



TECHNICAL WHITE PAPER

How It Works: Cloud-Native Protection for Amazon EC2

Bill Gurling
March 2021
RWP-0512

TABLE OF CONTENTS

3 INTRODUCTION

- 3 Audience
- 3 Objectives
- 3 Challenges
- 4 The Rubrik Approach
- 4 Key Features

6 ARCHITECTURE AND COMPONENTS

- 6 High Level Architecture
- 7 Components
- 7 How Cloud-Native Protection for AWS Works
 - 8 Authorize
 - Adding AWS Accounts
 - Pre-sharing KMS Keys
 - 12 Configure
 - SLA Domains and Cloud-Native Protection for AWS
 - 12 Protect
 - Account Level SLA Assignment
 - Tag Rules
 - Direct SLA Domain Assignment
 - Disk Exclusion
 - On Demand Snapshots
 - File Indexing

17 HOW IT WORKS

- 17 Backup
 - 18 Volume level crash consistency
 - 18 Instance level crash consistency
- 20 File Indexing
- 22 Cross-Account Replication
- 24 Restore
- 25 Export
 - 26 EBS Volume Exports
 - 27 EC2 Instance Exports

28 SUMMARY

29 GLOSSARY

- Amazon Elastic Block Store (EBS) Snapshot
- Amazon Elastic Block Store (EBS) Volume
- Amazon Elastic Compute Cloud (EC2) Instance
- Amazon Elastic Kubernetes Service (EKS)
- Amazon Machine Image (AMI)
- Amazon Resource Name (ARN)
- Amazon Simple Notification Service (SNS)
- Amazon Web Services (AWS)
- AWS Availability Zones (AZ)
- AWS CloudFormation
- AWS Identity and Access Management (IAM) Policy
- AWS Identity and Access Management (IAM) Role
- AWS Key Management Service (KMS)
- AWS Regions
- Customer AWS Account
- Exocompute
- Relic
- Rubrik AWS Account
- Rubrik Polaris
- Service Level Agreement (SLA) Domain

32 APPENDIX A - AWS TAGS

36 APPENDIX B - METADATA

37 VERSION HISTORY

INTRODUCTION

Welcome to *How It Works: Cloud-Native Protection for Amazon EC2*. The purpose of this document is to aid the reader in familiarizing themselves with the features, architecture, and workflows of Rubrik's Cloud-Native Protection for AWS. Such information will prove valuable while evaluating, designing, or implementing the technologies described herein.

AUDIENCE

This guide is for anyone who wants to better understand the capabilities of Cloud-Native Protection for AWS on Rubrik's Polaris platform and the technical architectures that underpin those capabilities. This includes architects, engineers, and administrators responsible for AWS infrastructure and data protection operations as well as individuals with a vested interest in security, compliance, or governance.

OBJECTIVES

The goal of this guide is to provide the reader with a clear and concise point of technical reference regarding architecture and workflows utilized by Cloud-Native Protection for AWS on Rubrik Polaris. After reading this document, the reader should be able to answer the following questions regarding Cloud-Native Protection for AWS:

- *What does Cloud-Native Protection for AWS do?*
- *What problem(s) does the Cloud-Native Protection for AWS solve?*
- *How does one configure and utilize Cloud-Native Protection for AWS?*
- *How is Cloud-Native Protection for AWS architected? Why?*
- *How does Cloud-Native Protection for AWS operate?*
- *How does Cloud-Native Protection for AWS compare to alternate solutions?*

CHALLENGES

Digital enterprises are increasingly using multiple private and public clouds to deploy applications, avoid vendor lock-in, and exploit best-of-breed solutions. However, this fragments data within clouds, as well as across hybrid and multi-cloud infrastructures, fracturing IT's ability to protect, manage, and secure their data, operations, and business.

Public cloud providers themselves are responsible for the protection and availability **of** the cloud, however it is still the customer's responsibility to protect resources in the cloud. What this means, practically speaking, is that it is ultimately the customer's responsibility to protect their applications and data running in a public cloud, regardless of provider. The [Shared Responsibility Model](#) published by AWS is a great point of reference for these concepts.

This leaves the customer at a critical decision point – *How do I efficiently and reliably protect my assets that reside in the cloud?* While the question seems simple on its face, it is in fact quite complex.

In hybrid or multi cloud environments, customers might be inclined to lift and shift legacy tooling into the cloud. Unfortunately, this approach often hampers the agility and elasticity that enterprises are seeking when adopting a cloud strategy.

The alternative, leveraging platform native tooling from the cloud provider themselves can be similarly flawed as this segments data protection operations between public cloud providers as well as between public and on-premises environments. Such an approach leads to significant headwinds in terms compliance, visibility, and operational efficiency.

The need for an alternate approach is clear.

THE RUBRIK APPROACH

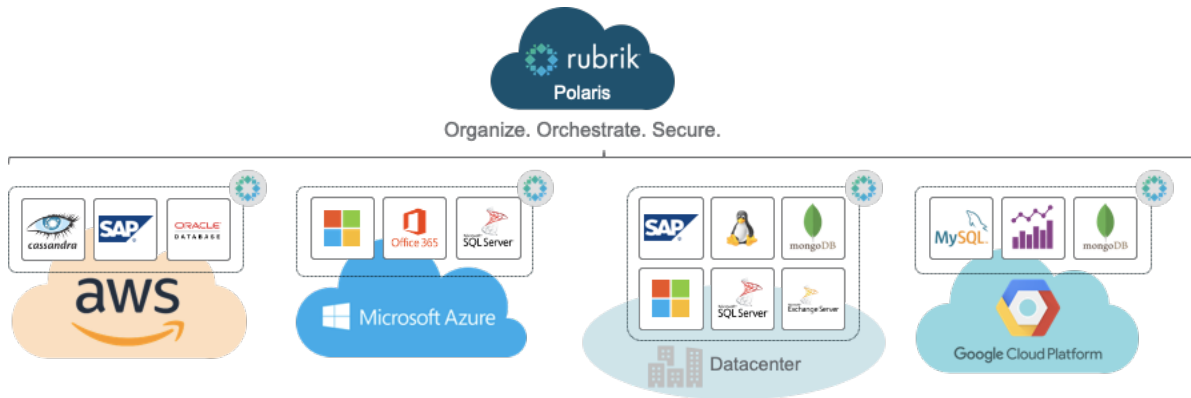


Figure 1 - Rubrik Polaris Multi-cloud Protection

Rubrik's goals are to collapse the footprint of legacy data protection solutions, simplify and automate backup and recovery using policy-based protection, minimize data loss, and to streamline operations. Rubrik's Cloud-Native Protection for AWS eliminates painful scripting and manual job scheduling. Cloud-Native Protection for AWS is a Software as a Service (SaaS) based data protection platform that provides automated backup, recovery, and replication schedules across regions, and even across clouds with a single global policy engine. It also allows the customer to quickly find and recover snapshots or files with predictive search. This solution allows Rubrik customers to reap the benefits of rapid innovation and reduced management complexity with data protection delivered as a service.

Protecting AWS workloads with Rubrik Polaris consists of 3 primary steps.

STEP	DETAIL
Authorize	Authorize Rubrik Polaris to access the AWS Account(s) that require protection via an AWS CloudFormation integrated workflow that aligns with AWS security best practices.
Configure	Use a single, declarative SLA policy engine to automatically create and expire AWS snapshots to suit backup and replication requirements. If file recovery is needed, Rubrik automatically spins up ephemeral compute in the cloud which is powered down once complete, minimizing consumption of cloud compute resources.
Protect	Assign SLA policies to the instances and volumes that require protection. Automatic protection based on account membership or tag values ensures workloads are protected when they are provisioned. Recover files, volumes, or instances rapidly through the Polaris UI or API. Polaris acts as a single pane of glass for hybrid or multi-cloud deployments.

KEY FEATURES

The key features of Rubrik's Cloud-Native Protection for AWS include:

- Unified data management across regions, accounts, private, and public cloud platforms
- Automated global data protection via Polaris SLA policies
- Rapid recovery for instances, volumes, and files within or across regions

UNIFIED DATA MANAGEMENT ACROSS ACCOUNTS AND CLOUD PLATFORMS

Single Point of Management and Automation via Rubrik Polaris – Rubrik’s Polaris SaaS platform is a single point of management and automation for hybrid and multi-cloud environments. It requires no persistently running instances or compute in the customer’s AWS environment. Polaris provides a simple, homogenous end-user data management experience across platforms and reduces the drag associated with legacy tooling and point solutions.

Consolidated Reporting – Easily track SLA policy assignment, protection and recovery activity, and SLA policy compliance across accounts, platforms, and clouds from a single easy to use reporting engine.

AUTOMATED GLOBAL DATA PROTECTION VIA POLARIS SLA POLICIES

SLA Domains – In the data protection world, Service Level Agreements (SLAs) define protection levels for workloads, availability targets, and objects that are crucial to a company. Collecting this information, implementing it, and staying compliant with the SLA is usually a tedious and difficult process. Rubrik uses global SLA Domains, a declarative, policy-driven framework, to make achieving your SLAs easier. When using an SLA Domain for Cloud-Native Protection for AWS, it is comprised of three components:

- Snapshot Frequency
- Snapshot Retention
- Replica Region and Duration

Global Protection – In Polaris SLA Domains can be assigned across object types. Even if those objects are spread across clouds, accounts, or on-premises. This allows one set of policies to be used to manage data wherever it may be in the environment.

Account Level Auto-Protection – Assign SLA policies to entire AWS accounts and ensure that every instance provisioned into the protected regions in those accounts receives the required level of data protection without the need for explicit SLA assignment. Account level SLA Domains can be overridden using tag-based assignment or by directly assigning SLA Domains to instances.

Tag-Based Auto-Protection – Allows for the assignment of SLA policies to instances or volumes whenever a specific tag key, or key value pair is found on any instance or volume in any AWS account in scope. These tag rules allow customers to leverage existing provisioning and governance logic to apply the appropriate SLA Domains across AWS accounts and regions without the need for manual intervention.

Filesystem Indexing – Customers can enable file recovery on the desired instances and volumes. When this feature is enabled Rubrik Polaris will index all snapshots of these assets using Rubrik’s Exocompute engine built on top of Amazon Elastic Kubernetes Service (EKS). This ephemeral compute engine runs only when there is indexing work to be done in order to maintain cost efficiency.

SLA Driven Snapshot Replication Across Accounts and Regions – Allows a customer to define a target account, region, and desired retention in an SLA Domain. Polaris will replicate any snapshots created by that SLA Domain to the target and expire them after the specified retention period has elapsed.

RAPID RECOVERY FOR INSTANCES, VOLUMES, AND FILES WITHIN OR ACROSS REGIONS

Amazon Elastic Block Store (EBS) Volume Export – Create a new encrypted or unencrypted EBS volume from the selected volume snapshot using either the original snapshot in the source account and region, or a replica in another. This workflow allows the user to define the volume type, size, account, region, KMS key, and availability zone. Tags that were on the source volume at the time of the snapshot can also be included or excluded from the export process.

Amazon Elastic Compute Cloud (EC2) Instance Restore – Select a point in time and automatically roll EC2 instance back to a known good point in time, preserving the `InstanceId` and private IP address. This operation ensures that volumes

are restored in a crash-consistent manner to the desired point in time, attached to the right mount points, and that tags are restored as well, if desired.

EC2 Instance Export – Create a new EC2 instance from the selected snapshot using either the original snapshot in the source account and region, or a replica in another. This workflow allows the user to define the instance name, type, size, region, KMS key (optional, used to encrypt instance volumes), VPC, subnet and security groups. Tags that were on the source instance at the time of the snapshot can also be included or excluded from the export process.

File/Folder Download – Search in and across instance or volume snapshots for specific files or folders and download the desired object(s) from the chosen snapshot to an existing S3 bucket or have Polaris create one on demand for the recovery.

ARCHITECTURE AND COMPONENTS

HIGH LEVEL ARCHITECTURE

Cloud-Native Protection for AWS allows Rubrik customers to take advantage of the power of the Rubrik’s SLA policy engine via the Polaris SaaS platform to protect workloads inside of their AWS accounts. This allows customers to experience the power of Rubrik without the need to deploy, manage, or patch any long running compute instances in the customer’s AWS environment nor on-premises. In order to accomplish this, Polaris leverages the native snapshot and image creation APIs provided by AWS to backup and recover EC2 instances, EBS volumes, and individual files and folders from within EC2 instances and volumes.

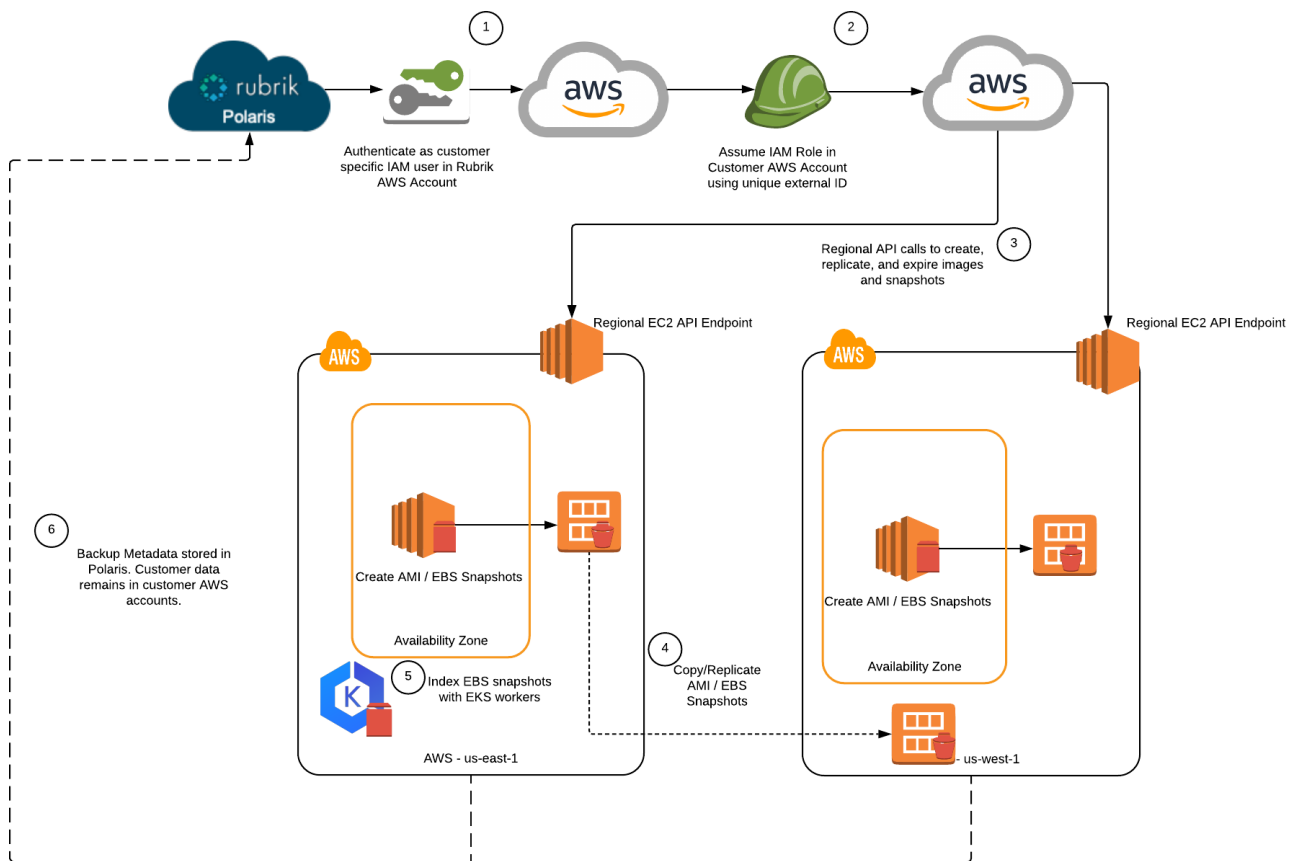


Figure 2 - High Level Architecture: Cloud-Native Protection for AWS

At a high level, the following workflow is used when protecting EC2 resources in a protected customer owned AWS Account using Rubrik Polaris.

1. Polaris authenticates into a Rubrik owned AWS account using a customer specific AWS Identity and Access Management (IAM) user. The credentials for this IAM user are stored in an encrypted format within a customer specific database.
2. Polaris assumes an IAM role that is pre-created by CloudFormation when the customer enables Cloud-Native Protection for their AWS account. This role has the necessary permissions to protect and restore EC2 assets in the customer's environment.
 - a. In order to utilize this role, AWS requires that API calls come from the trusted Rubrik AWS account.
 - b. Additionally, these API calls must include an external ID that is specific to the AWS account being protected. This external ID is encrypted, and securely stored in Polaris, outside of the trusted Rubrik owned AWS account.
3. Polaris calls the regional API endpoints in the customer's AWS account to create Amazon Machine Images (AMIs) and EBS snapshots in accordance with the frequency and retention defined in the SLA Domain policies assigned to these assets in Polaris.
4. If replication is configured in the SLA Domain, these images and snapshots are copied to the target AWS account and region defined as a replication target in the corresponding SLA. They remain in the remote region until the replication duration has elapsed, at which point they are expired.
5. If file indexing is enabled on the objects being protected, Rubrik's Exocompute engine leverages Amazon EKS to generate filesystem indexes as needed.
6. Snapshot and index metadata is securely stored in Polaris.

COMPONENTS

The solution depicted in *Figure 2* consists of many AWS and Rubrik components. Let's discuss each in more detail as well as describe its role within Cloud-Native Protection for AWS prior to delving deeper into the architectures and associated workflows.

Cloud-Native Protection for AWS is configured and managed via Rubrik's [Polaris SaaS platform](#). When protecting AWS assets, Polaris assumes an [IAM Role](#) from a [Rubrik owned AWS account](#) into the [customer owned AWS account](#) being protected, granting Polaris the permissions required to protect [EC2 instances](#) and [EBS volumes](#) in the customer's account. Polaris leverages the EC2 API endpoint in each of the protected AWS [region\(s\)](#) to create [AMIs](#) and [EBS snapshots](#) in accordance with the [SLA Domain policy](#) applied. These images and snapshots are created, indexed (if desired), and expired automatically by Polaris. If required, filesystem indexing and file level recovery is facilitated by the use of the [Amazon EKS](#). Only [metadata](#) is sent to Polaris; the AMIs and EBS snapshots that contain customer data reside solely in the customer owned AWS account.

HOW CLOUD-NATIVE PROTECTION FOR AWS WORKS

As stated previously in this document, protecting AWS workloads with Rubrik Polaris consists of 3 primary steps: *Authorize*, *Configure*, and *Protect*. This section of the document dives deeper into each of these steps both operationally and architecturally. This should leave the reader with a basic familiarity of how Cloud-Native Protection for AWS on Rubrik Polaris is architected, configured, and utilized.

AUTHORIZE

ADDING AWS ACCOUNTS

Authorizing Polaris to protect workloads in AWS is a very simple process in terms of execution. The customer clicks the **Add Cloud Account** button in the **Cloud Accounts** section of **Remote Settings** in Polaris. The customer selects **AWS**, then **EC2 and EBS Protection** and enters the **AWS Account ID** for the account requiring protection and an **Account Name** to be displayed in the Polaris console. A simple wizard then walks the customer through launching a **CloudFormation** Stack to create the service principles and permissions required to protect this account with Cloud-Native Protection for AWS. Let's explore how this workflow operates under the covers.

In order to protect workloads running in AWS, Rubrik Polaris needs a means by which to interact with the customer's AWS account(s). As stated prior, Polaris leverages the native snapshotting and image creation capabilities of AWS in order to backup, replicate, and restore the assets it is protecting. These capabilities are available via the AWS API to which access is controlled by the **AWS Identity and Access Management (IAM) service**. The IAM service itself is quite powerful and supports a variety of service principals such as users, groups, federated users and groups, as well as roles. Permissions are delegated to or revoked from these service principals via associated IAM Policies. Rubrik Polaris leverages IAM roles for EC2 native protection.

The figure below depicts how the workflow interacts with a customer's account from an AWS IAM perspective.

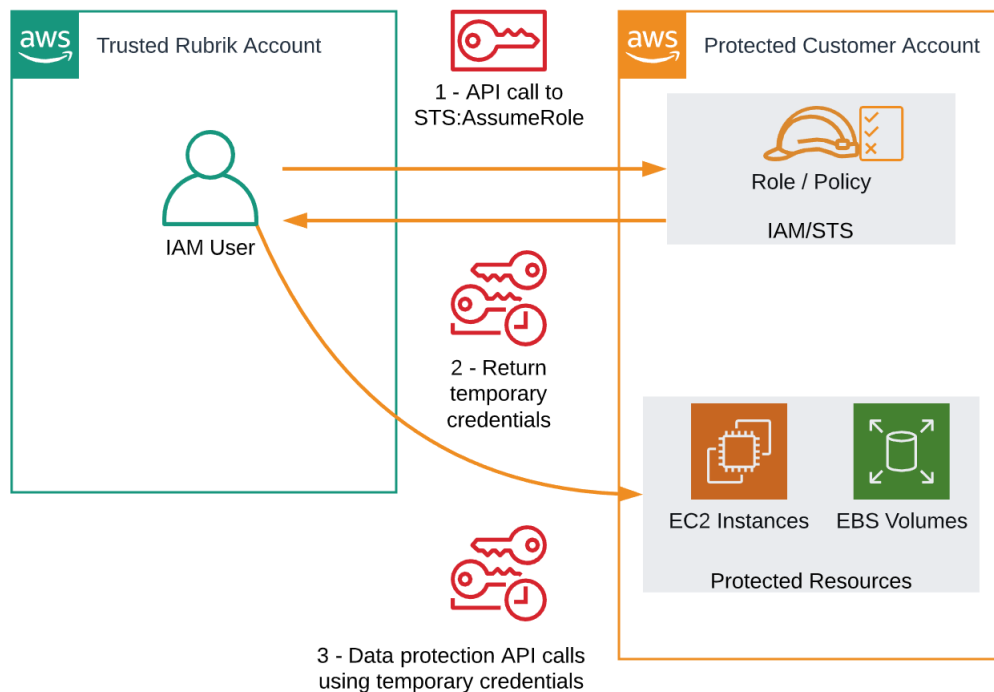


Figure 3 - Polaris AssumeRole Workflow

Roles are an identity that AWS customers can create in AWS and assign permissions to. Roles can be utilized only by trusted AWS services or accounts. These roles are **assumed** from a trusted Rubrik owned AWS account via the **AWS Simple Token Service (STS)** API and allow Rubrik to interact with resources in the customer's environment without the need for long lived static credentials. Leveraging roles for this type of workflow is in line with **AWS best practices**. The alternative, leveraging IAM Access keys assigned to a user, is a significantly less desirable approach which unfortunately, is still frequently employed by many. This is due to the fact that the Access keys are long lived credentials that are not restricted to a trusted entity. If these keys leak, they can be used by anyone, from anywhere.

The role and policy indicated above have to be created and maintained. To accomplish this, Polaris leverages AWS CloudFormation. AWS CloudFormation allows AWS customers to model infrastructure and application resources with templates written in formats like JSON or YAML. These templates can be submitted to the CloudFormation service on AWS to create a collection of resources known as a stack. These stacks and their life cycles are managed via CloudFormation itself rather than individually in AWS. Update the template in CloudFormation and the stack will be updated accordingly, delete the stack and all resources provisioned as part of the stack will be deleted as well.

Once the role is built in the customer account, Polaris needs its [Amazon Resource Name \(ARN\)](#) in order to utilize it when interacting with the newly added AWS account. This value is securely transmitted back to Rubrik using [Amazon's Simple Notification Service \(SNS\)](#) thus completing the workflow and allowing Rubrik secure access to the customer's AWS account for data protection purposes. The diagram below depicts this provisioning process.

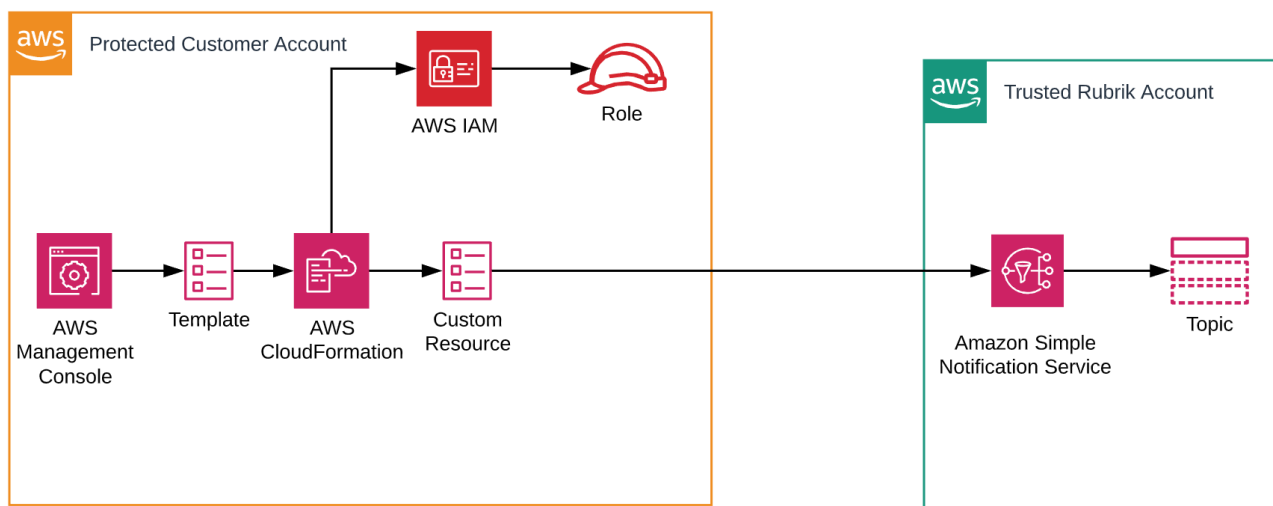


Figure 4 - Polaris IAM Role Provisioning Process

Once this process is complete Polaris knows the necessary permissions are in place and has all the information needed to begin protecting the AWS account. Examples of the CloudFormation templates and IAM policies used during this process are [available for reference on GitHub](#).

Another benefit of this method is that if these permissions need to be modified in the future, Polaris can prompt the user to walk through updating the resources via CloudFormation. As an example, this update process is used when users enable file indexing because Polaris needs additional permissions to use Amazon EKS. More detail on this process is available in the *How It Works* sections covering Backup and File Indexing.

PRE-SHARING KMS KEYS

Cloud Native Protection for AWS offers customers the ability to replicate snapshots from one AWS account to another. This functionality requires that customers pre-create and share certain KMS keys between the source and target account. The impetus for this requirement stems from the way KMS keys and EBS volume encryption work in AWS. By default, these keys are only accessible from within the account they were created in.

This becomes particularly important during restore in place operations when the original source key must be used to create a restored volume from a snapshot in another account and/or region. The issue is further complicated by the fact that the default KMS keys in each region cannot be shared across the account boundary. Luckily, Rubrik has abstracted away most of the complexity associated with these recovery workflows through automation. The only requirement is that the appropriate KMS keys are pre-shared with the appropriate accounts.

The diagram below depicts these requirements visually for an example environment where we are replicating instance snapshots from us-west-2 in Account A to us-east-2 in Account B. With this configuration, the environment has the necessary permissions to protect instances and volumes within the us-west-2 region of account A while replicating those snapshots to the us-east-2 region of Account B. The keys and shares depicted allow for replication and in place restore regardless of whether the source volumes were encrypted with the default KMS key or a Customer Managed KMS key at the source. More detail on the purposes of each key is provided below.

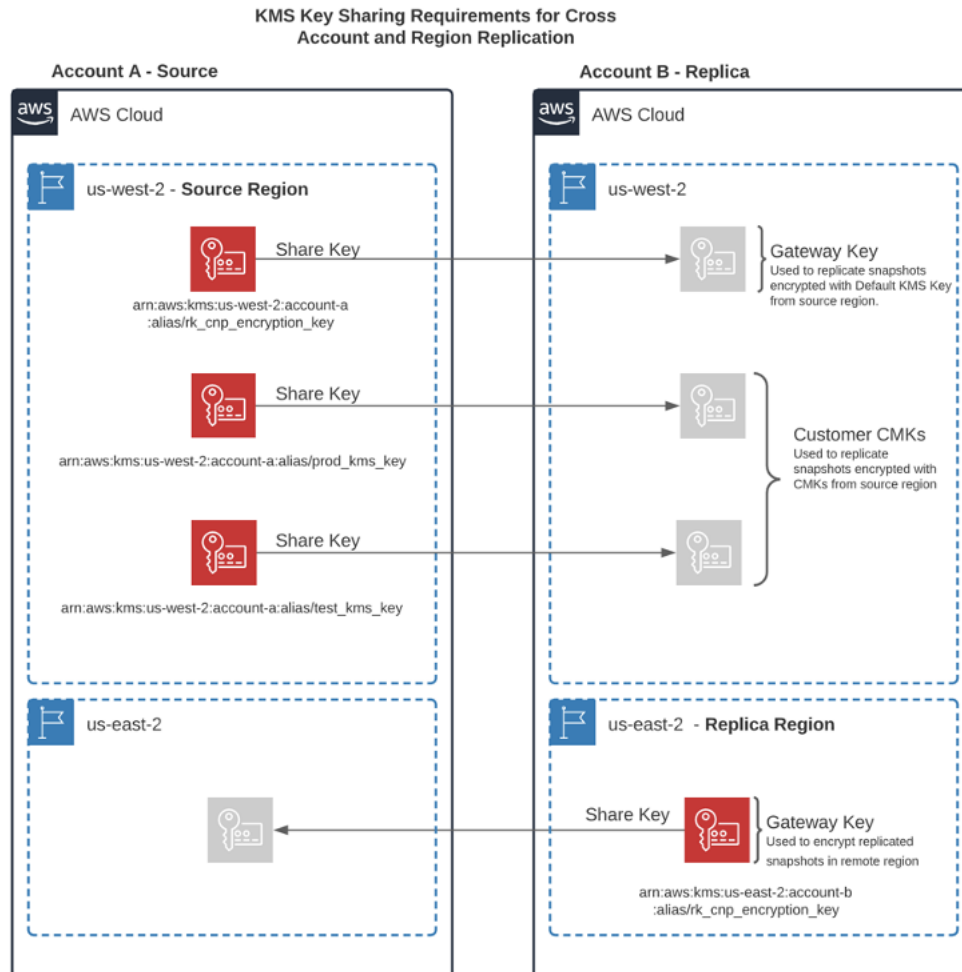


Figure 5 - KMS Key Pre-Share Purposes and Requirements

This configuration allows for the restore of instances and volumes in the source region of Account A using either the local images and snapshots in that region, or the replicas in the replica region of Account B. This logical topology is depicted in the diagram below.

Logical Configuration for Replication Example

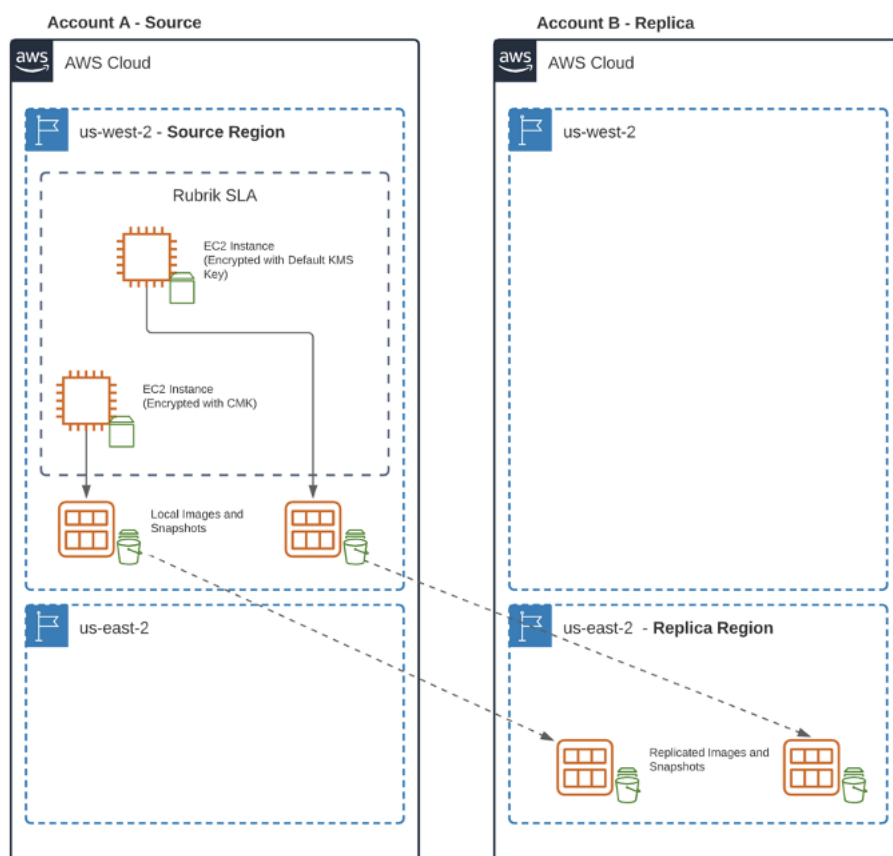


Figure 6 - Cross-Account Replication Logical Topology

Gateway Keys

Gateway keys are used for two purposes. 1) To encrypt source snapshots of volumes encrypted with the default KMS key in their source region so that they can be replicated to the target account. 2) To encrypt snapshots stored in the replica region at rest.

You can create a CloudFormation stack using the [CloudFormation Template](#) in this repository, and CloudFormation will create a Rubrik gateway key in that region where you create the stack. This means that that you will need to create two stacks for most use cases, one in the source region of the source account trusting the replica account, and one in the replica region of the replica account targeting the source account.

That said, in more complicated configurations, you may need to modify this template to accommodate all of the replication and export flows required. Additionally, if no volumes in the source region are encrypted with the default KMS key the gateway in the source region is not required.

Customer Managed Keys (CMKs)

Depending on your AWS configuration, some KMS Keys (specifically [Customer Managed Keys](#)) may need to be shared with the replica account from the region containing instances and volumes whose snapshots you wish to replicate. These keys are made available to the replica account so that source snapshots encrypted with these CMKs can be replicated across accounts. The template linked above does not handle the sharing of these CMKs for you, but there are example [Key Policies](#) you can apply to your existing CMKs in order to share them as required above. Any instance or volume encrypted with a CMK in the source region should have its CMK shared with the target account if replication will be enabled for that object's snapshots.

CONFIGURE

SLA DOMAINS AND CLOUD-NATIVE PROTECTION FOR AWS

Once an AWS account is added to Polaris, the next step is to create one or more SLA Domains in order to begin protecting workloads in AWS. These SLA Domains can then be applied to EC2 instances and EBS volumes at which time Polaris will begin creating, replicating, and expiring AMIs and/or EBS snapshots to protect the relevant workloads according to the parameters defined in the SLA Domain. SLA Domains are a powerful replacement to the job scheduling approach used by many traditional data protection solutions largely due to their declarative nature which generally maps very nicely to the RPOs and RTOs required by businesses.

Building an SLA for use with Cloud-Native Protection for AWS is a straightforward and simple process that is outside of the scope of this document. Please reference the Polaris User Guide for details on SLA creation within Polaris. That said, there are a few specific elements of SLA domains relative to Cloud-Native Protection for AWS that will be highlighted.

When creating a SLA domain within Polaris, the customer can select an **AWS Account and/or Region** that will be used as a replication target for the images and snapshots created via Cloud-Native Protection for AWS. If the customer selects **Source Region** when configuring cross-account replication, all snapshots protected by the SLA will be replicated to the target account in the same region as their source. The customer can also define the **Retention** for the replicas in this target target. Both the source and destination accounts and regions must be configured for protection in order to enable replication. The required KMS assets must also be in place as described in the Authorize section of this document. The customer can verify account and regional protection by viewing the relevant accounts in **Remote Settings** under **AWS Native Protection**.

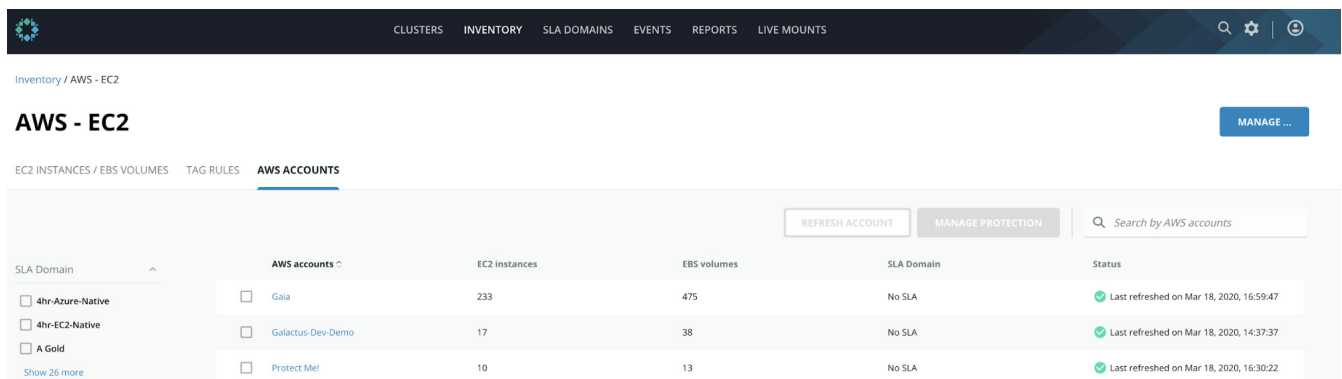
At this point in time the **Archiving** section does not apply to Cloud-Native Protection for AWS, The customer can leave this parameter unconfigured in SLA Domains that will be utilized exclusively for protecting AWS workloads. Please note, if the customer configures a parameter not applicable to Cloud-Native Protection for AWS, a warning will be displayed when assigning that SLA Domain policy to incompatible assets.

PROTECT

ACCOUNT LEVEL SLA ASSIGNMENT

Once all the required AWS accounts have been added to Polaris and the desired SLA Domains have been created, it's time to begin protecting workloads in AWS. Cloud-Native Protection for AWS has a few different options with regards to assigning SLA Domains to EC2 workloads and each of them provides a unique business benefit when employed properly.

Upon navigating to the **Inventory** screen in Polaris and selecting **AWS - EC2**, users are presented with an inventory of all EC2 instances that Polaris is currently capable of protecting. Selecting the **AWS Accounts** tab will present a view with a list of all AWS accounts added to Polaris and some relevant information on each account.



SLA Domain	AWS accounts	EC2 instances	EBS volumes	SLA Domain	Status
<input type="checkbox"/> 4hr-Azure-Native	<input type="checkbox"/> Gaia	233	475	No SLA	✓ Last refreshed on Mar 18, 2020, 16:59:47
<input type="checkbox"/> 4hr-EC2-Native	<input type="checkbox"/> Galactus-Dev-Demo	17	38	No SLA	✓ Last refreshed on Mar 18, 2020, 14:37:37
<input type="checkbox"/> A Gold	<input type="checkbox"/> Protect Me!	10	13	No SLA	✓ Last refreshed on Mar 18, 2020, 16:30:22

Figure 7 - Cloud-Native Protection for AWS Inventory

From this view the customer can select one or more accounts using the corresponding checkboxes and use the **Manage Protection** button to modify the SLA Domain assigned to that account. SLA Domains assigned at the account level will

automatically inherit down to all EC2 instances in all protected regions unless another SLA Domain is assigned via a tag rule or directly to an instance itself. This is a useful technique to automatically protect all instances provisioned into an AWS account with the desired SLA Domain. As an example, one might assign a fairly lightweight SLA Domain to a developer account so that all assets in that account are recoverable regardless of how they were provisioned.

SLA Domains with a **yellow exclamation point** over them in the SLA Domain selection window have some configuration that is incompatible with Cloud-Native Protection for AWS for the account(s) that have been selected. Examples include an archival configuration within the SLA Domain, or replication configured to a region that is not enabled on the selected AWS account(s). Hover over the warning icon to receive a description of the warning.

In addition to assigning an SLA Domain to the account, this dialogue box also allows the customer to select **Clear Existing Assignment** or **Do Not Protect**. Both of these configurations produce a similar result when utilized at the AWS account level. The selected accounts and instances within them would be unprotected unless an SLA Domain was assigned to them using another means.

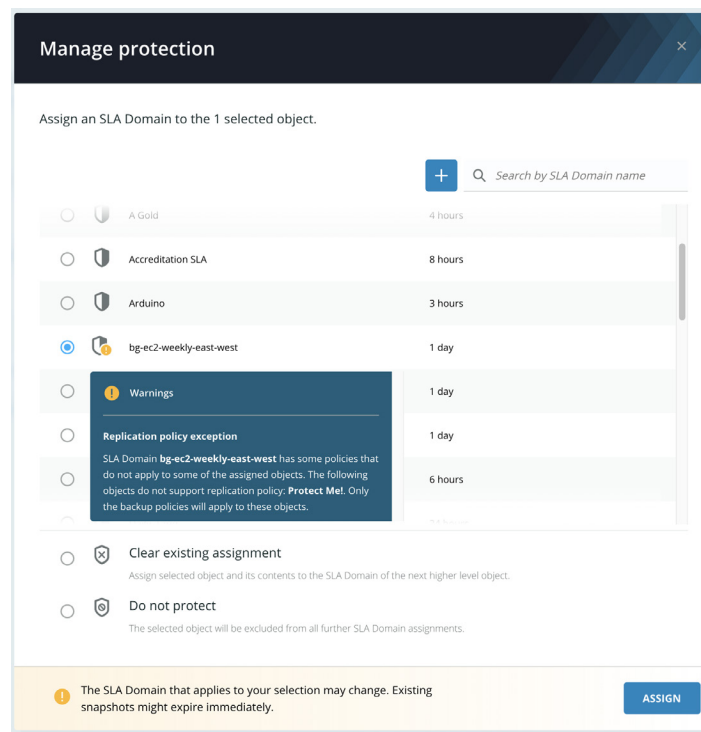


Figure 8 - Cloud-Native SLA Assignment Warning

TAG RULES

Tag Rules are a powerful construct that allow Cloud-Native Protection for AWS to leverage existing business logic and provisioning workflows to assign SLA Domains to the appropriate resources. This is accomplished by mapping an SLA Domain to an AWS Tag or to an AWS Tag / Value Pair. These rules can be scoped across any number of AWS regions and accounts and are applied to either EC2 instances or EBS volumes. This is an extremely simple, powerful, and lightweight mechanism for protecting EC2 instances and EBS volumes in large multi-account AWS environments.

When creating a tag rule, selecting (**All tag values**) in the value field will cause the rule to match either all instances or volumes (depending on the previous selection) with the specified **Tag Key** regardless of the value. Similarly, selecting (**No tag value**) will cause the rule to apply only when a matching **Tag Key** is found without a **Tag Value**.

Tag rules also require that the customer selects the SLA Domain to assign when the tag rule has a match. This SLA Domain will be assigned to all matching objects, unless they have a specific SLA Domain assigned directly to them. Additionally, Polaris will periodically poll the customer's AWS Account(s) and update the assigned SLA Domains in accordance with these

tag rules. This includes updated tag keys or values, newly provisioned EC2 instances or EBS volumes that match the rule, or modifications to the tag rule itself.

Lastly, leveraging **Do Not Protect** within a tag rule can oftentimes be useful for excluding specific workloads from account level Auto-Protection. For example, a tag rule matching `rk_instance_class:TransientStormInstance` with a **Do Not Protect** SLA Domain selection will prevent Cloud-Native Protection for AWS from snapshotting the **Storm** instances that Rubrik uses for jobs like CloudOn.

Since EC2 resources can have multiple tags, Polaris may apply multiple tag rules to the same EC2 instance or volume. If the tag rules specify different SLA Domains, Polaris selects one of them based upon the following order of precedence, highest to lowest:

- Do Not Protect
- The SLA Domain with the most frequent snapshots
- The SLA Domain with the longest retention

DIRECT SLA DOMAIN ASSIGNMENT

Users can also directly assign SLA Domains to EC2 instances and EBS volumes from the **AWS – EC2 Inventory** screen within Polaris. The **View** dropdown can be used to flip between the EC2 instance and EBS volume inventory. The filter options on the left-hand side of this view are particularly useful for tailoring the view to the desired resources, as is the instance search box at the top right of the view. In order to determine the current SLA Domain assigned to an instance or volume, simply reference the **SLA Domain** column in this view. The source of that SLA Domain can be viewed in the **Assignment** column of the same view.

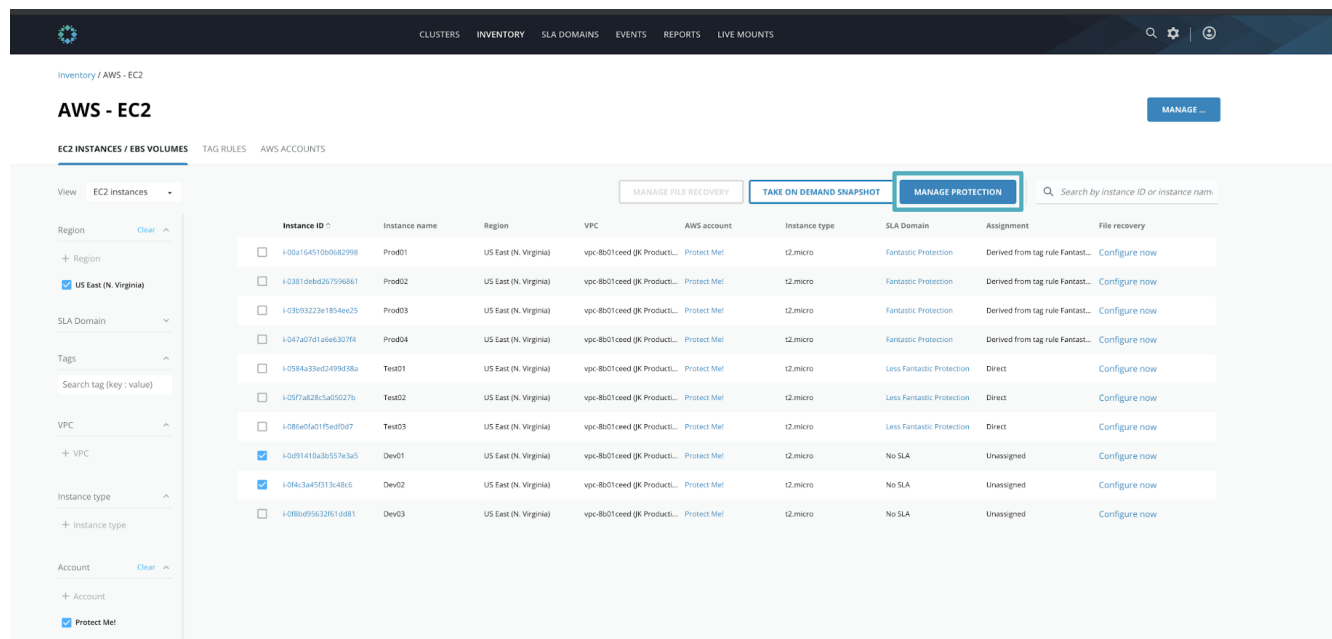


Figure 9 - Direct SLA Assignment

Ticking the checkboxes next to the desired instances or volumes and clicking **Manage Protection** will bring up the SLA Domain assignment dialogue box. This can also be accomplished by drilling down into an individual EC2 instance or EBS volume and clicking the same button. This view provides the option to select an existing SLA Domain as well as **Clear existing assignment**, which will remove any existing SLAs directly assigned to the object, and **Do Not Protect**, which forces Polaris to stop protecting the object.

The main difference between these two options is the fact that **Clear existing assignment** will allow SLA Domains assigned at the account level or via tag rules to inherit down to the selected object. **Do Not Protect** on the other hand, will force Rubrik not to protect the object regardless of any account level assignments or tag rules that might be in play. In general, direct SLA Domain assignment tends to be a last resort for overriding the SLA Domains inherited from account level SLA Domain assignment or tag rules as opposed to the primary means of assigning SLA Domains to EC2 instances or EBS volumes.

DISK EXCLUSION

Cloud-Native Protection for AWS on Polaris supports excluding volumes from protection when EC2 instance level backups are performed. This can be a useful approach in cases such as database servers, where the application may be backed up independent of the instance. In such cases, snapshotting the data volumes on these instances would offer little benefit and would result in increased data protection costs.

To exclude specific volumes attached to an instance from protection, the customer searches or navigates to the view for that instance, then clicks the ellipsis next to the **Take On Demand Snapshot** button. In the context menu that appears, select **Exclude Volumes**. In the dialogue box that appears volumes can be excluded and included in snapshots as needed. The root volume of an instance cannot be excluded.

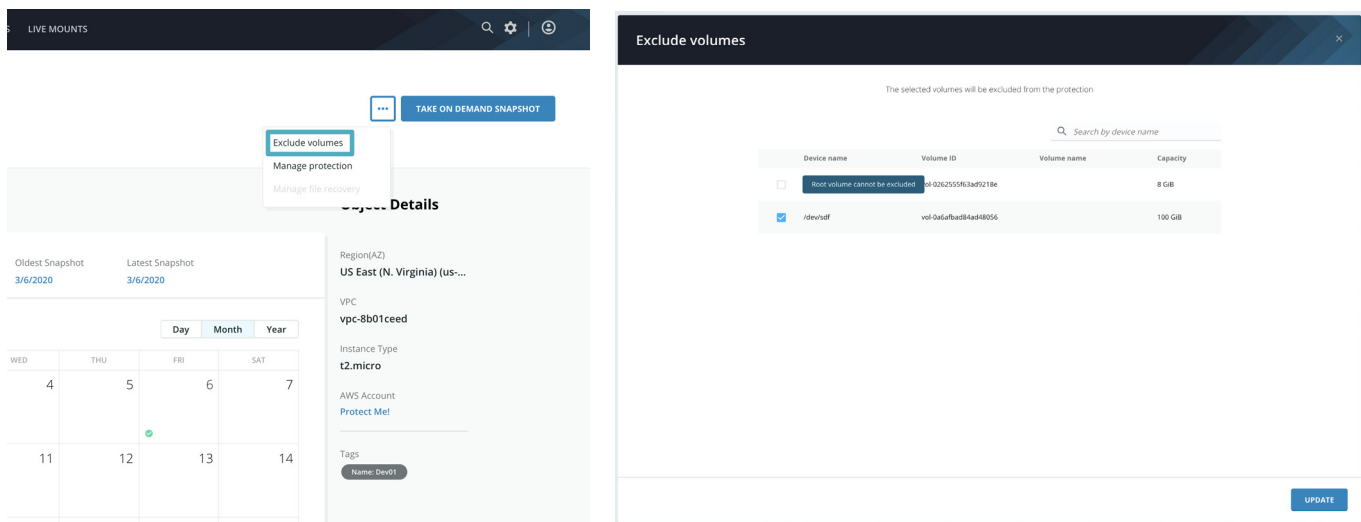


Figure 10 - Volume Exclusion

ON DEMAND SNAPSHOTS

On Demand snapshots are snapshots that are manually created and differ from SLA Domain created snapshots in a few significant ways. An On Demand snapshot is created when a user clicks the **Take On Demand Snapshot** button on the view for an EC2 instance or EBS volume. An on demand snapshot is not associated with an SLA, and thus, will be retained until it is deleted. Because of this, customers have the capability to delete On Demand snapshots via the Polaris console, they simply navigate to the On Demand snapshot and select **Delete** from the menu that appears when clicking the ellipsis next to the time of the corresponding snapshot. Only snapshots with a type of **On Demand** can be deleted by end-users. On Demand snapshots are a useful tool when employed prior to making any change inside of an instance that cannot be easily recreated through other means. Simply take an On Demand snapshot, perform the required task, and then delete the On Demand snapshot after confirming the change is non-breaking. On Demand snapshots can also be used in conjunction with scripting to pause applications prior to snapshotting an instance.

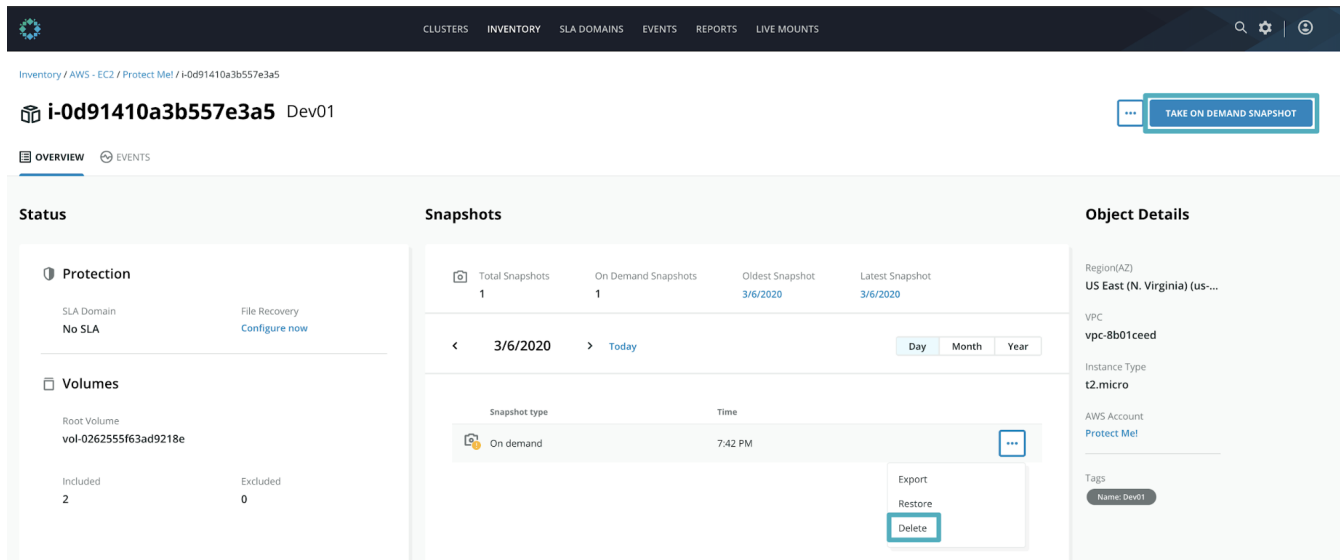


Figure 11 - On Demand Snapshot

FILE INDEXING

Cloud-Native Protection for AWS on Polaris allows customers to recover individual files and folders from the AWS images and snapshots it creates when protecting EC2 instances and EBS volumes. In depth detail on this process is available in the How It Works section on File Indexing. Filesystem indexing occurs after a snapshot is taken and leverages the Polaris [Exocompute](#) engine built on top of Amazon EKS to index the filesystems of the EBS volumes that Polaris is protecting. Exocompute must be configured for all accounts and regions where file level recovery is required. This is done in the **Exocompute Settings** tab of the **Remote Settings** console. The customer can click **Create Exocompute Setting** and select the appropriate AWS account. Polaris will prompt the customer to update the permissions assigned to Polaris' IAM Role via CloudFormation if necessary.

If prompted, the customer selects **Launch CloudFormation Stack**, logs into the appropriate AWS account, then launches the stack with the default parameters. Once **Update Stack** is run, Polaris will create the service principles and policies necessary to run Exocompute.

Once permissions are in place, the customer is prompted to select regions within the protected account that are in scope for File Indexing. Only [regions that support Amazon EKS](#) will be displayed in this view. For each region that is in scope, the customer will specify a VPC and two subnets for the EKS worker nodes to run in. These subnets must have internet access in order for Exocompute to communicate with Polaris. This can be accomplished through a variety of network architectures, the most common of which is placing the worker nodes in a private subnet with internet connectivity via a NAT gateway.

Additionally, the VPC utilized for the EKS worker nodes can be a VPC dedicated to EKS, or an existing shared services VPC of some sort. This design decision is based predominantly on customer preference. However, each EKS worker is allocated 30 private IP addresses for use by the EKS VPC Container Networking Interface Plugin (CNI). The CNI plugin is a Kubernetes plugin that enables pod networking using Elastic Network Interfaces on AWS. These IPs are reserved at runtime to reduce lag when scheduling pods on the worker node. This fact may lead some customers to prefer dedicated EKS worker VPCs in order to avoid contention around allocating IP addresses. **If in doubt, the customer should use a dedicated VPC for Exocompute to avoid private IP address exhaustion in their subnet(s).**

Once Exocompute is configured in the desired accounts and regions, instances and volumes in those regions will become eligible for file level recovery. The customer can enable the capability on the desired instances and volumes from the **Instances / EBS Volumes** tab in the **AWS – EC2 Inventory** screen by selecting **Enable file recovery**. Filtering by tag to narrow down the list of objects in this view can be particularly useful here. Once enabled, Polaris will index all existing and future snapshots of the selected objects and allow for file and folder recovery from within these snapshots once indexing is complete.

File or folder recovery is a simple process as well. Customers can search for files or folders across all snapshots by navigating to the protected instance or volume and using the **Search Files by Name** button. The wizard will walk the customer through selecting the file or folder to be recovered, the snapshot to recover it from, and the S3 bucket to recover it into. A new bucket can be automatically created if required. Customers can also browse or search within an individual snapshot by using the ellipses on the desired snapshot and selecting **Recover Files**. Once the process is complete, the customer simply retrieves the desired files from the specified S3 bucket.

HOW IT WORKS

BACKUP

Once accounts are added and SLA Domains are assigned, it's time to start protecting workloads in AWS. The Polaris job framework will begin automatically scheduling and snapshotting volumes and instances in accordance with the SLA Domains have been created and assigned without needing to schedule any jobs. Polaris handles batching all snapshot jobs so as not to overrun the API limits on AWS with one big batch of snapshot or replication activities. By default, Polaris will run a maximum of 20 snapshot jobs in parallel. Let's dig into the specifics of snapshotting an EBS volume, followed by that of EC2 instance snapshots.

EBS VOLUME SNAPSHOTS

When backing up individual EBS volumes, a `CreateSnapshot` API call is made to the appropriate regional EC2 API endpoint with the `VolumeId` of the source volume and the required tag values as parameters. Polaris will copy tags from the source volume to these snapshots, as well as apply some [Polaris specific tags](#) to the newly created snapshots. Snapshots that are taken from encrypted volumes are automatically encrypted. EBS volume snapshots are crash-consistent.

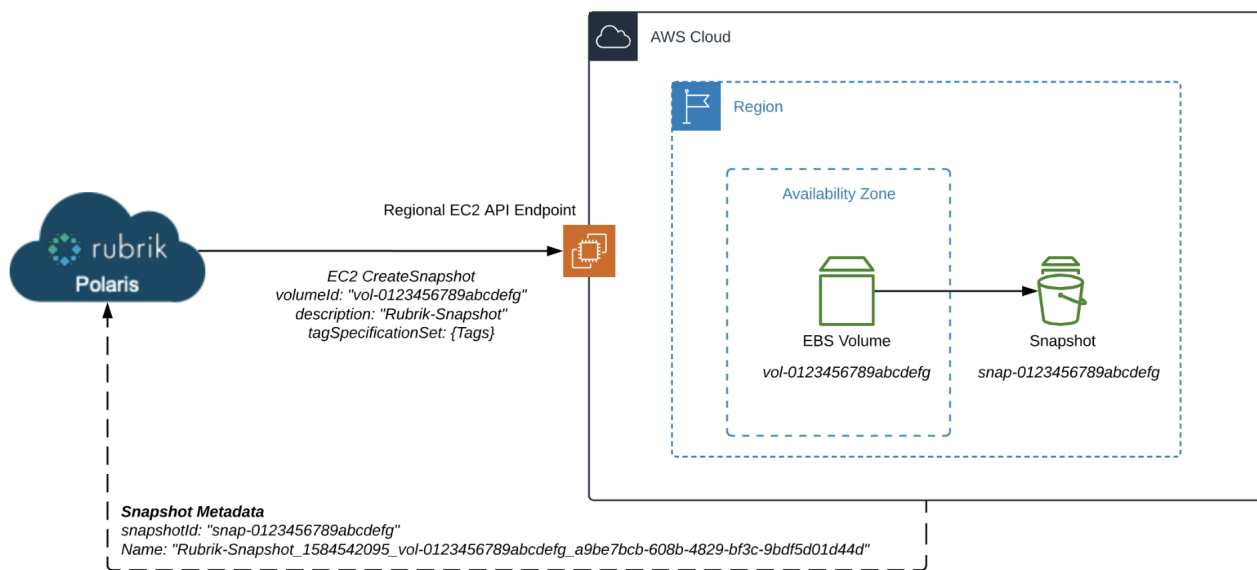


Figure 12 - EBS volume snapshot

EC2 INSTANCE SNAPSHOTS

Polaris EC2 Instance Snapshots are stored as a combination of an AMI and a set of EBS snapshots inside of AWS. The method of creation for these two assets varies based upon the type of instance backup being taken. Polaris supports *instance level crash consistency* and *volume level crash consistency* when protecting EC2 instances. The main difference between the two being the API calls used to create the final product.

The method employed is defined by a per customer feature flag that is configurable by Rubrik support. By default, *instance level crash consistency* is enabled on the EC2 instance snapshots created by Cloud-Native Protection for AWS in any region where aws supports creating these types of snapshots. Typically, this default is the best approach for protecting EC2 instances from a cost and performance perspective. However, if the customer has many instances with large EBS volumes that are excluded from EC2 instance backups and no need for multi-volume crash consistency it may be more cost effective to utilize *volume level crash consistency* instead.

VOLUME LEVEL CRASH CONSISTENCY

EC2 instance snapshots taken with *volume level crash consistency* are created using a process very similar to that of the EBS snapshot process described prior.

In this consistency model Polaris uses the AWS EC2 [CreateImage](#) API to create an AMI of the protected instance as well snapshots all non-excluded EBS volumes. When using this method, the EBS snapshots created will be individually crash-consistent. In other words, there is no guarantee that each volume's snapshot will be crash-consistent to the exact same point in time as the snapshots of the other volumes attached to the source instance when the image was created. Let's take a look at the specifics of this approach.

When creating a snapshot with this approach, a [CreateImage](#) API call is made to the appropriate regional EC2 API endpoint with the **InstanceId** of the source instance, a list of block devices to include in the snapshot (all that have not been specifically excluded in Polaris), and the required Tag values as parameters. The required tag values consist of tags that will be copied from the source instance to the newly created AMI, tags that will be copied from the source volumes to the newly created EBS snapshots, as well as a set of [Rubrik specific tags](#). Tags that conflict with the Rubrik specific tags will not be copied from the source objects. Snapshots that are taken of instances with encrypted volumes are automatically encrypted. The EBS volume snapshots are individually crash-consistent with this approach.

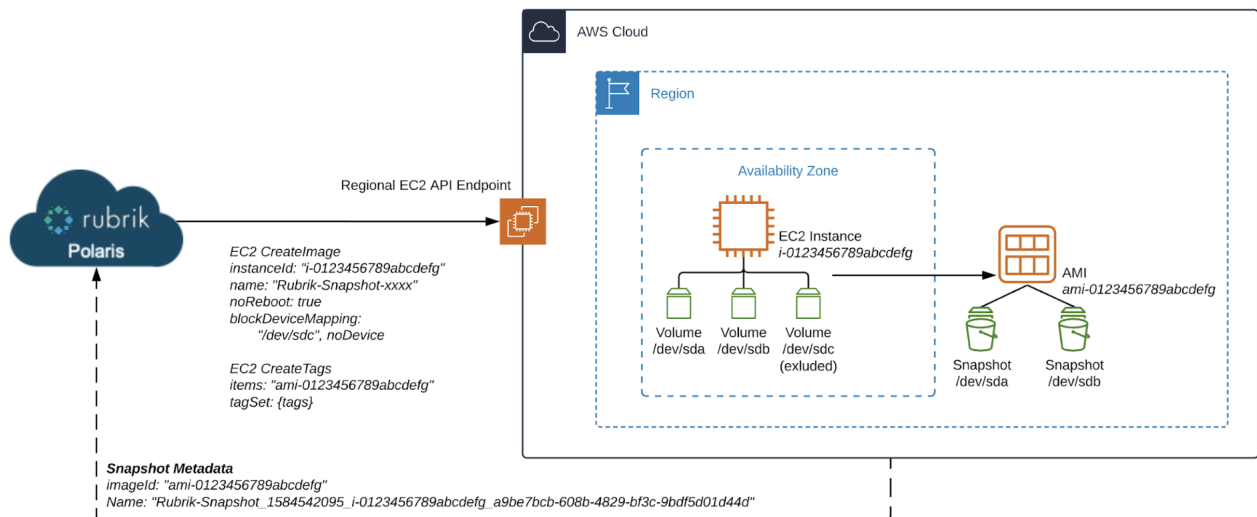


Figure 13 - EC2 Instance Snapshot via CreateImage

INSTANCE LEVEL CRASH CONSISTENCY

EC2 instance snapshots taken with *instance level crash consistency* are created using a multistage process different to that of the EBS snapshot or CreateImage based EC2 instance snapshot process described prior. The primary driver for this approach is the desire to support both multi volume crash consistency and disk exclusion for EC2 instance snapshots. Let's explore the process in depth to illuminate the value of Rubrik's approach.

This first phase of this process leverages a [CreateSnapshots](#) API call to the appropriate regional EC2 API endpoint with the `InstanceId` of the source instance and the required Tag values as parameters. As with the previously described approach, tags are copied from the source volumes to the EBS snapshots created here. Additionally the [Rubrik specific tags](#) are added, and any conflicting tags are not copied from their source. Snapshots that are taken of instances with encrypted volumes are automatically encrypted. The EBS volume snapshots are crash-consistent across EC2 volumes with this approach.

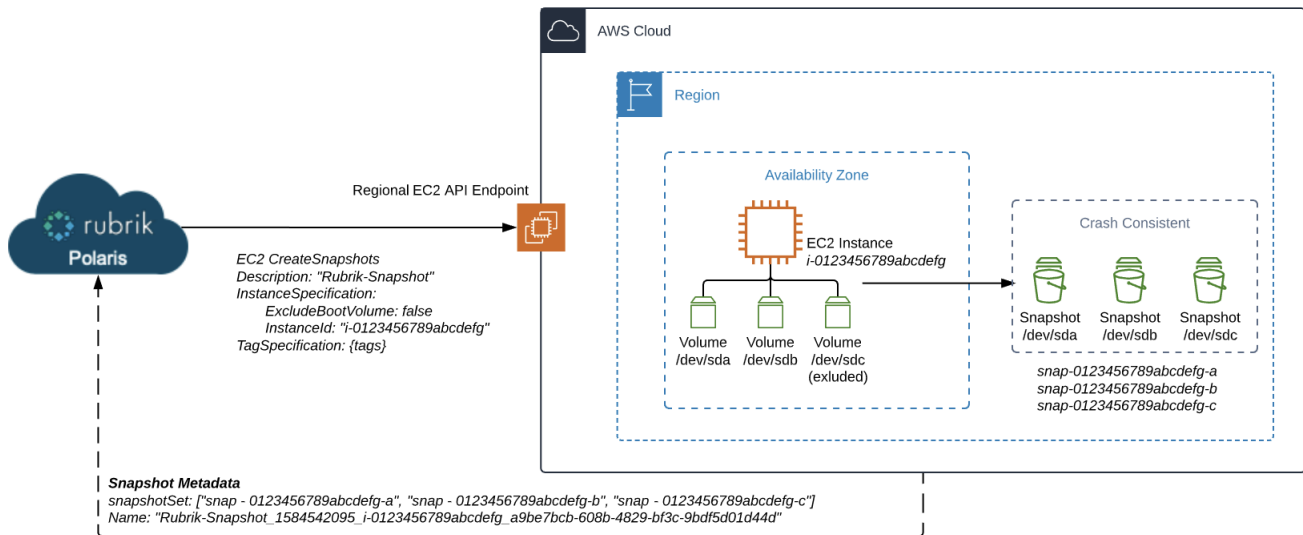


Figure 14 - Step 1: EC2 Instance Snapshot via CreateSnapshots

The next step leverages a [CreateImage](#) API call to the appropriate regional EC2 API endpoint with the `InstanceId` of the source instance, a list of block devices to include in the snapshot (root volume only), and the required tag values as parameters. As with the previously described approach, tags are copied from the source instance to the AMI created here. Additionally the [Rubrik specific tags](#) are added, and any conflicting tags are not copied from their source. Snapshots that are taken of instances with encrypted volumes are automatically encrypted. The root EBS volume snapshot is individually crash-consistent with this approach.

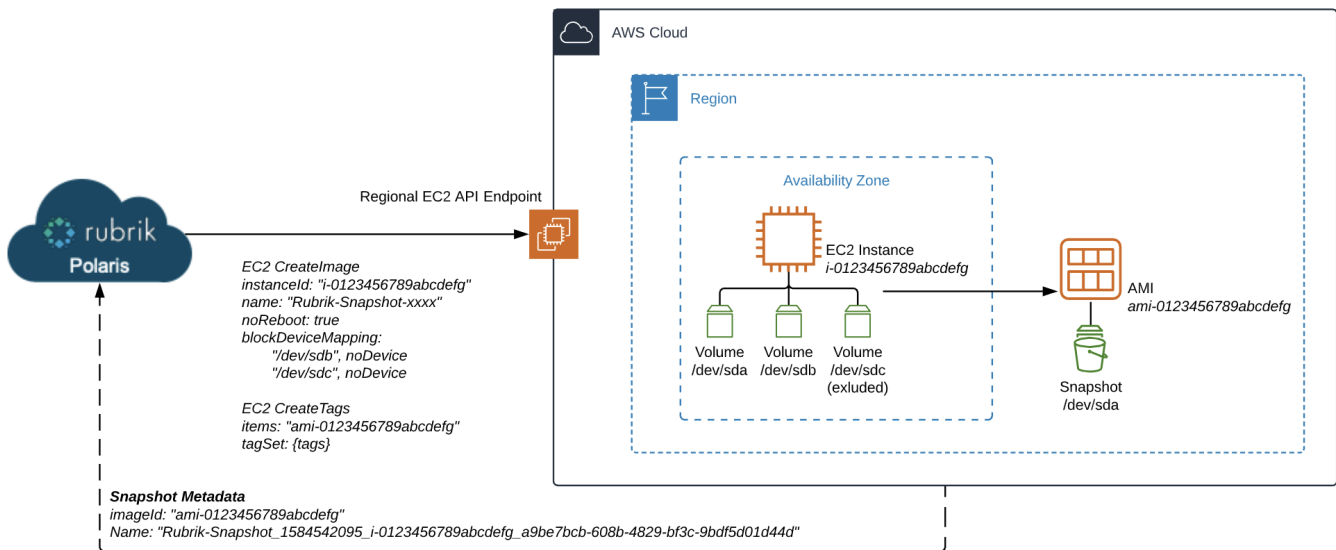


Figure 15 - Step 2: EC2 Instance Snapshot via CreateImage

Assuming AMI creation succeeds, the workflow tags the existing data volume snapshots (from the **create volume snapshots phase**) with the ID of the AMI that was just created.

At this point, the workflow will transition into the **delete excluded snapshots** phase. During this phase, Polaris identifies any excluded volumes whose snapshots are eligible for deletion. This step is required because `CreateSnapshots` does not allow for volume exclusion. Without this step volume exclusion would not be possible, and EBS snapshots of the excluded EBS volume would continue to be stored at a cost to the customer.

There is one caveat to this fact: one full snapshot of all excluded volumes will be kept at any given time. This is to reduce snapshot times. If one snapshot is not maintained for each disk then the time to produce each subsequent instance snapshot would increase to the duration required to create a full snapshot of the excluded volume(s) despite the fact that all other volumes required only time to create incremental snapshots of their EBS volumes.

Ultimately, this process produces an instance snapshot that consists of the AMI created from the source instance as well as the multivolume crash-consistent EBS snapshots of the non-excluded EBS volumes.

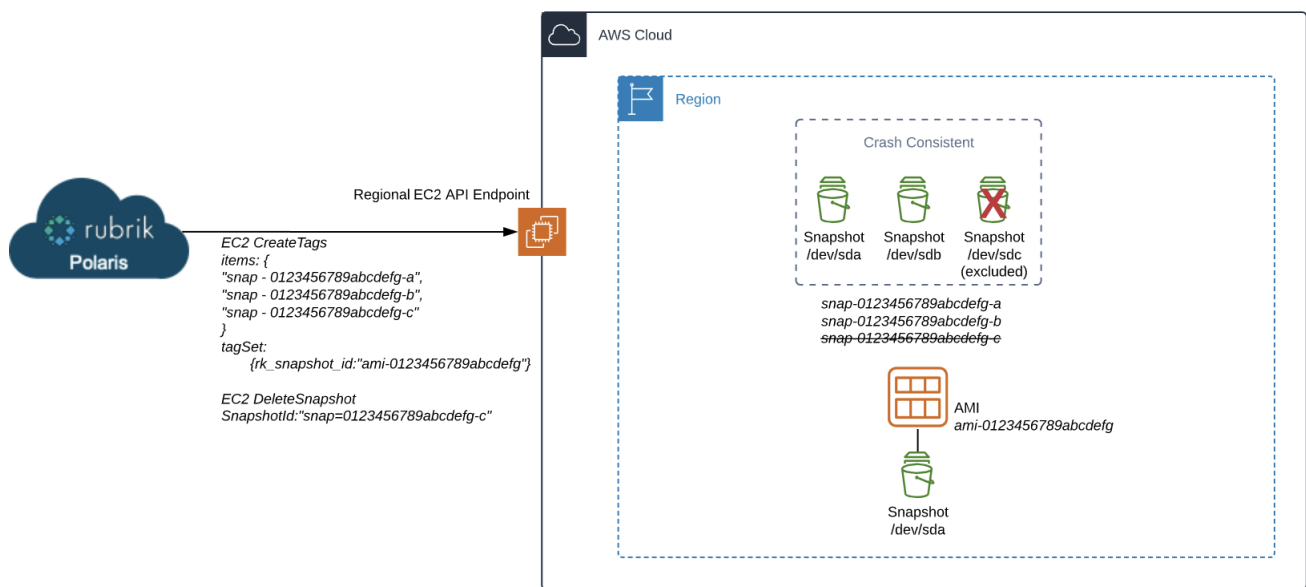


Figure 16 - Step 3: EC2 Instance Snapshot via `CreateSnapshot`

FILE INDEXING

Cloud-Native Protection for AWS leverages the Exocompute framework in Polaris to extract filesystem index metadata from the EBS snapshots created when protecting EC2 instances and EBS volumes. This metadata is then used when searching for, and recovering, individual files and folders from within these snapshots. Exocompute is an abstraction layer that allows other Polaris components to request and utilize compute assets in various clouds without concern for the nuances of that particular platform's implementation. In AWS, Exocompute leverages Amazon EKS which is a managed Kubernetes service.

The job responsible for indexing cloud-native snapshots in Polaris runs every two hours by default, this value can be altered by Rubrik support if required. The filesystem indexing workflow begins with a prepare task responsible for identifying snapshots in scope and eligible for indexing. A snapshot is considered eligible if:

- Indexing is enabled on its source EC2 instance or EBS volume
- Exocompute is configured and functional for the snapshot's account and region
- The snapshot is unexpired and does not have an expiry hint set
- The snapshot is currently unindexed
- The snapshot is not currently marked as unindexable

SLA created snapshots, On Demand snapshots, and relics are all eligible for indexing. Each run will index up to 5 snapshots per object and up to 20 objects in parallel, more recent snapshots are prioritized over older snapshots. In the event of a failure, Polaris will retry up to 5 times at which point the snapshot will be marked as unindexable.

Once prepared, Polaris will request an exocluster if necessary. This will ensure that an exocluster is running and available to run the index job in the required accounts and regions. This could mean launching a new exocluster, or it may mean utilizing an already running cluster. If a new exocluster is required, Polaris will use the EKS and EC2 APIs to initialize one as shown below.

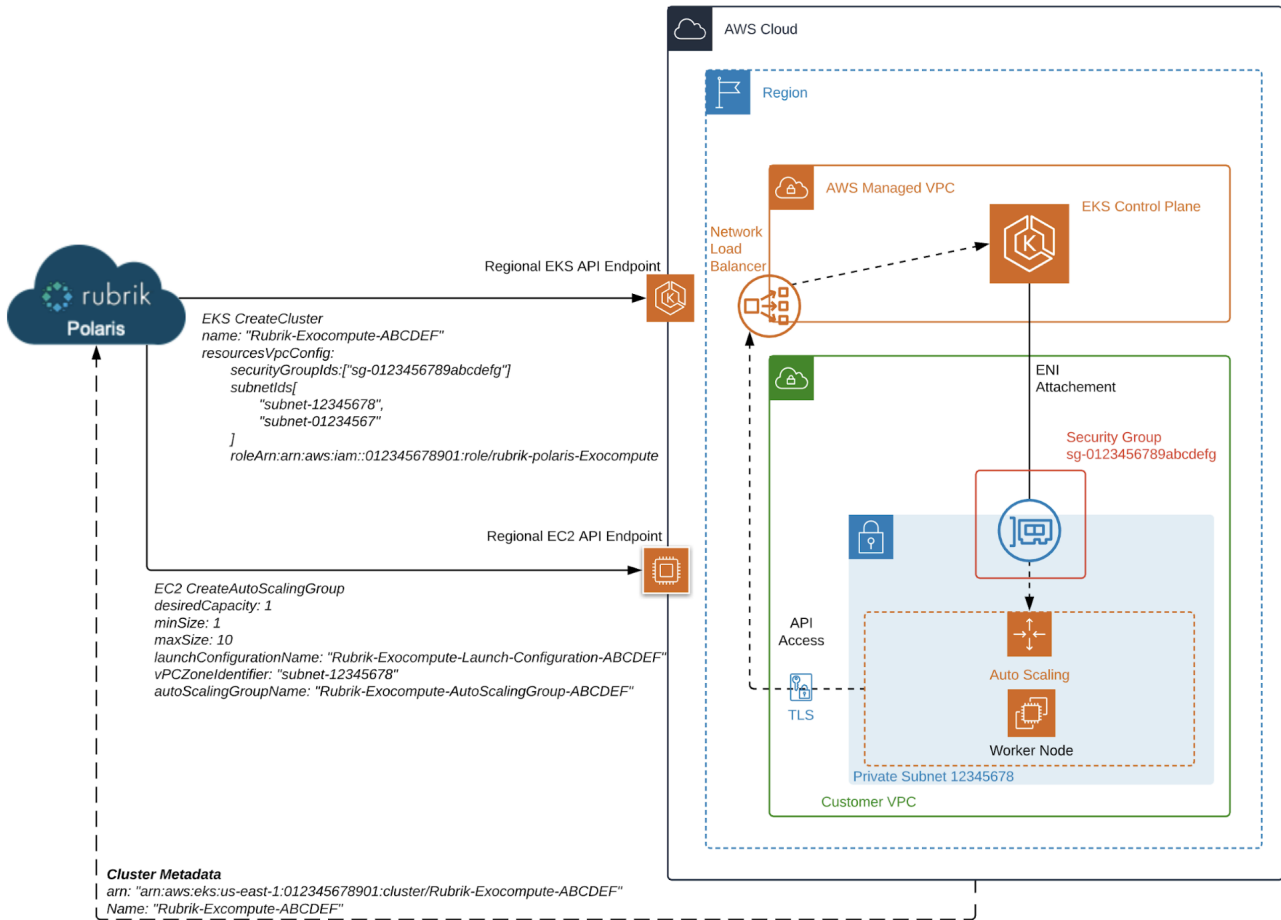


Figure 17 - Exocluster Launch

Once the cluster is active Polaris will create EBS volumes from the snapshots in scope for indexing and the index phase begins. During this phase, a pod is launched, the disks from the snapshot being indexed are attached to it, and indexing begins. Upon completion, indexes are securely transmitted back for storage on Polaris.

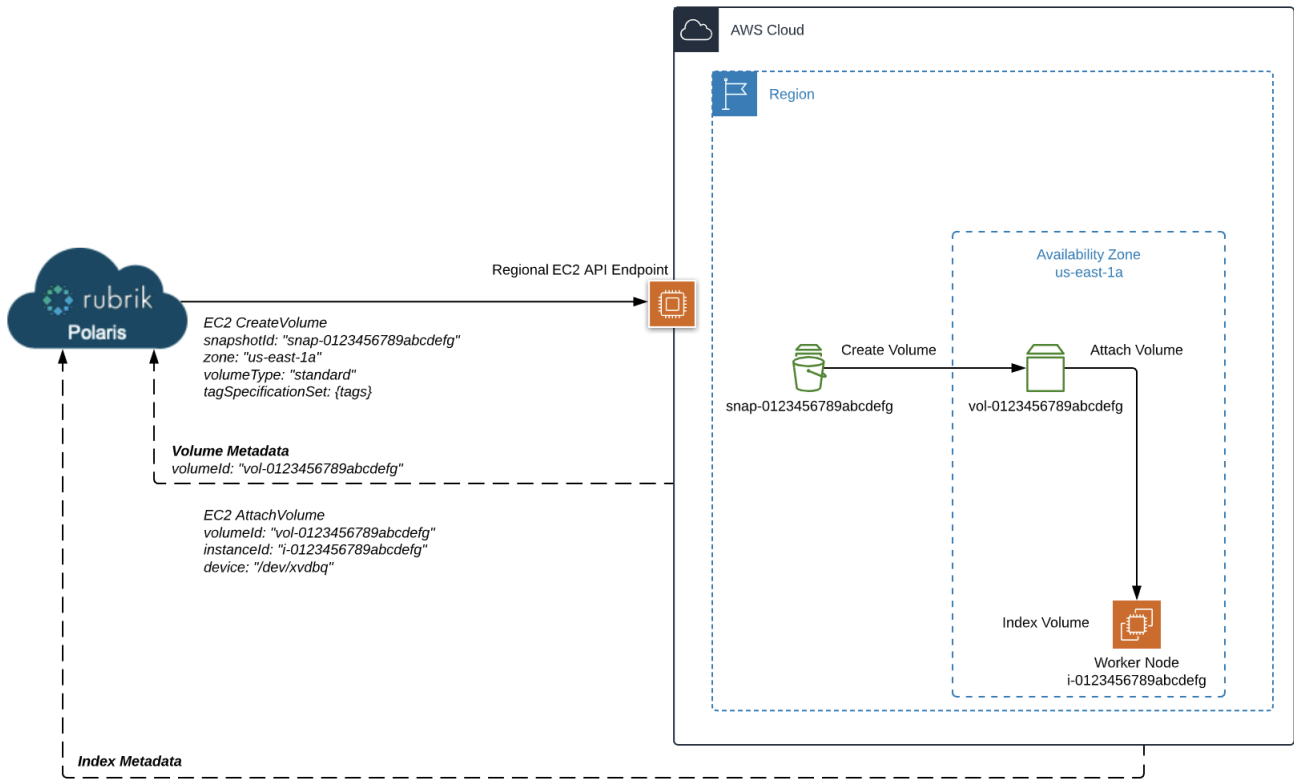


Figure 18 - Filesystem Indexing

Once complete, Polaris releases the handle on the Exocluster that was performing this work. The cluster may be torn down, or it may still be running other jobs.

Recovering a file is a nearly identical process to the index creation job just described. The exception being that instead of indexing the filesystem, Exocompute fetches the file(s) and places them in the S3 bucket selected for restore.

CROSS-ACCOUNT REPLICATION

When replicating snapshots across AWS accounts, Polaris relies on the native snapshot and image copy APIs available in EC2. Please note that in order to facilitate cross-account replication, the [appropriate KMS assets](#) must be in place prior to the first replication run. By default, replication runs every 30 minutes and will replicate all outstanding snapshots less than 30 days old at runtime. Replica snapshots will be stored in the target region in target account for the duration specified in the corresponding SLA Domain. These snapshots are encrypted using a rubrik specific gateway key in the target account.

The process for replicating CMK backed snapshots is straightforward so long as the CMKs are pre-shared as described prior. Polaris creates the local snapshots as described in the [Backup](#) section of this document. Then shares the snapshots and images with the target account. Following which, they are copied to the target account using the [CopySnapshot](#) and [CopyImage](#) APIs. These APIs allow for re-encryption as part of the copy operation, and the corresponding gateway key in the target account is used for this purpose. Once the copy operation is complete, the source image and snapshot shares are removed leaving an isolated copy in each account.

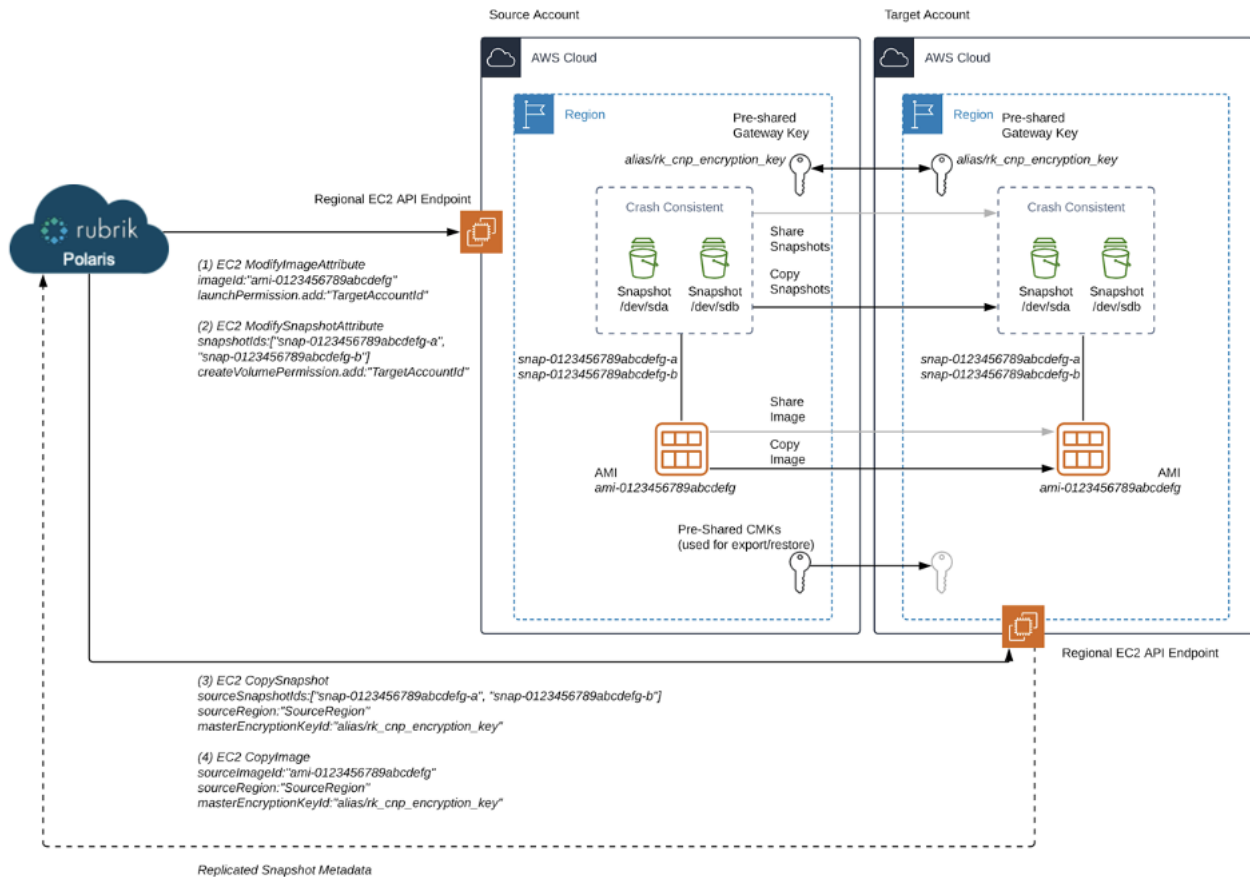


Figure 19 - Cross-Account Replication Steps

There is one deviation from this pattern. Objects encrypted with the default KMS key for EBS have to be re-encrypted in the source account prior to being copied over to the target account. This is because the default EBS key cannot be shared with another account. Thus, the creation of an intermediary snapshot is required so that it can be copied and re-encrypted with the gateway key in the target account. This is accomplished by using a gateway key in the source account that is provisioned into the same region as the source objects. Polaris will create intermediary snapshots here, share them with the target account, then continue the copy process. Once complete, the shares are removed, and the unnecessary intermediary snapshots are discarded. The exception being, that one intermediary snapshot is persisted to maintain replica incrementality.

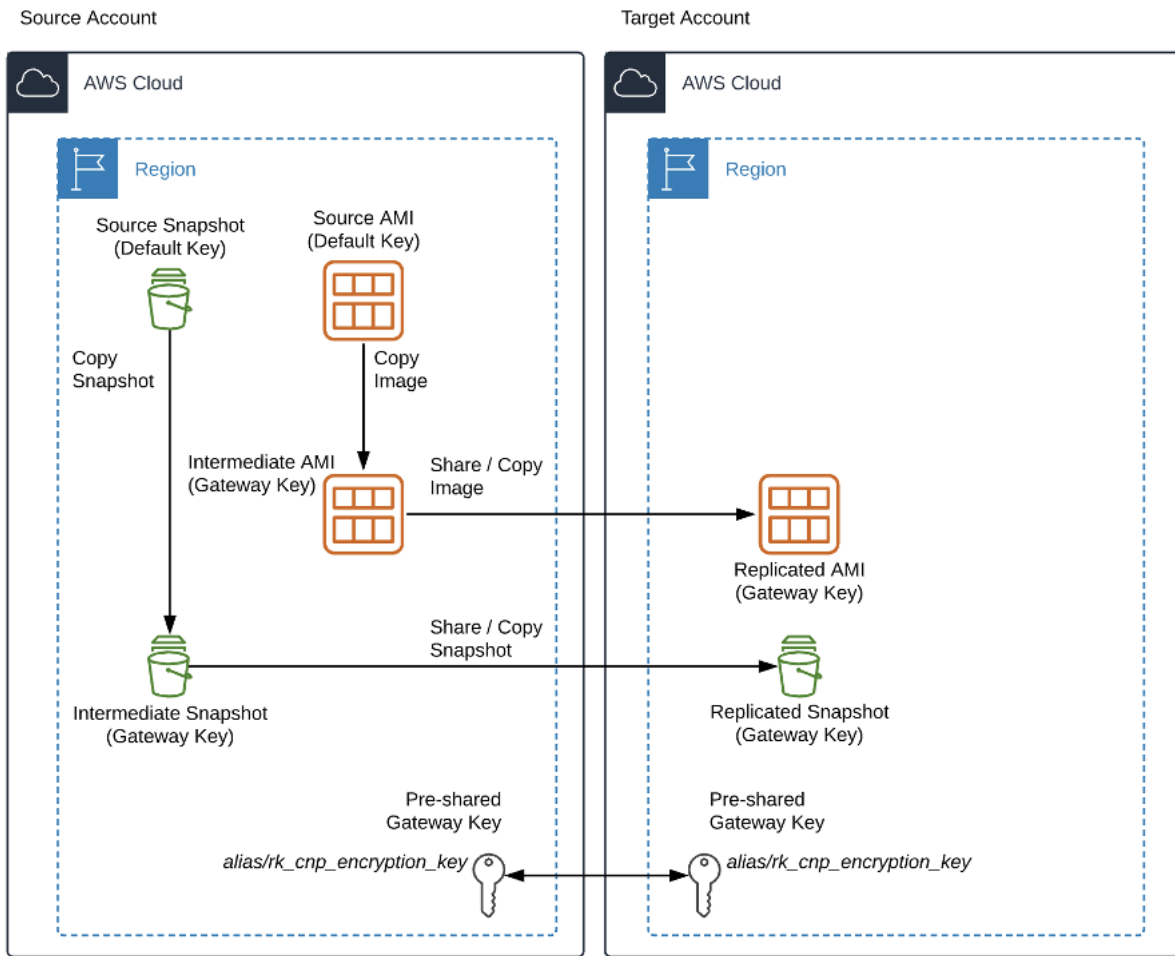


Figure 20 - Cross-Account with Default EBS Key

RESTORE

In order to restore an instance protected by Cloud-Native Protection for AWS the customer simply navigates to the instance they want to restore, selects the appropriate snapshot, and then chooses **Restore** from the context menu for that snapshot. This operation essentially rolls the currently running instance back to the point in time of the selected snapshot while allowing the restored instance to retain its instance id and private IP address. This is accomplished by stopping the running instance and replacing the disks on the instance with disks created from the snapshot being restored. This restore process is intelligent and will account for replication, disk exclusion, and tag recovery.

If the snapshot being restored was replicated to a remote region and the EBS Snapshots or AMI within the local region are unavailable, Polaris will copy the necessary volume snapshots or image from the remote region to the source region in order to utilize them for the restore.

Once Polaris has identified that the required snapshots and image are available it will launch the required EBS volumes from their corresponding EBS snapshots in preparation for the rollback. Volumes that were excluded from protection have no corresponding snapshots and thus, will not be recreated. Polaris polls the state of these newly created volumes until they show as *available*.

Once the volumes required for the restore are available Polaris shuts down the instance being restored and then detaches all volumes from the existing instance as soon as it reaches a *stopped* state. Once the detached volumes show as *available* they are [tagged with Rubrik metadata](#). The detached volumes are not deleted. Once the restore is complete and the customer has

validated the results, these detached volumes are typically deleted. However, if customer requirements dictate retaining them for some period of time, the recommended approach would be to snapshot the volume and retain that rather than the volume itself. This significantly reduces the cost of storing the data.

At this stage of the process Polaris attaches the volume(s) created prior to their original mount points on the instance being restored. Once the newly attached volumes are in the attached state, the restore process starts the instance and waits until it is in the *running* state. Polaris then tags the restored instance with the [appropriate tags](#). If the customer has chosen to restore tags, this includes the tags that were on the instance at the time of the snapshot being restored.

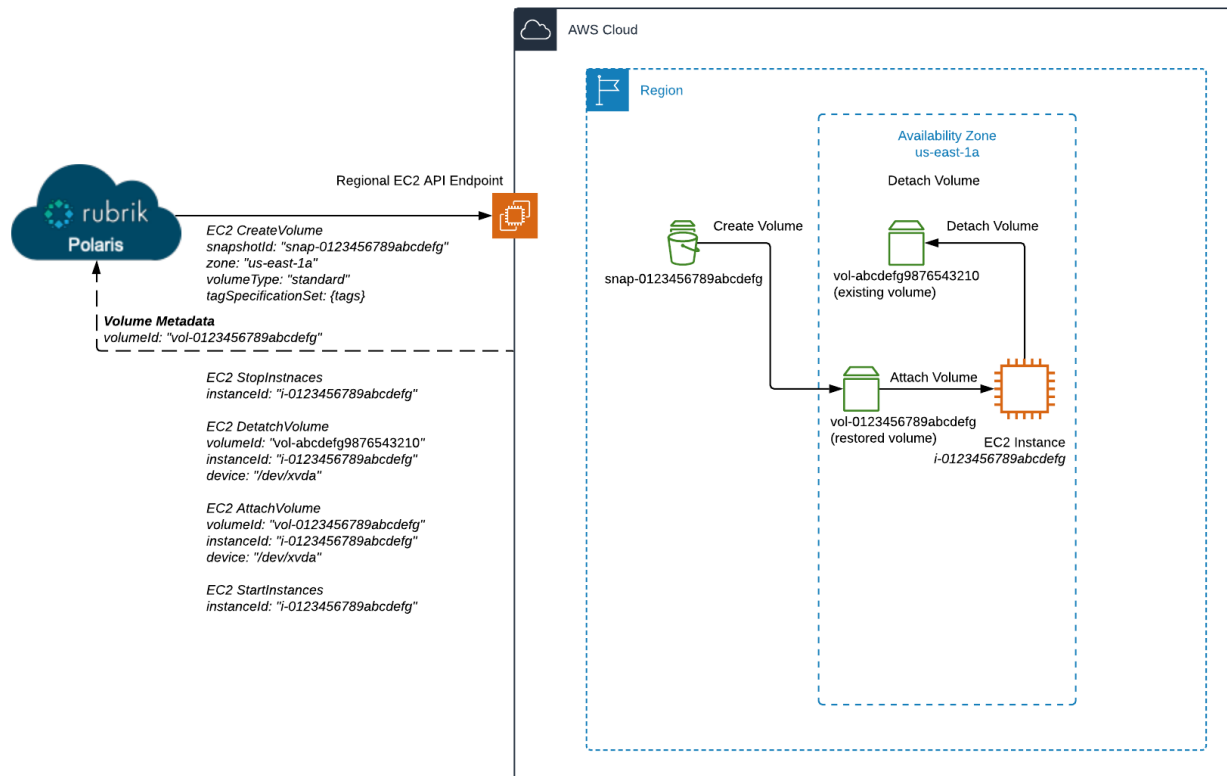


Figure 21 - EC2 Instance Restore

EXPORT

When running an export of an EC2 instance or EBS volume via Cloud-Native Protection for AWS on Polaris the customer creates a new EC2 instance or EBS volume from the snapshot being exported. This process is initiated by choosing **Export** in the context menu for the snapshot the customer wants to export.

In the dialogue box that appears the customer is able to specify the export parameters for the object being exported. Both EC2 instances and EBS volumes can be exported from either the local snapshot, or a Rubrik snapshot replica in a remote region (if one exists) and can be exported to any AZ within any region supported by Polaris. Cross region exports are a useful tool for test/dev and disaster recovery scenarios. Both object types allow the customer to define the name of the exported object, what KMS key should be used to encrypt EBS volumes, as well as whether or not the tags at the time of the snapshot should be applied to the export. The following sections explore each operation in more detail.

EBS VOLUME EXPORTS

EBS volume exports allow the customer to create a new volume from a snapshot of their choosing. In addition to the parameters above, customers can also specify the volume type (gp2, io1, sc1, st1, etc.), the volume size, and whether or not the volume should replace the existing source volume.

Polaris copies any required assets from the remote region if the objects required for the export are not currently in the target export region. This can happen in a few scenarios:

- An export of source region snapshot to a remote region
- An export of a replica region snapshot to any region other than the replica region
- An export of a missing source region snapshot to any region other than the replica region
- An export of a missing replica region snapshot to any region other than the source region

During this phase, the required snapshot is copied to the export target region and then polled until its status is *completed*. Once the required volume snapshot is available in the specified region Polaris will launch a new EBS volume from the corresponding EBS snapshot and tag it with the [appropriate tags](#). If export tags was selected, the tags from the original volume will be included as well. Polaris then polls the state of this newly created volume until it shows as *available*.

At this stage of the process, if the customer has selected the option to **Replace the existing volume where attached**, Polaris stops the instance where the source volume is attached and polls it until it is in a *stopped* state. After which, the original source volume for the snapshot being exported will be detached (but not deleted) from its instance and the newly exported volume will be attached in its place. Lastly Polaris restarts the instance and polls for it to be in a *running* state and [tags the detached volume](#).

Upon completion the exported volume is now available in the desired region. If **Replace the existing volume where attached** option was selected the newly exported volume will be attached in place of the existing source volume and the existing source volume will be detached and ready for use or deletion. The diagram below depicts this process when exporting a volume and replacing the existing volume attachment.

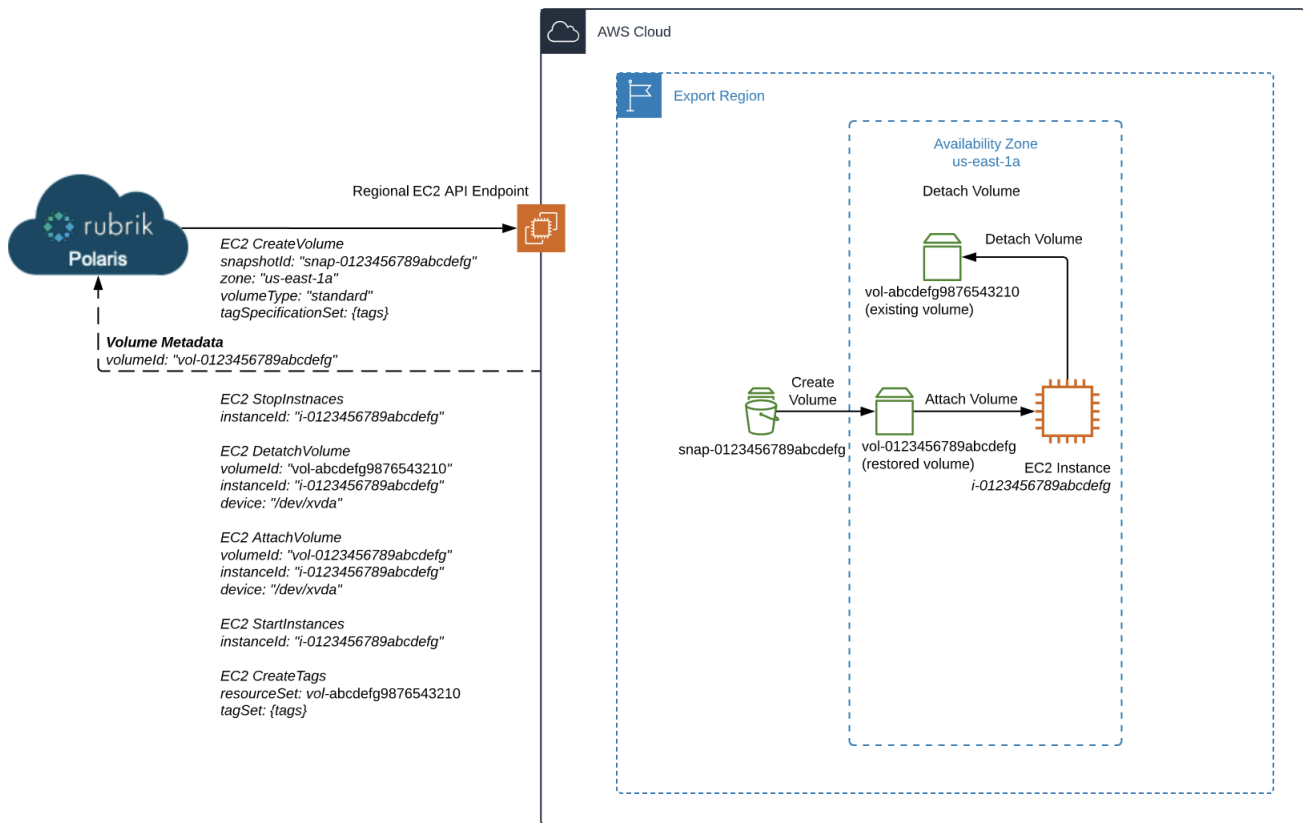


Figure 22 - EBS Volume Export with Replace Attachment

EC2 INSTANCE EXPORTS

EC2 instance exports allow the customer to create a new EC2 instance from a snapshot of their choosing. In addition to the previously mentioned parameters, customers can also specify the instance size and type, as well as the vpc, subnet, and security groups to apply to the new instance.

As with EBS Snapshot exports, EC2 instance exports will only Copy Volume Snapshots / Copy Image if the snapshots or image being used for the export are not currently in the target region for the export. This can happen in a few scenarios:

- An export of source region snapshot to a remote region
- An export of a replica region snapshot to any region region other than the replica region
- An export of a missing source region snapshot or image to any region other than the replica region
- An export of a missing replica region snapshot or image to any region other than the source region

If necessary, the required image and snapshots are copied to the export target region and then polled until the AMI is *available* and the EBS snapshots show a status of *completed*. Once the required assets are available in the export region Polaris will launch an instance from the corresponding image. During this phase, an EC2 instance is launched in the target region utilizing the AMI corresponding to the snapshot being exported. If the customer selected export tags when exporting the instance, the tags from the source instance at time of snapshot are copied to the exported instance. The [Rubrik specific tags](#) are always applied regardless of selection.

The new instance is polled until it shows as *running*. At this point there are two possible next steps. If the snapshot being exported was created with *volume level crash consistency* the process will move on to deleting any copied assets if required and then move onto completion.

The more common scenario is that of a snapshot that was created with *instance level crash consistency*, the default consistency type for EC2 instances protected via Cloud-Native Protection for AWS. In this scenario, the EBS volumes for the exported instance are created by a launch volumes task. These volumes must be launched independent of the AMI since they were created via the [CreateSnapshots](#) API to achieve *instance level crash consistency*. These volumes are tagged with the same values as the instance itself.

Polaris then stops the exported instance. Once the instance is in a stopped state, Polaris will detach the root volume (in this scenario, the only volume) from the exported instance. Once this volume is in an *available* state, it is Deleted, and Polaris attaches the volumes with *instance level crash consistency* to the appropriate mount points. Once the attached volumes are in an *in-use* state, Polaris restarts the exported instance and cleans up any copied objects as required.

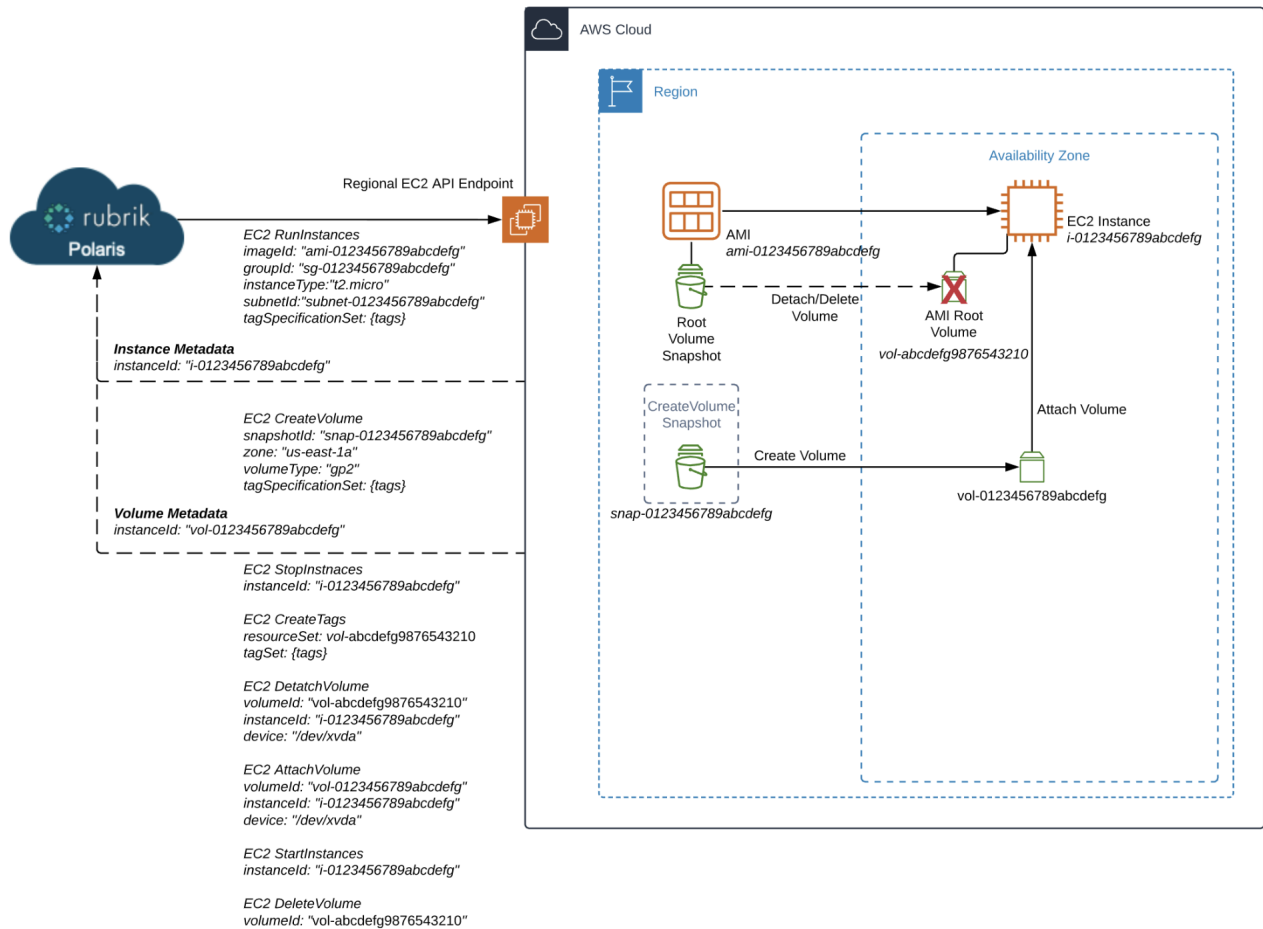


Figure 23 - EC2 Instance Export

SUMMARY

This concludes the How it Works guide on Cloud-Native Protection for AWS on Rubrik Polaris. The guide explained the architecture and the value proposition of Cloud-Native Protection for AWS as well as educated the reader on the nuances of each major workflow within the product. Additionally, the guide equipped the reader with some common techniques and best practices for using the product efficiently from both an operations and cost perspective allowing them to fully understand and unlock the potential of Cloud-Native Protection for AWS.

GLOSSARY

AMAZON ELASTIC BLOCK STORE (EBS) SNAPSHOT

EBS snapshots are a point in time copy of the EBS volume they are associated with and are stored in S3, although the AWS customer can only interact with them through the EC2 console and APIs. EBS snapshots are incremental in nature, which means that only the blocks on the volume that have changed since the last snapshot will be stored when taking subsequent snapshots. AWS handles all the logic required to consolidate blocks when creating and deleting snapshots of the same volume. EBS Snapshots are redundant across AZs within a region but have to be copied across regions if regional redundancy is required.

AMAZON ELASTIC BLOCK STORE (EBS) VOLUME

EBS volumes are block storage devices that AWS customers can connect to EC2 instances, similar to VMware VMDKs. Each EBS volume is redundant within an AZ but not across AZs or regions. Most EC2 instances have at least 1 EBS volume (known as the root volume) attached to them. They may also have additional data volumes attached to them as well.

When Polaris takes back ups of EBS volumes, it is calling the AWS APIs to create EBS snapshots. This means that these snapshots are visible and accessible in the EC2 console.

AMAZON ELASTIC COMPUTE CLOUD (EC2) INSTANCE

An EC2 instance is essentially a virtual server running in AWS. The instance size and type dictate the CPU, RAM, and other performance characteristics of the instance. Instances run in a subnet within virtual networks known as VPCs. Each subnet is associated with one AZ, thus each instance runs within a single AZ.

AMAZON ELASTIC KUBERNETES SERVICE (EKS)

Amazon Elastic Kubernetes Service is a managed service that makes it easy for AWS customers to run Kubernetes on AWS without needing to stand up or maintain their own Kubernetes control plane. Rubrik's Exocompute engine uses EKS whenever local compute is required in the customer's AWS environment to perform a task. In terms of Cloud-Native Protection for AWS, Exocompute is utilized to perform file level indexing and recovery without the need for agents inside of the customer's EC2 instances.

AMAZON MACHINE IMAGE (AMI)

An AMI is an image file in AWS that contains all the necessary information required to launch an EC2 instance. When an AWS customer launches an instance through the AWS console or through the EC2 API they must specify the AMI that will be used for launch. An AMI includes a list of the EBS snapshots, permissions that control which AWS accounts can use the AMI, and a mapping file that specifies where the volumes corresponding to each EBS snapshot should be mounted.

When Polaris backs up EC2 instances, it is calling both the AWS APIs to create AMIs as well as the AWS APIs to create EBS snapshots. This means that both the image files for the corresponding instances and the snapshots of the underlying EBS volumes themselves are visible and accessible in the customer's EC2 console.

AMAZON RESOURCE NAME (ARN)

Amazon Resource Names (ARNs) uniquely identify AWS resources. We require an ARN when you need to specify a resource unambiguously across all of AWS, such as in IAM policies, Amazon Relational Database Service (Amazon RDS) tags, and API calls.

AMAZON SIMPLE NOTIFICATION SERVICE (SNS)

Amazon Simple Notification Service (Amazon SNS) is a web service that makes it easy to set up, operate, and send notifications from the cloud. It provides developers with a highly scalable, flexible, and cost-effective capability to publish messages from an application and immediately deliver them to subscribers or other applications.

AMAZON WEB SERVICES (AWS)

Amazon Web Services (AWS) is a subsidiary of Amazon that provides on-demand cloud computing platforms and APIs to individuals, companies, and governments, on a metered pay-as-you-go basis.

AWS AVAILABILITY ZONES (AZ)

An AWS Availability Zone is a subset of a region, except in the case where a Region only has one AZ, which is rare. AZs are discrete data centers within a region and have redundant power, networking, and connectivity within the Region Network performance across AZs is typically high throughput and low latency. AZs are typically many miles away from other AZs but are generally within 60 miles of one and other within the region. Certain entities such as EC2 instances and EBS volumes exist within a single AZ, although the latter is redundant within the AZ but not across AZs or regions.

AWS CLOUDFORMATION

AWS CloudFormation allows AWS customers to model infrastructure and application resources with templates written in formats like JSON or YAML. These templates can be submitted to the CloudFormation service on AWS to create a collection of resources known as a stack. These stacks and their life cycles are managed via CloudFormation itself rather than individually in AWS. Update the template in CloudFormation and the stack will be updated accordingly, delete the stack and all resources provisioned as part of the stack will be deleted as well.

AWS IDENTITY AND ACCESS MANAGEMENT (IAM) POLICY

AWS customers manage access in AWS by creating policies and attaching them to IAM identities (users, groups of users, or roles) or AWS resources. A policy is an object in AWS that, when associated with an identity or resource, defines their permissions. AWS evaluates these policies when an IAM principal (user or role) makes a request. Permissions in the policies determine whether the request is allowed or denied.

AWS IDENTITY AND ACCESS MANAGEMENT (IAM) ROLE

Roles are an identity that AWS customers can create in AWS and assign permissions to. Roles can be utilized only by trusted AWS services or accounts. IAM Policies attached to these roles dictate the actions that an entity using the role can take.

AWS KEY MANAGEMENT SERVICE (KMS)

AWS KMS is a managed service that enables AWS customers to easily create and control the keys used for cryptographic operations. The service provides a highly available key generation, storage, management, and auditing solution for AWS customers to encrypt or digitally sign data within their own applications or control the encryption of data across AWS services. KMS Keys can be created and managed by AWS and are typically referred to as Default Keys, or by customers themselves and known as Customer Managed Keys (CMKs).

AWS REGIONS

AWS is made up of [regions](#) spread across the globe, each region is completely independent of the other regions in terms of location, power, cooling, water supply, etc. Each region consists of one or more (typically two or more) availability zones. Many AWS services (such as EC2) are managed at the regional level, leveraging separate consoles and API endpoints when interacting with different regions within an AWS account.

CUSTOMER AWS ACCOUNT

The customer owned AWS account houses the resources protected by Cloud-Native Protection for AWS. Polaris assumes an IAM Role from the Rubrik owned AWS account into the customer owned account in order to make the API calls necessary to protect EC2 instances and EBS volumes. The images and snapshots created by this process continue to reside in the customer owned AWS account, only metadata is stored in Polaris.

EXOCOMPUTE

Exocompute extends Polaris to support a “bring-your-own-compute” approach to protecting EC2 instances and EBS volumes. Exocompute is a framework for running containers in the customers VPC with a bidirectional communication path between the containers and Polaris. This framework abstracts away the underlying cloud’s container service and allows Cloud-Native Protection to support and index workloads across cloud platforms.

RELIC

In terms of Cloud-Native Protection for AWS, a relic is an EC2 instance or EBS volume that has been deleted while there are still existing snapshots for the object. These snapshots are retained according to the SLA assigned to the object at the time of deletion. On demand snapshots can be deleted as usual.

RUBRIK AWS ACCOUNT

The Rubrik owned AWS account is used as an identity source and as a bastion for interacting with the customer owned AWS accounts that are under protection via Polaris. Making AWS API calls from this account allows Rubrik to leverage IAM Roles to interact with customer owned AWS accounts.

RUBRIK POLARIS

Polaris is Rubrik's SaaS platform which aggregates metadata related to data protection activities across hybrid and multi-cloud environments. The end result is a robust framework consisting of metadata from several data sources, often referred to as a unified system of record. Rubrik Polaris enables customers to exploit the business value hidden in this dataset leveraging AI and ML driven tools within Polaris.

Full details on the capabilities of Polaris are outside the scope of this document but they include: centralized management for Rubrik CDM clusters, ransomware detection and rollback via Polaris Radar, data classification and compliance via Polaris Sonar, as well as Cloud-Native Protection for public cloud workloads, the topic of this document.

A Polaris tenant is a microservices application built on top of a Kubernetes engine and is comprised of, but limited to the following services:

- HTML 5 User Interface
- API Services
- User identity and access services
- Distributed task/job schedulers
- Reporting services
- Gateway services for interacting with customer endpoints (CDM Clusters, Public Clouds, etc.)
- Compute engine management services
- Database services
- Logging and Metrics services

Ultimately, none of this detail typically matters to Polaris customers. The application is provisioned and managed by Rubrik with delivery to the end-user via a web based SaaS portal.

SERVICE LEVEL AGREEMENT (SLA) DOMAIN

Rubrik SLA Domains are data protection policies built within Polaris and then assigned to assets requiring protection. SLA Domains are comprised of these three components when utilized by Cloud-Native Protection for AWS:

- Snapshot Frequency
- Snapshot Retention
- Replica Region and Duration

APPENDIX A - AWS TAGS

VOLUME SNAPSHOT TAGS:

TAG KEY	TAG VALUE
Name	Rubrik-Snapshot_snapshot-creation-time-mils_ebs-volume-native-id_UUID
rk_aws_native_account_id	Polaris UUID for source AWS Account
rk_component	Cloud Native Protection
rk_object	EBS Backup
rk_snapshot_type	Primary / Replica / Replica Intermediate
rk_sla_name	Name of assigned SLA Policy
rk_source_volume_native_id	AWS ID for source EBS volume
rk_source_volume_native_name	AWS Name of source EBS volume
rk_taskchain_id	Polaris UUID for source task chain

INSTANCE SNAPSHOT TAGS:

TAG KEY	TAG VALUE	TAG LOCATION
AMI Name	Rubrik-Snapshot_snapshot-creation-time-mils_ec2-instance-id_UUID	AMI
rk_aws_native_account_id	Polaris UUID for source AWS Account	AMI / EBS Snapshots
rk_component	Cloud Native Protection	AMI / EBS Snapshots
rk_gc_leaked_resource	False	AMI / EBS Snapshots
rk_object	EC2 Backup	AMI / EBS Snapshots
rk_sla_name	Source Snapshot SLA Name	AMI / EBS Snapshots
rk_snapshot_type	Primary	AMI / EBS Snapshots
rk_source_vm_native_id	AWS ID for source EC2 instance	AMI / EBS Snapshots
rk_source_vm_native_name	AWS Name of source EC2 instance	AMI / EBS Snapshots
rk_source_volume_native_id	AWS ID for source EBS volume	EBS Snapshots
rk_source_volume_native_name	Name for source EBS volume	EBS Snapshots
rk_taskchain_id	Polaris UUID for source task chain	AMI / EBS Snapshots

REPLICA SNAPSHOT TAGS:

TAG KEY	TAG VALUE	TAG LOCATION
AMI Name	Rubrik-Snapshot_snapshot-creation-time-mils_ec2-instance-id_UUID	AMI
rk_aws_native_account_id	Polaris UUID for source AWS Account	AMI / EBS Snapshots
rk_component	Cloud Native Protection	AMI / EBS Snapshots
rk_object	EC2 / EBS Backup	EBS Snapshot
rk_snapshot_type	Primary / Replica / Replica Intermediate	
rk_sla_name	Name of assigned SLA Policy	
rk_gc_creation_time	Snapshot Creation Time	EBS Snapshots
rk_gc_leaked_resource	false	AMI / EBS Snapshots
rk_object	EC2/ EBS Backup	AMI / EBS Snapshots
rk_sla_name	Source Snapshot SLA Name	AMI / EBS Snapshots
rk_snapshot_type	Replica	AMI / EBS Snapshots
rk_source_vm_native_id	AWS ID for source EC2 instance	AMI / EBS Snapshots
rk_source_vm_native_name	AWS Name of source EC2 instance	AMI / EBS Snapshots
rk_source_volume_native_id	AWS ID for source EBS volume	EBS Snapshots
rk_source_volume_native_name	Name for source EBS volume	EBS Snapshots
rk_taskchain_id	Polaris UUID for source task chain	AMI / EBS Snapshots
rk_snapshot_id	AWS ID of AMI corresponding to this volume snapshot	EBS Snapshots

RESTORE DETACHED VOLUME TAGS:

TAG KEY	TAG VALUE
rk_detached_by	Name of job that detached this volume
rk_detached_from	Instance id and device path that volume was detached from
rk_detached_time	The UTC time when the volume was detached
rk_component	Cloud Native Protection
rk_object	Detached EBS Volume
rk_taskchain_id	Polaris UUID for source task chain
rk_user	The Polaris user that triggered the restore operation
rk_source_snapshot_native_id	The Resource ID of the AMI used to trigger the restore
rk_restore_source_region	The region of the snapshot used for restore

RESTORED INSTANCE AND VOLUME TAGS:

TAG KEY	TAG VALUE
rk_restore_timestamp	The UTC time when the restore was initiated
rk_user	The Polaris user that triggered the restore operation
rk_source_snapshot_native_id	The Resource ID of the AMI used to trigger the restore
rk_restore_source_region	Source region of the snapshot used for restore
rk_taskchain_id	Polaris UUID for source task chain
rk_aws_native_account_id	The ID of the AWS Account on Rubrik
rk_component	Cloud Native Protection
rk_object	Restored instance / volume
rk_source_vm_native_id	AWS ID of the restored instance
rk_source_vm_native_name	Name of the restored instance
rk_restore_source_region	The region of the snapshot used for restore
rk_taskchain_id	Polaris UUID for source task chain
rk_aws_native_account_id	The ID of the AWS Account on Rubrik
rk_component	Cloud Native Protection
rk_object	EBS Volume of restored instance

EXPORTED INSTANCE AND INSTANCE VOLUME TAGS:

TAG KEY	TAG VALUE
Name	The name assigned to the exported instance
rk_aws_native_account_id	The ID of the AWS Account on Rubrik
rk_export_source_region	The region of the source snapshot
rk_export_timestamp	The UTC time when the export job was initiated
rk_source_snapshot_native_id	The AWS ID of the source snapshot for this volume
rk_source_vm_native_id	The AWS ID of the source instance for this export
rk_source_vm_native_name	The AWS name of the source instance for this export
rk_user	The Polaris user ID that executed this export
rk_taskchain_id	Polaris UUID for source task chain
rk_gc_leaked_resource	False
rk_gc_creation_time	Snapshot Creation Time
rk_component	Cloud Native Protection
rk_object	Exported Instance / EBS Volume of exported instance

EXPORT DETACHED VOLUME TAGS:

TAG KEY	TAG VALUE
rk_detached_by	export-volume
rk_detached_from	Instance id and mount point that volume was detached from
rk_detached_time	The UTC time when the volume was detached
rk_component	Cloud Native Protection
rk_object	Detached EBS Volume
rk_taskchain_id	Polaris UUID for source task chain
rk_user	The Polaris user that triggered the restore operation
rk_source_snapshot_native_id	The Resource ID of the AMI used to trigger the restore
rk_restore_source_region	The region of the snapshot used for export

EXPORTED VOLUME TAGS:

TAG KEY	TAG VALUE
Name	The name assigned to the exported volume
rk_aws_native_account_id	The ID of the AWS Account on Rubrik
rk_export_source_region	The region of the source snapshot
rk_export_timestamp	The UTC time when the export job was initiated
rk_source_snapshot_native_id	The AWS ID of the source snapshot for this volume
rk_source_volume_native_id	The AWS ID of the source volume for this export
rk_user	The email id of the Polaris user that executed this export
rk_component	Cloud Native Protection
rk_object	Exported Volume
rk_taskchain_id	Polaris UUID for source task chain

APPENDIX B - METADATA

METADATA	USE CASE
Account ID Account Name IAM Role (created by Rubrik) Stack ARN (created by Rubrik)	For setup/upgrade/disable operations for each account added on Polaris.
Following properties of EC2 instances in the accounts added on Polaris: Instance ID Instance Name Instance Type Region and Availability Zone VPC ID, name Private/Public IP OS Type Tags Is VM Marketplace or not. SSH key pair name Indexing status of the instance	Display instances list on Inventory and protect/recover them. Only applies to protected regions.
Following properties of EBS volumes in the accounts added on Polaris: Volume ID Volume name Volume Type Region and Availability Zone Size, IOPS Is encrypted or not. Is Marketplace or not. Tags Indexing status of the volume ID of snapshot from which volume was created (if applicable) Device path (if the volume is attached to an EC2 instance) Is root volume (if the volume is attached to an EC2 instance)	Display EBS volumes on Inventory and protect/recover them. Only applies to protected regions.
Following properties of VPCs in the accounts added on Polaris: VPC ID VPC Name Region	To allow filtering by VPCs in the EC2 instance/EBS volume list pages on Inventory. Only applies to protected regions.
Following properties of snapshots (AMI, EBS Volume Snapshots) taken by the Rubrik: Snapshot ID, name, region, account, block device paths Metadata of corresponding EC2 instance/EBS Volume	Backup and Recovery.
For File Recovery feature: VPC, 2 Subnets in which Exocompute is launched. Cluster Control Plane Security Group and Worker Node Security Group (Created by Rubrik) Cluster Role ARN, Worker Node Role ARN, Node Instance Profile (Created by Rubrik) Index files (which includes filesystem metadata: file paths, stat information etc.).	To be able to launch EKS clusters in provided VPC and run indexing/file download.

VERSION HISTORY

Version	Date	Summary of Changes
1.0	April 2020	Initial Release
1.1	March 2021	Adds cross region replication, updates tags and metadata



Global HQ

1001 Page Mill Rd., Building 2
Palo Alto, CA 94304
United States

1-844-4RUBRIK
inquiries@rubrik.com
www.rubrik.com

Rubrik, the Multi-Cloud Data Control™ Company, enables enterprises to maximize value from data that is increasingly fragmented across data centers and clouds. Rubrik delivers a single, policy-driven platform for data recovery, governance, compliance, and cloud mobility. For more information, visit www.rubrik.com and follow [@rubrikinc](https://twitter.com/rubrikinc) on Twitter. © 2021 Rubrik. Rubrik is a registered trademark of Rubrik, Inc. Other marks may be trademarks of their respective owners.