# Table of Contents

What is Route 53 ARC? ........................................................................................................................................ 1  
Components ................................................................................................................................................ 2  
  Zonal shift components .............................................................................................................................. 2  
  Readiness check components .................................................................................................................... 3  
  Routing control components ....................................................................................................................... 6  
AWS Regions .............................................................................................................................................. 8  
How it works ................................................................................................................................................ 11  
  Monitoring your application replica with readiness checks .................................................................... 11  
  Rerouting traffic for recovery with routing control .................................................................................. 12  
  Moving traffic away from an Availability Zone with zonal shift .............................................................. 13  
  Data and control planes for Route 53 ARC .................................................................................................. 13  
Zonal shift and routing control ..................................................................................................................... 14  
Use cases .................................................................................................................................................... 15  
Tagging ......................................................................................................................................................... 16  
Pricing .......................................................................................................................................................... 17  
Getting started with multi-Region .............................................................................................................. 18  
  Recovery readiness with an existing application ....................................................................................... 19  
  Recovery readiness with a new application ............................................................................................... 20  
    How to create an example application .................................................................................................... 20  
    Download AWS CloudFormation or HashiCorp Terraform templates .................................................... 21  
Routing control for traffic failover ............................................................................................................ 21  
Working with AWS SDKs ............................................................................................................................ 21  
Examples of using API operations .............................................................................................................. 23  
Readiness check with the CLI ..................................................................................................................... 23  
  1. Create cells ........................................................................................................................................ 23  
  2. Create a recovery group ......................................................................................................................... 24  
  3. Create a resource set ............................................................................................................................ 25  
  4. Create a readiness check ...................................................................................................................... 27  
  5. Monitor readiness checks ..................................................................................................................... 28  
Routing control with the CLI ..................................................................................................................... 30  
  1. Create a cluster .................................................................................................................................... 30  
  2. Create a control panel ........................................................................................................................... 32  
  3. Create a routing control ....................................................................................................................... 33  
  4. Create safety rules ............................................................................................................................... 35  
  5. Create health checks ............................................................................................................................ 37  
Updating control states with the CLI ........................................................................................................... 39  
Zonal shift with the CLI .............................................................................................................................. 41  
  Start zonal shift ...................................................................................................................................... 41  
  Get managed resource ............................................................................................................................. 42  
  List managed resources ............................................................................................................................ 42  
  List zonal shifts ........................................................................................................................................ 42  
  Update zonal shift .................................................................................................................................. 43  
  Cancel zonal shift .................................................................................................................................... 43  
Best practices ............................................................................................................................................. 44  
  Best practices for recovery ....................................................................................................................... 44  
  Best practices for zonal shifts .................................................................................................................. 44  
  Best practices for readiness checks and routing controls ....................................................................... 45  
Actions ....................................................................................................................................................... 47  
  Recovery readiness actions ...................................................................................................................... 47  
  Recovery control configuration actions .................................................................................................... 49  
  Routing control actions ............................................................................................................................ 50  
  Zonal shift actions ................................................................................................................................... 51  
Zonal shift .................................................................................................................................................. 52  
  How a zonal shift works ............................................................................................................................ 52
What is Amazon Route 53 Application Recovery Controller?

Amazon Route 53 Application Recovery Controller provides three distinct capabilities: **readiness checks**, **routing controls**, and **zonal shifts**. With Route 53 ARC, you can gain insights into whether your applications and resources are prepared for recovery, and quickly mitigate impairments for a multi-Availability Zone or multi-Region application. Route 53 ARC helps you prepare for and accomplish faster recovery operations for applications running on AWS.

The AWS Global Cloud Infrastructure provides fault tolerance and resilience, with each AWS Region comprised of multiple, fully-isolated Availability Zones. Route 53 ARC works within this AWS structure to help your applications be resilient.

**Multi-AZ recovery**

**Zonal shifts** enable you to quickly recover from Availability Zone issues, by temporarily moving traffic for a resource away from an Availability Zone. Starting a zonal shift helps your application recover quickly, for example, from a developer's bad code deployment or from an AWS infrastructure failure in a single Availability Zone, reducing the impact and time lost from an issue in one zone.

You can start a zonal shift for any managed resource in your account in a Region. Supported AWS resources are automatically registered with Route 53 ARC. Resources that are registered for zonal shifts in Route 53 ARC are managed resources in Route 53 ARC.

Zonal shifts are temporary. You must specify an expiration when you start a zonal shift, of up to three days initially. If you want to still keep traffic away from an Availability Zone, you can update the zonal shift and set a new expiration.

**Multi-Region recovery**

**Routing controls** enable you to re-balance traffic across application replicas during failures, so that you can ensure that your application is available. Safety rules help protect you from unintended outcomes by imposing guard rails that you define. Using these rules, you can make sure, for example that only one of your endpoints, active or standby, is enabled and in service at a time.

For multi-Region recovery, Route 53 ARC can help you centrally coordinate failovers across multiple AWS Regions. Extremely reliable routing controls enable you to recover applications by rerouting traffic, for example, across Regions. To do this, you partition your applications into redundant failure-containment units, or replicas. The boundary of each replica can be a Region or an Availability Zone, or even a smaller unit.

**Readiness checks** continually monitor AWS resource quotas, capacity, and network routing policies, and can notify you about changes that would affect your ability to fail over to a replica and recover. Continual readiness checks help make sure that, on an ongoing basis, your multi-Region applications are scaled and configured to handle failover traffic.

**Topics**

- Amazon Route 53 Application Recovery Controller components (p. 2)
- Amazon Route 53 Application Recovery Controller AWS Region availability (p. 8)
- How Amazon Route 53 Application Recovery Controller works (p. 11)
Amazon Route 53 Application Recovery Controller components

This section defines the components included in Amazon Route 53 Application Recovery Controller zonal shift, readiness check, and routing control features.

- the section called “Zonal shift components”
- the section called “Readiness check components”
- the section called “Routing control components”

Zonal shift components

The following diagram illustrates an example of a zonal shift moving traffic away from an Availability Zone in an AWS Region. Safety rules built into Route 53 ARC prevent you from starting another zonal shift for a resource when it already has an active zonal shift.

The following are components of the zonal shift feature in Route 53 ARC.
Zonal shift

You start a zonal shift for a managed resource in your AWS account to temporarily move traffic away from an Availability Zone in an AWS Region. Supported AWS resources are automatically registered with Route 53 ARC, and then they are managed resources for zonal shifts in your account. Currently you can start a zonal shift only for Network Load Balancers and Application Load Balancers that do not have cross-zone load balancing configured.

Starting a zonal shift helps your application quickly recover, for example, from a developer’s bad code deployment or from an AWS infrastructure failure in a single Availability Zone, reducing the impact and time lost from an issue in one zone.

Resource identifier

The identifier for a resource to include in a zonal shift. The identifier is the Amazon Resource Name (ARN) for the resource.

You can only include in a zonal shift the resources in your account that are in an AWS service that is supported by Route 53 ARC. Resources in those AWS services are registered with Route 53 ARC by the AWS service.

Note

At this time, you can only start a zonal shift for Network Load Balancers and Application Load Balancers with cross-zone load balancing turned off.

Managed resource

AWS services register resources automatically with Route 53 ARC for zonal shifts. A resource that has been registered is a managed resource in Route 53 ARC.

Resource name

The name of a managed resource in Route 53 ARC.

Status (zonal shift status)

A status for a zonal shift. The status for a zonal shift can have one of the following values:

- **ACTIVE**: The zonal shift is started and active.
- **EXPIRED**: The zonal shift has expired (the expiry time was exceeded).
- **CANCELED**: The zonal shift was canceled.

Expiry time (expiration time)

The expiry time (expiration time) for the zonal shift. Zonal shifts are temporary. A zonal shift can initially be set to be active for up to three days (72 hours).

When you start a zonal shift, you specify how long you want it to be active, which Route 53 ARC converts to an expiry time (expiration time). You can cancel a zonal shift, for example, if you're ready to restore traffic to the Availability Zone. Or you can extend a zonal shift by updating it to specify another length of time to expire in.

Readiness check components

The following diagram illustrates a sample recovery group that is configured to support the readiness check feature. Resources in this example are grouped into cells (by AWS Region) and nested cells (by Availability Zones) in a recovery group. There is an overall readiness status for the recovery group (application), as well as individual readiness statuses for each cell (Region) and nested cell (Availability Zone).
Recovery group

Cell

REGION 1

Cell

REG1-AZ1

ASG 1

DDB 1

Not ready

Ready

Ready
The following are components of the readiness check feature in Route 53 ARC.

**Cell**

A cell defines your application's replicas or independent units of failover. It groups all the AWS resources that are necessary for your application to run independently within the replica. For example, you might have one set of resources in a primary cell and another set in a standby cell. You determine the boundary of what a cell includes, but cells typically represent an Availability Zone or a Region. You can have multiple cells (nested cells) within a cell, such as AZs within a Region. Each nested cell represents an isolated unit of failover.

**Recovery group**

Cells are collected into a recovery group. A recovery group represents an application or group of applications that you want to check failover readiness for. It consists of two or more cells, or replicas, that match each other in terms of functionality. For example, if you have a web application that is replicated across us-east-1a and us-east-1b, where us-east-1b is your failover environment, you can represent this application in Route 53 ARC as a recovery group with two cells: one in us-east-1a and one in us-east-1b. A recovery group can also include a global resource, such as a Route 53 health check.

**Resources and resource identifiers**

When you create components for readiness checks in Route 53 ARC, you specify a resource, such as an Amazon DynamoDB table, a Network Load Balancer, or a DNS target resource, by using a resource identifier. A resource identifier is either the Amazon Resource Name (ARN) for the resource or, for a DNS target resource, the identifier that Route 53 ARC generates when it creates the resource.

**DNS target resource**

A DNS target resource is the combination of your application's domain name and other DNS information, such as the AWS resource that the domain points to. Including an AWS resource is optional but if you provide it, it must be a Route 53 resource record or a Network Load Balancer. When you provide the AWS resource, you can get more detailed architectural recommendations that can help you improve your application's recovery resiliency. You can create resource sets in Route 53 ARC for DNS target resources, and then create a readiness check for the resource set so that you can get architecture recommendations for your application. The readiness check also monitors the DNS routing policy for your application, based on the readiness rules for DNS target resources.

**Resource set**

A resource set is a set of resources, including AWS resources or DNS target resources, that span multiple cells. For example, you might have a load balancer in us-east-1a and another one in us-east-1b. To monitor the recovery readiness of the load balancers, you can create a resource set that includes both load balancers, and then create a readiness check for the resource set. Route 53 ARC will continually check the readiness of the resources in the set. You can also add a readiness scope to associate resources in a resource set with the recovery group that you create for your application.

**Readiness rule**

Readiness rules are audits that Route 53 ARC performs against a set of resources in a resource set. Route 53 ARC has a set of readiness rules for each type of resource that it supports readiness checks for. Each rule includes an ID and a description that explains what Route 53 ARC inspects the resources for.

**Readiness check**

A readiness check monitors a resource set in your application, such as a set of Amazon Aurora instances, that Route 53 ARC is auditing recovery readiness for. Readiness checks can include auditing, for example, capacity configurations, AWS quotas, or routing policies. For example, if you want to audit readiness for your Amazon EC2 Auto Scaling groups across two Availability Zones, you can create a readiness check for a resource set with two resource ARNs, one for each Auto Scaling
group. Then, to make sure that each group is scaled equally, Route 53 ARC continually monitors the instance types and the counts in the two groups.

Readiness scope

A readiness scope identifies the grouping of resources that a specific readiness check encompasses. The scope of a readiness check can be a recovery group (that is, global to the whole application) or a cell (that is, a Region or Availability Zone). For a resource that is a global resource for Route 53 ARC, set the readiness scope at to recovery group or global resource level. For example, a Route 53 health check is a global resource in Route 53 ARC because it isn't specific to a Region or Availability Zone.

Routing control components

The following diagram illustrates an example of components that support the routing control feature in Route 53 ARC. The routing controls shown here (grouped into one control panel) let you manage traffic to two Availability Zones in each of two Regions. When you update routing control states, Route 53 ARC changes health checks in Amazon Route 53, which redirect DNS traffic to different cells. Safety rules that you configure for routing controls help avoid fail-open scenarios and other unintentional consequences.
Routing controls

Control panel

Routing control 3

REGION 1

REG1-AZ1

ON
The following are components of the routing control feature in Route 53 ARC.

**Cluster**

A cluster is a set of five redundant Regional endpoints against which you initiate API calls to update or get routing control states. A cluster includes a default control panel, and you can host multiple control panels and routing controls on one cluster.

**Routing controls**

A routing control is a simple on/off switch, hosted on a cluster, that you use to control routing of client traffic in and out of cells. When you create a routing control, you add a Route 53 ARC health check in Route 53. This enables you to reroute traffic (using the health checks, configured with DNS records for your applications) when you update the routing control state in Route 53 ARC.

**Routing control health check**

Routing controls are integrated with health checks in Route 53. The health checks are associated with DNS records that front each application replica, for example, failover records. When you change routing control states, Route 53 ARC updates the corresponding health checks, which redirect traffic—for example, to failover to your standby replica.

**Control panel**

A control panel groups together a set of related routing controls. You can associate multiple routing controls with one control panel, and then create safety rules for the control panel to ensure that the traffic redirection updates that you make are safe. For example, you can configure a routing control for each of your load balancers in each Availability Zone, and then group them in the same control panel. Then you can add a safety rule (an “assertion rule”) that makes sure that at least one zone (represented by a routing control) is active at any one time, to avoid unintended “fail-open” scenarios.

**Default control panel**

When you create a cluster, Route 53 ARC creates a default control panel. By default, all routing controls that you create on the cluster are added to the default control panel. Or, you can create your own control panels to group related routing controls.

**Safety rule**

Safety rules are rules that you add to Route 53 ARC to ensure that recovery actions don’t accidentally impair your application’s availability. For example, you can create a safety rule that creates a routing control that acts as an overall “on/off” switch so that you can enable or disable a set of other routing controls.

**Endpoint (cluster endpoint)**

Each cluster in Route 53 ARC has five Regional endpoints that you can use for setting and retrieving routing control states. Your process for accessing the endpoints should assume that Route 53 ARC regularly brings the endpoints up and down for maintenance, so you should try each endpoint in succession until you connect to one. You access the endpoints to get the current state of routing controls (On or Off) and to trigger failovers for your applications by changing routing control states.

Amazon Route 53 Application Recovery Controller

AWS Region availability

For detailed information about Regional support and service endpoints for Amazon Route 53 Application Recovery Controller, see Amazon Route 53 Application Recovery Controller endpoints and quotas in the Amazon Web Services General Reference.
Note
Readiness check and routing control in Amazon Route 53 Application Recovery Controller are global features. However, you must specify the US West (Oregon) Region (that is, specify the parameter `--region us-west-2`) in Regional Route 53 ARC AWS CLI commands. That is, when you create resources such as recovery groups, readiness checks, or clusters.

Zonal shift in Route 53 ARC is currently available in the following AWS Regions:

<table>
<thead>
<tr>
<th>Region Name</th>
<th>Region</th>
<th>Endpoint</th>
<th>Protocol</th>
</tr>
</thead>
<tbody>
<tr>
<td>US East (Ohio)</td>
<td>us-east-2</td>
<td>arc-zonal-shift.us-east-2.amazonaws.com</td>
<td>HTTPS</td>
</tr>
<tr>
<td>US East (N. Virginia)</td>
<td>us-east-1</td>
<td>arc-zonal-shift.us-east-1.amazonaws.com</td>
<td>HTTPS</td>
</tr>
<tr>
<td>US West (N. California)</td>
<td>us-west-1</td>
<td>arc-zonal-shift.us-west-1.amazonaws.com</td>
<td>HTTPS</td>
</tr>
<tr>
<td>US West (Oregon)</td>
<td>us-west-2</td>
<td>arc-zonal-shift.us-west-2.amazonaws.com</td>
<td>HTTPS</td>
</tr>
<tr>
<td>Africa (Cape Town)</td>
<td>af-south-1</td>
<td>arc-zonal-shift.af-south-1.amazonaws.com</td>
<td>HTTPS</td>
</tr>
<tr>
<td>Asia Pacific (Hong Kong)</td>
<td>ap-east-1</td>
<td>arc-zonal-shift.ap-east-1.amazonaws.com</td>
<td>HTTPS</td>
</tr>
<tr>
<td>Asia Pacific (Hyderabad)</td>
<td>ap-south-2</td>
<td>arc-zonal-shift.ap-south-2.amazonaws.com</td>
<td>HTTPS</td>
</tr>
<tr>
<td>Asia Pacific (Jakarta)</td>
<td>ap-southeast-3</td>
<td>arc-zonal-shift.ap-southeast-3.amazonaws.com</td>
<td>HTTPS</td>
</tr>
<tr>
<td>Asia Pacific (Melbourne)</td>
<td>ap-southeast-4</td>
<td>arc-zonal-shift.ap-southeast-4.amazonaws.com</td>
<td>HTTPS</td>
</tr>
<tr>
<td>Asia Pacific (Mumbai)</td>
<td>ap-south-1</td>
<td>arc-zonal-shift.ap-south-1.amazonaws.com</td>
<td>HTTPS</td>
</tr>
<tr>
<td>Asia Pacific (Osaka)</td>
<td>ap-northeast-3</td>
<td>arc-zonal-shift.ap-northeast-3.amazonaws.com</td>
<td>HTTPS</td>
</tr>
<tr>
<td>Asia Pacific (Seoul)</td>
<td>ap-northeast-2</td>
<td>arc-zonal-shift.ap-northeast-2.amazonaws.com</td>
<td>HTTPS</td>
</tr>
<tr>
<td>Region Name</td>
<td>Region</td>
<td>Endpoint</td>
<td>Protocol</td>
</tr>
<tr>
<td>--------------------------</td>
<td>---------------</td>
<td>--------------------------------------------------------</td>
<td>----------</td>
</tr>
<tr>
<td>Asia Pacific (Singapore)</td>
<td>ap-southeast-1</td>
<td>arc-zonal-shift.ap-southeast-1.amazonaws.com</td>
<td>HTTPS</td>
</tr>
<tr>
<td>Asia Pacific (Sydney)</td>
<td>ap-southeast-2</td>
<td>arc-zonal-shift.ap-southeast-2.amazonaws.com</td>
<td>HTTPS</td>
</tr>
<tr>
<td>Asia Pacific (Tokyo)</td>
<td>ap-northeast-1</td>
<td>arc-zonal-shift.ap-northeast-1.amazonaws.com</td>
<td>HTTPS</td>
</tr>
<tr>
<td>Canada (Central)</td>
<td>ca-central-1</td>
<td>arc-zonal-shift.ca-central-1.amazonaws.com</td>
<td>HTTPS</td>
</tr>
<tr>
<td>Europe (Frankfurt)</td>
<td>eu-central-1</td>
<td>arc-zonal-shift.eu-central-1.amazonaws.com</td>
<td>HTTPS</td>
</tr>
<tr>
<td>Europe (Ireland)</td>
<td>eu-west-1</td>
<td>arc-zonal-shift.eu-west-1.amazonaws.com</td>
<td>HTTPS</td>
</tr>
<tr>
<td>Europe (London)</td>
<td>eu-west-2</td>
<td>arc-zonal-shift.eu-west-2.amazonaws.com</td>
<td>HTTPS</td>
</tr>
<tr>
<td>Europe (Milan)</td>
<td>eu-south-1</td>
<td>arc-zonal-shift.eu-south-1.amazonaws.com</td>
<td>HTTPS</td>
</tr>
<tr>
<td>Europe (Paris)</td>
<td>eu-west-3</td>
<td>arc-zonal-shift.eu-west-3.amazonaws.com</td>
<td>HTTPS</td>
</tr>
<tr>
<td>Europe (Spain)</td>
<td>eu-south-2</td>
<td>arc-zonal-shift.eu-south-2.amazonaws.com</td>
<td>HTTPS</td>
</tr>
<tr>
<td>Europe (Stockholm)</td>
<td>eu-north-1</td>
<td>arc-zonal-shift.eu-north-1.amazonaws.com</td>
<td>HTTPS</td>
</tr>
<tr>
<td>Europe (Zurich)</td>
<td>eu-central-2</td>
<td>arc-zonal-shift.eu-central-2.amazonaws.com</td>
<td>HTTPS</td>
</tr>
<tr>
<td>Israel (Tel Aviv)</td>
<td>il-central-1</td>
<td>arc-zonal-shift.il-central-1.amazonaws.com</td>
<td>HTTPS</td>
</tr>
<tr>
<td>Middle East (Bahrain)</td>
<td>me-south-1</td>
<td>arc-zonal-shift.me-south-1.amazonaws.com</td>
<td>HTTPS</td>
</tr>
<tr>
<td>Middle East (UAE)</td>
<td>me-central-1</td>
<td>arc-zonal-shift.me-central-1.amazonaws.com</td>
<td>HTTPS</td>
</tr>
<tr>
<td>South America (São Paulo)</td>
<td>sa-east-1</td>
<td>arc-zonal-shift.sa-east-1.amazonaws.com</td>
<td>HTTPS</td>
</tr>
</tbody>
</table>
How Amazon Route 53 Application Recovery Controller works

Amazon Route 53 Application Recovery Controller helps you to prepare for and quickly mitigate impairments for applications on AWS.

- **A readiness check** continually audits AWS resource capacity, configuration, AWS quotas, and routing policies, for an application and provides information that you can use to help successfully recover from application failure. Readiness checks help to ensure that your recovery environment is scaled and configured to fail over to when needed.

- **Routing controls** enable you to rebalance traffic across application replicas during failures, to ensure that your application is available. You can also pair routing controls with safety rules that you create to help avoid unintended consequences. For example, you might want to prevent inadvertantly turning off all the routing controls for an application, which would stop all traffic flow, resulting in a fail-open scenario.

- **A zonal shift** temporarily moves traffic for a resource away from an Availability Zone (AZ), to enable you to quickly and reliably recover from issues for multi-AZ applications. Currently supported resources are Network Load Balancers and Application Load Balancers with cross-zone load balancing turned off.

Learn more about how Route 53 ARC works in the following sections.

- the section called “Monitoring your application replica with readiness checks”
- the section called “Rerouting traffic for recovery with routing control”
- the section called “Moving traffic away from an Availability Zone with zonal shift”
- the section called “Data and control planes for Route 53 ARC”

Monitoring your application replica with readiness checks

Route 53 ARC audits your application replicas by using **readiness checks** to ensure that each one has the same configuration setup and the same runtime state.

To be prepared for recovery, for example, you must maintain sufficient spare capacity at all times to absorb failover traffic from another Availability Zone or Region. Route 53 ARC continually (once a minute) inspects your application to ensure that your provisioned capacity matches across all Availability Zones or Regions. The capacity that Route 53 ARC inspects includes, for example, Amazon EC2 instance counts, Aurora read and write capacity units, and Amazon EBS volume size. If you scale up the capacity in your primary replica for resource values but forget to also increase the corresponding values in your standby replica, Route 53 ARC detects the mismatch so that you can increase the values in the standby.

**Important**

Readiness checks are most useful for verifying, on an ongoing basis, that application replica configurations and runtime states are aligned. Readiness checks shouldn’t be used to indicate whether your production replica is healthy, nor should you rely on readiness checks as a primary trigger for failover during a disaster event.

In an active-standby configuration, you should make decisions about whether to fail away from or to a cell based on your monitoring and health check systems, and consider readiness checks as a complementary service to those systems. Route 53 ARC readiness checks are not highly available, so you should not depend on the checks being accessible during an outage. In addition, the resources that are checked might also not be available during a disaster event.
You can monitor the readiness status for your application's resources in specific cells (AWS Regions or Availability Zones) or for your overall application. You can be notified when a readiness check status changes, for example, to Not ready, by creating rules in EventBridge. For more information, see *Using Route 53 ARC with Amazon EventBridge* (p. 104). You can also view readiness status in the AWS Management Console, or by using API operations, such as get-recovery-readiness. For more information, see *Recovery readiness (readiness check) actions* (p. 47).

**Rerouting traffic for recovery with routing control**

A Route 53 ARC routing control is an on/off switch that changes the state of a Route 53 ARC health check, which can then be associated with a DNS record that redirects traffic, for example, from a primary to a standby deployment replica.

If there's an application failure or latency issue, you can update routing control states to shift traffic from your primary replica to, for example, a standby replica. By using the highly reliable Route 53 ARC data plane API operations to make routing control queries and routing control state updates, you can rely on Route 53 ARC for failover during disaster recovery scenarios. For more information, see *Getting and updating routing control states using the Route 53 ARC API (recommended)* (p. 88).

Route 53 ARC maintains routing control states in a cluster, which is a set of five redundant Regional endpoints. Route 53 ARC propagates routing control state changes across the cluster, which is located in an Amazon EC2 fleet, to get a quorum across five AWS Regions. After propagation, when you query Route 53 ARC for a routing control state, using the API and the highly-reliable data plane, it returns the consensus view.

You can interact with any one of the five cluster endpoints to update the state of a routing control from, for example, Off to On. Then Route 53 ARC propagates the update across the five Regions of the cluster.

Data consistency across all five cluster endpoints is achieved within 5 seconds on average, and after no more than 15 seconds maximum.

Route 53 ARC offers extreme reliability with its data plane for you to manually fail over your application across cells. Route 53 ARC ensures that at least three out of the five cluster endpoints are always accessible to you to perform routing control state changes. Note that each Route 53 ARC cluster is single-tenant, to ensure that you're not affected by "noisy neighbors" that might slow down your access patterns.

When you make changes to routing control states, you rely on the following three criteria, which are highly unlikely to fail:

- At least three of your five endpoints are available and take part in the quorum.
- You have working IAM credentials and can authenticate against a working Regional cluster endpoint.
- The Route 53 data plane is healthy (this data plane is designed to meet a 100% availability SLA).

**Resilience in Route 53 ARC**

Here's an example of incorporating routing controls into your failover strategy, to improve the resilience and availability of your applications in AWS.

You can support highly available AWS applications on AWS by running multiple (typically three) redundant replicas across Regions. Then you can use Amazon Route 53 routing control to route traffic to the appropriate replica.

For example, you can set up one application replica to be active and serve application traffic, while another is a standby replica. When your active replica has failures, you can reroute user traffic there to restore availability to your application. Readiness checks can help you make sure that a standby replica matches your production replica on an ongoing basis. However, you should decide whether to fail away
from or to a replica based on information from your monitoring and health check systems, and consider
readiness checks as a complementary service to those systems.

If you want to enable faster recoveries, another option that you can choose for your architecture is an
active-active implementation. With this approach, all of your replicas are active at the same time. This
means that you can recover from failures by moving users away from your impaired application replica
by just rerouting traffic to another active replica.

**Moving traffic away from an Availability Zone with zonal shift**

With zonal shift, you can move traffic for a load balancing resource away from an Availability Zone (AZ),
so that you can continue operating your application normally in the other AZs in an AWS Region. At this
time, you can start a zonal shift for Network Load Balancers and Application Load Balancers with cross-
zone load balancing turned off.

When you deploy and run AWS applications on load balancers in multiple (typically three) AZs in a
Region, you can quickly recover an application in an impaired AZ by starting a zonal shift. Shifting your
application traffic to other AZs reduces the duration and severity of impact caused by power outages, or
hardware or software failures in the impaired AZ.

When you start a zonal shift for an AZ, Route 53 ARC sets Amazon Route 53 health checks to unhealthy
for the corresponding IP addresses for the load balancer resource, so that traffic for the resource is no
longer directed to the AZ. When the zonal shift expires or you cancel it, Route 53 ARC sets the Route 53
health checks to healthy again and the original zonal IP addresses are restored.

A zonal shift must have an expiry (expiration date), when it will end and traffic will return to the AZ. You
can initially set a zonal shift to expire in a maximum of three days (72 hours). However, you can update a
zonal shift to set a new expiration at any time (which still, however, has a maximum of three days). You
can also cancel a zonal shift, before it expires, if you're ready to restore traffic to the AZ earlier.

In a few specific scenarios, zonal shift does not shift traffic from the AZ. For example, if the load balancer
target groups in the AZs don't have any instances, or if all of the instances are unhealthy, then the load
balancer is in a fail open state. If you start a zonal shift for a load balancer in this scenario, the zonal
shift does not change which AZs the load balancer uses because the load balancer is already in a fail
open state. This is expected behavior. Zonal shift cannot force one AZ to be unhealthy and shift traffic to
the other AZs in a Region if all AZs are failing open (unhealthy). A second scenario is if you start a zonal
shift for an Application Load Balancer that is an endpoint for an accelerator in AWS Global Accelerator.
Zonal shift isn’t supported for Application Load Balancers that are endpoints of accelerators in Global
Accelerator.

To learn more about starting a zonal shift, see [Zonal shift in Amazon Route 53 Application Recovery
Controller (p. 52)](#).

**Data and control planes for Route 53 ARC**

As you plan for failover and disaster recovery, it’s important to consider how resilient your failover
mechanisms are and make sure that the mechanisms that you depend on are highly available, so that
you can use them when you need them in a disaster scenario. Typically you should use data plane
functions for your mechanisms when you can, for the greatest reliability and fault tolerance. With that in
mind, it’s important to understand how the functionality of a service is divided between control planes
and data planes, and when you can rely on an expectation of extreme reliability with a service's data
plane.

Route 53 ARC includes two sets of functionality, readiness checks and routing control for recovery.
As with most AWS services, the Route 53 ARC functionality is supported by control planes and data
planes. While both types are built to be reliable, a control plane is optimized for data consistency, while
a data plane is optimized for availability. A data plane is designed for resilience so that it can maintain
availability even during disruptive events, when a control plane might become unavailable. Because of this, we recommend that you use data plane operations when availability is important, for example, when you need to reroute traffic to a standby replica during an outage.

In general, a control plane enables you to do basic management functions, such as create, update, and delete resources in the service. A data plane provides a service’s core functionality.

For Route 53 ARC, the control planes and data planes are divided as follows:

- For zonal shifts, supported resources are automatically registered with Route 53 ARC. When a resource is registered, it becomes a managed resource for zonal shifts in Route 53 ARC. Route 53 ARC has a data plane in each AWS Region that provides API operations to get, list, create, and update zonal shifts for managed resources. The zonal shift data plane is highly available.

- For readiness checks, there is a single API, the Recovery Readiness API, for both the control plane and data plane. Readiness checks and readiness resources are only in the US West (Oregon) Region (us-west-2). The readiness checks control plane and data plane are not highly available.

- For routing control, the control plane API is the Recovery Control Configuration API, supported in the US West (Oregon) Region (us-west-2). You use these API operations or the AWS Management Console to create or delete clusters, control panels, and routing controls, to help prepare for a disaster recovery event when you might need to reroute traffic for your application. The routing control configuration control plane is not highly available.

- The routing control data plane in Route 53 ARC is a dedicated cluster across five geographically-isolated AWS Regions. Each customer creates one or more clusters using the routing control control plane. The cluster hosts control panels and routing controls. Then you use the Routing Control (Recovery Cluster) API to get, list, and update routing control states when you want to reroute traffic for your application. The routing control data plane is highly available.

To learn more about recovery readiness and preparing for failover with Route 53 ARC, see Best practices for Amazon Route 53 Application Recovery Controller (p. 44).

For more information about data planes, control planes, and how AWS builds services to meet high availability targets, see the Static stability using Availability Zones paper in the Amazon Builders' Library.

Comparing zonal shift and routing control in Amazon Route 53 Application Recovery Controller

Zonal shift and routing control in Amazon Route 53 Application Recovery Controller can both achieve rapid recovery and help maintain resilience for AWS applications. Both options are highly available and help support recovery in scenarios when you have increased latency or reduced availability. Both also enable you to recover applications quickly by moving traffic, limiting the impact and time lost from issues.

Routing control is primarily focused on AWS applications that are in multiple AWS Regions, while zonal shift only supports AWS applications with load balancers in multi-Availability Zone (multi-AZ) deployments. There are other differences as well.

The information in this section describes some of the key features of each option and how they compare to each other. These descriptions can help you better understand how each might apply to your organization’s disaster recovery needs.
Use cases

<table>
<thead>
<tr>
<th>Routing control</th>
<th>Zonal shift</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reroutes traffic from one AWS Region to another (primarily)</td>
<td>Moves traffic away from an Availability Zone</td>
</tr>
<tr>
<td>Can also be used to reroute across Availability Zones</td>
<td>Traffic goes to other Availability Zones in the Region, not to a specific target</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Requires setup</th>
<th>Available without setup</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requires configuration and setup</td>
<td>Enabled automatically by supported services</td>
</tr>
<tr>
<td>(currently Network Load Balancer and Application Load Balancer)</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Fee-based</th>
<th>Included with services</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requires separate charges for routing control</td>
<td>Moving traffic away from AZs is included for supported load balancers</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Does not expire</th>
<th>Temporary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Traffic can be rerouted to a replica indefinitely</td>
<td>All zonal shifts must be set to expire</td>
</tr>
</tbody>
</table>

To learn more about each of these features, see the following chapters:

- Zonal shift in Amazon Route 53 Application Recovery Controller (p. 52)
- Routing control in Amazon Route 53 Application Recovery Controller (p. 79)

Amazon Route 53 Application Recovery Controller use cases

During a failure, you can use routing control and zonal shift to quickly move ensure traffic to restore availability for your application.

The routing control feature in Amazon Route 53 Application Recovery Controller is designed for enterprises that have applications with extremely high availability requirements, such as a less than five minute recovery time objective (RTO) or a greater than 99.99% availability requirement. Typical applications include national payment authentication systems, real-time payment processing, or stock trading workloads that can have a broad financial impact if they go down. These applications might be required to protect against even partial failures, such as a millisecond increase in latency or a 5% error rate.

An enterprise use case for Route 53 ARC zonal shift is to manage multi-Availability Zone recovery, to protect against common application failures, such as a bad deployment in a single Availability Zone. Another enterprise use case, for routing control, is cross-Region recovery, where an organization wants to be able to recover from a large-scale event, such as a natural disaster, and oversee the recovery centrally.

To summarize, Route 53 ARC provides the following benefits:

- With no initial setup, you can use zonal shifts to mitigate partial application failures by quickly move a load balancer’s traffic away from an Availability Zone, to reliably recover from an issue, temporarily. This gives you time to investigate, while your application continues to run in the other Availability Zones.
- If you’ve set up routing control, you can respond to partial failure states by using Route 53 ARC in ways that automated recovery systems might not be able to. For example, after you’ve set up routing
Tagging in Amazon Route 53 Application Recovery Controller

Tags are words or phrases (meta data) that you use to identify and organize your AWS resources. You can add multiple tags to each resource, and each tag includes a key and a value that you define. For example, the key might be environment and the value might be production. You can search and filter your resources based on the tags you add.

You can tag the following resources in Route 53 ARC:

- Recovery groups
- Cells
- Resource sets
- Readiness checks
- Clusters
- Control panels
- Safety rules in routing control

Tagging in Route 53 ARC is available only through the API, for example, by using the AWS CLI. The following are examples of tagging in Route 53 ARC by using the AWS CLI.

Create resources with tags

```bash
aws route53-recovery-readiness --region us-west-2 create-cell --cell-name pdx_cell --tags Region=PDX,Stage=Prod
aws route53-recovery-readiness --region us-west-2 create-recovery-group --recovery-group-name pdx_recovery_group --tags Region=PDX,Stage=Prod
aws route53-recovery-readiness --region us-west-2 create-readiness-check --readiness-check-name dynamodb_readiness_check --resource-set-name dynamodb_resource_set --tags Stage=Prod
aws route53-recovery-control-config --region us-west-2 create-cluster --cluster-name example1-cluster --tags Region=PDX,Stage=Prod
```
aws route53-recovery-control-config --region us-west-2 create-control-panel --control-panel-name example1-control-panel --cluster-arn arn:aws:route53-recovery-control::111122223333:cluster/5678abcd-abcd-5678-abcd-5678abcdefgh --tags Region=PDX,Stage=Prod

**Tag and untag existing resources**


For more information, see [TagResource](#) in the [Recovery Readiness API Reference Guide for Amazon Route 53 Application Recovery Controller](#) and [TagResource](#) in the [Recovery Control Configuration API Reference Guide for Amazon Route 53 Application Recovery Controller](#).

---

**Pricing in Amazon Route 53 Application Recovery Controller**

With Amazon Route 53 Application Recovery Controller, you only pay for what you configure to use in the service. The following is a summary of how pricing works for Route 53 ARC:

- **For zonal shift**: You can use a zonal shift to recover your application from an issue in an Availability Zone. There is no additional charge for using zonal shift.
- **For readiness check**: You pay an hourly cost per readiness check that you configure.
- **For a cluster**: You pay an hourly cost per cluster that you create. Each cluster can host multiple routing controls, which you use to trigger application failovers.

For detailed pricing information and examples, see [Amazon Route 53 Application Recovery Controller Pricing](#) and scroll down to Amazon Route 53.
Getting started with multi-Region recovery in Amazon Route 53 Application Recovery Controller

To use Amazon Route 53 Application Recovery Controller with AWS applications that are in multiple AWS Regions, there are guidelines to follow to set up your applications for recovery readiness. Then you can create readiness checks for your application, and set up routing controls to reroute traffic for failover. You can also review the recommendations Route 53 ARC provides to about your application's architecture that can improve resiliency.

Note
No setup is required to use Route 53 ARC zonal shift to reliably recover applications from Availability Zone impairments. To move traffic away from an Availability Zone for load balancer resources that have been registered with Route 53 ARC, start a zonal shift in the Route 53 ARC console or in the Elastic Load Balancing Console, or by using the AWS Command Line Interface or AWS SDK with zonal shift API actions. For more information, see Zonal shift in Amazon Route 53 Application Recovery Controller (p. 52).

To use Route 53 ARC routing control to recover from application failures, we recommend that you set up at least two (typically, three) application replicas, or cells. Each cell represents an AWS Region or Availability Zone. After you’ve set up your application resources in cells that align with Availability Zones within a Region, there are some additional steps, listed here, that you must take to make sure that your application follows a recovery-oriented design. After you’ve implemented these steps, you can use Route 53 ARC for cross-Availability Zone failover.

Tip
To help simplify set up, we provide AWS CloudFormation and HashiCorp Terraform templates that create an application with redundant replicas that fail independently of one another. Learn more and download the templates in Recovery readiness with a new application (p. 20).

To prepare to use Route 53 ARC, set up a resilient recovery process for your application. The following steps are an overview of how you can prepare your application environment to use Route 53 ARC:

1. Deploy independent copies of your application stack (networking and compute layer) as standby replicas so that you can fail over traffic across the stacks. You should not have any cross-cell dependencies in the application code where a failure of one cell would impact others. To fail over between Availability Zones or AWS Regions, the boundaries of your cells must align with the isolated infrastructure constructs in AWS (Availability Zones or AWS Regions). However, if you don’t set up your boundaries to align with Availability Zones or AWS Regions, Route 53 ARC still supports fail over across cells.

2. Replicate all required stateful data across the cells. You can use AWS database services to help replicate your data. For example, for high availability, you can add read replicas for Aurora instances across Availability Zones. During failover, you can then promote a replica to be the primary database instance.

3. Configure each cell to expose a DNS domain name. The domain name must represent the top-level AWS resource in the cell, and act as the front door for servicing client requests to that cell. A top-level resource might be, for example, an Elastic Load Balancing load balancer or an API from API Gateway. Regardless of the resource that you add the name to, the domain name must only direct requests to infrastructure within the cell.
4. To help you determine the best structure for your application, Route 53 ARC can run an analysis and provide architecture recommendations for you. In the AWS Management Console, provide the application domain name and other information about the application. Route 53 ARC will supply architecture suggestions to help you modify your application to use a well-architected structure that enables fast and complete failover recovery. For more information, see Getting architecture recommendations in Route 53 ARC (p. 76).

5. So that traffic failover that uses Route 53 ARC doesn’t create data consistency issues, design data reconciliation logic in your failover Region for Regional failovers of stateful applications that require strict consistency.

The following sections include more detailed information about getting started with Route 53 ARC, depending on whether you have an existing application or if you’re setting up a new application.

- Recovery readiness with an existing application (p. 19)
- Recovery readiness with a new application (p. 20)
- Routing control for traffic failover (p. 21)

For more information about working with Route 53 ARC, see the following:

- To learn about Route 53 ARC features, see Readiness check in Amazon Route 53 Application Recovery Controller (p. 55) and Routing control in Amazon Route 53 Application Recovery Controller (p. 79).
- To see examples of using Route 53 ARC with the AWS CLI, see Examples of using Route 53 ARC API operations with the AWS CLI (p. 23).
- To see information and examples of using the Route 53 ARC API with AWS SDKs, see Using Route 53 ARC with an AWS SDK (p. 21) and Code examples for Application Recovery Controller using AWS SDKs (p. 140).
- To see a list of Route 53 ARC API operations, see Common actions that you can use with Amazon Route 53 Application Recovery Controller (p. 47).

Recovery readiness with an existing application

With Amazon Route 53 Application Recovery Controller, you can understand the recovery readiness of your application and prepare for failover. If you have an existing application, take the following steps before you set up Route 53 ARC for it:

- Identify the application that you want to set up with recovery readiness.
- Review the definitions of the components in Route 53 ARC. For more information, see Readiness check components (p. 3).
- Review the information in Recovery readiness with a new application (p. 20).
- Set up the required user (or users), roles, and policies for Route 53 ARC. For more information, see Security in Amazon Route 53 Application Recovery Controller (p. 109).

To set up the structure in Route 53 ARC that enables recovery readiness, you can use the Route 53 ARC API – for example, by using the AWS CLI – or the AWS Management Console. You can also use AWS CloudFormation or HashiCorp Terraform templates to quickly get started with Route 53 ARC.

Using one of these options, you model replicas, or failure-containment units, for your application. Within each replica, you define the resources that your application uses, such as Amazon EC2 Auto Scaling groups and load balancers. You can then understand the recovery readiness of your application as a whole, or as individual replicas within your application. You can view readiness status by using API
Recovery readiness with a new application

If you're designing a new application, structure it to be recovery-oriented from the start so that you can be sure to get the most from the recovery features in Amazon Route 53 Application Recovery Controller.

A recovery-oriented application consists of multiple redundant replicas, or failure-containment units, that fail independently of one another. You can set up failure-containment silos by deploying replicas that align with AWS Availability Zone boundaries, which is easier to do if you're starting with a new application than if you need to rearchitect an existing one.

The following sections include an example that illustrates how you can design a recovery-oriented application with siloed replicas in AWS Availability Zones. The example uses AWS CloudFormation templates to simplify the process, as well as downloadable AWS CloudFormation and HashiCorp Terraform templates with a sample app so that you can quickly explore setting up and using Route 53 ARC yourself.

Topics

- How to create an example application (p. 20)
- Download AWS CloudFormation or HashiCorp Terraform templates (p. 21)

How to create an example application

As an example, let's look at an application that routes traffic to a service that runs on Amazon Elastic Container Service (Amazon ECS), is fronted by a Network Load Balancer, and interacts with an Amazon Aurora database. You can launch this application with an AWS CloudFormation template and provision it as one stack.

To make sure that you deploy siloed replicas that are each scoped to an Availability Zone, do the following: make sure that your application architecture uses a Network Load Balancer that is local to one replica that routes to an Amazon ECS cluster, which is also local to the replica. Then, connect these replicas by using an Amazon Route 53 weighted routing policy DNS record. Next, define separate stacks for each replica in a single AWS CloudFormation template by using parameters in the template. (You can learn more about using nested AWS CloudFormation structures by reading Working with nested stacks in the AWS CloudFormation User Guide.)

You can build the application in AWS CloudFormation by following these steps:

1. Create a parent template that defines your AWS managed services in each AWS Region, but not in each Availability Zone. You might include, for example, Regional Aurora tables or Amazon S3 buckets that are in addition to the replicas that you create within each Availability Zone. You'll need to export these resources.

2. In another template, define the AWS resources that are scoped to a replica, such as Network Load Balancers that are scoped to an Availability Zone. Ensure that these resources use template parameters for resource configuration properties that are different in each replica.
3. Create each replica by using the replica template, and pass in the parameters or import values from your parent template.

By using infrastructure-as-code features that support provisioning infrastructure based on dynamic parameters, you can reuse definitions in your AWS CloudFormation template. You can see this illustrated in the example downloadable AWS CloudFormation templates that we provide in the next section. Using parameters lets you define an application that aligns with the Region-focused design patterns in Route 53 ARC, so your application is more resilient by using about the same number of definitions in your template.

Download AWS CloudFormation or HashiCorp Terraform templates

To help you get started using Route 53 ARC, we provide AWS CloudFormation and HashiCorp Terraform templates, together with a sample application and step-by-step instructions, that you can download and deploy locally.

After you deploy the sample app, you can use the templates to create Route 53 ARC components, and then explore using routing controls to manage traffic flow to the app. You can adapt the templates and process for your own scenario and applications.

• **AWS CloudFormation**: To get started with a sample application and AWS CloudFormation templates, see the README instructions here on this Amazon S3 bucket. You can learn more about using AWS CloudFormation templates by reading AWS CloudFormation concepts in the AWS CloudFormation User Guide.

• **HashiCorp Terraform**: To get started with a sample application and Terraform templates, see the README instructions here on this Amazon S3 bucket. You can learn more about using Terraform templates by reading the HashiCorp documentation.

Routing control for traffic failover

The routing control feature of Amazon Route 53 Application Recovery Controller triggers traffic failovers between redundant application copies, or replicas, that are running in separate AWS Regions or Availability Zones. To trigger failovers, you associate Route 53 ARC routing controls with the top-level domain names of your replicas. Then, add a routing control health check so you can control traffic routing across replicas of your application. You can update routing control states in the AWS Management Console, but we recommend that you use Route 53 ARC actions, using the API or AWS CLI, to update routing control states.

For example, if you want to fail over between Availability Zones, from us-west-1a to us-west-1b, you can use the update-routing-control-state API action to set the state of us-east-1a to Off and us-east-1b to On.

For more information about setting up and using routing control in Route 53 ARC, see Routing control in Amazon Route 53 Application Recovery Controller (p. 79).

Using Route 53 ARC with an AWS SDK

AWS software development kits (SDKs) are available for many popular programming languages. Each SDK provides an API, code examples, and documentation that make it easier for developers to build applications in their preferred language.
<table>
<thead>
<tr>
<th>SDK documentation</th>
<th>Code examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>AWS SDK for C++</td>
<td>AWS SDK for C++ code examples</td>
</tr>
<tr>
<td>AWS SDK for Go</td>
<td>AWS SDK for Go code examples</td>
</tr>
<tr>
<td>AWS SDK for Java</td>
<td>AWS SDK for Java code examples</td>
</tr>
<tr>
<td>AWS SDK for JavaScript</td>
<td>AWS SDK for JavaScript code examples</td>
</tr>
<tr>
<td>AWS SDK for Kotlin</td>
<td>AWS SDK for Kotlin code examples</td>
</tr>
<tr>
<td>AWS SDK for .NET</td>
<td>AWS SDK for .NET code examples</td>
</tr>
<tr>
<td>AWS SDK for PHP</td>
<td>AWS SDK for PHP code examples</td>
</tr>
<tr>
<td>AWS SDK for Python (Boto3)</td>
<td>AWS SDK for Python (Boto3) code examples</td>
</tr>
<tr>
<td>AWS SDK for Ruby</td>
<td>AWS SDK for Ruby code examples</td>
</tr>
<tr>
<td>AWS SDK for Rust</td>
<td>AWS SDK for Rust code examples</td>
</tr>
<tr>
<td>AWS SDK for SAP ABAP</td>
<td>AWS SDK for SAP ABAP code examples</td>
</tr>
<tr>
<td>AWS SDK for Swift</td>
<td>AWS SDK for Swift code examples</td>
</tr>
</tbody>
</table>

For examples specific to Route 53 ARC, see Code examples for Application Recovery Controller using AWS SDKs (p. 140).

**Example availability**

Can’t find what you need? Request a code example by using the Provide feedback link at the bottom of this page.
Examples of using Route 53 ARC API operations with the AWS CLI

This section walks through simple application examples, using the AWS Command Line Interface to work with Amazon Route 53 Application Recovery Controller features using API operations. The examples are intended to help you develop a basic understanding of how to work with Route 53 ARC using the CLI.

Topics

- Get started with readiness check by using the AWS CLI (p. 23)
- Get started with routing control by using the AWS CLI (p. 30)
- List and update routing controls and states with the AWS CLI (p. 39)
- Get started with zonal shift by using the AWS CLI (p. 41)

Get started with readiness check by using the AWS CLI

Readiness check in Amazon Route 53 Application Recovery Controller allows you to check the resources in your applications for their readiness to fail over.

Let’s look at a simple case where you have an application named Simple-Service that currently runs in the US East (N. Virginia) Region (us-east-1). You also have a standby copy of the application in the US West (Oregon) Region (us-west-2). In this example, we'll configure readiness checks to compare these two versions of the application. This lets us ensure that the standby, US West (Oregon) Region, is ready to receive traffic, if it needs to in a failover scenario.

For more information about using the AWS CLI, see the AWS CLI Command Reference. For more information about readiness API actions in Route 53, see Recovery readiness (readiness check) actions (p. 47) or the Recovery Readiness API Reference Guide for Amazon Route 53 Application Recovery Controller.

Cells represent fault boundaries (like Availability Zones or Regions) and are collected into recovery groups. A recovery group represents an application that you want to check failover readiness for. For more information about the components of readiness checks, see Readiness check components (p. 3).

Note

Route 53 ARC is a global service that supports endpoints in multiple AWS Regions but you must specify the US West (Oregon) Region (that is, specify the parameter --region us-west-2) in most Route 53 ARC CLI commands. For example, to create resources such as recovery groups, readiness checks, or clusters.

When you create a cluster, Route 53 ARC provides you with a set of Regional endpoints. To get or update routing control states, you must specify the Regional endpoint (the AWS Region and the endpoint URL) in your CLI command.

For our application example, we'll start by creating one cell for each Region where we have resources. Then we'll create a recovery group, and then complete the setup for a readiness check.

1. Create cells

1a. Create a us-east-1 cell.
2. Create a recovery group

Recovery groups are the top-level resource for recovery readiness in Route 53 ARC. A recovery group represents an application as a whole. In this step, we'll create a recovery group to model an overall application, and then add the two cells that we created.

2a. Create a recovery group.

```bash
aws route53-recovery-readiness --region us-west-2 create-recovery-group \
  --recovery-group-name simple-service-recovery-group
```
3. Create a resource set

Let's assume our Simple-Service application is indeed very simple and only uses DynamoDB tables. It has a DynamoDB table in us-east-1 and another one in us-west-2. A resource set also contains a readiness scope, which identifies the cell that each resource is contained in.

3a. Create a resource set that reflects our Simple-Service application's resources.

```bash
aws route53-recovery-readiness --region us-west-2 create-resource-set \
  --resource-set-name ImportantInformationTables \
  --resource-set-type AWS::DynamoDB::Table \
  --resources 
```

```json
{
}
```
3. Create a resource set

```json
"ResourceSetName": "ImportantInformationTables",
"Resources": [
    {
        "ReadinessScopes": [
            "arn:aws:route53-recovery-readiness::111122223333:cell/west-cell"
        ],
        "ResourceArn": "arn:aws:dynamodb:us-west-2:111122223333:table/TableInUsWest2"
    },
    {
        "ReadinessScopes": [
            "arn:aws:route53-recovery-readiness::111122223333:cell/east-cell"
        ],
        "ResourceArn": "arn:aws:dynamodb:us-west-2:111122223333:table/TableInUsEast1"
    }
],
"Tags": {}
]
```

3b. (Optional) You can verify what's included in the resource set by calling the `list-resource-sets` API. This lists all the resource sets for an AWS account. Here you can see that we have just the one resource set that we created above.

```
aws route53-recovery-readiness --region us-west-2 list-resource-sets
```

```json
{
    "ResourceSets": [
        {
            "ResourceSetName": "ImportantInformationTables",
            "Resources": [
                {
                    "ReadinessScopes": [
                        "arn:aws:route53-recovery-readiness::111122223333:cell/west-cell"
                    ],
                    "ResourceArn": "arn:aws:dynamodb:us-west-2:111122223333:table/TableInUsWest2"
                },
                {
                    "ReadinessScopes": [
                        "arn:aws:route53-recovery-readiness::111122223333:cell/east-cell"
                    ],
                    "ResourceArn": "arn:aws:dynamodb:us-west-2:111122223333:table/TableInUsEast1"
                }
            ],
            "Tags": {}
        }
    ]
}
```
Now we've created the cells, recovery group, and resource set to model the Simple-Service application in Route 53 ARC. Next, we'll set up readiness checks to monitor the readiness of the resources for fail over.

### 4. Create a readiness check

A readiness check applies a set of rules to each resource in the resource set that is attached to the check. Rules are specific to each resource type. That is, there are different rules for AWS::DynamoDB::Table, AWS::EC2::Instance, and so on. Rules check a variety of dimensions for a resource, including configuration, capacity (where available and applicable), limits (where available and applicable), and routing configurations.

**Note**

To see the rules that are applied to a resource in a readiness check, you can use the `get-readiness-check-resource-status` API, as described in step 5. To see a list of all the readiness rules in Route 53 ARC, use `list-rules` or see [Readiness rules descriptions in Route 53 ARC](p. 63). Route 53 ARC has a specific set of rules that it runs for each resource type; they're not customizable at this time.

#### 4a. Create a readiness check for the resource set, ImportantInformationTables.

```bash
aws route53-recovery-readiness --region us-west-2 create-readiness-check --readiness-check-name ImportantInformationTableCheck --resource-set-name ImportantInformationTables
```

```json
{
    "ReadinessCheckArn": "arn:aws:route53-recovery-readiness::111122223333:readiness-check/ImportantInformationTableCheck",
    "ReadinessCheckName": "ImportantInformationTableCheck",
    "ResourceSet": "ImportantInformationTables",
    "Tags": {}
}
```

#### 4b. (Optional) To verify that the readiness check was created successfully, run the `list-readiness-checks` API. This API shows all the readiness checks in an account.

```bash
aws route53-recovery-readiness --region us-west-2 list-readiness-checks
```

```json
{
    "ReadinessChecks": [
        {
            "ReadinessCheckArn": "arn:aws:route53-recovery-readiness::111122223333:readiness-check/ImportantInformationTableCheck",
            "ReadinessCheckName": "ImportantInformationTableCheck",
        }
    ]
}
```
5. Monitor readiness checks

Now that we've modeled the application and added a readiness check, we're ready to monitor resources. You can model the readiness of your application at four levels: the readiness check level (a group of resources), the individual resource level, the cell level (all the resources in an Availability Zone or Region), and the recovery group level (the application as a whole). Commands for getting each of these types of readiness statuses are provided below.

5a. See the status of your readiness check.

```
aws route53-recovery-readiness --region us-west-2 get-readiness-check-status
  --readiness-check-name ImportantInformationTableCheck
```

```
{
  "Readiness": "READY",
  "Resources": [
    {
      "LastCheckedTimestamp": "2021-01-07T00:53:39Z",
      "Readiness": "READY",
      "ResourceArn": "arn:aws:dynamodb:us-west-2:111122223333:table/TableInUsWest2"
    },
    {
      "LastCheckedTimestamp": "2021-01-07T00:53:39Z",
      "Readiness": "READY",
      "ResourceArn": "arn:aws:dynamodb:us-west-2:111122223333:table/TableInUsEast2"
    }
  ]
}
```

5b. See the detailed readiness status of a single resource in a readiness check, including the status of each rule that is checked.

```
aws route53-recovery-readiness --region us-west-2 get-readiness-check-resource-status
  --readiness-check-name ImportantInformationTableCheck
  --resource-identifier "arn:aws:dynamodb:us-west-2:111122223333:table/TableInUsWest2"
```

```
["Readiness": "READY",
 "Rules": [
   {
     "LastCheckedTimestamp": "2021-01-07T00:55:41Z",
     "Messages": [],
     "Readiness": "READY",
     "RuleId": "DynamoTableStatus"
   },
   {
     "LastCheckedTimestamp": "2021-01-07T00:55:41Z",
     "Messages": [],
     "Readiness": "READY",
     "RuleId": "DynamoCapacity"
   },
   {
     "LastCheckedTimestamp": "2021-01-07T00:55:41Z",
     "Messages": [],
     "Readiness": "READY",
     "RuleId": "DynamoCapacity"
   }
]
```

5c. See the overall readiness for a cell.

```
aws route53-recovery-readiness --region us-west-2 get-cell-readiness-summary \
   --cell-name west-cell
```

```
{
   "Readiness": "READY",
   "ReadinessChecks": [
      {
         "Readiness": "READY",
         "ReadinessCheckName": "ImportantTableCheck"
      }
   ]
}
```

5d. Finally, see the top-level readiness of your application, at the recovery group level.
Get started with routing control by using the AWS CLI

With routing control in Amazon Route 53 Application Recovery Controller, you can trigger traffic failovers between redundant application copies, or replicas, that are running in separate AWS Regions or Availability Zones.

You can organize routing controls into groups called control panels that are provisioned on a cluster. A Route 53 ARC cluster is a Regional set of endpoints that is globally deployed. Cluster endpoints provide a highly available API that you can use to set and retrieve routing control states. For more information about the components of the routing control feature, see Routing control components (p. 6).

Our first step is to create a cluster. A Route 53 ARC cluster is a set of five Regional endpoints that are deployed in a global distribution. The infrastructure in Route 53 ARC supports these endpoints to work in coordination to provide a guarantee of high availability and sequential consistency of failover operations.

Note
Route 53 ARC is a global service that supports endpoints in multiple AWS Regions but you must specify the US West (Oregon) Region (that is, specify the parameter `--region us-west-2`) in most Route 53 ARC CLI commands. For example, to create resources such as recovery groups, readiness checks, or clusters.

When you create a cluster, Route 53 ARC provides you with a set of Regional endpoints. To get or update routing control states, you must specify the Regional endpoint (the AWS Region and the endpoint URL) in your CLI command.

For more information about using the AWS CLI, see the AWS CLI Command Reference. For more information about recovery control configuration API actions in Route 53 ARC, see Recovery control configuration actions (p. 49) or the Recovery Control Configuration API Reference Guide for Amazon Route 53 Application Recovery Controller.

1. Create a cluster

1a. Create a cluster.

aws route53-recovery-control-config --region us-west-2 create-cluster --cluster-name NewCluster

[  "Cluster": {  
}  ]
When you first create a Route 53 ARC resource, it has a status of PENDING while the cluster is created. You can check in on its progress by calling describe-cluster.

1b. Describe a cluster.

```
aws route53-recovery-control-config --region us-west-2 \
  describe-cluster --cluster-arn arn:aws:route53-recovery-control::111122223333:cluster/5678abcd-abcd-5678-abcd-5678abcdefgh
```

```
{
  "Cluster": {
    "ClusterEndpoints": [
      {
        "Endpoint": "https://host-aaaaa.us-east-1.example.com", "Region": "us-east-1"
      },
      {
        "Endpoint": "https://host-bbbbb.ap-southeast-2.example.com", "Region": "ap-southeast-2"
      },
      {
        "Endpoint": "https://host-cccccc.eu-west-1.example.com", "Region": "eu-west-1"
      },
      {
        "Endpoint": "https://host-dddddd.us-west-2.example.com", "Region": "us-west-2"
      },
      {
        "Endpoint": "https://host-eeeeee.ap-northeast-1.example.com", "Region": "ap-northeast-1"
      }
    ],
    "Name": "NewCluster",
    "Status": "DEPLOYED"
  }
}
```

When the status is DEPLOYED, Route 53 ARC has successfully created the cluster with the set of endpoints for you to interact with. You can list all of your clusters by calling list-clusters.

1c. List your clusters.

```
aws route53-recovery-control-config --region us-west-2 list-clusters
```

```
{
  "Clusters": [
    {
      "ClusterEndpoints": [
        {
          "Endpoint": "https://host-aaaaa.us-east-1.example.com", "Region": "us-east-1"
        },
        {
          "Endpoint": "https://host-bbbbb.ap-southeast-2.example.com", "Region": "ap-southeast-2"
        },
        {
          "Endpoint": "https://host-cccccc.eu-west-1.example.com", "Region": "eu-west-1"
        },
        {
          "Endpoint": "https://host-dddddd.us-west-2.example.com", "Region": "us-west-2"
        },
        {
          "Endpoint": "https://host-eeeeee.ap-northeast-1.example.com", "Region": "ap-northeast-1"
        }
      ],
      "Name": "AnotherCluster",
      "Status": "DEPLOYED"
    }
  ]
```
2. Create a control panel

A control panel is a logical grouping for organizing your Route 53 ARC routing controls. When you create a cluster, Route 53 ARC automatically provides a control panel for you called DefaultControlPanel. You can use this control panel right away.

A control panel can only exist in one cluster. If you want to move a control panel to another cluster, you must delete it and then create it in the second cluster. You can see all of the control panels in your account by calling `list-control-panels`. To see just the control panels in a specific cluster, add the `--cluster-arn` field.

2a. List control panels.

```bash
aws route53-recovery-control-config --region us-west-2 \
    list-control-panels --cluster-arn arn:aws:route53-recovery-control::111122223333:cluster/5678abcd-abcd-5678-abcd-5678abcdefgh
```

```json
{
    "ControlPanels": [
        {
            "ControlPanelArn": "arn:aws:route53-recovery-control::111122223333:controlpanel/1234567dddd1234567d1234567",
            "DefaultControlPanel": true,
            "Name": "DefaultControlPanel",
            "RoutingControlCount": 0,
            "Status": "DEPLOYED"
        }
    ]
}
```

Optionally, create your own control panel by calling `create-control-panel`.

2b. Create a control panel.

```bash
aws route53-recovery-control-config --region us-west-2 create-control-panel \
    --control-panel-name NewControlPanel2 \
```

```json
{
    "ClusterEndpoints": [
        {
            "Endpoint": "https://host-ffffff.us-east-1.example.com",
            "Region": "us-east-1"
        },
        {
            "Endpoint": "https://host-gggggg.ap-southeast-2.example.com",
            "Region": "ap-southeast-2"
        },
        {
            "Endpoint": "https://host-hhhhhh.eu-west-1.example.com",
            "Region": "eu-west-1"
        },
        {
            "Endpoint": "https://host-iiiiii.us-west-2.example.com",
            "Region": "us-west-2"
        },
        {
            "Endpoint": "https://host-jjjjjj.ap-northeast-1.example.com",
            "Region": "ap-northeast-1"
        }
    ],
    "Name": "NewCluster",
    "Status": "DEPLOYED"
}
```
3. Create a routing control

Now that you've set up the cluster and looked at control panels, you can begin creating routing controls. When you create a routing control, you must at least specify the Amazon Resource Name (ARN) of the cluster that you want the routing control to be in. You can also specify the ARN of a control panel for the routing control. You'll also need to specify the cluster where the control panel is located.

If you don't specify a control panel, your routing control is added to the automatically created control panel, DefaultControlPanel.

Create a routing control by calling `create-routing-control`.

```bash
```

When you first create a Route 53 ARC resource, it has a status of PENDING while it's being created. You can check on progress by calling `describe-control-panel`.

```bash
aws route53-recovery-control-config --region us-west-2 describe-control-panel --control-panel-arn arn:aws:route53-recovery-control::111122223333:controlpanel/0123456bbbbbbb0123456bbbbbb0123456
```

When you first create a Route 53 ARC resource, it has a status of PENDING while it's being created. You can check on progress by calling `describe-control-panel`.

2c. Describe a control panel.

```bash
aws route53-recovery-control-config --region us-west-2 describe-control-panel --control-panel-arn arn:aws:route53-recovery-control::111122223333:controlpanel/0123456bbbbbbb0123456bbbbbb0123456
```

2c. Describe a control panel.

```bash
aws route53-recovery-control-config --region us-west-2 describe-control-panel --control-panel-arn arn:aws:route53-recovery-control::111122223333:controlpanel/0123456bbbbbbb0123456bbbbbb0123456
```

3. Create a routing control
Routing controls follow the same creation pattern as other Route 53 ARC resources, so you can track their progress by calling a describe operation.

3b. Describe routing control.

```

```json
{
   "RoutingControl": {
      "ControlPanelArn": "arn:aws:route53-recovery-control::111122223333:controlpanel/0123456bbbbbbb0123456bbbbbb0123456",
      "Name": "NewRc1",
      "RoutingControlArn": "arn:aws:route53-recovery-control::111122223333:controlpanel/0123456bbbbbbb0123456bbbbbb0123456/routingcontrol/abcdefg1234567",
      "Status": "DEPLOYED"
   }
}
```

You can list the routing controls in a control panel by calling `list-routing-controls`. The control panel ARN is required.

3c. List routing controls.

```

```json
{
   "RoutingControls": [
      {
         "ControlPanelArn": "arn:aws:route53-recovery-control::111122223333:controlpanel/0123456bbbbbbb0123456bbbbbb0123456",
         "Name": "Rc1",
         "RoutingControlArn": "arn:aws:route53-recovery-control::111122223333:controlpanel/0123456bbbbbbb0123456bbbbbb0123456/routingcontrol/abcdefg1234567",
         "Status": "DEPLOYED"
      },
      {
         "ControlPanelArn": "arn:aws:route53-recovery-control::111122223333:controlpanel/0123456bbbbbbb0123456bbbbbb0123456/routingcontrol/hijklmnop987654321",
         "Name": "Rc2",
         "RoutingControlArn": "arn:aws:route53-recovery-control::111122223333:controlpanel/0123456bbbbbbb0123456bbbbbb0123456/routingcontrol/hijklmnop987654321",
         "Status": "DEPLOYED"
      }
   ]
}
```
In the following example, where we work with routing control states, we assume that you have the two routing controls listed in this section (Rc1 and Rc2). In this example, each routing control represents an Availability Zone that your application is deployed in.

## 4. Create safety rules

When you work with several routing controls at the same time, you might decide that you want some safeguards in place when you enable and disable them, to avoid unintentional consequences, like turning both routing controls off and stopping all traffic flow. To create these safeguards, you create Route 53 ARC safety rules.

There are two types of safety rules: assertion rules and gating rules. To learn more about safety rules, see [Creating safety rules in Route 53 ARC](p. 89).

The following call provides an example of creating an assertion rule that makes sure that at least one of two routing controls is set to On at any given time. To create the rule, you run `create-safety-rule` with the assertion-rule parameter.

For detailed information about the assertion rule API operation, see [AssertionRule](p. 35) in the Routing Control API Reference Guide for Amazon Route 53 Application Recovery Controller.

### 4a. Create an assertion rule.

```
aws route53-recovery-control-config --region us-west-2 create-safety-rule \
```

```
{
    "Rule": {
        "ASSERTION": {
            "AssertedControls": [ 
                "arn:aws:route53-recovery-control::888888888888:controlpanel/zzz123yyyy456xxx789zzz123yyyy456xxx/routingcontrol/def123def123def", 
                "arn:aws:route53-recovery-control::888888888888:controlpanel/zzz123yyyy456xxx789zzz123yyyy456xxx/routingcontrol/ghi456ghi456ghi"], 
            "ControlPanelArn": "arn:aws:route53-recovery-control::888888888888:controlpanel/zzz123yyyy456xxx789zzz123yyyy456xxx", 
            "Name": "TestAssertionRule", 
            "RuleConfig": { 
                "Inverted": false, 
                "Threshold": 1, 
                "Type": "ATLEAST" 
            }, 
            "Status": "PENDING", 
            "WaitPeriodMs": 5000
        }
    }
}
The following call provides an example of creating a gating rule that provides an overall "on/off" or "gating" switch for a set of target routing controls in a control panel. This lets you disallow updating the target routing controls so that, for example, automation can't make unauthorized updates. In this example, the gating switch is a routing control specified by the GatingControls parameter and the two routing controls that are controlled or "gated" are specified by the TargetControls parameter.

**Note**
Before you create the gating rule, you must create the gating routing control, which does not include DNS failover records, and the target routing controls, which you do configure with DNS failover records.

To create the rule, you run `create-safety-rule` with the `gating-rule` parameter.

For detailed information about the assertion rule API operation, see [GatingRule](https://docs.aws.amazon.com/efs/latest/APIReference/API_CreateSafeguard.html) in the Routing Control API Reference Guide for Amazon Route 53 Application Recovery Controller.

**4b. Create a gating rule.**

```bash
aws route53-recovery-control-config --region us-west-2 create-safety-rule --gating-rule "{
  "Name": "TestGatingRule",
  "ControlPanelArn": "arn:aws:route53-recovery-control::888888888888:controlpanel/zzz123yyy456xxx789zzz123yyy456xxx",
  "WaitPeriodMs": 5000,
  "GatingControls": ["arn:aws:route53-recovery-control::888888888888:controlpanel/zzz123yyy456xxx789zzz123yyy456xxx/routingcontrol/def123def123def"],
  "RuleConfig": {"Threshold": 0, "Type": "OR", "Inverted": false}}"
```

```json
{
"Rule": {
  "GATING": {
    "Arn": "arn:aws:route53-recovery-control::888888888888:controlpanel/zzz123yyy456xxx789zzz123yyy456xxx/safetyrule/44444444444",
    "GatingControls": ["arn:aws:route53-recovery-control::888888888888:controlpanel/zzz123yyy456xxx789zzz123yyy456xxx/routingcontrol/def123def123def"],
    "ControlPanelArn": "arn:aws:route53-recovery-control::888888888888:controlpanel/zzz123yyy456xxx789zzz123yyy456xxx",
    "Name": "TestGatingRule",
    "RuleConfig": {
      "Inverted": false,
      "Threshold": 0,
      "Type": "OR"
    },
    "Status": "PENDING",
    "WaitPeriodMs": 5000
  }
}
```
As with other Route 53 ARC resources, you can describe, list, or delete safety rules after they propagate to the data plane.

After you set up one or more safety rules, you can continue to interact with the cluster, to set, or retrieve state for routing controls. If a set-routing-control-state operation breaks a rule that you created, you'll receive an exception similar to the following:

```
Cannot modify control state for [0123456bbbbbb0123456bbbbbb01234560123
abcdefg1234567] due to failed rule evaluation
0123456bbbbbb0123456bbbbbb0123456333333444444
```

The first identifier is the control panel ARN concatenated with the routing control ARN. The second identifier is the control panel ARN concatenated with the safety rule ARN.

## 5. Create health checks

To use routing controls to fail over traffic, you create health checks in Amazon Route 53 and associate the health checks with your DNS records. As an example, let's say you have two cells, one that you've configured as the primary cell for your application, and the other that you've configured as the secondary, to fail over to.

To set up health checks for failover, you can do the following, for example:

1. Use the Route 53 ARC CLI to create a routing control for each cell.
2. Use the Route 53 CLI to create a Route 53 ARC health check in Route 53 for each routing control.
3. Use the Route 53 CLI to create two failover DNS records in Route 53, and associate a health check with each one.

### 5a. Create a routing control for each cell.

```bash
aws route53-recovery-control-config --region us-west-2 create-routing-control
--routing-control-name RoutingControlCell1

aws route53-recovery-control-config --region us-west-2 create-routing-control
--routing-control-name RoutingControlCell2
```

### 5b. Create a health check for each routing control.

**Note**

You create Route 53 ARC health checks by using the Amazon Route 53 CLI.

```bash
aws route53 create-health-check --caller-reference RoutingControlCell1
--health-check-config
 Type=RECOVERY_CONTROL,RoutingControlArn=arn:aws:route53-recovery-control::111122223333:controlpanel/0123456bbbbbb0123456bbbbbb0123456/routingcontrol/abcdefg1234567

aws route53 create-health-check --caller-reference RoutingControlCell2
--health-check-config
 Type=RECOVERY_CONTROL,RoutingControlArn=arn:aws:route53-recovery-control::111122223333:controlpanel/0123456bbbbbb0123456bbbbbb0123456/routingcontrol/abcdefg1234567
```

```json
{
```
5. Create health checks

```
"HealthCheck": {
    "Id": "xxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
    "CallerReference": "RoutingControlCell1",
    "HealthCheckConfig": {
        "Type": "RECOVERY_CONTROL",
        "Inverted": false,
        "Disabled": false,
        "RoutingControlArn": "arn:aws:route53-recovery-control::111122223333:controlpanel/0123456bbbbbbb0123456bbbbbb0123456/routingcontrol/abcdefg1234567",
    },
    "HealthCheckVersion": 1
},
```

```
aws route53 create-health-check --caller-reference RoutingControlCell2 --health-check-config Type=RECOVERY_CONTROL,RoutingControlArn=arn:aws:route53-recovery-control::111122223333:controlpanel/0123456bbbbbbb0123456bbbbbb0123456/routingcontrol/abcdefg1234567
```

```
[
    "HealthCheck": {
        "Id": "xxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
        "CallerReference": "RoutingControlCell2",
        "HealthCheckConfig": {
            "Type": "RECOVERY_CONTROL",
            "Inverted": false,
            "Disabled": false,
            "RoutingControlArn": "arn:aws:route53-recovery-control::111122223333:controlpanel/0123456bbbbbbb0123456bbbbbb0123456/routingcontrol/abcdefg1234567",
        },
        "HealthCheckVersion": 1
    }
]
```

5c. Create two failover DNS records, and associate a health check with each one.

You create failover DNS records in Route 53 using the Route 53 CLI. To create the records, follow the directions in the Amazon Route 53 AWS CLI Command Reference for the change-resource-record-sets command. In the records, specify the DNS value for each cell together with the corresponding HealthCheckID value that Route 53 created for the health check (see 6b).

For the primary cell:

```
{
    "Name": "myapp.yourdomain.com",
    "Type": "CNAME",
    "SetIdentifier": "primary",
    "Failover": "PRIMARY",
    "TTL": 0,
    "ResourceRecords": [
        {
            "Value": "cell1.yourdomain.com"
        }
    ],
    "HealthCheckId": "xxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"
}
```
For the secondary cell:

```
{
  "Name": "myapp.yourdomain.com",
  "Type": "CNAME",
  "SetIdentifier": "secondary",
  "Failover": "SECONDARY",
  "TTL": 0,
  "ResourceRecords": [
    {
      "Value": "cell2.yourdomain.com"
    }
  ],
  "HealthCheckId": "yyyyyy-yyyy-yyyy-yyyy-yyyy-yyyy-yyyyyyyyyyyy"
}
```

Now, to fail over from your primary cell to your secondary cell, you can follow the CLI example in step 4b to update the state of RoutingControlCell1 to OFF and RoutingControlCell2 to ON.

**List and update routing controls and states with the AWS CLI**

After you create your Amazon Route 53 Application Recovery Controller resources—cluster, routing controls, and control panels—you can interact with the cluster to list and update routing control states.

For each cluster that you create, Route 53 ARC provides you with a set of cluster endpoints, one in each of five AWS Regions. You must specify one of these Regional endpoints (the AWS Region and the endpoint URL) when you make calls to the cluster to retrieve or set routing control states to On or Off. In addition to the Regional endpoint, you must also specify the --region of the Regional endpoint when you use the AWS CLI with Route 53 ARC, as shown in the examples in this section.

You can use any of the Regional cluster endpoints. We recommend that your systems be prepared to retry with each of the available endpoints. For code samples that illustrate trying cluster endpoints in sequence, see [Actions for Application Recovery Controller using AWS SDKs](p. 140).

For more information about using the AWS CLI, see the AWS CLI Command Reference. For more information about routing control API actions in Route 53 ARC, see [Recovery cluster (routing control) data plane actions](p. 50) or the [Routing Control API Reference Guide](p. 50) for Amazon Route 53 Application Recovery Controller.

**Important**

Although you can update a routing control state on the Amazon Route 53 console, we recommend that you [update routing control states](p. 88) by using the AWS CLI or an AWS SDK. Route 53 ARC offers extreme reliability with the Route 53 ARC routing control data plane for rerouting traffic and failing over across cells. For more recommendations about using Route 53 ARC for failover, see [Best practices for Amazon Route 53 Application Recovery Controller](p. 44).

When you create a routing control, the state is set to Off. This means that traffic is not routed to the target cell for that routing control. You can verify the state of the routing control by running the command `get-routing-control-state`.

To determine the Region and the endpoint to specify, run the `describe-clusters` command to view the `ClusterEndpoints`. Each `ClusterEndpoint` includes a Region and corresponding endpoint that you can use to get or update routing control states. `DescribeCluster` is a recovery control configuration API operation. We recommend that you keep a local copy of your Route 53 ARC Regional cluster endpoints, in bookmarks or hardcoded in automation code that you use to retry your endpoints.
You can view your routing controls and routing control states using the highly reliable Route 53 ARC data plane endpoints.

1. List routing controls for a specific control panel. If you don't specify a control panel, list-routing-controls returns all the routing controls in the cluster.

```bash
aws route53-recovery-cluster list-routing-controls --control-panel-arn
  arn:aws:route53-recovery-control::1111222333:controlpanel/0123456bbbbbbb0123456bbbbbb0123456
  --region us-west-2
  --endpoint-url https://host-dddddd.us-west-2.example.com/v1
```

```json
{
  "RoutingControls": [
    {
      "ControlPanelArn": "arn:aws:route53-recovery-control::1111222333:controlpanel/0123456bbbbbbb0123456bbbbbb0123456",
      "ControlPanelName": "ExampleControlPanel",
      "RoutingControlArn": "arn:aws:route53-recovery-control::1111222333:controlpanel/0123456bbbbbbb0123456bbbbbb0123456/routingcontrol/abcdefg1234567",
      "RoutingControlName": "RCOne",
      "RoutingControlState": "On"
    },
    {
      "ControlPanelArn": "arn:aws:route53-recovery-control::023759465626:controlpanel/0123456bbbbbbb0123456bbbbbb0123456/routingcontrol/zzzzxxxxyyyy123456",
      "ControlPanelName": "ExampleControlPanel",
      "RoutingControlArn": "arn:aws:route53-recovery-control::023759465626:controlpanel/0123456bbbbbbb0123456bbbbbb0123456/routingcontrol/zzzzxxxxyyyy123456",
      "RoutingControlName": "RCTwo",
      "RoutingControlState": "Off"
    }
  ]
}
```

2. Get a routing control state.

```bash
aws route53-recovery-cluster get-routing-control-state --routing-control-arn
  arn:aws:route53-recovery-control::1111222333:controlpanel/0123456bbbbbbb0123456bbbbbb0123456/routingcontrol/abcdefg1234567
  --region us-west-2
  --endpoint-url https://host-dddddd.us-west-2.example.com/v1
```

```json
{"RoutingControlArn": "arn:aws:route53-recovery-control::1111222333:controlpanel/0123456bbbbbbb0123456bbbbbb0123456/routingcontrol/abcdefg1234567",
  "RoutingControlName": "RCOne",
  "RoutingControlState": "On"
}
```

To route traffic to the target endpoint controlled by the routing control, you update the routing control state to On. Update the routing control state by running the command update-routing-control-state. (When the request is successful, the response is empty.)

2a. Update a routing control state.

```bash
aws route53-recovery-cluster update-routing-control-state
  --routing-control-arn
```

```bash
aws route53-recovery-cluster update-routing-control-state
  --routing-control-arn
```
Zonal shift with the CLI

You can update several routing controls at the same time with one API call: update-routing-control-states. (When the request is successful, the response is empty.)

2b. Update several routing control states at once (batch updates).

```
aws route53-recovery-cluster update-routing-control-states \
  --update-routing-control-state-entries \
  '[{"RoutingControlArn": "arn:aws:route53-recovery-control::111122223333:controlpanel/0123456bbbbbbb0123456bbbbbb0123456/routingcontrol/abcdefg1234567", \n    "RoutingControlState": "Off"}, \n    "RoutingControlArn": "arn:aws:route53-recovery-control::111122223333:controlpanel/0123456bbbbbbb0123456bbbbbb0123456/routingcontrol/hijklmnop987654321", \n    "RoutingControlState": "On"}]' \
  --region us-west-2 \
  --endpoint-url https://host-dddddd.us-west-2.example.com/v1
```

Get started with zonal shift by using the AWS CLI

Zonal shift in Amazon Route 53 Application Recovery Controller enables you to temporarily move traffic for your load balancers away from an Availability Zone so your application can continue to operate normally with other Availability Zones in an AWS Region. Zonal shift currently supports Network Load Balancers and Application Load Balancers with cross-zone load balancing turned off.

Let's look at an example of starting a zonal shift using the AWS Command Line Interface. You can also use the AWS CLI to update a zonal shift, for example, to set a new expiration. All zonal shifts are temporary and must be set initially to expire within three days. However, you can update a zonal shift later to set a new expiration.

For more information about using the AWS CLI, see the AWS CLI Command Reference. For more information about zonal shift API actions, see Zonal shift actions (p. 51) or the Zonal Shift API Reference Guide for Amazon Route 53 Application Recovery Controller.

Start zonal shift

You can start a zonal shift with the CLI by using the start-zonal-shift command.

```
aws arc-zonal-shift start-zonal-shift \
  --resource-identifier="arn:aws:testservic::111122223333:ExampleALB123456890" \
  --away-from="usw2-az1" \
  --expires-in="5m" \
  --comment="Shifting traffic away from USW2-AZ1"
```
Get managed resource

You can get information about a managed resource with the CLI by using the `get-managed-resource` command.

```
aws arc-zonal-shift get-managed-resource \
   --resource-identifier="arn:aws:testservice::111122223333:ExampleALB123456890"
```

```
{
   "arn": "arn:aws:testservice::111122223333:ExampleALB123456890",
   "name": "TestResource",
   "appliedWeights": {
      "usw2-az1": 1.0,
      "usw2-az2": 1.0,
      "usw2-az3": 1.0
   },
   "zonalShifts": []
}
```

List managed resources

You can list the managed resources in your account with the CLI by using the `list-managed-resources` command.

```
aws arc-zonal-shift list-managed-resources
```

```
{
   "items": [
      {
         "arn": "arn:aws:testservice::111122223333:ExampleALB123456890",
         "name": "TestResource",
         "availabilityZones": [
            "usw2-az1",
            "usw2-az2",
            "usw2-az3"
         ]
      }
   ]
}
```

List zonal shifts

You can list the zonal shifts in your account with the CLI by using the `list-zonal-shifts` command.

```
aws arc-zonal-shift list-zonal-shifts
```
Update zonal shift

You can update a zonal shift with the CLI by using the `update-zonal-shift` command.

```
aws arc-zonal-shift update-zonal-shift \
    --zonal-shift-id="arn:aws:testservice::111122223333:ExampleALB123456890" \
    --expires-in="1h" \
    --comment="Still shifting traffic away from USW2-AZ1"
```

```
{
    "zonalShiftId": "2222222-3333-444-1111",
    "resourceIdentifier": "arn:aws:testservice::111122223333:ExampleALB123456890",
    "awayFrom": "usw2-az1",
    "expiryTime": 2022-11-15T10:42+00:00,
    "startTime": 2022-11-15T09:35:42+00:00,
    "status": "ACTIVE",
    "comment": "Still shifting traffic away from USW2-AZ1"
}
```

Cancel zonal shift

You can cancel a zonal shift with the CLI by using the `cancel-zonal-shift` command.

```
aws arc-zonal-shift cancel-zonal-shift \
    --zonal-shift-id="arn:aws:testservice::111122223333:ExampleALB123456890"
```

```
{
    "zonalShiftId": "2222222-3333-444-1111",
    "resourceIdentifier": "arn:aws:testservice::111122223333:ExampleALB123456890",
    "awayFrom": "usw2-az1",
    "expiryTime": 2022-11-15T10:42+00:00,
    "startTime": 2022-11-15T09:35:42+00:00,
    "status": "CANCELED",
    "comment": "Shifting traffic away from USW2-AZ1"
}
```
Best practices for Amazon Route 53 Application Recovery Controller

To minimize disruption and help provide for operational continuity, follow best practices to plan for and execute disaster recovery with Amazon Route 53 Application Recovery Controller. Review the guidelines in this chapter to learn more.

Topics

- Best practices for recovery in Route 53 ARC (p. 44)
- Best practices for zonal shifts in Route 53 ARC (p. 44)
- Best practices for readiness checks and routing controls in Route 53 ARC (p. 45)

Best practices for recovery in Route 53 ARC

We recommend the following best practices for recovery and failover preparedness in Amazon Route 53 Application Recovery Controller.

Keep purpose-built, long-lived AWS credentials secure and always accessible

In a disaster recovery (DR) scenario, keep system dependencies to a minimum by using a simple approach to accessing AWS and performing recovery tasks. Create IAM long-lived credentials specifically for DR tasks, and keep the credentials securely in an on-premises physical safe or a virtual vault, to access when needed. With IAM, you can centrally manage security credentials, such as access keys, and permissions for access to AWS resources. For non-DR tasks, we recommend that you continue to use federated access, using AWS services such as AWS Single Sign-On.

To perform failover tasks in Route 53 ARC with the recovery cluster data plane API, you can attach a Route 53 ARC IAM policy to your user. To learn more, see Identity-based policy examples for Amazon Route 53 Application Recovery Controller (p. 121).

Choose lower TTL values for DNS records involved in failover

For DNS records that you might need to change as part of your failover mechanism, especially records that are health checked, using lower TTL values is appropriate. Setting a TTL of 60 or 120 seconds is a common choice for this scenario.

The DNS TTL (time to live) setting tells DNS resolvers how long to cache a record before requesting a new one. When you choose a TTL, you make a trade-off between latency and reliability, and responsiveness to change. With a shorter TTL on a record, DNS resolvers notice updates to the record more quickly because the TTL specifies that they must query more frequently.

For more information, see Choosing TTL values for DNS records in Best practices for Amazon Route 53 DNS.

Best practices for zonal shifts in Route 53 ARC

We recommend the following best practices for using zonal shifts for multi-AZ recovery in Route 53 ARC. Zonal shifts typically remove capacity from a live application, so it’s important to be careful when you use them in production.
Capacity planning and pre-scaling

Ensure that you have planned for, and either pre-scaled or can auto-scale, sufficient capacity to accommodate the extra load imposed on Availability Zones when you start a zonal shift. With a recovery-oriented architecture, a typical recommendation is to pre-scale compute capacity to include enough headroom to serve your peak traffic when one of your (typically) three replicas is offline.

When you start a zonal shift for a single load balancer resource, for example, the capacity of one Availability Zone is temporarily removed from behind the load balancer. Depending on the zonal shifts that you start and how your load balancers are configured, you must make sure that you've carefully planned for managing the increased load on the remaining Availability Zones.

Test starting zonal shifts, in advance

Regularly test moving traffic away from Availability Zones for your application by starting zonal shifts. Plan for and execute starting zonal shifts, preferably in both test and production environments, as part of regular failover testing for recovering your applications in the event of a disaster. Regular testing is a critical part of ensuring that you're ready for and have the confidence to mitigate issues when an operational event occurs.

Ensure that all Availability Zones are healthy and taking traffic

Zonal shifts work by marking a resource, that is, an application replica, as unhealthy in an Availability Zone. This means that it's critical to ensure that the targets in the load balancers for your applications are generally healthy and actively taking traffic in the Availability Zones in a Region. We recommend that you have dashboards to track this, including, for example, Elastic Load Balancing metrics for unhealthy targets and bytesProcessed per Availability Zone.

Consider monitoring health of your resources from a second, adjacent Region. Advantages of this approach are that it can be more representative of your end users' experience, and it also reduces the risk of both your application and your monitoring being impacted by the same disaster at the same time ("shared fate").

Use data plane API operations for disaster recovery

For starting a zonal shift when you need to recover an application quickly, with few dependencies, we recommend using the AWS Command Line Interface or API with zonal shift actions, with pre-stored credentials, if possible. You can also start zonal shifts in the AWS Management Console, for ease of use. But when fast, reliable recovery is critical, data plane operations are a better choice. For more information, see Zonal Shift API Reference Guide.

Move traffic with a zonal shift only temporarily

A zonal shift moves traffic away from an Availability Zone on a temporary basis, to mitigate an impairment. You should restore the resource for the application to service as soon as you've taken action to correct a problem. This ensures that your overall application is restored to its original fully redundant, resilient state.

Best practices for readiness checks and routing controls in Route 53 ARC

We recommend the following best practices for recovery readiness and failover preparedness when you set up and use Route 53 ARC with readiness checks and routing control, for example, for Regional failover.

Bookmark or hard code your five Regional cluster endpoints and routing control ARNs

We recommend that you keep a local copy of your Route 53 ARC Regional cluster endpoints, in bookmarks or saved in automation code that you use to retry your endpoints. During a failure event,
you might not be able to access some API operations, including Route 53 ARC API operations that are not hosted on the extremely reliable data plane cluster. You can list the endpoints for your Route 53 ARC clusters by using the `DescribeCluster` API operation.

**Choose one of your endpoints at random to update your routing control states**

We recommend that when you need to fail over, you update (and retrieve) routing control states using a random endpoint from your five Regional cluster endpoints. If that endpoint fails, then retry each of your other Regional endpoints. For information about using code examples with the AWS SDK, including examples for trying cluster endpoints, see [Code examples for Application Recovery Controller using AWS SDKs](p. 140).

**Use the extremely reliable data plane API to list and update routing control states, not the console**

Using the Route 53 ARC data plane API, view your routing controls and states with the `ListRoutingControls` operation and update routing control states to redirect traffic for failover with the `UpdateRoutingControlState` operation. You can use the AWS CLI (as in these examples) (p. 39) or code that you write using one of the AWS SDKs. Route 53 ARC offers extreme reliability with the API in the data plane to fail over traffic. We recommend using the API instead of changing routing control states in the AWS Management Console.

Connect to one of your Regional cluster endpoints for Route 53 ARC to use the data plane API. If the endpoint is unavailable, try connecting to another cluster endpoint.

If a safety rule blocks a routing control state update, you can bypass it to make the update and fail over traffic. For more information, see [Overriding safety rules to reroute traffic](p. 91).

**Test failover with Route 53 ARC**

Test failover regularly with Route 53 ARC routing control, to fail over from your primary application stack to a secondary application stack. It's important to make sure that the Route 53 ARC structures that you've added are aligned with the correct resources in your stack, and that everything works as you expect it to. You should test this after you set up Route 53 ARC for your environment, and continue to test periodically, so that your failover environment is prepared, before you experience a failure situation in which you need your secondary system to be up and running quickly to avoid downtime for your users.

**Add notifications for readiness status changes**

Set a rule in Amazon EventBridge to send a notification whenever a readiness check status changes, for example, from READY to NOT READY. When you receive a notification, you can investigate and address the issue, to make sure that your application and resources are ready for failover when you expect them to be.

You can set EventBridge rules to send notifications for several readiness check status changes, including for your recovery group (for your application), for a cell (such as an AWS Region), or for a readiness check for a resource set.

For more information, see [Using Route 53 ARC with Amazon EventBridge](p. 104).
Common actions that you can use with Amazon Route 53 Application Recovery Controller

This section lists common Amazon Route 53 Application Recovery Controller API actions that you can use, with links to relevant documentation.

For examples of how to use many of these actions with the AWS Command Line Interface, see Examples of using Route 53 ARC API operations with the AWS CLI (p. 23).

Topics
- Recovery readiness (readiness check) actions (p. 47)
- Recovery control configuration actions (p. 49)
- Recovery cluster (routing control) data plane actions (p. 50)
- Zonal shift actions (p. 51)

Recovery readiness (readiness check) actions

The following table lists common Route 53 ARC actions that you can use for recovery readiness (readiness check), with links to relevant documentation.

<table>
<thead>
<tr>
<th>Action</th>
<th>Using the Route 53 ARC console</th>
<th>Using the Route 53 ARC API</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a cell</td>
<td>See Creating, updating, and deleting recovery groups in Route 53 ARC (p. 59)</td>
<td>See CreateCell</td>
</tr>
<tr>
<td>Get a cell</td>
<td>See Creating, updating, and deleting recovery groups in Route 53 ARC (p. 59)</td>
<td>See GetCell</td>
</tr>
<tr>
<td>Delete a cell</td>
<td>See Creating, updating, and deleting recovery groups in Route 53 ARC (p. 59)</td>
<td>See DeleteCell</td>
</tr>
<tr>
<td>Update a cell</td>
<td>N/A</td>
<td>See UpdateCell</td>
</tr>
<tr>
<td>List cells for an account</td>
<td>See Creating, updating, and deleting recovery groups in Route 53 ARC (p. 59)</td>
<td>See ListCells</td>
</tr>
<tr>
<td>Create a recovery group</td>
<td>See Creating, updating, and deleting recovery groups in Route 53 ARC (p. 59)</td>
<td>See CreateRecoveryGroup</td>
</tr>
<tr>
<td>Get a recovery group</td>
<td>See Creating, updating, and deleting recovery groups in Route 53 ARC (p. 59)</td>
<td>See GetRecoveryGroup</td>
</tr>
</tbody>
</table>

47
## Recovery readiness actions

<table>
<thead>
<tr>
<th>Action</th>
<th>Using the Route 53 ARC console</th>
<th>Using the Route 53 ARC API</th>
</tr>
</thead>
<tbody>
<tr>
<td>Update a recovery group</td>
<td>See <a href="https://docs.aws.amazon.com/route53/latest/developerguide/creating-updating-and-deleting-recovery-groups.html">Creating, updating, and deleting recovery groups in Route 53 ARC (p. 59)</a></td>
<td>See <a href="https://docs.aws.amazon.com/route53/latest/APIReference/API_UpdateRecoveryGroup.html">UpdateRecoveryGroup</a></td>
</tr>
<tr>
<td>Create a resource set</td>
<td>See <a href="https://docs.aws.amazon.com/route53/latest/developerguide/creating-readiness-checks.html">Creating and updating readiness checks in Route 53 ARC (p. 60)</a></td>
<td>See <a href="https://docs.aws.amazon.com/route53/latest/APIReference/API_CreateResourceSet.html">CreateResourceSet</a></td>
</tr>
<tr>
<td>Create a readiness check</td>
<td>See <a href="https://docs.aws.amazon.com/route53/latest/developerguide/creating-readiness-checks.html">Creating and updating readiness checks in Route 53 ARC (p. 60)</a></td>
<td>See <a href="https://docs.aws.amazon.com/route53/latest/APIReference/API_CreateReadinessCheck.html">CreateReadinessCheck</a></td>
</tr>
<tr>
<td>Update a readiness check</td>
<td>See <a href="https://docs.aws.amazon.com/route53/latest/developerguide/creating-readiness-checks.html">Creating and updating readiness checks in Route 53 ARC (p. 60)</a></td>
<td>See <a href="https://docs.aws.amazon.com/route53/latest/APIReference/API_UpdateReadinessCheck.html">UpdateReadinessCheck</a></td>
</tr>
<tr>
<td>Delete a readiness check</td>
<td>See <a href="https://docs.aws.amazon.com/route53/latest/developerguide/creating-readiness-checks.html">Creating and updating readiness checks in Route 53 ARC (p. 60)</a></td>
<td>See <a href="https://docs.aws.amazon.com/route53/latest/APIReference/API_DeleteReadinessCheck.html">DeleteReadinessCheck</a></td>
</tr>
<tr>
<td>List readiness checks</td>
<td>See <a href="https://docs.aws.amazon.com/route53/latest/developerguide/creating-readiness-checks.html">Creating and updating readiness checks in Route 53 ARC (p. 60)</a></td>
<td>See <a href="https://docs.aws.amazon.com/route53/latest/APIReference/API_ListReadinessChecks.html">ListReadinessChecks</a></td>
</tr>
</tbody>
</table>
### Recovery control configuration actions

The following table lists common Route 53 ARC actions that you can use for recovery control configuration, with links to relevant documentation.

<table>
<thead>
<tr>
<th>Action</th>
<th>Using the Route 53 ARC console</th>
<th>Using the Route 53 ARC API</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a cluster</td>
<td>See Creating routing control components in Route 53 ARC (p. 80)</td>
<td>See CreateCluster</td>
</tr>
<tr>
<td>Describe a cluster</td>
<td>See Creating routing control components in Route 53 ARC (p. 80)</td>
<td>See DescribeCluster</td>
</tr>
<tr>
<td>Delete a cluster</td>
<td>See Creating routing control components in Route 53 ARC (p. 80)</td>
<td>See DeleteCluster</td>
</tr>
<tr>
<td>List clusters for an account</td>
<td>See Creating routing control components in Route 53 ARC (p. 80)</td>
<td>See ListClusters</td>
</tr>
<tr>
<td>Create a routing control</td>
<td>See Creating routing control components in Route 53 ARC (p. 80)</td>
<td>See CreateRoutingControl</td>
</tr>
<tr>
<td>Describe a routing control</td>
<td>See Creating routing control components in Route 53 ARC (p. 80)</td>
<td>See DescribeRoutingControl</td>
</tr>
<tr>
<td>Update a routing control</td>
<td>See Creating routing control components in Route 53 ARC (p. 80)</td>
<td>See UpdateRoutingControl</td>
</tr>
<tr>
<td>Delete a routing control</td>
<td>See Creating routing control components in Route 53 ARC (p. 80)</td>
<td>See DeleteRoutingControl</td>
</tr>
<tr>
<td>List routing controls</td>
<td>See Creating routing control components in Route 53 ARC (p. 80)</td>
<td>See ListRoutingControls</td>
</tr>
<tr>
<td>Create a control panel</td>
<td>See Creating routing control components in Route 53 ARC (p. 80)</td>
<td>See CreateControlPanel</td>
</tr>
</tbody>
</table>
## Recovery cluster (routing control) data plane actions

The following table lists common Route 53 ARC actions that you can use for managing traffic failover with the routing control (recovery cluster) data plane, with links to relevant documentation.

<table>
<thead>
<tr>
<th>Action</th>
<th>Using the Route 53 ARC console</th>
<th>Using the Route 53 ARC API</th>
</tr>
</thead>
<tbody>
<tr>
<td>Get a routing control state</td>
<td>See [Getting and updating routing control states in the AWS Management Console](p. 88)</td>
<td>See [GetRoutingControlState](p. 88)</td>
</tr>
<tr>
<td>List routing controls</td>
<td>N/A</td>
<td>See [ListRoutingControls](p. 88)</td>
</tr>
</tbody>
</table>
Zonal shift actions

The following table lists common Route 53 ARC actions that you can use using zonal shifts, which move traffic away from an Availability Zone for multi-AZ applications, with links to relevant documentation.

<table>
<thead>
<tr>
<th>Action</th>
<th>Using the Route 53 ARC console</th>
<th>Using the Route 53 ARC API</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Action</th>
<th>Using the Route 53 ARC console</th>
<th>Using the Route 53 ARC API</th>
</tr>
</thead>
<tbody>
<tr>
<td>Update a zonal shift</td>
<td>See <a href="https://docs.aws.amazon.com/route53/latest/developerguide/zonal-shifts.html">Updating or canceling a zonal shift (p. 53)</a></td>
<td>See <a href="https://docs.aws.amazon.com/route53/latest/developerguide/zonal-shifts.html">UpdateZonalShift</a></td>
</tr>
<tr>
<td>List managed resources</td>
<td>See <a href="https://docs.aws.amazon.com/route53/latest/developerguide/zonal-shifts.html">Resources supported for zonal shifts (p. 54)</a></td>
<td>See <a href="https://docs.aws.amazon.com/route53/latest/developerguide/zonal-shifts.html">ListManagedResources</a></td>
</tr>
<tr>
<td>Cancel a zonal shift</td>
<td>See <a href="https://docs.aws.amazon.com/route53/latest/developerguide/zonal-shifts.html">Updating or canceling a zonal shift (p. 53)</a></td>
<td>See <a href="https://docs.aws.amazon.com/route53/latest/developerguide/zonal-shifts.html">CancelZonalShift</a></td>
</tr>
</tbody>
</table>
Zonal shift in Amazon Route 53 Application Recovery Controller

This chapter explains how to use zonal shift in Amazon Route 53 Application Recovery Controller to reliably recover your application from an issue in an Availability Zone. You can start a zonal shift to move traffic for a managed Elastic Load Balancing resource in an AWS Region away from an Availability Zone, for example, because a bad deployment is causing latency issues, or because the Availability Zone is impaired.

In addition to starting a zonal shift in Route 53 ARC, you can also start a zonal shift for a load balancer in the Elastic Load Balancing console. To learn more about starting a zonal shift with Elastic Load Balancing, see Zonal shift in the Elastic Load Balancing User Guide.

All zonal shifts are temporary. You must set an initial expiration when you start a zonal shift, from one hour up to three days (72 hours). But you can update active zonal shifts at any time to set new expirations. The new expiration starts from the time that you set it and has the same constraints.

Topics

- How a zonal shift works (p. 52)
- Starting a zonal shift (p. 53)
- Updating or canceling a zonal shift (p. 53)
- Resources supported for zonal shifts (p. 54)

How a zonal shift works

When you start a zonal shift for a load balancer resource, Amazon Route 53 Application Recovery Controller requests that the resource move traffic away from the Availability Zone that you've specified. This request causes the load balancer health check for the Availability Zone to be set to unhealthy so that it fails its health check. An unhealthy health check, in turn, results in Amazon Route 53 withdrawing the corresponding IP addresses for the resource from DNS, so traffic is redirected from the Availability Zone. New connections are now routed to other Availability Zones in the AWS Region instead.

When you start a zonal shift, the zonal shift is created in Route 53 ARC, but because of the steps in the process, you might not see traffic move out of the Availability Zone immediately. It also can take a short time for existing, in-progress connections in the Availability Zone to complete, depending on client behavior and connection reuse. Typically, however, this takes just a few minutes.

When a zonal shift expires or you cancel it, Route 53 ARC reverses the process, requesting the Route 53 health checks to be set to healthy again, so the original zonal IP addresses are restored and the Availability Zone is included in the load balancer's routing again.

Route 53 ARC uses health checks to move traffic away from Availability Zones, by requesting health checks to be set to unhealthy, and then to healthy again when you cancel a zonal shift or it expires. It's important to note that zonal shift does not, however, include health checks that monitor the underlying health of load balancers or applications.

You must set all zonal shifts to expire when you start them. You can initially set a zonal shift to expire in a maximum of three days (72 hours). However, you can update a zonal shift to set a new expiration at any time. You can also cancel a zonal shift before it expires, if you're ready to restore traffic to the Availability Zone.
In a few specific scenarios, zonal shift does not shift traffic from the AZ. For example, if the load balancer target groups in the AZs don't have any instances, or if all of the instances are unhealthy, then the load balancer is in a fail open state. If you start a zonal shift for a load balancer in this scenario, the zonal shift does not change which AZs the load balancer uses because the load balancer is already in a fail open state. This is expected behavior. Zonal shift cannot force one AZ to be unhealthy and shift traffic to the other AZs in a Region if all AZs are failing open (unhealthy). A second scenario is if you start a zonal shift for an Application Load Balancer that is an endpoint for an accelerator in AWS Global Accelerator. Zonal shift isn't supported for Application Load Balancers that are endpoints of accelerators in Global Accelerator.

For more information about zonal shift support, see Resources supported for zonal shifts (p. 54).

Starting a zonal shift

The steps in this section explain how to start a zonal shift on the Amazon Route 53 Application Recovery Controller console. To work with zonal shift programmatically, see the Zonal Shift API Reference Guide.

To start a zonal shift

2. Under Multi-AZ, choose Zonal shift.
3. On the Zonal shift page, choose Start zonal shift.
4. Select the Availability Zone that you want to move traffic away from.
5. Select a load balancer from the Resources table to move traffic away for.
6. For Set zonal shift expiration, choose or enter an expiration for the zonal shift. A zonal shift can set to be active initially for 1 minute or up to three days (72 hours).
7. All zonal shifts are temporary. You must set an expiration, but you can update active shifts later to set a new expiration period of up to three days.
8. Enter a comment. You can update the zonal shift later to edit the comment, if you like.
9. Select the check box to acknowledge that starting a zonal shift will reduce available capacity for your application by shifting traffic away from the Availability Zone.
10. Choose Start.

Updating or canceling a zonal shift

The steps in this section explain how to update or cancel a zonal shift on the Amazon Route 53 Application Recovery Controller console. To work with zonal shift programmatically, see the Zonal Shift API Reference Guide.

You can update a zonal shift to set a new expiration, or edit or replace the comment for the zonal shift. You can cancel a zonal shift any time before it expires.

To update a zonal shift

2. Under Multi-AZ, choose Zonal shift.
3. Select a zonal shift that you want to update, and then choose Update zonal shift.
4. For Set zonal shift expiration, optionally select or enter an expiration.
5. For **Comment**, optionally edit the existing comment or enter a new comment.
6. Choose **Update**.

**To cancel a zonal shift**

2. Under **Multi-AZ**, choose **Zonal shift**.
3. Select a zonal shift that you want to cancel, and then choose **Cancel zonal shift**.
4. On the confirmation modal dialog, choose **Confirm**.

**Resources supported for zonal shifts**

Amazon Route 53 Application Recovery Controller currently supports zonal shifts for Network Load Balancers and Application Load Balancers. Both public and internal (private) Network Load Balancers and Application Load Balancers are supported. You can start a zonal shift for a load balancer in the Elastic Load Balancing console or in Route 53 ARC.

Review the following conditions for working with zonal shifts and resources in Route 53 ARC:

- **Zonal shift isn't supported with cross-zone load balancing.** For a load balancer to be registered with Route 53 ARC, make sure that you've turned off cross-zone load balancing for the load balancer in ELB.
- **A resource must be active and fully provisioned to shift traffic for it.** Before you start a zonal shift for a resource, check to make sure that it's a managed resource in Route 53 ARC. For example, you can view the list of managed resources in the AWS Management Console, or you can use the `get-managed-resource` operation with the resource's identifier.
- **Zonal shift isn't supported for Application Load Balancers that are endpoints of accelerators in AWS Global Accelerator.**
- **When an Application Load Balancer is the target of a Network Load Balancer, start the zonal shift from the Network Load Balancer.** If you start the zonal shift from the Application Load Balancer, the Network Load Balancer doesn't stop sending traffic to the Application Load Balancer and its targets.
- **The resource for a zonal shift must be a managed resource that has been registered with Route 53 ARC by an AWS service.** Elastic Load Balancing automatically registers with Route 53 ARC Network Load Balancers and Application Load Balancers with cross-zone load balancing turned off.
- **To start a zonal shift with a resource, it must be deployed in the Availability Zone and AWS Region where you start the shift.** Make sure that you start a zonal shift in the same Region that the AZ for the shift is in, and that the resource you're shifting traffic for is in the same AZ and Region as well.
- **Make sure that you have the correct IAM permissions to use zonal shift with a resource.** For more information, see [IAM and permissions for zonal shift](p. 120).
Readiness check in Amazon Route 53 Application Recovery Controller

This chapter explains how to model your application in Amazon Route 53 Application Recovery Controller by creating a recovery group and cells, and then how to add readiness checks and readiness scopes so that Route 53 ARC can audit readiness for your application.

After you create readiness checks, you can monitor the readiness status of your resources. Readiness checks help you to ensure that your standby application replica and its resources match your production replica on an ongoing basis, reflecting the capacity, routing policies, and other configuration details of your production application. If it doesn't, you can add capacity or change a configuration so that the replicas are aligned again.

Important
Readiness checks are most useful for verifying, on an ongoing basis, that application replica configurations and runtime states are aligned. Readiness checks shouldn't be used to indicate whether your production replica is healthy, nor should you rely on readiness checks as a primary trigger for failover during a disaster event.

A readiness check in Route 53 ARC continually (at one-minute intervals) audits for mismatches in AWS provisioned capacity, service quotas, throttle limits, and configuration and version discrepancies for the resources included in the check. Readiness checks can notify you of these differences so that you can make sure that each replica has the same configuration setup and the same runtime state. Although readiness checks ensure that your configured capacities across replicas are consistent, you should not expect them to decide on your behalf what the capacity of your replica should be. For example, you should understand your application requirements so that you size your Auto Scaling groups with enough buffer capacity in each replica to manage if another cell is unavailable.

For quotas, when Route 53 ARC detects a mismatch with a readiness check, it can take steps to align the quotas for the replicas by increasing the lower quota to match the higher quota. When the quotas match, the readiness check status shows READY. (Note that this isn't an immediate update process, and the total time depends on the specific resource type and other factors.)

The first step is setting up readiness checks to create a recovery group (p. 59) that represents your application. Each recovery group includes cells for each individual failure-containment unit or replica of your application. Next, you create resource sets (p. 56) for each resource type in your application, and associate readiness checks with the resource sets. Finally, you associate the resources with readiness scopes, so you can get readiness status about the resources in a recovery group (your application) or individual cells (replicas, which are Regions or Availability Zones (AZs)).

Readiness (that is, READY or NOT READY) is based on the resources that are in the scope of the readiness check and the set of rules for a resource type. There are sets of readiness rules (p. 64) for each resource type, which Route 53 ARC checks use to audit resources for readiness. Whether a resource is READY or not is based on how each readiness rule is defined. All readiness rules evaluate resources, but some compare resources to each other and some look at specific information about each resource in the resource set.

By adding readiness checks, you can monitor readiness status, in one of several ways: with EventBridge, in the AWS Management Console, or by using Route 53 ARC API actions. You can also monitor readiness status of resources in different contexts, including the readiness of cells and the readiness of your
application. Use the cross-account authorization (p. 77) feature in Route 53 ARC to make it easier to set up and monitor distributed resources from a single AWS account.

Topics
- Readiness checks and disaster recovery scenarios (p. 56)
- Readiness checks, resource sets, and readiness scopes (p. 56)
- How readiness rules determine readiness status (p. 57)
- DNS target resource readiness checks: Auditing resiliency readiness (p. 59)
- Creating, updating, and deleting recovery groups in Route 53 ARC (p. 59)
- Creating and updating readiness checks in Route 53 ARC (p. 60)
- Monitoring readiness status in Route 53 ARC (p. 62)
- Readiness rules descriptions in Route 53 ARC (p. 63)
- Resource types and ARN formats in Route 53 ARC (p. 72)
- Getting architecture recommendations in Route 53 ARC (p. 76)
- Create cross-account authorizations in Route 53 ARC (p. 77)

Readiness checks and disaster recovery scenarios

Route 53 ARC readiness checks give you insights into whether your applications and resources are ready for recovery by helping you make sure that your applications are scaled to handle failover traffic. Readiness check statuses should not be used as a signal to indicate that a production replica is healthy. You can, however, use readiness checks as a supplement to your application and infrastructure monitoring or health checker systems to determine whether to fail away from or to a replica.

In an urgent situation or an outage, use a combination of health checks and other information to determine that your standby is scaled up, healthy, and ready for you to fail over production traffic. For example, check to see if canaries that run against your standby cell are meeting your success criteria, in addition to verifying that readiness check statuses for the standby are READY.

Be aware that Route 53 ARC readiness checks are hosted in a single AWS Region, US West (Oregon), and during an outage or disaster, readiness check information could become stale or the checks could become unavailable. For more information, see Data and control planes for Route 53 ARC (p. 13).

Readiness checks, resource sets, and readiness scopes

Readiness checks always audit groups of resources in resource sets. You create resource sets (separately, or while you're creating a readiness check) to group the resources that are in the cells (Availability Zones or AWS Regions) in your Route 53 ARC recovery group, so that you can define readiness checks. A resource set is typically a group of same type of resources (like Network Load Balancers) but can also be DNS target resources, for architectural readiness checks.

You typically create one resource set and readiness check for each type of resource in your application. For an architectural readiness check, you create a top level DNS target resource and a global (recovery group level) resource set for it, and then create cell level DNS target resources, for a separate resource set.

The following diagram shows an example of a recovery group with three cells (Availability Zones), each with a Network Load Balancer (NLB) and Auto Scaling group (ASG).
In this scenario, you would create a resource set and readiness check for the three Network Load Balancers, and a resource set and readiness check for the three Auto Scaling groups. Now you have a readiness check for each set of resources for your recovery group, by resource type.

By creating readiness scopes for resources, you can add readiness check summaries for cells or recovery groups. To specify a readiness scope for a resource, you associate the ARN of the cell or recovery group with each resource in a resource set. You can do this when you’re creating a readiness check for a resource set.

For example, when you add a readiness check for a resource set for the Network Load Balancers for this recovery group, you can add readiness scopes to each NLB at the same time. In this case, you would associate the ARN of AZ 1a to the NLB in AZ 1a, the ARN of AZ 1b to the NLB AZ 1b, and the ARN of AZ 1c to the NLB in AZ 1c. When you create a readiness check for the Auto Scaling groups, you would do the same, assigning readiness scopes to each of them when you create the readiness check for the Auto Scaling group resource set.

It’s optional to associate readiness scopes when you create a readiness check, however, we strongly recommend that you set them. Readiness scopes enable Route 53 ARC to show the correct READY or NOT READY readiness status for recovery group summary readiness checks and cell level summary readiness checks. Unless you set readiness scopes, Route 53 ARC can’t provide these summaries.

Note that when you add an application-level or a global resource, such as a DNS routing policy, you don’t choose a recovery group or cell for the readiness scope. Instead, you choose global resource (no cell).

How readiness rules determine readiness status

Route 53 ARC readiness checks determine readiness status based on the predefined rules for each resource type and the way those rules are defined. Route 53 ARC includes one group of rules for each resource type.
type of resource that it supports. For example, Route 53 ARC has groups of readiness rules for Amazon Aurora clusters, Auto Scaling groups, and so on. Some readiness rules compare resources in a set to each other, and some look at specific information about each resource in the resource set.

You can't add, edit, or remove readiness rules, or groups of rules. However, you can create an Amazon CloudWatch alarm and create a readiness check to monitor the state of the alarm. For example, you can create a custom CloudWatch alarm to monitor Amazon EKS container services, and create a readiness check to audit the readiness status of the alarm.

You can view all the readiness rules for each resource type in the AWS Management Console when you create a resource set, or you can view the readiness rules later by navigating to the details page for a resource set. You can also view readiness rules in the following section: Readiness rules in Route 53 ARC (p. 64).

When a readiness check audits a set of resources with a set of rules, the way each rule is defined determines whether the result will be READY or NOT READY for all the resources or if the result will be different for different resources. In addition, you can view readiness status in multiple ways. For example, you can view the readiness status of a group of resources in a resource set or view a summary of readiness status for a recovery group or a cell (that is, an AWS Region or Availability Zone, depending on how you've set up your recovery group).

The wording in each rule description explains how it evaluates the resources to determine the readiness status when that rule is applied. A rule is defined to inspect each resource or to inspect all resources in a resource set to determine readiness. Specifically, the rules work as follows:

- The rule inspects each resource in the resource set to ensure a condition.
  - If all resources succeed, all resources are set as READY.
  - If one resource fails, that resource is set as NOT READY, and the other cells remain READY.

For example: **MskClusterState**: Inspects each Amazon MSK cluster to ensure that it is in an ACTIVE state.

- The rule inspects all resources in the resource set to ensure a condition.
  - If the condition is ensured, all resources are set as READY.
  - If any fails to meet the condition, all resources are set as NOT READY.

For example: **VpcSubnetCount**: Inspects all VPC subnets to ensure that they have the same number of subnets.

- Non-critical rule: The rule inspects all resources in the resource set to ensure a condition.
  - If any fails, the readiness status is unchanged. A rule with this behavior has a note in its description.

For example: **ElbV2CheckAzCount**: Inspects each Network Load Balancer to ensure that it is attached to only one Availability Zone. Note: This rule does not affect readiness status.

In addition, Route 53 ARC takes an extra step for quotas. If a readiness check detects a mismatch across cells for service quotas (the maximum value for resource creation and operations) for any supported resource, Route 53 ARC automatically raises the quota for the resource with the lower quota. This applies only to quotas (limits). For capacity, you should add additional capacity as required for your application needs.

You can also set up an Amazon EventBridge notification for readiness checks, for example, when any readiness check status changes to NOT READY. Then when a configuration mismatch is detected, EventBridge sends you a notification and you can take corrective action to make sure that your application replicas are aligned and prepared for recovery. For more information, see Using Route 53 ARC with Amazon EventBridge (p. 104).
DNS target resource readiness checks: Auditing resiliency readiness

With DNS target resource readiness checks in Route 53 ARC, you can audit the architectural and resiliency readiness of your application. This type of readiness check continually scans your application's architecture and Amazon Route 53 routing policies to audit for cross-zone and cross-Region dependencies.

A recovery-oriented application has multiple replicas that are siloed into Availability Zones or AWS Regions, so that the replicas can fail independently of one another. If your application needs adjusting to be siloed correctly, Route 53 ARC will suggest changes that you can make, if needed, to update your architecture to help ensure that it's resilient and ready for failover.

Route 53 ARC automatically detects the number and the scope of cells (representing replicas, or failure-containment units) in your application, and whether the cells are siloed by Availability Zone or by Region. Then, Route 53 ARC identifies and provides information to you about the application resources in the cells, to determine if they are correctly siloed to zones or Regions. For example, if you have cells that are scoped to specific zones, readiness checks can monitor if your load balancers and the targets behind them are also siloed to those zones.

With this information, you can determine if there are changes that you need to make to align resources in your cells to the correct zones or Regions.

To get started, you create DNS target resources for your application, and resource sets and readiness checks for them. For more information, see Getting architecture recommendations in Route 53 ARC (p. 76).

Creating, updating, and deleting recovery groups in Route 53 ARC

A recovery group represents your application in Amazon Route 53 Application Recovery Controller. It typically consists of two or more cells that are replicas of each other in terms of resources and functionality, so that you can fail over from one to the other. Each cell includes the Amazon Resource Names (ARNs) for the active resources for one AWS Region or Availability Zone. The resources might be an Elastic Load Balancing load balancer, an Auto Scaling group, or other resources. A corresponding cell representing another zone or Region has standby resources of the same type that are in your active cell – a load balancer, Auto Scaling group, and so on.

A cell represents replicas of your application. Readiness checks in Route 53 ARC help you determine if your application is ready to fail over from one replica to another. However, you should make decisions about whether to fail away from or to a replica based on your monitoring and health check systems, and consider readiness checks as a complementary service to those systems.

Readiness checks audit resources to determine their readiness based on a set of pre-defined rules for that type of resource. After you create your recovery group with the replicas, you add Route 53 ARC readiness checks for the resources in your application, so Route 53 ARC can help make sure that the replicas have the same setup and configuration over time.

Topics
- Creating recovery groups (p. 60)
- Updating and deleting recovery groups and cells (p. 60)
Creating recovery groups

The steps in this section explain how to create a recovery group on the Route 53 ARC console. To learn about using recovery readiness API operations with Amazon Route 53 Application Recovery Controller, see Recovery readiness (readiness check) actions (p. 47).

To create a recovery group

2. Under Application Recovery Controller, choose Readiness check.
3. On the Recovery readiness page, choose Create, and then choose a Recovery group.
4. Enter a name for your recovery group, and then choose Next.
5. Choose Create cells, and then choose Add cell.
6. Enter a name for the cell. For example, if you have an application replica in US West (N. California), you could add a cell named MyApp-us-west-1.
7. Choose Add cell, and add a name for a second cell. For example, if you have a replica in US East (Ohio), you could add a cell named MyApp-us-east-2.
8. If you want to add nested cells (replicas in Availability Zones within Regions), choose Action, choose Add nested cell, and then enter a name.
9. When you’ve added all of the cells and nested cells for your application replicas, choose Next.
10. Review your recovery group, and then choose Create recovery group.

Updating and deleting recovery groups and cells

The steps in this section explain how to update and delete a recovery group, and delete a cell on the Route 53 ARC console. To learn about using recovery readiness API operations with Amazon Route 53 Application Recovery Controller, see Recovery readiness (readiness check) actions (p. 47).

To update or delete a recovery group, or delete a cell

2. Under Application Recovery Controller, choose Readiness check.
3. On the Recovery readiness page, choose a recovery group.
4. To work with a recovery group, choose Action, and then choose Edit recovery group or Delete recovery group.
5. When you edit a recovery group, you can add or remove cells or nested cells.
   • To add a cell, choose Add cell.
   • To remove a cell, under the Action label next to the cell, choose Delete cell.

Creating and updating readiness checks in Route 53 ARC

Creating and updating a readiness check

The steps in this section explain how to create a readiness check on the Route 53 ARC console. To learn about using recovery readiness API operations with Amazon Route 53 Application Recovery Controller, see Recovery readiness (readiness check) actions (p. 47).
To update a readiness check, you can edit the resource set for the readiness check, to add or remove resources or to change the readiness scope for a resource.

**To create a readiness check**

2. Under Application Recovery Controller, choose Readiness check.
3. On the Readiness page, choose Create, and then choose a Readiness check.
4. Enter a name for your readiness check, choose the resource type that you want to check, and then choose Next.
5. Add a resource set for your readiness check. A resource set is a group of resources of the same type in different replicas. Choose one of the following:
   - Create a readiness check with resources in a resource set that you've already created.
   - Create a new resource set.

   If you choose to create a new resource set, enter a name for it and choose Add.
6. Copy and paste Amazon Resource Names (ARNs) one by one for each resource that you want to include in the set, and then choose Next.

   **Tip**
   For examples and more information about the ARN format that Route 53 ARC expects for each resource type, see Resource types and ARN formats in Route 53 ARC (p. 72).
7. If you like, view the readiness rules that will be used when Route 53 ARC checks the type of resource you included in this readiness check. Then choose Next.
8. (Optional) Under Recovery group name, choose a recovery group to associate the readiness check with and then, for each resource ARN, choose a cell (Region or Availability Zone) from the drop-down menu that the resource is in. If it's an application-level resource, like a DNS routing policy, choose global resource (no cell).

   *This specifies the readiness scopes for the resources in the readiness check.*

   **Important**
   Although this step is optional, readiness scopes must be added to get summary readiness information for your recovery group and cells. If you skip this step and don't associate the readiness check with your recovery group's resources by choosing readiness scopes here, Route 53 ARC cannot return summary readiness information for the recovery group or cells.
9. Choose Next.
10. Review the information on the confirmation page, and then choose Create readiness check.

**To delete a readiness check**

2. Under Application Recovery Controller, choose Readiness check.
3. Choose a readiness check, and under Actions, choose Delete.

**Creating and editing resource sets**

Typically, you create a resource set as part of creating a readiness check, but you can create a resource set separately as well. You can also edit a resource set to add or remove resources. The steps in this section explain how to create or edit a resource set on the Route 53 ARC console. To learn about using recovery readiness API operations with Amazon Route 53 Application Recovery Controller, see Recovery readiness (readiness check) actions (p. 47).
To create a resource set
2. Under Application Recovery Controller, choose Resource sets.
3. Choose Create.
4. Enter a name for the resource set, and then choose the type of resource to include in the set.
5. Choose Add, and then enter the Amazon Resource Name (ARN) for the resource to add to the set.
6. After you’ve finished adding resources, choose Create resource set.

To edit a resource set
2. Under Application Recovery Controller, choose Resource sets.
3. Choose Action, and then choose Edit.
4. Do one of the following:
   - To remove a resource from the set, choose Remove.
   - To add a resource to the set, choose Add, and then enter the Amazon Resource Name (ARN) for the resource.
5. You can also edit the readiness scope for the resource, to associate the resource with a different cell for the readiness check.
6. Choose Save.

Monitoring readiness status in Route 53 ARC

You can see readiness for your application in Amazon Route 53 Application Recovery Controller at the following levels:

- The readiness check level for the resources in a resource set
- The individual resource level
- The cell (application replica) level for all the resources in an Availability Zone or AWS Region
- The recovery group level for the application as a whole

You can be notified about readiness status changes, or you can monitor readiness status changes in the Route 53 console or by using Route 53 ARC CLI commands.

Readiness status notification

You can use Amazon EventBridge to set up event-driven rules to monitor Route 53 ARC resources and notify you about changes in readiness status. For more information, see Using Route 53 ARC with Amazon EventBridge (p. 104).

Monitoring readiness status in the Route 53 ARC console

This section explains how to monitor recovery readiness in the AWS Management Console.

To monitor recovery readiness in the Route 53 ARC console
2. Under **Application Recovery Controller**, choose **Readiness check**.
3. On the **Readiness** page, under **Recovery group**, view the **Recovery group readiness status** for each recovery group (application).

You can also view the readiness of specific cells or individual resources.

### Monitoring readiness status by using CLI commands

This section provides examples of AWS CLI commands to use to see the readiness status for your application and resources at different levels.

#### Readiness for a resource set

The status of a readiness check you've created for a resource set (a group of resources).

```
aws route53-recovery-readiness --region us-west-2 get-readiness-check-status --readiness-check-name ReadinessCheckName
```

#### Readiness for a single resource

To get the status of a single resource in a readiness check, including the status of each readiness rule that is checked, specify the readiness check name and a resource ARN. For example:

```
aws route53-recovery-readiness --region us-west-2 get-readiness-check-status --readiness-check-name ReadinessCheckName --resource-arn "arn:aws:dynamodb:us-west-2:111122223333:table/TableName"
```

#### Readiness for a cell

The status of a single cell, that is, a Region or Availability Zone.

```
aws route53-recovery-readiness --region us-west-2 get-cell-readiness-summary --cell-name CellName
```

#### Readiness for an application

The status of the overall application, at the recovery group level.

```
aws route53-recovery-readiness --region us-west-2 get-recovery-group-readiness-summary --recovery-group-name RecoveryGroupName
```

### Readiness rules descriptions in Route 53 ARC

This section lists the readiness rules descriptions for all the types of resources supported by Amazon Route 53 Application Recovery Controller. To see a list of the resource types supported by Route 53 ARC, see [Resource types and ARN formats in Route 53 ARC](p. 72).

You can also view the readiness rules descriptions on the Route 53 ARC console or by using an API operation, by doing the following:

- To view readiness rules in the console, follow the steps in the following procedure: [View readiness rules on the console](p. 72).
- To view readiness rules by using the API, see the [ListRules](p. 72) operation.

#### Topics

- [Readiness rules in Route 53 ARC](p. 64)
- [View readiness rules on the console](p. 72)
Readiness rules in Route 53 ARC

This section lists the set of readiness rules for each resource type that is supported by Route 53 ARC.

As you look through the rule descriptions, you can see that most of them include the terms **Inspects all** or **Inspects each**. To understand how these terms explain how a rule works in the context of a readiness check, and other details about how Route 53 ARC sets readiness status, see How readiness rules determine readiness status (p. 57).

**Readiness rules**

Route 53 ARC audits resources by using the following readiness rules.

**Amazon API Gateway Version 1 stages**

- **ApiGwV1ApiKeyCount**: Inspects all API Gateway stages to ensure that they have the same number of API Keys linked to them.
- **ApiGwV1ApiKeySource**: Inspects all API Gateway stages to ensure that they have the same value for API Key Source.
- **ApiGwV1BasePath**: Inspects all API Gateway stages to ensure that they are linked to the same base path.
- **ApiGwV1BinaryMediaTypes**: Inspects all API Gateway stages to ensure that they support the same binary media types.
- **ApiGwV1CacheClusterEnabled**: Inspects all API Gateway stages to ensure that either all have Cache Cluster enabled, or none do.
- **ApiGwV1CacheClusterSize**: Inspects all API Gateway stages to ensure that they have the same Cache Cluster Size. If one has a greater value, the others are marked NOT READY.
- **ApiGwV1CacheClusterStatus**: Inspects all API Gateway stages to ensure that the Cache Cluster is in the AVAILABLE state.
- **ApiGwV1DisableExecuteApiEndpoint**: Inspects all API Gateway stages to ensure that either all have Execute API Endpoint disabled, or none do.
- **ApiGwV1DomainName**: Inspects all API Gateway stages to ensure that they are linked to the same domain name.
- **ApiGwV1EndpointConfiguration**: Inspects all API Gateway stages to ensure that they are linked to a domain with the same endpoint configuration.
- **ApiGwV1EndpointDomainNameStatus**: Inspects all API Gateway stages to ensure that the domain name that they are linked to is in the AVAILABLE state.
- **ApiGwV1MethodSettings**: Inspects all API Gateway stages to ensure that they have the same value for Method Settings.
- **ApiGwV1MutualTlsAuthentication**: Inspects all API Gateway stages to ensure that they have the same value for Mutual TLS Authentication.
- **ApiGwV1Policy**: Inspects all API Gateway stages to ensure that either all use API level policies, or none do.
- **ApiGwV1RegionalDomainName**: Inspects all API Gateway stages to ensure that they are linked to the same Regional domain name. Note: This rule does not affect readiness status.
- **ApiGwV1ResourceMethodConfigs**: Inspects all API Gateway stages to ensure that they have a similar resource hierarchy, including the related configurations.
- **ApiGwV1SecurityPolicy**: Inspects all API Gateway stages to ensure that they have the same value for Security Policy.
- **ApiGwV1Quotas**: Inspects all API Gateway groups to ensure that they conform to quotas (limits) that are managed by Service Quotas.
- **ApiGwV1UsagePlans**: Inspects all API Gateway stages to ensure that they are linked to Usage Plans with the same configuration.
Amazon API Gateway Version 2 stages

- **ApiGwV2ApiKeySelectionExpression**: Inspects all API Gateway stages to ensure that they have the same value for API Key Selection Expression.
- **ApiGwV2ApiMappingSelectionExpression**: Inspects all API Gateway stages to ensure that they have the same value for API Mapping Selection Expression.
- **ApiGwV2CorsConfiguration**: Inspects all API Gateway stages to ensure that they have the same CORS related configuration.
- **ApiGwV2DomainName**: Inspects all API Gateway stages to ensure that they are linked to the same domain name.
- **ApiGwV2DomainNameStatus**: Inspects all API Gateway stages to ensure that the domain name is in the AVAILABLE state.
- **ApiGwV2EndpointType**: Inspects all API Gateway stages to ensure that they have the same value for Endpoint Type.
- **ApiGwV2Quotas**: Inspects all API Gateway groups to ensure that they conform to quotas (limits) that are managed by Service Quotas.
- **ApiGwV2MutualTlsAuthentication**: Inspects all API Gateway stages to ensure that they have the same value for Mutual TLS Authentication.
- **ApiGwV2ProtocolType**: Inspects all API Gateway stages to ensure that they have the same value for Protocol Type.
- **ApiGwV2RouteConfigs**: Inspects all API Gateway stages to ensure that they have the same hierarchy of routes with the same configuration.
- **ApiGwV2RouteSelectionExpression**: Inspects all API Gateway stages to ensure that they have the same value for Route Selection Expression.
- **ApiGwV2RouteSettings**: Inspects all API Gateway stages to ensure that they have the same value for Default Route Settings.
- **ApiGwV2SecurityPolicy**: Inspects all API Gateway stages to ensure that they have the same value for Security Policy.
- **ApiGwV2StageVariables**: Inspects all API Gateway stages to ensure that they all have the same Stage Variables as the other stages.
- **ApiGwV2ThrottlingBurstLimit**: Inspects all API Gateway stages to ensure that they have the same value for Throttling Burst Limit.
- **ApiGwV2ThrottlingRateLimit**: Inspects all API Gateway stages to ensure that they have the same value for Throttling Rate Limit.

Amazon Aurora clusters

- **RdsClusterStatus**: Inspects each Aurora cluster to ensure that it has a status of either AVAILABLE or BACKING-UP.
- **RdsEngineMode**: Inspects all Aurora clusters to ensure that they have the same value for Engine Mode.
- **RdsEngineVersion**: Inspects all Aurora clusters to ensure that they have the same value for Major Version.
- **RdsGlobalReplicaLag**: Inspects each Aurora cluster to ensure that it has a Global Replica Lag of less than 30 seconds.
- **RdsNormalizedCapacity**: Inspects all Aurora clusters to ensure that they have a normalized capacity within 15% of the maximum in the resource set.
- **RdsInstanceType**: Inspects all Aurora clusters to ensure that they have the same instance types.
- **RdsQuotas**: Inspects all Aurora clusters to ensure that they conform to quotas (limits) that are managed by Service Quotas.

Auto Scaling groups

- **AsgMinSizeAndMaxSize**: Inspects all Auto Scaling groups to ensure that they have the same minimum and maximum group sizes.
• **AsgAZCount**: Inspects all Auto Scaling groups to ensure that they have the same number of Availability Zones.

• **AsgInstanceTypes**: Inspects all Auto Scaling groups to ensure that they have the same instance types. Note: This rule does not affect readiness status.

• **AsgInstanceSizes**: Inspects all Auto Scaling groups to ensure that they have the same instance sizes.

• **AsgNormalizedCapacity**: Inspects all Auto Scaling groups to ensure that they have a normalized capacity within 15% of the maximum in the resource set.

• **AsgQuotas**: Inspects all Auto Scaling groups to ensure that they conform to quotas (limits) that are managed by Service Quotas.

**CloudWatch alarms**

• **CloudWatchAlarmState**: Inspects CloudWatch alarms to ensure that each is not in the ALARM or INSUFFICIENT_DATA state.

**Customer gateways**

• **CustomerGatewayIpAddress**: Inspects all customer gateways to ensure that they have the same IP address.

• **CustomerGatewayState**: Inspects customer gateways to ensure that each is in the AVAILABLE state.

• **CustomerGatewayVPNType**: Inspects all customer gateways to ensure that they have the same VPN type.

**DNS target resources**

• **DnsTargetResourceHostedZoneConfigurationRule**: Inspects all DNS target resources to ensure that they have the same Amazon Route 53 hosted zone ID and that each hosted zone is not private. Note: This rule does not affect readiness status.

• **DnsTargetResourceRecordSetConfigurationRule**: Inspects all DNS target resources to ensure that they have the same resource record cache time to live (TTL) and that the TTLs are less than or equal to 300.

• **DnsTargetResourceRoutingRule**: Inspects each DNS target resource associated with an alias resource record set to ensure that it routes traffic to the DNS name configured on the target resource. Note: This rule does not affect readiness status.

• **DnsTargetResourceHealthCheckRule**: Inspects all DNS target resources to ensure that health checks are associated with their resource record sets when appropriate and not otherwise. Note: This rule does not affect readiness status.

**Amazon DynamoDB tables**

• **DynamoConfiguration**: Inspects all DynamoDB tables to ensure that they have the same keys, attributes, server-side encryption, and streams configurations.

• **DynamoTableStatus**: Inspects each DynamoDB table to ensure that it has a status of ACTIVE.

• **DynamoCapacity**: Inspects all DynamoDB tables to ensure that their provisioned read capacities and write capacities are within 20% of the maximum capacities in the resource set.

• **DynamoPeakRcuWcu**: Inspects each DynamoDB table to ensure that it has had similar peak traffic to the other tables, to assure provisioned capacity.

• **DynamoGsiPeakRcuWcu**: Inspects each DynamoDB table to ensure that it has had similar maximum read and write capacity to the other tables, to assure provisioned capacity.

• **DynamoGsiConfig**: Inspects all DynamoDB tables that have global secondary indexes to ensure that the tables use the same index, key schema, and projection.

• **DynamoGsiStatus**: Inspects all DynamoDB tables that have global secondary indexes to ensure that the global secondary indexes have an ACTIVE status.

• **DynamoGsiCapacity**: Inspects all DynamoDB tables that have global secondary indexes to ensure that the tables have provisioned GSI read capacities and GSI write capacities within 20% of the maximum capacities in the resource set.
• **DynamoReplicationLatency**: Inspects all DynamoDB tables that are global tables to ensure that they have the same replication latency.

• **DynamoAutoScalingConfiguration**: Inspects all DynamoDB tables that have Auto Scaling enabled to ensure that they have the same minimum, maximum, and target read and write capacities.

• **DynamoQuotas**: Inspects all DynamoDB tables to ensure that they conform to quotas (limits) that are managed by Service Quotas.

**Elastic Load Balancing (Classic Load Balancers)**

• **ElbV1CheckAzCount**: Inspects each Classic Load Balancer to ensure that it is attached to only one Availability Zone. Note: This rule does not affect readiness status.

• **ElbV1AnyInstances**: Inspects all Classic Load Balancers to ensure that they have at least one EC2 instance.

• **ElbV1AnyInstancesHealthy**: Inspects all Classic Load Balancers to ensure that they have at least one healthy EC2 instance.

• **ElbV1Scheme**: Inspects all Classic Load Balancers to ensure that they have the same load balancer scheme.

• **ElbV1HealthCheckThreshold**: Inspects all Classic Load Balancers to ensure that they have the same health check threshold value.

• **ElbV1HealthCheckInterval**: Inspects all Classic Load Balancers to ensure that they have the same health check interval value.

• **ElbV1CrossZoneRoutingEnabled**: Inspects all Classic Load Balancers to ensure that they have the same value for cross-zone load balancing (ENABLED or DISABLED).

• **ElbV1AccessLogsEnabledAttribute**: Inspects all Classic Load Balancers to ensure that they have the same value for access logs (ENABLED or DISABLED).

• **ElbV1ConnectionDrainingEnabledAttribute**: Inspects all Classic Load Balancers to ensure that they have the same value for connection draining (ENABLED or DISABLED).

• **ElbV1ConnectionDrainingTimeoutAttribute**: Inspects all Classic Load Balancers to ensure that they have the same connection draining timeout value.

• **ElbV1IdleTimeoutAttribute**: Inspects all Classic Load Balancers to ensure that they have the same value for idle timeout.

• **ElbV1ProvisionedCapacityLcuCount**: Inspects all Classic Load Balancers with a provisioned LCU greater than 10 to ensure that they are within 20% of the highest provisioned LCU in the resource set.

• **ElbV1ProvisionedCapacityStatus**: Inspects the provisioned capacity status on each Classic Load Balancer to ensure that it does not have a value of DISABLED or PENDING.

**Amazon EBS volumes**

• **EbsVolumeEncryption**: Inspects all EBS volumes to ensure that they have the same value for encryption (ENABLED or DISABLED).

• **EbsVolumeEncryptionDefault**: Inspects all EBS volumes to ensure that they have the same value for encryption by default (ENABLED or DISABLED).

• **EbsVolumesIops**: Inspects all EBS volumes to ensure that they have the same input/output operations per second (IOPS).

• **EbsVolumeKmsKeyId**: Inspects all EBS volumes to ensure that they have the same default AWS KMS key ID.

• **EbsVolumeMultiAttach**: Inspects all EBS volumes to ensure that they have the same value for multi-attach (ENABLED or DISABLED).

• **EbsVolumeQuotas**: Inspects all EBS volumes to ensure that they conform to quotas (limits) that are set by Service Quotas.

• **EbsVolumeSize**: Inspects all EBS volumes to ensure that they have the same readable size.

• **EbsVolumeState**: Inspects all EBS volumes to ensure that they have the same volume state.

• **EbsVolumeType**: Inspects all EBS volumes to ensure that they have the same volume type.
AWS Lambda functions

- **LambdaMemorySize**: Inspects all Lambda functions to ensure that they have the same memory size. If one has more memory, the others are marked **NOT READY**.
- **LambdaFunctionTimeout**: Inspects all Lambda functions to ensure that they have the same timeout value. If one has a greater value, the others are marked **NOT READY**.
- **LambdaFunctionRuntime**: Inspects all Lambda functions to ensure that they all have the same runtime.
- **LambdaFunctionReservedConcurrentExecutions**: Inspects all Lambda functions to ensure that they all have the same value for Reserved Concurrent Executions. If one has a greater value, the others are marked **NOT READY**.
- **LambdaFunctionDeadLetterConfig**: Inspects all Lambda functions to ensure that they either all have a Dead Letter Config defined, or that none of them do.
- **LambdaFunctionProvisionedConcurrencyConfig**: Inspects all Lambda functions to ensure that they have the same value for Provisioned Concurrency.
- **LambdaFunctionSecurityGroupCount**: Inspects all Lambda functions to ensure that they have the same value for Security Groups.
- **LambdaFunctionSubnetIdCount**: Inspects all Lambda functions to ensure that they have the same value for Subnet IDs.
- **LambdaFunctionEventSourceMappingMatch**: Inspects all Lambda functions to ensure that all of the chosen Event Source Mapping properties match between them.
- **LambdaFunctionLimitsRule**: Inspects all Lambda functions to ensure that they conform to quotas (limits) that are managed by Service Quotas.

Network Load Balancers and Application Load Balancers

- **ElbV2CheckAzCount**: Inspects each Network Load Balancer to ensure that it is attached to only one Availability Zone. Note: This rule does not affect readiness status.
- **ElbV2TargetGroupsCanServeTraffic**: Inspects each Network Load Balancer and Application Load Balancer to ensure that it has at least one healthy Amazon EC2 instance.
- **ElbV2State**: Inspects each Network Load Balancer and Application Load Balancer to ensure that it is in the **ACTIVE** state.
- **ElbV2IpAddressType**: Inspects all Network Load Balancers and Application Load Balancers to ensure that they have the same IP address types.
- **ElbV2Scheme**: Inspects all Network Load Balancers and Application Load Balancers to ensure that they have the same scheme.
- **ElbV2Type**: Inspects all Network Load Balancers and Application Load Balancers to ensure that they have the same type.
- **ElbV2S3LogsEnabled**: Inspects all Network Load Balancers and Application Load Balancers to ensure that they have the same value for Amazon S3 server access logs (ENABLED or DISABLED).
- **ElbV2DeletionProtection**: Inspects all Network Load Balancers and Application Load Balancers to ensure that they have the same value for deletion protection (ENABLED or DISABLED).
- **ElbV2IdleTimeoutSeconds**: Inspects all Network Load Balancers and Application Load Balancers to ensure that they have the same value for idle time seconds.
- **ElbV2HttpDropInvalidHeaders**: Inspects all Network Load Balancers and Application Load Balancers to ensure that they have the same value for HTTP drop invalid headers.
- **ElbV2Http2Enabled**: Inspects all Network Load Balancers and Application Load Balancers to ensure that they have the same value for HTTP2 (ENABLED or DISABLED).
- **ElbV2CrossZoneEnabled**: Inspects all Network Load Balancers and Application Load Balancers to ensure that they have the same value for cross-zone load balancing (ENABLED or DISABLED).
- **ElbV2ProvisionedCapacityLcuCount**: Inspects all Network Load Balancers and Application Load Balancers with a provisioned LCU greater than 10 to ensure that they are within 20% of the highest provisioned LCU in the resource set.
Readiness rules in Route 53 ARC

Amazon MSK clusters

- **MskClusterClientSubnet**: Inspects each MSK cluster to ensure that it has only two or only three client subnets.
- **MskClusterInstanceType**: Inspects all MSK clusters to ensure that they have the same Amazon EC2 instance type.
- **MskClusterSecurityGroups**: Inspects all MSK clusters to ensure that they have the same security groups.
- **MskClusterStorageInfo**: Inspects all MSK clusters to ensure that they have the same EBS storage volume size. If one has a greater value, the others are marked NOT READY.
- **MskClusterACMCertificate**: Inspects all MSK clusters to ensure that they have the same list of client authorization certificate ARNs.
- **MskClusterServerProperties**: Inspects all MSK clusters to ensure that they have the same value for Current Broker Software Info.
- **MskClusterKafkaVersion**: Inspects all MSK clusters to ensure that they have the same Kafka version.
- **MskClusterEncryptionInTransitInCluster**: Inspects all MSK clusters to ensure that they have the same value for Encryption In Transit In Cluster.
- **MskClusterEncryptionInClientBroker**: Inspects all MSK clusters to ensure that they have the same value for Encryption In Transit Client Broker.
- **MskClusterEnhancedMonitoring**: Inspects all MSK clusters to ensure that they have the same value for Enhanced Monitoring.
- **MskClusterOpenMonitoringInJmx**: Inspects all MSK clusters to ensure that they have the same value for Open Monitoring JMX Exporter.
- **MskClusterOpenMonitoringInNode**: Inspects all MSK clusters to ensure that they have the same value for Open Monitoring Not Exporter.
- **MskClusterLoggingInS3**: Inspects all MSK clusters to ensure that they have the same value for Is Logging in S3.
- **MskClusterLoggingInFirehose**: Inspects all MSK clusters to ensure that they have the same value for Is Logging In Firehose.
- **MskClusterLoggingInCloudWatch**: Inspects all MSK clusters to ensure that they have the same value for Is Logging Available In CloudWatch Logs.
- **MskClusterNumberOfBrokerNodes**: Inspects all MSK clusters to ensure that they have the same value for Number of Broker Nodes. If one has a greater value, the others are marked NOT READY.
- **MskClusterState**: Inspects each MSK cluster to ensure that it is in an ACTIVE state.
- **MskClusterLimitsRule**: Inspects all Lambda functions to ensure that they conform to quotas (limits) that are managed by Service Quotas.

Amazon Route 53 health checks

- **R53HealthCheckType**: Inspects each Route 53 health check to ensure that it is not of type CALCULATED and that all checks are of the same type.
- **R53HealthCheckDisabled**: Inspects each Route 53 health check to ensure that it does not have a DISABLED state.
- **R53HealthCheckStatus**: Inspects each Route 53 health check to ensure that it has a SUCCESS status.
- **R53HealthCheckRequestInterval**: Inspects all Route 53 health checks to ensure that they all have the same value for Request Interval.
- **R53HealthCheckFailureThreshold**: Inspects all Route 53 health checks to ensure that they all have the same value for Failure Threshold.
• **R53HealthCheckEnableSNI:** Inspects all Route 53 health checks to ensure that they all have the same value for Enable SNI.

• **R53HealthCheckSearchString:** Inspects all Route 53 health checks to ensure that they all have the same value for Search String.

• **R53HealthCheckRegions:** Inspects all Route 53 health checks to ensure that they all have the same list of AWS Regions.

• **R53HealthCheckMeasureLatency:** Inspects all Route 53 health checks to ensure that they all have the same value for Measure Latency.

• **R53HealthCheckInsufficientDataHealthStatus:** Inspects all Route 53 health checks to ensure that they all have the same value for Insufficient Data Health Status.

• **R53HealthCheckInverted:** Inspects all Route 53 health checks to ensure that they are all Inverted, or are all not Inverted.

• **R53HealthCheckResourcePath:** Inspects all Route 53 health checks to ensure that they all have the same value for Resource Path.

• **R53HealthCheckCloudWatchAlarm:** Inspects all Route 53 health checks to ensure that the CloudWatch alarms associated with them have the same settings and configurations.

### Amazon SNS subscriptions

• **SnsSubscriptionProtocol:** Inspects all SNS subscriptions to ensure that they have the same protocol.

• **SnsSubscriptionSqsLambdaEndpoint:** Inspects all SNS subscriptions that have Lambda or SQS endpoints to ensure that they have different endpoints.

• **SnsSubscriptionNonAwsEndpoint:** Inspects all SNS subscriptions that have a non-AWS service endpoint type, for example, email, to ensure that the subscriptions have the same endpoint.

• **SnsSubscriptionPendingConfirmation:** Inspects all SNS subscriptions to ensure that they have the same value for 'Pending Confirmations'.

• **SnsSubscriptionDeliveryPolicy:** Inspects all SNS subscriptions that use HTTP/S to ensure that they have the same value for 'Effective Delivery Period'.

• **SnsSubscriptionRawMessageDelivery:** Inspects all SNS subscriptions to ensure that they have the same value for 'Raw Message Delivery'.

• **SnsSubscriptionFilter:** Inspects all SNS subscriptions to ensure that they have the same value for 'Filter Policy'.

• **SnsSubscriptionRedrivePolicy:** Inspects all SNS subscriptions to ensure that they have the same value for 'Redrive Policy'.

• **SnsSubscriptionEndpointEnabled:** Inspects all SNS subscriptions to ensure that they have the same value for 'Endpoint Enabled'.

• **SnsSubscriptionLambdaEndpointValid:** Inspects all SNS subscriptions that have Lambda endpoints to ensure that they have valid Lambda endpoints.

• **SnsSubscriptionSqsEndpointValidRule:** Inspects all SNS subscriptions that use SQS endpoints to ensure that they have valid SQS endpoints.

• **SnsSubscriptionQuotas:** Inspects all SNS subscriptions to ensure that they conform to quotas (limits) that are managed by Service Quotas.

### Amazon SNS topics

• **SnsTopicDisplayName:** Inspects all SNS topics to ensure that they have the same value for DisplayName.

• **SnsTopicDeliveryPolicy:** Inspects all SNS topics that have HTTPS subscribers to ensure that they have the same EffectiveDeliveryPolicy.

• **SnsTopicSubscription:** Inspects all SNS topics to ensure that they have the same number of subscribers for each of their protocols.

• **SnsTopicAwsKmsKey:** Inspects all SNS topics to ensure that all of the topics or none of the topics have an AWS KMS key.
• **SnsTopicQuotas**: Inspects all SNS topics to ensure that they conform to quotas (limits) that are managed by Service Quotas.

Amazon SQS queues
• **SqsQueueType**: Inspects all SQS queues to ensure that they are all the same value for Type.
• **SqsQueueDelaySeconds**: Inspects all SQS queues to ensure that they all have the same value for Delay Seconds.
• **SqsQueueMaximumMessageSize**: Inspects all SQS queues to ensure that they all have the same value for Maximum Message Size.
• **SqsQueueMessageRetentionPeriod**: Inspects all SQS queues to ensure that they all have the same value for Message Retention Period.
• **SqsQueueReceiveMessageWaitTimeSeconds**: Inspects all SQS queues to ensure that they all have the same value for Receive Message Wait Time Seconds.
• **SqsQueueRedrivePolicyMaxReceiveCount**: Inspects all SQS queues to ensure that they all have the same value for Redrive Policy Max Receive Count.
• **SqsQueueVisibilityTimeout**: Inspects all SQS queues to ensure that they all have the same value for Visibility Timeout.
• **SqsQueueContentBasedDeduplication**: Inspects all SQS queues to ensure that they all have the same value for Content-Based Deduplication.
• **SqsQueueQuotas**: Inspects all SQS queues to ensure that they conform to quotas (limits) that are managed by Service Quotas.

Amazon VPCs
• **VpcCidrBlock**: Inspects all VPCs to ensure that they all have the same value for CIDR block network size.
• **VpcCidrBlocksSameProtocolVersion**: Inspects all VPCs that have the same CIDR blocks to ensure that they have the same value for Internet Stream Protocol version number.
• **VpcCidrBlocksStateInAssociationSets**: Inspects all CIDR block association sets for all VPCs to ensure that they all have CIDR blocks that are in an ASSOCIATED state.
• **Vpclv6Cidr BlocksStateInAssociationSets**: Inspects all CIDR block association sets for all VPCs to ensure that they all have CIDR blocks with the same number of addresses.
• **VpcCidrBlocksInAssociationSets**: Inspects all CIDR block association sets for all VPCs to ensure that they all have the same size.
• **Vpclv6CidrBlocksInAssociationSets**: Inspects all IPv6 CIDR block association sets for all VPCs to ensure that they have the same size.
• **VpcState**: Inspects each VPC to ensure that it is in an AVAILABLE state.
• **VpcInstanceTenancy**: Inspects all VPCs to ensure that they all have the same value for Instance Tenancy.
• **VpcIsDefault**: Inspects all VPCs to ensure that they have the same value for Is Default.
• **VpcSubnetState**: Inspects each VPC subnet to ensure that it is in an AVAILABLE state.
• **VpcSubnetAvailableIpAddressCount**: Inspects each VPC subnet to ensure that it has an available IP address count greater than zero.
• **VpcSubnetCount**: Inspects all VPC subnets to ensure that they have the same number of subnets.
• **VpcQuotas**: Inspects all VPC subnets to ensure that they conform to quotas (limits) that are managed by Service Quotas.

AWS VPN connections
• **VpnConnectionsRouteCount**: Inspects all VPN connections to ensure that they have at least one route, and also the same number of routes.
• **VpnConnectionsEnableAcceleration**: Inspects all VPN connections to ensure that they have the same value for Enable Accelerations.
- **VpnConnectionsStaticRoutesOnly**: Inspects all VPN connections to ensure that they have the same value for Static Routes Only.
- **VpnConnectionsCategory**: Inspects all VPN connections to ensure that they have a category of VPN.
- **VpnConnectionsCustomerConfiguration**: Inspects all VPN connections to ensure that they have the same value for Customer Gateway Configuration.
- **VpnConnectionsCustomerGatewayId**: Inspects each VPN connection to ensure that it has a customer gateway attached.
- **VpnConnectionsRoutesState**: Inspects all VPN connections to ensure that they are in an AVAILABLE state.
- **VpnConnectionsVgwTelemetryStatus**: Inspects each VPN connection to ensure that it has a VGW status of UP.
- **VpnConnectionsVgwTelemetryIpAddress**: Inspects each VPN connection to ensure that it has a different outside IP address for each VGW telemetry.
- **VpnConnectionsTunnelOptions**: Inspects all VPN connections to ensure that they have the same tunnel options.
- **VpnConnectionsRoutesCidr**: Inspects all VPN connections to ensure that they have the same destination CIDR blocks.
- **VpnConnectionsInstanceType**: Inspects all VPN connections to ensure that they have the same Instance Type.

### AWS VPN gateways
- **VpnGatewayState**: Inspects all VPN gateways to ensure that they are in an AVAILABLE state.
- **VpnGatewayAsn**: Inspects all VPN gateways to ensure that they have the same ASN.
- **VpnGatewayType**: Inspects all VPN gateways to ensure that they have the same type.
- **VpnGatewayAttachment**: Inspects all VPN gateways to ensure that they have the same attachment configurations.

## View readiness rules on the console

You can view readiness rules on the AWS Management Console, listed by each resource type.

### To view readiness rules on the console

2. Under Application Recovery Controller, choose Readiness rules.
3. Under Resource type, choose the resource type that you want to view the rules for.

## Resource types and ARN formats in Route 53 ARC

When you create a resource set in Amazon Route 53 Application Recovery Controller, you specify the type of resource to include in the set and Amazon Resource Names (ARNs) for each of the resources to include. Route 53 ARC expects a specific ARN format for each resource type. This section lists the resource types supported by Route 53 ARC and the associated ARN formats for each one.

The specific format depends on the resource. When you provide an ARN, replace the italicized text with your resource-specific information.

**Note**

Be aware that the ARN format that Route 53 ARC requires for resources might differ from the ARN format that a service itself requires for its resources. For example, the ARN formats that are...
described in the Resource type sections for each service in the Service Authorization Reference might not include the AWS account ID or other information that Route 53 ARC needs to support features in the Route 53 ARC service.

AWS::ApiGateway::Stage
An Amazon API Gateway Version 1 stage.
- **ARN format:** `arn:partition:apigateway:region:account:/restapis/api-id/stages/stage-name`
  
  Example: `arn:aws:apigateway:us-east-1:111122223333:/restapis/123456789/stages/ExampleStage`
  
  For more information, see [API Gateway Amazon Resource Name (ARN) reference](#).

AWS::ApiGatewayV2::Stage
An Amazon API Gateway Version 2 stage.
- **ARN format:** `arn:partition:apigateway:region:account:/apis/api-id/stages/stage-name`
  
  Example: `arn:aws:apigateway:us-east-1:111122223333:/apis/123456789/stages/ExampleStage`
  
  For more information, see [API Gateway Amazon Resource Name (ARN) reference](#).

AWS::CloudWatch::Alarm
An Amazon CloudWatch alarm.
- **ARN format:** `arn:partition:cloudwatch:region:account:alarm:alarm-name`
  
  
  For more information, see [Resource types defined by Amazon CloudWatch](#).

AWS::DynamoDB::Table
An Amazon DynamoDB table.
- **ARN format:** `arn:partition:dynamodb:region:account:table/table-name`
  
  Example: `arn:aws:dynamodb:us-west-2:111122223333:table/BigTable`
  
  For more information, see [DynamoDB resources and operations](#).

AWS::EC2::CustomerGateway
A customer gateway device.
- **ARN format:** `arn:partition:ec2:region:account:customer-gateway/CustomerGatewayId`
  
  
  For more information, see [Resource types defined by Amazon EC2](#).

AWS::EC2::Volume
An Amazon EBS volume.
- **ARN format:** `arn:partition:ec2:region:account:volume/VolumeId`
  
  
  For more information, see [API Gateway Amazon Resource Name (ARN) reference](#).
AWS::ElasticLoadBalancing::LoadBalancer

A Classic Load Balancer.
- **ARN format:**
  - `arn:partition:elasticloadbalancing:region:account:loadbalancer/LoadBalancerName`


For more information, see [Elastic Load Balancing resources](#).

AWS::ElasticLoadBalancingV2::LoadBalancer

A Network Load Balancer or an Application Load Balancer.
- **ARN format for Network Load Balancer:**
  - `arn:partition:elasticloadbalancing:region:account:loadbalancer/net/LoadBalancerName`


- **ARN format for Application Load Balancer:**
  - `arn:partition:elasticloadbalancing:region:account:loadbalancer/app/LoadBalancerName`


For more information, see [Elastic Load Balancing resources](#).

AWS::Lambda::Function

An AWS Lambda function.
- **ARN format:**
  - `arn:partition:lambda:region:account:function:FunctionName`


For more information, see [Resources and conditions for Lambda actions](#).

AWS::MSK::Cluster

An Amazon MSK cluster.
- **ARN format:**
  - `arn:partition:kafka:region:account:cluster/ClusterName/UUID`


For more information, see [Resource types defined by Amazon Managed Streaming for Apache Kafka](#).

AWS::RDS::DBCluster

An Aurora DB cluster.
- **ARN format:**
  - `arn:partition:rds:region:account:cluster:DbClusterInstanceName`


For more information, see [Working with Amazon Resource Names (ARNs) in Amazon RDS](#).

AWS::Route53::HealthCheck

An Amazon Route 53 health check.
- **ARN format:**
  - `arn:partition:route53::healthcheck/Id`
Example: arn:aws:route53:::healthcheck/123456-1111-2222-3333

AWS::SQS::Queue

An Amazon SQS queue.

- **ARN format**: arn:partition:sqs:region:account:QueueName


  For more information, see Amazon Simple Queue Service resource and operations.

AWS::SNS::Topic

An Amazon SNS topic.

- **ARN format**: arn:partition:sns:region:account:TopicName


  For more information, see Amazon SNS resource ARN format.

AWS::SNS::Subscription

An Amazon SNS subscription.

- **ARN format**: arn:partition:sns:region:account:TopicName:SubscriptionId


AWS::EC2::VPC

A virtual private cloud (VPC).

- **ARN format**: arn:partition:ec2:region:account:vpc/VpcId


  For more information, see VPC Resources.

AWS::EC2::VPNTunnel

A virtual private network (VPN) connection.

- **ARN format**: arn:partition:ec2:region:account:vpn-connection/VpnConnectionId


  For more information, see Resource types defined by Amazon EC2.

AWS::EC2::VPNGateway

A virtual private network (VPN) gateway.

- **ARN format**: arn:partition:ec2:region:account:vpn-gateway/VpnGatewayId


  For more information, see Resource types defined by Amazon EC2.

AWS::Route53RecoveryReadiness::DNSTargetResource

A DNS target resource for readiness checks includes the DNS record type, domain name, Route 53 hosted zone ARN, and Network Load Balancer ARN or Route 53 record set ID.

- **ARN format for hosted zone**: arn:partition:route53::account:hostedzone/Id

  Example for a hosted zone: arn:aws:route53::111122223333:hostedzone/abcHostedZone
NOTE: You must include the account ID in hosted zone ARNs, as specified here. The account ID is required so that Route 53 ARC can poll the resource. The format is intentionally different from the ARN format that Amazon Route 53 requires, described in the Route 53 service Resource types in the Service Authorization Reference.

• ARN format for Network Load Balancer:
  arn:partition:elasticloadbalancing:region:account:loadbalancer/net/LoadBalancerName


For more information, see Elastic Load Balancing resources.

Getting architecture recommendations in Route 53 ARC

If you have an existing application, Amazon Route 53 Application Recovery Controller can evaluate the architecture of your application and routing policies to provide recommendations for modifying the design to improve your application's recovery resiliency. After you create a recovery group in Route 53 ARC that represents your application, follow the steps in this section to get recommendations for your application's architecture.

We recommend that you specify a target resource for the DNS target resource for your recovery group, if you haven't specified one yet, so that we can provide more detailed recommendations. When you provide additional information, Route 53 ARC can provide better recommendations for you. For example, if you enter an Amazon Route 53 resource record or a Network Load Balancer as a target resource, Route 53 ARC can provide information about whether you've created the optimal number of cells for your recovery group.

Note the following for DNS target resources:

• Specify only a Route 53 resource record or Network Load Balancer for a target resource.
• Create only one DNS target resource for each recovery group.
• Recommended: Create one DNS target resource for each cell.
• Group the DNS target resources into one resource set with a readiness check.

The following procedure explains how to create DNS target resources and get architecture recommendations for your application.

To get recommendations for updating your architecture

2. Under Application Recovery Controller, choose Readiness check.
3. Under Recovery group name, choose the recovery group that represents your application.
4. On the Recovery group details page, on the Action menu, choose Get architecture recommendations for this recovery group.
5. If you haven't created a DNS target resource readiness check yet, create one so that Route 53 ARC can provide architecture recommendations. Choose Create a DNS target resource.

For more information about DNS target resources, see Readiness check components (p. 3).
6. To create a resource set for a DNS target resource, you create a readiness check. Enter a name for the readiness check, and then, for the type of readiness check, choose DNS target resource.
7. Enter a name for the resource set.
8. Enter the attributes for your application, including the DNS name, hosted zone ARN, and record set ID.

**Tip**
To see the format for a hosted zone ARN, see ARN format for hosted zone in Resource types and ARN formats in Route 53 ARC (p. 72).

Optionally, but strongly recommended, choose Add optional attribute and provide a Network Load Balancer ARN or your domain's Route 53 resource record.

9. (Optional) In Recovery group configuration, choose a cell for your DNS target resource, to set the readiness scope.
10. Choose Create resource set.

Review the list of recommendations. Then you can decide whether and how to make changes to improve your app's recovery resilience.

### Create cross-account authorizations in Route 53 ARC

You might have your resources distributed across multiple AWS accounts, which can make it challenging to get a comprehensive view of your application's health and to get the information required to make quick decisions. To help streamline this in Amazon Route 53 Application Recovery Controller, you can use cross-account authorization.

Cross-account authorization in Route 53 ARC works with the readiness check feature. With cross-account authorization, you can use one central AWS account to monitor your resources that are located in multiple AWS accounts. In each account that has resources that you want to monitor, you authorize the central account to have access to those resources. Then the central account can create readiness checks for the resources in all the accounts and from the central account, you can monitor readiness for failover.

**Note**
Cross-account authorization setup isn't available in the console. Instead, use Route 53 ARC API operations to set up and work with cross-account authorization. To help you get started, this section provides AWS CLI command examples.

Let's say that an application has an account that has resources in the US West (Oregon) Region (us-west-2), and there's also an account that has resources that you'd like to monitor in the US East (N. Virginia) Region (us-east-1). Route 53 ARC can allow access for you to monitor both sets of resources from one account, us-west-2, by using cross-account authorization.

For example, let's say that you have the following AWS accounts:

- US-West account: 999999999999
- US-East account: 111111111111

In the us-east-1 account (111111111111), we can enable cross-account authorization to allow access by the us-west-2 account (999999999999) by specifying the Amazon Resource Name (ARN) for the (root) user in the us-west-2 IAM account: arn:aws:iam::999999999999:root. After we create the authorization, the us-west-2 account can add resources owned by us-east-1 to resource sets and create readiness checks to run on the resource sets.
The following example illustrates setting up cross-account authorization for one account. You must enable cross-account authorization in each additional account that has AWS resources that you want to add and monitor in Route 53 ARC.

**Note**
Route 53 ARC is a global service that supports endpoints in multiple AWS Regions but you must specify the US West (Oregon) Region (that is, specify the parameter `--region us-west-2`) in most Route 53 ARC CLI commands.

The following AWS CLI command shows how to set up cross-account authorization for this example:

```
aws route53-recovery-readiness --region us-west-2 --profile profile-in-us-east-1-account \
create-cross-account-authorization --cross-account-authorization arn:aws:iam::999999999999:root
```

To disable this authorization, do the following:

```
aws route53-recovery-readiness --region us-west-2 --profile profile-in-us-east-1-account \
delete-cross-account-authorization --cross-account-authorization arn:aws:iam::999999999999:root
```

To check in a specific account for all the accounts that you've provided cross-account authorization for, use the `list-cross-account-authorizations` command. Note that at this time, you can't check in the other direction. That is, there isn't an API operation that you can use with an account profile to list all of the accounts for which it has been granted cross-account authorization to add and monitor resources.

```
aws route53-recovery-readiness --region us-west-2 --profile profile-in-us-east-1-account \
list-cross-account-authorizations
```

```
{
   "CrossAccountAuthorizations": [
      "arn:aws:iam::999999999999:root"
   ]
}
```
Routing control in Amazon Route 53 Application Recovery Controller

To fail over traffic to application replicas in Amazon Route 53 Application Recovery Controller, you use routing controls that are integrated with a specific kind of health check in Amazon Route 53. Routing controls are simple on-off switches that enable you to switch your client traffic from one replica to another. The traffic rerouting is accomplished by routing control health checks that are set up with Amazon Route 53 DNS records. For example, DNS failover records, associated with domain names that front your application replicas. This chapter explains how routing control works, how to set up routing control components, and how to use them to reroute traffic for failover.

The routing control components in Route 53 ARC are: clusters, control panels, routing controls, and routing control health checks. All routing controls are grouped on control panels. You can group them on the default control panel that Route 53 ARC creates for your cluster, or create your own custom control panels. You must create a cluster before you can create a control panel or a routing control. Each cluster in Route 53 ARC is a data plane of endpoints in five AWS Regions.

After you create routing controls and routing control health checks, you can create safety rules to help prevent unintentional recovery automation side effects. You can update routing control states to reroute traffic, individually or in batches, by using the AWS CLI or API actions (recommended), or by using the AWS Management Console.

This chapter explains how routing controls work, and how to create and use them to reroute traffic for your application.

**Important**

To learn about preparing to use Route 53 ARC to reroute traffic as part of a failover plan for your application in a disaster scenario, see [Best practices for Amazon Route 53 Application Recovery Controller](p. 44).

**Topics**

- [About routing control](p. 79)
- [Creating routing control components in Route 53 ARC](p. 80)
- [Viewing and updating routing control states in Route 53 ARC](p. 87)
- [Creating safety rules in Route 53 ARC](p. 89)
- [Support cross-account for clusters in Route 53 ARC](p. 92)

**About routing control**

Routing control redirects traffic by using health checks in Amazon Route 53 that are configured with DNS records associated with the top-level resource of the cells in your recovery group, such as an Elastic Load Balancing load balancer. You can redirect traffic from one cell to another, for example, by updating a routing control state to Off (to stop traffic flow to one cell) and updating another routing control state to On (to start traffic flow to another). The process that changes the traffic flow is the Route 53 health check associated with the routing control, after it's updated to be healthy or unhealthy by the corresponding routing control state.

Routing controls support failover across any AWS service that has a DNS endpoint. You can update routing control states to fail over traffic for disaster recovery, or when you detect latency drops for your application, or other issues.
You can also configure safety rules in Route 53 ARC to make sure that rerouting traffic by using routing controls doesn't impair availability. For more information, see Creating safety rules in Route 53 ARC (p. 89).

It's important to note that routing controls are not themselves health checks that monitor the underlying health of endpoints. For example, unlike a Route 53 health check, a routing control doesn't monitor response times or TCP connection times. A routing control is a simple on-off switch that controls a health check. Typically, you change the state to redirect traffic, and that state change moves the traffic to go to a particular endpoint for an entire application stack, or prevents routing to the whole application stack. For example, in a simple scenario, when you change a routing control state from On to Off, it updates a Route 53 health check, which you've associated with a DNS failover record to move the traffic off of an endpoint.

To update a routing control state and reroute traffic, you must connect to one of your cluster endpoints in Route 53 ARC. If the endpoint that you try to connect to is unavailable, try changing the state with another cluster endpoint. Your process for changing routing control states should be prepared to try each endpoint in rotation, since cluster endpoints are cycled through available and unavailable states for regular maintenance and updates.

When you create routing controls, you configure your DNS records to associate routing control health checks with Route 53 DNS names that front each application replica. For example, to control traffic failovers across two load balancers, one in each of two Regions, you create two routing control health checks and associate them with two DNS records, for example, Alias records with failover routing policies, with the domain names of the respective load balancers.

You can also set up more complex traffic failover scenarios by using Route 53 ARC routing control together with Route 53 health checks and DNS record sets, using DNS records with weighted routing policies. To see a detailed example, see the section on failing over user traffic in the following AWS blog post: Building highly resilient applications using Amazon Route 53 Application Recovery Controller, Part 2: Multi-Region stack

A routing control in Route 53 ARC has several benefits over rerouting traffic with traditional health checks. For example:

- A routing control gives you a way to fail over an entire application stack. This is in contrast to failing over individual components of a stack, as Amazon EC2 instances do, based on resource-level health checks.
- A routing control gives you a safe, simple manual override that you can use to shift traffic to do maintenance or to recover from failures when internal monitors don't detect an issue.
- You can use a routing control together with safety rules to prevent common side effects that can happen with fully automated health check-based automation, such as failing over to standby infrastructure that isn't prepared for failover.

Creating routing control components in Route 53 ARC

This section explains how to create a cluster, routing controls, health checks, and control panels for working with routing control in Amazon Route 53 Application Recovery Controller.

Start by creating a cluster, to host your routing controls and the control panels that you use to group them. Then create routing controls and health checks so you can reroute traffic to fail over from one cell to another, so that traffic goes to your backup replica, for example.

Note that you are charged by the hour for each cluster that you create. You typically only need one cluster to host the routing controls and control panels for recovery control management for an
application. In addition, you can set up resource sharing by using AWS Resource Access Manager, so that one cluster can host routing controls and other Route 53 ARC resources owned by multiple AWS accounts. To learn about resource sharing in Route 53 ARC, see Amazon Route 53 Application Recovery Controller integrates with AWS Resource Access Manager to enable resource sharing. AWS RAM is a service that enables you to share resources with other AWS accounts or through AWS Organizations. For Route 53 ARC, you can share the cluster resource.

With AWS RAM, you share resources that you own by creating a resource share. A resource share specifies the resources to share, and the participants to share them with. Participants can include:

- Specific AWS accounts inside or outside of owner's organization in AWS Organizations
- An organizational unit inside its organization in AWS Organizations
- Its entire organization in AWS Organizations

For more information about AWS RAM, see the AWS RAM User Guide.

By using AWS Resource Access Manager to share cluster resources across accounts in Route 53 ARC, you can use one cluster to host control panels and routing controls owned by several different AWS accounts. When you opt to share a cluster, other AWS accounts that you specify can use the cluster to host their own control panels and routing controls, allowing more control and flexibility over routing capabilities across different teams.

AWS RAM is a service that helps AWS customers to securely share resources across AWS accounts. With AWS RAM, you can share resources within an organization or organizational units (OUs) in AWS Organizations, by using IAM roles and users. AWS RAM is a centralized and controlled way to share a cluster.

When you share a cluster, you can reduce the number of total clusters that your organization requires. With a shared cluster, you can allocate the total cost of running the cluster across different teams, to maximize the benefits of Route 53 ARC with lower cost. (Creating resources that are hosted in a cluster does not have additional costs, for the owner or for participants.) Sharing clusters across accounts can also ease the process of onboarding multiple applications to Route 53 ARC, especially if you have a large number of applications distributed across several accounts and operations teams.

To get started with cross-account sharing in Route 53 ARC, you create a resource share in AWS RAM. The resource share specifies participants who are authorized to share the cluster that your account owns. Then, participants can create resources, such as control panels and routing controls, in the cluster, by using the AWS Management Console or by running Route 53 ARC API operations using the AWS Command Line Interface or AWS SDKs.

This topic explains how to share resources that you own, and how to use resources that are shared with you.

Contents

- Prerequisites for sharing clusters (p. 93)
- Sharing a cluster (p. 93)
- Unsharing a shared cluster (p. 93)
- Identifying a shared cluster (p. 94)
- Responsibilities and permissions for shared clusters (p. 94)
- Billing costs (p. 95)
- Quotas (p. 95)
Prerequisites for sharing clusters

- To share a cluster, you must own it in your AWS account. This means that the resource must be allocated or provisioned in your account. You cannot share a cluster that has been shared with you.

- To share a cluster with your organization or an organizational unit in AWS Organizations, you must enable sharing with AWS Organizations. For more information, see Enable sharing with AWS Organizations in the AWS RAM User Guide.

Sharing a cluster

When you share a cluster that you own, the participants that you specify to share the cluster can create and host their own Route 53 ARC resources in the cluster.

To share a cluster, you must add it to a resource share. A resource share is an AWS RAM resource that lets you share your resources across AWS accounts. A resource share specifies the resources to share, and the participants they’re shared with. To share a cluster you can create a new resource share or add the resource to an existing resource share. To create a new resource share, you can use the AWS RAM console, or use AWS RAM API operations in the AWS Command Line Interface or AWS SDKs.

If you are part of an organization in AWS Organizations and sharing within your organization is enabled, participants in your organization are automatically granted access to the shared cluster. Otherwise, participants receive an invitation to join the resource share and are granted access to the shared cluster after accepting the invitation.

You can share a cluster that you own by using the AWS RAM console, or by using AWS RAM API operations with the AWS CLI or SDKs.

To share a cluster that you own by using the AWS RAM console

See Creating a resource share in the AWS RAM User Guide.

To share a cluster that you own by using the AWS CLI

Use the create-resource-share command.

Unsharing a shared cluster

When you unshare a cluster, the following applies to participants and owners:

- Current participant resources continue to exist in the unshared cluster.

- Participants can continue to update routing control states in the unshared cluster, to manage routing for application failover.

- Participants can no longer create new resources in the unshared cluster.

- If participants still have resources in an unshared cluster, the owner cannot delete the shared cluster.

To unshare a shared cluster that you own, remove it from the resource share. You can do this by using the AWS RAM console or by using AWS RAM API operations with the AWS CLI or SDKs.

To unshare a shared cluster that you own using the AWS RAM console

See Updating a resource share in the AWS RAM User Guide.
To unshare a shared cluster that you own using the AWS CLI

Use the disassociate-resource-share command.

Identifying a shared cluster

Owners and participants can identify shared clusters by viewing information in AWS RAM. They can also get information about shared resources by using the Route 53 ARC console and AWS CLI.

In general, to learn more about the resources that you've shared or that have been shared with you, see the information in the AWS Resource Access Manager User Guide:

- As an owner, you can view all resources that you are sharing with others by using AWS RAM. For more information, see Viewing your shared resources in AWS RAM.
- As a participant, you can view all resources shared with you by using AWS RAM. For more information, see Viewing your shared resources in AWS RAM.

As an owner, you can determine if you're sharing a cluster by viewing information in the AWS Management Console or by using the AWS Command Line Interface with Route 53 ARC API operations.

To identify if a cluster that you own is shared by using the console

In the AWS Management Console, on the details page for a cluster, see the Cluster sharing status.

To identify if a cluster that you own is shared by using the AWS CLI

Use the get-resource-policy command. If there is a resource policy for a cluster, the command returns information about the policy.

As a participant, when a cluster is shared with you, you typically must accept the share. In addition, the Owner field for the cluster contains the account of the cluster owner.

Responsibilities and permissions for shared clusters

Permissions for owners

When you share a cluster that you own with other AWS accounts, participants who are permitted to use the cluster can create control panels, routing controls, and other resources in the cluster.

As a cluster owner, you are responsible for creating, managing, and deleting clusters. You can't modify or delete resources created by participants, such as routing controls and safety rules. For example, you can't update a routing control created by a participant to change the routing control state.

However, you can view the details for routing controls that are created by participants in a cluster that you own. For example, you can view routing control states by calling a Route 53 ARC routing control API operation (p. 50), using the AWS Command Line Interface or AWS SDKs.

If you need to modify resources created by participants, they can set up a role in IAM with permission to access the resources, and add your account to the role.
Permissions for participants

In general, participants can create and use control panels, routing controls, safety rules, and health checks that they create in a cluster that is shared with them. They can only view, modify, or delete cluster resources in the shared cluster if they own the resources. For example, participants can create and delete safety rules for control panels that they have created.

The following restrictions apply for participants:

- Participants cannot view, modify, or delete control panels created by other accounts using a shared cluster.
- Participants cannot view, create, or modify routing controls, including routing control states, for resources created in a shared cluster by other accounts.
- Participants cannot create, modify, or view safety rules created by other accounts in a shared cluster.
- Participants cannot add resources in the default control panel in a shared cluster because it belongs to the cluster owner.

As noted, participants cannot create routing controls in the default control panel for a shared cluster, because the cluster owner owns the default control panel. However, the cluster owner can create a cross-account IAM role that provides permission to access the default control panel for the cluster. Then, the owner can grant a participant permissions to assume the role, so that the participant can access the default control panel to use it however the owner has specified through the role’s permissions.

Billing costs

The owner of a cluster in Route 53 ARC is billed for costs associated with the cluster. There are no additional costs, for cluster owners or for participants, for creating resources hosted in a cluster.

For detailed pricing information and examples, see Amazon Route 53 Application Recovery Controller Pricing and scroll down to Amazon Route 53 Application Recovery Controller.

Quotas

All resources created in a shared cluster—including resources created by all participants with access to the shared cluster—count toward quotas in effect for the cluster and other resources, such as routing controls.

For more information about quotas, see Quotas in Amazon Route 53 Application Recovery Controller (p. 145).

(p. 92). For pricing information, see Amazon Route 53 Application Recovery Controller Pricing and scroll down to Amazon Route 53.

To use routing controls to fail over traffic, you create routing control health checks that you associate with Amazon Route 53 DNS records for resources in your application. As an example, let’s say you have two cells, one that you've configured as the primary cell for your application, and the other that you've configured as the secondary, to fail over to.

To set up health checks for failover, do the following:

1. Create a routing control for each cell.
2. Create a health check for each routing control.
3. Create two DNS records, for example, two DNS failover records, and associate a health check with each one.

Another scenario when you might create a routing control is when you create a safety rule that is a gating rule. In this case, you don't associate health checks and DNS records with the routing control because you will use it as a gating routing control. For more information, see Creating safety rules in Route 53 ARC (p. 89).

The steps to create the components for routing control on the Route 53 ARC console are included in these sections. To learn about using recovery control configuration API operations with Route 53 ARC, see the Recovery control configuration actions (p. 49).

Topics
- Creating a cluster in Route 53 ARC (p. 85)
- Creating a routing control in Route 53 ARC (p. 85)
- Creating a routing control health check in Route 53 ARC (p. 86)
- Creating a control panel in Route 53 ARC (p. 87)

Creating a cluster in Route 53 ARC

You must create a cluster to host routing controls and control panels in Route 53 ARC.

A cluster is a set of redundant Regional endpoints against which you can execute API calls to update or get the state of one or more routing controls. A single cluster can host a number of routing controls.

Important
- Be aware that you are charged by the hour for each cluster that you create. One cluster can host a number of routing controls and control panels for recovery control management, typically enough for an application.

To create a cluster
2. Under Application Recovery Controller, choose Routing control page, and then choose Clusters.
3. Choose Create, and then enter a name for your cluster.
4. Choose Create cluster.

Creating a routing control in Route 53 ARC

Create a routing control for each cell that you want to route traffic to. For example, when you have an application with resources that you have siloed for recoverability, you might have a cell for each AWS Region, and nested cells for each Availability Zone within each Region. In this scenario, you would create a routing control for each cell and each nested cell.

When you create routing controls, keep in mind that routing control names must be unique within each control panel.

After you create routing controls to use for rerouting traffic, you associate each one with a health check, which allows you to route traffic to cells, based on the DNS records that you've associated with each one. If you're setting up a gating rule as a safety rule and creating a gating routing control, you don't add a health check to the routing control.

To create a routing control
2. Under Application Recovery Controller, choose Routing control.
3. On the Routing control page, choose Create, and then choose a Routing control.
4. Enter a name for your routing control, choose the cluster to add the control to, and choose to add it to an existing control panel, including using the default control panel. Or, create a new control panel.
5. If you choose to create a new control panel, choose a cluster to create the control panel on, and then enter a name for the panel.
6. Choose Create routing control.
7. Follow the steps to name and create the routing control.

Creating a routing control health check in Route 53 ARC

You associate a routing control health check with each routing control that you want to use for rerouting traffic. Then you configure each health check with a Amazon Route 53 DNS record, for example, a failover DNS record. Then you can reroute traffic in Amazon Route 53 Application Recovery Controller simply by updating the state of the associated routing control, to set it to On or Off.

Note
You can't edit an existing routing control health check to associate it with a different routing control.

To create a routing control health check

2. Under Application Recovery Controller, choose Routing control.
3. On the Routing control page, choose a routing control.
4. On the Routing control detail page, choose a Create health check.
5. Enter a name for the health check, and then choose Create.

Next, you create Route 53 DNS records, and associate your routing control health checks with each one. For example, let's assume you have two DNS failover records that you want to associate your routing control health checks with. For failover to work correctly, create two failover records: a primary and a secondary. For more information about configuring DNS failover records, see Health checking concepts.

After you've created the primary failover record, the values should be something like the following:

```
Name: myapp.yourdomain.com
Type: CNAME
Set Identifier: Primary
Failover: Primary
TTL: 0
Resource Records:
  Value: cell1.yourdomain.com
  Health Check ID: xxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
```

The secondary failover record values should be something like the following:

```
Name: myapp.yourdomain.com
Type: CNAME
Set Identifier: Secondary
Failover: Secondary
```
Creating a control panel

A control panel in Amazon Route 53 Application Recovery Controller lets you group together related routing controls. A control panel can have routing controls that represent a microservice within an application, an entire application itself, or a group of applications, depending on the scope of your failover. A benefit of grouping routing controls into a control panel is that you can use safety rules with a control panel to help safeguard traffic routing changes.

When you create a cluster, Route 53 ARC creates a default control panel. You can use the default control panel for your routing controls, or you can create one or more control panels to group your routing controls. Note that only ASCII characters are supported for control panel names.

The steps to create a control panel on the Route 53 ARC console are included in this section. For information about using recovery control configuration API operations with Route 53 ARC, see the Recovery control configuration actions (p. 49).

To create a control panel

2. Under Application Recovery Controller, choose Routing control.
3. On the Routing control page, choose Create, and then choose a Control panel.
4. Choose a cluster to create the control panel on, and then enter a name for the panel.
5. Choose Create control panel.

Viewing and updating routing control states in Route 53 ARC

This section describes how to view and update routing control states in Amazon Route 53 Application Recovery Controller. Routing controls are simple on-off switches that manage traffic flow to cells in your recovery group. Cells are typically AWS Regions, or sometimes Availability Zones, that includes your resources. When a routing control state is On, traffic flows to the cell that is controlled by that routing control.

You group routing controls into control panels, which are logical failover groupings. When you open a control panel on the console, for example, you can view all of the routing controls for a grouping at once, to see where traffic is flowing.

You can update a routing control state on the Route 53 ARC console or by using the Route 53 ARC API. We recommend that you update routing control states by using the API. First, Route 53 ARC offers extreme reliability with the API in the data plane to perform these actions. That's important when you're changing these states because routing state changes fail over across cells by rerouting application traffic.
In addition, by using the API, you can try connecting to different cluster endpoints in rotation, as needed, if a cluster endpoint that you try connecting to is unavailable.

You can update one routing control state, or you can update several routing control states at once. For example, you might want to set one routing control state to Off to stop traffic from flowing to one cell, such as an Availability Zone where an application is experiencing increased latency. At the same time, you might want to set another routing control state to On to start traffic flowing to another cell or Availability Zone. In this scenario, you can update both routing control states at the same time, so traffic continues to flow.

Topics

• Getting and updating routing control states using the Route 53 ARC API (recommended) (p. 88)
• Getting and updating routing control states in the AWS Management Console (p. 88)

Getting and updating routing control states using the Route 53 ARC API (recommended)

You can use the Amazon Route 53 Application Recovery Controller API to get or update routing control states by using an AWS CLI command or by using code that you have developed to use Route 53 ARC API operations with one of the AWS SDKs. We recommend using API operations, with the CLI or in code, for working with routing control states.

Route 53 ARC offers extreme reliability for failing over across cells by updating routing control states because routing controls are stored in a highly available cluster. Route 53 ARC ensures that at least three out of the five Regional cluster endpoints are always accessible to you to perform routing control state changes. To get or change a routing control state with the API, you connect to one of the Regional cluster endpoints. If the endpoint is unavailable, you can try connecting to another cluster endpoint.

You can view the list of Regional cluster endpoints for your cluster in the Route 53 console, or by using an API action, DescribeCluster. Your process for getting and changing routing control states should try each endpoint in rotation, as needed, since cluster endpoints are cycled through available and unavailable states for regular maintenance and updates.

We provide detailed information and code examples for using the Route 53 ARC API to get and update routing control states, and work with Regional cluster endpoints. See the following for more information:

• For code examples that explain how to rotate through Regional cluster endpoints to get and set routing control states, see Actions for Application Recovery Controller using AWS SDKs (p. 140).
• For information about using the AWS CLI to get and update routing control states, see List and update routing controls and states with the AWS CLI (p. 39).

Getting and updating routing control states in the AWS Management Console

You can get and update routing control states in the AWS Management Console. Be aware, though, that you can't choose different Regional cluster endpoints in the console. That is, there isn't a process for choosing and rotating through cluster endpoints in the console as you can do by using the Amazon Route 53 Application Recovery Controller API. In addition, the console is not highly available while the Route 53 ARC data plane offers extreme reliability. For these reasons, we recommend that you use the Route 53 ARC API to get and update routing control states for production operations.

For more recommendations about using Route 53 ARC for failover, see Best practices for Amazon Route 53 Application Recovery Controller (p. 44).
To view and update routing controls in the console, follow the steps in the following procedures.

**To get routing control states**

2. Under **Application Recovery Controller**, choose **Routing control**.
3. From the list, choose a control panel and view the routing controls.

**To update one or multiple routing control states**

2. Under **Application Recovery Controller**, choose **Routing control**.
3. Choose **Action**, and then choose **Change traffic routing**.
4. Update the states of one or more routing controls to be **Off** or **On**, depending on where you want traffic to flow or stop flowing for your application.
5. Enter **confirm** in the text box.
6. Choose **Update traffic routing**.

---

**Creating safety rules in Route 53 ARC**

When you work with several routing controls at the same time in Amazon Route 53 Application Recovery Controller, you might decide that you want safeguards in place to avoid unintended consequences. For example, you might want to prevent inadvertently turning off all the routing controls for an application, which would stop all traffic flow, resulting in a fail-open scenario. Or you might want to implement a master on-off switch to disable a set of routing controls, perhaps to prevent automation from rerouting traffic. To establish safeguards like these for routing control in Route 53 ARC, you create safety rules.

You configure safety rules with a combination of routing controls, rules, and other options that you specify. Each safety rule is associated with a single control panel, but a control panel can have more than one safety rule. When you create safety rules, keep in mind that safety rule names must be unique within each control panel.

**Topics**

- Types of safety rules (p. 89)
- Creating a safety rule on the console (p. 90)
- Editing or deleting a safety rule on the console (p. 91)
- Overriding safety rules to reroute traffic (p. 91)

**Types of safety rules**

There are two types of safety rules, **assertion rules** and **gating rules**, which you can use to safeguard failover in different ways.

**Assertion rule**

With an assertion rule, when you change one or a set of routing control states, Route 53 ARC enforces that the criteria that you set when you configured the rule is met, or else the routing control states aren't changed.

An example of when this is useful is to prevent a fail-open scenario, like a scenario where you stop traffic from going to one cell but do not start traffic flowing to another cell. To avoid this, an assertion rule makes sure that at least one routing control in a set of routing controls in a control
Creating a safety rule

Panel is on at any given time. This ensures that traffic flows to at least one Region or Availability Zone for an application.

To see an example AWS CLI command that creates an assertion rule to enforce this criteria, see Create safety rules in Get started with routing control by using the AWS CLI (p. 30).

For detailed information about the assertion rule API operation properties, see AssertionRule in the Routing Control API Reference Guide for Amazon Route 53 Application Recovery Controller.

Gating rule

With a gating rule, you can enforce an overall on-off switch over a set of routing controls so that whether those routing control states can be changed is enforced based on a set of criteria that you specify in the rule. The simplest criteria is whether a single routing control that you specify as the switch is set to ON or OFF.

To implement this, you create a gating routing control, to use as the overall switch, and target routing controls, to control traffic flow to different Regions or Availability Zones. Then, to prevent manual or automated state updates to the target routing controls that you've configured for the gating rule, you set the gating routing control state to Off. To allow updates, you set it to On.

To see an example AWS CLI command that creates a gating rule that implements this kind of overall switch, see Create safety rules in Get started with routing control by using the AWS CLI (p. 30).

For detailed information about the gating rule API operation properties, see GatingRule in the Routing Control API Reference Guide for Amazon Route 53 Application Recovery Controller.

Creating a safety rule on the console

The steps in this section explain how to create a safety rule on the Route 53 ARC console. The steps are similar whether you create an assertion rule or a gating rule. The differences are noted in the procedure.

To learn about using recovery and routing control API operations with Amazon Route 53 Application Recovery Controller, see Recovery control configuration actions (p. 49).

To create a safety rule

2. Under Application Recovery Controller, choose Routing control.
3. On the Routing control page, choose a control panel.
4. On the control panel details page, choose Action, and then choose Add safety rule.
5. Choose a type of rule to add: Assertion rule or Gating rule.
6. Choose a name and, optionally, change the wait period.
7. Specify the configuration options for the safety rule.

   • For an assertion rule, specify the asserted routing controls.
   • For a gating rule, specify the gating routing control and target routing controls.

For both rules, specify the rule configuration by choosing the type and threshold, and whether the rule is inverted.

Note

To learn more about specifying an assertion rule, see the information provided for AssertionRule operation in the Routing Control API Reference Guide for Amazon Route 53 Application Recovery Controller. To learn more about specifying a gating rule, see the information provided for the GatingRule operation in the Routing Control API Reference Guide for Amazon Route 53 Application Recovery Controller.
8. Choose Create.

Editing or deleting a safety rule on the console

The steps in this section explain how to edit or delete a safety rule on the Route 53 ARC console. You can make only limited edits to a safety rule, to change the name or update the wait period. To make more extensive changes, delete and recreate the safety rule.

To learn about using API operations with Amazon Route 53 Application Recovery Controller, see the Common actions that you can use with Amazon Route 53 Application Recovery Controller (p. 47).

To delete a safety rule
2. Under Application Recovery Controller, choose Routing control.
3. On the Routing control page, choose a control panel.
4. On the control panel details page, choose a safety rule, and then choose Delete or Edit.

Overriding safety rules to reroute traffic

There are scenarios when you might want to bypass the routing control safeguards that are enforced with safety rules that you've configured. For example, you might want to fail over quickly for disaster recovery, and one or more safety rules might be unexpectedly preventing you from updating a routing control state to reroute traffic. In a "break glass" scenario like this, you can override one or more safety rules to change a routing control state and fail over your application.

You can bypass safety rules when you update a routing control state (or multiple routing control states) by using the update-routing-control-state or update-routing-control-states AWS CLI command with the safety-rules-to-override parameter. Specify the parameter with the Amazon Resource Name (ARN) of the safety rule that you want to override, or specify a comma-separated list of ARNs to override two or more safety rules.

When a safety rule blocks a routing control state update, the error message includes the ARN of the rule that blocked the update. So you can make a note of the ARN, and then specify it in a routing control state CLI command with the safety rule override parameter.

**Note**

Because more than one safety rule might be in place for the routing controls that you're updating, you could run the CLI command to update your routing control state with one safety rule override but get an error that another safety rule is blocking the update. Continue to add safety rule ARNs to the list of rules to override in the update command, separated by commas, until the update command completes successfully.

To learn more about using the SafetyRulesToOverride property with the API and SDKs, see UpdateRoutingControlState.

The following are two examples of CLI commands to override safety rules to update routing control states.

**Override one safety rule**

```
aws route53-recovery-cluster --region us-west-2 update-routing-control-state \ 
   --routing-control-arn \ 
   arn:aws:route53-recovery-control::11112223333:controlpanel/0123456bbbbb0123456bbbbb0123456/routingcontrol/ \ 
   abcdefg1234567 
```
Support cross-account clusters in Route 53 ARC

Amazon Route 53 Application Recovery Controller integrates with AWS Resource Access Manager to enable resource sharing. AWS RAM is a service that enables you to share resources with other AWS accounts or through AWS Organizations. For Route 53 ARC, you can share the cluster resource.

With AWS RAM, you share resources that you own by creating a resource share. A resource share specifies the resources to share, and the participants to share them with. Participants can include:

- Specific AWS accounts inside or outside of owner's organization in AWS Organizations
- An organizational unit inside its organization in AWS Organizations
- Its entire organization in AWS Organizations

For more information about AWS RAM, see the AWS RAM User Guide.

By using AWS Resource Access Manager to share cluster resources across accounts in Route 53 ARC, you can use one cluster to host control panels and routing controls owned by several different AWS accounts. When you opt to share a cluster, other AWS accounts that you specify can use the cluster to host their own control panels and routing controls, allowing more control and flexibility over routing capabilities across different teams.

AWS RAM is a service that helps AWS customers to securely share resources across AWS accounts. With AWS RAM, you can share resources within an organization or organizational units (OUs) in AWS Organizations, by using IAM roles and users. AWS RAM is a centralized and controlled way to share a cluster.

When you share a cluster, you can reduce the number of total clusters that your organization requires. With a shared cluster, you can allocate the total cost of running the cluster across different teams, to maximize the benefits of Route 53 ARC with lower cost. (Creating resources that are hosted in a cluster does not have additional costs, for the owner or for participants.) Sharing clusters across accounts can also ease the process of onboarding multiple applications to Route 53 ARC, especially if you have a large number of applications distributed across several accounts and operations teams.

To get started with cross-account sharing in Route 53 ARC, you create a resource share in AWS RAM. The resource share specifies participants who are authorized to share the cluster that your account
owns. Then, participants can create resources, such as control panels and routing controls, in the cluster, by using the AWS Management Console or by running Route 53 ARC API operations using the AWS Command Line Interface or AWS SDKs.

This topic explains how to share resources that you own, and how to use resources that are shared with you.

Contents
• Prerequisites for sharing clusters (p. 93)
• Sharing a cluster (p. 93)
• Unsharing a shared cluster (p. 93)
• Identifying a shared cluster (p. 94)
• Responsibilities and permissions for shared clusters (p. 94)
• Billing costs (p. 95)
• Quotas (p. 95)

Prerequisites for sharing clusters

• To share a cluster, you must own it in your AWS account. This means that the resource must be allocated or provisioned in your account. You cannot share a cluster that has been shared with you.
• To share a cluster with your organization or an organizational unit in AWS Organizations, you must enable sharing with AWS Organizations. For more information, see Enable sharing with AWS Organizations in the AWS RAM User Guide.

Sharing a cluster

When you share a cluster that you own, the participants that you specify to share the cluster can create and host their own Route 53 ARC resources in the cluster.

To share a cluster, you must add it to a resource share. A resource share is an AWS RAM resource that lets you share your resources across AWS accounts. A resource share specifies the resources to share, and the participants they’re shared with. To share a cluster you can create a new resource share or add the resource to an existing resource share. To create a new resource share, you can use the AWS RAM console, or use AWS RAM API operations in the AWS Command Line Interface or AWS SDKs.

If you are part of an organization in AWS Organizations and sharing within your organization is enabled, participants in your organization are automatically granted access to the shared cluster. Otherwise, participants receive an invitation to join the resource share and are granted access to the shared cluster after accepting the invitation.

You can share a cluster that you own by using the AWS RAM console, or by using AWS RAM API operations with the AWS CLI or SDKs.

To share a cluster that you own by using the AWS RAM console

See Creating a resource share in the AWS RAM User Guide.

To share a cluster that you own by using the AWS CLI

Use the create-resource-share command.

Unsharing a shared cluster

When you unshare a cluster, the following applies to participants and owners:
• Current participant resources continue to exist in the unshared cluster.
• Participants can continue to update routing control states in the unshared cluster, to manage routing for application failover.
• Participants can no longer create new resources in the unshared cluster.
• If participants still have resources in an unshared cluster, the owner cannot delete the shared cluster.

To unshare a shared cluster that you own, remove it from the resource share. You can do this by using the AWS RAM console or by using AWS RAM API operations with the AWS CLI or SDKs.

To unshare a shared cluster that you own using the AWS RAM console
See Updating a resource share in the AWS RAM User Guide.

To unshare a shared cluster that you own using the AWS CLI
Use the disassociate-resource-share command.

Identifying a shared cluster

Owners and participants can identify shared clusters by viewing information in AWS RAM. They can also get information about shared resources by using the Route 53 ARC console and AWS CLI.

In general, to learn more about the resources that you've shared or that have been shared with you, see the information in the AWS Resource Access Manager User Guide:

• As an owner, you can view all resources that you are sharing with others by using AWS RAM. For more information, see Viewing your shared resources in AWS RAM.
• As a participant, you can view all resources shared with you by using AWS RAM. For more information, see Viewing your shared resources in AWS RAM.

As an owner, you can determine if you're sharing a cluster by viewing information in the AWS Management Console or by using the AWS Command Line Interface with Route 53 ARC API operations.

To identify if a cluster that you own is shared by using the console
In the AWS Management Console, on the details page for a cluster, see the Cluster sharing status.

To identify if a cluster that you own is shared by using the AWS CLI
Use the get-resource-policy command. If there is a resource policy for a cluster, the command returns information about the policy.

As a participant, when a cluster is shared with you, you typically must accept the share. In addition, the Owner field for the cluster contains the account of the cluster owner.

Responsibilities and permissions for shared clusters

Permissions for owners

When you share a cluster that you own with other AWS accounts, participants who are permitted to use the cluster can create control panels, routing controls, and other resources in the cluster.

As a cluster owner, you are responsible for creating, managing, and deleting clusters. You can't modify or delete resources created by participants, such as routing controls and safety rules. For example, you can't update a routing control created by a participant to change the routing control state.
However, you can view the details for routing controls that are created by participants in a cluster that you own. For example, you can view routing control states by calling a [Route 53 ARC routing control API operation](p. 50), using the AWS Command Line Interface or AWS SDKs.

If you need to modify resources created by participants, they can set up a role in IAM with permission to access the resources, and add your account to the role.

### Permissions for participants

In general, participants can create and use control panels, routing controls, safety rules, and health checks that they create in a cluster that is shared with them. They can only view, modify, or delete cluster resources in the shared cluster if they own the resources. For example, participants can create and delete safety rules for control panels that they have created.

The following restrictions apply for participants:

- Participants cannot view, modify, or delete control panels created by other accounts using a shared cluster.
- Participants cannot view, create, or modify routing controls, including routing control states, for resources created in a shared cluster by other accounts.
- Participants cannot create, modify, or view safety rules created by other accounts in a shared cluster.
- Participants cannot add resources in the default control panel in a shared cluster because it belongs to the cluster owner.

As noted, participants cannot create routing controls in the default control panel for a shared cluster, because the cluster owner owns the default control panel. However, the cluster owner can create a cross-account IAM role that provides permission to access the default control panel for the cluster. Then, the owner can grant a participant permissions to assume the role, so that the participant can access the default control panel to use it however the owner has specified through the role's permissions.

### Billing costs

The owner of a cluster in Route 53 ARC is billed for costs associated with the cluster. There are no additional costs, for cluster owners or for participants, for creating resources hosted in a cluster.

For detailed pricing information and examples, see [Amazon Route 53 Application Recovery Controller Pricing](p. 145) and scroll down to Amazon Route 53 Application Recovery Controller.

### Quotas

All resources created in a shared cluster—including resources created by all participants with access to the shared cluster—count toward quotas in effect for the cluster and other resources, such as routing controls.

For more information about quotas, see [Quotas in Amazon Route 53 Application Recovery Controller](p. 145).
Logging and monitoring in Amazon Route 53 Application Recovery Controller

You can use Amazon CloudWatch and AWS CloudTrail for monitoring in Amazon Route 53 Application Recovery Controller, to analyze traffic patterns and help troubleshoot issues with features, such as readiness checks and clusters.

**Note**
You must view CloudWatch metrics and logs for Route 53 ARC in the US West (Oregon) Region, both in the console or when using the AWS CLI. When you use the AWS CLI, specify the US West (Oregon) Region for your command by including the following parameter: `--region us-west-2`.

**Topics**
- Using Amazon CloudWatch with Route 53 ARC (p. 96)
- Logging Route 53 ARC API calls using AWS CloudTrail (p. 99)
- Using Route 53 ARC with Amazon EventBridge (p. 104)

Using Amazon CloudWatch with Route 53 ARC

Amazon Route 53 Application Recovery Controller publishes data points to Amazon CloudWatch for your readiness checks. CloudWatch enables you to retrieve statistics about those data points as an ordered set of time-series data, known as *metrics*. Think of a metric as a variable to monitor, and the data points as the values of that variable over time. For example, you can monitor traffic through an AWS Region over a specified time period. Each data point has an associated time stamp and an optional unit of measurement.

You can use metrics to verify that your system is performing as expected. For example, you can create a CloudWatch alarm to monitor a specified metric and initiate an action (such as sending a notification to an email address) if the metric goes outside what you consider an acceptable range.

For more information, see the [Amazon CloudWatch User Guide](https://docs.aws.amazon.com/AmazonCloudWatch/latest/userguide/).

**Topics**
- Route 53 ARC metrics (p. 96)
- Statistics for Route 53 ARC metrics (p. 97)
- View CloudWatch metrics in Route 53 ARC (p. 97)

Route 53 ARC metrics

The AWS/Route53RecoveryReadiness namespace includes the following metrics.

<table>
<thead>
<tr>
<th>Metric</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReadinessChecks</td>
<td>Represents the number of readiness checks processed by Route 53 ARC. The metric can be dimensioned by its states, listed below.</td>
</tr>
<tr>
<td>Metric</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td></td>
<td><strong>Unit:</strong> Count.</td>
</tr>
<tr>
<td></td>
<td><strong>Reporting criteria:</strong> There is a nonzero value.</td>
</tr>
<tr>
<td></td>
<td><strong>Statistics:</strong> The only useful statistic is <strong>Sum</strong>.</td>
</tr>
<tr>
<td></td>
<td><strong>Dimensions</strong></td>
</tr>
<tr>
<td></td>
<td>• READY</td>
</tr>
<tr>
<td></td>
<td>• NOT_READY</td>
</tr>
<tr>
<td></td>
<td>• NOT_AUTHORIZED</td>
</tr>
<tr>
<td></td>
<td>• UNKNOWN</td>
</tr>
</tbody>
</table>

**Resources**

- Represents the number of resources processed by Route 53 ARC, which can be dimensioned by their resource identifier, as defined by the API.

- **Unit:** Count.
- **Reporting criteria:** There is a nonzero value.
- **Statistics:** The only useful statistic is **Sum**.
- **Dimensions**
  - **ResourceSetType:** These are the resource types, filtered by the number of resources per given type evaluated by Route 53 ARC

For example: `AWS::CloudWatch::Alarm`

---

**Statistics for Route 53 ARC metrics**

CloudWatch provides statistics based on the metric data points published by Route 53 ARC. Statistics are aggregations of metric data over a specified period of time. When you request statistics, the returned data stream is identified by the metric name and dimension. A dimension is a name/value pair that uniquely identifies a metric.

The following are examples of metric/dimension combinations that you might find useful:

- View the number of readiness checks evaluated for readiness by Route 53 ARC.
- View the total number of resources for a given resource set type evaluated by Route 53 ARC.

**View CloudWatch metrics in Route 53 ARC**

You can view the CloudWatch metrics for Route 53 ARC using the CloudWatch console or the AWS CLI. In the console, metrics are displayed as monitoring graphs.

You must view CloudWatch metrics for Route 53 ARC in the US West (Oregon) Region, both in the console or when using the AWS CLI. When you use the AWS CLI, specify the US West (Oregon) Region for your command by including the following parameter: `--region us-west-2`.

**To view metrics using the CloudWatch console**

2. In the navigation pane, choose **Metrics**.
3. Select the **Route53RecoveryReadiness** namespace.
4. (Optional) To view a metric across all dimensions, type its name in the search field.

**To view metrics using the AWS CLI**

Use the following `list-metrics` command to list the available metrics:

```
aws cloudwatch list-metrics --namespace AWS/Route53RecoveryReadiness --region us-west-2
```

**To get the statistics for a metric using the AWS CLI**

Use the following `get-metric-statistics` command to get statistics for a specified metric and dimension. Note that CloudWatch treats each unique combination of dimensions as a separate metric. You can't retrieve statistics using combinations of dimensions that were not specifically published. You must specify the same dimensions that were used when the metrics were created.

The following example lists the total readiness checks evaluated, per minute, for an account in Route 53 ARC.

```
aws cloudwatch get-metric-statistics --namespace AWS/Route53RecoveryReadiness --metric-name ReadinessChecks --region us-west-2 --statistics Sum --period 60 --dimensions Name=State,Value=READY --start-time 2021-07-03T01:00:00Z --end-time 2021-07-03T01:20:00Z
```

The following is example output from the command:

```json
{
    "Label": "ReadinessChecks",
    "Datapoints": [
        {
            "Timestamp": "2021-07-08T18:00:00Z",
            "Sum": 1.0,
            "Unit": "Count"
        },
        {
            "Timestamp": "2021-07-08T18:04:00Z",
            "Sum": 1.0,
            "Unit": "Count"
        },
        {
            "Timestamp": "2021-07-08T18:01:00Z",
            "Sum": 1.0,
            "Unit": "Count"
        },
        {
            "Timestamp": "2021-07-08T18:02:00Z",
            "Sum": 1.0,
            "Unit": "Count"
        },
        {
            "Timestamp": "2021-07-08T18:03:00Z",
            "Sum": 1.0,
            "Unit": "Count"
        }
    ]
}
```
Logging Route 53 ARC API calls using AWS CloudTrail

Amazon Route 53 Application Recovery Controller is integrated with AWS CloudTrail, a service that provides a record of actions taken by a user, role, or an AWS service in Route 53 ARC. CloudTrail captures all API calls for Route 53 ARC as events. The calls captured include calls from the Route 53 ARC console and code calls to the Route 53 ARC API operations.

If you create a trail, you can enable continuous delivery of CloudTrail events to an Amazon S3 bucket, including events for Route 53 ARC. If you don't configure a trail, you can still view the most recent events in the CloudTrail console in Event history.

Using the information collected by CloudTrail, you can determine the request that was made to Route 53 ARC, the IP address from which the request was made, who made the request, when it was made, and additional details.

To learn more about CloudTrail, see the AWS CloudTrail User Guide.

Route 53 ARC information in CloudTrail

CloudTrail is enabled on your AWS account when you create the account. When activity occurs in Route 53 ARC, that activity is recorded in a CloudTrail event along with other AWS service events in Event history. You can view, search, and download recent events in your AWS account. For more information, see Viewing events with CloudTrail Event history.

For an ongoing record of events in your AWS account, including events for Route 53 ARC, create a trail. A trail enables CloudTrail to deliver log files to an Amazon S3 bucket. By default, when you create a trail in the console, the trail applies to all AWS Regions. The trail logs events from all Regions in the AWS partition and delivers the log files to the Amazon S3 bucket that you specify. Additionally, you can configure other AWS services to further analyze and act upon the event data collected in CloudTrail logs. For more information, see the following:

- Overview for creating a trail
- CloudTrail supported services and integrations
- Configuring Amazon SNS notifications for CloudTrail
- Receiving CloudTrail log files from multiple regions and Receiving CloudTrail log files from multiple accounts

All Route 53 ARC actions are logged by CloudTrail and are documented in the Recovery Readiness API Reference Guide for Amazon Route 53 Application Recovery Controller, Recovery Control Configuration API Reference Guide for Amazon Route 53 Application Recovery Controller, and Routing Control API Reference Guide for Amazon Route 53 Application Recovery Controller. For example, calls to the CreateCluster, UpdateRoutingControlState and CreateRecoveryGroup actions generate entries in the CloudTrail log files.

Every event or log entry contains information about who generated the request. The identity information helps you determine the following:

- Whether the request was made with root or AWS Identity and Access Management (IAM) user credentials.
- Whether the request was made with temporary security credentials for a role or federated user.
- Whether the request was made by another AWS service.

For more information, see the CloudTrail userIdentity element.
Viewing Route 53 ARC events in event history

CloudTrail lets you view recent events in Event history. To view events for Route 53 ARC API requests, you must choose US West (Oregon) in the Region selector at the top of the console. For more information, see Viewing events with CloudTrail event history in the AWS CloudTrail User Guide.

Understanding Route 53 ARC log file entries

A trail is a configuration that enables delivery of events as log files to an Amazon S3 bucket that you specify. CloudTrail log files contain one or more log entries. An event represents a single request from any source and includes information about the requested action, the date and time of the action, request parameters, and so on. CloudTrail log files aren't an ordered stack trace of the public API calls, so they don't appear in any specific order.

The following example shows a CloudTrail log entry that demonstrates the CreateCluster action for control configuration.

```json
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "A1B2C3D4E5F6G7EXAMPLE",
        "arn": "arn:aws:iam::111122223333:user/smithj",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "A1B2C3D4E5F6G7EXAMPLE",
                "arn": "arn:aws:iam::111122223333:role/smithj",
                "accountId": "111122223333",
                "userName": "smithj"
            },
            "webIdFederationData": {},
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "2021-06-30T04:44:41Z"
            }
        },
        "sessionIssuer": {
            "type": "Role",
            "principalId": "A1B2C3D4E5F6G7EXAMPLE",
            "arn": "arn:aws:iam::111122223333:role/smithj",
            "accountId": "111122223333",
            "userName": "smithj"
        },
        "webIdFederationData": {},
        "attributes": {
            "mfaAuthenticated": "false",
            "creationDate": "2021-06-30T04:44:41Z"
        }
    },
    "eventTime": "2021-06-30T04:45:46Z",
    "eventSource": "route53-recovery-control-config.amazonaws.com",
    "eventName": "CreateCluster",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.50",
    "userAgent": "aws-cli/2.0.0 Python/3.8.2 Darwin/19.6.0 botocore/2.0.0dev7",
    "requestParameters": {
        "ClientToken": "12345abcdef-1234-5678-abcd-12345abcdef",
        "ClusterName": "XYZCluster"
    },
    "responseElements": {
        "Cluster": {
            "Arn": "arn:aws:route53-recovery-control::012345678901:cluster/abc123456-aa11-bb22-cc33-abc123456",
            "ClusterArn": "arn:aws:route53-recovery-control::012345678901:cluster/abc123456-aa11-bb22-cc33-abc123456",
            "Name": "XYZCluster",
            "Status": "PENDING"
        }
    },
    "requestID": "6090509a-5a97-4be6-8e6a-7d73example",
    "eventID": "9cab44ef-0777-41e6-838f-f249example"
}
```
The following example shows a CloudTrail log entry that demonstrates the UpdateRoutingControlState action for routing control.

```json
{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "A1B2C3D4E5F6G7EXAMPLE",
    "arn": "arn:aws:sts::111122223333:assumed-role/admin smithj",
    "accountId": "111122223333",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "sessionContext": {
      "sessionIssuer": {
        "type": "Role",
        "principalId": "A1B2C3D4E5F6G7EXAMPLE",
        "arn": "arn:aws:iam::111122223333:role/admin",
        "accountId": "111122223333",
        "userName": "admin"
      },
      "webIdFederationData": {},
      "attributes": {
        "mfaAuthenticated": "false",
        "creationDate": "2021-06-30T04:44:41Z"
      }
    }
  },
  "eventTime": "2021-06-30T04:45:46Z",
  "eventSource": "route53-recovery-control-config.amazonaws.com",
  "eventName": "UpdateRoutingControl",
  "awsRegion": "us-west-2",
  "sourceIPAddress": "192.0.2.50",
  "userAgent": "aws-cli/2.0.0 Python/3.8.2 Darwin/19.6.0 botocore/2.0.0dev7",
  "requestParameters": {
    "RoutingControlName": "XYZRoutingControl3",
    "RoutingControlArn": "arn:aws:route53-recovery-control::012345678:controlpanel/0123456bbbbbbb0123456bbbbbb0123456/routingcontrol/abcdefg1234567"
  },
  "responseElements": {
    "RoutingControl": {
      "ControlPanelArn": "arn:aws:route53-recovery-control::012345678:controlpanel/0123456bbbbbbb0123456bbbbbb0123456",
      "Name": "XYZRoutingControl3",
      "Status": "DEPLOYED",
      "RoutingControlArn": "arn:aws:route53-recovery-control::012345678:controlpanel/0123456bbbbbbb0123456bbbbbb0123456/routingcontrol/abcdefg1234567"
    }
  },
  "requestID": "6090509a-5a97-4be6-8e6a-7d73example",
  "eventID": "9cab44ef-0777-41e6-838f-f249example",
  "readOnly": false,
  "eventType": "AwsApiCall",
  "managementEvent": true,
  "eventCategory": "Management",
  "recipientAccountId": "111122223333"
}
```
The following example shows a CloudTrail log entry that demonstrates the CreateRecoveryGroup action for readiness check.

```json
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "A1B2C3D4E5F6G7EXAMPLE",
        "arn": "arn:aws:iam::111122223333:role/admin",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "type": "AssumedRole",
            "principalId": "A1B2C3D4E5F6G7EXAMPLE",
            "arn": "arn:aws:iam::111122223333:role/admin",
            "accountId": "111122223333",
            "userName": "EXAMPLENAME"
        },
        "webIdFederationData": {},
        "attributes": {
            "mfaAuthenticated": "false",
            "creationDate": "2021-07-06T17:38:05Z"
        }
    },
    "eventTime": "2021-07-06T18:08:03Z",
    "eventSource": "route53-recovery-readiness.amazonaws.com",
    "eventName": "CreateRecoveryGroup",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.50",
    "userAgent": "Boto3/1.17.101 Python/3.8.10 Linux/4.14.231-180.360.amzn2.x86_64 exec-env/AWS_Lambda_python3.8 Botocore/1.20.102",
    "requestParameters": {
        "recoveryGroupName": "MyRecoveryGroup"
    },
    "responseElements": {
        "Access-Control-Expose-Headers": "x-amzn-errortype,x-amzn-requestid,x-amzn-errormessage,x-amzn-trace-id,x-amzn-requestid,x-amz-apigw-id,date",
        "cells": [],
        "recoveryGroupName": "MyRecoveryGroup",
        "recoveryGroupArn": "arn:aws:route53-recovery-readiness::111122223333:recovery-group/MyRecoveryGroup",
        "tags": "****",
        "requestID": "fd42dcf7-6446-41e9-b400-d096example",
        "eventId": "4b5c42df-1174-46c8-be99-d67example",
        "readOnly": false,
        "eventType": "AwsApiCall",
        "managementEvent": true,
        "eventCategory": "Management",
        "recipientAccountId": "111122223333"
    }
}
```

The following example shows a CloudTrail log entry that demonstrates the ListManagedResources action for zonal shift.

```json
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "A1B2C3D4E5F6G7EXAMPLE",
        "arn": "arn:aws:iam::111122223333:role/admin",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
```
The following example shows a CloudTrail log entry that demonstrates the **StartZonalShift** action with a conflict exception for zonal shift.

```json
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "A1B2C3D4E5F6G7EXAMPLE",
        "arn": "arn:aws:iam::111122223333:role/admin",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "type": "Role",
            "principalId": "AROA33L3W36EXAMPLE",
            "arn": "arn:aws:iam::111122223333:role/admin",
            "accountId": "111122223333",
            "userName": "EXAMPLENAME"
        },
        "webIdFederationData": {},
        "attributes": {
            "creationDate": "2022-11-14T16:01:51Z",
            "mfaAuthenticated": "false"
        }
    },
    "eventTime": "2022-11-14T16:10:38Z",
    "eventSource": "arc-zonal-shift.amazonaws.com",
    "eventName": "StartZonalShift",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.50",
    "userAgent": "Boto3/1.17.101 Python/3.8.10 Linux/4.14.231-180.360.amzn2.x86_64 exec- env/AWS_Lambda_python3.8 BotoCore/1.20.102",
    "requestParameters": null,
    "responseElements": null,
    "requestID": "VGXG4ZUE7UZTVCMTJGIAF_EXAMPLE",
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333"
}
```

The following example shows a CloudTrail log entry that demonstrates the **StartZonalShift** action with a conflict exception for zonal shift.
"awsRegion": "us-west-2",
"sourceIPAddress": "192.0.2.50",
"userAgent": "Boto3/1.17.101 Python/3.8.10 Linux/4.14.231-180.360.amzn2.x86_64 exec-sdk/aws_lambda_python3.8 Botocore/1.20.102",
"errorCode": "ConflictException",
"requestParameters": {
  "awayFrom": "usw2-az1",
  "expiresIn": "2m",
  "comment": "HIDDEN_FOR_SECURITY_REASONS"
},
"responseElements": null,
"requestID": "OP4OYXZ54HUPMIPGWH_EXAMPLE",
"eventID": "0bca6660-e999-43a5-9008-EXAMPLE",
"readOnly": false,
"eventType": "AwsApiCall",
"managementEvent": true,
"recipientAccountId": "111122223333"
}

Using Route 53 ARC with Amazon EventBridge

Using Amazon EventBridge, you can set up event-driven rules that monitor your Amazon Route 53 Application Recovery Controller resources and initiate target actions that use other AWS services. For example, you can set a rule for sending out email notifications by signaling an Amazon SNS topic whenever a readiness check status changes from READY to NOT READY.

**Note**

Route 53 ARC only publishes EventBridge events in the US West (Oregon) (us-west-2) AWS Region. To receive EventBridge events for Route 53 ARC, create EventBridge rules in the US West (Oregon) Region.

You can create rules in Amazon EventBridge to act on any of the following Route 53 ARC events:

- **Recovery group readiness.** The event specifies if recovery group readiness status changes, for example, from READY to NOT READY.
- **Cell readiness.** The event specifies if cell readiness status changes, for example, from READY to NOT READY.
- **Readiness check readiness.** The event specifies if readiness check status changes, for example, from READY to NOT READY.

To capture specific Route 53 ARC events that you're interested in, define event-specific patterns that EventBridge can use to detect the events. Event patterns have the same structure as the events that they match. The pattern quotes the fields that you want to match and provides the values that you're looking for.

Events are emitted on a best effort basis. They're delivered from Route 53 ARC to EventBridge in near real-time under normal operational circumstances. However, situations can arise that might delay or prevent delivery of an event.

For information about how EventBridge rules work with event patterns, see [Events and Event Patterns in EventBridge](#).
Monitor a Route 53 ARC resource with EventBridge

With EventBridge, you can create rules that define actions to take when Route 53 ARC emits events for its resources. For example, you can create a rule that sends you an email message whenever the readiness status of a specific recovery group changes.

The EventBridge console has a Pre-defined pattern option for building Route 53 ARC event patterns. If you select this option in the EventBridge console when you create a rule, you can build a Route 53 ARC event pattern quickly. You only need to select the event fields and values. As you make selections, the console builds and displays the event pattern. Alternatively, you can manually edit the event pattern that you build and can save it as a custom pattern. The console also displays a detailed Sample Event that you can copy and paste to the event pattern that you’re building.

If you prefer to type or copy and paste an event pattern into the EventBridge console, you can select to use the Custom pattern option in the console. By doing this, you don’t need to go through the steps of selecting fields and values. This topic includes examples of both Route 53 ARC event-matching patterns (p. 106) and Route 53 ARC events (p. 107) that you can use.

To create a rule for a resource event

1. Open the Amazon EventBridge console at https://console.aws.amazon.com/events/.
2. Choose Create rule.
3. Enter a Name for the rule, and, optionally, a description.
4. Under Define pattern choose Event pattern.
5. Under Event matching pattern, choose Pre-defined pattern by service.
   
   Note
   
   If you already have text for an event pattern and don’t need the EventBridge console to build it for you, you can select Customer pattern. You can then either manually enter, or copy and paste, text into the Event Pattern box. Select Save, and then skip to Step 12.

6. For Service provider, select AWS.
7. For Service name, select Route 53 ARC.
8. For Event type, select Status Change.
9. For Environment select Any environment or Specific environment(s).
   
   • If you select Specific environment(s), you can choose one or more environments from the drop-down list. EventBridge adds all of environments that you select inside the EnvironmentName[ ] list in the detail section of the event pattern. Then, your rule filters all events to include only the specific environments that you choose.
   • If you select Any environment, then no environments are added to your event pattern. Because of this, your rule doesn’t filter any of the Route 53 ARC events based on environment.
10. The Select event bus section defaults to AWS default event bus. Leave the default selected and confirm that Enable the rule on the selected event bus is toggled on.
11. Under Select targets, choose the target action to take when a resource state change event is received from Route 53 ARC.

For example, you can use an Amazon Simple Notification Service (SNS) topic to send an email or text message when an event occurs. To do this, you need to create an Amazon SNS topic using the Amazon SNS console. To learn more, see Using Amazon SNS for user notifications.

Important

Some target actions might require the use of other services and incur additional charges, such as the Amazon SNS or Lambda service. For more information about AWS pricing, see https://aws.amazon.com/pricing/. Some services are part of the AWS Free Usage Tier. If you’re a new customer, you can try using these services for free. For more information, see https://aws.amazon.com/free/.
12. (Optional) Choose **Add target** to specify an additional target action for the event rule.

13. Choose **Create**.

### Example Route 53 ARC event patterns

Event patterns have the same structure as the events they match. The pattern quotes the fields that you want to match and provides the values that you're looking for.

- **Select all events from Route 53 ARC readiness check.**

```json
{
    "source": [
        "aws.route53-recovery-readiness"
    ]
}
```

- **Select only events related to cells.**

```json
{
    "source": [
        "aws.route53-recovery-readiness"
    ],
    "detail-type": [
        "Route 53 Application Recovery Controller cell readiness status change"
    ]
}
```

- **Select only events related to a specific cell called MyExampleCell.**

```json
{
    "source": [
        "aws.route53-recovery-readiness"
    ],
    "detail-type": [
        "Route 53 Application Recovery Controller cell readiness status change"
    ],
    "resources": [
        "arn:aws:route53-recovery-readiness::111122223333:cell/MyExampleCell"
    ]
}
```

- **Select only events when any recovery group, cell, or readiness check status becomes NOT READY.**

```json
{
    "source": ["aws.route53-recovery-readiness"],
    "detail-type": {"new-state": {"readiness-status": ["NOT READY"]}}
}
```

- **Select only events when any recovery group, cell, or readiness check becomes anything except READY**

```json
{
    "source": ["aws.route53-recovery-readiness"],
    "detail-type": {"readiness-status": ["NOT READY"]}
}
```
Example Route 53 ARC events

The following is an example Route 53 ARC event for a recovery group readiness status change:

```
{
    "version": "0",
    "account": "111122223333",
    "detail-type": "Route 53 Application Recovery Controller recovery group readiness status change",
    "source": "route53-recovery-readiness.amazonaws.com",
    "time": "2020-11-03T00:31:54Z",
    "id": "1234a678-1b23-c123-12fd3f456e78",
    "region": "us-west-2",
    "resources": [
        "arn:aws:route53-recovery-readiness::111122223333:recovery-group/BillingApp"
    ],
    "detail": {
        "recovery-group-name": "BillingApp",
        "previous-state": {
            "readiness-status": "READY|NOT_READY|UNKNOWN|NOT_AUTHORIZED"
        },
        "new-state": {
            "readiness-status": "READY|NOT_READY|UNKNOWN|NOT_AUTHORIZED"
        }
    }
}
```

The following is an example Route 53 ARC event for a cell readiness status change:

```
{
    "version": "0",
    "account": "111122223333",
    "detail-type": "Route 53 Application Recovery Controller cell readiness status change",
    "source": "route53-recovery-readiness.amazonaws.com",
    "time": "2020-11-03T00:31:54Z",
    "id": "1234a678-1b23-c123-12fd3f456e78",
    "region": "us-west-2",
    "resources": [
        "arn:aws:route53-recovery-readiness::111122223333:cell/PDXCell"
    ],
    "detail": {
        "cell-name": "PDXCell",
        "previous-state": {
            "readiness-status": "READY|NOT_READY|UNKNOWN|NOT_AUTHORIZED"
        },
        "new-state": {
            "readiness-status": "READY|NOT_READY|UNKNOWN|NOT_AUTHORIZED"
        }
    }
}
```
The following is an example Route 53 ARC event for a *readiness check status change*:

```
{
  "version": "0",
  "account": "111122223333",
  "detail-type": "Route 53 Application Recovery Controller readiness check status change",
  "source": "route53-recovery-readiness.amazonaws.com",
  "time": "2020-11-03T00:31:54Z",
  "id": "1234a678-1b23-c123-12fd3f456e78",
  "region": "us-west-2",
  "resources": [
    "arn:aws:route53-recovery-readiness::111122223333:readiness-check/UserTableReadinessCheck"
  ],
  "detail": {
    "readiness-check-name": "UserTableReadinessCheck",
    "previous-state": {
      "readiness-status": "READY|NOT_READY|UNKNOWN|NOT_AUTHORIZED"
    },
    "new-state": {
      "readiness-status": "READY|NOT_READY|UNKNOWN|NOT_AUTHORIZED"
    }
  }
}
```
Security in Amazon Route 53 Application Recovery Controller

Cloud security at AWS is the highest priority. As an AWS customer, you benefit from data centers and network architectures that are built to meet the requirements of the most security-sensitive organizations.

Security is a shared responsibility between AWS and you. The shared responsibility model describes this as security of the cloud and security in the cloud:

- **Security of the cloud** – AWS is responsible for protecting the infrastructure that runs AWS services in the AWS Cloud. AWS also provides you with services that you can use securely. Third-party auditors regularly test and verify the effectiveness of our security as part of the AWS Compliance Programs. To learn about the compliance programs that apply to Amazon Route 53 Application Recovery Controller, see AWS Services in Scope by Compliance Program.

- **Security in the cloud** – Your responsibility is determined by the AWS service that you use. You are also responsible for other factors including the sensitivity of your data, your company’s requirements, and applicable laws and regulations.

This documentation helps you understand how to apply the shared responsibility model when using Route 53 ARC. The following topics show you how to configure Route 53 ARC to meet your security and compliance objectives. You also learn how to use other AWS services that help you to monitor and secure your Route 53 ARC resources.

**Topics**

- Data protection in Amazon Route 53 Application Recovery Controller (p. 109)
- Identity and Access Management for Amazon Route 53 Application Recovery Controller (p. 110)
- Logging and monitoring in Amazon Route 53 Application Recovery Controller (p. 137)
- Compliance validation for Amazon Route 53 Application Recovery Controller (p. 138)
- Resilience in Amazon Route 53 Application Recovery Controller (p. 139)
- Infrastructure security in Amazon Route 53 Application Recovery Controller (p. 139)

Data protection in Amazon Route 53 Application Recovery Controller

The AWS shared responsibility model applies to data protection in Amazon Route 53 Application Recovery Controller. As described in this model, AWS is responsible for protecting the global infrastructure that runs all of the AWS Cloud. You are responsible for maintaining control over your content that is hosted on this infrastructure. This content includes the security configuration and management tasks for the AWS services that you use. For more information about data privacy, see the Data Privacy FAQ. For information about data protection in Europe, see the AWS Shared Responsibility Model and GDPR blog post on the AWS Security Blog.

For data protection purposes, we recommend that you protect AWS account credentials and set up individual users with AWS IAM Identity Center or AWS Identity and Access Management (IAM). That way, each user is given only the permissions necessary to fulfill their job duties. We also recommend that you secure your data in the following ways:
• Use multi-factor authentication (MFA) with each account.
• Use SSL/TLS to communicate with AWS resources. We require TLS 1.2 and recommend TLS 1.3.
• Set up API and user activity logging with AWS CloudTrail.
• Use AWS encryption solutions, along with all default security controls within AWS services.
• Use advanced managed security services such as Amazon Macie, which assists in discovering and securing sensitive data that is stored in Amazon S3.
• If you require FIPS 140-2 validated cryptographic modules when accessing AWS through a command line interface or an API, use a FIPS endpoint. For more information about the available FIPS endpoints, see Federal Information Processing Standard (FIPS) 140-2.

We strongly recommend that you never put confidential or sensitive information, such as your customers’ email addresses, into tags or free-form text fields such as a Name field. This includes when you work with Route 53 ARC or other AWS services using the console, API, AWS CLI, or AWS SDKs. Any data that you enter into tags or free-form text fields used for names may be used for billing or diagnostic logs. If you provide a URL to an external server, we strongly recommend that you do not include credentials information in the URL to validate your request to that server.

Encryption at rest

Customer configuration information is stored in service-owned Amazon DynamoDB global tables, and is encrypted at rest.

Datasets that contain the status of cells in a Route 53 ARC cluster are written to an Amazon EBS volume for backup. Route 53 ARC uses the default Amazon EBS encryption while the data is at rest.

Encryption in transit

Customer requests and responses—for Route 53 ARC configuration, readiness status queries, cell state updates, and so on—are encrypted during transport throughout the service by using TLS.

Identity and Access Management for Amazon Route 53 Application Recovery Controller

AWS Identity and Access Management (IAM) is an AWS service that helps an administrator securely control access to AWS resources. IAM administrators control who can be authenticated (signed in) and authorized (have permissions) to use Route 53 ARC resources. IAM is an AWS service that you can use with no additional charge.

Contents
• Audience (p. 111)
• Authenticating with identities (p. 111)
• Managing access using policies (p. 113)
• How Amazon Route 53 Application Recovery Controller works with IAM (p. 115)
• IAM and permissions for zonal shift (p. 120)
• Identity-based policy examples for Amazon Route 53 Application Recovery Controller (p. 121)
• Using service-linked roles for Route 53 ARC (p. 126)
Audience

How you use AWS Identity and Access Management (IAM) differs, depending on the work that you do in Route 53 ARC.

**Service user** – If you use the Route 53 ARC service to do your job, then your administrator provides you with the credentials and permissions that you need. As you use more Route 53 ARC features to do your work, you might need additional permissions. Understanding how access is managed can help you request the right permissions from your administrator. If you cannot access a feature in Route 53 ARC, see Troubleshooting Amazon Route 53 Application Recovery Controller identity and access (p. 136).

**Service administrator** – If you’re in charge of Route 53 ARC resources at your company, you probably have full access to Route 53 ARC. It’s your job to determine which Route 53 ARC features and resources your service users should access. You must then submit requests to your IAM administrator to change the permissions of your service users. Review the information on this page to understand the basic concepts of IAM. To learn more about how your company can use IAM with Route 53 ARC, see How Amazon Route 53 Application Recovery Controller works with IAM (p. 115).

**IAM administrator** – If you’re an IAM administrator, you might want to learn details about how you can write policies to manage access to Route 53 ARC. To view example Route 53 ARC identity-based policies that you can use in IAM, see Identity-based policy examples for Amazon Route 53 Application Recovery Controller (p. 121).

Authenticating with identities

Authentication is how you sign in to AWS using your identity credentials. You must be authenticated (signed in to AWS) as the AWS account root user, as an IAM user, or by assuming an IAM role.

You can sign in to AWS as a federated identity by using credentials provided through an identity source. AWS IAM Identity Center (IAM Identity Center) users, your company’s single sign-on authentication, and your Google or Facebook credentials are examples of federated identities. When you sign in as a federated identity, your administrator previously set up identity federation using IAM roles. When you access AWS by using federation, you are indirectly assuming a role.

Depending on the type of user you are, you can sign in to the AWS Management Console or the AWS access portal. For more information about signing in to AWS, see How to sign in to your AWS account in the AWS Sign-In User Guide.

If you access AWS programmatically, AWS provides a software development kit (SDK) and a command line interface (CLI) to cryptographically sign your requests by using your credentials. If you don’t use AWS tools, you must sign requests yourself. For more information about using the recommended method to sign requests yourself, see Signing AWS API requests in the IAM User Guide.

Regardless of the authentication method that you use, you might be required to provide additional security information. For example, AWS recommends that you use multi-factor authentication (MFA) to increase the security of your account. To learn more, see Multi-factor authentication in the AWS IAM Identity Center User Guide and Using multi-factor authentication (MFA) in AWS in the IAM User Guide.

AWS account root user

When you create an AWS account, you begin with one sign-in identity that has complete access to all AWS services and resources in the account. This identity is called the AWS account root user and is accessed by signing in with the email address and password that you used to create the account. We
strongly recommend that you don’t use the root user for your everyday tasks. Safeguard your root user credentials and use them to perform the tasks that only the root user can perform. For the complete list of tasks that require you to sign in as the root user, see Tasks that require root user credentials in the IAM User Guide.

Federated identity

As a best practice, require human users, including users that require administrator access, to use federation with an identity provider to access AWS services by using temporary credentials.

A federated identity is a user from your enterprise user directory, a web identity provider, the AWS Directory Service, the Identity Center directory, or any user that accesses AWS services by using credentials provided through an identity source. When federated identities access AWS accounts, they assume roles, and the roles provide temporary credentials.

For centralized access management, we recommend that you use AWS IAM Identity Center. You can create users and groups in IAM Identity Center, or you can connect and synchronize to a set of users and groups in your own identity source for use across all your AWS accounts and applications. For information about IAM Identity Center, see What is IAM Identity Center? in the AWS IAM Identity Center User Guide.

IAM users and groups

An IAM user is an identity within your AWS account that has specific permissions for a single person or application. Where possible, we recommend relying on temporary credentials instead of creating IAM users who have long-term credentials such as passwords and access keys. However, if you have specific use cases that require long-term credentials with IAM users, we recommend that you rotate access keys. For more information, see Rotate access keys regularly for use cases that require long-term credentials in the IAM User Guide.

An IAM group is an identity that specifies a collection of IAM users. You can’t sign in as a group. You can use groups to specify permissions for multiple users at a time. Groups make permissions easier to manage for large sets of users. For example, you could have a group named IAMAdmins and give that group permissions to administer IAM resources.

Users are different from roles. A user is uniquely associated with one person or application, but a role is intended to be assumable by anyone who needs it. Users have permanent long-term credentials, but roles provide temporary credentials. To learn more, see When to create an IAM user (instead of a role) in the IAM User Guide.

IAM roles

An IAM role is an identity within your AWS account that has specific permissions. It is similar to an IAM user, but is not associated with a specific person. You can temporarily assume an IAM role in the AWS Management Console by switching roles. You can assume a role by calling an AWS CLI or AWS API operation or by using a custom URL. For more information about methods for using roles, see Using IAM roles in the IAM User Guide.

IAM roles with temporary credentials are useful in the following situations:

- **Federated user access** – To assign permissions to a federated identity, you create a role and define permissions for the role. When a federated identity authenticates, the identity is associated with the role and is granted the permissions that are defined by the role. For information about roles for federation, see Creating a role for a third-party Identity Provider in the IAM User Guide. If you use IAM Identity Center, you configure a permission set. To control what your identities can access after they authenticate, IAM Identity Center correlates the permission set to a role in IAM. For information about permissions sets, see Permission sets in the AWS IAM Identity Center User Guide.
Managing access using policies

You control access in AWS by creating policies and attaching them to AWS identities or resources. A policy is an object in AWS that, when associated with an identity or resource, defines their permissions. AWS evaluates these policies when a principal (user, root user, or role session) makes a request. Permissions in the policies determine whether the request is allowed or denied. Most policies are stored in AWS as JSON documents. For more information about the structure and contents of JSON policy documents, see Overview of JSON policies in the IAM User Guide.

Administrators can use AWS JSON policies to specify who has access to what. That is, which principal can perform actions on what resources, and under what conditions.

By default, users and roles have no permissions. To grant users permission to perform actions on the resources that they need, an IAM administrator can create IAM policies. The administrator can then add the IAM policies to roles, and users can assume the roles.

IAM policies define permissions for an action regardless of the method that you use to perform the operation. For example, suppose that you have a policy that allows the iam:GetRole action. A user with that policy can get role information from the AWS Management Console, the AWS CLI, or the AWS API.
Identity-based policies

Identity-based policies are JSON permissions policy documents that you can attach to an identity, such as an IAM user, group of users, or role. These policies control what actions users and roles can perform, on which resources, and under what conditions. To learn how to create an identity-based policy, see Creating IAM policies in the IAM User Guide.

Identity-based policies can be further categorized as inline policies or managed policies. Inline policies are embedded directly into a single user, group, or role. Managed policies are standalone policies that you can attach to multiple users, groups, and roles in your AWS account. Managed policies include AWS managed policies and customer managed policies. To learn how to choose between a managed policy or an inline policy, see Choosing between managed policies and inline policies in the IAM User Guide.

Resource-based policies

Resource-based policies are JSON policy documents that you attach to a resource. Examples of resource-based policies are IAM role trust policies and Amazon S3 bucket policies. In services that support resource-based policies, service administrators can use them to control access to a specific resource. For the resource where the policy is attached, the policy defines what actions a specified principal can perform on that resource and under what conditions. You must specify a principal in a resource-based policy. Principals can include accounts, users, roles, federated users, or AWS services.

Resource-based policies are inline policies that are located in that service. You can’t use AWS managed policies from IAM in a resource-based policy.

Access control lists (ACLs)

Access control lists (ACLs) control which principals (account members, users, or roles) have permissions to access a resource. ACLs are similar to resource-based policies, although they do not use the JSON policy document format.

Amazon S3, AWS WAF, and Amazon VPC are examples of services that support ACLs. To learn more about ACLs, see Access control list (ACL) overview in the Amazon Simple Storage Service Developer Guide.

Other policy types

AWS supports additional, less-common policy types. These policy types can set the maximum permissions granted to you by the more common policy types.

- Permissions boundaries – A permissions boundary is an advanced feature in which you set the maximum permissions that an identity-based policy can grant to an IAM entity (IAM user or role). You can set a permissions boundary for an entity. The resulting permissions are the intersection of an entity’s identity-based policies and its permissions boundaries. Resource-based policies that specify the user or role in the Principal field are not limited by the permissions boundary. An explicit deny in any of these policies overrides the allow. For more information about permissions boundaries, see Permissions boundaries for IAM entities in the IAM User Guide.

- Service control policies (SCPs) – SCPs are JSON policies that specify the maximum permissions for an organization or organizational unit (OU) in AWS Organizations. AWS Organizations is a service for grouping and centrally managing multiple AWS accounts that your business owns. If you enable all features in an organization, then you can apply service control policies (SCPs) to any or all of your accounts. The SCP limits permissions for entities in member accounts, including each AWS account root user. For more information about Organizations and SCPs, see How SCPs work in the AWS Organizations User Guide.

- Session policies – Session policies are advanced policies that you pass as a parameter when you programmatically create a temporary session for a role or federated user. The resulting session’s
permissions are the intersection of the user or role’s identity-based policies and the session policies. Permissions can also come from a resource-based policy. An explicit deny in any of these policies overrides the allow. For more information, see Session policies in the IAM User Guide.

Multiple policy types

When multiple types of policies apply to a request, the resulting permissions are more complicated to understand. To learn how AWS determines whether to allow a request when multiple policy types are involved, see Policy evaluation logic in the IAM User Guide.

How Amazon Route 53 Application Recovery Controller works with IAM

Before you use IAM to manage access to Route 53 ARC, learn what IAM features are available to use with Route 53 ARC.

IAM features you can use with Amazon Route 53 Application Recovery Controller

<table>
<thead>
<tr>
<th>IAM feature</th>
<th>Route 53 ARC support</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identity-based policies (p. 115)</td>
<td>Yes</td>
</tr>
<tr>
<td>Resource-based policies (p. 116)</td>
<td>No</td>
</tr>
<tr>
<td>Policy actions (p. 116)</td>
<td>Yes</td>
</tr>
<tr>
<td>Policy resources (p. 117)</td>
<td>Yes</td>
</tr>
<tr>
<td>Policy condition keys (p. 117)</td>
<td>Yes</td>
</tr>
<tr>
<td>ACLs (p. 118)</td>
<td>No</td>
</tr>
<tr>
<td>ABAC (tags in policies) (p. 118)</td>
<td>Partial</td>
</tr>
<tr>
<td>Temporary credentials (p. 119)</td>
<td>Yes</td>
</tr>
<tr>
<td>Principal permissions (p. 119)</td>
<td>Yes</td>
</tr>
<tr>
<td>Service roles (p. 120)</td>
<td>No</td>
</tr>
<tr>
<td>Service-linked roles (p. 120)</td>
<td>Yes</td>
</tr>
</tbody>
</table>

To get a high-level view of how Route 53 ARC and other AWS services work with most IAM features, see AWS services that work with IAM in the IAM User Guide.

Identity-based policies for Route 53 ARC

| Supports identity-based policies   | Yes |

Identity-based policies are JSON permissions policy documents that you can attach to an identity, such as an IAM user, group of users, or role. These policies control what actions users and roles can perform,
on which resources, and under what conditions. To learn how to create an identity-based policy, see
Creating IAM policies in the IAM User Guide.

With IAM identity-based policies, you can specify allowed or denied actions and resources as well as the
conditions under which actions are allowed or denied. You can't specify the principal in an identity-based
policy because it applies to the user or role to which it is attached. To learn about all of the elements
that you can use in a JSON policy, see IAM JSON policy elements reference in the IAM User Guide.

To view examples of Route 53 ARC identity-based policies, see Identity-based policy examples for
Amazon Route 53 Application Recovery Controller (p. 121).

### Resource-based policies within Route 53 ARC

<table>
<thead>
<tr>
<th>Supports resource-based policies</th>
<th>No</th>
</tr>
</thead>
</table>

Resource-based policies are JSON policy documents that you attach to a resource. Examples of resource-based
policies are IAM role trust policies and Amazon S3 bucket policies. In services that support
resource-based policies, service administrators can use them to control access to a specific resource.

### Policy actions for Route 53 ARC

<table>
<thead>
<tr>
<th>Supports policy actions</th>
<th>Yes</th>
</tr>
</thead>
</table>

Administrators can use AWS JSON policies to specify who has access to what. That is, which principal can
perform actions on what resources, and under what conditions.

The Action element of a JSON policy describes the actions that you can use to allow or deny access in a
policy. Policy actions usually have the same name as the associated AWS API operation. There are some
exceptions, such as permission-only actions that don't have a matching API operation. There are also
some operations that require multiple actions in a policy. These additional actions are called dependent
actions.

Include actions in a policy to grant permissions to perform the associated operation.

To see a list of Route 53 ARC actions, see Actions defined by Amazon Route 53 Application Recovery
Controller in the Service Authorization Reference.

Policy actions in Route 53 ARC use the following prefixes before the action, depending on the API that
you're working with:

route53-recovery-readiness
route53-recovery-control-config
route53-recovery-cluster
arc-zonal-shift

To specify multiple actions in a single statement, separate them with commas.

"Action": [
    "route53-recovery-readiness:action1",
    "route53-recovery-readiness:action2"
]

You can specify multiple actions using wildcards (*). For example, to specify all actions that begin with
the word Describe, include the following action:
"Action": "route53-recovery-readiness:Describe*

To view examples of Route 53 ARC identity-based policies, see Identity-based policy examples for Amazon Route 53 Application Recovery Controller (p. 121).

**Policy resources for Route 53 ARC**

| Supports policy resources | Yes |

Administrators can use AWS JSON policies to specify who has access to what. That is, which principal can perform actions on what resources, and under what conditions.

The Resource JSON policy element specifies the object or objects to which the action applies. Statements must include either a Resource or a NotResource element. As a best practice, specify a resource using its Amazon Resource Name (ARN). You can do this for actions that support a specific resource type, known as resource-level permissions.

For actions that don't support resource-level permissions, such as listing operations, use a wildcard (*) to indicate that the statement applies to all resources.

"Resource": "*"

To see a list of resource types and their ARNs, and the actions that you can specify with the ARN of each resource, see the following topics in the Service Authorization Reference:

- Resource types defined by Amazon Route 53 Recovery Cluster
- Resource types defined by Amazon Route 53 Recovery Controls
- Resource types defined by Amazon Route 53 Recovery Readiness
- Resource types defined by Amazon Route 53 Zonal Shift

To see the actions and resources that you can use with a condition key, see the following topics in the Service Authorization Reference:

- Actions defined by Amazon Route 53 Recovery Cluster
- Actions defined by Amazon Route 53 Recovery Controls
- Actions defined by Amazon Route 53 Recovery Readiness
- Actions defined by Amazon Route 53 Zonal Shift

To view examples of Route 53 ARC identity-based policies, see Identity-based policy examples for Amazon Route 53 Application Recovery Controller (p. 121).

**Policy condition keys for Route 53 ARC**

| Supports service-specific policy condition keys | Yes |

Administrators can use AWS JSON policies to specify who has access to what. That is, which principal can perform actions on what resources, and under what conditions.
The **Condition element** (or **Condition block**) lets you specify conditions in which a statement is in effect. The Condition element is optional. You can create conditional expressions that use condition operators, such as equals or less than, to match the condition in the policy with values in the request.

If you specify multiple Condition elements in a statement, or multiple keys in a single Condition element, AWS evaluates them using a logical AND operation. If you specify multiple values for a single condition key, AWS evaluates the condition using a logical OR operation. All of the conditions must be met before the statement's permissions are granted.

You can also use placeholder variables when you specify conditions. For example, you can grant an IAM user permission to access a resource only if it is tagged with their IAM user name. For more information, see [IAM policy elements: variables and tags](https://docs.aws.amazon.com/IAM/latest/UserGuide/id委组织部 variable-terms.html) in the **IAM User Guide**.

AWS supports global condition keys and service-specific condition keys. To see all AWS global condition keys, see [AWS global condition context keys](https://docs.aws.amazon.com/IAM/latest/UserGuide/id委组织部 global-condition-key-context-key.html) in the **IAM User Guide**.

To see a list of Route 53 ARC condition keys, see [Condition keys for Amazon Route 53 Application Recovery Controller](https://docs.aws.amazon.com/AmazonRoute53/latest/Route53ApplicationRecoveryController/ConditionKeysForAmazonRoute53ApplicationRecoveryController.html) in the **Service Authorization Reference**. To learn with which actions and resources you can use a condition key, see [Actions defined by Amazon Route 53 Application Recovery Controller](https://docs.aws.amazon.com/AmazonRoute53/latest/Route53ApplicationRecoveryController/ActionsDefinedByAmazonRoute53ApplicationRecoveryController.html).

To see a list of Route 53 ARC condition keys, see the following topics in the **Service Authorization Reference**:

- [Condition keys for Amazon Route 53 Recovery Cluster](https://docs.aws.amazon.com/AmazonRoute53/latest/Route53ApplicationRecoveryController/ConditionKeysForAmazonRoute53RecoveryCluster.html)
- [Condition keys for Amazon Route 53 Recovery Controls](https://docs.aws.amazon.com/AmazonRoute53/latest/Route53ApplicationRecoveryController/ConditionKeysForAmazonRoute53RecoveryControls.html)
- [Condition keys for Amazon Route 53 Recovery Readiness](https://docs.aws.amazon.com/AmazonRoute53/latest/Route53ApplicationRecoveryController/ConditionKeysForAmazonRoute53RecoveryReadiness.html)
- [Condition keys for Amazon Route 53 Zonal Shift](https://docs.aws.amazon.com/AmazonRoute53/latest/Route53ApplicationRecoveryController/ConditionKeysForAmazonRoute53ZonalShift.html)

To see the actions and resources that you can use with a condition key, see the following topics in the **Service Authorization Reference**:

- [Actions defined by Amazon Route 53 Recovery Cluster](https://docs.aws.amazon.com/AmazonRoute53/latest/Route53ApplicationRecoveryController/ActionsDefinedByAmazonRoute53RecoveryCluster.html)
- [Actions defined by Amazon Route 53 Recovery Controls](https://docs.aws.amazon.com/AmazonRoute53/latest/Route53ApplicationRecoveryController/ActionsDefinedByAmazonRoute53RecoveryControls.html)
- [Actions defined by Amazon Route 53 Recovery Readiness](https://docs.aws.amazon.com/AmazonRoute53/latest/Route53ApplicationRecoveryController/ActionsDefinedByAmazonRoute53RecoveryReadiness.html)
- [Actions defined by Amazon Route 53 Zonal Shift](https://docs.aws.amazon.com/AmazonRoute53/latest/Route53ApplicationRecoveryController/ActionsDefinedByAmazonRoute53ZonalShift.html)

To view examples of Route 53 ARC identity-based policies, see [Identity-based policy examples for Amazon Route 53 Application Recovery Controller](https://docs.aws.amazon.com/AmazonRoute53/latest/Route53ApplicationRecoveryController/IdentityBasedPolicyExamplesForAmazonRoute53ApplicationRecoveryController.html) (p. 121).

### Access control lists (ACLs) in Route 53 ARC

<table>
<thead>
<tr>
<th>Supports ACLs</th>
<th>No</th>
</tr>
</thead>
</table>

Access control lists (ACLs) control which principals (account members, users, or roles) have permissions to access a resource. ACLs are similar to resource-based policies, although they do not use the JSON policy document format.

### Attribute-based access control (ABAC) with Route 53 ARC

<table>
<thead>
<tr>
<th>Supports ABAC (tags in policies)</th>
<th>Partial</th>
</tr>
</thead>
</table>
Attribute-based access control (ABAC) is an authorization strategy that defines permissions based on attributes. In AWS, these attributes are called tags. You can attach tags to IAM entities (users or roles) and to many AWS resources. Tagging entities and resources is the first step of ABAC. Then you design ABAC policies to allow operations when the principal’s tag matches the tag on the resource that they are trying to access.

ABAC is helpful in environments that are growing rapidly and helps with situations where policy management becomes cumbersome.

To control access based on tags, you provide tag information in the condition element of a policy using the `aws:ResourceTag/key-name`, `aws:RequestTag/key-name`, or `aws:TagKeys` condition keys.

If a service supports all three condition keys for every resource type, then the value is Yes for the service. If a service supports all three condition keys for only some resource types, then the value is Partial.

For more information about ABAC, see What is ABAC? in the IAM User Guide. To view a tutorial with steps for setting up ABAC, see Use attribute-based access control (ABAC) in the IAM User Guide.

Route 53 ARC includes the following partial support for ABAC:

- Recovery Readiness (readiness check) and Recovery Controls support ABAC.
- Zonal shift supports ABAC for managed resources that are registered in Route 53 ARC for zonal shift. For more information about ABAC for Network Load Balancer and Application Load Balancer managed resources, see ABAC with Elastic Load Balancing in the Elastic Load Balancing User Guide.
- Recovery Cluster (routing control) does not support ABAC.

### Using temporary credentials with Route 53 ARC

<table>
<thead>
<tr>
<th>Supports temporary credentials</th>
<th>Yes</th>
</tr>
</thead>
</table>

Some AWS services don't work when you sign in using temporary credentials. For additional information, including which AWS services work with temporary credentials, see AWS services that work with IAM in the IAM User Guide.

You are using temporary credentials if you sign in to the AWS Management Console using any method except a user name and password. For example, when you access AWS using your company's single sign-on (SSO) link, that process automatically creates temporary credentials. You also automatically create temporary credentials when you sign in to the console as a user and then switch roles. For more information about switching roles, see Switching to a role (console) in the IAM User Guide.

You can manually create temporary credentials using the AWS CLI or AWS API. You can then use those temporary credentials to access AWS. AWS recommends that you dynamically generate temporary credentials instead of using long-term access keys. For more information, see Temporary security credentials in IAM.

### Cross-service principal permissions for Route 53 ARC

<table>
<thead>
<tr>
<th>Supports principal permissions</th>
<th>Yes</th>
</tr>
</thead>
</table>

When you use an IAM entity (user or role) to perform actions in AWS, you are considered a principal. Policies grant permissions to a principal. When you use some services, you might perform an action that then triggers another action in a different service. In this case, you must have permissions to perform both actions.
To see whether an action requires additional dependent actions in a policy, see the following topics in the Service Authorization Reference:

- Amazon Route 53 Recovery Cluster
- Amazon Route 53 Recovery Controls
- Amazon Route 53 Recovery Readiness
- Amazon Route 53 Zonal Shift

Service roles for Route 53 ARC

| Supports service roles | No |

A service role is an IAM role that a service assumes to perform actions on your behalf. An IAM administrator can create, modify, and delete a service role from within IAM. For more information, see Creating a role to delegate permissions to an AWS service in the IAM User Guide.

Service-linked roles for Route 53 ARC

| Supports service-linked roles | Yes |

A service-linked role is a type of service role that is linked to an AWS service. The service can assume the role to perform an action on your behalf. Service-linked roles appear in your AWS account and are owned by the service. An IAM administrator can view, but not edit the permissions for service-linked roles.

For details about creating or managing Route 53 ARC service-linked roles, see Using service-linked roles for Route 53 ARC (p. 126).

For details about creating or managing service-linked roles, see AWS services that work with IAM. Find a service in the table that includes a Yes in the Service-linked role column. Choose the Yes link to view the service-linked role documentation for that service.

IAM and permissions for zonal shift

This section describes how permissions work for the zonal shift feature provided by Amazon Route 53 Application Recovery Controller, especially if you work with the feature from another AWS service, such as Elastic Load Balancing. To learn about how Route 53 ARC features works with IAM and permissions in general, review the information in How Amazon Route 53 Application Recovery Controller works with IAM (p. 115).

In addition to the overall permissions information that applies to Route 53 ARC, the following applies to zonal shift for IAM and permissions:

- Make sure that you have the required permissions for working with zonal shift in Route 53 ARC. For more information, see Route 53 ARC console access (p. 122) and Route 53 ARC actions access (p. 123).
- You do not need to add additional Elastic Load Balancing permissions with IAM to work with zonal shifts for managed load balancer resources in your account in Route 53 ARC.
- An AWS managed policy that provides full access for Elastic Load Balancing includes permissions for working with zonal shifts. If you use AWS managed policies for Elastic Load Balancing access, you do not need additional permissions in IAM for zonal shift to start zonal shifts for load balancers or work...
Identity-based policy examples for Amazon Route 53 Application Recovery Controller

By default, users and roles don’t have permission to create or modify Route 53 ARC resources. They also can’t perform tasks by using the AWS Management Console, AWS Command Line Interface (AWS CLI), or AWS API. To grant users permission to perform actions on the resources that they need, an IAM administrator can create IAM policies. The administrator can then add the IAM policies to roles, and users can assume the roles.

To learn how to create an IAM identity-based policy by using these example JSON policy documents, see Creating IAM policies in the IAM User Guide.

For details about actions and resource types defined by Route 53 ARC, including the format of the ARNs for each of the resource types, see Actions, resources, and condition keys for Amazon Route 53 Application Recovery Controller in the Service Authorization Reference.

Policy best practices

Identity-based policies determine whether someone can create, access, or delete Route 53 ARC resources in your account. These actions can incur costs for your AWS account. When you create or edit identity-based policies, follow these guidelines and recommendations:

• **Get started with AWS managed policies and move toward least-privilege permissions** – To get started granting permissions to your users and workloads, use the AWS managed policies that grant permissions for many common use cases. They are available in your AWS account. We recommend that you reduce permissions further by defining AWS customer managed policies that are specific to your use cases. For more information, see AWS managed policies or AWS managed policies for job functions in the IAM User Guide.

• **Apply least-privilege permissions** – When you set permissions with IAM policies, grant only the permissions required to perform a task. You do this by defining the actions that can be taken on specific resources under specific conditions, also known as least-privilege permissions. For more information about using IAM to apply permissions, see Policies and permissions in IAM in the IAM User Guide.

• **Use conditions in IAM policies to further restrict access** – You can add a condition to your policies to limit access to actions and resources. For example, you can write a policy condition to specify that all requests must be sent using SSL. You can also use conditions to grant access to service actions if they are used through a specific AWS service, such as AWS CloudFormation. For more information, see IAM JSON policy elements: Condition in the IAM User Guide.

• **Use IAM Access Analyzer to validate your IAM policies to ensure secure and functional permissions** – IAM Access Analyzer validates new and existing policies so that the policies adhere to the IAM policy language (JSON) and IAM best practices. IAM Access Analyzer provides more than 100 policy checks and actionable recommendations to help you author secure and functional policies. For more information, see IAM Access Analyzer policy validation in the IAM User Guide.

• **Require multi-factor authentication (MFA)** – If you have a scenario that requires IAM users or a root user in your AWS account, turn on MFA for additional security. To require MFA when API operations are
called, add MFA conditions to your policies. For more information, see Configuring MFA-protected API access in the IAM User Guide.

For more information about best practices in IAM, see Security best practices in IAM in the IAM User Guide.

Example: Route 53 ARC console access

To access the Amazon Route 53 Application Recovery Controller console, you must have a minimum set of permissions. These permissions must allow you to list and view details about the Route 53 ARC resources in your AWS account. If you create an identity-based policy that is more restrictive than the minimum required permissions, the console won't function as intended for entities (users or roles) with that policy.

You don't need to allow minimum console permissions for users that are making calls only to the AWS CLI or the AWS API. Instead, allow access to only the actions that match the API operation that they're trying to perform.

To ensure that users and roles can still use the Route 53 ARC console, also attach a ReadOnly AWS managed policy for Route 53 ARC to the entities. For more information, see the Route 53 ARC managed policies page or Adding permissions to a user in the IAM User Guide.

To give users full access to use Route 53 ARC features through the console, attach a policy like the following to the user, to give the user full permissions to configure Route 53 ARC resources and operations:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "route53-recovery-cluster:GetRoutingControlState",
        "route53-recovery-cluster:UpdateRoutingControlState",
        "route53-recovery-cluster:UpdateRoutingControlStates",
        "route53-recovery-control-config:CreateCluster",
        "route53-recovery-control-config:CreateControlPanel",
        "route53-recovery-control-config:CreateRoutingControl",
        "route53-recovery-control-config:CreateSafetyRule",
        "route53-recovery-control-config:DeleteCluster",
        "route53-recovery-control-config:DeleteControlPanel",
        "route53-recovery-control-config:DeleteRoutingControl",
        "route53-recovery-control-config:DeleteSafetyRule",
        "route53-recovery-control-config:DescribeCluster",
        "route53-recovery-control-config:DescribeControlPanel",
        "route53-recovery-control-config:DescribeSafetyRule",
        "route53-recovery-control-config:DescribeRoutingControl",
        "route53-recovery-control-config:DescribeRoutingControlByName",
        "route53-recovery-control-config:ListAssociatedRoute53HealthChecks",
        "route53-recovery-control-config:ListClusters",
        "route53-recovery-control-config:ListControlPanels",
        "route53-recovery-control-config:ListRoutingControls",
        "route53-recovery-control-config:ListSafetyRules",
        "route53-recovery-control-config:UpdateControlPanel",
        "route53-recovery-control-config:UpdateRoutingControl",
        "route53-recovery-readiness:CreateCell",
        "route53-recovery-readiness:CreateCrossAccountAuthorization",
        "route53-recovery-readiness:CreateReadinessCheck",
        "route53-recovery-readiness:CreateRecoveryGroup",
        "route53-recovery-readiness:CreateResourceSet"
      ]
    }
  ]
}
```
Examples: Route 53 ARC API actions

There are four separate APIs that you can use with Amazon Route 53 Application Recovery Controller:

- The zonal shift API, to work with the Route 53 ARC zonal shift data plane – to temporarily move traffic away from an Availability Zone to recover an application.
- The recovery readiness API, to work with the Route 53 ARC readiness check control plane – for example, to create recovery groups, resource sets, and readiness checks.
- The recovery control API, to work with the Route 53 ARC routing control control plane – for example, to create clusters, control panels, and routing controls.
• The recovery control data plane API, to work with the Route 53 ARC routing control data plane – to query and update routing control states to perform Regional failover and recovery.

To ensure that a user can use Route 53 ARC API actions, attach a policy that corresponds to the API that the user needs to work with, as described below.

To work with the zonal shift API, attach a policy like the following to the user:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "arc-zonal-shift:ListManagedResources",
        "arc-zonal-shift:GetManagedResource",
        "arc-zonal-shift:ListZonalShifts",
        "arc-zonal-shift:StartZonalShift",
        "arc-zonal-shift:UpdateZonalShift",
        "arc-zonal-shift:CancelZonalShift"
      ],
      "Resource": "*"
    }
  ]
}
```

To work with the recovery readiness API, attach a policy like the following to the user:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "route53-recovery-readiness:CreateCell",
        "route53-recovery-readiness:CreateCrossAccountAuthorization",
        "route53-recovery-readiness:CreateReadinessCheck",
        "route53-recovery-readiness:CreateRecoveryGroup",
        "route53-recovery-readiness:CreateResourceSet",
        "route53-recovery-readiness:DeleteCell",
        "route53-recovery-readiness:DeleteCrossAccountAuthorization",
        "route53-recovery-readiness:DeleteReadinessCheck",
        "route53-recovery-readiness:DeleteRecoveryGroup",
        "route53-recovery-readiness:DeleteResourceSet",
        "route53-recovery-readiness:GetArchitectureRecommendations",
        "route53-recovery-readiness:GetCell",
        "route53-recovery-readiness:GetCellReadinessSummary",
        "route53-recovery-readiness:GetReadinessCheck",
        "route53-recovery-readiness:GetReadinessCheckResourceStatus",
        "route53-recovery-readiness:GetReadinessCheckStatus",
        "route53-recovery-readiness:GetRecoveryGroup",
        "route53-recovery-readiness:GetRecoveryGroupReadinessSummary",
        "route53-recovery-readiness:GetResourceSet",
        "route53-recovery-readiness:ListCells",
        "route53-recovery-readiness:ListCrossAccountAuthorizations",
        "route53-recovery-readiness:ListReadinessChecks",
        "route53-recovery-readiness:ListRecoveryGroups",
        "route53-recovery-readiness:ListResourceSets",
        "route53-recovery-readiness:ListRules",
        "route53-recovery-readiness:ListTagsForResources",
        "route53-recovery-readiness:UpdateCell",
        "route53-recovery-readiness:UpdateReadinessCheck",
        "route53-recovery-readiness:UpdateRecoveryGroup"
      ]
    }
  ]
}
```
To work with the recovery control API, attach a policy like the following to the user:

```json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "route53-recovery-control-config:CreateCluster",
        "route53-recovery-control-config:CreateControlPanel",
        "route53-recovery-control-config:CreateRoutingControl",
        "route53-recovery-control-config:CreateSafetyRule",
        "route53-recovery-control-config:DeleteCluster",
        "route53-recovery-control-config:DeleteControlPanel",
        "route53-recovery-control-config:DeleteRoutingControl",
        "route53-recovery-control-config:DeleteSafetyRule",
        "route53-recovery-control-config:DescribeCluster",
        "route53-recovery-control-config:DescribeControlPanel",
        "route53-recovery-control-config:DescribeRoutingControl",
        "route53-recovery-control-config:DescribeSafetyRule",
        "route53-recovery-control-config:DescribeControlPanel",
        "route53-recovery-control-config:DescribeRoutingControl",
        "route53-recovery-control-config:DescribeControlPanelByIndex",
        "route53-recovery-control-config:ListControlPanels",
        "route53-recovery-control-config:ListControlPanels",
        "route53-recovery-control-config:ListRoutingControls",
        "route53-recovery-control-config:ListRoutingControls",
        "route53-recovery-control-config:ListSafetyRules",
        "route53-recovery-control-config:ListSafetyRules",
        "route53-recovery-control-config:ListTagsForResource",
        "route53-recovery-control-config:ListTagsForResource",
        "route53-recovery-control-config:UpdateControlPanel",
        "route53-recovery-control-config:UpdateRoutingControl",
        "route53-recovery-control-config:UpdateRoutingControl",
        "route53-recovery-control-config:UpdateRoutingControl",
        "route53-recovery-control-config:TagResource",
        "route53-recovery-control-config:TagResource",
        "route53-recovery-control-config:UntagResource",
        "route53-recovery-control-config:UntagResource"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "route53:GetHealthCheck",
        "route53:CreateHealthCheck",
        "route53:DeleteHealthCheck",
        "route53:ChangeTagsForResource"
      ],
      "Resource": "*"
    }
  ]
}
```

To perform tasks in Route 53 ARC with the recovery cluster data plane API, for example, updating routing control states to fail over during a disaster event, you can attach a Route 53 ARC IAM policy such as the following to your IAM user:

The `AllowSafetyRuleOverride` boolean gives permission to override safety rules that you've configured as safeguards for routing controls. This permission might be required in "break glass"
scenarios to bypass the safeguards in disasters or other urgent failover scenarios. For example, an operator might need to fail over quickly for disaster recovery, and one or more safety rules might unexpectedly prevent a routing control state update required to reroute traffic. This permission allows the operator to specify safety rules to override when making API calls to update routing control states. For more information, see Overriding safety rules to reroute traffic (p. 91).

If you want to allow an operator to use the recovery cluster data plane API but prevent overriding safety rules, you can attach a policy such as the following, but set the AllowSafetyRuleOverrides boolean to false.

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Action": [
            "route53-recovery-cluster:GetRoutingControlState",
            "route53-recovery-cluster:ListRoutingControls"
         ],
         "Resource": "*"
      },
      {
         "Effect": "Allow",
         "Action": [
            "route53-recovery-cluster:UpdateRoutingControlStates",
            "route53-recovery-cluster:UpdateRoutingControlState"
         ],
         "Resource": "*",
         "Condition": {
            "Bool": {
               "route53-recovery-cluster:AllowSafetyRulesOverrides": "true"
            }
         }
      }
   ]
}
```

### Using service-linked roles for Route 53 ARC

Amazon Route 53 Application Recovery Controller uses an AWS Identity and Access Management (IAM) service-linked role. A service-linked role is a unique type of IAM role that is linked directly to Route 53 ARC. The service-linked role is predefined by Route 53 ARC and includes all the permissions that the service requires to call other AWS services on your behalf.

A service-linked role makes setting up Route 53 ARC easier because you don’t have to manually add the necessary permissions. Route 53 ARC defines the permissions of its service-linked role, and unless defined otherwise, only Route 53 ARC can assume its roles. The defined permissions include the trust policy and the permissions policy, and that permissions policy cannot be attached to any other IAM entity.

You can delete a service-linked role only after first deleting its related resources. This protects your Route 53 ARC resources because you can’t inadvertently remove permission to access the resources.

For information about other services that support service-linked roles, see AWS Services that work with IAM and look for the services that have Yes in the Service-linked role column. Choose a Yes with a link to view the service-linked role documentation for that service.

### Service-linked role permissions for Route 53 ARC

Route 53 ARC uses the service-linked role named Route53RecoveryReadinessServiceRolePolicy to access resources and configurations to check readiness.
This service-linked role uses the managed policy Route53RecoveryReadinessServiceRolePolicy.

The Route53RecoveryReadinessServiceRolePolicy service-linked role trusts the following service to assume the role:

- route53-recovery-readiness.amazonaws.com

The role permissions policy allows Route 53 ARC to complete the following actions on the specified resources:

```json
[
  {
    "Version": "2012-10-17",
    "Statement": [
      {
        "Effect": "Allow",
        "Action": [
          "dynamodb:DescribeReservedCapacity",
          "dynamodb:DescribeReservedCapacityOfferings"
        ],
        "Resource": "arn:aws:dynamodb:*:*:*"
      },
      {
        "Effect": "Allow",
        "Action": [
          "dynamodb:DescribeTable",
          "dynamodb:DescribeTimeToLive"
        ],
        "Resource": "arn:aws:dynamodb:*:*:table/*"
      },
      {
        "Effect": "Allow",
        "Action": [
          "ec2:DescribeCustomerGateways",
          "ec2:DescribeVpnGateways"
        ],
        "Resource": "arn:aws:ec2:*:*:*"
      },
      {
        "Effect": "Allow",
        "Action": [
          "iam:CreateServiceLinkedRole"
        ],
        "Resource": "arn:aws:iam::*:role/aws-service-role/servicequotas.amazonaws.com/AWSServiceRoleForServiceQuotas",
        "Condition": {
          "StringLike": {
            "iam:AWSServiceName": "servicequotas.amazonaws.com"
          }
        }
      }
    ],
    "Effect": "Allow",
    "Action": [
      "lambda:GetFunctionConcurrency",
      "lambda:GetFunctionConfiguration",
      "lambda:GetProvisionedConcurrencyConfig",
      "lambda:ListProvisionedConcurrencyConfigs",
      "lambda:ListAliases",
      "lambda:ListVersionsByFunction"
    ],
    "Resource": "arn:aws:lambda:*:*:function:*"
  }
]
```
"Action": ["rds:DescribeDBClusters"],
"Resource": "arn:aws:rds::*:cluster:*"},
{
"Effect": "Allow",
"Action": ["rds:DescribeDBInstances"],
"Resource": "arn:aws:rds::*:db:*"},
{
"Effect": "Allow",
"Action": ["route53:ListResourceRecordSets"],
"Resource": "arn:aws:route53::hostedzone/*"},
{
"Effect": "Allow",
"Action": ["route53:GetHealthCheck", "route53:GetHealthCheckStatus"],
"Resource": "arn:aws:route53::healthcheck/*"},
{
"Effect": "Allow",
"Action": ["servicequotas:RequestServiceQuotaIncrease"],
"Resource": "arn:aws:servicequotas::*:*:*"},
{
"Effect": "Allow",
"Action": ["sns:GetTopicAttributes", "sns:ListSubscriptionsByTopic"],
"Resource": "arn:aws:sns::*:*:*"},
{
"Effect": "Allow",
"Action": ["sqs:GetQueueAttributes", "sqs:GetQueueUrl"],
"Resource": "arn:aws:sqs::*:*:*"},
{
"Effect": "Allow",
You must configure permissions to allow an IAM entity (such as a user, group, or role) to create, edit, or delete a service-linked role. For more information, see Service-linked role permissions in the IAM User Guide.

Creating a service-linked role for Route 53 ARC

You don't need to manually create a service-linked role. When you create the first readiness check or cross account authorization in the AWS Management Console, the AWS CLI, or the AWS API, Route 53 ARC creates the service-linked role for you.

If you delete this service-linked role, and then need to create it again, you can use the same process to recreate the role in your account. When you create the first readiness check or cross account authorization, Route 53 ARC creates the service-linked role for you again.

Editing a service-linked role for Route 53 ARC

Route 53 ARC does not allow you to edit the Route53RecoveryReadinessServiceRolePolicy service-linked role. After you create a service-linked role, you cannot change the name of the role because various entities might reference the role. However, you can edit the description of the role because various entities might reference the role. However, you can edit the description of the role using IAM. For more information, see Editing a service-linked role in the IAM User Guide.

Deleting a service-linked role for Route 53 ARC

If you no longer need to use a feature or service that requires a service-linked role, we recommend that you delete that role. That way you don’t have an unused entity that is not actively monitored.
or maintained. However, you must clean up the resources for your service-linked role before you can manually delete it.

After you have removed your readiness checks and your cross-account authorizations, then you can delete the service-linked role. For more information about readiness checks, see Readiness check in Amazon Route 53 Application Recovery Controller (p. 55). For more information about cross-account authorizations, see Create cross-account authorizations in Route 53 ARC (p. 77).

**Note**

If the Route 53 ARC service is using the role when you try to delete the resources, then the deletion might fail. If that happens, wait for a few minutes and try the operation again.

**To manually delete the service-linked role using IAM**

Use the IAM console, the AWS CLI, or the AWS API to delete the Route53RecoveryReadinessServiceRolePolicy service-linked role. For more information, see Deleting a service-linked role in the IAM User Guide.

**Updates to the Route 53 ARC service-linked role**

For updates to Route53RecoveryReadinessServiceRolePolicy, the AWS managed policy for the Route 53 ARC service-linked role, see the AWS managed policies updates table (p. 133). You can also subscribe to automatic RSS alerts on the Route 53 ARC Document history page (p. 149).

**AWS managed policies for Amazon Route 53 Application Recovery Controller**

An AWS managed policy is a standalone policy that is created and administered by AWS. AWS managed policies are designed to provide permissions for many common use cases so that you can start assigning permissions to users, groups, and roles.

Keep in mind that AWS managed policies might not grant least-privilege permissions for your specific use cases because they're available for all AWS customers to use. We recommend that you reduce permissions further by defining customer managed policies that are specific to your use cases.

You cannot change the permissions defined in AWS managed policies. If AWS updates the permissions defined in an AWS managed policy, the update affects all principal identities (users, groups, and roles) that the policy is attached to. AWS is most likely to update an AWS managed policy when a new AWS service is launched or new API operations become available for existing services.

For more information, see AWS managed policies in the IAM User Guide.

**AWS managed policy:**

**Route53RecoveryReadinessServiceRolePolicy**

You can't attach Route53RecoveryReadinessServiceRolePolicy to your IAM entities. This policy is attached to a service-linked role that allows Amazon Route 53 Application Recovery Controller to access AWS services and resources that are used or managed by Route 53 ARC. For more information, see Using service-linked roles for Route 53 ARC (p. 126).

**AWS managed policy:**

**AmazonRoute53RecoveryReadinessFullAccess**

You can attach AmazonRoute53RecoveryReadinessFullAccess to your IAM entities. This policy grants full access to actions for working with recovery readiness (readiness check) in Route 53 ARC. Attach it to IAM users and other principals who need full access to recovery readiness actions.
AWS managed policy: AmazonRoute53RecoveryReadinessReadOnlyAccess

You can attach AmazonRoute53RecoveryReadinessReadOnlyAccess to your IAM entities. This policy grants read-only access to actions for working with recovery readiness in Route 53 ARC. It's useful for users who need to view readiness statuses and recovery group configurations. These users can't create, update, or delete recovery readiness resources.

```json
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "route53-recovery-readiness:GetCell",
                "route53-recovery-readiness:GetReadinessCheck",
                "route53-recovery-readiness:GetReadinessCheckResourceStatus",
                "route53-recovery-readiness:GetReadinessCheckStatus",
                "route53-recovery-readiness:GetRecoveryGroup",
                "route53-recovery-readiness:GetRecoveryGroupReadinessSummary",
                "route53-recovery-readiness:GetResourceSet",
                "route53-recovery-readiness:ListCells",
                "route53-recovery-readiness:ListCrossAccountAuthorizations",
                "route53-recovery-readiness:ListReadinessChecks",
                "route53-recovery-readiness:ListRecoveryGroups",
                "route53-recovery-readiness:ListResourceSets",
                "route53-recovery-readiness:ListRules",
                "route53-recovery-readiness:ListTagsForResources"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "route53-recovery-readiness:GetArchitectureRecommendations",
                "route53-recovery-readiness:GetCellReadinessSummary"
            ],
            "Resource": "arn:aws:route53-recovery-readiness::*:*"
        }
    ]
}
```
AWS managed policy: 
AmazonRoute53RecoveryControlConfigFullAccess

You can attach AmazonRoute53RecoveryControlConfigFullAccess to your IAM entities. This policy grants full access to actions for working with recovery control configuration in Route 53 ARC. Attach it to IAM users and other principals who need full access to recovery control configuration actions.

At your discretion, you can add access to additional Amazon Route 53 actions to enable users to create health checks for routing controls. For example, you might allow permission for one or more of the following actions: route53:GetHealthCheck, route53:CreateHealthCheck, route53:DeleteHealthCheck, and route53:ChangeTagsForResource.

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Action": [
            "route53-recovery-control-config:*"
         ],
         "Resource": "*"
      }
   ]
}
```

AWS managed policy: 
AmazonRoute53RecoveryControlConfigReadOnlyAccess

You can attach AmazonRoute53RecoveryControlConfigReadOnlyAccess to your IAM entities. It's useful for users who need to view routing control and safety rule configurations. This policy grants read-only access to actions for working with recovery control configuration in Route 53 ARC. These users can't create, update, or delete recovery control resources.

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Action": [
            "route53-recovery-control-config:DescribeCluster",
            "route53-recovery-control-config:DescribeControlPanel",
            "route53-recovery-control-config:DescribeRoutingControl",
            "route53-recovery-control-config:DescribeRoutingControlByName",
            "route53-recovery-control-config:DescribeSafetyRule",
            "route53-recovery-control-config:GetResourcePolicy",
            "route53-recovery-control-config:ListAssociatedRoute53HealthChecks",
            "route53-recovery-control-config:ListClusters",
            "route53-recovery-control-config:ListControlPanels",
            "route53-recovery-control-config:ListRoutingControls",
            "route53-recovery-control-config:ListSafetyRules",
            "route53-recovery-control-config:ListTagsForResource"
         ],
         "Resource": "*"
      }
   ]
}
```
AWS managed policy:
AmazonRoute53RecoveryClusterFullAccess

You can attach AmazonRoute53RecoveryClusterFullAccess to your IAM entities. This policy grants full access to actions for working with the cluster data plane in Route 53 ARC. Attach it to IAM users and other principals who need full access to updating and retrieving routing control states.

```json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "route53-recovery-cluster:*"
      ],
      "Resource": "*"
    }
  ]
}
```

AWS managed policy:
AmazonRoute53RecoveryClusterReadOnlyAccess

You can attach AmazonRoute53RecoveryClusterReadOnlyAccess to your IAM entities. This policy grants read-only access to the cluster data plane in Route 53 ARC. These users can retrieve routing control states but can't update them.

```json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "route53-recovery-cluster:GetRoutingControlState",
        "route53-recovery-cluster:ListRoutingControls"
      ],
      "Resource": "*"
    }
  ]
}
```

Route 53 ARC updates to AWS managed policies

View details about updates to AWS managed policies for Route 53 ARC since this service began tracking these changes. For automatic alerts about changes to this page, subscribe to the RSS feed on the Route 53 ARC Document history page (p. 149).

<table>
<thead>
<tr>
<th>Change</th>
<th>Description</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>AmazonRoute53RecoveryControlConfigReadOnlyAccess</td>
<td>Updated policy</td>
<td>September 19, 2023</td>
</tr>
<tr>
<td>AmazonRoute53RecoveryControlConfigReadOnlyAccess</td>
<td>Updated policy</td>
<td>September 19, 2023</td>
</tr>
</tbody>
</table>

Add permissions for [p. 132] to support returning details about AWS Resource Access Manager resource policies for shared resources.
<table>
<thead>
<tr>
<th>Change</th>
<th>Description</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Route53RecoveryReadinessServiceRolePolicy</td>
<td>Route 53 ARC added new permissions to query information about Amazon EC2 instances. Route 53 ARC uses the following permissions to support polling Amazon EC2 instances, to run readiness checks and determine the readiness status for the instances.</td>
<td>February 17, 2023</td>
</tr>
<tr>
<td>− Updated policy</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Route53RecoveryReadinessServiceRolePolicy</td>
<td>Route 53 ARC added a new permission to query information about Lambda functions. Route 53 ARC uses the following permission to query information about Lambda functions to run readiness checks and determine the readiness status for the functions.</td>
<td>August 31, 2022</td>
</tr>
<tr>
<td>− Updated policy</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AmazonRoute53RecoveryControlConfigFullAccess</td>
<td>Removed Amazon Route 53 permissions from the policy and added note listing the optional permissions.</td>
<td>May 26, 2022</td>
</tr>
<tr>
<td>− Updated policy</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AmazonRoute53RecoveryControlConfigFullAccess</td>
<td>Added missing required Amazon Route 53 permissions to the policy.</td>
<td>April 15, 2022</td>
</tr>
<tr>
<td>− Updated policy</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AmazonRoute53RecoveryClusterReadOnlyAccess</td>
<td>Route 53 ARC added a new permission, route53-recovery-cluster:ListRoutingControls, to allow listing routing control ARNs with high availability.</td>
<td>March 15, 2022</td>
</tr>
<tr>
<td>− Updated policy</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AmazonRoute53RecoveryControlReadOnlyAccess</td>
<td>Route 53 ARC added a new permission, route53-recovery-control-config:ListTagsForResources, to allow listing tags for a resource.</td>
<td>December 20, 2021</td>
</tr>
<tr>
<td>− Updated policy</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Change</td>
<td>Description</td>
<td>Date</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>--------------------</td>
</tr>
<tr>
<td><strong>Route53RecoveryReadinessServiceRolePolicy</strong></td>
<td>Route 53 ARC added a new permission to query information about Amazon API Gateway. Route 53 ARC uses the permission, apigateway:GET, to query information about API Gateway to run readiness checks and determine the readiness status.</td>
<td>October 28, 2021</td>
</tr>
<tr>
<td><strong>AmazonRoute53RecoveryReadinessReadOnlyAccess</strong></td>
<td>(Route 53 ARC added [131]) two new permissions to AmazonRoute53RecoveryReadinessReadOnlyAccess (p. 131): Route 53 ARC uses route53-recovery-readiness:GetArchitectureRecommendations and route53-recovery-readiness:GetCellReadinessSummary to allow read-only access to these actions for working with recovery readiness.</td>
<td>October 15, 2021</td>
</tr>
</tbody>
</table>
| **Route53RecoveryReadinessServiceRolePolicy** | Route 53 ARC added new permissions to query information about Lambda functions. Route 53 ARC uses the following permissions to query information about Lambda functions to run readiness checks and determine the readiness status for those functions.  

  - lambda:GetFunctionConcurrency  
  - lambda:GetFunctionConfiguration  
  - lambda:GetProvisionedConcurrencyConfig  
  - lambda:ListAliases  
  - lambda:ListVersionsByFunction  
  - lambda:ListEventSourceMappings  
  - lambda:ListFunctions | October 8, 2021 |
### Troubleshooting Amazon Route 53 Application Recovery Controller identity and access

Use the following information to help you diagnose and fix common issues that you might encounter when working with Route 53 ARC and IAM.

**Topics**
- I am not authorized to perform an action in Route 53 ARC (p. 136)
- I am not authorized to perform iam:PassRole (p. 136)
- I’m an administrator and want to allow others to access Route 53 ARC (p. 137)
- I want to allow people outside of my AWS account to access my Route 53 ARC resources (p. 137)

### I am not authorized to perform an action in Route 53 ARC

If the AWS Management Console tells you that you’re not authorized to perform an action, then you must contact your administrator for assistance. Your administrator is the person that provided you with your credentials.

The following example error occurs when the mateojackson IAM user tries to use the console to view details about a fictional **my-example-widget** resource but does not have the fictional **route53-recovery-readiness:GetWidget** permissions.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform: route53-recovery-readiness:GetWidget on resource: my-example-widget
```

In this case, Mateo asks his administrator to update his policies to allow him to access the **my-example-widget** resource using the **route53-recovery-readiness:GetWidget** action.

### I am not authorized to perform iam:PassRole

If you receive an error that you’re not authorized to perform the **iam:PassRole** action, your policies must be updated to allow you to pass a role to Route 53 ARC.

---

<table>
<thead>
<tr>
<th>Change</th>
<th>Description</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Added new managed policies</td>
<td>Route 53 ARC added the following new managed policies:</td>
<td>August 18, 2021</td>
</tr>
<tr>
<td>AmazonRoute53RecoveryReadinessFullAccess (p. 130)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AmazonRoute53RecoveryReadinessReadOnlyAccess (p. 131)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AmazonRoute53RecoveryClusterFullAccess (p. 133)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AmazonRoute53RecoveryClusterReadOnlyAccess (p. 133)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AmazonRoute53RecoveryControlConfigFullAccess (p. 132)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AmazonRoute53RecoveryControlConfigReadOnlyAccess (p. 132)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Route 53 ARC started tracking changes</td>
<td>Route 53 ARC started tracking changes for its AWS managed policies.</td>
<td>July 27, 2021</td>
</tr>
</tbody>
</table>
Some AWS services allow you to pass an existing role to that service instead of creating a new service role or service-linked role. To do this, you must have permissions to pass the role to the service.

The following example error occurs when an IAM user named marymajor tries to use the console to perform an action in Route 53 ARC. However, the action requires the service to have permissions that are granted by a service role. Mary does not have permissions to pass the role to the service.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform: iam:PassRole
```

In this case, Mary’s policies must be updated to allow her to perform the `iam:PassRole` action.

If you need help, contact your AWS administrator. Your administrator is the person who provided you with your sign-in credentials.

I'm an administrator and want to allow others to access Route 53 ARC

To allow others to access Route 53 ARC, you must create an IAM entity (user or role) for the person or application that needs access. They will use the credentials for that entity to access AWS. You must then attach a policy to the entity that grants them the correct permissions in Route 53 ARC.

To get started right away, see Creating your first IAM delegated user and group in the IAM User Guide.

I want to allow people outside of my AWS account to access my Route 53 ARC resources

You can create a role that users in other accounts or people outside of your organization can use to access your resources. You can specify who is trusted to assume the role. For services that support resource-based policies or access control lists (ACLs), you can use those policies to grant people access to your resources.

To learn more, consult the following:

- To learn whether Route 53 ARC supports these features, see How Amazon Route 53 Application Recovery Controller works with IAM (p. 115).
- To learn how to provide access to your resources across AWS accounts that you own, see Providing access to an IAM user in another AWS account that you own in the IAM User Guide.
- To learn how to provide access to your resources to third-party AWS accounts, see Providing access to AWS accounts owned by third parties in the IAM User Guide.
- To learn how to provide access through identity federation, see Providing access to externally authenticated users (identity federation) in the IAM User Guide.
- To learn the difference between using roles and resource-based policies for cross-account access, see How IAM roles differ from resource-based policies in the IAM User Guide.

Logging and monitoring in Amazon Route 53 Application Recovery Controller

Monitoring is an important part of maintaining the availability and performance of Amazon Route 53 Application Recovery Controller and your AWS solutions. You should collect monitoring data from all of the parts of your AWS solution so that you can more easily debug a multi-point failure if one occurs.
AWS provides several tools for monitoring your Route 53 ARC resources and activity, and responding to potential incidents:

**Amazon CloudWatch metrics and alarms**

Using CloudWatch, you can monitor, in real time, your AWS resources and the applications that you run on AWS. CloudWatch collects and tracks metrics, which are variables that you measure over time. You can create alarms that watch specific metrics, and then send notifications or automatically make changes to the resources you are monitoring when the metric exceeds a certain threshold for a period of time. For more information, see [Using Amazon CloudWatch with Route 53 ARC](p. 96).

**AWS CloudTrail logs**

CloudTrail provides a record of actions taken by a user, role, or an AWS service in Route 53 ARC. CloudTrail captures all API calls for Route 53 ARC as events, including calls from the Route 53 ARC console and from code calls to the Route 53 ARC API. For more information, see [Logging Route 53 ARC API calls using AWS CloudTrail](p. 99).

### Compliance validation for Amazon Route 53 Application Recovery Controller

Third-party auditors assess the security and compliance of Amazon Route 53 Application Recovery Controller as part of multiple AWS compliance programs. These include SOC, PCI, HIPAA, and others.

To learn whether an AWS service is within the scope of specific compliance programs, see [AWS services in Scope by Compliance Program](p. 86) and choose the compliance program that you are interested in. For general information, see [AWS Compliance Programs](p. 86).

You can download third-party audit reports using AWS Artifact. For more information, see [Downloading Reports in AWS Artifact](p. 86).

Your compliance responsibility when using AWS services is determined by the sensitivity of your data, your company's compliance objectives, and applicable laws and regulations. AWS provides the following resources to help with compliance:

- **Security and Compliance Quick Start Guides** – These deployment guides discuss architectural considerations and provide steps for deploying baseline environments on AWS that are security and compliance focused.
- **Architecting for HIPAA Security and Compliance on Amazon Web Services** – This whitepaper describes how companies can use AWS to create HIPAA-eligible applications.

  **Note**
  Not all AWS services are HIPAA eligible. For more information, see the [HIPAA Eligible Services Reference](p. 86).

- **AWS Compliance Resources** – This collection of workbooks and guides might apply to your industry and location.
- **Evaluating Resources with Rules** in the [AWS Config Developer Guide](p. 86) – The AWS Config service assesses how well your resource configurations comply with internal practices, industry guidelines, and regulations.
- **AWS Security Hub** – This AWS service provides a comprehensive view of your security state within AWS. Security Hub uses security controls to evaluate your AWS resources and to check your compliance against security industry standards and best practices. For a list of supported services and controls, see [Security Hub controls reference](p. 86).
- **AWS Audit Manager** – This AWS service helps you continuously audit your AWS usage to simplify how you manage risk and compliance with regulations and industry standards.
Resilience in Amazon Route 53 Application Recovery Controller

The AWS global infrastructure is built around AWS Regions and Availability Zones. AWS Regions provide multiple physically separated and isolated Availability Zones, which are connected with low-latency, high-throughput, and highly redundant networking. With Availability Zones, you can design and operate applications and databases that automatically fail over between zones without interruption. Availability Zones are more highly available, fault tolerant, and scalable than traditional single or multiple data center infrastructures.

For more information about AWS Regions and Availability Zones, see AWS Global Infrastructure.

In addition to the AWS global infrastructure, Route 53 ARC offers several features to help support your data resiliency and backup needs.

Infrastructure security in Amazon Route 53 Application Recovery Controller

As a managed service, Amazon Route 53 Application Recovery Controller is protected by AWS global network security. For information about AWS security services and how AWS protects infrastructure, see AWS Cloud Security. To design your AWS environment using the best practices for infrastructure security, see Infrastructure Protection in Security Pillar AWS Well-Architected Framework.

You use AWS published API calls to access Route 53 ARC through the network. Clients must support the following:

- Transport Layer Security (TLS). We require TLS 1.2 and recommend TLS 1.3.
- Cipher suites with perfect forward secrecy (PFS) such as DHE (Ephemeral Diffie-Hellman) or ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). Most modern systems such as Java 7 and later support these modes.

Additionally, requests must be signed by using an access key ID and a secret access key that is associated with an IAM principal. Or you can use the AWS Security Token Service (AWS STS) to generate temporary security credentials to sign requests.
Code examples for Application Recovery Controller using AWS SDKs

The following code examples show how to use Application Recovery Controller with an AWS software development kit (SDK).

*Actions* are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios and cross-service examples.

For a complete list of AWS SDK developer guides and code examples, see [Using Route 53 ARC with an AWS SDK](p. 21). This topic also includes information about getting started and details about previous SDK versions.

**Code examples**
- *Actions for Application Recovery Controller using AWS SDKs* (p. 140)
  - Get the state of an Application Recovery Controller routing control using an AWS SDK (p. 140)
  - Update the state of an Application Recovery Controller routing control using an AWS SDK (p. 142)

**Actions for Application Recovery Controller using AWS SDKs**

The following code examples demonstrate how to perform individual Application Recovery Controller actions with AWS SDKs. These excerpts call the Application Recovery Controller API and are code excerpts from larger programs that must be run in context. Each example includes a link to GitHub, where you can find instructions for setting up and running the code.

The following examples include only the most commonly used actions. For a complete list, see the [Amazon Route 53 Application Recovery Controller API Reference](p. 21).

**Examples**
- Get the state of an Application Recovery Controller routing control using an AWS SDK (p. 140)
- Update the state of an Application Recovery Controller routing control using an AWS SDK (p. 142)

**Get the state of an Application Recovery Controller routing control using an AWS SDK**

The following code examples show how to get the state of an Application Recovery Controller routing control.

**Java**

*SDK for Java 2.x*

*Note*

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](p. 21).
public static GetRoutingControlStateResponse getRoutingControlState(List<ClusterEndpoint> clusterEndpoints, String routingControlArn) {
    // As a best practice, we recommend choosing a random cluster endpoint to
get or set routing control states.
    // For more information, see https://docs.aws.amazon.com/r53recovery/
latest/dg/route53-arc-best-practices.html#route53-arc-best-practices.regional
Collections.shuffle(clusterEndpoints);
for (ClusterEndpoint clusterEndpoint : clusterEndpoints) {
    try {
        System.out.println(clusterEndpoint);
        Route53RecoveryClusterClient client =
        Route53RecoveryClusterClient.builder()
            .endpointOverride(URI.create(clusterEndpoint.endpoint()))
            .region(Region.of(clusterEndpoint.region())).build();
        return client.getRoutingControlState(
            GetRoutingControlStateRequest.builder()
                .routingControlArn(routingControlArn).build());
    } catch (Exception exception) {
        System.out.println(exception);
    }
}
return null;
}

• For API details, see GetRoutingControlState in AWS SDK for Java 2.x API Reference.

Python

SDK for Python (Boto3)

Note
There's more on GitHub. Find the complete example and learn how to set up and run in
the AWS Code Examples Repository.

import boto3
def create_recovery_client(cluster_endpoint):
    """
    Creates a Boto3 Route 53 Application Recovery Controller client for the
    specified
    cluster endpoint URL and AWS Region.
    :param cluster_endpoint: The cluster endpoint URL and Region.
    :return: The Boto3 client.
    """
return boto3.client(
    "route53-recovery-cluster",
    endpoint_url=cluster_endpoint["Endpoint"],
    region_name=cluster_endpoint["Region"],
)

def get_routing_control_state(routing_control_arn, cluster_endpoints):
    """
    Gets the state of a routing control. Cluster endpoints are tried in
    sequence until the first successful response is received.
Update the state of a routing control

```java
:param routing_control_arn: The ARN of the routing control to look up.
:param cluster_endpoints: The list of cluster endpoints to query.
:return: The routing control state response.

# As a best practice, we recommend choosing a random cluster endpoint to get or
set routing control states.
# For more information, see https://docs.aws.amazon.com/r53recovery/latest/dg/
route53-arc-best-practices.html#route53-arc-best-practices.regional
random.shuffle(cluster_endpoints)
for cluster_endpoint in cluster_endpoints:
    try:
        recovery_client = create_recovery_client(cluster_endpoint)
        response = recovery_client.get_routing_control_state(
            RoutingControlArn=routing_control_arn
        )
        return response
    except Exception as error:
        print(error)
        raise error
```

- For API details, see [GetRoutingControlState](https://boto3.amazonaws.com/v1/documentation/api/latest/reference/services/route53arc.html#Route53RecoveryCtrlClient.get_routing_control_state) in AWS SDK for Python (Boto3) API Reference.

For a complete list of AWS SDK developer guides and code examples, see [Using Route 53 ARC with an AWS SDK](p. 21). This topic also includes information about getting started and details about previous SDK versions.

### Update the state of an Application Recovery Controller routing control using an AWS SDK

The following code examples show how to update the state of an Application Recovery Controller routing control.

#### Java

**SDK for Java 2.x**

**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](https://github.com/aws/aws-sdk-java).

```java
public static UpdateRoutingControlStateResponse
    updateRoutingControlState(List<ClusterEndpoint> clusterEndpoints,
    String routingControlArn,
    String routingControlState) {
    // As a best practice, we recommend choosing a random cluster endpoint to
get or set routing control states.
    // For more information, see https://docs.aws.amazon.com/r53recovery/latest/dg/
route53-arc-best-practices.html#route53-arc-best-practices.regional
Collections.shuffle(clusterEndpoints);
for (ClusterEndpoint clusterEndpoint : clusterEndpoints) {
    try {
        System.out.println(clusterEndpoint);
```
Route53RecoveryClusterClient client = Route53RecoveryClusterClient.builder()
    .endpointOverride(URI.create(clusterEndpoint.endpoint()))
    .region(Region.of(clusterEndpoint.region()))
    .build();
return client.updateRoutingControlState(
    UpdateRoutingControlStateRequest.builder()
    .routingControlArn(routingControlArn).routingControlState(routingControlState).build());
}

} catch (Exception exception) {
System.out.println(exception);
}

return null;


• For API details, see UpdateRoutingControlState in AWS SDK for Java 2.x API Reference.

Python

SDK for Python (Boto3)

Note
There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

import boto3

def create_recovery_client(cluster_endpoint):
    """
    Creates a Boto3 Route 53 Application Recovery Controller client for the specified cluster endpoint URL and AWS Region.
    :param cluster_endpoint: The cluster endpoint URL and Region.
    :return: The Boto3 client.
    """
    return boto3.client(
        "route53-recovery-cluster",
        endpoint_url=cluster_endpoint["Endpoint"],
        region_name=cluster_endpoint["Region"],
    )

def update_routing_control_state(
    routing_control_arn, cluster_endpoints, routing_control_state
):
    """
    Updates the state of a routing control. Cluster endpoints are tried in sequence until the first successful response is received.
    :param routing_control_arn: The ARN of the routing control to update the state for.
    :param cluster_endpoints: The list of cluster endpoints to try.
    :param routing_control_state: The new routing control state.
    :return: The routing control update response.
    """
    # As a best practice, we recommend choosing a random cluster endpoint to get or set routing control states.
# For more information, see https://docs.aws.amazon.com/r53recovery/latest/dg/route53-arc-best-practices.html#route53-arc-best-practices.regional
random.shuffle(cluster_endpoints)
for cluster_endpoint in cluster_endpoints:
    try:
        recovery_client = create_recovery_client(cluster_endpoint)
        response = recovery_client.update_routing_control_state(
            RoutingControlArn=routing_control_arn,
            RoutingControlState=routing_control_state,
        )
        return response
    except Exception as error:
        print(error)

- For API details, see UpdateRoutingControlState in AWS SDK for Python (Boto3) API Reference.

For a complete list of AWS SDK developer guides and code examples, see Using Route 53 ARC with an AWS SDK (p. 21). This topic also includes information about getting started and details about previous SDK versions.
Amazon Route 53 Application Recovery Controller is subject to the following quotas (formerly referred to as limits).

### Quotas for Route 53 ARC readiness check

<table>
<thead>
<tr>
<th>Entity</th>
<th>Quota</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of recovery groups per account</td>
<td>5</td>
</tr>
<tr>
<td>Number of cells per account</td>
<td>15</td>
</tr>
<tr>
<td>Number of nested cells per cell</td>
<td>3</td>
</tr>
<tr>
<td>Number of cells per recovery group</td>
<td>3</td>
</tr>
<tr>
<td>Number of resources per cell</td>
<td>10</td>
</tr>
<tr>
<td>Number of resources per recovery group</td>
<td>10</td>
</tr>
<tr>
<td>Number of resources per resource set</td>
<td>6</td>
</tr>
<tr>
<td>Number of resource sets per account</td>
<td>200</td>
</tr>
<tr>
<td>Number of readiness checks per account</td>
<td>200</td>
</tr>
<tr>
<td>Number of cross-account authorizations</td>
<td>100</td>
</tr>
</tbody>
</table>

### Quotas for Route 53 ARC routing control

<table>
<thead>
<tr>
<th>Entity</th>
<th>Quota</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of clusters per account</td>
<td>2</td>
</tr>
<tr>
<td>Number of control panels per cluster</td>
<td>50</td>
</tr>
<tr>
<td>Number of routing controls per control panel</td>
<td>100</td>
</tr>
<tr>
<td>Total number of routing controls (in all control panels) per cluster</td>
<td>300</td>
</tr>
<tr>
<td>Number of safety rules per control panel</td>
<td>20</td>
</tr>
<tr>
<td>Number of routing controls per <code>UpdateRoutingControlStates</code> operation call</td>
<td>10</td>
</tr>
<tr>
<td>Entity</td>
<td>Quota</td>
</tr>
<tr>
<td>--------------------------------------------</td>
<td>-------</td>
</tr>
<tr>
<td>Number of mutating API calls to a cluster endpoint, per second</td>
<td>3</td>
</tr>
</tbody>
</table>
Amazon Route 53 Application
Recovery Controller Related information

The information and resources listed here can help you learn more about Amazon Route 53 Application Recovery Controller.

Topics
- Additional Amazon Route 53 Application Recovery Controller documentation (p. 147)
- Getting support (p. 147)
- Tips from the Amazon Web Services Blog (p. 148)

Additional Amazon Route 53 Application Recovery Controller documentation

The following related resources can help you as you work with Route 53 ARC.

- Recovery Readiness API Reference Guide for Amazon Route 53 Application Recovery Controller – Gives complete descriptions of the API actions, parameters, and data types, and a list of errors for recovery readiness.
- Recovery Control Configuration API Reference Guide for Amazon Route 53 Application Recovery Controller – Gives complete descriptions of the API actions, parameters, and data types, and a list of errors for recovery control configuration.
- Routing Control API Reference Guide for Amazon Route 53 Application Recovery Controller – Gives complete descriptions of the API actions, parameters, and data types, and a list of errors for routing control.
- Route 53 ARC product information – The primary web page for information about Route 53 ARC, including features and benefits.
- Route 53 ARC pricing information – Details about pricing.
- Terms of Use – Detailed information about our copyright and trademark; your account, license, and site access; and other topics.

Getting support

Support for Route 53 ARC is available in several forms.

- AWS Support Center – This site brings together information about your recent support cases and results from AWS Trusted Advisor and health checks, as well as providing links to discussion forums, technical FAQs, the service health dashboard, and information about AWS support plans.
- AWS Premium Support Information – The primary web page for information about AWS Premium Support, a one-on-one, fast-response support channel to help you build and run applications on AWS Infrastructure Services.
Tips from the Amazon Web Services Blog

The AWS Blog has a number of posts to help you use AWS services. For example, see the following blog posts about Amazon Route 53 Application Recovery Controller:

- To learn more about building fault tolerant services using Availability Zones (AZs) to more easily recover from hard failures and gray failures, including by starting a zonal shift, see the following AWS News blog post: Rapidly recover from application failures in a single AZ.
- To learn more about approaches for mitigating failures and then returning to normal operations with Route 53, including using Route 53 ARC, see the following AWS News blog post: Creating Disaster Recovery Mechanisms Using Amazon Route 53.
- To learn more about building a highly-resilient single-Region stack application with Route 53 ARC, see the following AWS News blog post, the first part in a series: Building highly resilient applications using Amazon Route 53 Application Recovery Controller, Part 1: Single-Region stack.
- To learn more about building a highly-resilient multi-Region stack application with Route 53 ARC, see the following AWS News blog post, the second part in a series: Building highly resilient applications using Amazon Route 53 Application Recovery Controller, Part 2: Multi-Region stack.
- To learn more about using Route 53 ARC and to download Hashicorp Terraform template to help you get started, see the following AWS News blog post: Running recovery-oriented applications with Amazon Route 53 Application Recovery Controller, AWS CI/CD tools, and Terraform.
- To learn more about using Route 53 ARC and to download an AWS CloudFormation template to help you get started, see the following AWS News blog post: Simplify recovery with Route 53 ARC.
Document history for the Amazon Route 53 Application Recovery Controller Developer Guide

The following entries describe important changes made to the Amazon Route 53 Application Recovery Controller documentation.

- **Version:** latest
- **Latest documentation update:** October 18, 2023

<table>
<thead>
<tr>
<th>Change</th>
<th>Description</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adds cross-account support for clusters</td>
<td>Adds cross-account support for clusters in Route 53 ARC with AWS Resource Access Manager, so that you can easily and securely use one cluster to host control panels and routing controls owned by several different AWS accounts. For more information, see <a href="#">Support cross-account for clusters in Route 53 ARC</a>.</td>
<td>October 18, 2023</td>
</tr>
<tr>
<td>Updates a managed policy</td>
<td>Updates the AmazonRoute53RecoveryControlConfigReadOnly managed policy to add permissions for GetResourcePolicy, to support returning details about AWS Resource Access Manager resource policies for shared resources. For more information, see <a href="#">AWS managed policies</a>.</td>
<td>September 19, 2023</td>
</tr>
<tr>
<td>Updated service linked role</td>
<td>Added new permissions, ec2:DescribeVpnGateways and ec2:DescribeCustomerGateways, to the service linked role for Route 53 ARC, to support polling Amazon EC2 instances. For more information, see <a href="#">Using service-linked roles for Route 53 ARC</a>.</td>
<td>February 17, 2023</td>
</tr>
<tr>
<td>Change</td>
<td>Description</td>
<td>Date</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
<td>------</td>
</tr>
<tr>
<td>GA release for zonal shift</td>
<td>Supports the GA release of zonal shift for Route 53 ARC, which includes attribute-based access control (ABAC) for managed resources that are registered in Route 53 ARC for zonal shift. For more information, see <a href="#">Attribute-based access control (ABAC) with Route 53 ARC.</a></td>
<td>January 10, 2023</td>
</tr>
<tr>
<td>Added new multi-AZ zonal shift</td>
<td>Added content describing a new service in Route 53 ARC, zonal shift, for multi-AZ applications. You can start a zonal shift to temporarily move traffic for a load balancer resource away from an Availability Zone. For more information, see <a href="#">Zonal shift in Route 53 ARC.</a></td>
<td>November 28, 2022</td>
</tr>
<tr>
<td>Updated service linked role</td>
<td>Added a new permission, <code>lambda:ListProvisionedConcurrencyConfigs</code>, to the service linked role for Route 53 ARC to query information about Lambda functions. For more information, see <a href="#">Using service-linked roles for Route 53 ARC.</a></td>
<td>August 31, 2022</td>
</tr>
<tr>
<td>Updated managed policy</td>
<td>Updated the <code>AmazonRoute53RecoveryControlConfigFullAccess</code> managed policy to remove Amazon Route 53 permissions and list them as optional. For more information, see <a href="#">AWS managed policies for Amazon Route 53 Application Recovery Controller.</a></td>
<td>May 26, 2022</td>
</tr>
<tr>
<td>Updated managed policy</td>
<td>Updated the <code>AmazonRoute53RecoveryControlConfigFullAccess</code> managed policy to include required Amazon Route 53 permissions. For more information, see <a href="#">AWS managed policies for Amazon Route 53 Application Recovery Controller.</a></td>
<td>April 15, 2022</td>
</tr>
<tr>
<td>Change</td>
<td>Description</td>
<td>Date</td>
</tr>
<tr>
<td>--------------------------------------------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>Added CLI example for the new list routing controls API</td>
<td>Added example CLI command and best practices recommendations for the new list routing controls API operation included in the extremely reliable Route 53 ARC data plane API. For more information, see <a href="#">List and update routing controls and states</a>.</td>
<td>March 31, 2022</td>
</tr>
<tr>
<td>Added support for overriding safety rules</td>
<td>Added support for overriding safety rules, which allows you to bypass routing control safeguards that are enforced with safety rules that you've configured. Safety rule overrides could be required, for example, in a “break glass” scenario during failover for disaster recovery. For more information, see <a href="#">Override safety rules to reroute traffic</a>.</td>
<td>March 2, 2022</td>
</tr>
<tr>
<td>Added additional tagging support</td>
<td>Added support for tagging additional resources in Route 53 ARC, including clusters, control panels, routing controls, and safety rules. For more information, see <a href="#">Tagging in Amazon Route 53 Application Recovery Controller</a>.</td>
<td>December 20, 2021</td>
</tr>
<tr>
<td>Updated managed policy</td>
<td>Updated the AmazonRoute53RecoveryControlConfigReadOnly managed policy to add permission to list tags for a resource. For more information, see <a href="#">AWS managed policies for Amazon Route 53 Application Recovery Controller</a>.</td>
<td>December 20, 2021</td>
</tr>
<tr>
<td>Change</td>
<td>Description</td>
<td>Date</td>
</tr>
<tr>
<td>-------------------------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>------------------------</td>
</tr>
<tr>
<td>Added support for real-time alerts with EventBridge</td>
<td>Added support for EventBridge, which means that now you can add rules to get alerts and act on Route 53 ARC readiness check status changes, for example, when a status changes from READY to NOT READY. For more information, see <a href="#">Using Route 53 ARC with Amazon EventBridge</a>.</td>
<td>December 20, 2021</td>
</tr>
<tr>
<td>Added routing control state code samples</td>
<td>Added code samples to illustrate trying cluster endpoints in sequence when you use API operations to get or update routing control states. For more information, see <a href="#">API examples for Amazon Route 53 Application Recovery Controller</a>.</td>
<td>November 16, 2021</td>
</tr>
<tr>
<td>Added new permissions to a read-only policy</td>
<td>Added two new permissions to the policy AmazonRoute53RecoveryReadinessReadOnlyAccess: route53-recovery-readiness:GetArchitectureRecommendations and route53-recovery-readiness:GetCellReadinessSummary. For more information, see <a href="#">AWS managed policies for Amazon Route 53 Application Recovery Controller</a>.</td>
<td>November 9, 2021</td>
</tr>
<tr>
<td>Added support for Amazon API Gateway resource type</td>
<td>Added a new resource type, Amazon API Gateway, and updated the Route 53 ARC service-linked role permissions so that Route 53 ARC can audit API Gateway with readiness checks. For more information, see <a href="#">Readiness rules and supported resource types</a> and <a href="#">Using service-linked roles for Route 53 ARC</a>.</td>
<td>October 28, 2021</td>
</tr>
<tr>
<td>Change</td>
<td>Description</td>
<td>Date</td>
</tr>
<tr>
<td>--------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>---------------------</td>
</tr>
</tbody>
</table>
| Added support for Lambda functions resource type | Added a new resource type, Lambda functions, and updated the Route 53 ARC service-linked role permissions so that Route 53 ARC can audit Lambda functions with readiness checks.  
For more information, see [Readiness rules and supported resource types](#) and [Using service-linked roles for Route 53 ARC](#). | October 8, 2021     |
| Added links to CloudFormation and Terraform templates | Added links to downloadable AWS CloudFormation and Hashicorp Terraform templates to help you quickly get started with using Route 53 ARC. For more information, see [Recovery readiness with a new application](#). | September 13, 2021  |
| Added new managed policies | Added the following AWS managed policies for Route 53 ARC:  
AmazonRoute53RecoveryReadinessFullAccess,  
AmazonRoute53RecoveryReadinessReadOnlyAccess,  
AmazonRoute53RecoveryClusterFullAccess,  
AmazonRoute53RecoveryClusterReadOnlyAccess,  
AmazonRoute53RecoveryControlConfigFullAccess, and  
AmazonRoute53RecoveryControlConfigReadOnlyAccess.  
For more information, see [AWS managed policies for Amazon Route 53 Application Recovery Controller](#). | August 18, 2021     |
| Started tracking AWS managed policies for Amazon Route 53 Application Recovery Controller | Updates for managed policies will be tracked from the initial release date forward.  
For more information, see [AWS managed policies for Amazon Route 53 Application Recovery Controller](#). | July 27, 2021       |
<table>
<thead>
<tr>
<th>Change</th>
<th>Description</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial release of Amazon Route 53 ARC</td>
<td>Route 53 ARC improves application availability by centrally coordinating failovers within an AWS Region or across multiple Regions. Route 53 ARC provides readiness checks to ensure that your applications are scaled to handle failover traffic and configured to route around failures. It also provides extremely reliable routing control so that you can recover applications by rerouting traffic, for example, across Availability Zones or Regions. For more information, see What is Route 53 ARC?</td>
<td>July 27, 2021</td>
</tr>
</tbody>
</table>
AWS Glossary

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