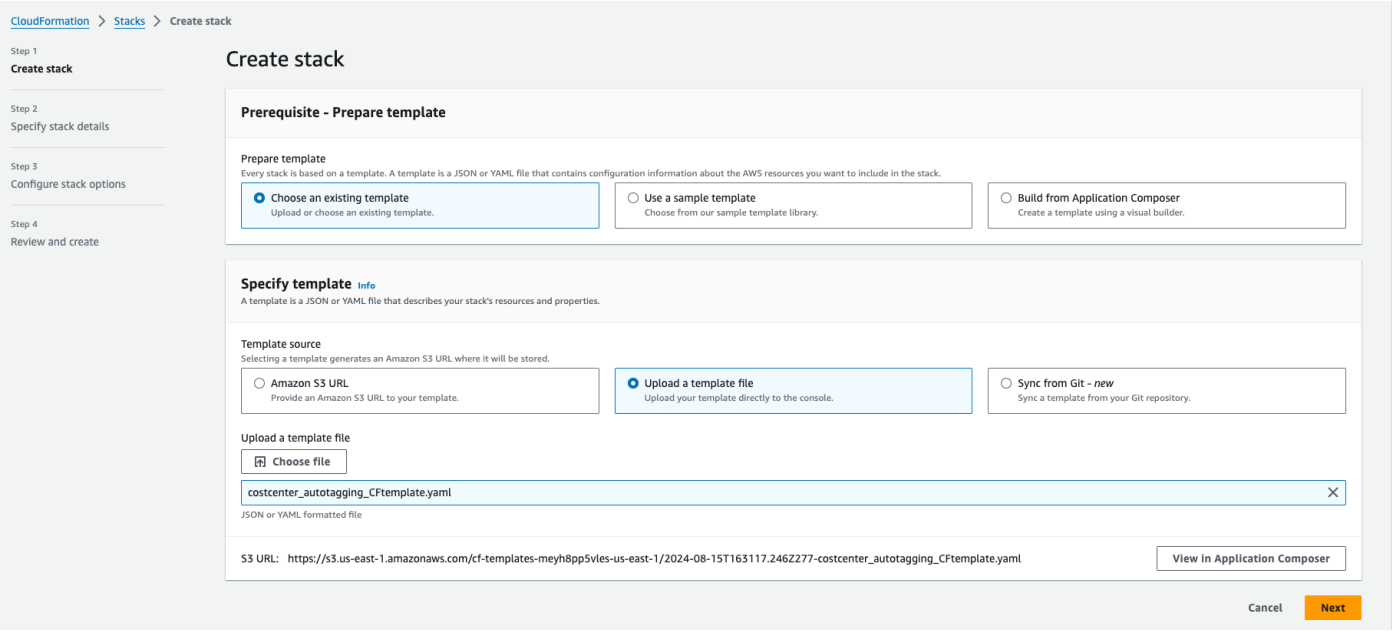
AWS CostCenter Autotagging

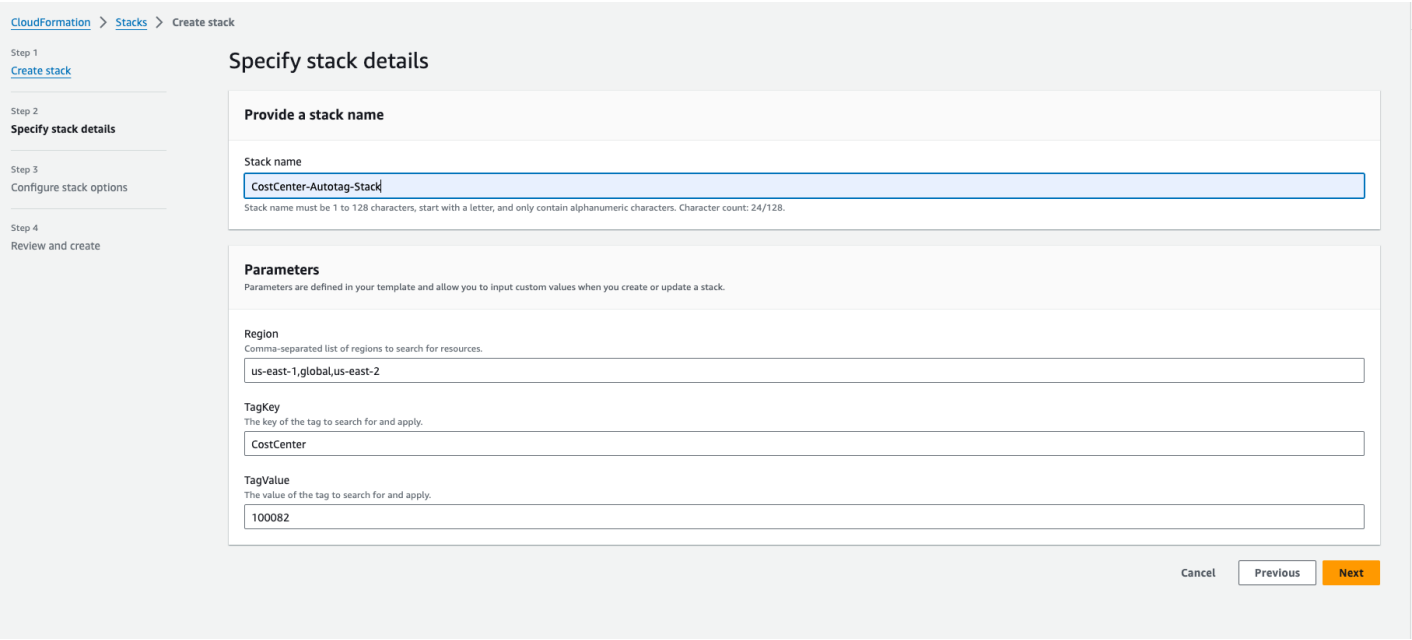
Setup Instructions

Follow the below steps to configure CostCenter autotagging in AWS accounts

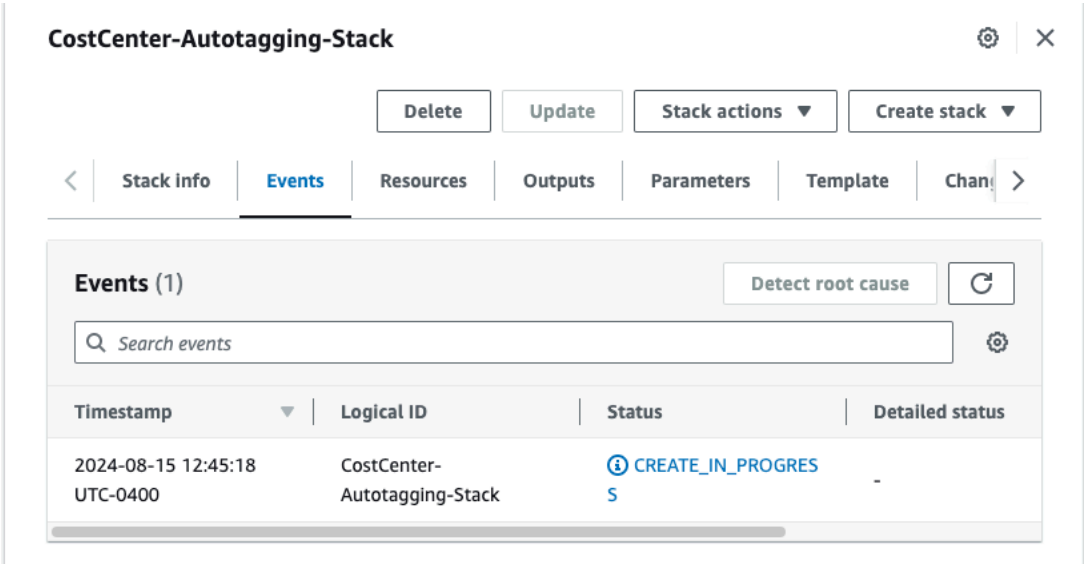
1) Load the Cloud Formation Template : `costcenter_autotagging-CFtemplate.yaml` to build respective resources and click next



2) Enter the Stack name and the Parameter values.
   Parameters:
   - **a)** Region : List of comma separated account specific regions that you want to cover. (example: us-east-1,global,us-east2) Note: No Spaces
   - **b)** TagKey : CostCenter
   - **c)** TagValue : value of the account specific cost center, (Default:100082)



3) Keep all the remaining values and options as default and build the stack.



Wait for the creation to be successful. Now the function is ready and will tag all the resources in the specified regions with the costcenter tag once every week.

CostCenter Autotagging Lambda Function Overview:

This program is an AWS Lambda function designed to automatically tag AWS resources with a specified CostCenter tag. It searches for resources without the designated tag across specified AWS regions and applies the tag to those resources.

Overview

The script performs the following main functions:

1. Searches for AWS resources without a specified CostCenter tag in given regions.
2. Tags the found resources with the specified CostCenter tag.
3. Logs the process and provides a summary of the tagging operation.

Key Components

1. **Resource Search :** `search_resources_with_tag` Function:

```python
def search_resources_with_tag(region, tag_key, tag_value):
    client = boto3.client('resource-explorer-2', region_name='us-east-1')

    query_string = f'-tag:{tag_key}={tag_value} -service:ssm'

    paginator = client.get_paginator('search')
    response_iterator = paginator.paginate(
        QueryString=f'region:{region} {query_string}',
        PaginationConfig={
            'PageSize': 50
        }
    )
    resources = []
    for page in response_iterator:
        for resource in page['Resources']:
            resources.append(resource['Arn'])
    total_count = len(resources)
    logger.info(f"Total Resources in {region} : {total_count}")
    return resources
```

This function searches for AWS resources without a specific tag:

- It creates a client for AWS Resource Explorer 2, always using 'us-east-1' as the region.
- It constructs a query string to find resources without the specified tag and excludes SSM resources.
- It uses a paginator to handle large numbers of results efficiently.
- It iterates through the results, collecting the ARNs of matching resources.
- It logs the total number of resources found and returns the list of resource ARNs

2. Resource Tagging : `tag_resources` Function

```
def tag_resources(region, resource_arns, tag_key, tag_value):
    if region == 'global':
        region = 'us-east-1'

    client = boto3.client('resourcegroupstaggingapi', region_name=region)

    success_count = 0
    error_count = 0

    for arn in resource_arns:
        try:
            client.tag_resources(
                ResourceARNList=[arn],
                Tags={tag_key: tag_value}
            )
            success_count += 1
        except ClientError as e:
            error_count += 1
            logger.error(f"Error tagging resource {arn} in region {region}: {str(e)}")

    return success_count, error_count
```

This function applies tags to the specified resources:

- It handles the 'global' region by using 'us-east-1'.
- It creates a client for the AWS Resource Groups Tagging API in the specified region.
- It iterates through each resource ARN, attempting to apply the tag.
- It keeps track of successful tags and errors.
- It logs any errors encountered during tagging.
- It returns the counts of successful tags and errors.

3. Lambda handler : `lambda_handler` Function

```
def lambda_handler(event, context):
    tag_key = os.getenv('TAG_KEY', 'CostCenter')
    tag_value = os.getenv('TAG_VALUE')
    regions = os.getenv('REGIONS', 'us-east-1,global').split(',')
    total_resources_found = 0
    total_successfully_tagged = 0
    total_errors = 0
    for region in regions:
        logger.info(f"Processing region: {region}")
        resources = search_resources_with_tag(region, tag_key, tag_value)
        resource_count = len(resources)
```

```
        total_resources_found += resource_count
        if resources:
            success_count, error_count = tag_resources(region, resources, tag_key, tag_value)
            total_successfully_tagged += success_count
            total_errors += error_count
        else:
            logger.info(f"No resources found without the specified tag in region {region}.")

    return {
        'TotalResourcesFound': total_resources_found,
        'TotalSuccessfullyTagged': total_successfully_tagged,
        'TotalErrors': total_errors
    }
```

This is the main function that AWS Lambda executes:

- It retrieves configuration from environment variables:
  - `TAG_KEY`: The key for the tag (default is 'CostCenter')
  - `TAG_VALUE`: The value for the tag
  - `REGIONS`: A comma-separated list of regions to process
- It initializes counters for tracking overall progress.
- For each region:
  - It calls `search_resources_with_tag` to find resources without the tag.
  - If resources are found, it calls `tag_resources` to apply the tag.
  - It updates the overall counters with the results.
- If no resources are found in a region, it logs this information.
- Finally, it returns a dictionary with the overall results:
  - Total resources found
  - Total resources successfully tagged
  - Total errors encountered

This function orchestrates the entire process of searching for resources and applying tags across multiple AWS regions.

4. Configuration

- Uses environment variables for flexible configuration:
  - `TAG_KEY`: The tag key to use (default: "CostCenter")
  - `TAG_VALUE`: The value to set for the tag
  - `REGIONS`: Comma-separated list of regions to process (default: "us-east-1,global")

Execution Flow

1. The Lambda function is triggered.
2. It retrieves configuration from environment variables.
3. For each specified region:
   - Searches for resources without the specified tag.
   - Attempts to apply the tag to found resources.
   - Logs the results of the tagging operation.
4. Returns a summary of the entire operation, including:
   - Total resources found
   - Total resources successfully tagged
   - Total errors encountered