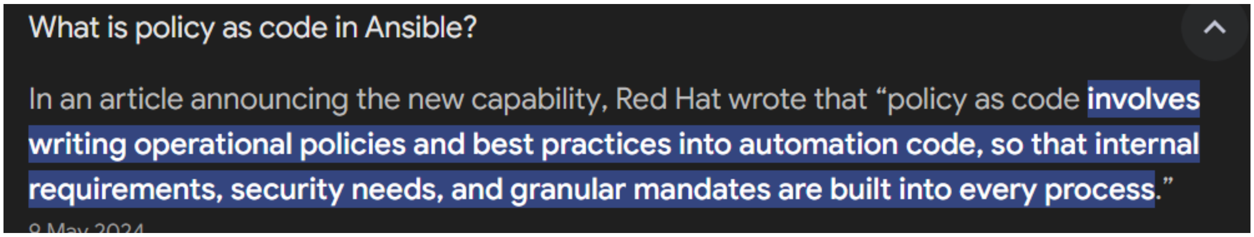
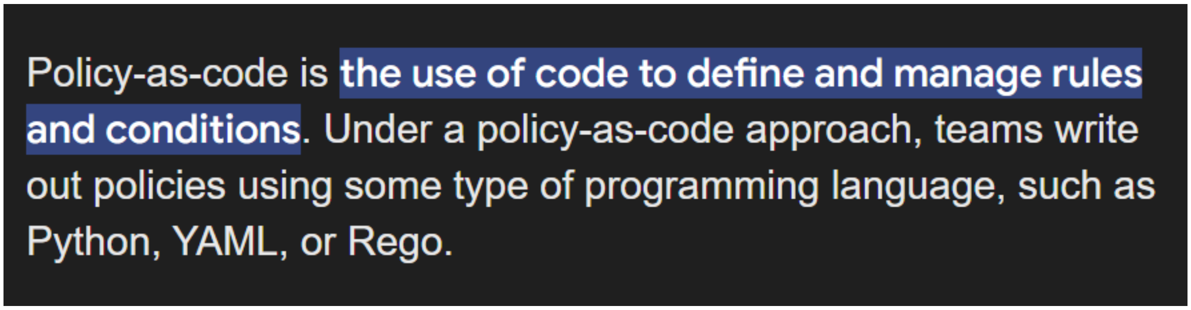


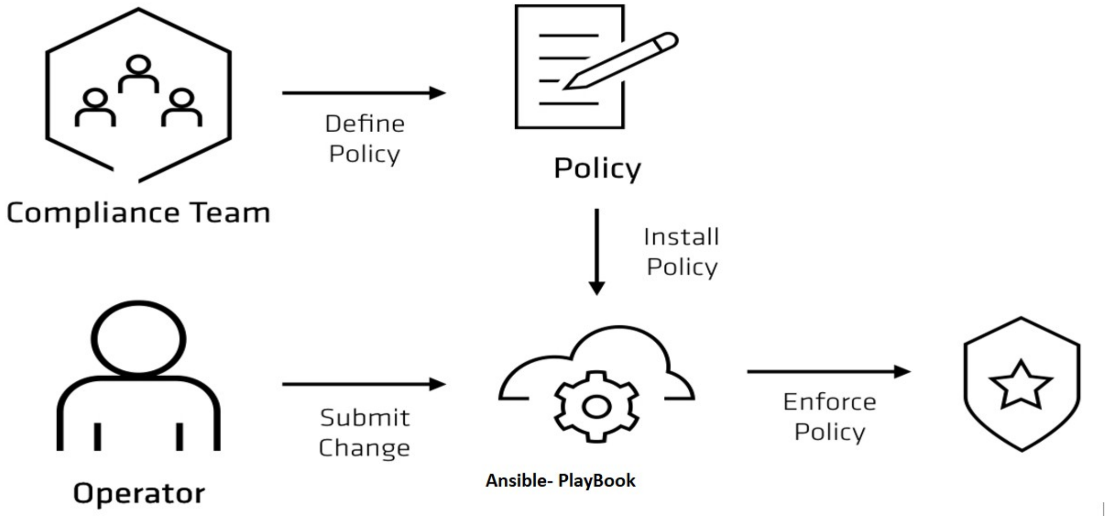
What is policy as code in Ansible?

In an article announcing the new capability, Red Hat wrote that "policy as code involves writing operational policies and best practices into automation code, so that internal requirements, security needs, and granular mandates are built into every process."

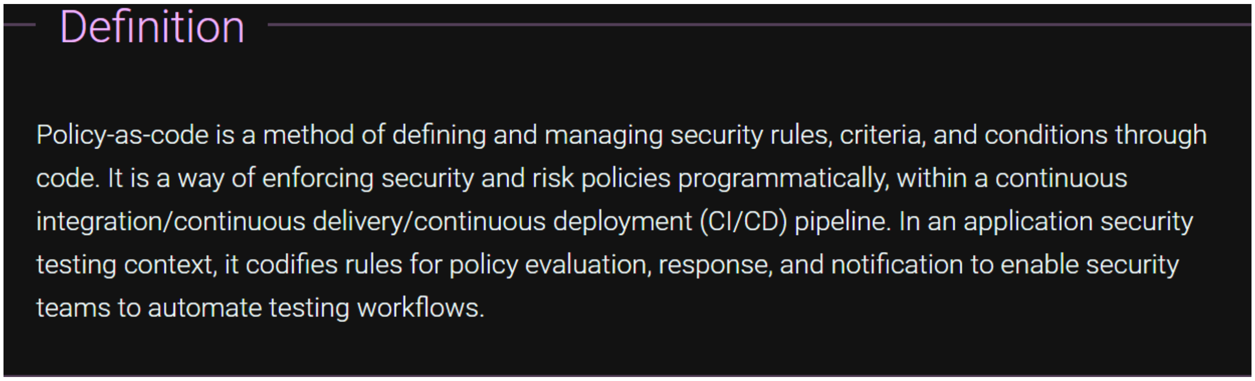
8 May 2024



Policy-as-code is the use of code to define and manage rules and conditions. Under a policy-as-code approach, teams write out policies using some type of programming language, such as Python, YAML, or Rego.



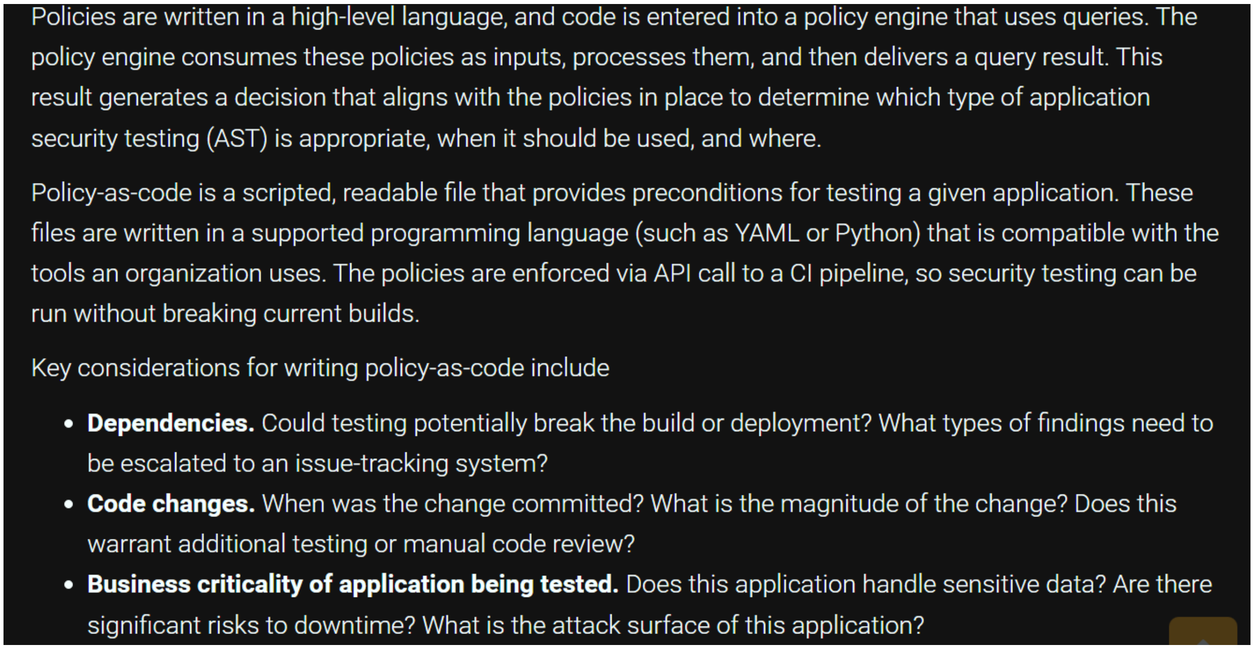
Policy-as-Code



Definition

Policy-as-code is a method of defining and managing security rules, criteria, and conditions through code. It is a way of enforcing security and risk policies programmatically, within a continuous integration/continuous delivery/continuous deployment (CI/CD) pipeline. In an application security testing context, it codifies rules for policy evaluation, response, and notification to enable security teams to automate testing workflows.

# How Policy Works



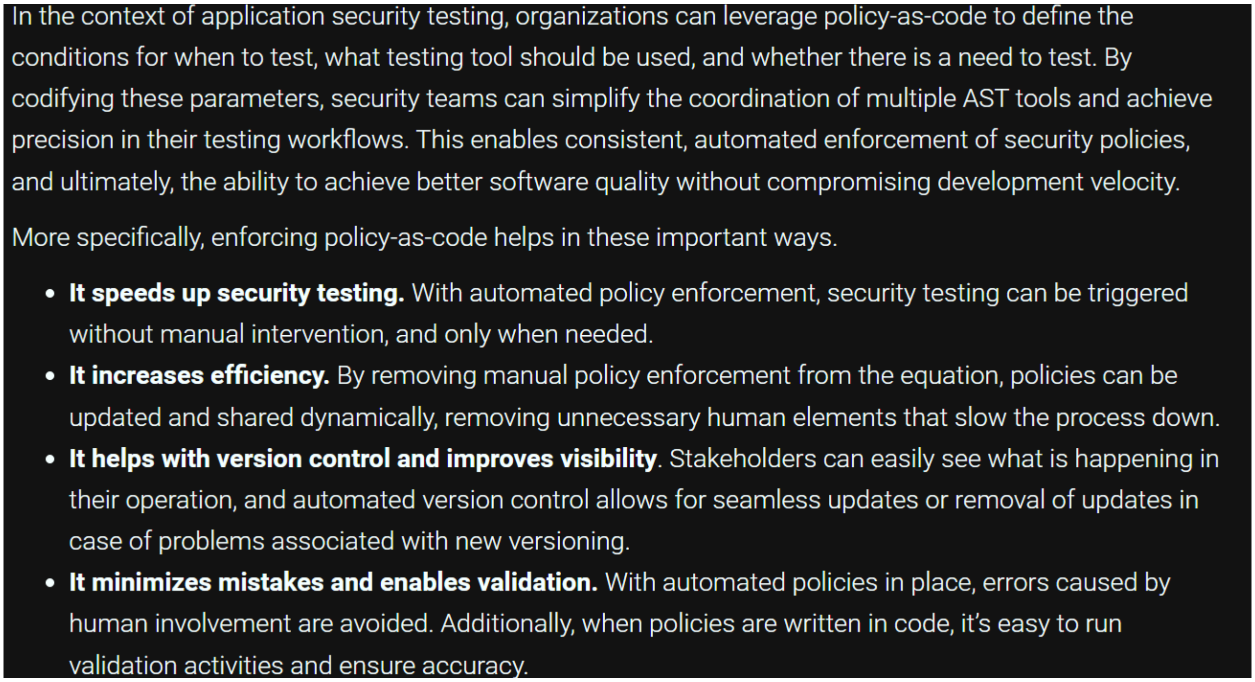
Policies are written in a high-level language, and code is entered into a policy engine that uses queries. The policy engine consumes these policies as inputs, processes them, and then delivers a query result. This result generates a decision that aligns with the policies in place to determine which type of application security testing (AST) is appropriate, when it should be used, and where.

Policy-as-code is a scripted, readable file that provides preconditions for testing a given application. These files are written in a supported programming language (such as YAML or Python) that is compatible with the tools an organization uses. The policies are enforced via API call to a CI pipeline, so security testing can be run without breaking current builds.

Key considerations for writing policy-as-code include

- **Dependencies.** Could testing potentially break the build or deployment? What types of findings need to be escalated to an issue-tracking system?
- **Code changes.** When was the change committed? What is the magnitude of the change? Does this warrant additional testing or manual code review?
- **Business criticality of application being tested.** Does this application handle sensitive data? Are there significant risks to downtime? What is the attack surface of this application?

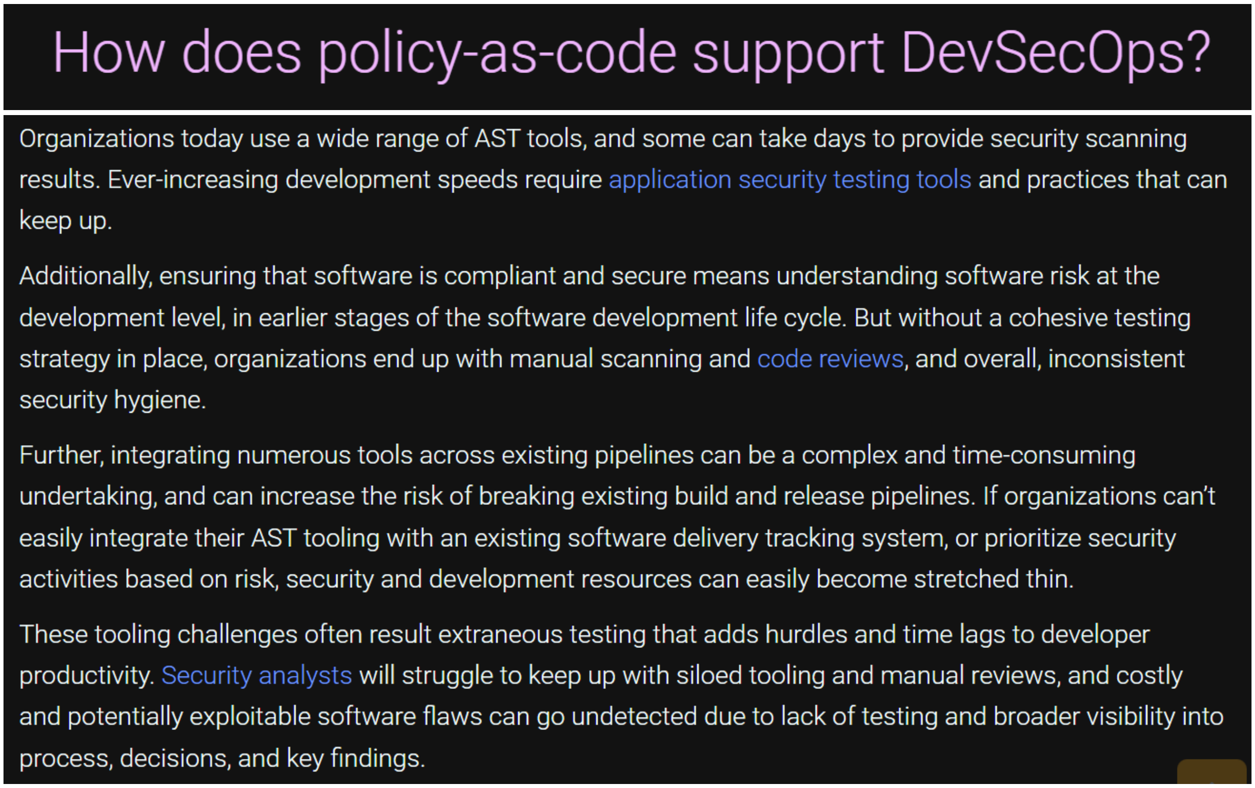
# Benefits



In the context of application security testing, organizations can leverage policy-as-code to define the conditions for when to test, what testing tool should be used, and whether there is a need to test. By codifying these parameters, security teams can simplify the coordination of multiple AST tools and achieve precision in their testing workflows. This enables consistent, automated enforcement of security policies, and ultimately, the ability to achieve better software quality without compromising development velocity.

More specifically, enforcing policy-as-code helps in these important ways.

- **It speeds up security testing.** With automated policy enforcement, security testing can be triggered without manual intervention, and only when needed.
- **It increases efficiency.** By removing manual policy enforcement from the equation, policies can be updated and shared dynamically, removing unnecessary human elements that slow the process down.
- **It helps with version control and improves visibility**. Stakeholders can easily see what is happening in their operation, and automated version control allows for seamless updates or removal of updates in case of problems associated with new versioning.
- **It minimizes mistakes and enables validation.** With automated policies in place, errors caused by human involvement are avoided. Additionally, when policies are written in code, it's easy to run validation activities and ensure accuracy.



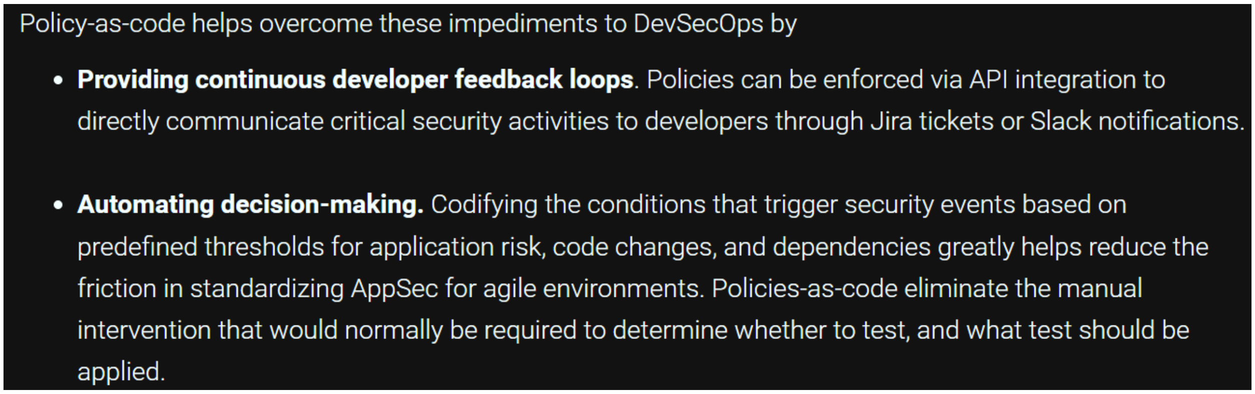
How does policy-as-code support DevSecOps?

Organizations today use a wide range of AST tools, and some can take days to provide security scanning results. Ever-increasing development speeds require application security testing tools and practices that can keep up.

Additionally, ensuring that software is compliant and secure means understanding software risk at the development level, in earlier stages of the software development life cycle. But without a cohesive testing strategy in place, organizations end up with manual scanning and code reviews, and overall, inconsistent security hygiene.

Further, integrating numerous tools across existing pipelines can be a complex and time-consuming undertaking, and can increase the risk of breaking existing build and release pipelines. If organizations can't easily integrate their AST tooling with an existing software delivery tracking system, or prioritize security activities based on risk, security and development resources can easily become stretched thin.

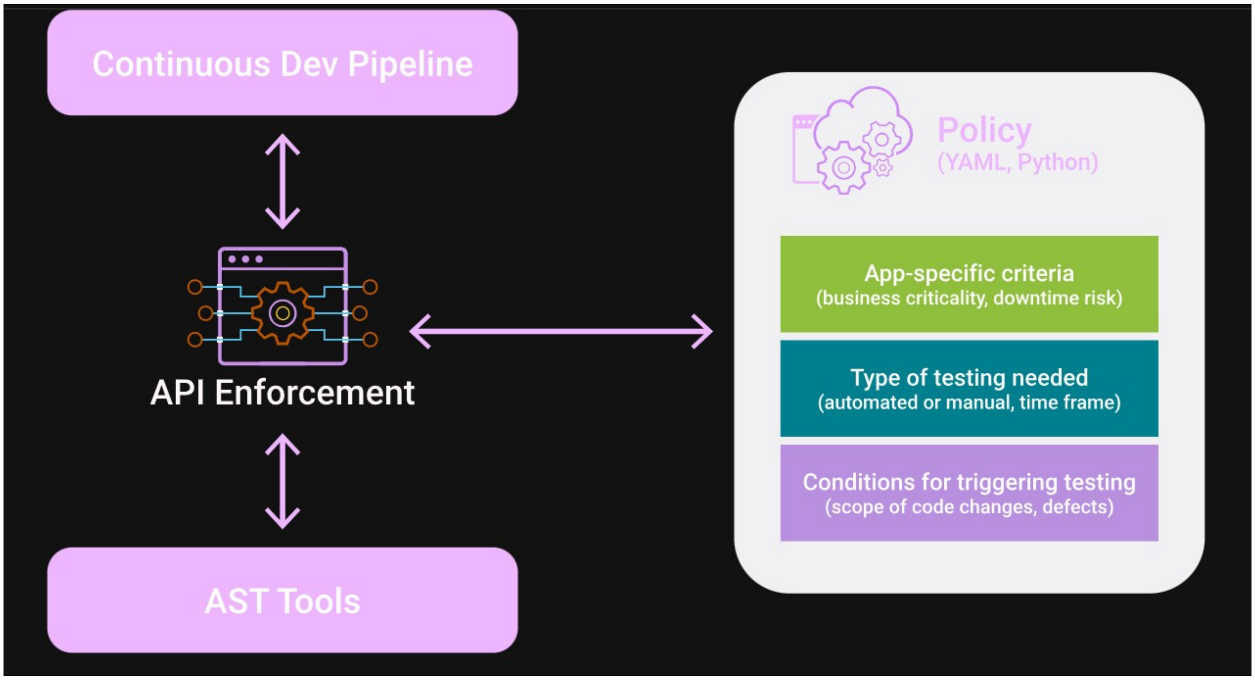
These tooling challenges often result extraneous testing that adds hurdles and time lags to developer productivity. Security analysts will struggle to keep up with siloed tooling and manual reviews, and costly and potentially exploitable software flaws can go undetected due to lack of testing and broader visibility into process, decisions, and key findings.



Policy-as-code helps overcome these impediments to DevSecOps by

- **Providing continuous developer feedback loops**. Policies can be enforced via API integration to directly communicate critical security activities to developers through Jira tickets or Slack notifications.

- **Automating decision-making.** Codifying the conditions that trigger security events based on predefined thresholds for application risk, code changes, and dependencies greatly helps reduce the friction in standardizing AppSec for agile environments. Policies-as-code eliminate the manual intervention that would normally be required to determine whether to test, and what test should be applied.



Continuous Dev Pipeline

API Enforcement

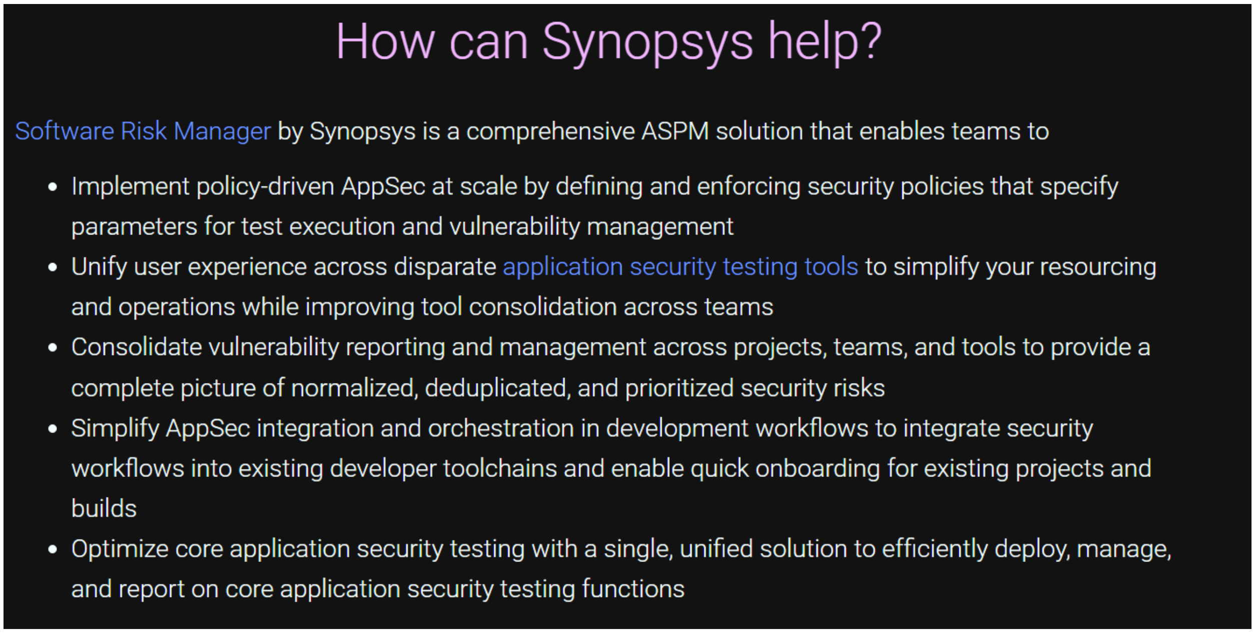
AST Tools

Policy
(YAML, Python)

App-specific criteria
(business criticality, downtime risk)

Type of testing needed
(automated or manual, time frame)

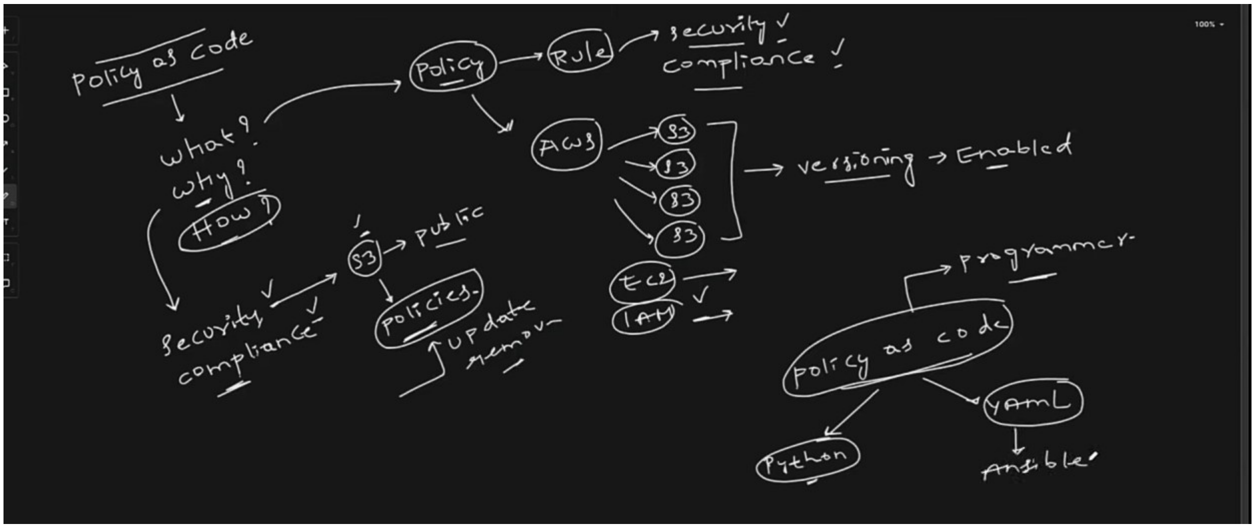
Conditions for triggering testing
(scope of code changes, defects)



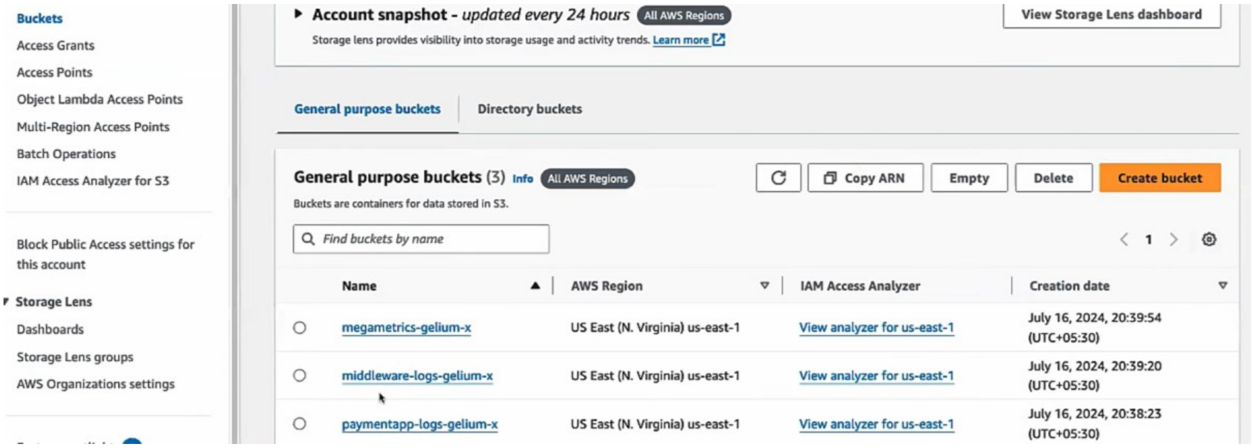
How can Synopsys help?

Software Risk Manager by Synopsys is a comprehensive ASPM solution that enables teams to

- Implement policy-driven AppSec at scale by defining and enforcing security policies that specify parameters for test execution and vulnerability management
- Unify user experience across disparate application security testing tools to simplify your resourcing and operations while improving tool consolidation across teams
- Consolidate vulnerability reporting and management across projects, teams, and tools to provide a complete picture of normalized, deduplicated, and prioritized security risks
- Simplify AppSec integration and orchestration in development workflows to integrate security workflows into existing developer toolchains and enable quick onboarding for existing projects and builds
- Optimize core application security testing with a single, unified solution to efficiently deploy, manage, and report on core application security testing functions



#Amazon S3 Buckets



- Buckets
- Access Grants
- Access Points
- Object Lambda Access Points
- Multi-Region Access Points
- Batch Operations
- IAM Access Analyzer for S3

Block Public Access settings for this account

▾ Storage Lens
- Dashboards
- Storage Lens groups
- AWS Organizations settings

▸ **Account snapshot -** *updated every 24 hours* All AWS Regions

Storage lens provides visibility into storage usage and activity trends. Learn more

View Storage Lens dashboard

General purpose buckets | Directory buckets

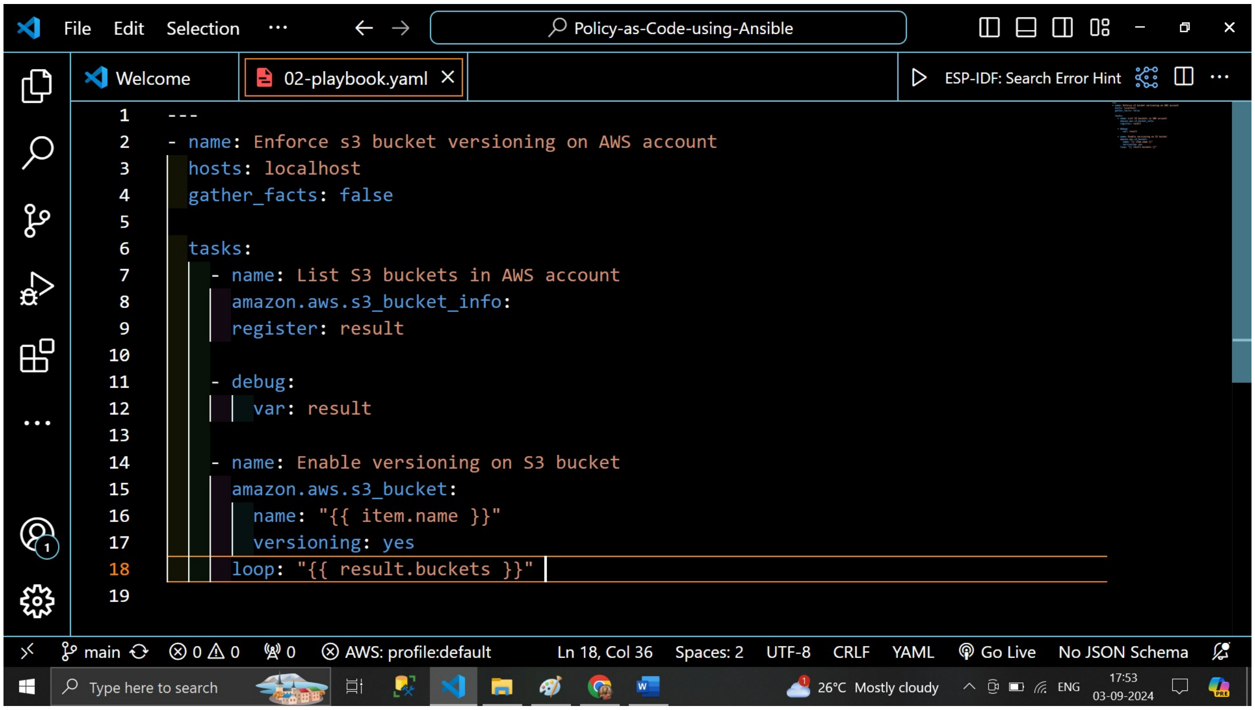
General purpose buckets (3) Info All AWS Regions

Copy ARN | Empty | Delete | Create bucket

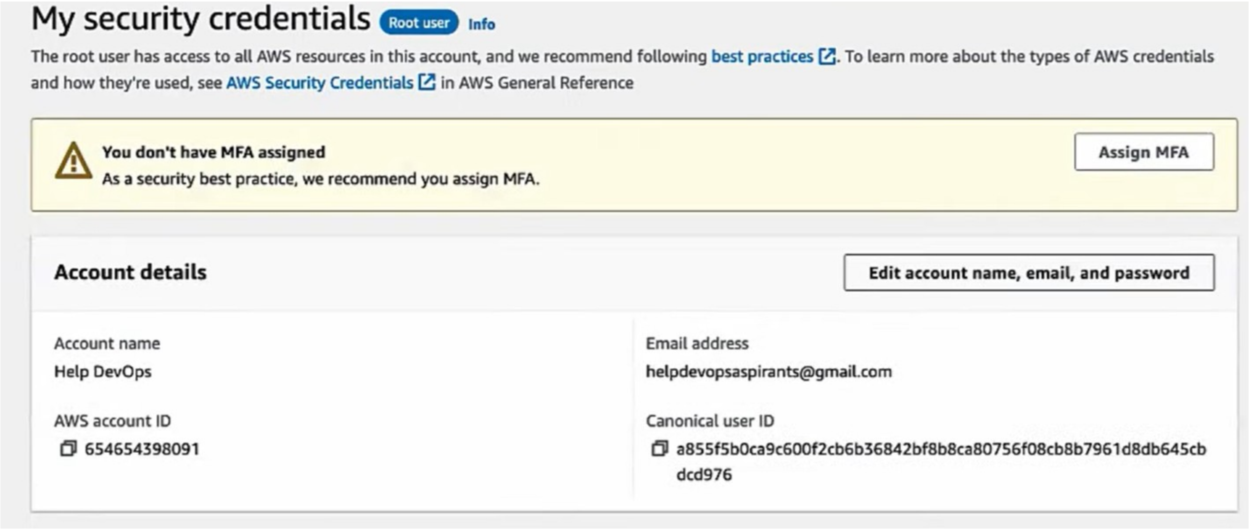
Buckets are containers for data stored in S3.

Find buckets by name

	Name ▲	AWS Region	IAM Access Analyzer	Creation date
○	megametrics-gelium-x	US East (N. Virginia) us-east-1	View analyzer for us-east-1	July 16, 2024, 20:39:54 (UTC+05:30)
○	middleware-logs-gelium-x	US East (N. Virginia) us-east-1	View analyzer for us-east-1	July 16, 2024, 20:39:20 (UTC+05:30)
○	paymentapp-logs-gelium-x	US East (N. Virginia) us-east-1	View analyzer for us-east-1	July 16, 2024, 20:38:23 (UTC+05:30)



```yaml
---
- name: Enforce s3 bucket versioning on AWS account
  hosts: localhost
  gather_facts: false

  tasks:
    - name: List S3 buckets in AWS account
      amazon.aws.s3_bucket_info:
      register: result

    - debug:
        var: result

    - name: Enable versioning on S3 bucket
      amazon.aws.s3_bucket:
        name: "{{ item.name }}"
        versioning: yes
      loop: "{{ result.buckets }}"

```



My security credentials Root user Info

The root user has access to all AWS resources in this account, and we recommend following best practices. To learn more about the types of AWS credentials and how they're used, see AWS Security Credentials in AWS General Reference

You don't have MFA assigned

As a security best practice, we recommend you assign MFA.

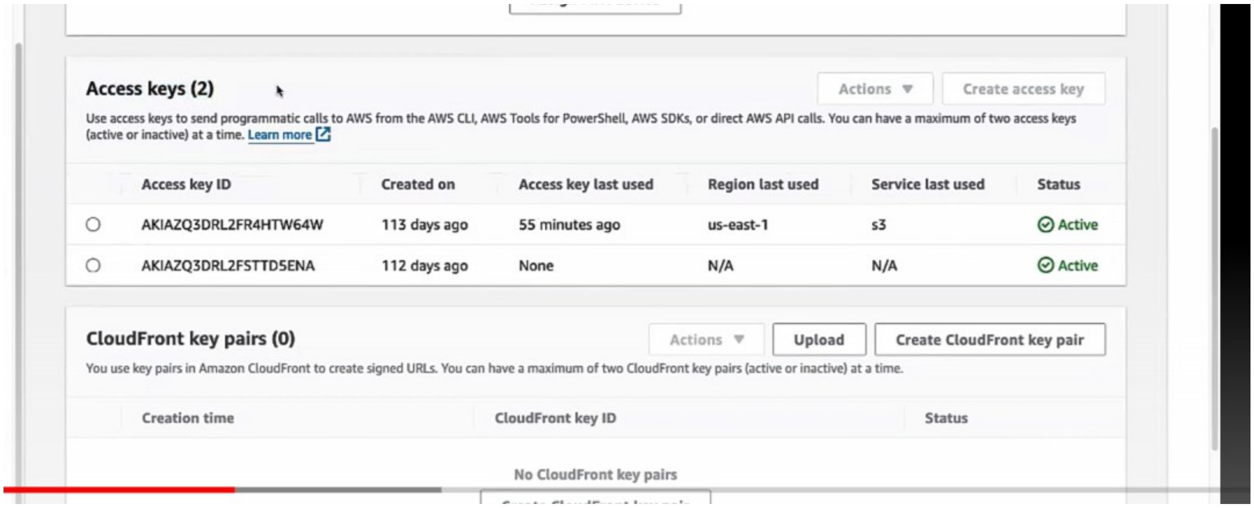
Account details

Account name
Help DevOps

AWS account ID
654654398091

Email address
helpdevopsaspirants@gmail.com

Canonical user ID
a855f5b0ca9c600f2cb6b36842bf8b8ca80756f08cb8b7961d8db645cbdcd976



Access keys (2)

Use access keys to send programmatic calls to AWS from the AWS CLI, AWS Tools for PowerShell, AWS SDKs, or direct AWS API calls. You can have a maximum of two access keys (active or inactive) at a time. Learn more

	Access key ID	Created on	Access key last used	Region last used	Service last used	Status
○	AKIAZQ3DRL2FR4HTW64W	113 days ago	55 minutes ago	us-east-1	s3	✓ Active
○	AKIAZQ3DRL2FSTTD5ENA	112 days ago	None	N/A	N/A	✓ Active

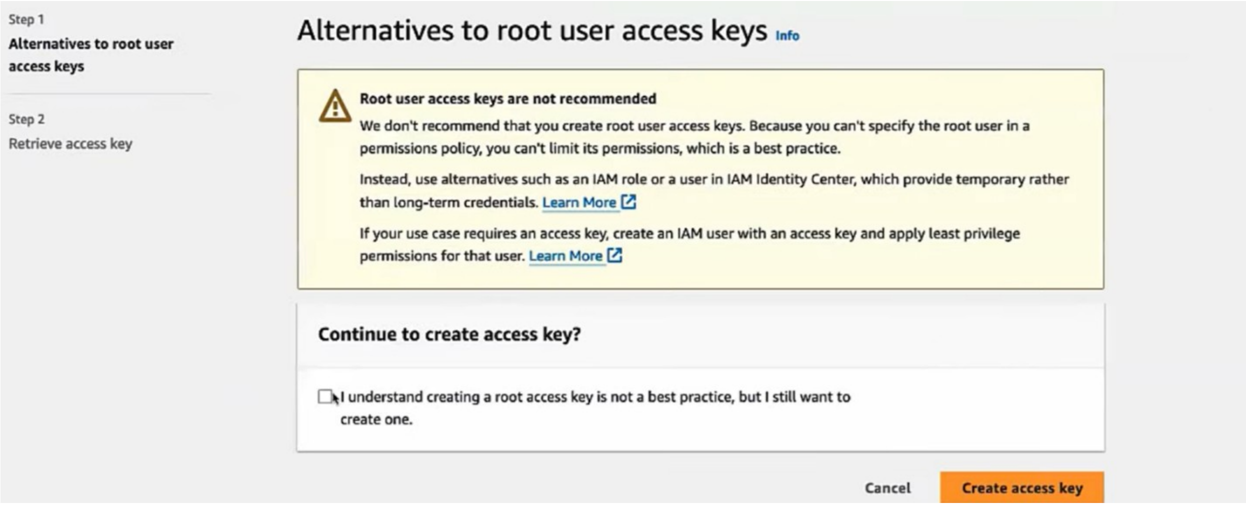
CloudFront key pairs (0)

You use key pairs in Amazon CloudFront to create signed URLs. You can have a maximum of two CloudFront key pairs (active or inactive) at a time.

	Creation time	CloudFront key ID	Status

No CloudFront key pairs

Create Access Key



Alternatives to root user access keys Info

Root user access keys are not recommended

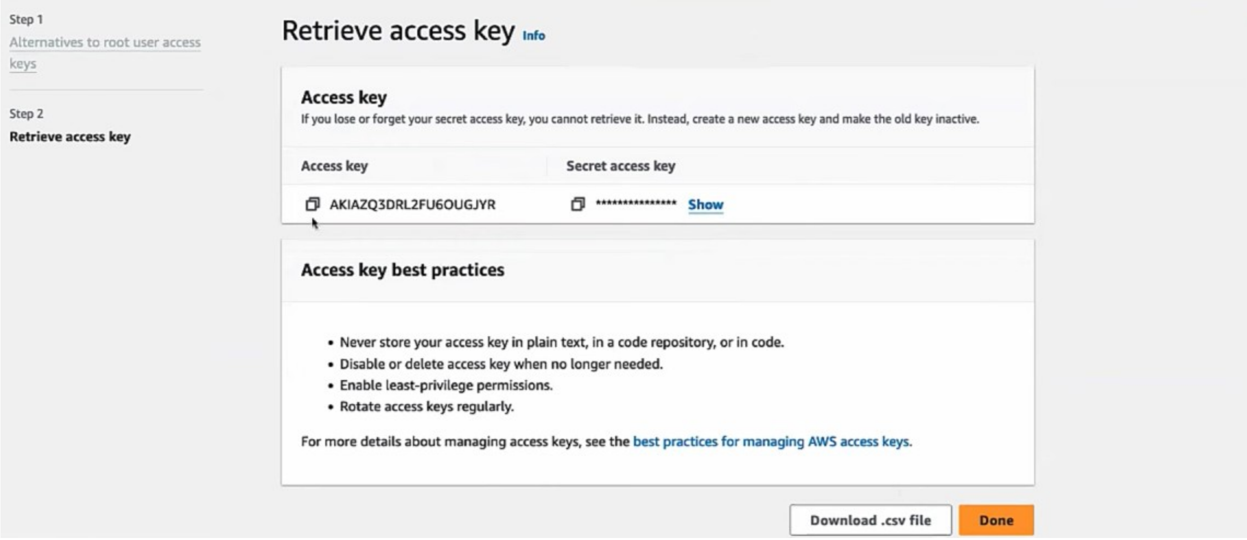
We don't recommend that you create root user access keys. Because you can't specify the root user in a permissions policy, you can't limit its permissions, which is a best practice.

Instead, use alternatives such as an IAM role or a user in IAM Identity Center, which provide temporary rather than long-term credentials. Learn More

If your use case requires an access key, create an IAM user with an access key and apply least privilege permissions for that user. Learn More

Continue to create access key?

□ I understand creating a root access key is not a best practice, but I still want to create one.



Retrieve access key Info

Access key

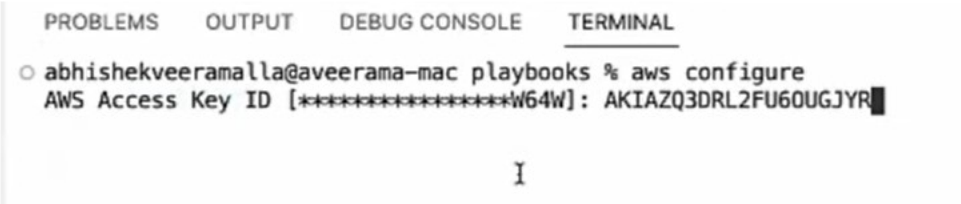
If you lose or forget your secret access key, you cannot retrieve it. Instead, create a new access key and make the old key inactive.

Access key	Secret access key
AKIAZQ3DRL2FU6OUGJYR	*************** Show

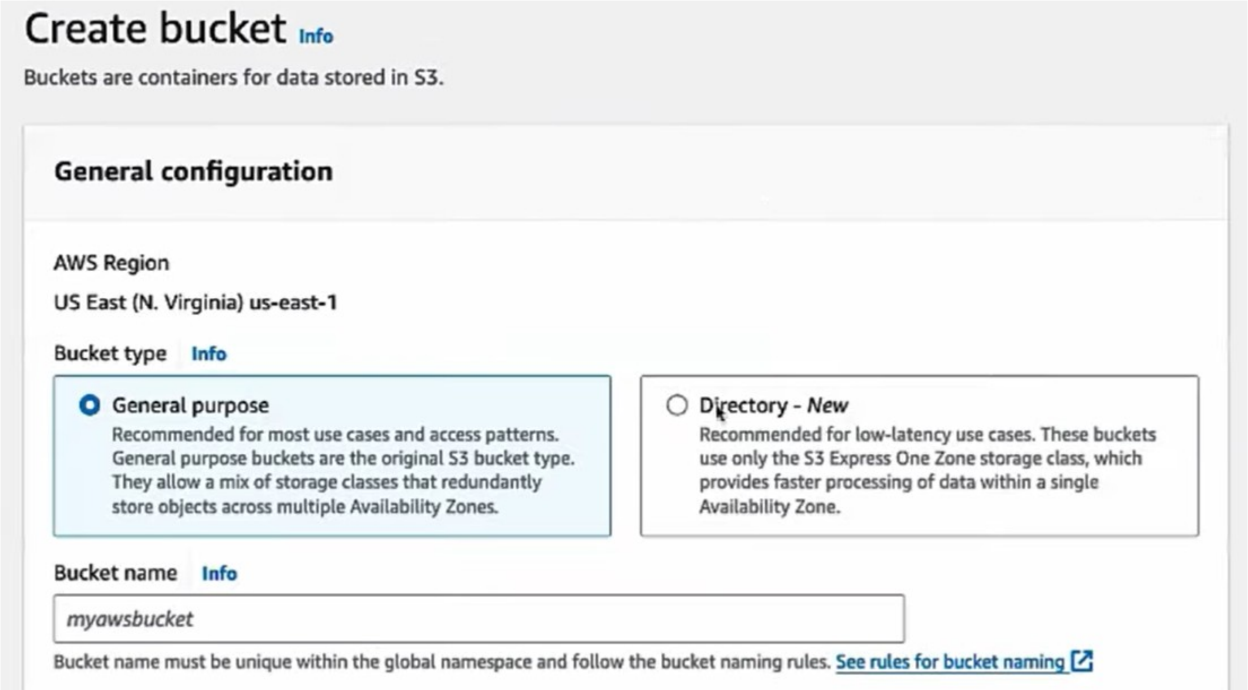
Access key best practices

- Never store your access key in plain text, in a code repository, or in code.
- Disable or delete access key when no longer needed.
- Enable least-privilege permissions.
- Rotate access keys regularly.

For more details about managing access keys, see the best practices for managing AWS access keys.



```
○ abhishekveeramalla@aveerama-mac playbooks % aws configure
  AWS Access Key ID [****************W64W]: AKIAZQ3DRL2FU6OUGJYR
```



Create bucket

Buckets are containers for data stored in S3.

General configuration

AWS Region

US East (N. Virginia) us-east-1

Bucket type

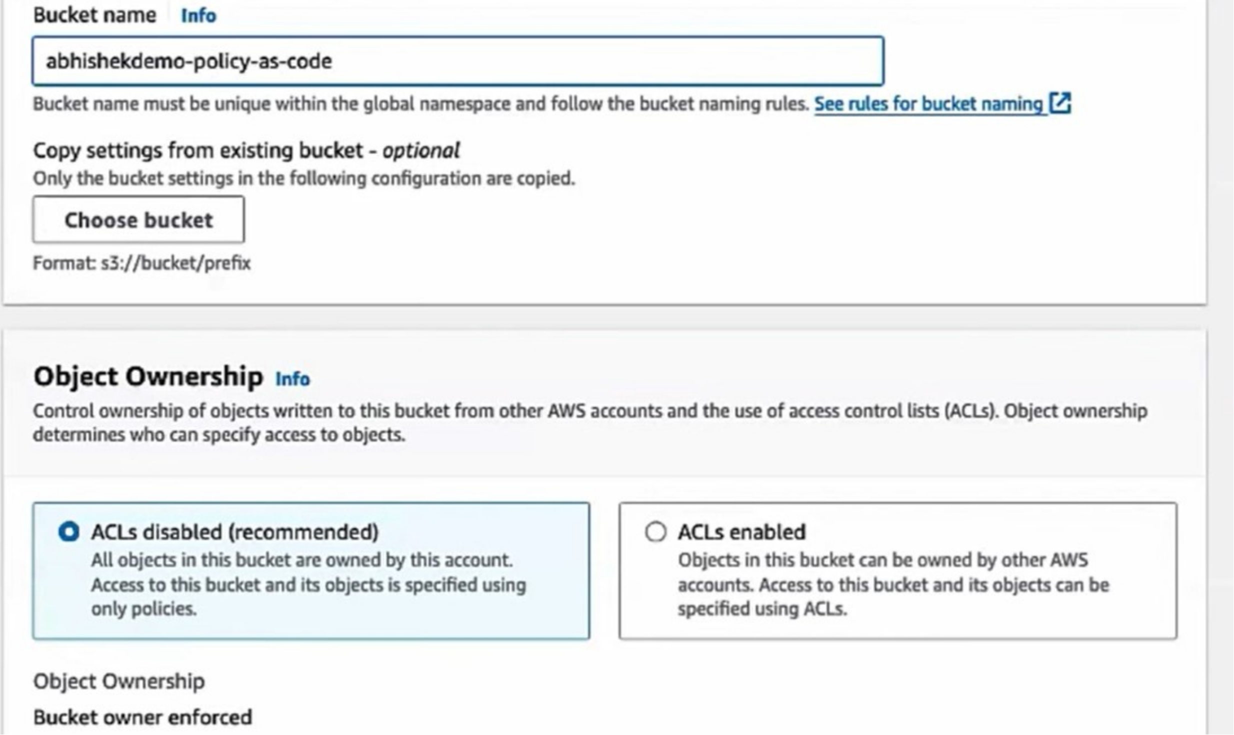
● General purpose
Recommended for most use cases and access patterns. General purpose buckets are the original S3 bucket type. They allow a mix of storage classes that redundantly store objects across multiple Availability Zones.

○ Directory - *New*
Recommended for low-latency use cases. These buckets use only the S3 Express One Zone storage class, which provides faster processing of data within a single Availability Zone.

Bucket name

myawsbucket

Bucket name must be unique within the global namespace and follow the bucket naming rules. See rules for bucket naming



Bucket name

abhishekdemo-policy-as-code

Bucket name must be unique within the global namespace and follow the bucket naming rules. See rules for bucket naming

Copy settings from existing bucket - *optional*
Only the bucket settings in the following configuration are copied.

Format: s3://bucket/prefix

Object Ownership

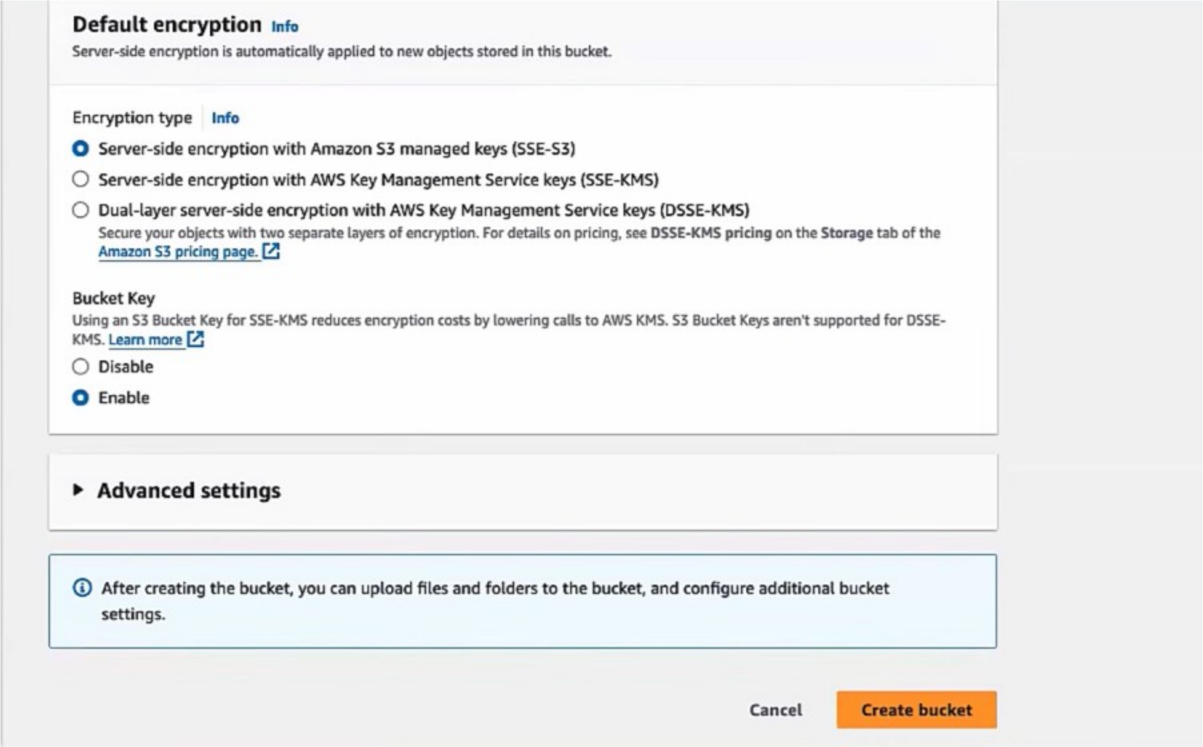
Control ownership of objects written to this bucket from other AWS accounts and the use of access control lists (ACLs). Object ownership determines who can specify access to objects.

● ACLs disabled (recommended)
All objects in this bucket are owned by this account. Access to this bucket and its objects is specified using only policies.

○ ACLs enabled
Objects in this bucket can be owned by other AWS accounts. Access to this bucket and its objects can be specified using ACLs.

Object Ownership

Bucket owner enforced



Default encryption Info

Server-side encryption is automatically applied to new objects stored in this bucket.

Encryption type | Info

● Server-side encryption with Amazon S3 managed keys (SSE-S3)

○ Server-side encryption with AWS Key Management Service keys (SSE-KMS)

○ Dual-layer server-side encryption with AWS Key Management Service keys (DSSE-KMS)
Secure your objects with two separate layers of encryption. For details on pricing, see **DSSE-KMS pricing** on the **Storage** tab of the Amazon S3 pricing page.

Bucket Key

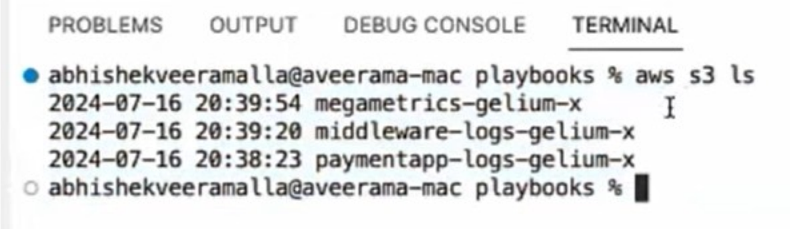
Using an S3 Bucket Key for SSE-KMS reduces encryption costs by lowering calls to AWS KMS. S3 Bucket Keys aren't supported for DSSE-KMS. Learn more

○ Disable

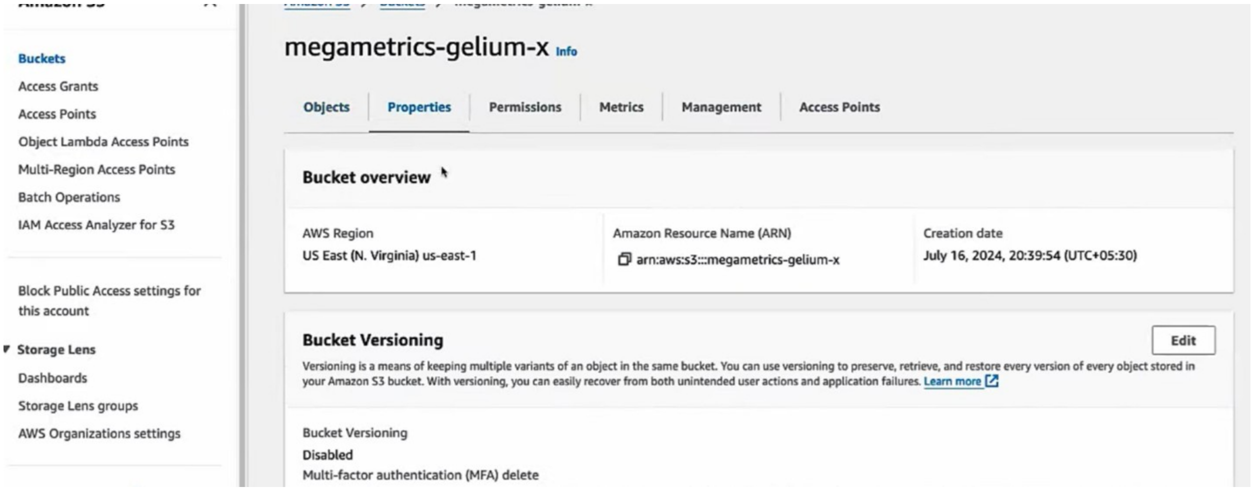
● Enable

▶ **Advanced settings**

ⓘ After creating the bucket, you can upload files and folders to the bucket, and configure additional bucket settings.



```
● abhishekveeramalla@aveerama-mac playbooks % aws s3 ls
  2024-07-16 20:39:54 megametrics-gelium-x
  2024-07-16 20:39:20 middleware-logs-gelium-x
  2024-07-16 20:38:23 paymentapp-logs-gelium-x
○ abhishekveeramalla@aveerama-mac playbooks % 
```



Buckets
Access Grants
Access Points
Object Lambda Access Points
Multi-Region Access Points
Batch Operations
IAM Access Analyzer for S3
Block Public Access settings for this account
▾ Storage Lens
Dashboards
Storage Lens groups
AWS Organizations settings

megametrics-gelium-x Info

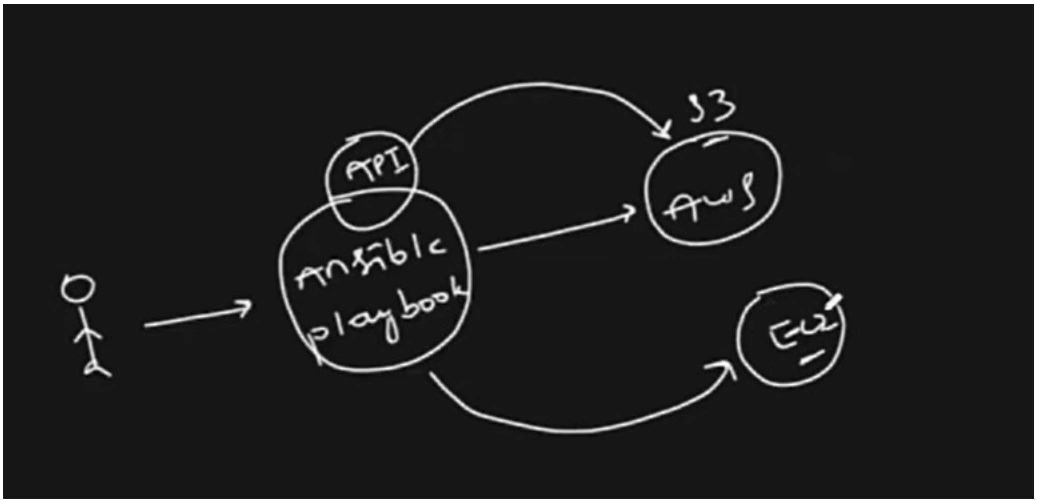
Bucket overview

AWS Region	Amazon Resource Name (ARN)	Creation date
US East (N. Virginia) us-east-1	arn:aws:s3:::megametrics-gelium-x	July 16, 2024, 20:39:54 (UTC+05:30)

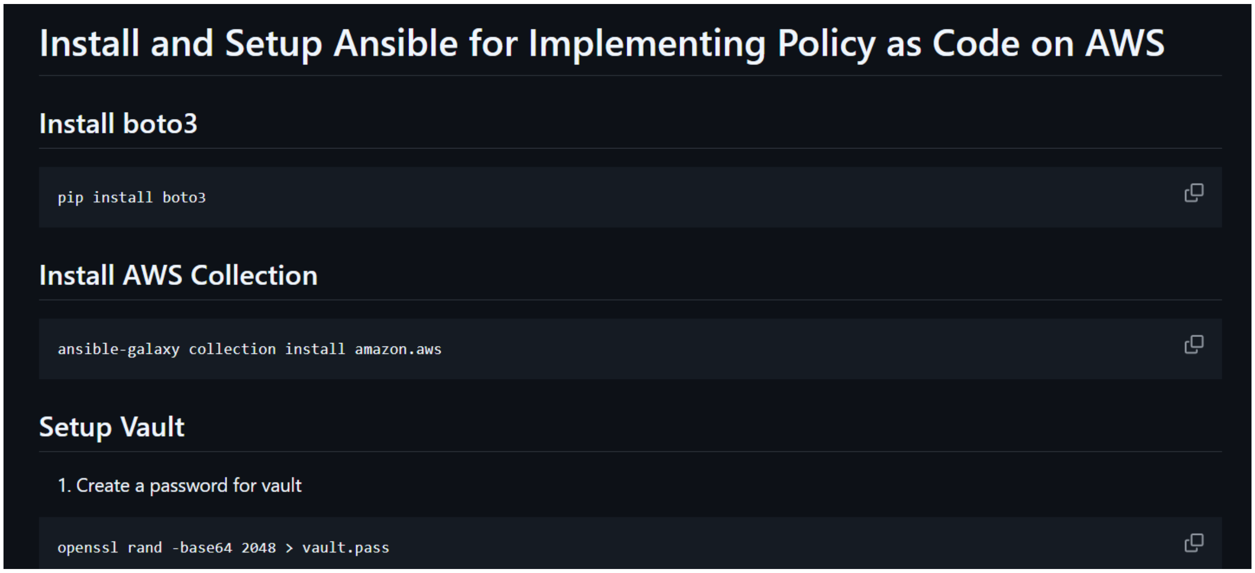
Bucket Versioning

Versioning is a means of keeping multiple variants of an object in the same bucket. You can use versioning to preserve, retrieve, and restore every version of every object stored in your Amazon S3 bucket. With versioning, you can easily recover from both unintended user actions and application failures. Learn more

Bucket Versioning
Disabled
Multi-factor authentication (MFA) delete



# Pre-requisites



Install and Setup Ansible for Implementing Policy as Code on AWS

Install boto3

```
pip install boto3
```

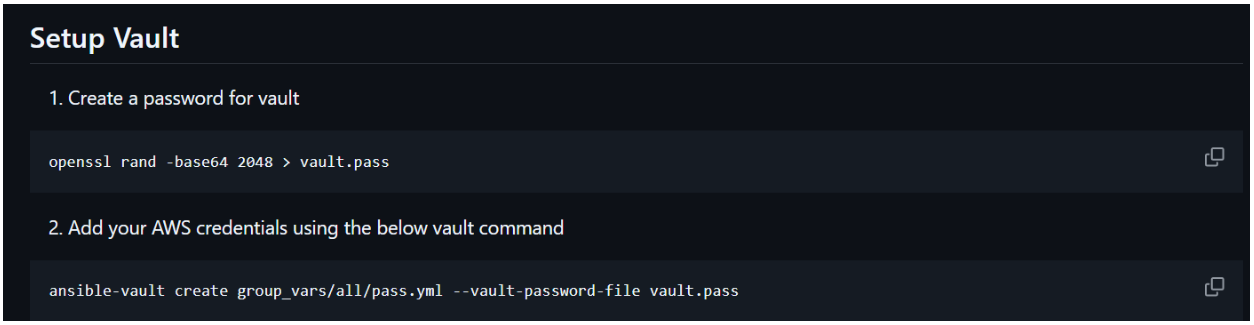
Install AWS Collection

```
ansible-galaxy collection install amazon.aws
```

Setup Vault

1. Create a password for vault

```
openssl rand -base64 2048 > vault.pass
```



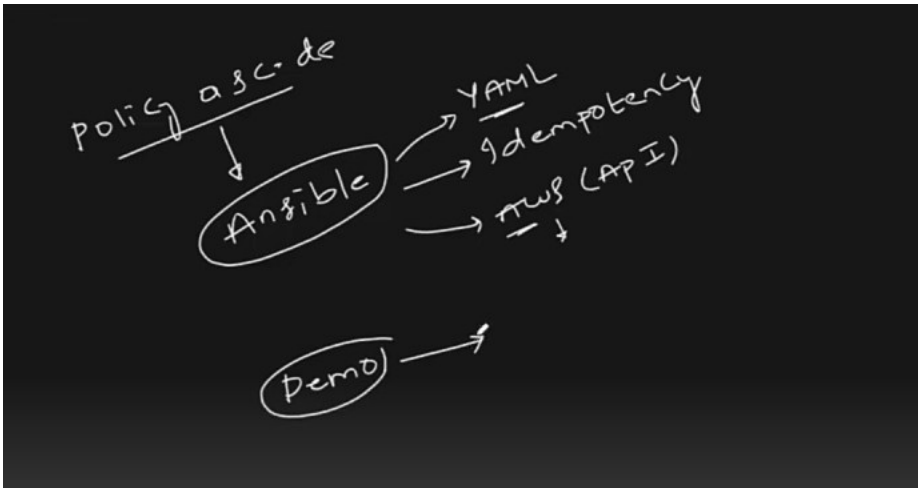
Setup Vault

1. Create a password for vault

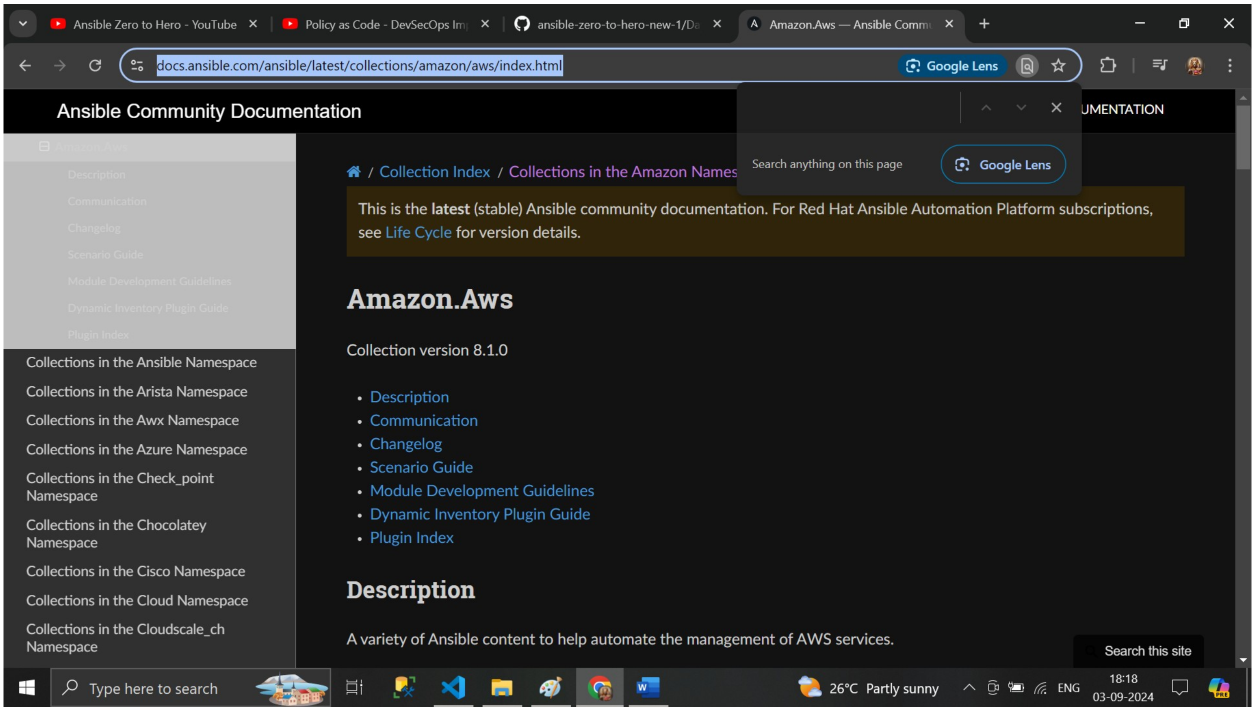
```
openssl rand -base64 2048 > vault.pass
```

2. Add your AWS credentials using the below vault command

```
ansible-vault create group_vars/all/pass.yml --vault-password-file vault.pass
```



https://docs.ansible.com/ansible/latest/collections/amazon/aws/index.html


Ansible Zero to Hero - YouTube
Policy as Code - DevSecOps Im
ansible-zero-to-hero-new-1/D
Amazon.Aws — Ansible Comm
docs.ansible.com/ansible/latest/collections/amazon/aws/index.html
Google Lens
Search anything on this page
Google Lens
UMENTATION
Search this site
Type here to search
26°C Partly sunny
ENG
18:18
03-09-2024

Ansible Community Documentation

- Amazon.Aws
  - Description
  - Communication
  - Changelog
  - Scenario Guide
  - Module Development Guidelines
  - Dynamic Inventory Plugin Guide
  - Plugin Index
- Collections in the Ansible Namespace
- Collections in the Arista Namespace
- Collections in the Awx Namespace
- Collections in the Azure Namespace
- Collections in the Check_point Namespace
- Collections in the Chocolatey Namespace
- Collections in the Cisco Namespace
- Collections in the Cloud Namespace
- Collections in the Cloudscale_ch Namespace

/ Collection Index / Collections in the Amazon Names

This is the **latest** (stable) Ansible community documentation. For Red Hat Ansible Automation Platform subscriptions, see Life Cycle for version details.

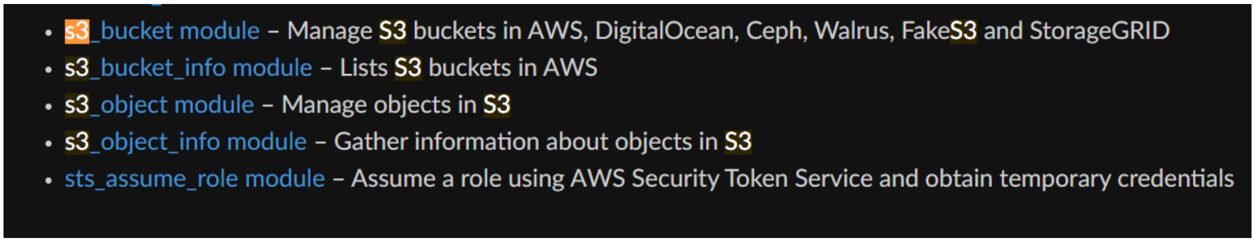
Amazon.Aws

Collection version 8.1.0

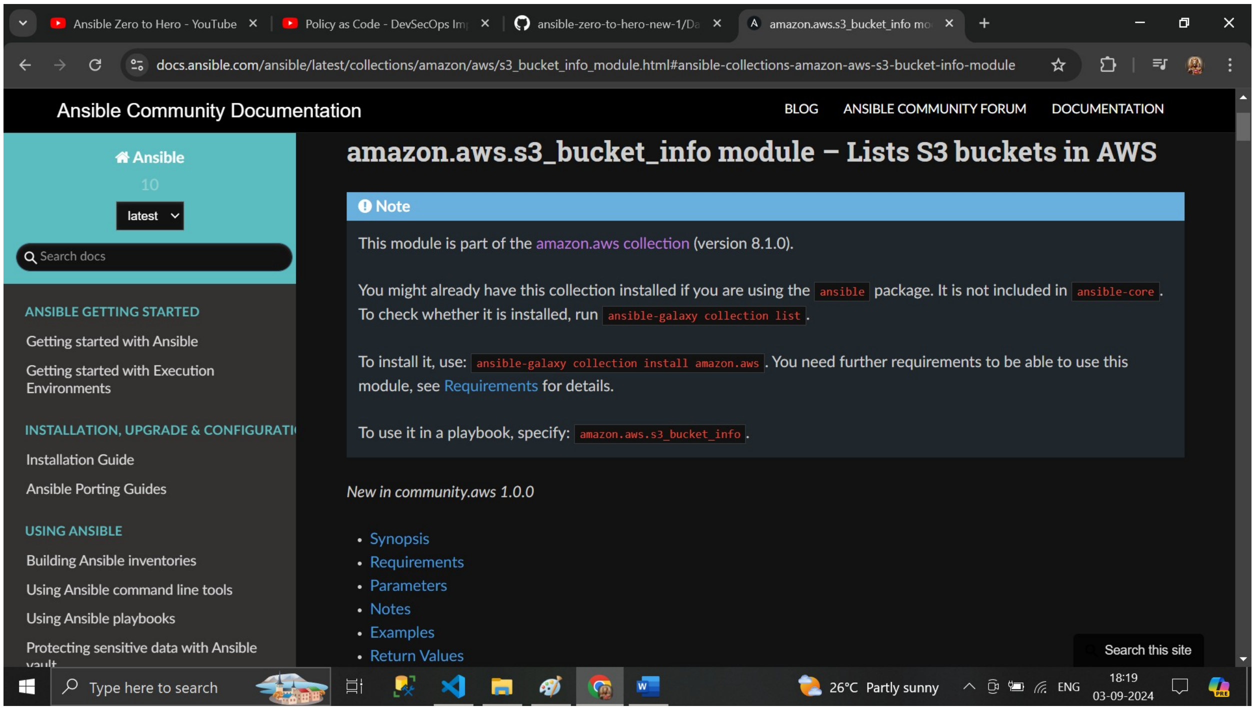
- Description
- Communication
- Changelog
- Scenario Guide
- Module Development Guidelines
- Dynamic Inventory Plugin Guide
- Plugin Index

Description

A variety of Ansible content to help automate the management of AWS services.



- s3_bucket module – Manage S3 buckets in AWS, DigitalOcean, Ceph, Walrus, FakeS3 and StorageGRID
- s3_bucket_info module – Lists S3 buckets in AWS
- s3_object module – Manage objects in S3
- s3_object_info module – Gather information about objects in S3
- sts_assume_role module – Assume a role using AWS Security Token Service and obtain temporary credentials



Ansible Zero to Hero - YouTube
Policy as Code - DevSecOps Im
ansible-zero-to-hero-new-1/D
amazon.aws.s3_bucket_info mo
docs.ansible.com/ansible/latest/collections/amazon/aws/s3_bucket_info_module.html#ansible-collections-amazon-aws-s3-bucket-info-module
Ansible Community Documentation
BLOG
ANSIBLE COMMUNITY FORUM
DOCUMENTATION
Ansible
10
latest
Search docs
ANSIBLE GETTING STARTED
Getting started with Ansible
Getting started with Execution Environments
INSTALLATION, UPGRADE & CONFIGURATI
Installation Guide
Ansible Porting Guides
USING ANSIBLE
Building Ansible inventories
Using Ansible command line tools
Using Ansible playbooks
Protecting sensitive data with Ansible vault
Search this site
Type here to search
26°C Partly sunny
ENG
18:19
03-09-2024

amazon.aws.s3_bucket_info module – Lists S3 buckets in AWS

Note

This module is part of the amazon.aws collection (version 8.1.0).

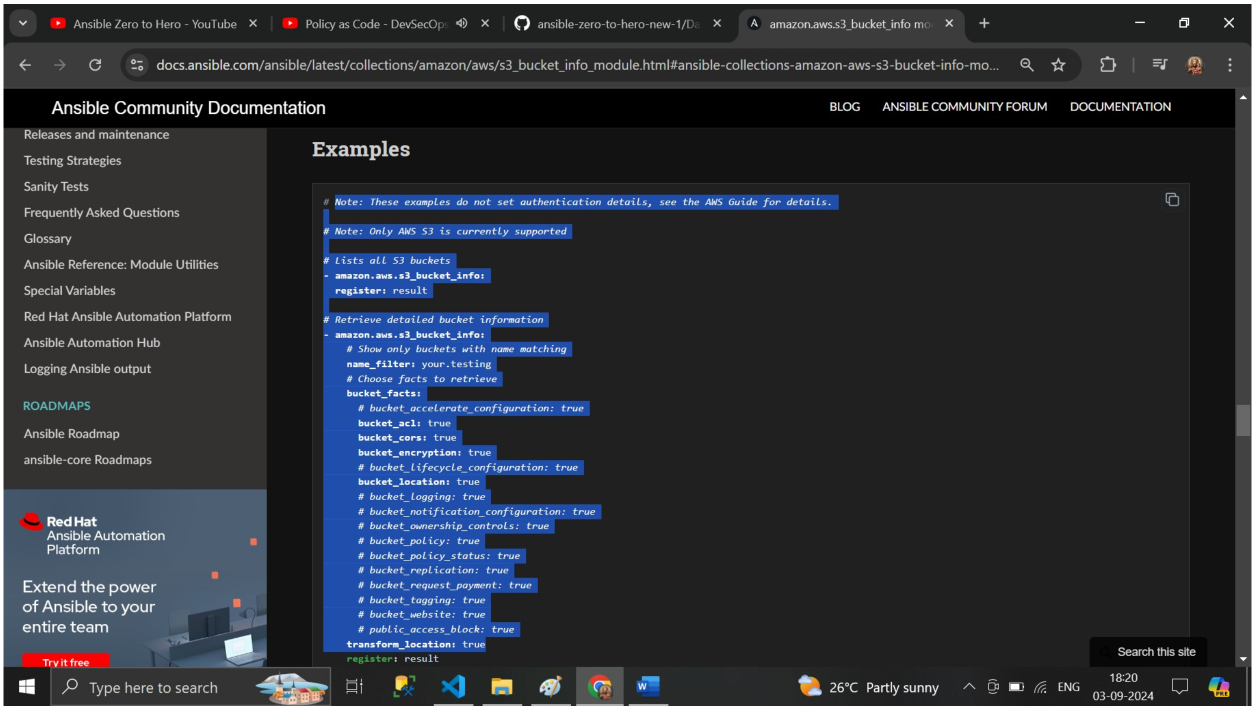
You might already have this collection installed if you are using the `ansible` package. It is not included in `ansible-core`. To check whether it is installed, run `ansible-galaxy collection list`.

To install it, use: `ansible-galaxy collection install amazon.aws`. You need further requirements to be able to use this module, see Requirements for details.

To use it in a playbook, specify: `amazon.aws.s3_bucket_info`.

New in community.aws 1.0.0

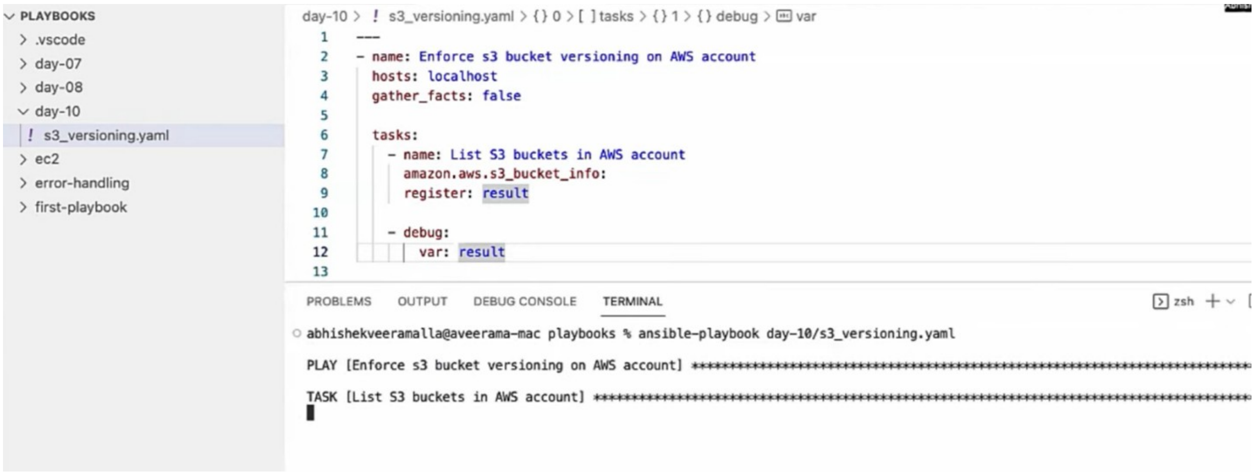
- Synopsis
- Requirements
- Parameters
- Notes
- Examples
- Return Values



Ansible Zero to Hero - YouTube
Policy as Code - DevSecOp
ansible-zero-to-hero-new-1/D
amazon.aws.s3_bucket_info mo
docs.ansible.com/ansible/latest/collections/amazon/aws/s3_bucket_info_module.html#ansible-collections-amazon-aws-s3-bucket-info-mo...
Ansible Community Documentation
BLOG
ANSIBLE COMMUNITY FORUM
DOCUMENTATION
Releases and maintenance
Testing Strategies
Sanity Tests
Frequently Asked Questions
Glossary
Ansible Reference: Module Utilities
Special Variables
Red Hat Ansible Automation Platform
Ansible Automation Hub
Logging Ansible output
ROADMAPS
Ansible Roadmap
ansible-core Roadmaps
Red Hat Ansible Automation Platform
Extend the power of Ansible to your entire team
Try it free
Search this site
Type here to search
26°C Partly sunny
ENG
18:20
03-09-2024

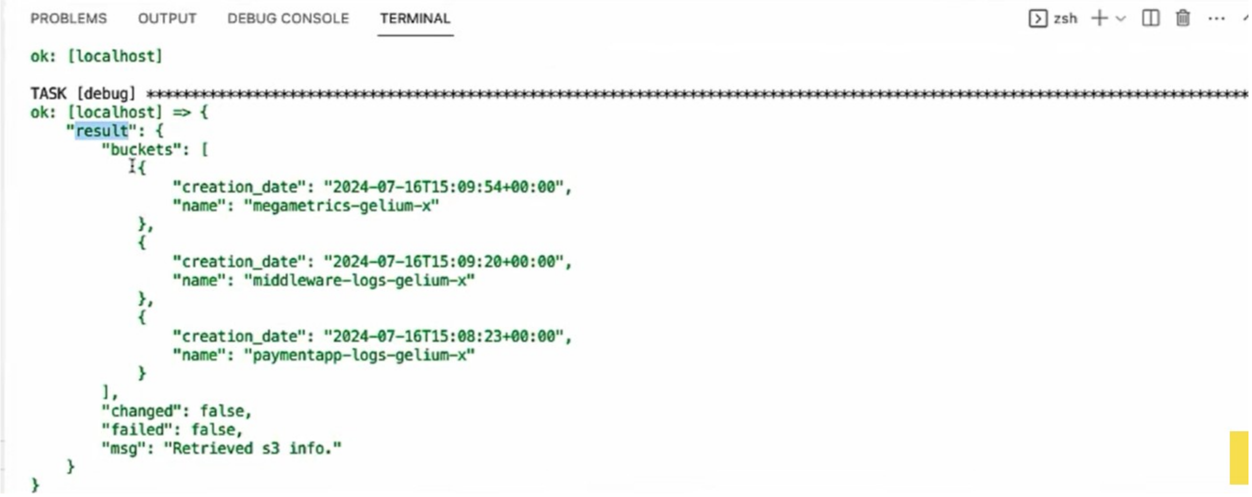
Examples

```
# Note: These examples do not set authentication details, see the AWS Guide for details.

# Note: Only AWS S3 is currently supported

# Lists all S3 buckets
- amazon.aws.s3_bucket_info:
  register: result

# Retrieve detailed bucket information
- amazon.aws.s3_bucket_info:
    # Show only buckets with name matching
    name_filter: your.testing
    # Choose facts to retrieve
    bucket_facts:
      # bucket_accelerate_configuration: true
      bucket_acl: true
      bucket_cors: true
      bucket_encryption: true
      # bucket_lifecycle_configuration: true
      bucket_location: true
      # bucket_logging: true
      # bucket_notification_configuration: true
      # bucket_ownership_controls: true
      # bucket_policy: true
      # bucket_policy_status: true
      # bucket_replication: true
      # bucket_request_payment: true
      # bucket_tagging: true
      # bucket_website: true
      # public_access_block: true
    transform_location: true
    register: result
```

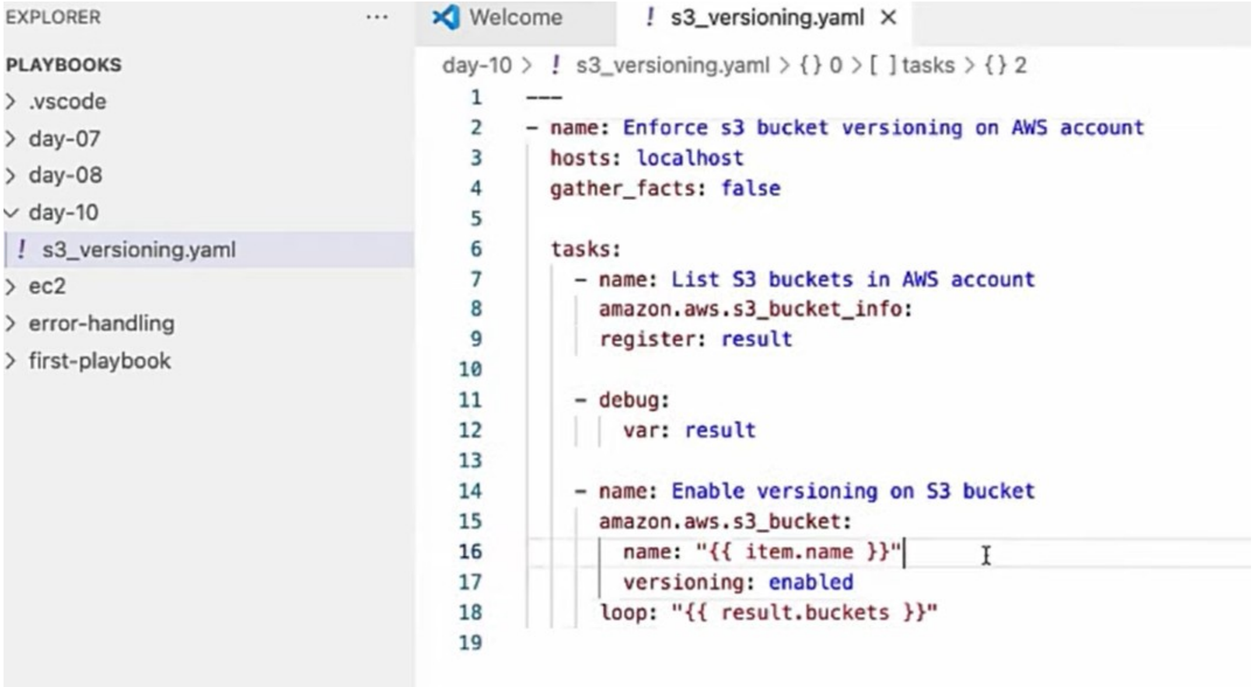


```yaml
---
- name: Enforce s3 bucket versioning on AWS account
  hosts: localhost
  gather_facts: false

  tasks:
    - name: List S3 buckets in AWS account
      amazon.aws.s3_bucket_info:
      register: result

    - debug:
        var: result

```

```
abhishekveeramalla@aveerama-mac playbooks % ansible-playbook day-10/s3_versioning.yaml

PLAY [Enforce s3 bucket versioning on AWS account] ******************************************************************************

TASK [List S3 buckets in AWS account] ****************************************************************************************
```

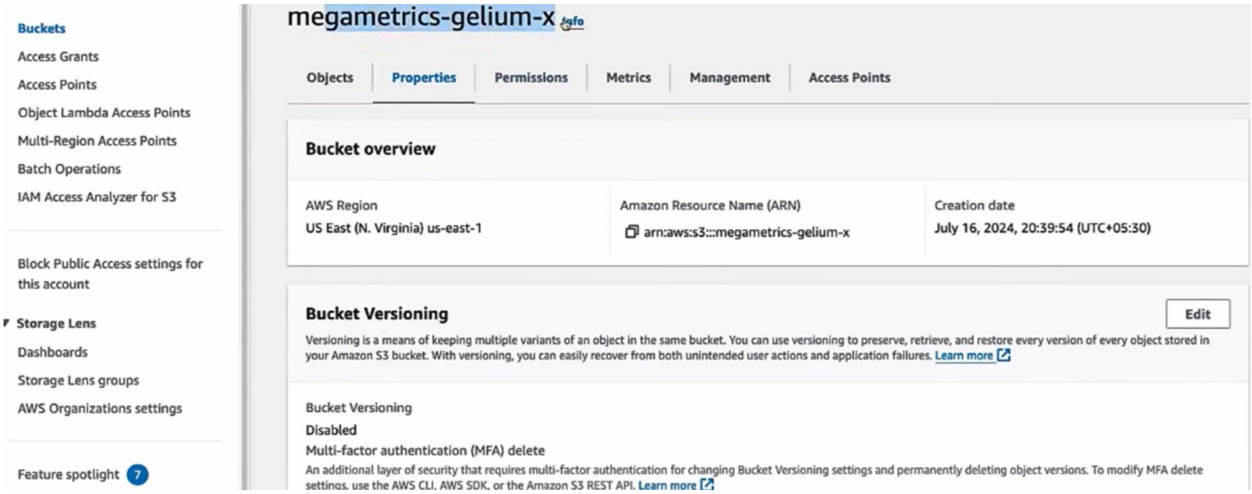


```
ok: [localhost]

TASK [debug] ******************************************************************************************************************
ok: [localhost] => {
    "result": {
        "buckets": [
            {
                "creation_date": "2024-07-16T15:09:54+00:00",
                "name": "megametrics-gelium-x"
            },
            {
                "creation_date": "2024-07-16T15:09:20+00:00",
                "name": "middleware-logs-gelium-x"
            },
            {
                "creation_date": "2024-07-16T15:08:23+00:00",
                "name": "paymentapp-logs-gelium-x"
            }
        ],
        "changed": false,
        "failed": false,
        "msg": "Retrieved s3 info."
    }
}
```



```yaml
---
- name: Enforce s3 bucket versioning on AWS account
  hosts: localhost
  gather_facts: false

  tasks:
    - name: List S3 buckets in AWS account
      amazon.aws.s3_bucket_info:
      register: result

    - debug:
        var: result

    - name: Enable versioning on S3 bucket
      amazon.aws.s3_bucket:
        name: "{{ item.name }}"
        versioning: enabled
      loop: "{{ result.buckets }}"

```

# Bucket Versioning is Disabled



megametrics-gelium-x

Bucket overview

AWS Region
US East (N. Virginia) us-east-1

Amazon Resource Name (ARN)
arn:aws:s3:::megametrics-gelium-x

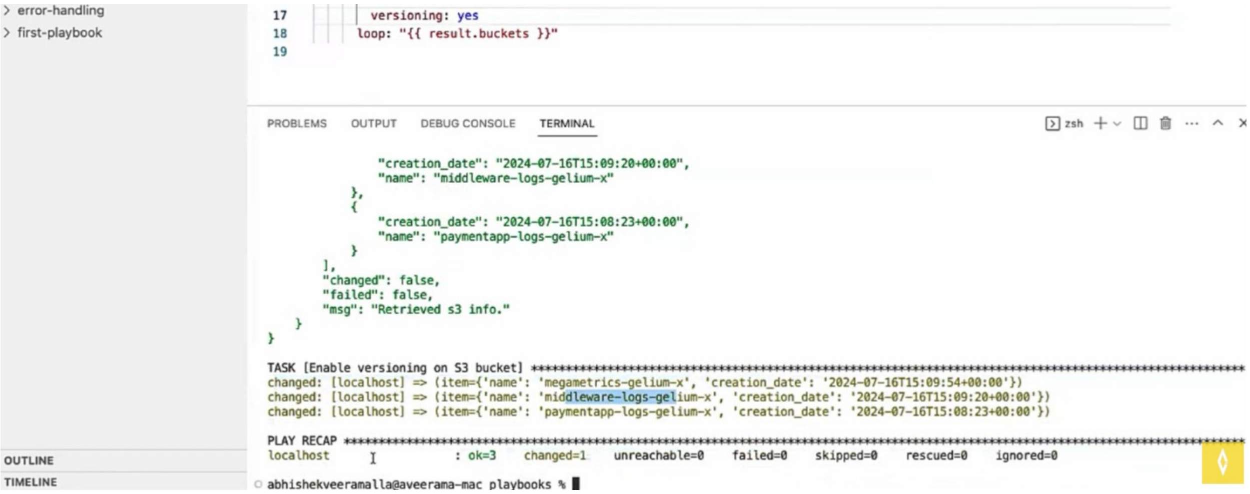
Creation date
July 16, 2024, 20:39:54 (UTC+05:30)

Bucket Versioning

Versioning is a means of keeping multiple variants of an object in the same bucket. You can use versioning to preserve, retrieve, and restore every version of every object stored in your Amazon S3 bucket. With versioning, you can easily recover from both unintended user actions and application failures. Learn more

Bucket Versioning
Disabled

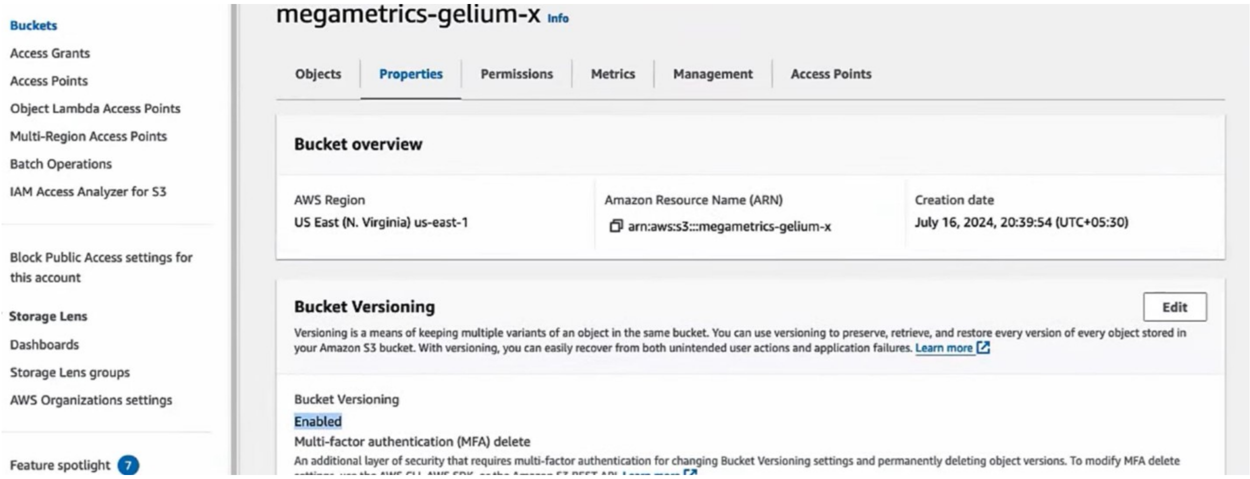
Multi-factor authentication (MFA) delete
An additional layer of security that requires multi-factor authentication for changing Bucket Versioning settings and permanently deleting object versions. To modify MFA delete settings, use the AWS CLI, AWS SDK, or the Amazon S3 REST API. Learn more



```
        versioning: yes
      loop: "{{ result.buckets }}"
```

```
                "creation_date": "2024-07-16T15:09:20+00:00",
                "name": "middleware-logs-gelium-x"
            },
            {
                "creation_date": "2024-07-16T15:08:23+00:00",
                "name": "paymentapp-logs-gelium-x"
            }
        ],
        "changed": false,
        "failed": false,
        "msg": "Retrieved s3 info."
    }
}

TASK [Enable versioning on S3 bucket] ****************************************************************************************
changed: [localhost] => (item={'name': 'megametrics-gelium-x', 'creation_date': '2024-07-16T15:09:54+00:00'})
changed: [localhost] => (item={'name': 'middleware-logs-gelium-x', 'creation_date': '2024-07-16T15:09:20+00:00'})
changed: [localhost] => (item={'name': 'paymentapp-logs-gelium-x', 'creation_date': '2024-07-16T15:08:23+00:00'})

PLAY RECAP ****************************************************************************************************************
localhost                  : ok=3    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

abhishekveeramalla@aveerama-mac playbooks % 
```

# **Bucket Versioning is Enabled – O/P**



megametrics-gelium-x

Bucket overview

AWS Region
US East (N. Virginia) us-east-1

Amazon Resource Name (ARN)
arn:aws:s3:::megametrics-gelium-x

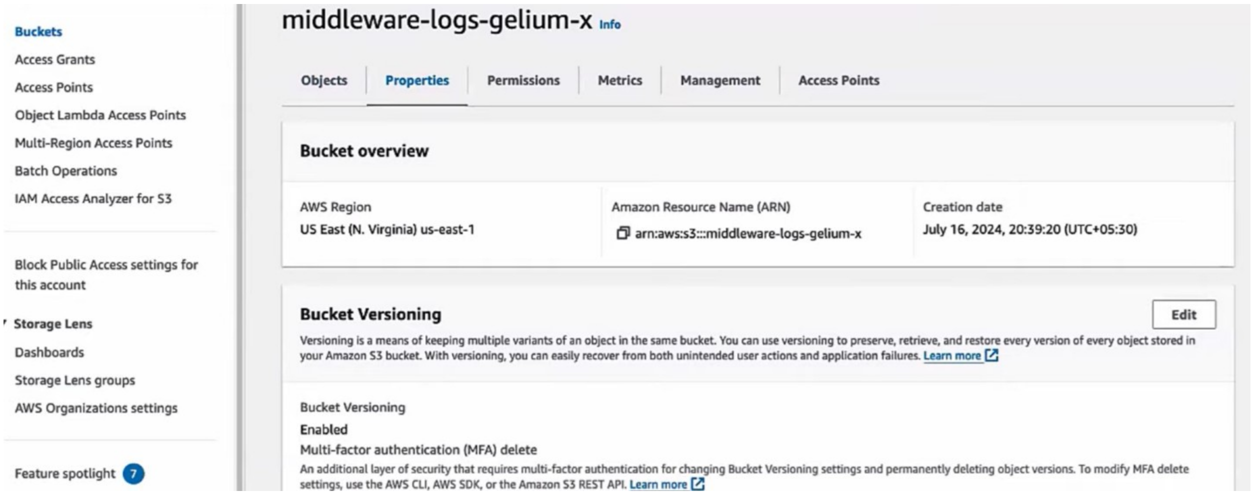
Creation date
July 16, 2024, 20:39:54 (UTC+05:30)

Bucket Versioning

Versioning is a means of keeping multiple variants of an object in the same bucket. You can use versioning to preserve, retrieve, and restore every version of every object stored in your Amazon S3 bucket. With versioning, you can easily recover from both unintended user actions and application failures. Learn more

Bucket Versioning
Enabled

Multi-factor authentication (MFA) delete
An additional layer of security that requires multi-factor authentication for changing Bucket Versioning settings and permanently deleting object versions. To modify MFA delete settings, use the AWS CLI, AWS SDK, or the Amazon S3 REST API. Learn more



middleware-logs-gelium-x

Bucket overview

AWS Region
US East (N. Virginia) us-east-1

Amazon Resource Name (ARN)
arn:aws:s3:::middleware-logs-gelium-x

Creation date
July 16, 2024, 20:39:20 (UTC+05:30)

Bucket Versioning

Versioning is a means of keeping multiple variants of an object in the same bucket. You can use versioning to preserve, retrieve, and restore every version of every object stored in your Amazon S3 bucket. With versioning, you can easily recover from both unintended user actions and application failures. Learn more

Bucket Versioning
Enabled

Multi-factor authentication (MFA) delete
An additional layer of security that requires multi-factor authentication for changing Bucket Versioning settings and permanently deleting object versions. To modify MFA delete settings, use the AWS CLI, AWS SDK, or the Amazon S3 REST API. Learn more