AWS Whitepaper

Hybrid Cloud DNS Options for Amazon VPC

Hybrid Cloud DNS Options for Amazon VPC: AWS Whitepaper

Table of Contents

Hybrid Cloud DNS Options for Amazon VPC

Publication date: **December 02, 2022** (*Document revisions*)

Abstract

The Domain Name System (DNS) is a foundational element of the internet that underpins many services offered by Amazon Web Services (AWS). Amazon Route 53 Resolver provides resolution with DNS for public domain names, Amazon Virtual Private Cloud (Amazon VPC), and Amazon Route 53 private hosted zones (PHZs).

This whitepaper includes solutions and considerations for advanced DNS architectures to help customers who have workloads with unique DNS requirements, or on-premises resources that require DNS resolution between on-premises data centers and Amazon Elastic Compute Cloud (Amazon EC2) instances in Amazon VPCs.

Are you Well-Architected?

The AWS Well-Architected Framework helps you understand the pros and cons of the decisions you make when building systems in the cloud. The six pillars of the Framework allow you to learn architectural best practices for designing and operating reliable, secure, efficient, cost-effective, and sustainable systems. Using the AWS Well-Architected Tool, available at no charge in the AWS Management Console, you can review your workloads against these best practices by answering a set of questions for each pillar.

For more expert guidance and best practices for your cloud architecture—reference architecture deployments, diagrams, and whitepapers—refer to the AWS Architecture Center.

Introduction

Many organizations have both on-premises resources and resources in the cloud. DNS name resolution is essential for on-premises and cloud-based resources. For customers with hybrid workloads, which include both on-premises and cloud-based resources, extra steps are necessary to configure DNS to work seamlessly across both environments.

AWS services that require name resolution could include Elastic Load Balancing (ELB), Amazon Relational Database Service (Amazon RDS), Amazon Redshift, and Amazon EC2.

Route 53 Resolver, which is available in all Amazon VPCs, responds to DNS queries for public records, Amazon VPC resources, and Route 53 PHZs.

You can configure Route 53 Resolver to forward queries to customer-managed authoritative DNS servers hosted on-premises, and to respond to DNS queries that your on-premises DNS servers forward to your Amazon VPC.

This whitepaper illustrates several different architectures that you can implement on AWS using native and custom-built solutions. These architectures meet the need for name resolution of on-premises infrastructure from your Amazon VPC, and address constraints that have only been partially addressed by previously published solutions.

Key concepts

Before we dive into the solutions, it is important to establish a few concepts and configuration options that we'll reference throughout this whitepaper.

Amazon VPC DHCP options set

The Dynamic Host Configuration Protocol (DHCP) provides a standard for passing configuration information to hosts on a TCP/IP network. The options field of a DHCP message contains configuration parameters such as `domain-name-servers`, `domain- name`, `ntp-servers`, and `netbios-node-type`. In any Amazon VPC, you can create DHCP options sets and specify up to four DNS servers. Currently, these options sets are created and applied per VPC, which means that you can't have a DNS server list at the Availability Zone level.

For more information about DHCP options sets and configuration, refer to the overview of DHCP option sets in the Amazon VPC Developer Guide.

Amazon Route 53 Resolver

Route 53 Resolver, also known as the Amazon DNS Server or Amazon Provided DNS, provides full public DNS resolution, with additional resolution for internal records for the VPC and customer-defined Route 53 private DNS records. Route 53 Resolver maps to a DNS server running on a reserved IP address at the base of the VPC network range, plus two. For example, the DNS Server on a `10.0.0.0/16` network is located at `10.0.0.2`. For VPCs with multiple Classless Inter-Domain Routing (CIDR) blocks, the DNS server IP address is located in the primary CIDR block.

Elastic Network Interfaces (ENIs)

Elastic network interfaces (referred to as *network interfaces* in the Amazon EC2 console) are virtual network interfaces that you can attach to an instance in a VPC. They're available only for instances running in a VPC. A virtual network interface, like any network adapter, is the interface that a device uses to connect to a network. Each instance in a VPC, depending on the instance type, can have multiple network interfaces attached to it.

For more information, refer to Elastic network interfaces in the *Amazon EC2 User Guide for Linux Instances*.

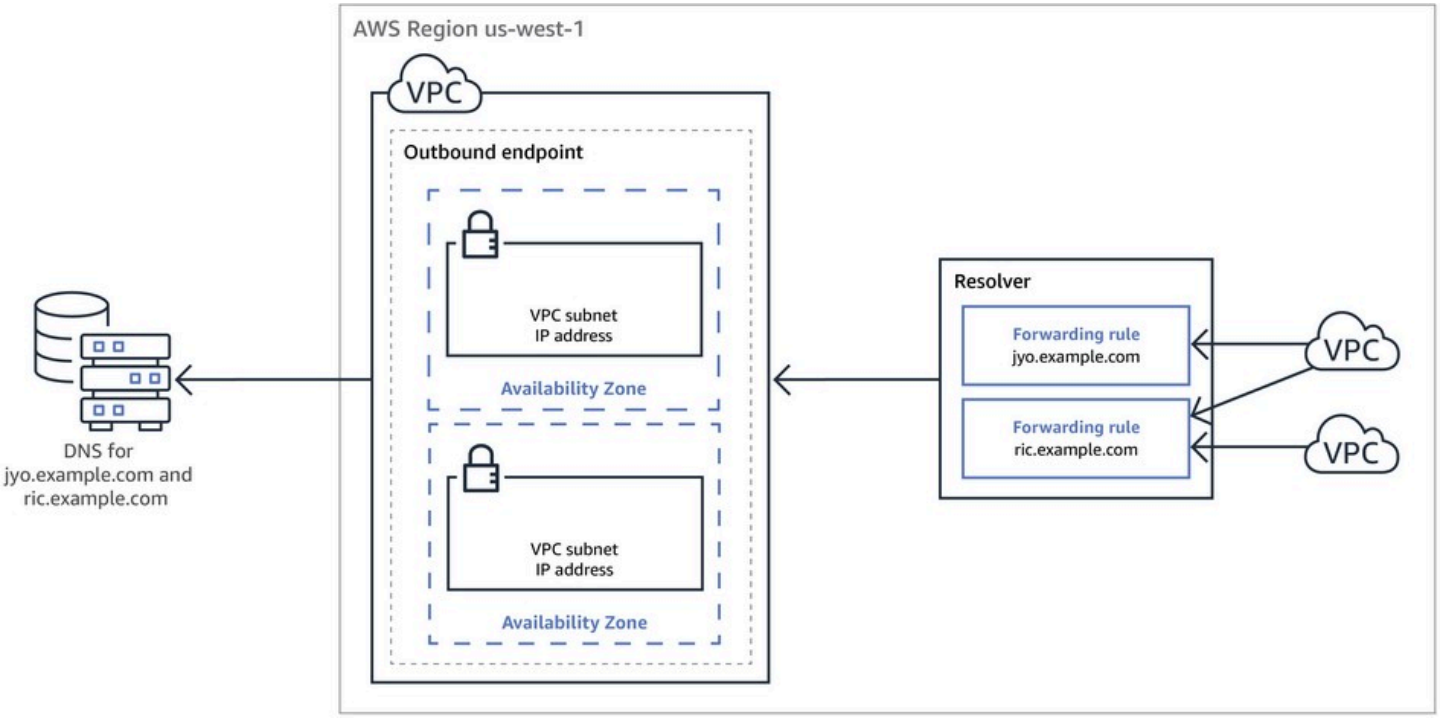
How ENIs work for Route 53 Resolver

A Route 53 Resolver endpoint is made up of one or more ENIs, which reside in your VPC. Each endpoint can only forward queries in a single direction.

Inbound endpoints are available as forwarding targets for DNS resolvers and use an IP address from the subnet space of the VPC to which it is attached. Queries forwarded to these endpoints have the DNS view of the VPC to which the endpoints are attached.

That means that if there are names local to the VPC, such as AWS PrivateLink endpoints, EFS clusters, EKS clusters, PHZs associated, and so on, the query can resolve any of those names. This is also true for any VPCs peered with the VPC that owns the endpoint.

Outbound endpoints serve as the path through which all queries are forwarded out of the VPC. Outbound endpoints are directly attached to the owner VPC and indirectly associated with other VPCs by rules. Therefore, if a forwarding rule is shared with VPC that does not own the outbound endpoint, all queries that match the forwarding rule pass through to the owner VPC and then forward out. It is important to realize this when you use queries to forward from one VPC to another. The outbound endpoint might reside in an entirely different Availability Zone than the VPC that originally sent the query, and there is potential for an Availability Zone outage in the owner VPC to impact query resolution in the VPC using the forwarding rule. This can be avoided by deploying outbound endpoints in multiple Availability Zones.



Route 53 Resolver with outbound endpoint

Refer to Getting Starting with Route 53 Resolver in the *Amazon Route 53 Developer Guide* for more information.

Route 53 PHZ

A Route 53 PHZ is a container that holds DNS records that are visible to one or more VPCs. VPCs can be associated to the PHZ at the time of or after the creation of the PHZ.

For more information, refer to Working with private hosted zones in the *Amazon Route 53 Developer Guide*.

Connection tracking

By default, Amazon EC2 security groups use connection tracking to track information about traffic to and from the instance. Security group rules are applied based on the connection state of the traffic to determine if the traffic is allowed or denied. This allows security groups to be *stateful*, which means that responses to inbound traffic are allowed to flow out of the instance regardless of outbound security group rules, and vice-versa.

Linux resolver

The stub resolver in Linux is responsible for initiating and sequencing DNS queries that ultimately lead to a full resolution. A resolver is configured through a configuration file, `/etc/resolv.conf`. The resolver queries the DNS server listed in the `resolv.conf` in the order they are listed.

The following is an example resolv.conf:

```
options timeout:1 nameserver 10.0.0.10
nameserver 10.0.1.10
```

Linux DHCP client

The Dynamic Host Configuration Protocol (DHCP) client on Linux provides the option to customize the set of DNS servers that the instance uses for DNS resolution. The DNS servers provided in the AWS DHCP options are picked up by this DHCP client to further update the resolv.conf with a list of DNS Server IP addresses. In addition, you can use the *supersede* DHCP client option to replace the

DNS servers provided by the AWS DHCP options set with a static list of DNS servers. You do this by modifying the DHCP client configuration file, `/etc/dhcp/dhclient.conf`:

```
interface "eth0"
{
supersede domain-name-servers 10.0.2.10, 10.0.3.10;
}
```

This example statement replaces DNS servers `10.0.0.10` and `10.0.1.10` in the resolv.conf sample with `10.0.2.10` and `10.0.3.10`. We discuss the use of this option in the *Zonal forwarders using supersede* solution.

Conditional forwarder: unbound

A conditional forwarder examines the DNS queries received from instances and forwards them to different DNS servers based on rules set in its configuration, typically using the domain name of the query to select the forwarder. In a hybrid architecture, conditional forwarders play a vital role to bridge name resolution between on-premises and cloud resources. For this particular solution we use *unbound*, which is conditional forwarder and a recursive and caching DNS resolver. Depending on your requirements, this option can act as an alternative or hybrid to forwarding rules in Amazon Route 53 Resolver.

For instructions on how to set up an Unbound DNS server, refer to the How to Set Up DNS Resolution Between On-Premises Networks and AWS by Using Unbound blog post in the AWS Security Blog.

The following is an example unbound`.conf`:

```
forward-zone:
name: "."
forward-addr: 10.0.0.2 # Amazon Provided DNS forward-zone:
name: "example.corp"
forward-addr: 192.168.1.10 # On-premises DNS
```

In this example, configuration queries to `example.corp` are forwarded to the on-premises DNS server, and the rest are forwarded to Route 53 Resolver.

Constraints

In addition to the concepts established so far, it is important to understand some constraints that are key in shaping the rest of this whitepaper and its solutions.

Packet per second (PPS) per elastic network interface limit

Each network interface in an Amazon VPC has a hard limit of 1024 packets that it can send to the Amazon-provided DNS server every second. Therefore, a computing resource on AWS that has a network interface attached to it and is sending traffic to the Amazon DNS resolver (for example, an Amazon EC2 instance or AWS Lambda function) falls under this hard-limit restriction. In this whitepaper, we refer to this limit as packet per second (PPS) per network interface. When you're designing a scalable solution for name resolution, you must consider this limit, because failure to do so can result in queries to Route 53 Resolver going unanswered if the limit is reached. This limit is a key factor to be considered for the solutions proposed in this whitepaper. This limit is higher for Route 53 resolver endpoints, which have a limit of approximately 10,000 queries per second (QPS) per elastic network interface.

Connection tracking

The number of simultaneous stateful connections that an Amazon EC2 security group can support by default is an extremely large value that the majority of standard TCP-based customers never encounter any issues with. In rare cases, customers with restrictive security group policies and applications that create a large number of concurrent connections, for instance a self-managed recursive DNS server, might run into issues of exhausting all simultaneous connection tracking resources. When that limit is exceeded, subsequent connections fail silently. In such cases, we recommend that you have a security group set up that you can use to disable connection tracking. To do this, set up permissive rules on both inbound and outbound connections.

Linux resolver

The default maximum number of DNS servers that you can specify in the resolv.conf configuration file of a Linux resolver is three, which means it isn't useful to specify four DNS servers in the DHCP options set because the additional DNS server won't be used. This limit further places an upper boundary on some of the solutions discussed in this whitepaper. It is also key to note that different operating systems can handle the assignment and failover of DNS queries differently.

Solutions

The solutions in this whitepaper present options and best practices to architect a DNS solution in the hybrid cloud, keeping in mind criteria such as ease of implementation, management overhead, cost, resilience, and the distribution of DNS queries directed toward the Route 53 Resolver. We cover the following solutions:

- **Route 53 Resolver endpoints and forwarding rules** – This solution focuses on using Route 53 Resolver endpoints to forward traffic between your Amazon VPC and on-premises data center over both AWS Direct Connect and Amazon VPN.
- **Secondary DNS in an Amazon VPC** – This solution focuses on using Route 53 to mirror on-premises DNS zones that can then be natively resolved from within VPCs, without the need for additional DNS forwarding resources.
- **Decentralized conditional forwarders** – This solution uses distributed conditional forwarders and provides two options for using them efficiently. While we use unbound as a conditional forwarder in some of these solutions, you can use any DNS server that supports conditional forwarding with similar features.
- **Scaling DNS management across multiple accounts and VPCs** – This solution walks through options for managing DNS names as you scale your hybrid DNS solution.

Route 53 Resolver endpoints and forwarding rules

Route 53 Resolver endpoints and forwarding rules allow you to forward traffic between your Amazon VPC and on-premises data center without having to deploy additional DNS servers.

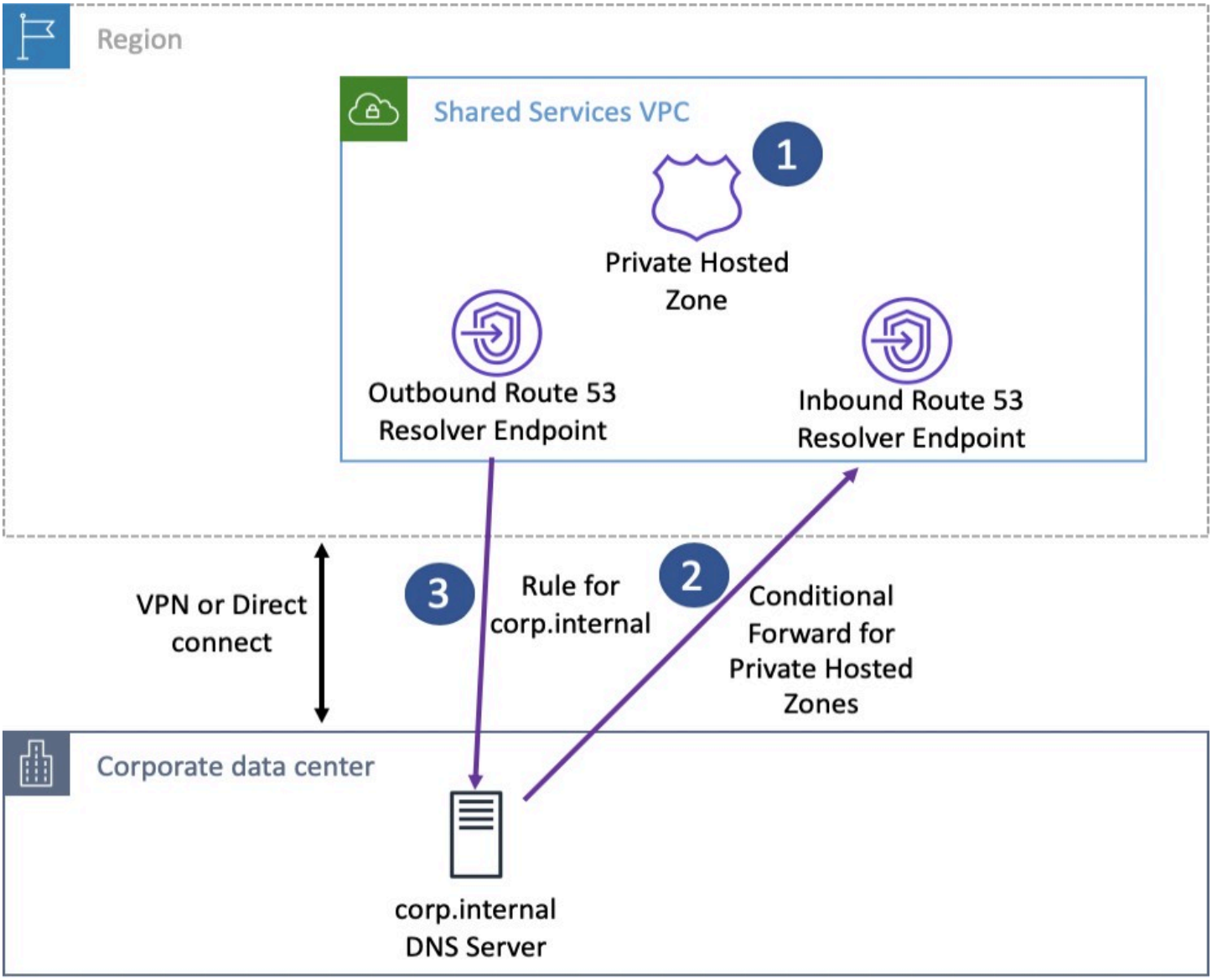
For more detailed information about Amazon Route 53 Resolver, refer to the Amazon Route 53 Resolver Developer Guide.

You use the following features of Route 53 resolver in this solution: *inbound endpoint*, *outbound endpoint*, and *forwarding rules* to make the hybrid resolution possible between on-premises and AWS.

Table 1 — Solution highlights - Route 53 Resolver endpoint

Use case	Advantages	Limitations
Customers that must forward queries between an Amazon VPC and on-premises data center Customers that have one or more VPCs connected to an on- premises environment through AWS Direct Connect or Amazon VPN	Low management overhead: you only have to manage forwarding rules and monitor query limits by using Amazon CloudWatch alarms. Uses the highly available AWS backbone	Approximately 10,000 QPS limit per elastic network interface on resolver endpoints No logging visibility on queries answered by Resolver Query source IP address is replaced with IP address of the endpoint from which it is forwarded

The following diagram represents a sample architecture utilizing Route 53 Resolver endpoints to resolve domain names in a hybrid fashion.



Route 53 Resolver endpoints and forwarding rules

1. Private Hosted Zones are associated with a Shared Service VPC.
2. Create forward rules in the on-premises DNS server for Route 53 names you want to resolve from on-premises. These rules use an inbound endpoint as their destination.
3. Create Route 53 Resolver rules for names you want to resolve on-premises from your Amazon VPC. These rules use an outbound endpoint and can be shared with other VPCs through Resource Access Manager (RAM).

Considerations:

- Though you can have multiple VPCs across many accounts, only a single, Availability Zone-redundant set of inbound endpoints is required in your Shared Services VPC.
- You need only one outbound endpoint for multiple VPCs. You don't have to create an outbound endpoint in each VPC. Instead, you share an outbound endpoint by sharing the rules created for that endpoint with additional accounts using RAM.
- Endpoints cannot be used across Regions.

Best practices:

- Manually specify the private IP addresses of the inbound Route 53 resolver endpoint while creating it, as opposed to having the resolver choose a random IP address from the subnet. This way, if there is an accidental deletion of the endpoint, you can reuse those IP addresses.
- When you create the inbound or outbound endpoints, we recommend that you use at least two subnets in different Availability Zones for high availability. For the inbound resolver, ensure that you use both endpoint IP addresses in your on-premises DNS resolver so the load can be spread across all available IP addresses.
- For environments that require a high number of queries per second, be aware that there is a limit of 10,000 queries per second per ENI in an endpoint. More ENIs can be added to an endpoint to scale QPS.
- We publish `InboundQueryVolume` and `OutboundQueryVolume` metrics through Amazon CloudWatch, and recommend that you set up monitoring rules that alert you if the threshold exceeds a certain value (for example, 80 percent of 10,000 QPS).

Secondary DNS in an Amazon VPC

Alternatively, you might decide to deploy and manage additional DNS infrastructure running on EC2 instances to handle DNS requests either from VPCs or on-premises, where you can still benefit from using AWS Managed Services.

This approach uses Route 53 private hosted zones with AWS Lambda and Amazon CloudWatch Events to mirror on-premises DNS zones. This can then be natively resolved from within a VPC without conditional forwarding, and without a real-time dependency on on-premises DNS servers.

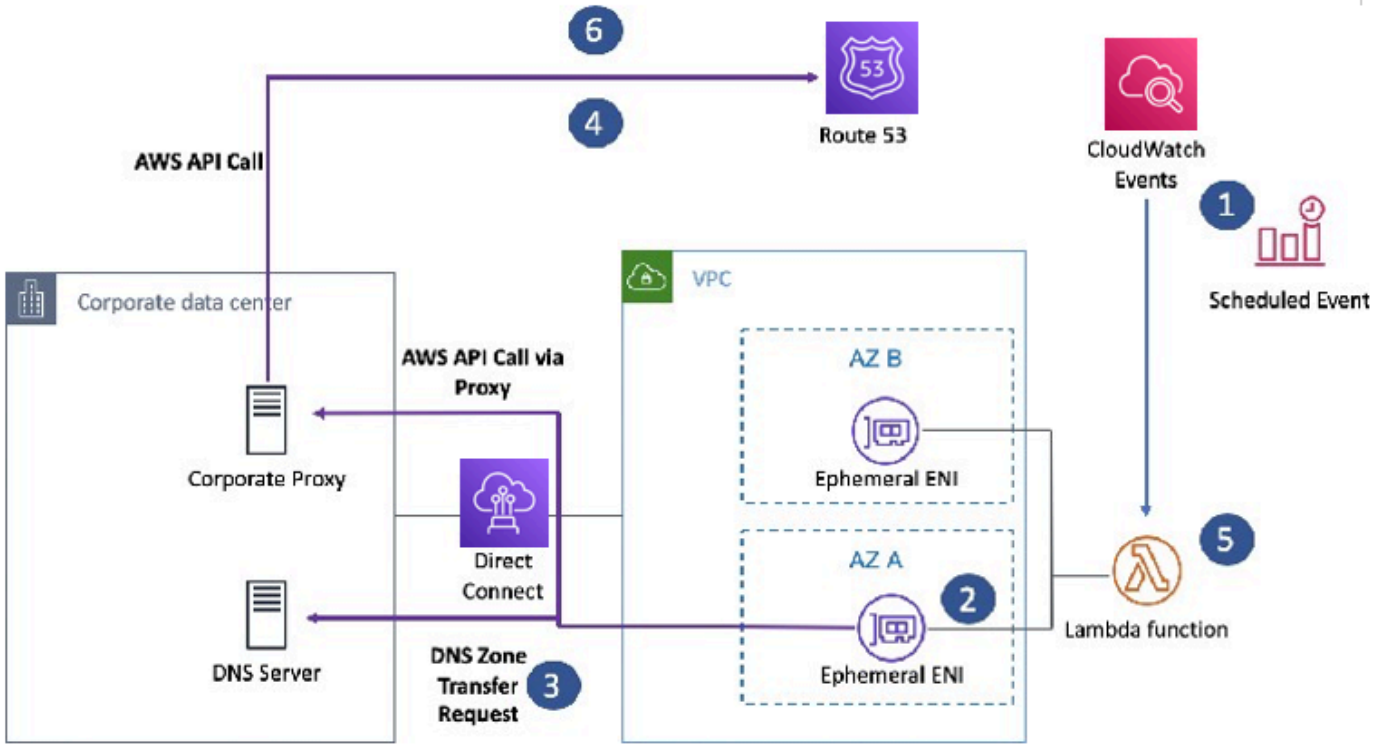
For the full solution, refer to Powering Secondary DNS in a VPC using AWS Lambda and Amazon Route 53 Private Hosted Zones on the AWS Compute blog.

The following table outlines this solution:

Table 2 – Solution highlights – secondary DNS in an Amazon VPC

Use case	Advantages	Limitations
Customers that cannot use the native Route 53 Resolver forwarding features Customers that don't want to build or manage conditional forwarder instances Customers that do not have in-house DevOps expertise Infrequently changing DNS environment	Low management overhead Low operational cost Highly resilient DNS infrastru cture Low possibility for instances to breach the PPS per network interface limit	On-premises instances can't query Route 53 Resolver directly for Amazon EC2 hostnames without creating a forwarding target Works well only when on- premises DNS server records must be replicated to Route 53 Requires on-premises DNS server to support full zone transfer query Requires working with the Route 53 API limits

The following diagram represents a sample architecture utilizing a secondary DNS within an Amazon VPC.



Secondary DNS running on Route 53 private hosted zones

1. CloudWatch Events invokes a Lambda function. The scheduled event is configured based on a JSON string that is passed to the Lambda function that sets a number of parameters, including the DNS domain, source DNS server, and Route 53 zone ID. This configuration lets you reuse a single Lambda function for multiple zones.
2. A new network interface is created in the VPC's subnets and attached to the Lambda function. This allows the function to access any internal network resources based on the security group that you defined.
3. The Lambda function transfers the source DNS zone from the IP address specified in the JSON parameters. Configure DNS servers to allow full zone transfers, which happen over TCP and UDP port 53.
4. The Route 53 DNS zone is retrieved using the AWS API.
5. The two zone files are compared, and then the resulting differences are returned as a set of actions to be performed using Route 53.
6. Updates to the Route 53 zone are made using the AWS API, and then the Start of Authority (SOA) is updated to match the source version.

There are several benefits to using this approach. Aside from the initial solution setup, there is little management overhead after the environment is set up as the solution continues working without any manual intervention. Also, there is no client-side setup because the DHCP options that you set configure each instance to use Route 53 Resolver (aka `AmazonProvidedDNS`) by default.

This solution can be one of the more scalable hybrid DNS solutions in a VPC because queries for any domain go directly to Route 53 Resolver from each instance and then to the Amazon Route 53 infrastructure. This ensures that each instance uses its own PPS per network interface limit. There is also no correlation and impact of implementing this solution in one or more VPCs as you choose to associate the Route 53 hosted zone with multiple VPCs. The possibility of failure of a DNS component is lower because of the highly available and reliable Amazon Route 53 infrastructure. Note, however, that there is a hard limit of 1,000 private hosted zone associations.

The main disadvantage of this solution is that it requires full zone transfer query so it isn't appropriate for customers that run DNS servers that don't support full zone transfer queries. Also, because this solution involves working with the Route 53 APIs, you must stay within the Route 53 API limits.

This solution does not provide a method for resolving EC2 records from on premises directly.

Decentralized conditional forwarders

While the Route 53 solution helps you avoid the complexities in running a hybrid DNS architecture, you might still prefer to configure your DNS infrastructure to use conditional forwarders within your VPCs. One reason to run your own forwarders is to log DNS queries. Refer to *DNS logging* (under *Additional considerations*) to determine if this is right for you.

There are two options under this solution. The first option, called *highly distributed forwarders,* discusses how to run forwarders on every instance of the environment trying to mimic the scale that the Route 53 solution provides. The second option, called *zonal forwarders using supersede,* presents a strategy of localizing forwarders to a specific Availability Zone and its instances.

The following table highlights these two options followed by their detailed discussion:

Table 3– Solution highlights – decentralized conditional forwarders

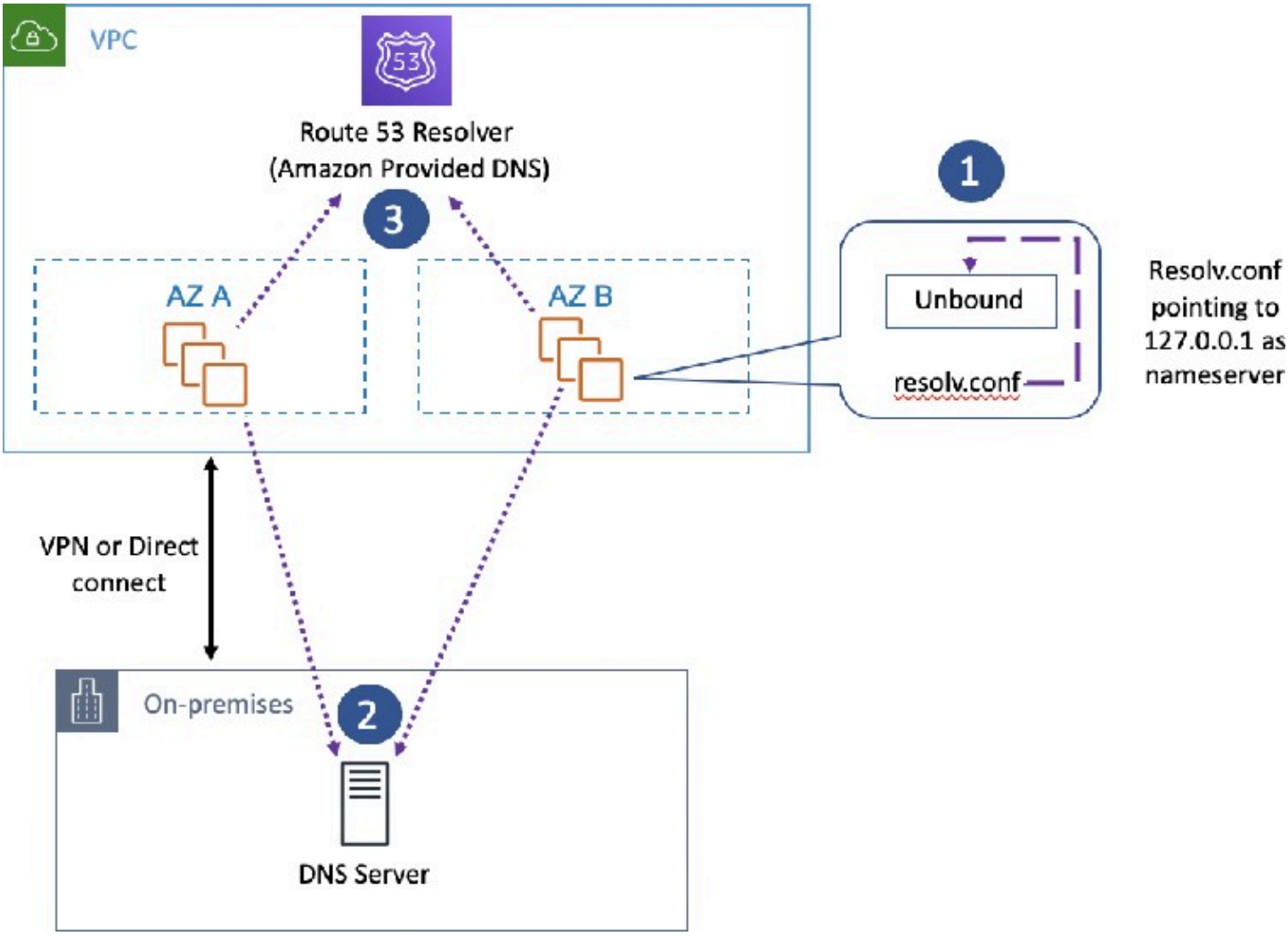
Option	Use case	Advantages	Limitations
Highly distributed forwarders	Workload generates high volumes of DNS queries Infrequently changing DNS environment	Resilient DNS infrastructure Low possibility for instances to breach the PPS per network interface limit	Complex setup and management Investment in relevant skill sets for configuration management
Zonal forwarders using supersede	Customers with existing set of conditional forwarders Environment that doesn't generate a high volume of DNS traffic	Fewer forwarders to manage Zonal isolation provides better overall resiliency	Complex setup and management as the DNS environment grows Possibility of breaching the PPS per network interfaces limit is higher than the highly distributed option

Highly distributed forwarders

This option decentralizes forwarders and runs a small lightweight DNS forwarder on every instance in the environment. The forwarder is configured to serve the DNS needs of only the instance it is running on, which reduces bottlenecks and dependency on a central set of instances.

Given the implementation and management complexity of this solution, we recommend that you use a mature configuration management solution.

The following diagram shows how this solution functions in a single VPC:



Distributed forwarders in a single VPC

1. Each instance in the VPC runs its own conditional forwarder (unbound). The resolv.conf has a single DNS Server entry pointing to `127.0.0.1`. A straightforward approach for modifying resolv.conf would be creating a DHCP options set that has `127.0.0.1` as the `domain-name-server` value. You may alternatively choose to overwrite any existing DHCP options settings using the supersede option in the `dhclient.conf`.
2. Records requested for on-premises hosted zones are forwarded to the on-premises DNS server by the forwarder running locally on the instance.
3. Any requests that don't match the on-premises forwarding filters are forwarded to Route 53 Resolver.

Similar to the Route 53 solution, this solution allows every single instance to use the limit of 1024 PPS per network interfaces to Route 53 Resolver to its full potential. The solution also scales up as additional instances are added and works the same way regardless of whether you're using a single or multi-VPC setup. The DNS infrastructure is low latency, and the failure of a DNS component such as an individual forwarder does not affect the entire fleet due to the decoupled nature of the design.

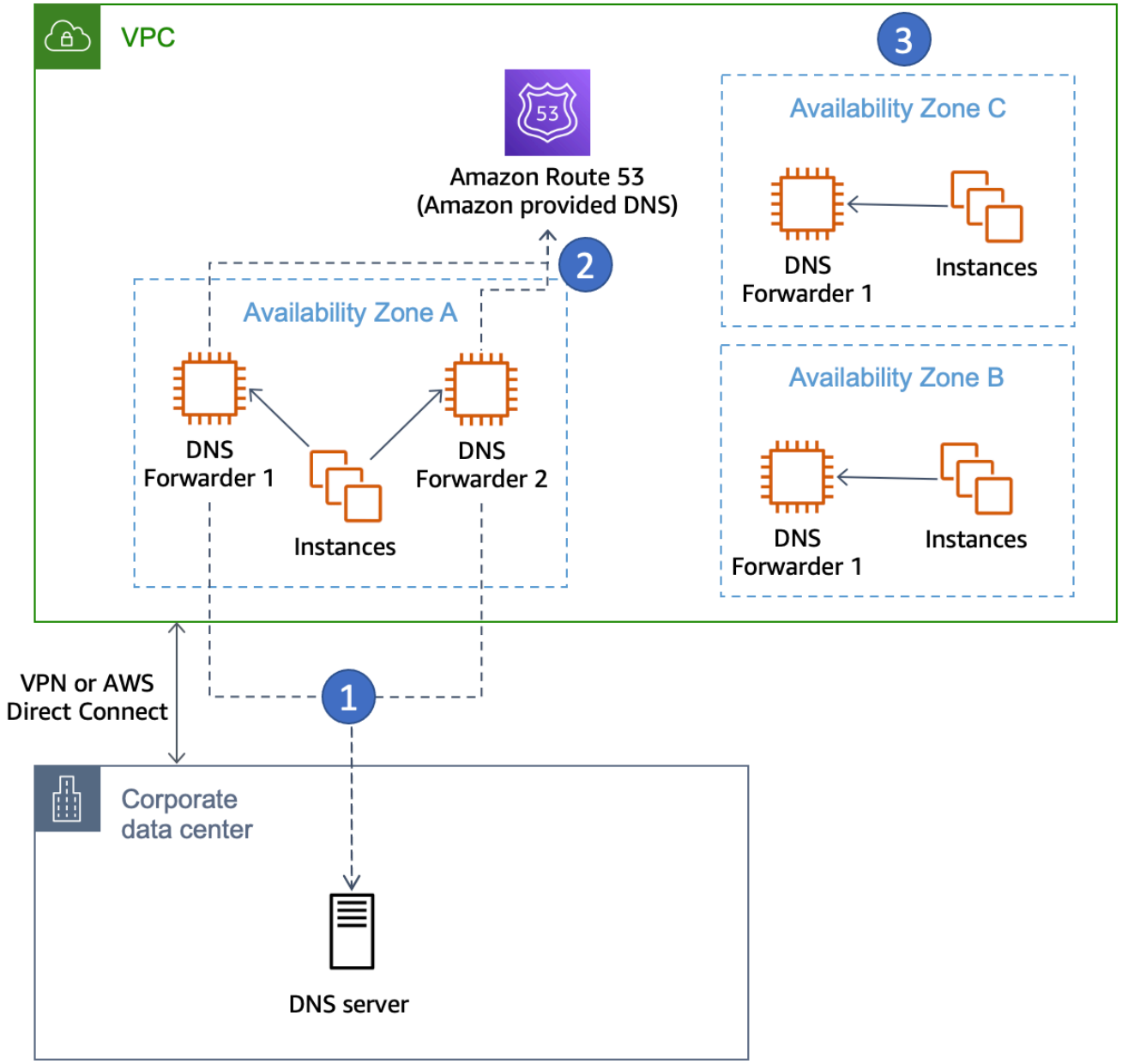
This solution poses implementation and management complexities, especially as the environment grows. You can manage and modify configuration files at instance launch using Amazon EC2 user data. After instance launch, you can use the Amazon EC2 Run command or AWS OpsWorks for Chef Automate to deploy and maintain your configuration files.

The implementation of these solutions is outside the scope of this whitepaper, but it is important to know that they provide the flexibility and power to manage configuration files and their state at a large scale. Greater flexibility brings with it the challenge of greater complexity. Consider additional operational costs, including the need to have an in-house DevOps workforce.

Zonal forwarders using supersede

If you don't want to manage and implement a forwarder on each instance of your environment, and you want to have conditional forwarder instances as the center piece of your hybrid DNS architecture, you should consider this option.

For this option, you localize instances in an Availability Zone (AZ) to forward queries to conditional forwarders only in the same Availability Zone of the Amazon VPC. For reasons discussed in the *Linux Resolver* section, each instance can have up to three DNS servers in their `resolv.conf`, as shown in the following diagram:



Zonal forwarders with supersede option

- Instances in Availability Zone A are configured using the supersede option, which uses a list of DNS forwarders that are local to that Availability Zone. To avoid burdening any specific forwarder in the Availability Zone, randomize the order for the DNS forwarders across instances in the Availability Zone.

1. Records requested for on-premises hosted zones are directly forwarded to the on- premises DNS server by the DNS forwarder.
2. Any requests that don't match the on-premises forwarding filters are forwarded to the Route 53 Resolver. This illustration doesn't depict the actual flow of traffic. It's presented for representation purposes
3. Similarly, other Availability Zones in the VPC can be set up to use their own set of local conditional forwarders that serve the respective Availability Zone. You determine the number of conditional forwarders serving an Availability Zone based on your need and the importance of the environment.

If one of the three instances in Availability Zone A fails, the other two instances continue serving DNS traffic. It is important to note that placement groups must be used in order to guarantee that the forwarders are not running on the same parent hardware, which is a single point of failure. To ensure separate parent hardware, you may set up and take advantage of Amazon Elastic Compute Cloud Placement Groups to avoid this type of failure domain.

If all three DNS forwarders in Availability Zone A fail at the same time, the instances in Availability Zone A fails to resolve any DNS requests because they are unaware of the presence of forwarders in other Availability Zones. This prevents the impact from spreading to multiple Availability Zones and ensures that other Availability Zones continue to function normally.

Currently, the DHCP options that you set apply to the VPC as a whole. Therefore, you must self-manage the list of DNS servers that are local to instances in each Availability Zone. In addition, we recommend that you don't use the same order of DNS servers in your `resolv.conf` for all instances in the Availability Zone, because it would burden the first server in the list and push it closer to breaching the PPS per network interfaces limit. Although each Linux instance can only have three resolvers, if you're managing the resolver list yourself, you can have as many resolvers as you want per Availability Zone. Each instance should be configured with three random resolvers from the resolver list.

Scaling DNS management across multiple accounts and VPCs

In alignment with AWS best practices, many organizations build out a cloud environment with multiple accounts. Whether you're using shared VPCs with multiple accounts hosted in a single VPC to share resources, or using the more traditional model where a VPC is tied to a single account, there are architectural considerations to make. This whitepaper focuses on the more traditional model.

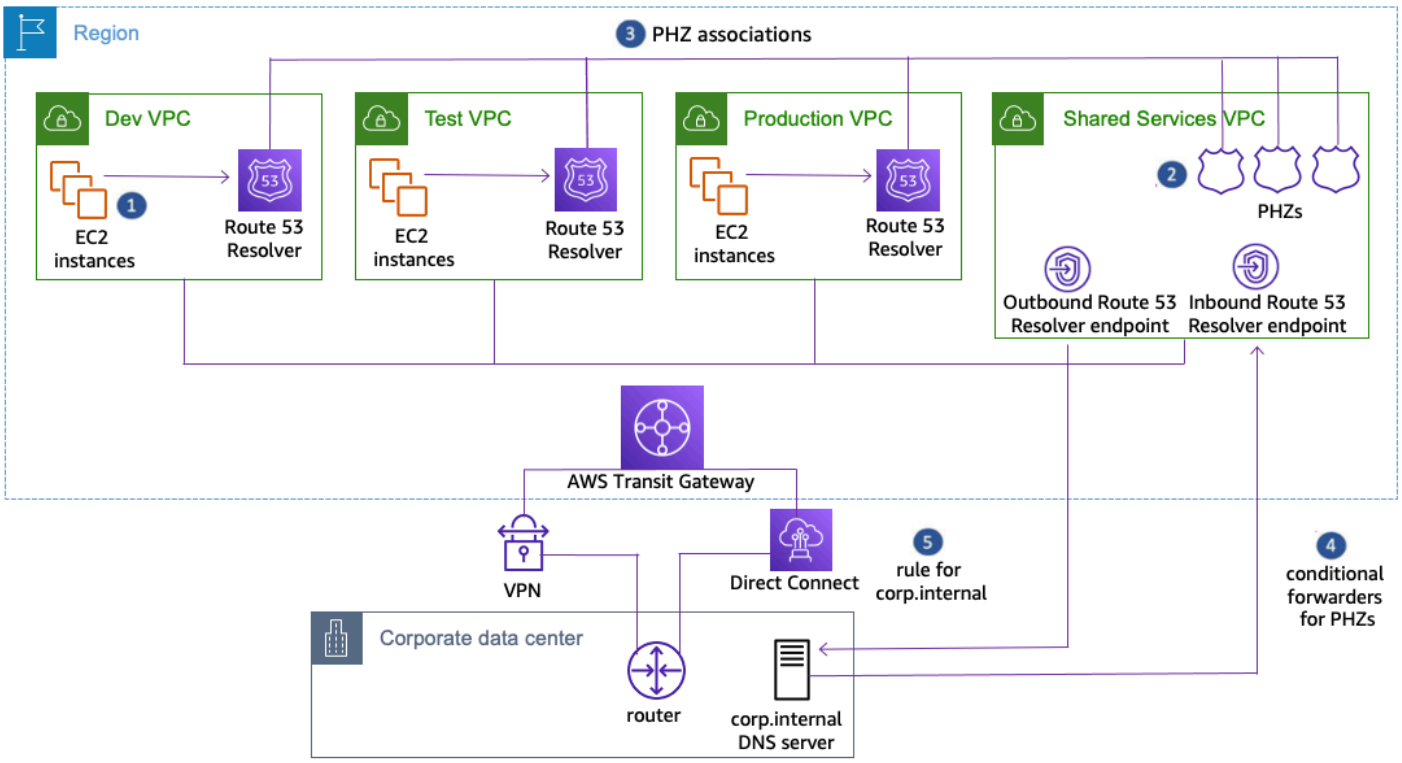
For more information on Shared VPCs, refer to Share your VPC with other accounts.

While having multiple accounts and VPCs helps provide a reduction of blast radius and granular account-level billing, it can make DNS infrastructure more complex. The Route 53 ability to associate PHZs with VPCs and accounts helps reduce these complexities for both centralized and decentralized architectures. We discuss both centralized and decentralized design paradigms in this section.

Multi-account centralized

In this type of architecture, Route 53 PHZs are centralized in a shared services VPC. This allows for central DNS management while enabling inbound Route 53 resolver endpoints to natively query the PHZs. This leaves the need for VPC-to-VPC DNS resolution unaddressed. Fortunately, PHZs can be associated with many VPCs. A simple command line interface (CLI) or API request can associate each PHZ with VPCs in accounts outside of the shared services VPC.

For more information about cross-account PHZ sharing, refer to Associating an Amazon VPC and a private hosted zone that you created with different AWS accounts.



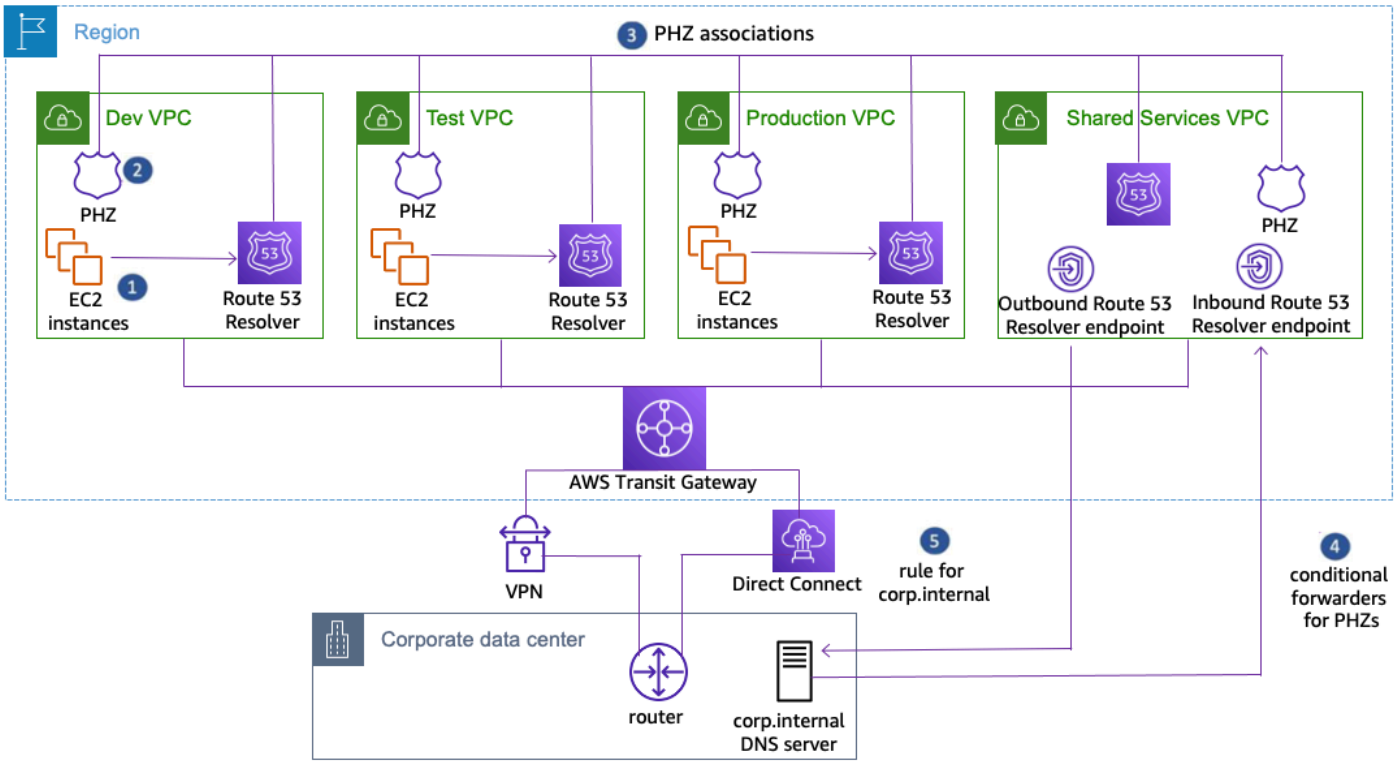
Multi-account centralized DNS with Private Hosted Zone sharing

1. Instances within a VPC use the Route 53 Resolver (Amazon-provided DNS).
2. PHZs are associated with a shared services VPC.
3. PHZs are also associated with other VPCs in the environment.
4. Conditional forward rule(s) from the on-premises DNS servers have an inbound Route 53 Resolver endpoint as their destination.
5. Rule(s) for on-premises domain names are created that use an outbound Route 53 Resolver endpoint.

While this architecture includes centralization, you might require each VPC to have its own fully qualified domain name (FQDN) hosted within each account, so that account owners can change and modify their own DNS records. The next section provides more information on how this design paradigm is accomplished.

Multi-account decentralized

An organization might want to delegate DNS ownership and management to each AWS account. Advantages of this method include decentralization of control and isolating the blast radius for failure to a specific account. The ability to associate PHZs to VPCs between accounts again becomes useful in this scenario. Each VPC can have its own PHZ(s) and then associate it with multiple other VPCs, across accounts and across Regions. This architecture is depicted in the following diagram. For unified resolution with the on-premises environment, this requires only that the shared services VPC be associated with each VPC hosting a PHZ.



Multi-account DNS decentralized

1. EC2 instances within a VPC use the Route 53 Resolver (Amazon-provided DNS).
2. PHZs are associated with a shared services VPC.
3. PHZs are also associated with other VPCs in the environment.
4. Conditional forward rules from the on-premises DNS servers have an inbound Route 53 Resolver endpoint as their destination.
5. Create rules for on-premises domain names that use an outbound Route 53 Resolver endpoint.

Alternative approaches

Alternative approaches have historically been to deploy DNS proxy servers in EC2 instances or to reply on Active Directory DNS servers. This centralization was desired, but did not take advantage of the benefits of using the Route 53 Resolver, and can cause scaling and availability constraints.

A common anti-pattern is to use Route 53 Resolver endpoints to centralize the management of DNS within a shared services VPC or Transit Gateway. This is done by creating both an inbound and

an outbound endpoint in the shared services VPC, then creating forwarding rules whose target is the IP address of the inbound endpoint in the centralized VPC. These rules are then associated with other VPCs, which use the inbound endpoint of the central VPC to resolve their DNS queries. This has the effect of allowing spoke VPCs to use the DNS view of the central VPC. For example, if you have an EFS mount in the central VPC, the spoke VPC can resolve the EFS mount's DNS name by forwarding its query to the inbound endpoint of the VPC where the file system is mounted.

This approach is not preferred. Cross-account sharing of PHZs is highly available and less costly than query forwarding. This is because PHZ sharing preserves Availability Zone isolation, meaning that your queries in VPC A are answered by an Availability Zone local to VPC A, whereas your queries in VPC B are answered by an Availability Zone local to VPC B. This means that, in the event of an availability problem in VPC A, VPC B's queries would not be affected, as long as they are in two different Availability Zones. There is no additional cost to associate a PHZ with a VPC and you can share a VPC with upwards of 1000 zones.

Query forwarding is optimized for sending queries to other DNS resolvers located outside the AWS network. It provides a way to allow DNS resolvers from different networks to access each other when they would normally not be visible via a recursive DNS lookup. If you choose to use query forwarding to resolve DNS answers local to another VPC, you would must get an endpoint for every VPC for which you want this view of DNS. Additionally, using endpoints to answer queries between VPCs breaks the previously mentioned Availability Zone isolation. This means that instead of each VPC resolving queries within its local Availability Zone, you have now made several VPCs dependent on the availability of a single VPC.

Regarding limits, each endpoint ENI has a limit of 10,000 QPS, but keep in mind that if you want to use an endpoint to centralize DNS management, you are forwarding more query volume to a central VPC instead of distributing the query load between multiple VPCs. This anti-pattern is generally not recommended.

Selecting the best solution for your organization

There are various advantages and trade-offs with each of these solutions. Choosing the right solution for your organization depends on the specific requirements of each workload. You might choose to run different solutions in different VPCs to meet the needs of your specific workloads. The following table summarizes the criteria that you can use to evaluate what will work best for your organization. These include the complexity of the implementation, the management overhead, the availability of the solution, probability of hitting the PPS per network interface limit, and the cost of the solution.

Table 5 – Solutions selection criteria

	Route 53 Resolver	Secondary DNS in a VPC	Highly distributed forwarders	Zonal forwarders
Implementation complexity	Low	Medium	High	High
Management overhead	Low	Low	High	Medium
DNS Infrastructure resiliency	High	High	High	Medium
PPS limit breach	Low	Low	Low	Medium
Cost*	Low	Low	High	Medium

* *Cost is a combination of the infrastructure and operational expense.*

Additional considerations

This section describes additional aspects that you should consider on top of the previously presented solutions. For example, DNS logging, or DNS server resiliency.

DNS logging

DNS logging refers to logging specific DNS query from individual host. Typically, these logs are stored for security forensics and compliance. Amazon GuardDuty provides machine learning (ML)-based forensics and anomaly detection on recursive queries originating from local VPC resources. If raw historical logging is not required, GuardDuty might satisfy your requirements without any additional heavy lifting.

Route 53 provides query logs for public hosted zones. If customers require logging for Private Hosted Zones and queries that originate from resources within a VPC, they have several options while still following the Well-Architected Framework and DNS best practices.

Centralized query logging, distributed (on-instance) query logging, and a hybrid approach to log a percentage of queries based on user-defined domain allowlisting, are three of the most popular and scalable methods for query logging currently available.

Centralized query logging

Query logging is accomplished in a centralized fashion when all queries are forwarded to a resolver that is not the Route 53 Resolver (Amazon-provided DNS). This resolver can be local to the VPC, such as several instances running unbound, or an on-premises resource over Direct Connect, VPN, or the Internet Gateway. The latter adds additional latency and dependencies outside of the VPC, and is typically not recommended for that reason. As with any centralized or distributed system, it comes with pros and cons.

Centralization of query logs allows for easy aggregation and a single plane of glass to view and parse DNS client queries. With centralization, additional attention needs be directed at the scale of the instances acting as resolvers and number of queries that are directed at any single instance. These instances become single points of failure and can become a barrier due to DNS packets per second limits. Each EC2 instance is limited to 1024 packets per second for DNS queries toward the Route 53 Resolver. If the requests being sent to the customer managed instance-based DNS resolvers are not distributed effectively and are not implementing caching techniques, with high

volume the DNS instances may exceed the 1024 per instance packet per second limit to the Route 53 Resolver DNS resolver with the VPC.

Distributed query logging

Another approach is logging DNS queries in a distributed fashion on instance. This is accomplished by running unbound or another logging capable resolver or forwarder on each instance that requires logging. With the distributed model of logging DNS queries, each instance runs a local resolver in order to capture all DNS queries locally on each instance. These logs can then be aggregated upstream to a centralized Amazon Simple Storage Service (Amazon S3) bucket for historical collection and centralized parsing. Depending on the aggregation process, this might create a delayed ability for centralized parsing and forensics, but removes any single points of failure and reduces the overall blast radius of any given upstream instance-based resolver failure. If On-Demand Instance parsing is required, the delivery window can be shorted. Depending on your operational model, you might want to allow on-box forensics or external access, so the logging delivery schedule should be considered.

VPC traffic mirroring is an alternative off-instance distributed logging mechanism that can be achieved for supported instance types. At this time, all AWS Nitro-based instances support VPC traffic mirroring. By enabling traffic mirroring for Transmission Control Protocol (TCP)- and User Diagram Protocol (UDP)-based traffic on port 53 on individual instance ENIs, you have the ability to capture DNS requests in pcap format. Traffic mirroring for DNS logs shares similar availability and scalability constructs as other distributed methods, but increases simplicity and flexibility because it does not require the application or Amazon Machine Image (AMI) to incorporate any additional DNS logic. A traffic mirroring session can be attached and detached to instance ENIs as needed. Traffic mirroring is priced per elastic network interface that traffic mirroring is enabled on, and the customer is responsible for configuring and managing the traffic mirror target.

For more information on Amazon VPC traffic mirroring, refer to Traffic Mirroring concepts.

Hybrid query logging

The third option is a hybrid approach that allows more granularity on what queries are filtered. Companies that are able to define “trusted” zones and “untrusted” zones might prefer this approach.

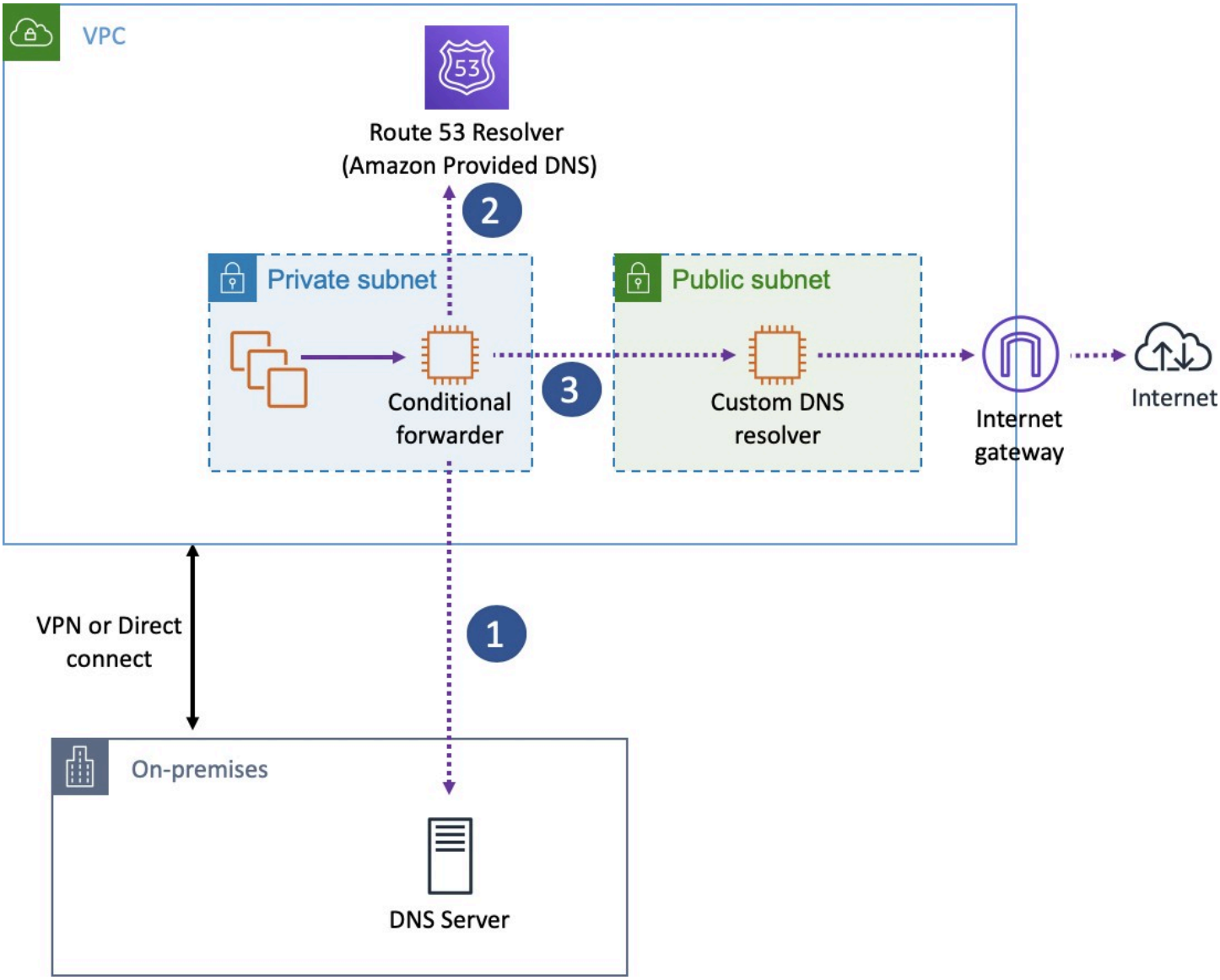
Trusted zones are approved by the organization and might not require logging. Anything unapproved falls under the *untrusted* category, to be logged and possibly acted upon, such as a

deny list of the response. For example, any zones that are owned and operated by the organization and VPC local resources are trusted, and everything else should be logged and controlled. This hybrid approach is now possible with the release of the Amazon Route 53 Resolver service, because of its ability to provide conditional forwarding rules by zone. In this approach, all local VPC resources resolve to Route 53 Resolver as usual, but when a query is made to an untrusted zone that matches an Amazon Route 53 Resolver conditional forwarding rule, it's forwarded to a specified instance or on-premises-based resolver such as the previously mentioned centralized DNS resolver. This approach does not require any modifications on the instance, and removes any single points of failure for all trusted zones.

Custom EC2 DNS Resolver

You can choose to host your own custom DNS resolver on Amazon EC2 that uses public DNS servers to perform recursive public DNS resolution instead of using Route 53 Resolver. This is a good choice because of the nature of the application and the ability to have more control and flexibility over the DNS environment. You could also do this if the PPS per network interface limit is a hindrance to your ability to scale and none of the solutions discussed thus far suit your needs.

This whitepaper does not describe the details of architecting such a solution, but we point out some caveats that will help you plan better in such a scenario. The following diagram illustrates an approach to a hybrid VPC DNS setup where you have your own DNS resolver on Amazon EC2.



Amazon EC2 DNS instances with segregated resolver and forwarder

1. DNS queries for internal EC2 names and Route 53 PHZs are forwarded to Route 53 Resolver.
2. DNS queries bound for on-premises servers are conditionally forwarded to on-premises DNS servers.
3. DNS queries for public domains are conditionally forwarded to the custom DNS resolver in the public subnet. The resolver then recursively resolves public domains using the latest root hints available from the Internet Assigned Number Authority (IANA).

For security reasons, we recommend that the conditional forwarder instance that requires connectivity to on-premises sits separately in a private subnet of the VPC. Because the custom DNS resolver must be able to query public DNS servers, it runs in its own public subnet of the VPC.

Ideally, you would have security group rules on the EC2 instance running the custom DNS resolver, but if this custom DNS resolver has high rates of querying out to the internet, then there is a possibility that you will hit connection tracking limits, as discussed in the *Connection tracking* section. To avoid this scenario, avoid connection tracking by itself by opening up all ports, TCP and UDP, to the whole world at the security group level, both inbound and outbound.

Because this grants permissive rules to the instance-level security group, you will have to handle the security of the instance at a different layer. At the least, we recommend you control the traffic entering into the entire public subnet by using Network Access Control Lists (network ACLs), which thereby restricts access to the instance. Alternatively, you could use application-level control mechanisms such as access-control provided by a DNS resolver such as unbound.

Custom DNS resolvers might develop a reputation upstream on the internet. If the instance is assigned a dynamic public IP address that belonged to another customer and previously earned a bad reputation, requests upstream could be throttled or even blocked. To avoid being throttled or blocked, consider assigning Elastic IP addresses to these resolver instances. This provides these IP addresses that talk to the upstream servers with the opportunity to build a good reputation over time that can be owned and maintained. Scaling concerns can be mitigated through the use of a DNS server fleet sitting behind a Network Load Balancer (NLB) that is configured with both TCP and UDP listener on port 53.

Microsoft Windows instances

Typically, Microsoft Windows instances are joined using Active Directory Domain Services (AD DS). In scenarios where you use the Amazon VPC DHCP options set, unlike the Linux resolver, you can set the full set of four DNS servers. You can set the DNS servers independently from the DHCP supplied IP address similar to the supersede option discussed earlier. This can be accomplished using Active Directory Group Policy or configuration management tools such as Amazon EC2 Run Command or AWS OpsWorks for Chef Automate mentioned earlier. In addition, the Windows DNS client also enables you to cache recently resolved queries, which reduces the overall demand on the primary DNS server.

The Windows DNS client service is designed to prompt a dynamic update from the DNS server if a change is made to its IP address information. When prompted, the DNS server updates the host record IP address for that computer (according to RFC 2136).

Microsoft DNS provides support for dynamic updates and this is enabled by default in any Active Directory integrated DNS zone. When you use a lightweight forwarder such as unbound for Windows instances, note that there isn't any support for these dynamic updates, and it can't support RFC 2126. If you want to do this, you should use the Microsoft DNS server as a primary for these instances.

Unbound – additional options

Unbound caches the results for subsequent queries until the time to live (TTL) expires, after which it forwards the request. By enabling the *prefetch* option in unbound, you can ensure that frequently used records are pre-fetched before they expire to keep the cache up-to-date. Also, if the on-premises DNS server is not available when the cache expires, unbound returns `SERVFAIL`. To protect yourself against such a situation, you can enable the serve-expired option to serve old responses from the cache with a TTL of zero in the response without waiting for the actual resolution to finish. After the resolution is completed, the response is cached for subsequent use.

DNS forwarder – forward first

Some DNS servers (notably `BIND`) include a *forward first* option enabled by default, which causes the server to query the forwarder first and, if there is no response, to recursively retry the internet DNS servers. For private DNS domains in this scenario, the internet DNS servers return an authoritative `NXDOMAIN`, which is a non-existent internet or intranet domain name. They might also return the public address if you're using split horizon DNS for public zones, which is used to provided different answers for private vs. public IP addresses. Therefore, it is critical to specify the `forward only` option, which specifies that retries are made against the forwarders, which means that you avoid ever seeing the response from public name servers. The unbound DNS server has the `forward first` option disabled by default.

DNS server resiliency

The solutions in this whitepaper are intended to provide high availability in the event that there is an issue with your primary DNS server. However, there are factors that can prevent or delay this failover from occurring. These factors include, but are not limited to, the timeout value in `resolv.conf`, configuration issues with the superseded DNS, or incorrect DHCP options settings. In some cases, these factors could impact the availability of applications that are dependent on name resolution. There are a few simple approaches to ensure the resilience of your forwarders in

case there is an issue with the underlying hardware or instance software. While these approaches don't eliminate the need for Well-Architected design, they can help you increase the overall resiliency of your solution.

EC2 instance recovery

In the case of an underlying hardware failure of a DNS forwarder instance, you can use EC2 instance recovery to start the instance on a new host. A recovered instance is identical to the original instance, including the instance ID, private IP addresses, Elastic IP addresses, and all instance metadata. To do this, you can create a CloudWatch alarm that monitors an EC2 instance and automatically recovers the instance if it becomes impaired. You can use the CloudWatch alarm to monitor issues like loss of network connectivity, loss of system power, software issues on the physical host, or hardware issues on the physical host that affect network reachability.

For more information about instance recovery, refer to Recover your instance in the *Amazon EC2 User Guide for Linux Instances*. For step-by-step instructions on using CloudWatch alarms to recover an instance, refer to Create alarms that stop, terminate, reboot, or recover an instance in the *Amazon EC2 User Guide for Linux Instances*.

Secondary IP address

In an Amazon VPC, instances can be assigned secondary IP addresses, which are transferrable. If an instance fails, the secondary IP can be transferred to a standby instance and this avoids the need for every instance to reconfigure their resolver IP addresses. This approach redirects traffic to the healthy instance so that it can respond to DNS queries. This approach is appropriate for scenarios where EC2 instance recovery might not provide fast enough recovery, or might not be appropriate (for example, an operating system fault or software issue). For more information about working with multiple IP addresses, refer to Multiple IP addresses in the *Amazon EC2 User Guide for Linux Instances*.

Conclusion

For organizations with on-premises resources, operating in a hybrid architecture is a necessary part of the cloud adoption process. As such, architecture patterns that streamline this transition are essential for success.

We discussed concepts as well as constraints to help you better understand the fundamental building blocks of the solutions provided here, as well as the limitations that help to create the most optimal solution for your workload. The provided solutions included how to use Route 53 Resolver endpoints with conditional forwarding rules, how to set up Secondary DNS in the Amazon VPC with AWS Lambda and Route 53 Private hosted zones, and solutions that use decentralized forwarders using the unbound DNS server. We also provided guidance on how to select the appropriate solution for your intended workload. Finally, we examined some additional considerations to help you to better tailor your solution for different workload requirements, faster failover, and better DNS server resiliency.

By using the architectures provided, you can achieve the most ideal private DNS interoperability between your on-premises environments and your Amazon VPC.

Contributors

Contributors to this document include:

- Anthony Galleno, Senior Technical Account Manager
- Gavin McCullagh, Principal Systems Development Engineer
- Gokul Bellala Kuppuraj, Technical Account Manager
- Harsha Warrdhan Sharma, Technical Account Manager
- James Devine, Senior Specialist Solutions Architect
- Justin Davies, Principal Network Specialist
- Maritza Mills, Senior Product Manager - Technical
- Sohamn Chaterjee, Cloud Infrastructure Architect
- Pablo Sánchez Carmona, Network Specialist Solutions Architect
- Ivo Pinto, Solutions Architect
- Tyler Applebaum, Solutions Architect

Document revisions

To be notified about updates to this whitepaper, subscribe to the RSS feed.

Change	Description	Date
Document updated	Fifth publication.	December 2, 2022
Document updated	Minor updates.	November 1, 2019
Document updated	Fourth publication.	September 1, 2019
Document updated	Third publication.	June 1, 2018
Document updated	Second publication.	November 1, 2017
Initial publication	Whitepaper published.	October 1, 2017

Notices

Customers are responsible for making their own independent assessment of the information in this document. This document: (a) is for informational purposes only, (b) represents current AWS product offerings and practices, which are subject to change without notice, and (c) does not create any commitments or assurances from AWS and its affiliates, suppliers, or licensors. AWS products or services are provided “as is” without warranties, representations, or conditions of any kind, whether express or implied. The responsibilities and liabilities of AWS to its customers are controlled by AWS agreements, and this document is not part of, nor does it modify, any agreement between AWS and its customers.

AWS Glossary

For the latest AWS terminology, see the AWS glossary in the *AWS Glossary Reference*.