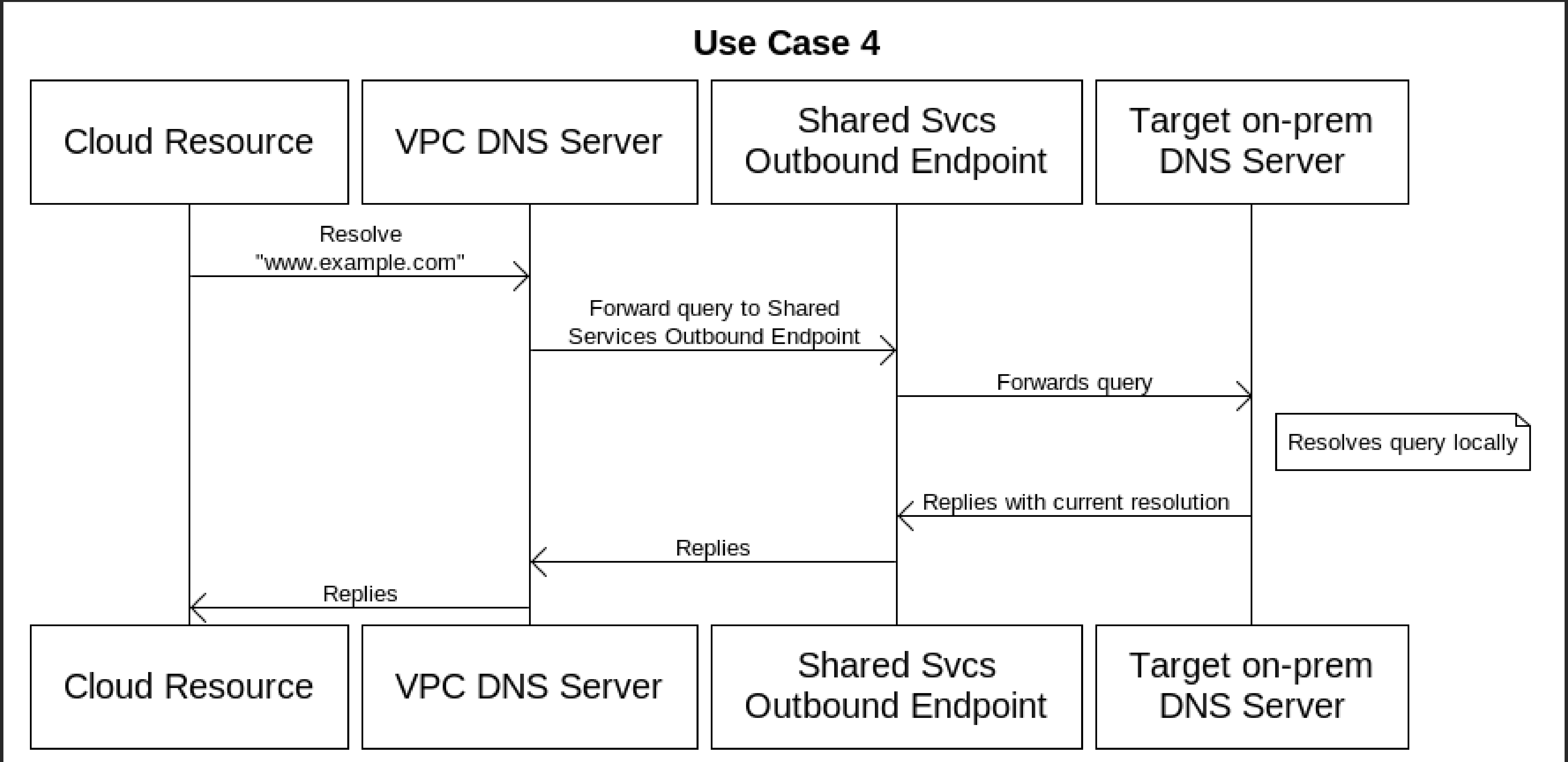

Use Case 4
Cloud Resource
VPC DNS Server
Shared Svcs Outbound Endpoint
Target on-prem DNS Server
Resolve "www.example.com"
Forward query to Shared Services Outbound Endpoint
Forwards query
Resolves query locally
Replies with current resolution
Replies
Replies
Cloud Resource
VPC DNS Server
Shared Svcs Outbound Endpoint
Target on-prem DNS Server

Diagram - use case 5
(Cloud resource to resolve ANYTHING else)

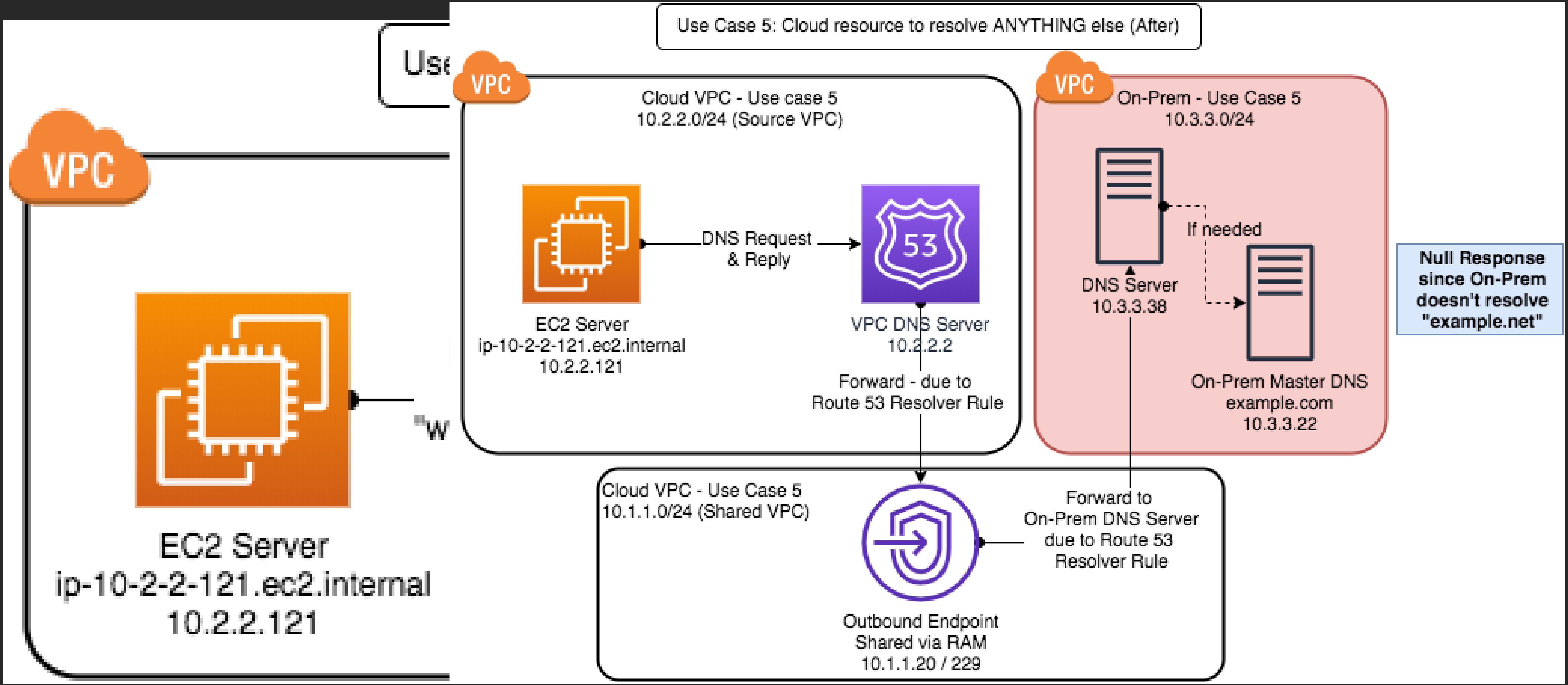
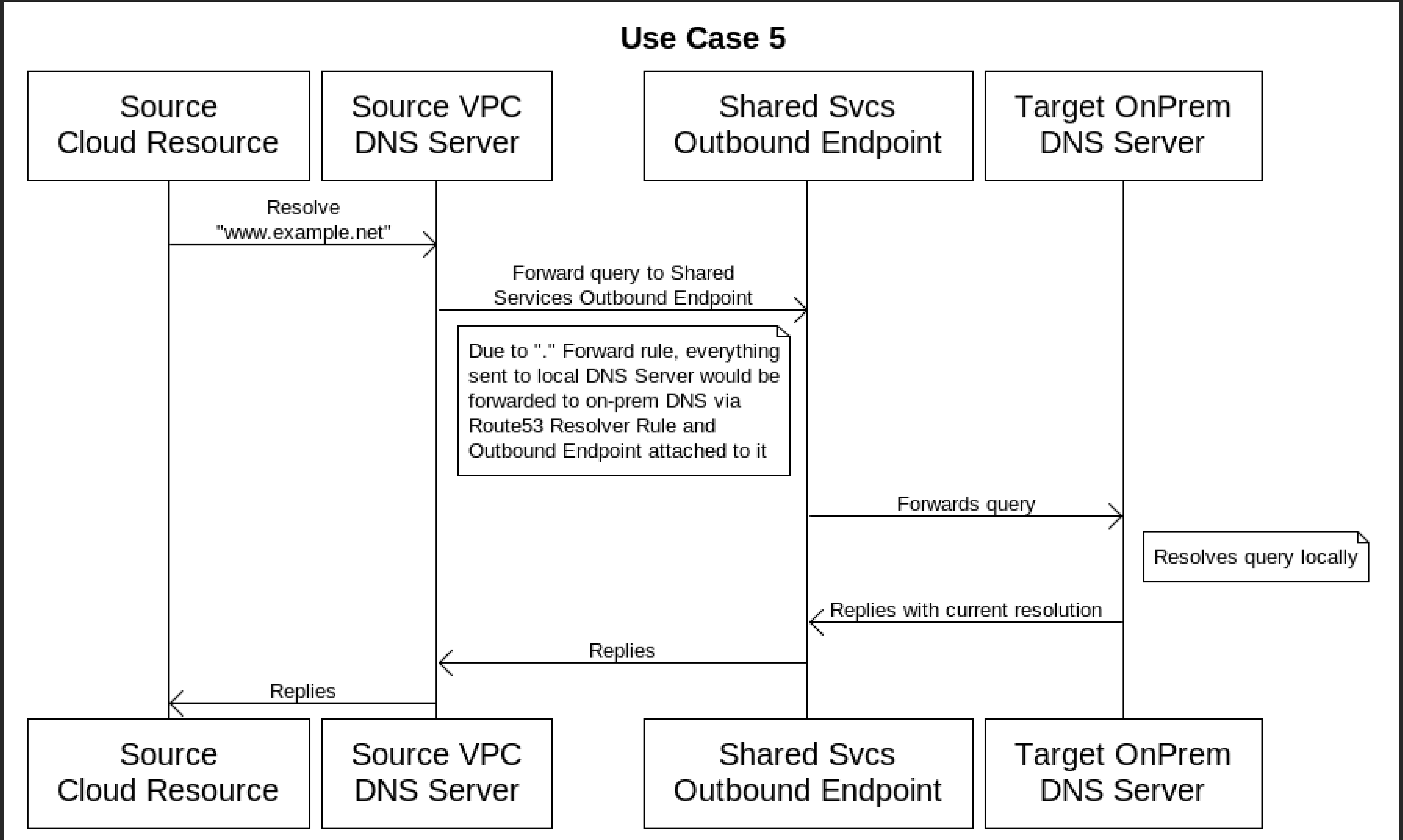


Diagram - use case 5
(Cloud resource to resolve an on-prem name)



Use Case 5
Source Cloud Resource
Source VPC DNS Server
Shared Svcs Outbound Endpoint
Target OnPrem DNS Server
Resolve "www.example.net"
Forward query to Shared Services Outbound Endpoint
Due to "." Forward rule, everything sent to local DNS Server would be forwarded to on-prem DNS via Route53 Resolver Rule and Outbound Endpoint attached to it
Forwards query
Resolves query locally
Replies with current resolution
Replies
Replies

Previous Solutions

Former solutions

- Native VPC DNS
  - Doesn’t resolve on-premises names
- Private hosted zones w/ Lambda to mirror on-prem zones
  - Complex
- Unbound (DNS proxies)
  - Have to manage additional servers
  - PPS DNS limits
- DNSMasq on every instance
  - A lot to administer
- Point to shared services DNS (AD) & forward to AWS DNS
  - Doesn’t handle managed service names properly
  - Reverse resolution can be problematic

Design Points to Consider

Managed services considerations

- Amazon EKS
  - The names of the cluster are known only within the individual VPC
- Amazon API Gateway
  - The API Gateway service (used in “Private mode”) provides a name for the APIGW via the Internet – although it’s not a “friendly” name
- Amazon EMR
  - The communication between the master node and the children use both forward and reverse DNS
  - Console access to Amazon EMR is typically required, so connecting the master node to AD is a typical use case
  - Since the children nodes cannot be joined to AD, using the AD DNS (for authentication) without being able to resolve the children nodes (both ways) will prevent the Amazon EMR cluster from working properly
  - This has been previously solved by using DNSMasq—and it works, but Route 53 Rules are better
- AWS Glue
  - The communication between nodes and AWS Glue has used—by default—reverse DNS to **authenticate** the caller to the AWS Glue service
- Amazon EFS
  - Amazon EFS is only properly resolvable from within the VPC you’re using it within. While it’s **accessible** from elsewhere, only the local VPC DNS server knows the proper name. Therefore, if you want to use the name, you need to resolve the name through the specific, appropriate VPC DNS.

VPC endpoints and DNS

- VPC endpoints (interface) work by creating “hidden” DNS names that redirect via IP
- This resolution is ONLY available within the specific VPC that you’ve created the endpoint*
- Therefore, the **proper** resolution is critical to ensure you’re using the private endpoints and not an internet gateway

- If you create a PHZ for the endpoint names – you can share endpoints across VPCs, and drastically reduce your costs on endpoints.
- Automate the association of new VPCs with the PHZ in that region

Order of DNS resolution

Order of resolution for each DNS query that arrives at the +2 AWSProvidedDNS endpoint:

1. Is there a Route 53 Resolver rule that applies?
   - Pick most specific applicable rule
   - This includes a few hidden rules/zones (URL needed)
2. Is there a Private Hosted Zone that applies?
   - Pick the most specific applicable zone
3. Resolve within AmazonProvidedDNS

Forwarding vs. Delegation

- Wouldn’t it be nice if we could delegate zone authority to the inbound resolvers or a DNS server within the cloud that then did a recursive lookup (forward) for us?
  - YES – that would be awesome
  - But, it’s antithetical to how DNS works
  - If we delegate – it’s because the server we’re delegating to is AUTHORITATIVE for that zone
  - If it was authoritative, it doesn’t need to forward
  - If we forward – we know it’s going to ask someone else, therefore, we ask a recursive question
  - Therefore – we can’t delegate and then expect it to forward; we can **either** forward **or** delegate

Previous Solutions

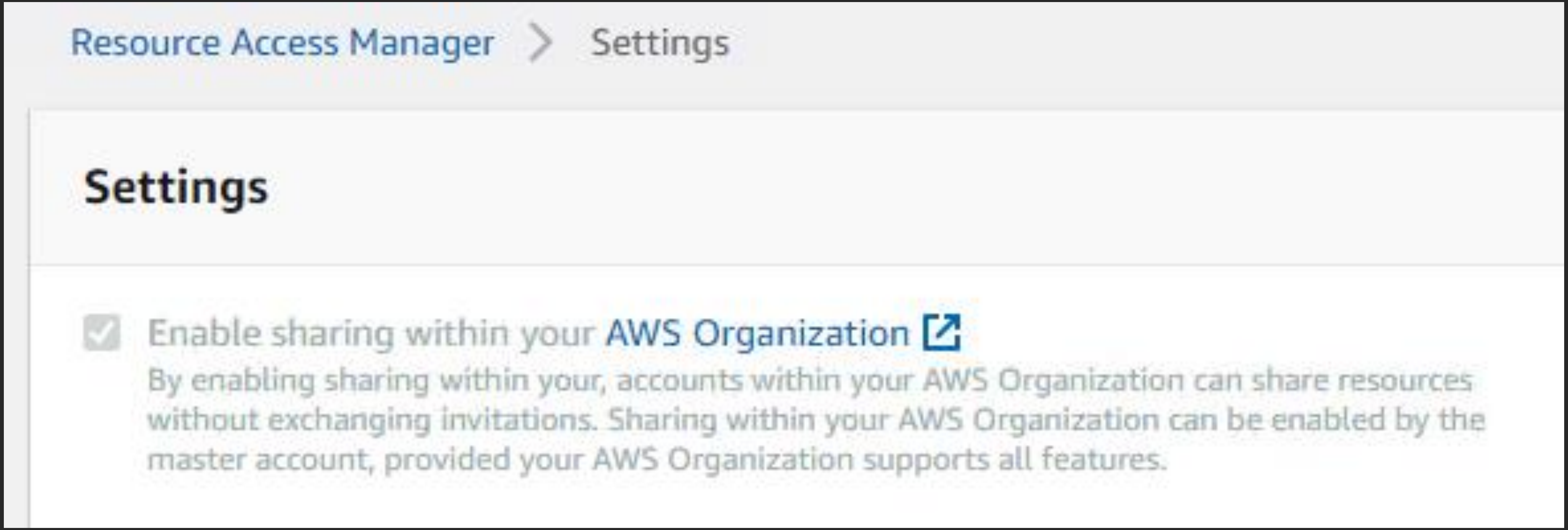
Amazon Route 53 Resolver rule

- Shared rules contain
  - Domain name
    - Forward rules: domain name *(*.example.com)* being forwarded
    - System rules: domain name *(*.dontforward.example.com)* being excepted from the rule
  - Inbound/outbound endpoint references
    - **Must** be on 2+ subnets
  - Target IPs to forward to *(usually on-prem DNS servers)*
  - Tags
- Locally enabled rules contain
  - Local account's VPC associations
- Rule can be shared via AWS Resource Access Manager
  - The sharing of the **rule** includes the original endpoint—so while rules do need to be associated with each VPC (even in additional accounts), there is no need to re-create the **endpoints**

Best Practices

AWS RAM setting *(in MBA)*

- Enable sharing at the top-most level within your organization
- This setting affects all services, and not just Route 53 Resolver, so ensure that this setting complies with your security policy



Resource Access Manager > Settings

Settings

✓ Enable sharing within your AWS Organization

By enabling sharing within your, accounts within your AWS Organization can share resources without exchanging invitations. Sharing within your AWS Organization can be enabled by the master account, provided your AWS Organization supports all features.

Route 53 Resolver best practices

- By maintaining your Route 53 Resolver rules in a single, centralized account, multiple rules can all use the same inbound/outbound endpoints
- Determine where you will forward internet-resolvable names—to internet (via AWS) or on-prem?
- Make use of RAM to ensure you’re not duplicating rules and endpoints in multiple accounts.
    - *Remember that use of shared Outbound endpoints does NOT require peering or TGW!*
- For services that **require** “individualized VPC” resolution, create an Inbound endpoint in the VPC and access from shared VPC.

Demo

- Use-Case 1
- Use-Case 2
- Use-Case 3
- Use-Case 4
- Use-Case 5 (Before)
- Use-Case 5 (After)

Code Repo

- Blog explaining how to Associate PHZ with VPCs
- Python code to add EC2 instances to a PHZ upon startup
- VPC Setup to replicate this environment here

Questions?

Thank you!

Paul Bayer

paulbaye@amazon.com

Please complete the session survey in the mobile app.

AWS re:Invent

aws