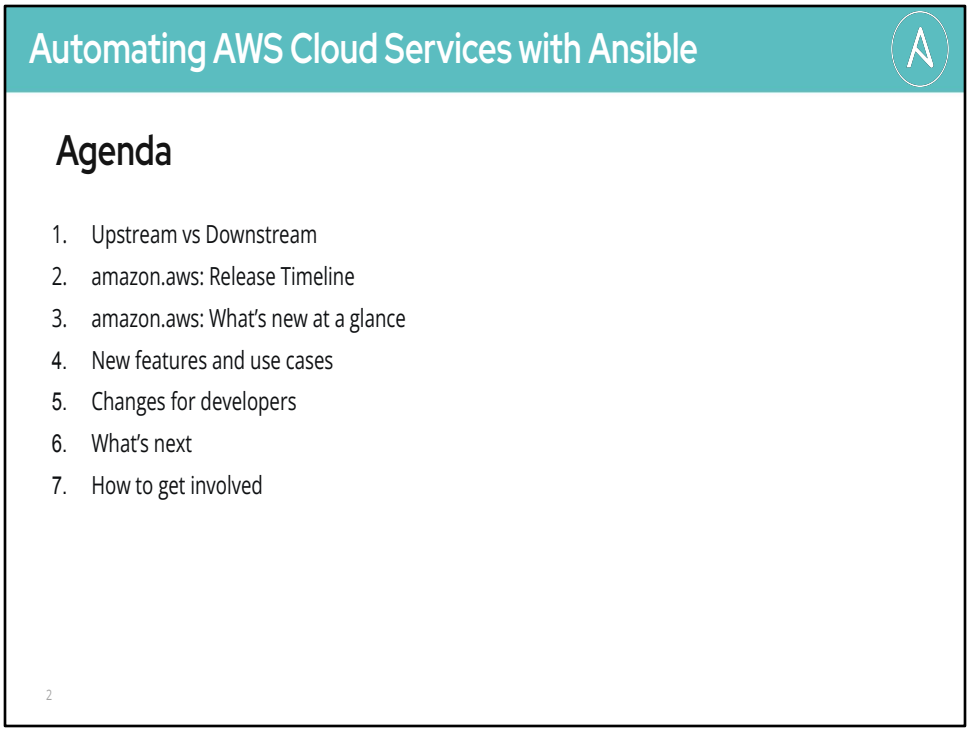
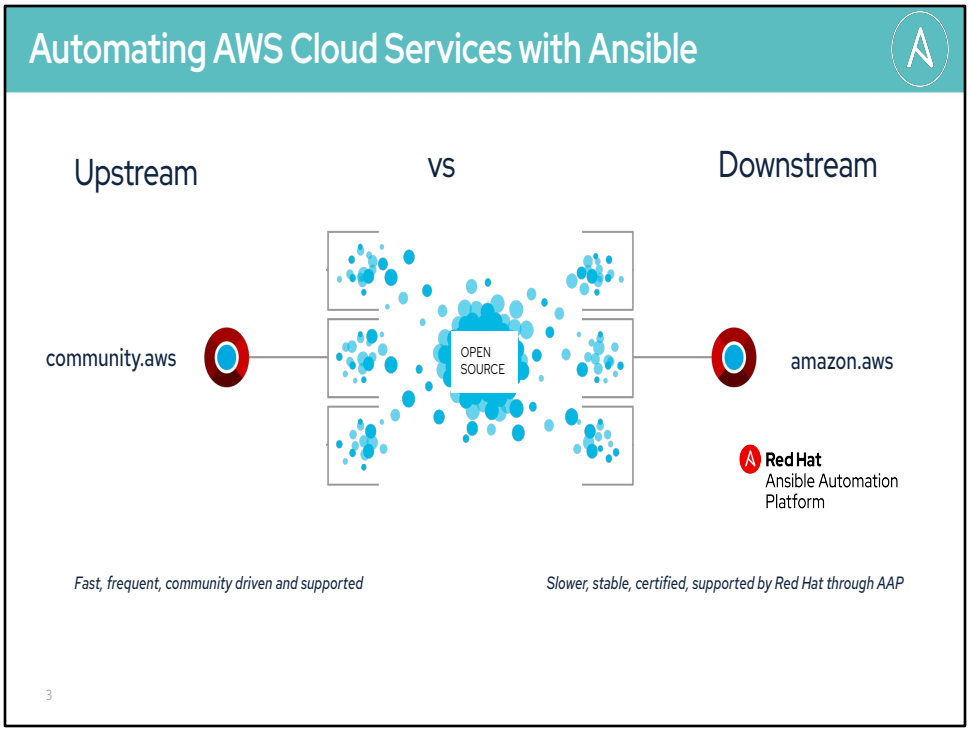
Hi, I'm Alina Buzachis, I hold a PhD in Distributed Systems and I am a Senior Software Engineer in the Ansible Cloud Content Team, Today, I'll guide you through key aspects of working with the Ansible Collection for Amazon Web Services and show how it can enhance your AWS automation experience. Feel free to reach out if you'd like to discuss anything in more detail. You can connect with me on GitHub and Matrix.

Let's dive in and discover how you can unlock the full potential of AWS automation with Ansible!

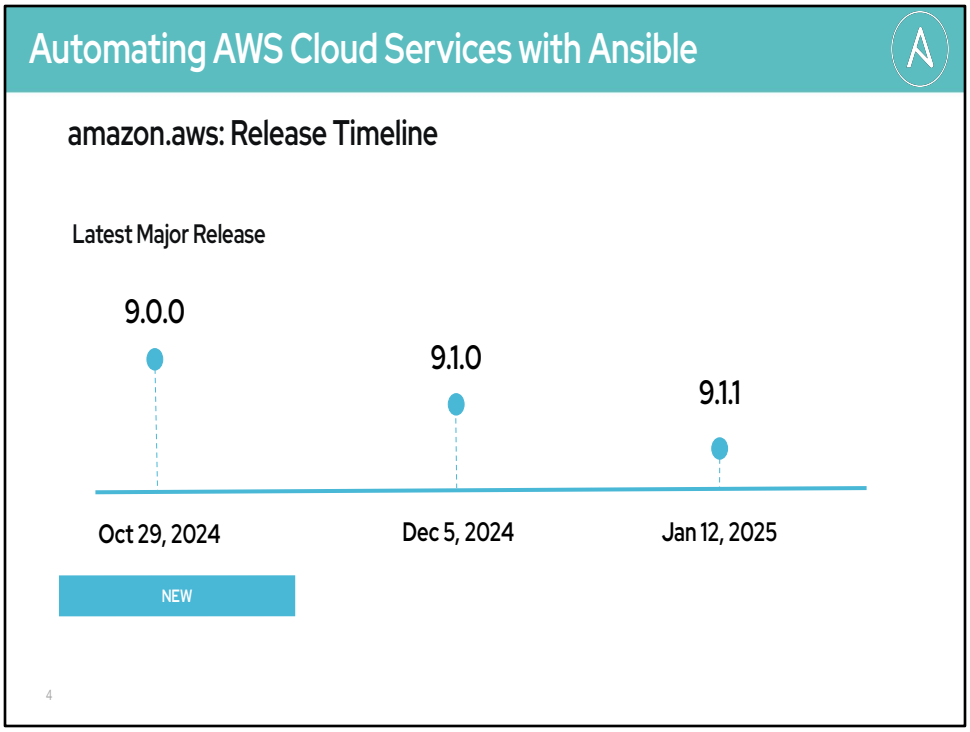


In this talk:

- I'll explain the difference between the upstream and downstream.
- We'll explore the amazon.aws releases timeline with recent updates, and new features.
- I'll show how your AWS workflows can benefit the AWS new features, with practical use cases.
- Then we will shift focus on updates and changes developers or contributors need to know.
- Looking ahead at what's in store for AWS automation with Ansible.
- Opportunities to contribute and engage with the Ansible and AWS communities.



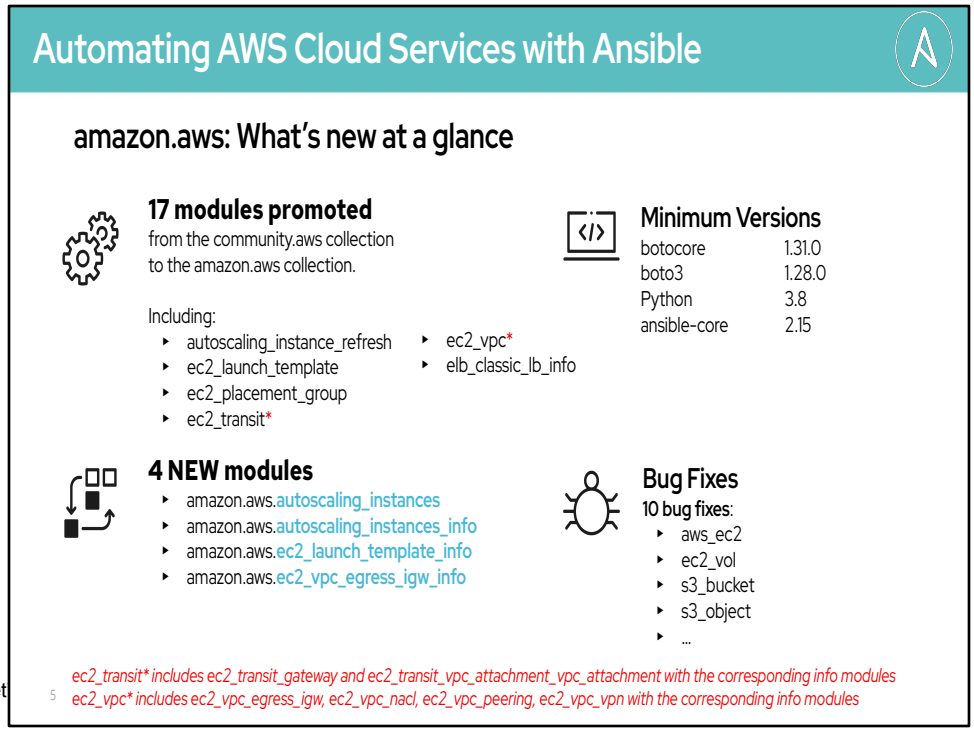
When working with Ansible collections for AWS, it's important to understand the distinction between upstream and downstream.
Upstream (`community.aws`): Community-driven collection focused on delivering the latest AWS features. Ideal for users wanting cutting-edge functionality.
Downstream (`amazon.aws`): A stable, Red Hat certified collection with a deliberate release cycle. Supported by Red Hat through the Ansible Automation Platform (AAP), it's perfect for people and enterprises needing reliable, production-ready solution. Modules in community.aws can be promoted to amazon.aws once they are stable, well-tested, and aligned with customer use cases. This process guarantees that only reliable and robust modules are included in the amazon.aws collection. Being Red Hat Ansible Certified Content, this collection is eligible for support through the Ansible Automation Platform. Both collections are part of the Ansible community package.



The `amazon.aws` collection continues to evolve, delivering new features, updates, and improvements to support AWS automation. Let's take a closer look at the recent release timeline:

- The latest major release, version 9.0.0,on October 29, 2024, marked a major milestone, introducing significant updates and new features.
- Shortly after, version 9.1.0 was released on December 5, 2024, bringing additional improvements and expanding functionality.
- Most recently, version 9.1.1, a maintenance release, went live on January 12, 2025, addressing bug fixes and ensuring greater stability.

Regarding the release cadence, we've adopted a monthly release cycle, with releases scheduled for the first Tuesday of every month. This structured approach ensures a steady cadence of enhancements, bug fixes, and new features while maintaining high standards of quality.
amazon.aws 9,0 and also community.aws 9.0 are available in the latest release of the Ansible Community Package. (11.0).



Let

1. Migrated Modules
    - A total of 17 modules have been promoted from the `community.aws` collection to the `amazon.aws` collection, including critical ones like `autoscaling_instance_refresh`, `ec2_launch_template`, and the `ec2_transit*`and `ec2_vpc*` groups, which now include corresponding info modules for enhanced functionality.
1. New Modules
    - The collection introduces 4 brand-new modules:
        - `amazon.aws.autoscaling_instances` and `amazon.aws.autoscaling_instances_info` for working with auto-scaling configurations.
        - `amazon.aws.ec2_launch_template_info` for managing launch template details.
        - `amazon.aws.ec2_vpc_egress_igw_info` for handling VPC egress internet gateways.
1. Minimum Requirements
    - To ensure compatibility with the latest updates, the minimum versions for dependencies are:
        - botocore: 1.31.0
        - boto3: 1.28.0
        - Python: 3.8
        - ansible-core: 2.15

The amazon.aws Collection has dropped support for `botocore<1.31.0` and `boto3<1.28.0`. Most modules will continue to work with older versions of the AWS software development kit (SDK), however, compatibility with older versions of the AWS SDK is not guaranteed and will not be tested. When using older versions of the AWS SDK, Red Hat Ansible Automation Platform will display a warning. Check out the module documentation for the minimum required version for each module.

The amazon.aws Collection requires python 3.8 at least.

Tested with the Ansible Core >= 2.15.0 versions, and the current development version of Ansible. Ansible Core versions prior to 2.15.0 are not supported.

1. Bug Fixes

- These releases also addresses 10 bugs, improving stability for modules like `aws_ec2`, `ec2_vol`, `s3_bucket`, `s3_object`, and more.

Now, let's dive into some of the exciting new features that have been added to the amazon.aws collection and discover some practical use cases.

New Features

ANSIBLE

New Features

- **Module:** amazon.aws.ec2_vpc_route_table
- **New Feature:** transit_gateway_id in routes.
- **Benefits:**
  - Simplifies inter-VPC communication.
  - Centralized connectivity management.
- **Use Case:** Centralized Network Routing with AWS Transit Gateway

```
- name: Add a route to public route table
  amazon.aws.ec2_vpc_route_table:
    vpc_id: "{{ vpc.vpc.id }}"
    tags:
      Public: "true"
      Name: Public route table
    routes:
      - dest: "0.0.0.0/0"
        gateway_id: igw
      - dest: ::/0
        gateway_id: igw
      - dest: "10.0.0.0/16"
        transit_gateway_id: "{{
transit_gateway.transit_gateway.transit_gateway_
id }}"
    register: add_routes
```


One of the new features introduced in the `amazon.aws.ec2_vpc_route_table` module is the ability to define `transit_gateway_id` directly in the `routes` option. This feature is particularly useful for setting up centralized routing through an AWS Transit Gateway, making it easier to manage and scale your VPC network architecture. It ensures efficient routing across multiple VPCs, improving both scalability and maintainability.

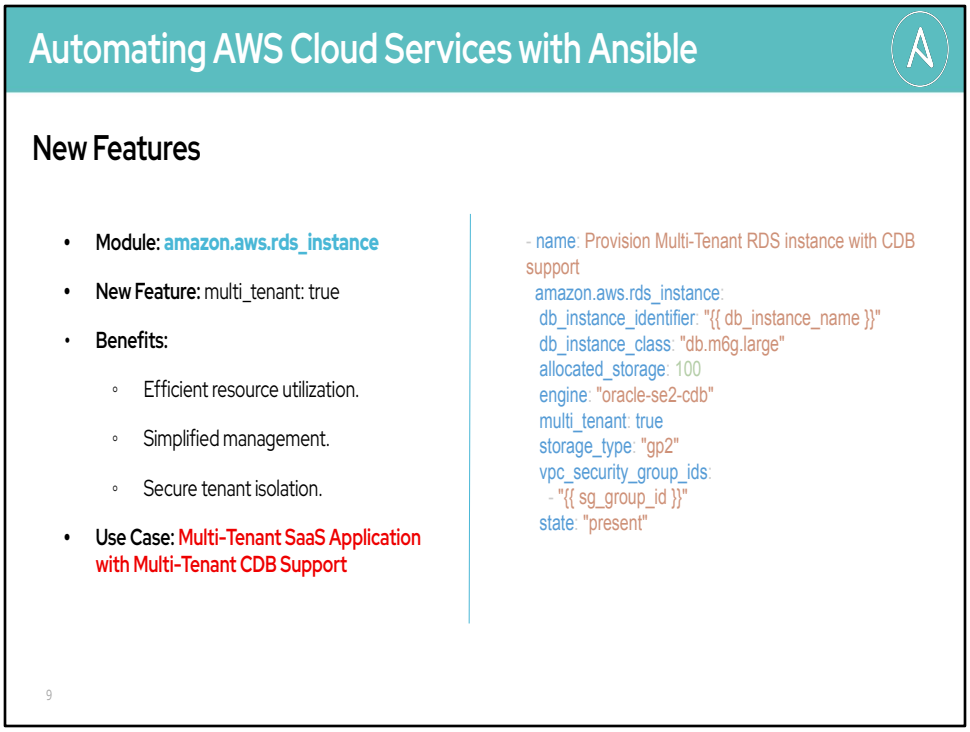
Automating AWS Cloud Services with Ansible

New Features

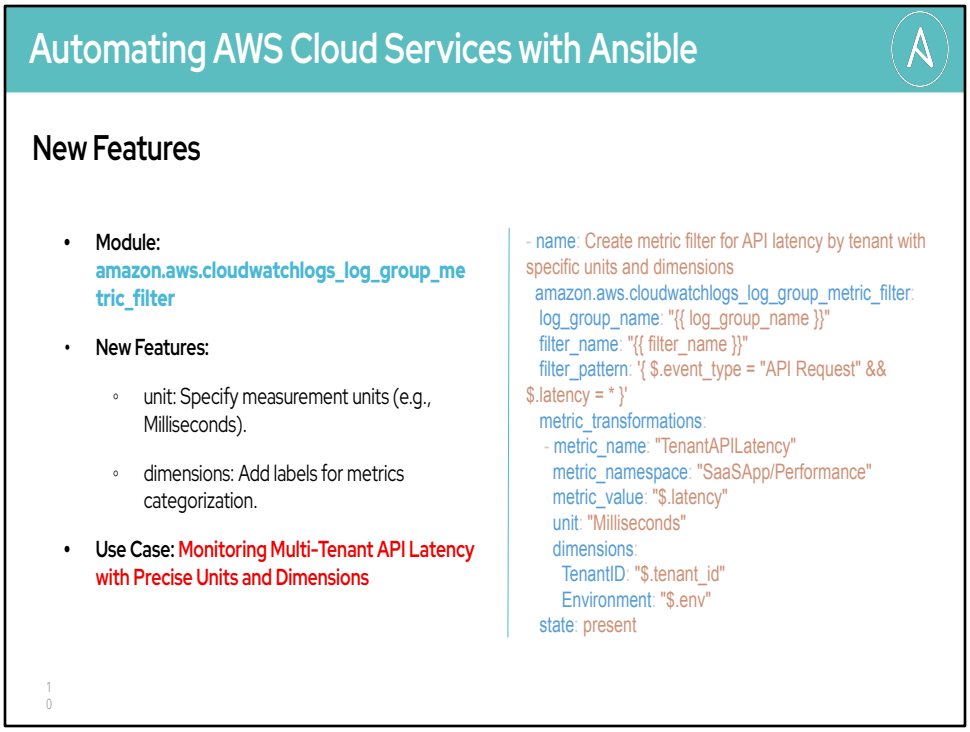
- **Module:** amazon.aws.ec2_instance
- **New Capability:** Automate resizing operations.
  - Stop, modify type, and restart in one step.
- **Benefits:**
  - Minimized downtime.
  - Streamlined scaling for performance and cost optimization.
- **Use Case:** Managing EC2 Instance Types for Cost Optimization and Performance Scaling

```
- name: Upgrade EC2 instance type for peak usage
  amazon.aws.ec2_instance:
    name: "{{ ec2_instance_name }}"
    image_id: "{{ ec2_ami_id }}"
    instance_type: "t3.large"    # Upgrade instance type to a
larger size
    state: "present"
    subnet_id: "{{ subnet.id }}"
    wait: true
```


The `amazon.aws.ec2_instance` module now simplifies the management of EC2 instances, enabling on-the-fly scaling without manual intervention (no need to manually stop, modify the instance type, and restart). This new capability is ideal for situations where you need to scale EC2 instances up or down for peak usage times or to adjust for cost optimization. For example, during periods of high demand, you can automatically scale up the instance type to ensure your application has the necessary resources, and scale down when demand decreases, reducing costs.

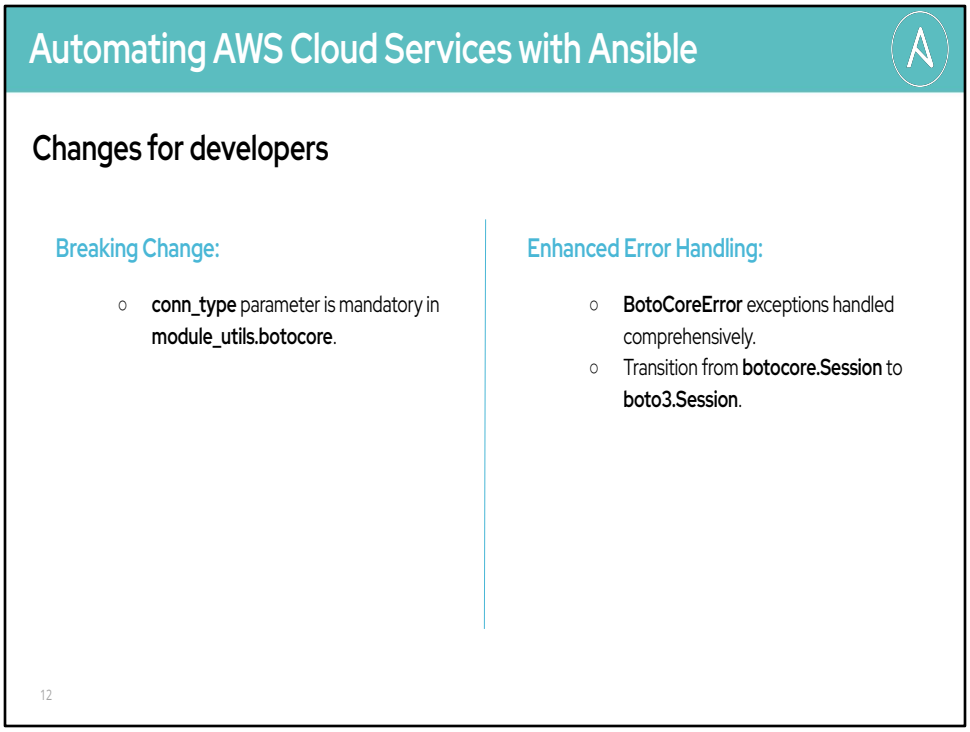


The `amazon.aws.rds_instance` module now supports the `multi_tenant: true` option, enabling you to provision multitenant container databases (CDBs). This feature is particularly beneficial for Software-as-a-Service (SaaS) application that serves multiple clients, each with different data storage needs. The architecture uses a multitenant database design in Amazon RDS where multiple tenants share the same CDB, but their data is stored in separate pluggable databases (PDBs) for isolation and security.

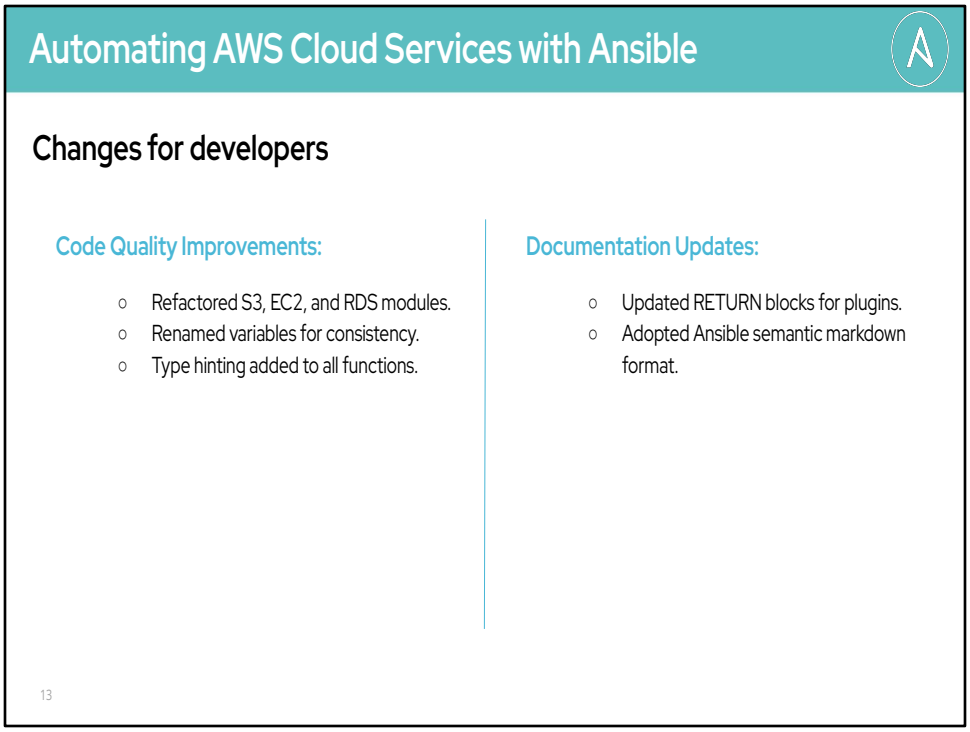


The `amazon.aws.cloudwatchlogs_log_group_metric_filter` module now introduces two new features: the ability to specify units and dimensions for metrics. These enhancements allow for more precise monitoring and categorization of your logs, making it easier to track and analyze API performance at a granular level. You want to track API request latency precisely in milliseconds and categorize the metrics by TenantID and Environment (production or staging) for deeper insights and quicker troubleshooting.

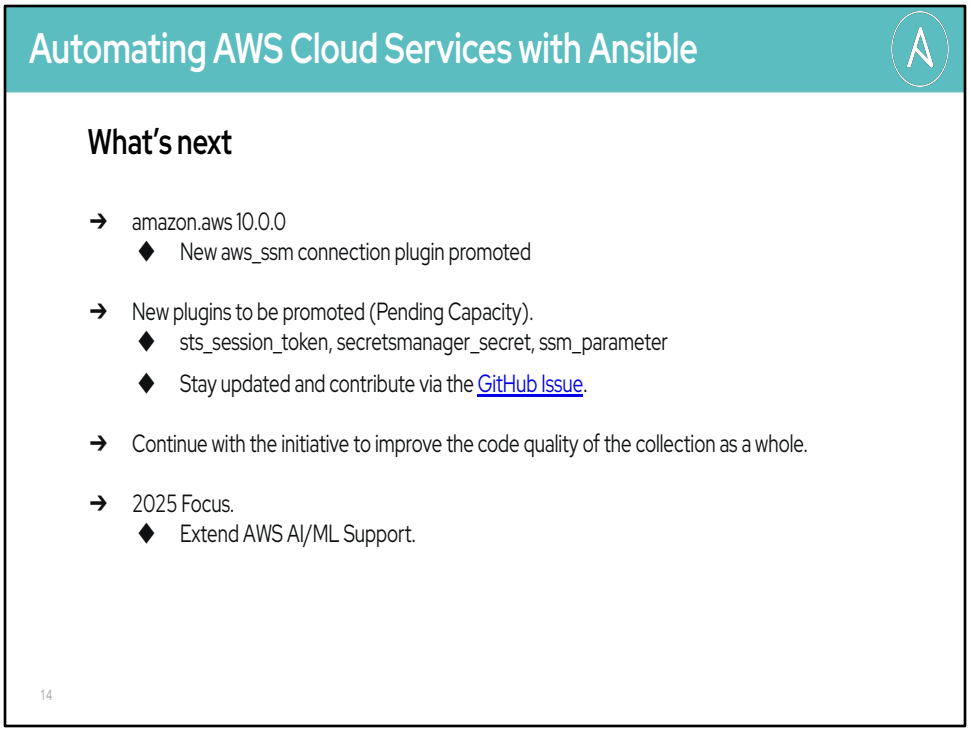
Let’s now shift focus to some important updates for developers. If you are an active contributor to the amazon.aws collection or are interested in becoming one, there are a couple of key changes you should be aware of.



If you're an active contributor or planning to contribute to the `Ansible Collection for AWS`, there are two key updates to be aware of.
First, there has been breaking change for module_utils.botocore that makes the `conn_type` parameter to be mandatory, ensuring more reliable connection configuration.
Second, improvements have been made to error handling by catching the `BotoCoreError` exception more comprehensively. Additionally, the transition from `botocore.Session` to `boto3.Session` standardizes session management for better consistency and stability across the codebase. These changes impact the community.aws collection also because amazon.aws and community.aws share some base module utils.



In addition, the effort to enhance the `amazon.aws` collection, focusing on refactoring key modules like S3, EC2, and RDS for improved readability, maintainability, and performance. It includes variable renaming for consistency and the addition of type hinting to strengthen code clarity.
Although this update covers major modules, refactoring for others will continue in future releases. Documentation has also been improved, with updated RETURN blocks for accuracy and the adoption of the Ansible semantic markdown format, enhancing readability and usability.



amazon.aws 10.0.0:

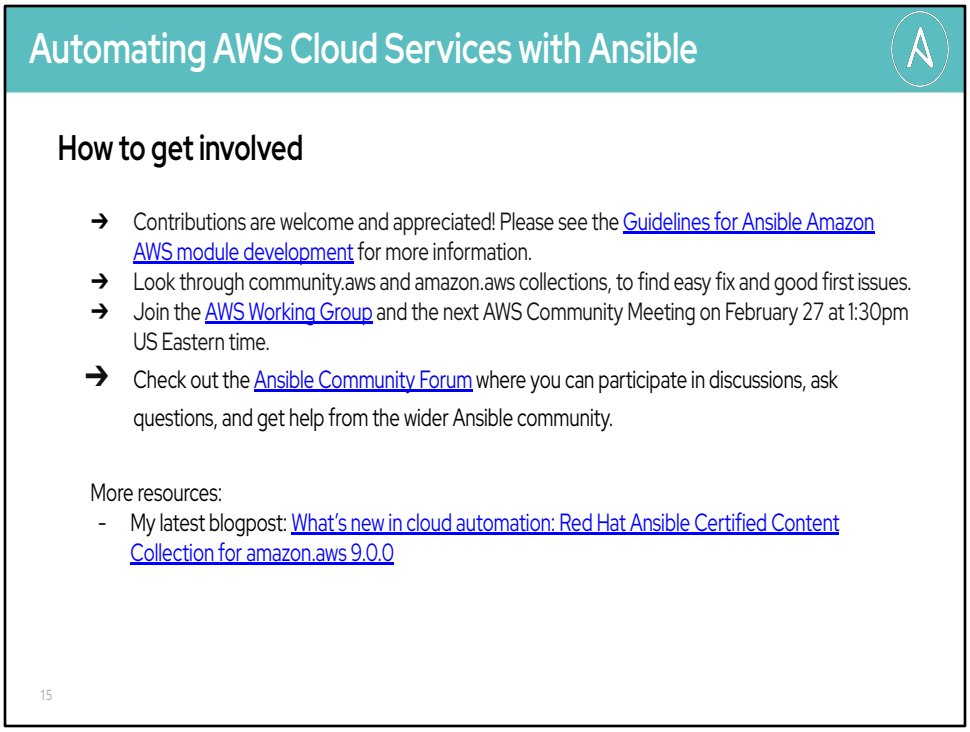
- aws_ssm Plugin Promotion
  The aws_ssm plugin is being considered for promotion in version 10.0.0. This will enhance AWS Systems Manager integration, improving automation for managing EC2 instances and other AWS resources.

New Module Promotions (Pending Capacity):

- Several new modules are under consideration for promotion, contingent on available capacity. Stay updated and contribute via the GitHub Issue.

2025 Focus: AWS AI/ML Support

- In 2025, we'll be exploring enhanced support for AWS AI/ML services. This investigation will pave the way for automation tools that integrate with AWS's growing AI/ML offerings, meeting the needs of next-generation workloads.



Contributions to the `Ansible` collections for AWS are always welcome and highly appreciated! If you're interested in contributing, we encourage you to follow the Guidelines for Ansible Amazon AWS Module Development to get started. These guidelines will help you understand best practices and how to contribute effectively.

To ease your journey into contribution, take a look at both the community.aws and amazon.aws collections for easy fixes and good first issues—these are great opportunities for newcomers to make their first contribution. Additionally, you can join the AWS Working Group and participate in the upcoming AWS Community Meeting on February 27 at 1:30 pm US Eastern time. This is a great chance to connect with other community members, ask questions, and share your ideas. You can also join the join the Ansible Community Forum where you can participate in discussions, ask questions, and get help from the wider Ansible community.

For more resources, check out my latest blog post: "What's New in Cloud Automation: Red Hat Ansible Certified Content Collection for amazon.aws 9.0.0." It offers insights into the new features and improvements, helping you stay up-to-date with the latest in cloud automation.

Thank you all for joining today's session! I hope you gained valuable insights into the Ansible collections for AWS and the exciting developments around automating AWS cloud services with Ansible.

Thank you again, and I look forward to connecting with you all!