ITIL Change Management for AWS

Table of Contents

01

Executive summary

There is considerable discussion on implementing effective change management in the cloud. Some opinions go so far as declaring ITIL change management incompatible with DevOps. One thing is certain, AWS has changed developers' and infrastructure teams' expectations of what is possible and when.

Of course, infrastructure teams are still responsible for maintaining reliability, but this gets challenging when there are 100-times more changes than with physical infrastructure. With multiple developers launching resources on AWS using different workflows, things can spiral out of control quickly. It's hard to gauge the impact of each change, and there are typically wasted resources and time cleaning up unused resources.

Lean thinking calls for eliminating Muda (the Japanese word for waste). Waste is anything that doesn't add value to the customer. Companies move to the cloud so they can utilize its dynamic nature, launching resources, as needed. But this practice hasn't worked on AWS, as expected. Teams launch resources on AWS like they are free. We see teams launch resources like mytest1 and mytest2, and suddenly, their account is full of test instances. A year later, everyone is afraid to stop these instances because no one knows what might break. Normally, when resources get out of hand, a company puts together a SWAT team to eliminate waste on the cloud. This leads to the company wasting more time trying to eliminate waste. If that's the position your organization is in, you need a change management process.

Security is another major concern on AWS. Cyberattacks are on the rise, and every company is a target. The number of security incidents increased 40% in 2016 (Bloomberg), with the cost of a data breach costing companies an average of $4 million per incident (IBM). Automation on AWS can help you respond quickly to attacks, but automation could be a pain. When things don't work, people usually log in and make "temporary" changes for debugging, and then forget to revert these changes. You need a system to monitor all of your infrastructure changes to ensure they are in compliance with the configurations specified in your internal guidelines. In addition to going through the code approval process for infrastructure changes, like git pull requests for all your infrastructure, you need to make sure actual resources provisioned through automation are in compliance, and they remain in compliance after provisioning.

This is where change management comes in.

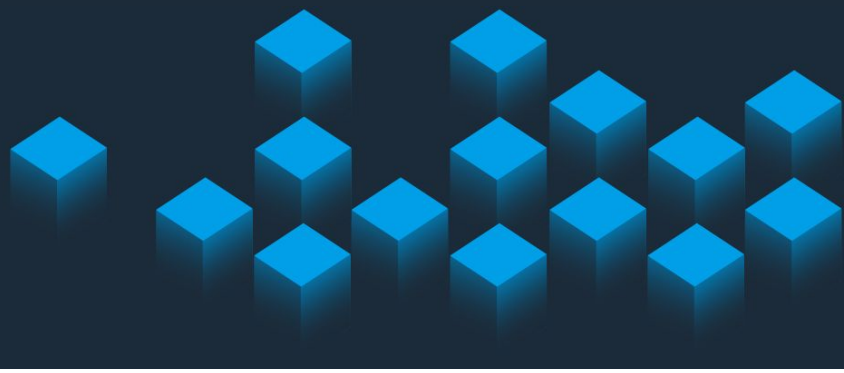
02

Change management

An essential objective of the change management process is to ensure all changes made by IT cause minimum disruption and meet internal guidelines. As noted, when working with AWS, there are often multiple engineers making changes, so it's hard to track who is doing what. With change management, all AWS changes are reviewed and approved, simultaneously helping reduce waste and bringing visibility to changes for improved security and reliability. It can save hours for auditors, assessors, or anyone else working in your value stream.

But to have effective change management in a dynamic environment like AWS, there are essential components you need, like auto-discovery change management database (CMDB), an infrastructure delta, and standard and manual change filters.

Let's talk about what these components look like on AWS.



CMDB

The change management database (CMDB) needs to auto update on AWS, registering and storing information that other services need to interact with (e.g., IP address, port numbers, URIs). This solves the manual nature of the ITIL CMDB and is absolutely necessary when services are comprised of hundreds (or thousands, or even millions) of nodes, each with dynamically assigned IP addresses.

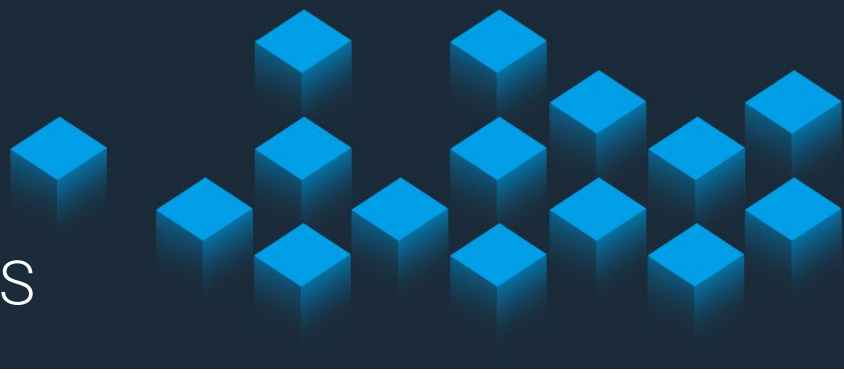
CMDB provides config history for each configuration item. This is needed since cloud changes are so rapid, and resources come up and go down all the time. On AWS, there are also recurring events, such as resources running out of limits and scheduled retirements from AWS. CMDB should always be monitoring for these events. Let's talk about these items in bit more detail.

Time-based changes

In the cloud, some changes might become noncompliant after a specified duration, regardless of whether they were regular changes originally. For example, your engineers might generate an access key to launch various resources; they need that key for day-to-day operation. However, if they never use the key again and forget to delete it, this suddenly becomes a security issue. In another example, your team may launch resources for a specified duration, but if they forget to shut the resources down, the resources will end up getting wasted. CMDB needs to regularly update config items to make sure they are compliant with company policy, and to increase security and eliminate waste.

Configuration update and dependency

Although your team may have a good practice in place for reviewing and testing your infrastructure code before you apply it, you should review changes and dependencies after resources have been provisioned. CMDB should provide context for all configuration updates and dependencies, so you can ensure the launched resources are compliant and meet the original intent.



Types of changes on AWS

Standard changes

Standard changes are low-risk changes. For example, automated deployment, autoscaling, and ephemeral infrastructure for testing are all changes that can be pre-approved. However, you should log the changes for traceability. To ensure these repeated changes don't have any effect on infrastructure reliability or security, you need to have a config history of similar changes.

Normal changes

In traditional environments, these are higher-risk changes that require review or approval from the agreed upon change authority. In many organizations, this responsibility is inappropriately placed on the change advisory board (CAB) or emergency change advisory board (ECAB), which may lack the required expertise to understand the full impact of the change.

Therefore, context is extremely important. Without the context, the CAB will spend hours trying to figure out the impact of the changes, or might approve the changes without knowing the impact.

03

Infrastructure delta

Most companies leverage auto scaling heavily, where new resources can be launched based on specified metrics. However, this behavior makes approving changes difficult. If new resources are appearing and disappearing, how do you track what truly changed in the infrastructure? To gain an accurate picture, we need an infrastructure delta, which shows resource updates for the still-running resources for a defined period.

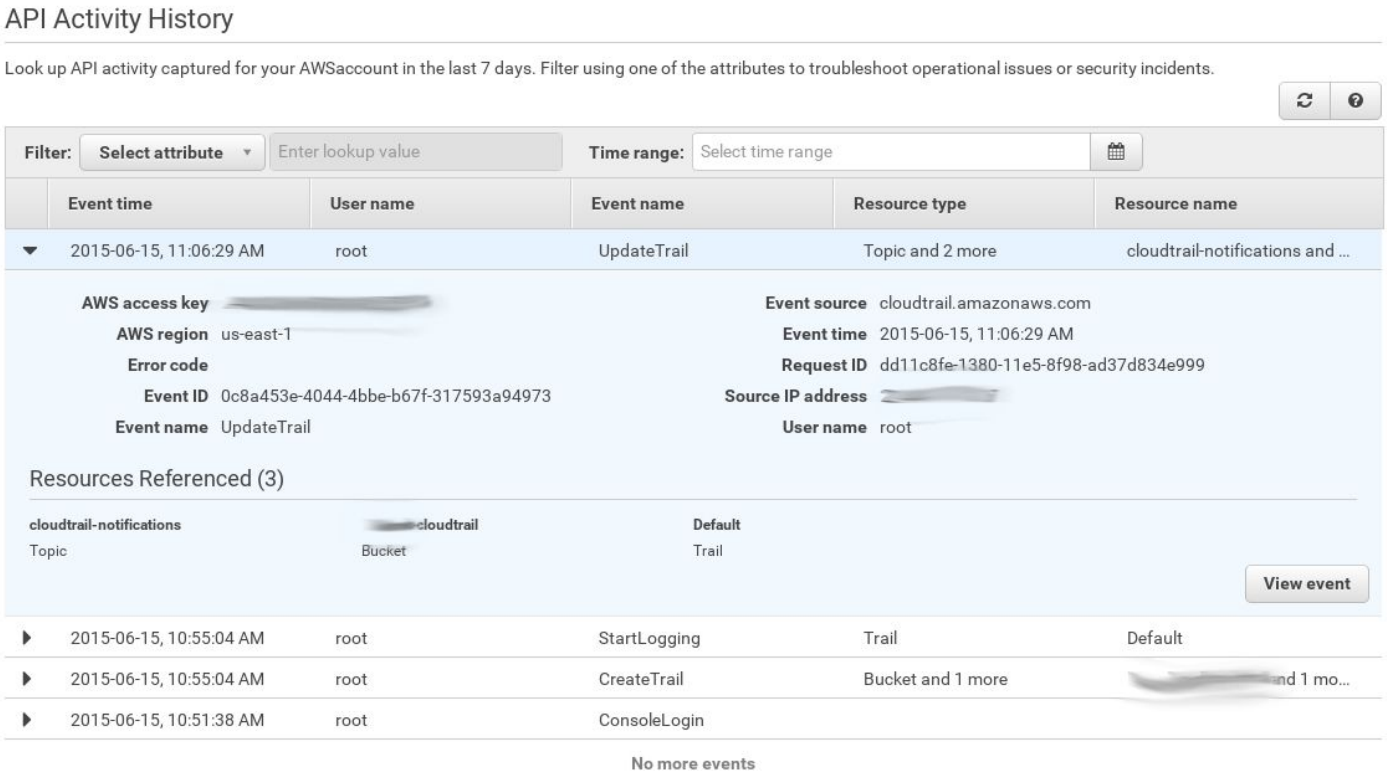
04

Tools for change management

There are several AWS tools that help you gain better visibility into all changes made to your infrastructure. These tools not only give you the ability to keep track of all your AWS resource configurations, but they also help you gain insights into the business impact of infrastructure changes and reduce the risk of experiencing service disruptions.

CloudTrail

AWS CloudTrail is an indispensable tool for your AWS account. It allows you to manage governance, compliance, operational auditing, and risk auditing, and also gives you a history of AWS API calls made on your account so you can quickly track all your infrastructure changes. Although CloudTrail is a regional service, you can enable it across your entire cloud environment with one click.

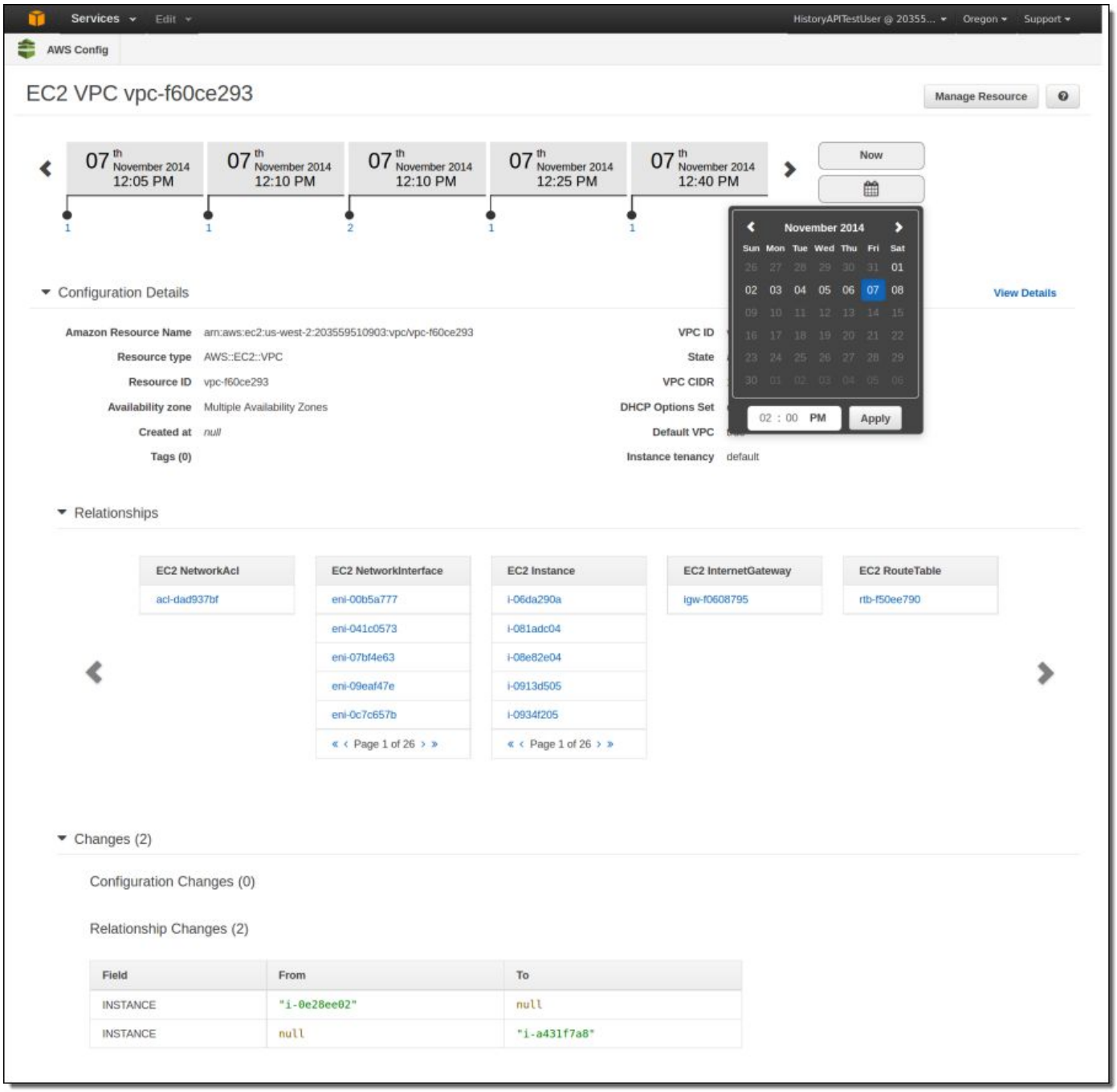


AWS CloudTrail console in AWS.

AWS Config

AWS Config allows you to assess, audit, and evaluate the configurations of your AWS resources. The service continuously monitors and records your AWS resource configurations and enables you to evaluate your change management process against desired configurations.

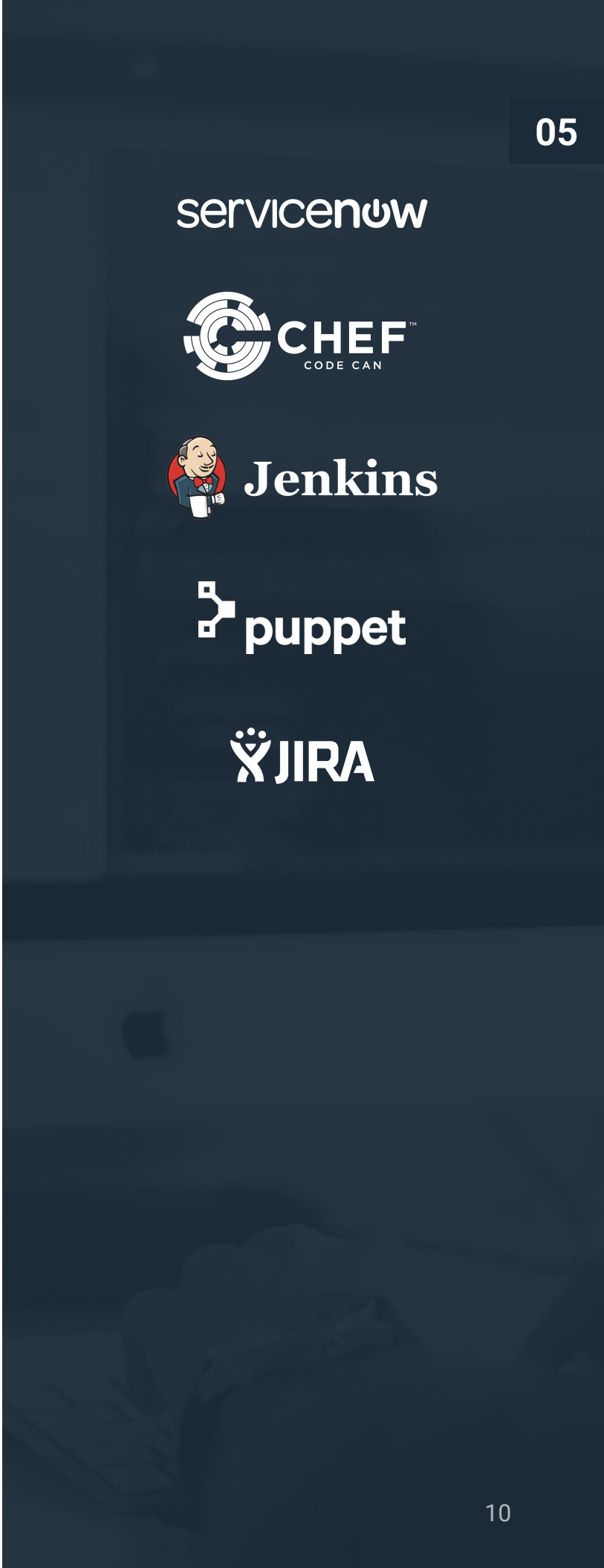
Unlike AWS CloudTrail, AWS Config requires you to enable it across multiple regions and accounts using tools like Terraform. You can also write custom Lambda rules based on your needs.



AWS Config showing config history.

Automation

You should record and visualize standard changes in your infrastructure using your change management systems such as Remedy or ServiceNow. You should also use deployment pipeline tools such as Chef, Jenkins and Puppet to perform deployments and record results automatically. Doing so allows everyone in your organization to have visibility into every change that happens in your company.



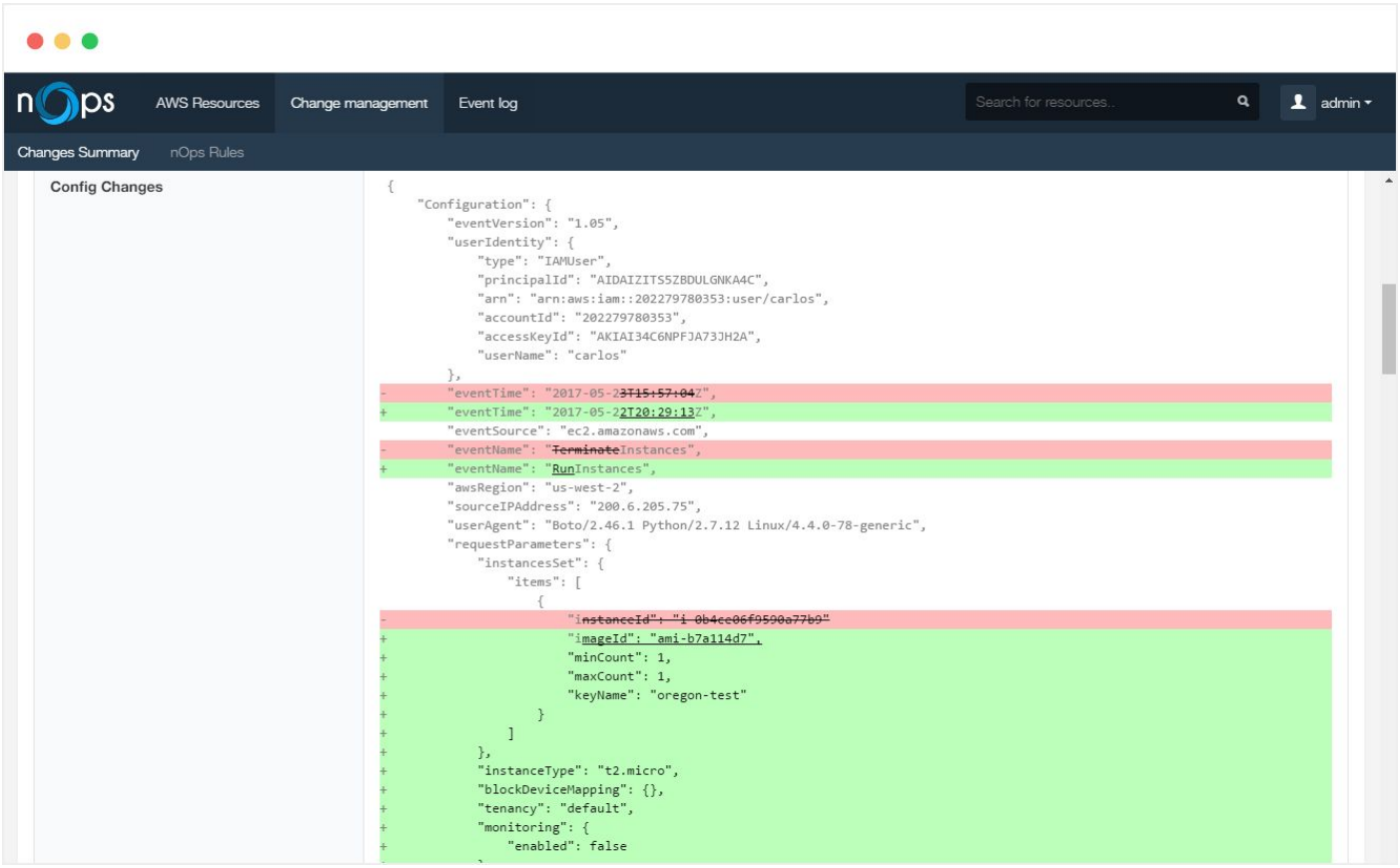
nOps.io

We built nOps with the objective of automating change management processes for cloud computing environments.

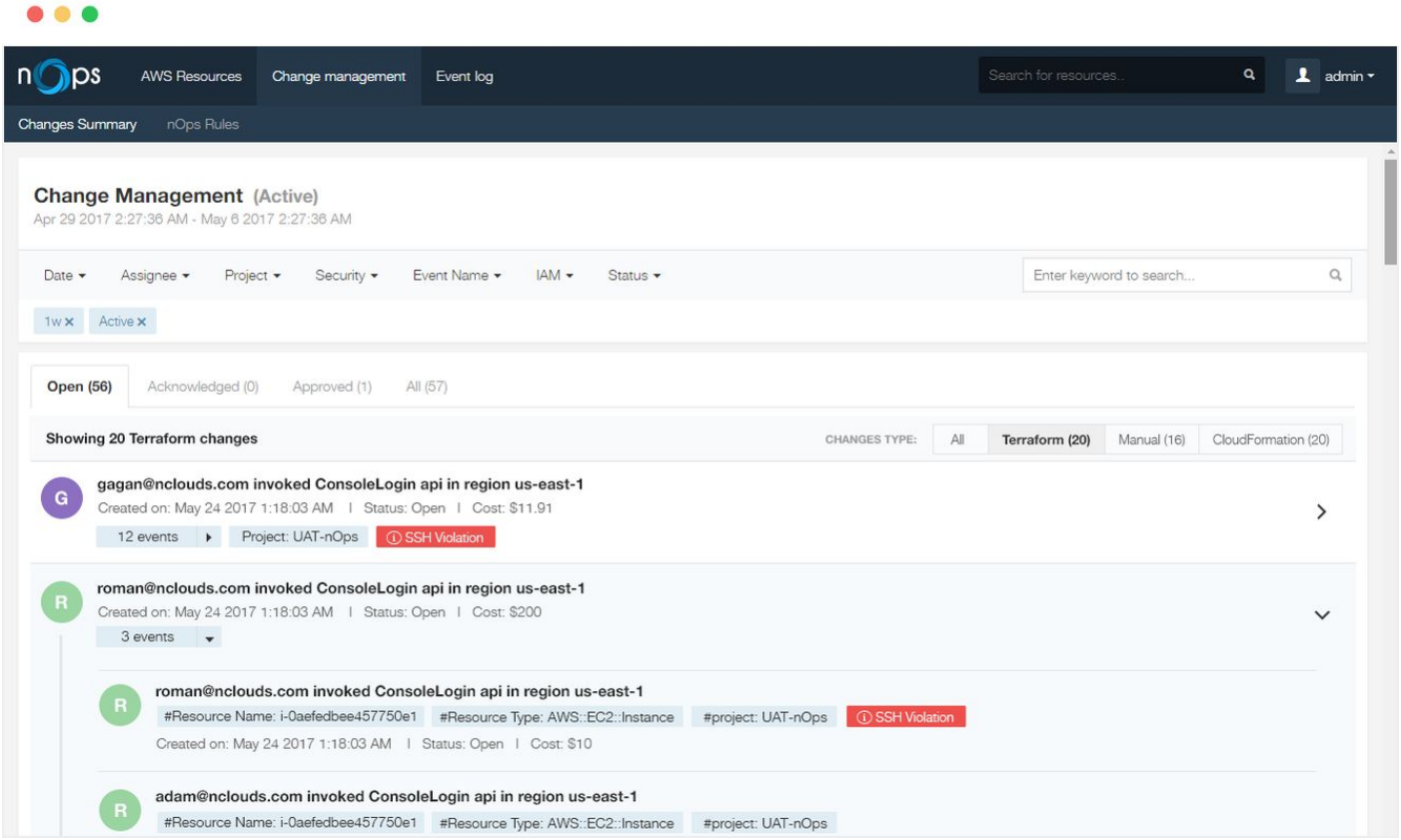
Its change management features and powerful CMDB allow you to review all newly introduced resources quickly. It gives you security and billing context, saving engineers hours looking up this information manually, and saving thousands of dollars by eliminating undocumented, unused resources.

nOps is a proven and trusted tool that provides real-time security monitoring for your AWS infrastructure. nOps can be an essential tool for your organization, helping you implement effective and efficient change management on AWS.

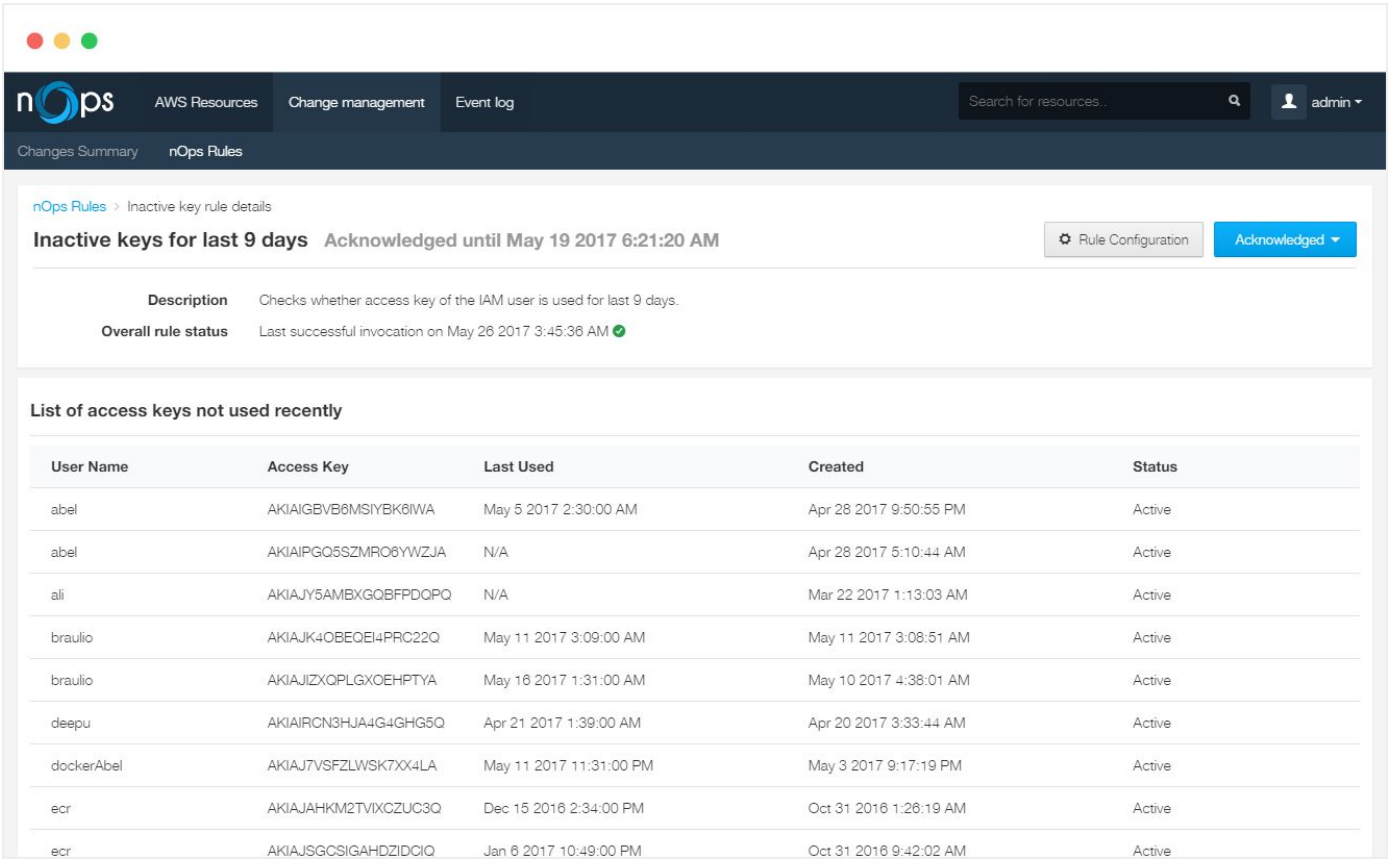
Request a demo at www.nops.io



Showing diffs from latest changes.



Filter manual changes from Terraform and CloudFormation.



Use nOps to monitor inactive keys and receive notifications.

We can help you assess and implement change management for AWS.

Let us share with you what we've learned from hundreds of managed services and cloud infrastructure implementations.

Request a demo at www.nops.io

Contact Info

866-673-9330

sales@nops.io

99 S Almaden Blvd Suite 600,
San Jose, CA 95113