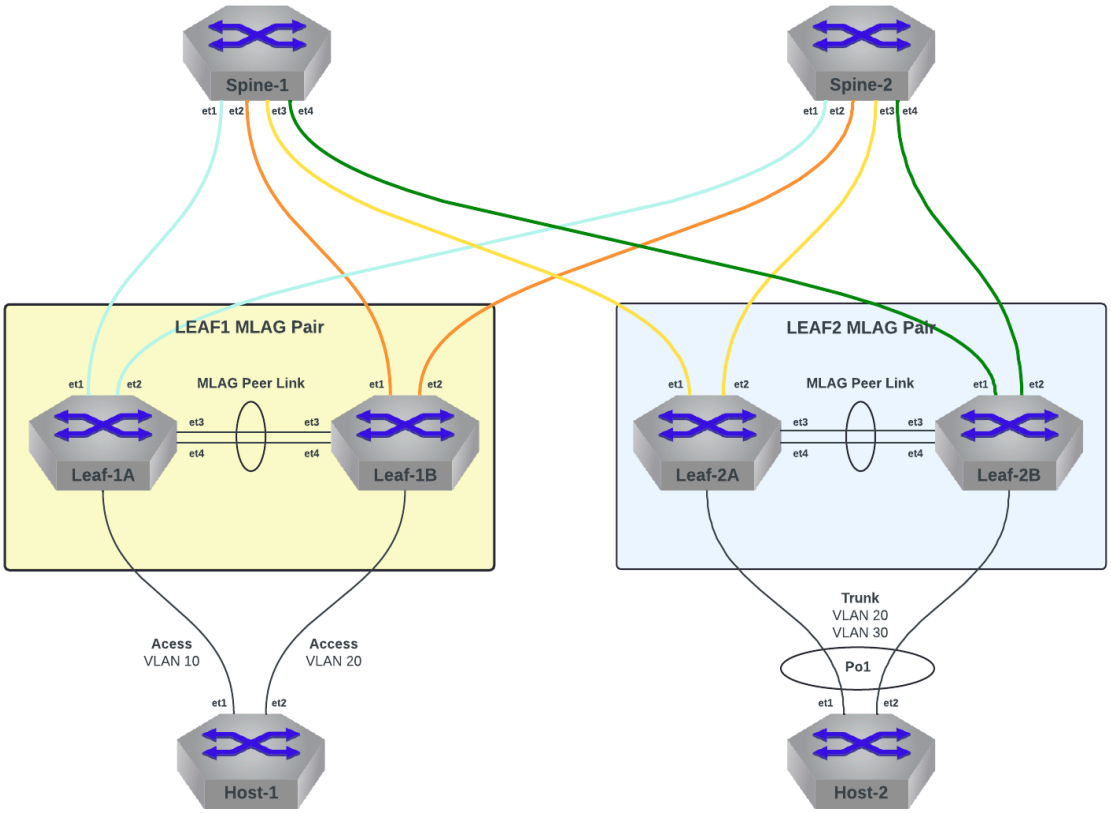
# CloudVision Mastery Workshop
# Lab Guide


## Table of Contents

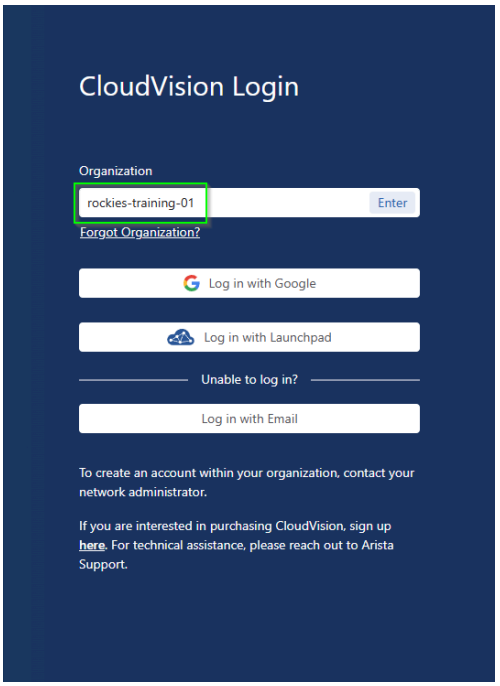
## Lab Topology

Each lab environment consists of a Layer 3 Leaf & Spine design. There are 2 Spine switches and 4 Leaf switches. The Leaf switches are connected in MLAG pairs. There is a single host device connected to each MLAG pair. Host-1 is simulating a single-homed device, connected on Leaf-1A and Leaf-1B on different access VLAN's. Host-2 is acting as a dual-homed server using an LACP Port-Channel to allow for active/active sharing of workloads.

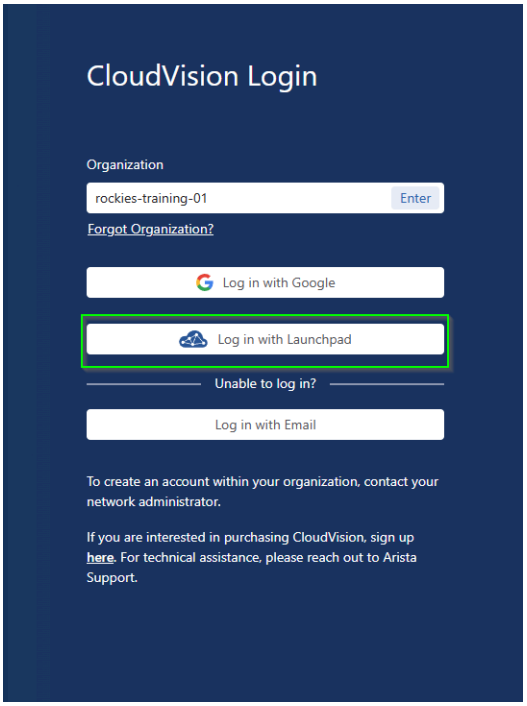


# Connecting to your CVaaS Instance

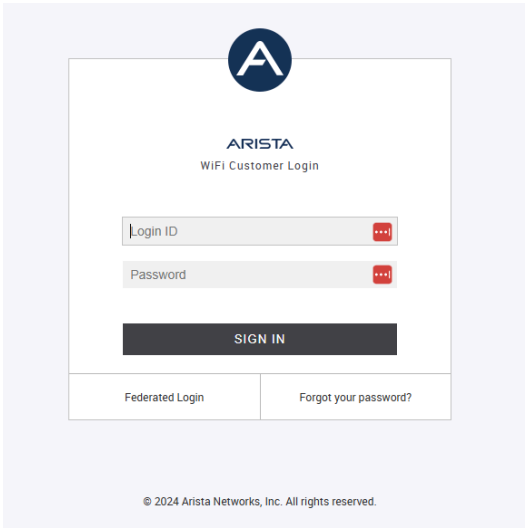
1. Open a browser (preferably Chrome or Firefox) and go to https://www.arista.io/.
2. In the "**Organization**" box, enter your assigned CVaaS instance name. For example, ***rockies-training-01***.



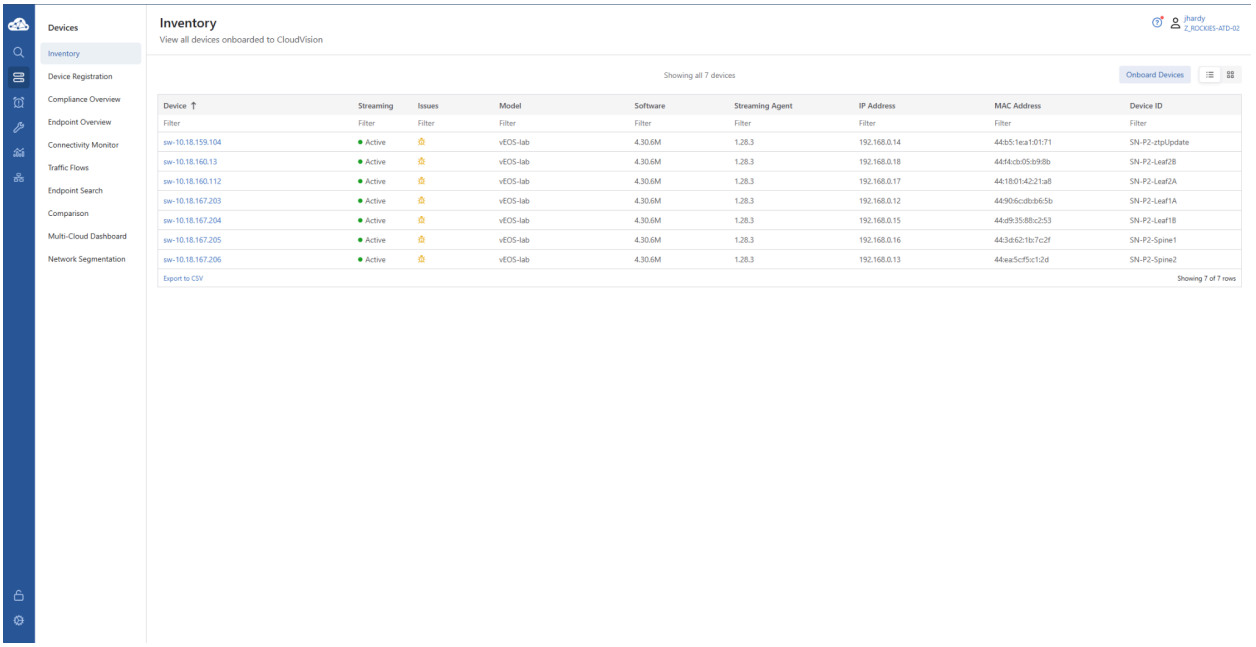

3. Click "**Log in with Launchpad**"




4. Enter in your assigned email address and password




5. Welcome to CloudVision!

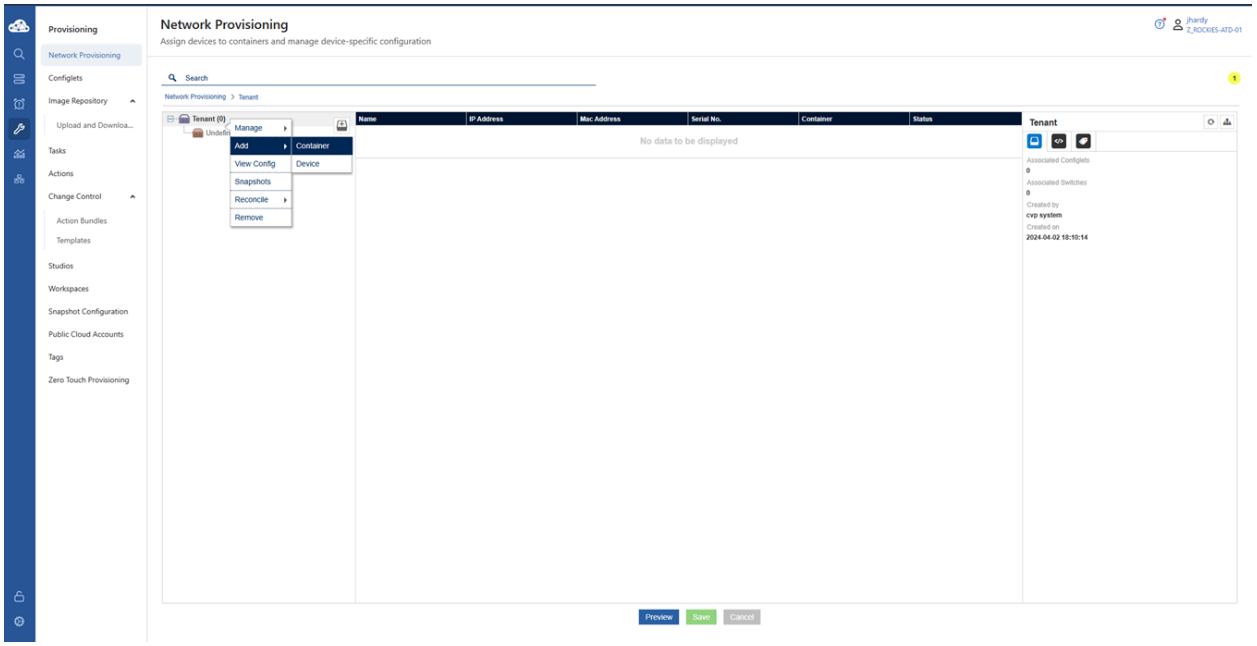



# Containers

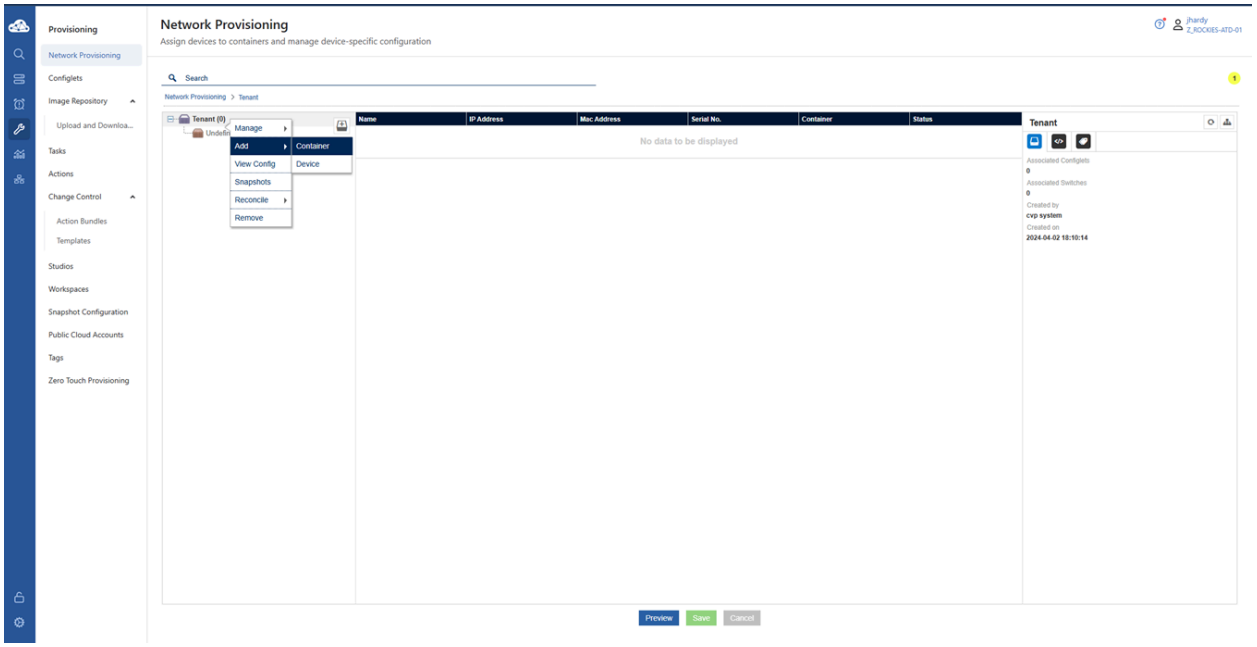
Containers are a logical entity used to group network devices and to define a hierarchy to which configurations can be applied. A container can contain a device, a group of devices, or other containers.

In this lab we will be creating a Container structure for our virtual data center deployment.

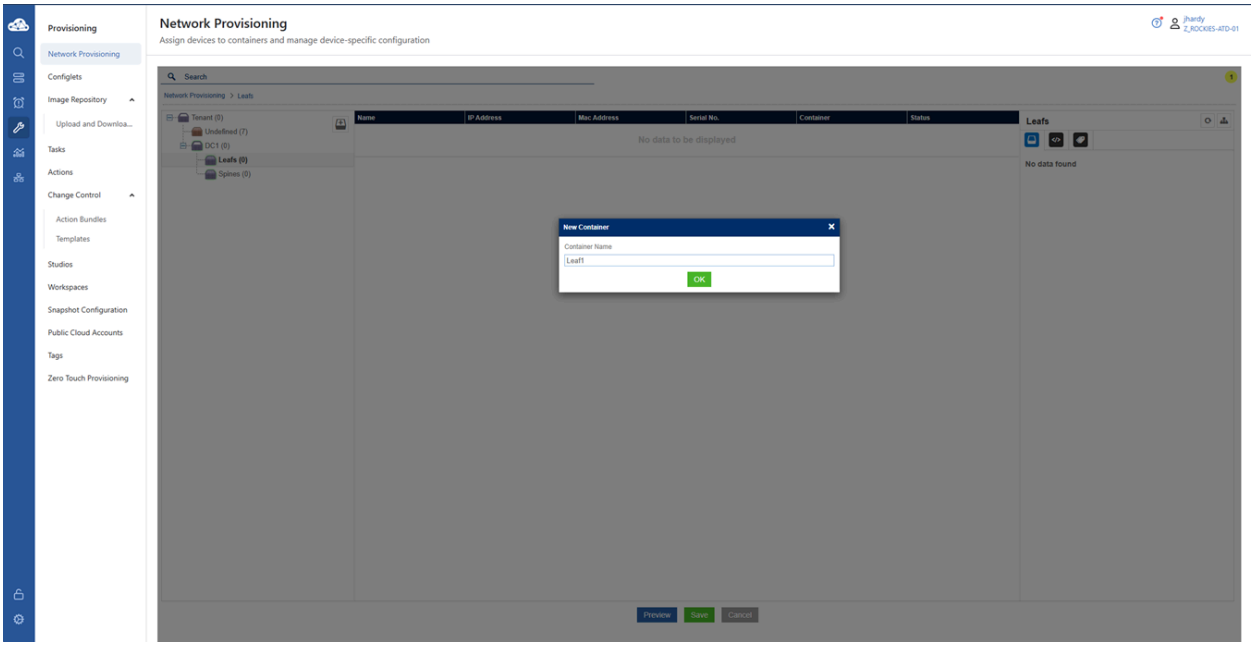
1. Navigate to **Provisioning** > **Network Provisioning**
2. Right-click on the “**Tenant**” container and select **Add** > **Container**




3. Name the new container "**DC1**" and click OK.
4. Now, right-click on the "**DC1**" container and select **Add** > **Container**
5. Name the new container "**Spines**"

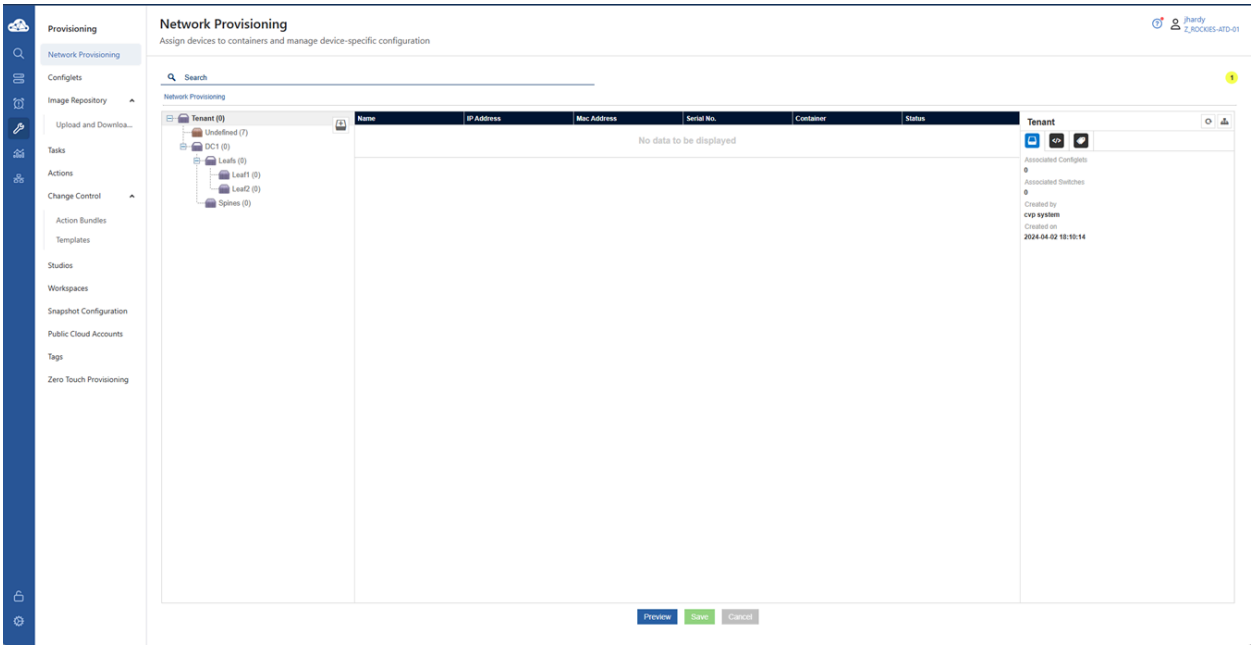



6. Repeat steps 4 & 5 and create a new container named "**Leafs**"
7. Right-click the "**Leafs**" container and create 2 new containers named "**Leaf1**" & "**Leaf2**" (see example below)




**NOTE:** You'll notice that all new containers are highlighted in green. This means there are changes that have yet to be saved.

8. Click "**Save**" at the bottom of the screen. Our finished container structure should look something like this:



# Configlets

Configlets are a snippet of configuration that CloudVision users code and maintain independently. These configlets can be later applied to devices or containers in the topology. When you apply a configlet to a container, the configlet is automatically applied to all of the devices in the container, including devices in sub-containers. Multiple configlets can be applied to a single device or container. The collection of applied configlets create the “Designed Configuration” on the devices.

## Creating Static Configlets

In this lab there will be several pre-built configlets and several that need to be built. In the steps below, we will be creating several new static configlets. We will then assign all configlets to the appropriate containers and devices.

A note on configlet naming convention. Based on our experience helping with many large customer deployments, we are using a naming convention that we found works very well. All configlets that are assigned directly to containers will be in all-caps (i.e. **SPINES, LEAF1, LEAF2,** etc) and any configlet that is assigned directly to a device will be in all lower-case (i.e. **spine-1, spine-2, leaf-1A, leaf-1B,** etc).

**NOTE**: In this section we’ll be copying code blocks that may span multiple pages. So please ensure you’re copying the entire code block before pasting into your configlet. Also, you must use keyboard shortcuts (Ctrl-C, Ctrl-V, Command-C, Command-V) in order to copy/paste inside of CVP.

1. Navigate to **Provisioning** > **Configlets**. In the upper right-hand corner, click on the “**+**” icon  and select “**Configlets**”

2. Name the new configlet “**spine-2**” and add the following configuration and click “**Save**”

```
hostname Spine-2
!
interface Ethernet1
   description Leaf-1A_eth2
   no switchport
   ip address 10.242.1.1/30
!
interface Ethernet2
   description Leaf-1B_eth2
   no switchport
```

```
   ip address 10.242.1.5/30
!
interface Ethernet3
   description Leaf-2A_eth2
   no switchport
   ip address 10.242.2.1/30
!
interface Ethernet4
   description Leaf-2B_eth2
   no switchport
   ip address 10.242.2.5/30
!
interface Loopback0
   ip address 10.250.0.2/32
!
interface Management1
   ip address dhcp
   dhcp client accept default-route
!
router bgp 65000
   router-id 10.250.0.2
   neighbor 10.242.1.2 peer group MLAG-1
   neighbor 10.242.1.6 peer group MLAG-1
   neighbor 10.242.2.2 peer group MLAG-2
   neighbor 10.242.2.6 peer group MLAG-2
!
```

3. Let's create another static configlet named "**LEAF2**" and add the following configuration and click "**Save**"

```
vlan 30,40
!
vlan 4093-4094
   trunk group MLAG_Peer_Link
!
interface Port-Channel1
   description Host-2
   switchport mode trunk
   mlag 1
!
interface Port-Channel2000
   description MLAG_Peer_Link
   switchport mode trunk
   switchport trunk group MLAG_Peer_Link
!
interface Ethernet3
   description MLAG-PEER-LINK
   channel-group 2000 mode active
!
interface Ethernet4
```

```
   description MLAG-PEER-LINK
   channel-group 2000 mode active
!
interface Vlan30
   ip virtual-router address 10.30.30.1
!
interface Vlan40
   ip virtual-router address 10.40.40.1
!
ip virtual-router mac-address 00:1c:73:00:00:01
!
mlag configuration
   domain-id MLAG
   peer-link Port-Channel2000
   reload-delay 60
!
router bgp 65002
   no bgp default ipv4-unicast
   maximum-paths 2
   neighbor MLAG-PEER peer group
   neighbor MLAG-PEER remote-as 65002
   neighbor SPINES peer group
   neighbor SPINES remote-as 65000
   redistribute connected
   !
   address-family ipv4
      neighbor MLAG-PEER activate
      neighbor SPINES activate
!
```

4. Now let's create configlets for each of our devices in the "**Leaf2**" container (**leaf-2a** & **leaf-2b**). Create a configlet named "**leaf-2a**" and add the following configuration

```
hostname Leaf-2A
!
interface Ethernet1
   description Spine-1_eth3
   no switchport
   ip address 10.241.2.2/30
!
interface Ethernet2
   description Spine-2_eth3
   no switchport
   ip address 10.242.2.2/30
!
interface Ethernet6
   description Host-2_eth1
   channel-group 1 mode active
!
interface Loopback0
```

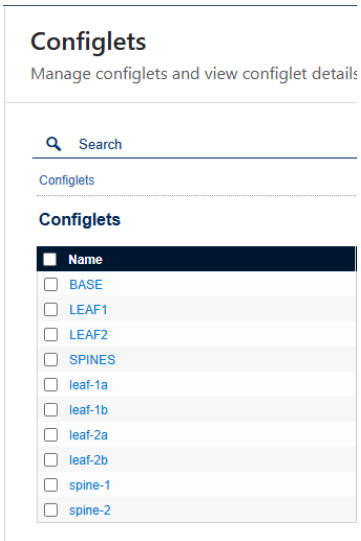
```
   ip address 10.250.2.1/32
!
interface Management1
   ip address dhcp
   dhcp client accept default-route
!
interface Vlan30
   ip address 10.30.30.2/24
!
interface Vlan40
   ip address 10.40.40.2/24
!
interface Vlan4093
   description iBGP_Peering
   no autostate
   ip address 192.168.0.1/30
!
interface Vlan4094
   description MLAG_Peering
   no autostate
   ip address 192.168.1.1/30
!
mlag configuration
   local-interface Vlan4094
   peer-address 192.168.1.2
!
router bgp 65002
   router-id 10.250.2.1
   neighbor 10.241.2.1 peer group SPINES
   neighbor 10.242.2.1 peer group SPINES
   neighbor 192.168.0.2 peer group MLAG-PEER
!
```

5. Create a configlet named “**leaf-2b**” and add the following configuration

```
hostname Leaf-2B
!
interface Ethernet1
   description Spine-1_eth4
   no switchport
   ip address 10.241.2.6/30
!
interface Ethernet2
   description Spine-2_eth4
   no switchport
   ip address 10.242.2.6/30
!
interface Ethernet6
   description Host-2_eth2
   channel-group 1 mode active
!
```

```
interface Loopback0
   ip address 10.250.2.2/32
!
interface Management1
   ip address dhcp
   dhcp client accept default-route
!
interface Vlan30
   ip address 10.30.30.3/24
!
interface Vlan40
   ip address 10.40.40.3/24
!
interface Vlan4093
   description iBGP_Peering
   no autostate
   ip address 192.168.0.2/30
!
interface Vlan4094
   description MLAG_Peering
   no autostate
   ip address 192.168.1.2/30
!
mlag configuration
   local-interface Vlan4094
   peer-address 192.168.1.1
!
router bgp 65002
   router-id 10.250.2.2
   neighbor 10.241.2.5 peer group SPINES
   neighbor 10.242.2.5 peer group SPINES
   neighbor 192.168.0.1 peer group MLAG-PEER
!
```

6. We should now have the following list of configlets

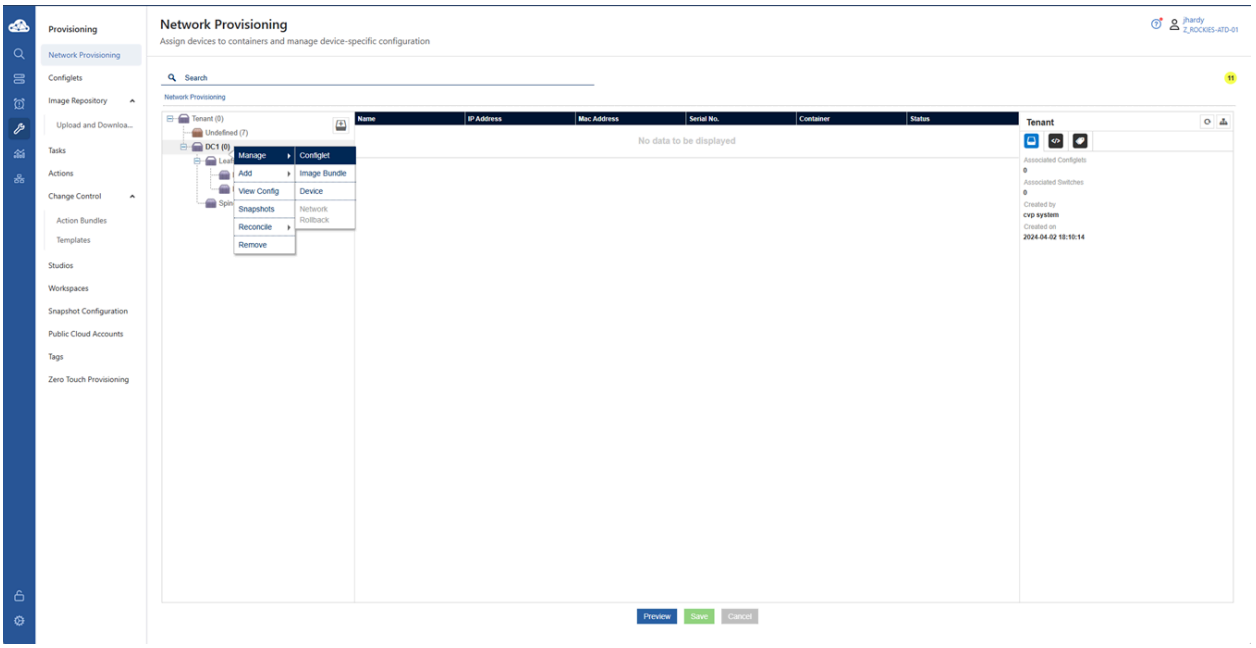


# Assigning Configlets to Containers

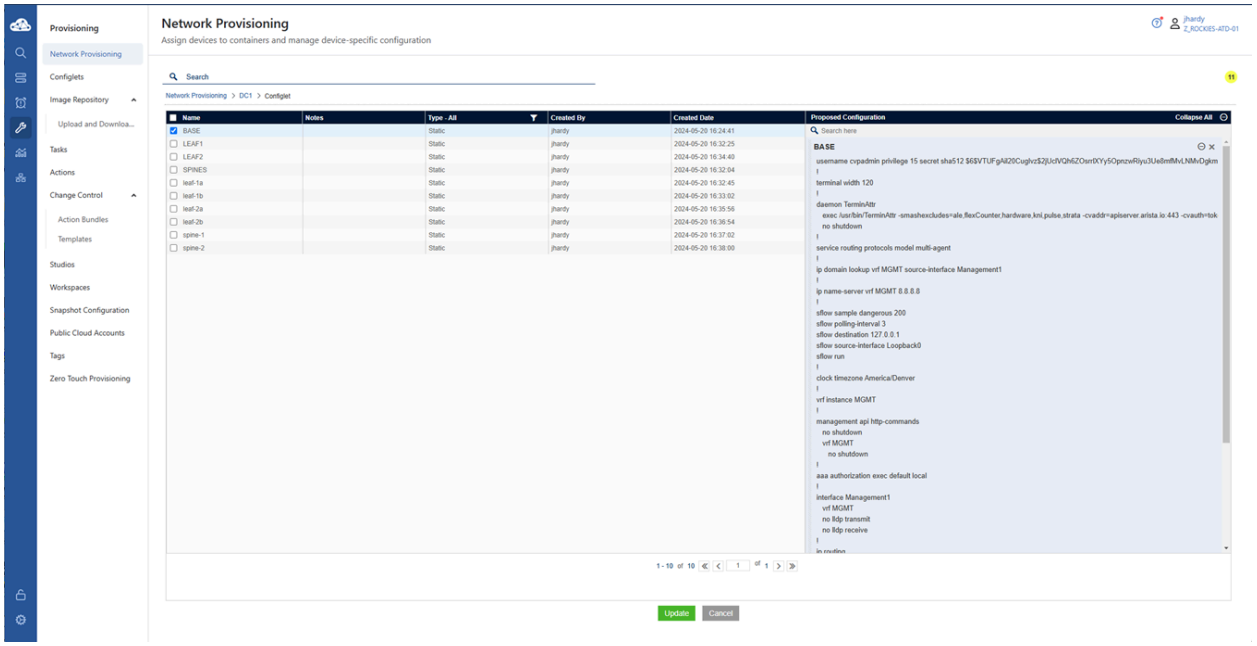
Now that we have our container structure and our static configlets built, let's apply the configlets to our containers.

**NOTE:** Remember, any configlet that is assigned to a container will be inherited by the devices under that container.

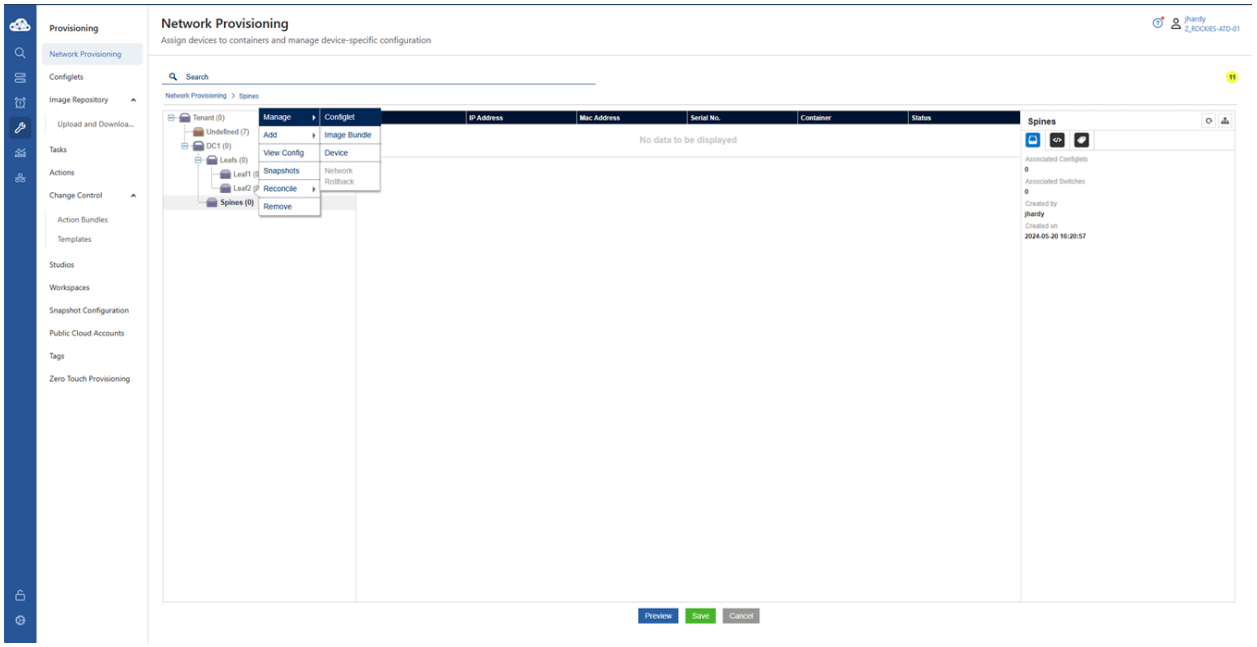
1. Let's start by assigning the "**BASE**" configlet to the "**DC1**" container. Navigate to **Provisioning** > **Network Provisioning**. Right-click on the "**DC1**" container and select **Manage** > **Configlet**



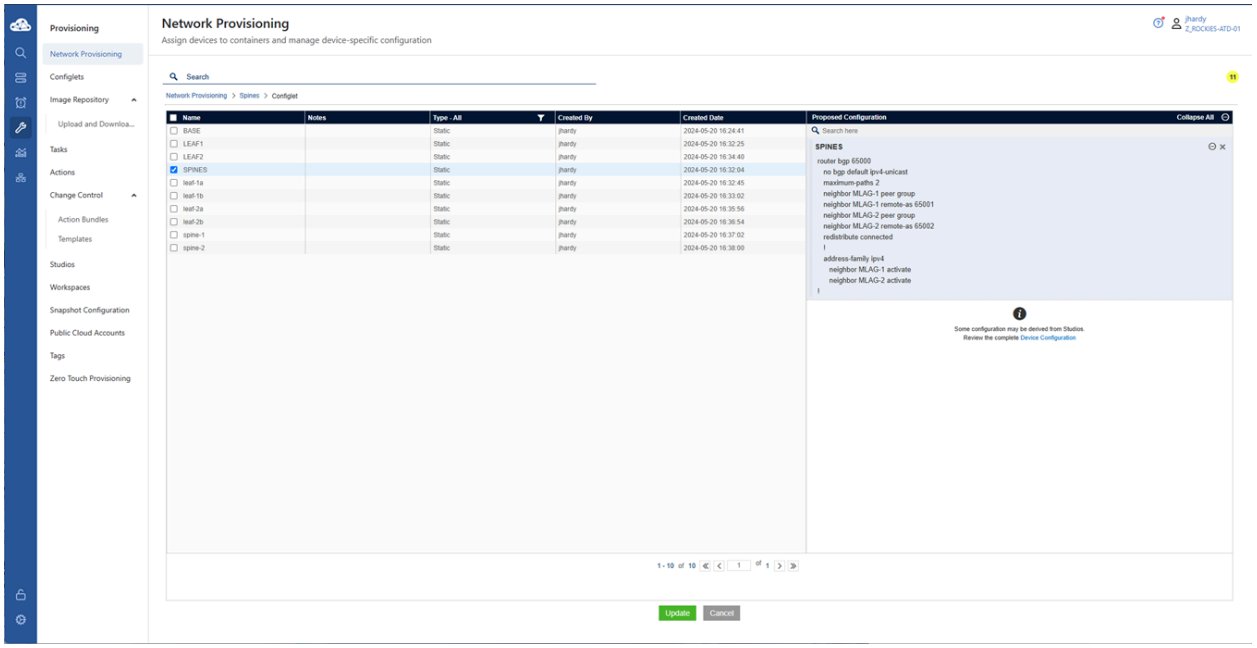

2. Now select "**BASE**" from the list of configlets. You will see the "**BASE**" configlet show up on the right side of your screen under the "**Proposed Configuration**" section. You can see a preview of the configuration that will be applied by this configlet by clicking the "**+**" icon on the far right corner of the "**Proposed Configuration**" section for the "**BASE**" configlet. Click "**Update**" at the bottom of the screen.



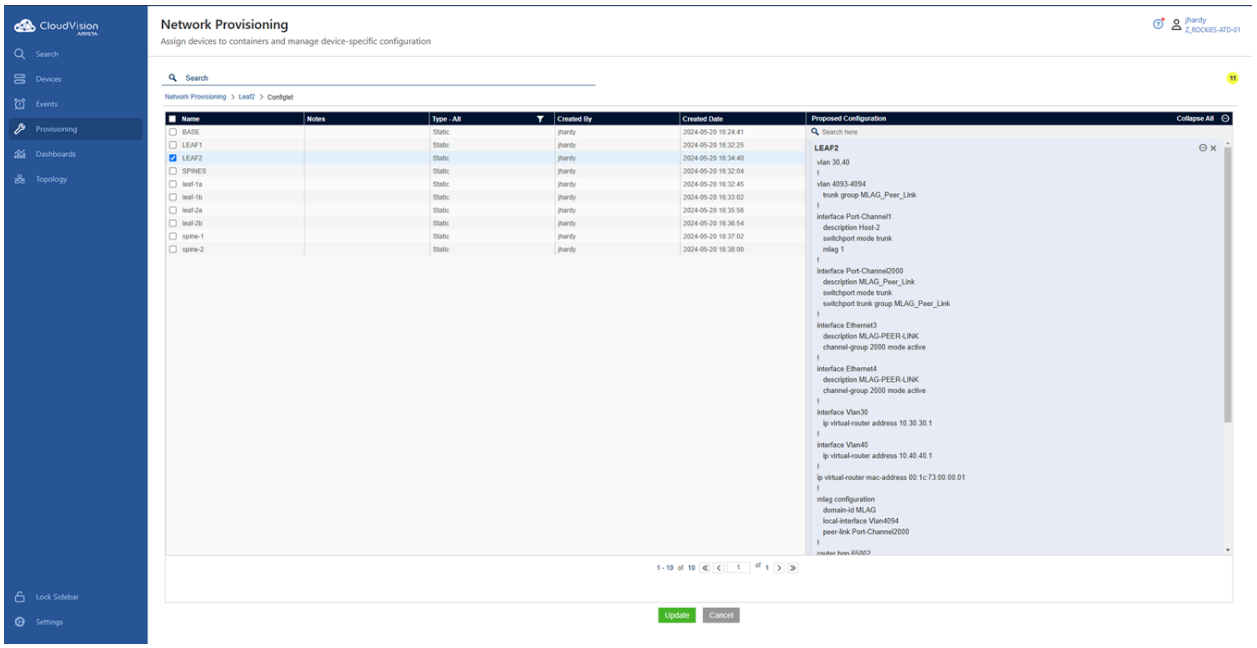

3. Next, let's assign the "**SPINES**" configlet to the "**Spines**" container. Right-click on the "**Spines**" container and select **Manage** > **Configlet**




4. From the list, select the "**SPINES**" configlet and click "**Update**" at the bottom of the screen




5. Next, we're going to assign the "**LEAF1**" configlet to the "**Leaf1**" container. Expand the "**Leafs**" container to expose the individual leafs containers. Right-click on the "**Leaf1**" container and select **Manage** > **Configlet**. Select the "**LEAF1**" configlet and click "**Update**"




6. Repeat above steps for the "**Leaf2**" container
7. We've now assigned configlets to the appropriate containers. Don't forget to click "**Save**" at the bottom of the screen

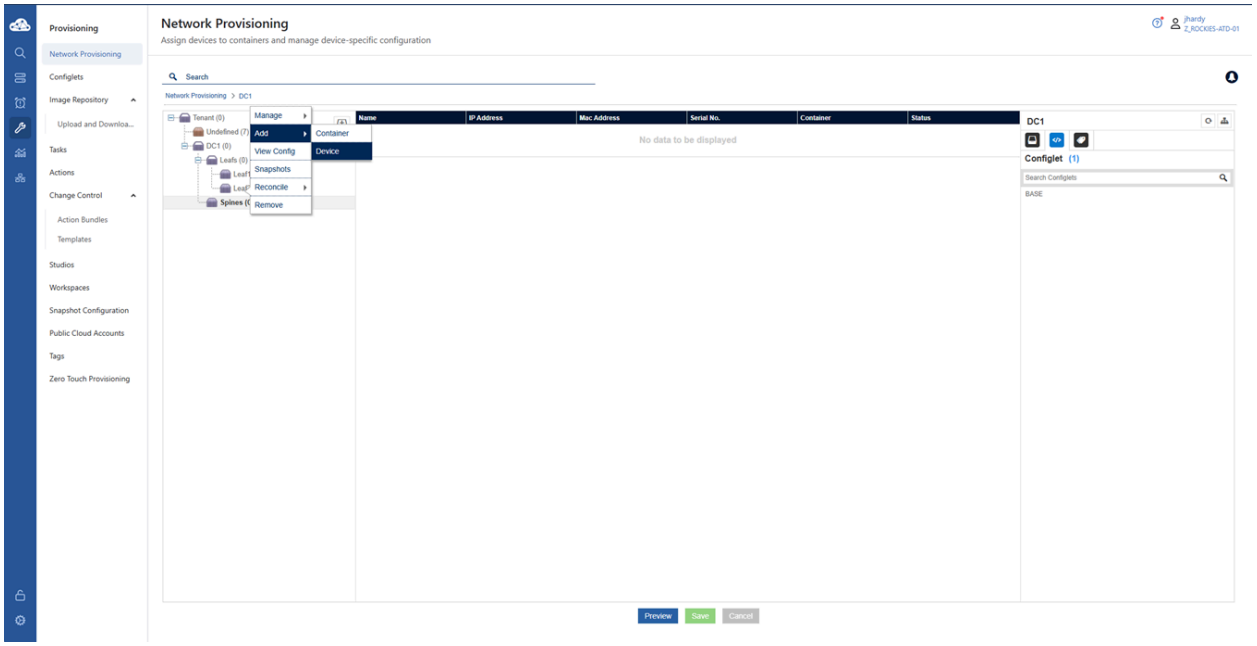
8. From the Network Provisioning screen, we can verify which configlets are assigned to the containers. Click on a container in the hierarchy, in this example we'll highlight the "**DC1**" container. From the menu on the right side of the screen, click on the middle icon. This will show you which configlets are assigned to that container


## Provisioning Switches

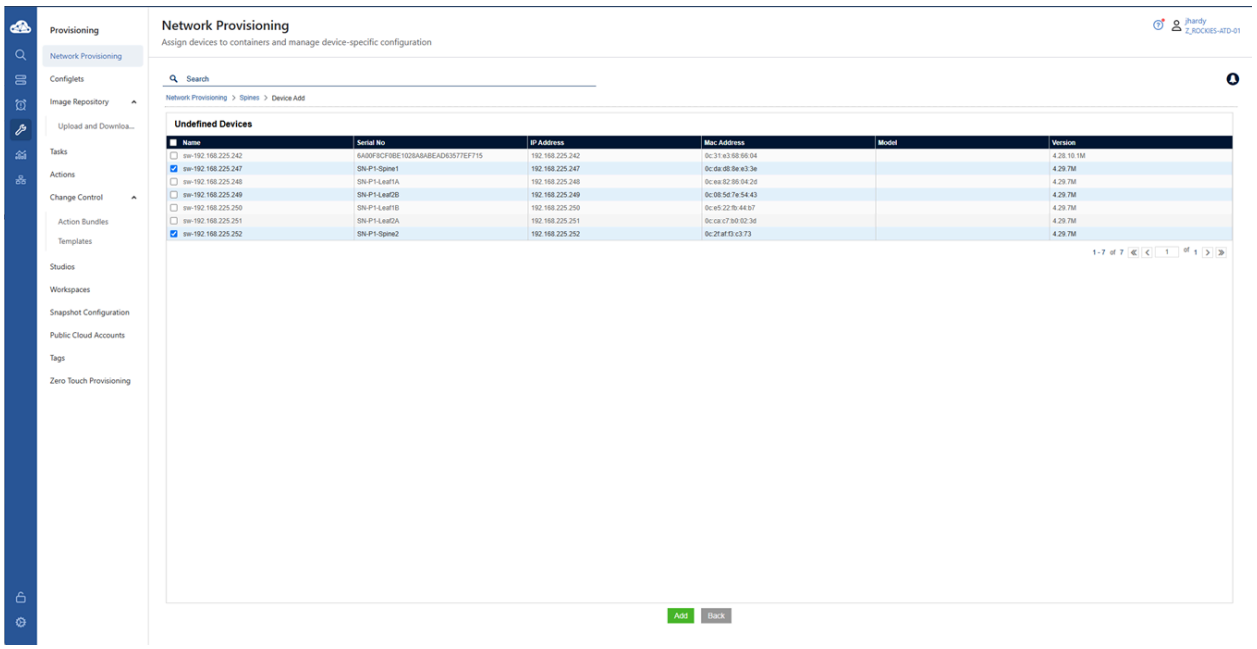
Now that we've assigned configlets to containers, we can move our switches from the "**undefined**" container to their assigned containers. Remember, all container-level configlets will be inherited by the devices under those containers.

In this lab we will provision our devices from ZTP mode, move them into their assigned containers, and apply their device-specific configlets.

1. Navigate to **Provisioning** > **Network Provisioning** and right-click on the "**Spines**" container. Now select **Add** > **Device**

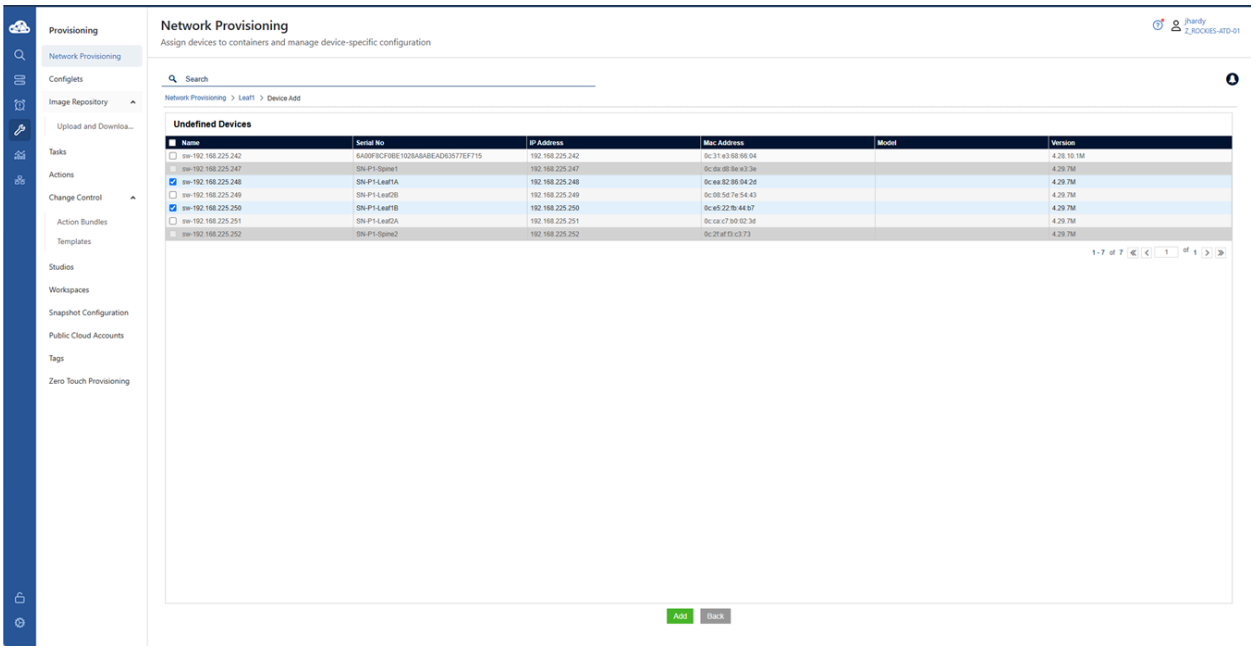



2. Select “**Spine1**” and “**Spine2**” (denoted in the Serial Number) from the list and click “**Add**” at the bottom of the screen




**NOTE**: Do not click “**Save**” at this point as we have yet to assign any device-specific configlets to the switches. This is especially important when using static IP addresses to manage the devices. Otherwise we could lose access to the switches after provisioning.

3. Open the container hierarchy and right-click on the "**Leaf1**" container and select **Add** > **Device**
4. Select "**Leaf1A**" and "**Leaf1B**" from the list and click "**Add**"

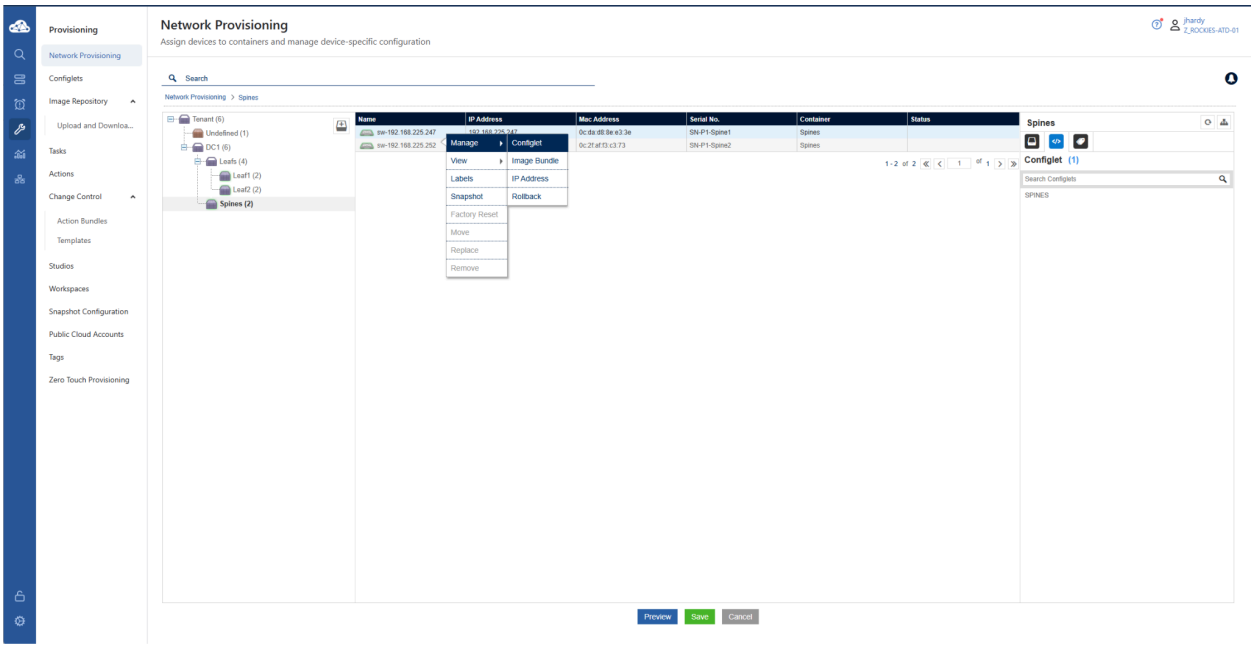



5. Let's repeat the process for container "**Leaf2**", adding "**Leaf2A**" and "**Leaf2B**"

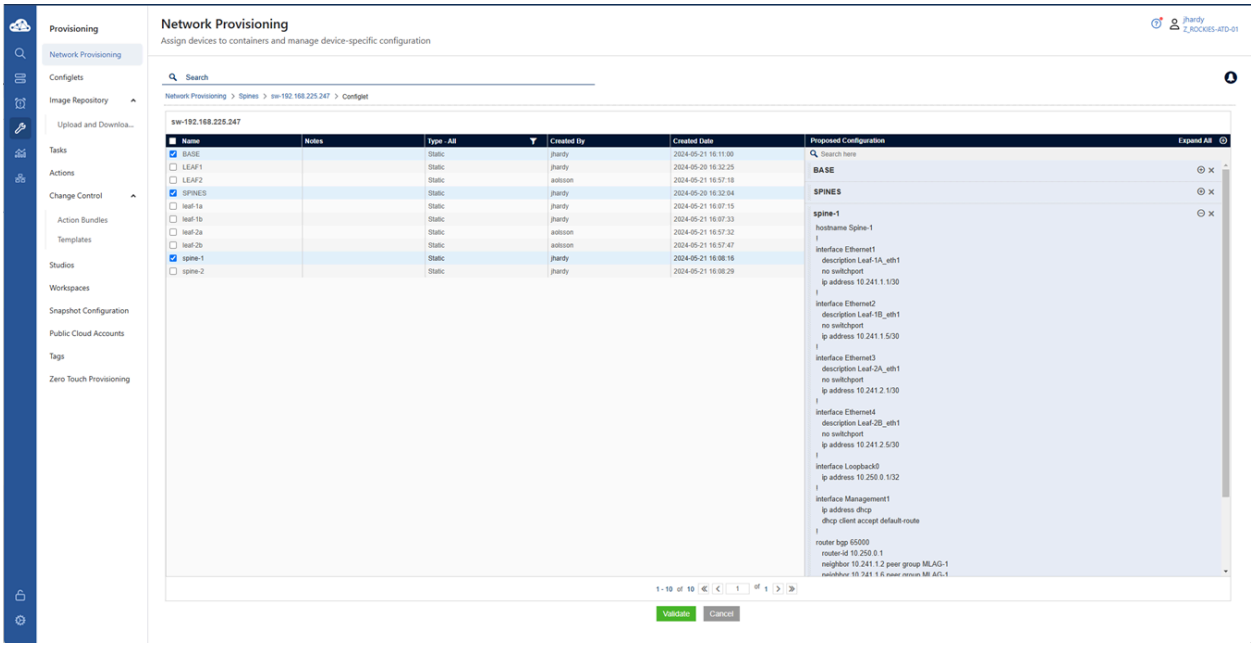
> **NOTE**: Do not click "**Save**" at this point as we have yet to assign any device-specific configlets to the switches. This is especially important when using static IP addresses to manage the devices. Otherwise we could lose access to the switches after provisioning.

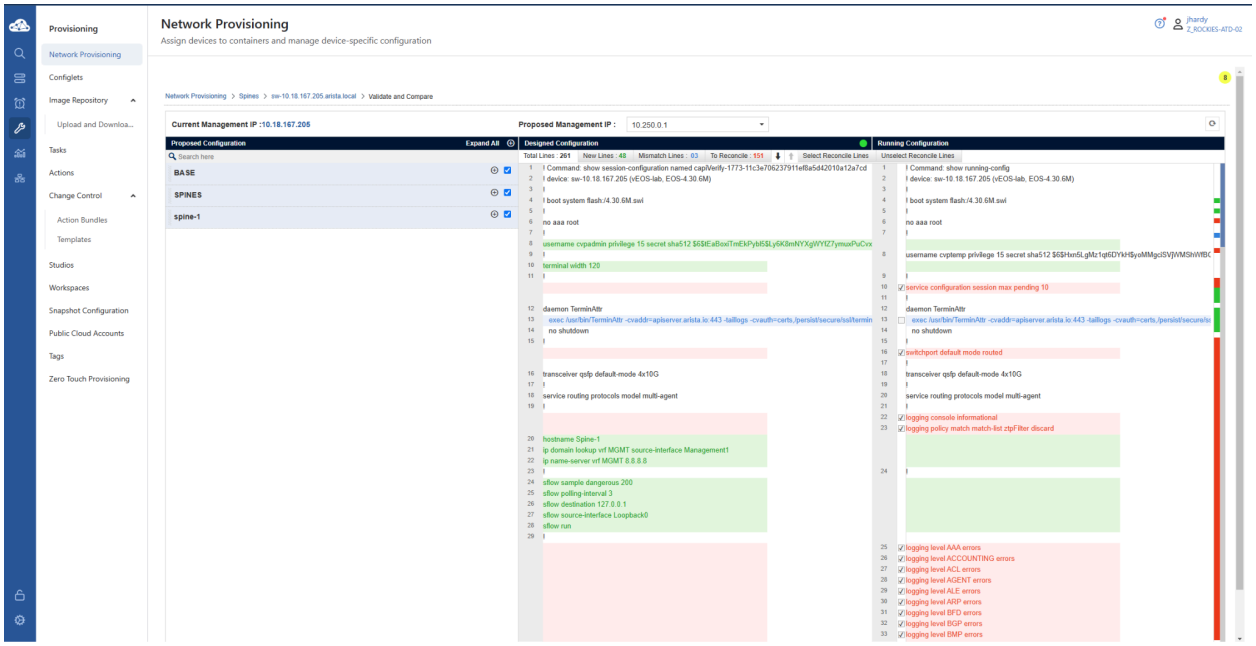
We are now ready to assign our device-specific configlets to the switches themselves.

6. Click on the "**Spines**" container and right-click on "**Spine1**". Select **Manage** > **Configlet**



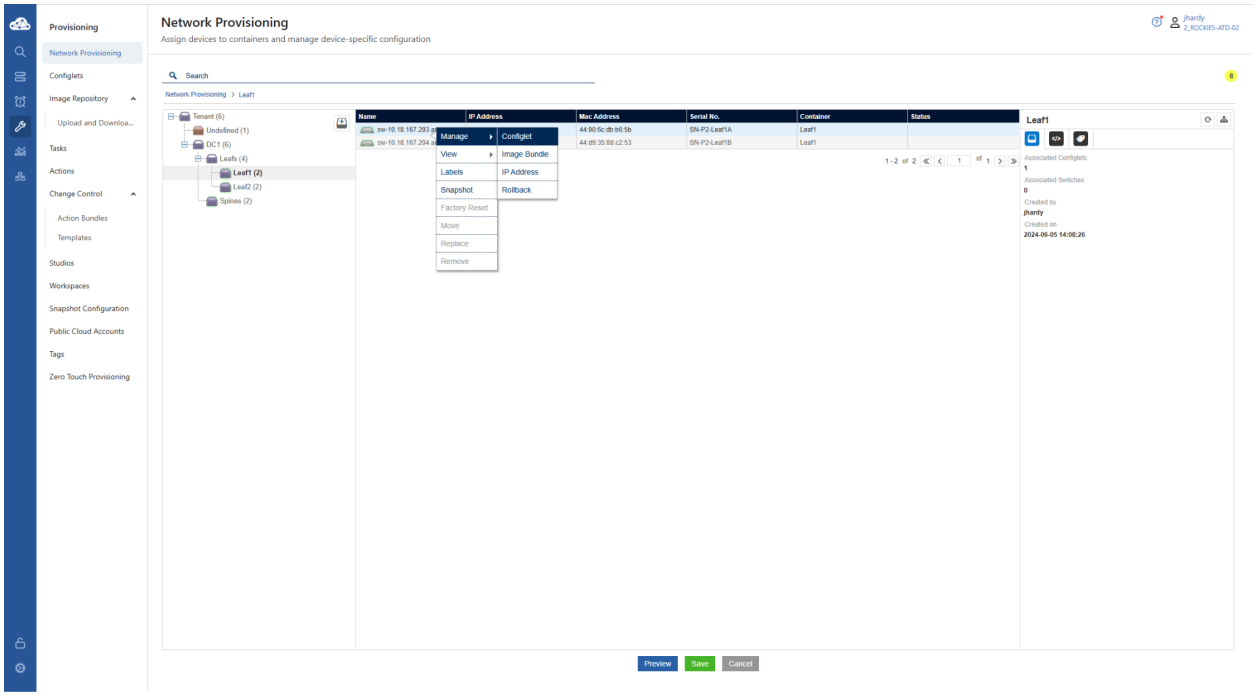

7. Now select the "**spine-1**" configlet from the list and click "**Validate**" at the bottom of the screen



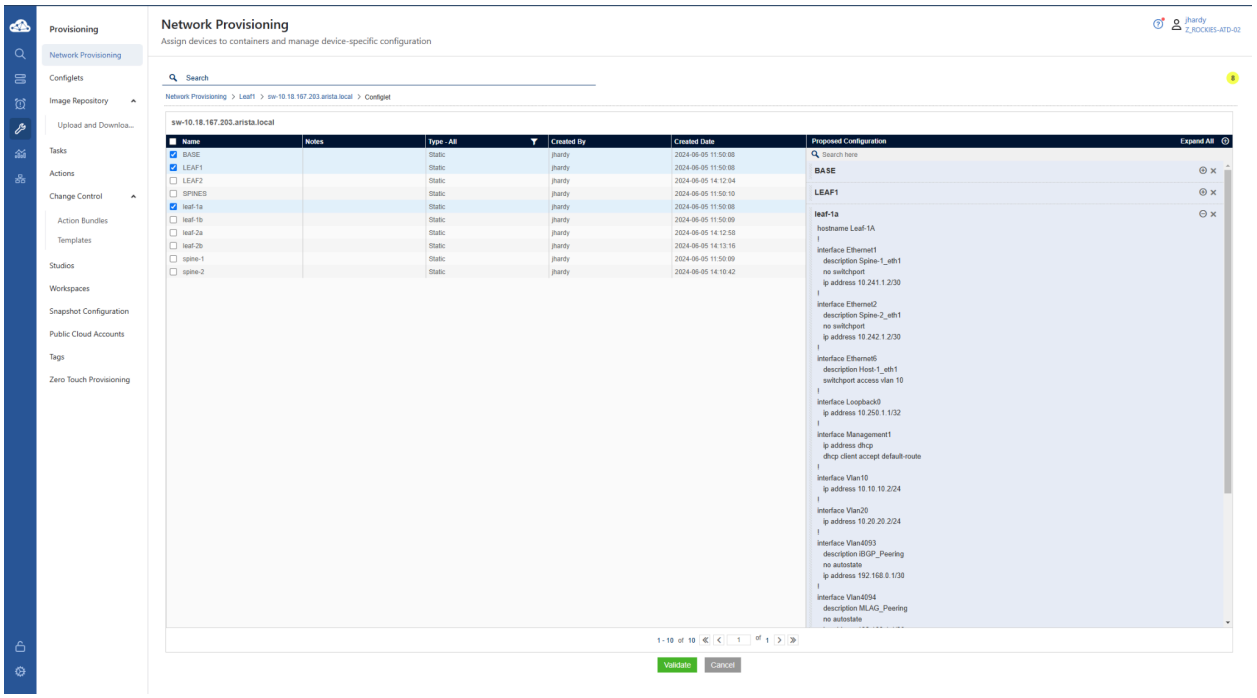



8. Under the "**Designed Configuration**" pane, we should see approximately 48 New Lines, 03 Mismatch Lines, and 151 Lines To Reconcile.


9. Verify the "**Designed Configuration**" against the "**Running Configuration**". If you're satisfied with the proposed changes, click "**Save**" at the bottom of the screen.
10. Let's repeat the above process for "**Spine2**" by assigning the "**spine-2**" configlet to that device. We should see the same number of proposed config changes for "**Spine2**" as we did for "**Spine1**".
11. Now let's assign the device-specific configlets to the Leaf switches. Click on the "**Leaf1**" container and right-click on "**Leaf1A**" and select **Manage** > **Configlet**.




12. Now select “**leaf-1a**” from the list of configlets and click “**Validate**” at the bottom of the screen.

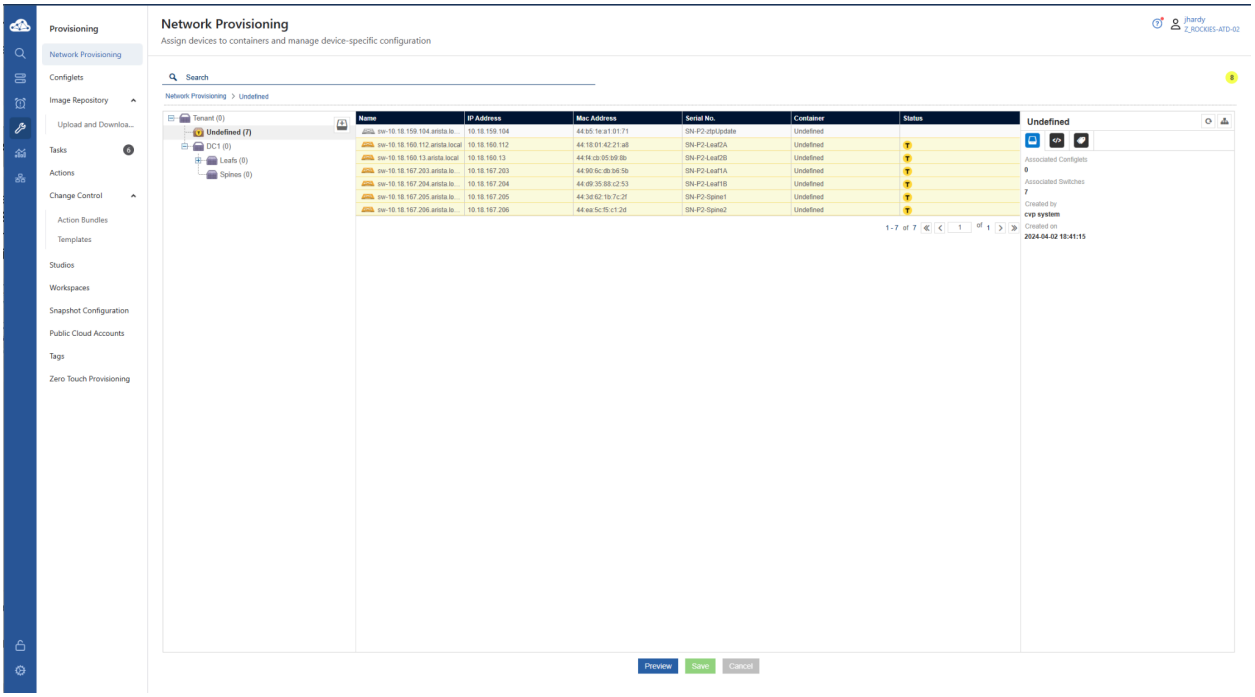



13. Under the “**Designed Configuration**” pane, we should see approximately 77 New Lines, 03 Mismatch Lines, and 157 Lines To Reconcile.


14. Verify the "**Designed Configuration**" against the "**Running Configuration**". If you're satisfied with the proposed changes, click "**Save**" at the bottom of the screen.
15. Let's repeat the above steps for "**Leaf1B**", "**Leaf2A**", & "**Leaf2B**" by assigning all device-level configlets to the appropriate switches.

**NOTE**: For "**Leaf1B**" we should see the same number of New Lines, Mismatch Lines and Lines To Reconcile as we did for "**Leaf1A**". For "**Leaf2A & 2B**" we should see approximately 81 New Lines, 03 Mismatch Lines, and 157 Lines To Reconcile.

16. Once all configlets have been assigned to the individual switches, we can go ahead and click "**Save**" at the bottom of the Network Provisioning screen.
17. You should now notice that all of the switches in the Network Provisioning view still show up under the "Undefined" container.  TYou'll now see the switches are all shown in yellow with a "**T**" in the status section, indicating that these devices have available tasks.



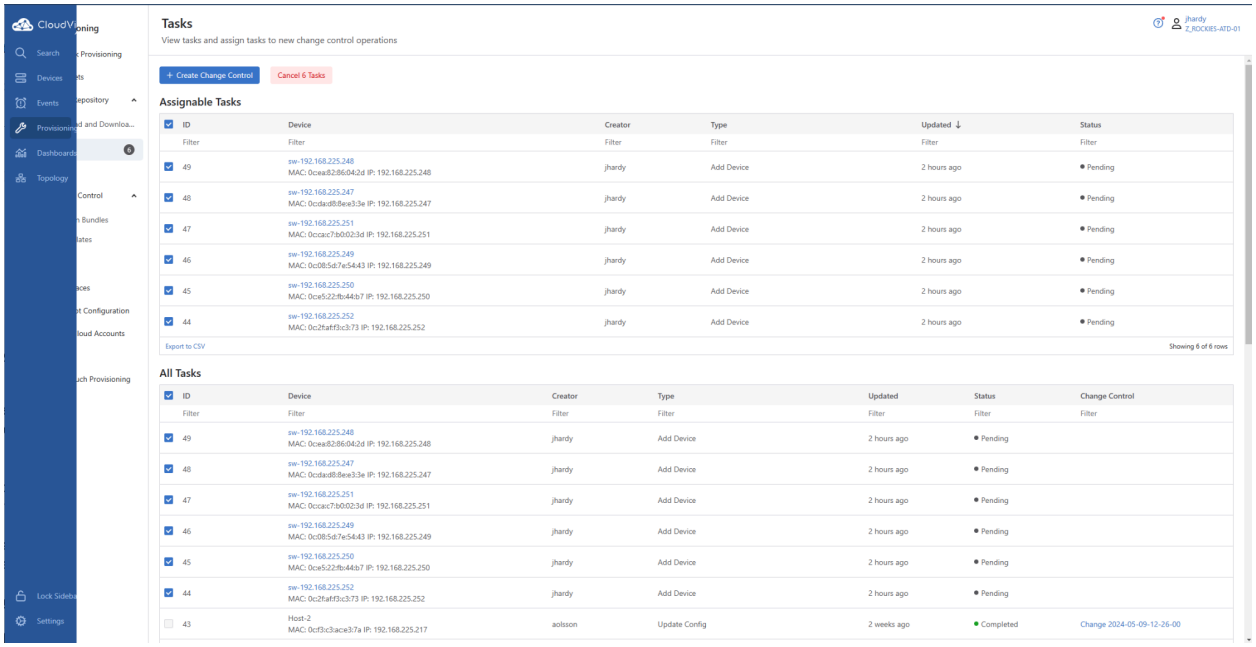

We can execute these tasks using a Change Control - which we'll cover in the next section. Only after the tasks have been executed will the switches show up under their appropriate container structure in the Network Provisioning hierarchy.

# Tasks & Change Controls

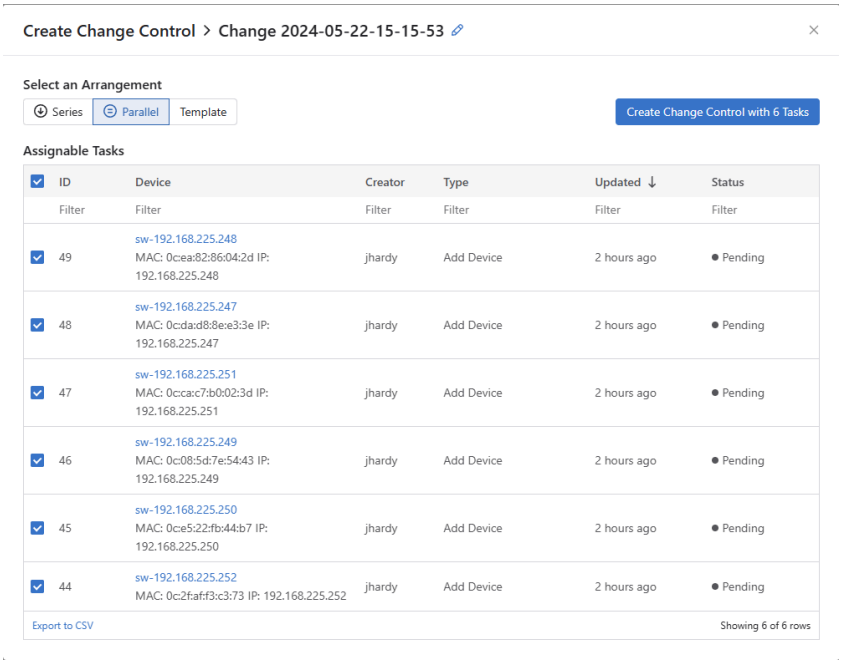
Tasks in CloudVision are generated whenever there are changes made to configlets, image bundles, etc. Those Tasks are executed using a Change Control operation. A Change Control allows a user to select and execute an individual task or a group of tasks that you want to process simultaneously.

In this lab we will assign our existing Tasks to a Change Control and then execute it so that our switches will finish the Provisioning process.

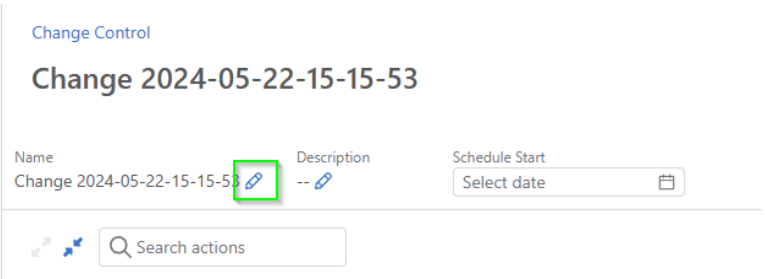
1. Navigate to **Provisioning** > **Tasks**. You should see our 6 pending "**Add Device**" Tasks. Under "**Assignable Tasks**" click the top check box next to "**ID**". This will select all 6 tasks in the list. Now click "**Create Change Control**"



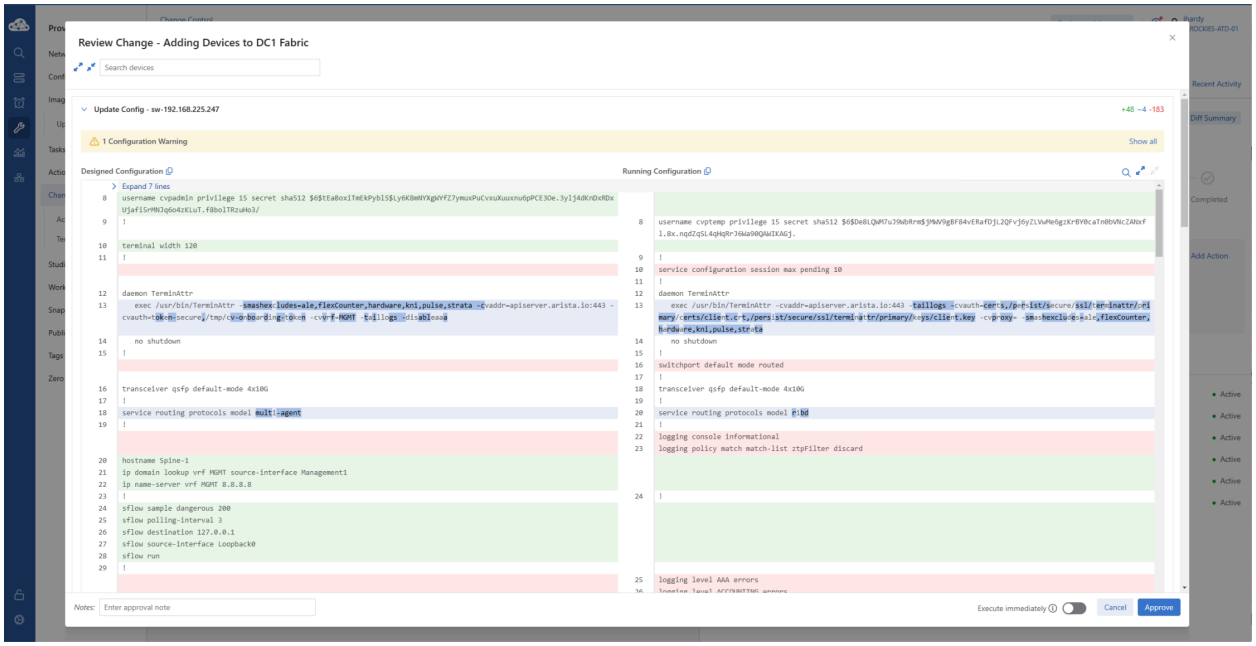

2. Under "**Select an Arrangement**" click on "**Parallel**" and then click on "**Create Change Control with 6 Tasks**"



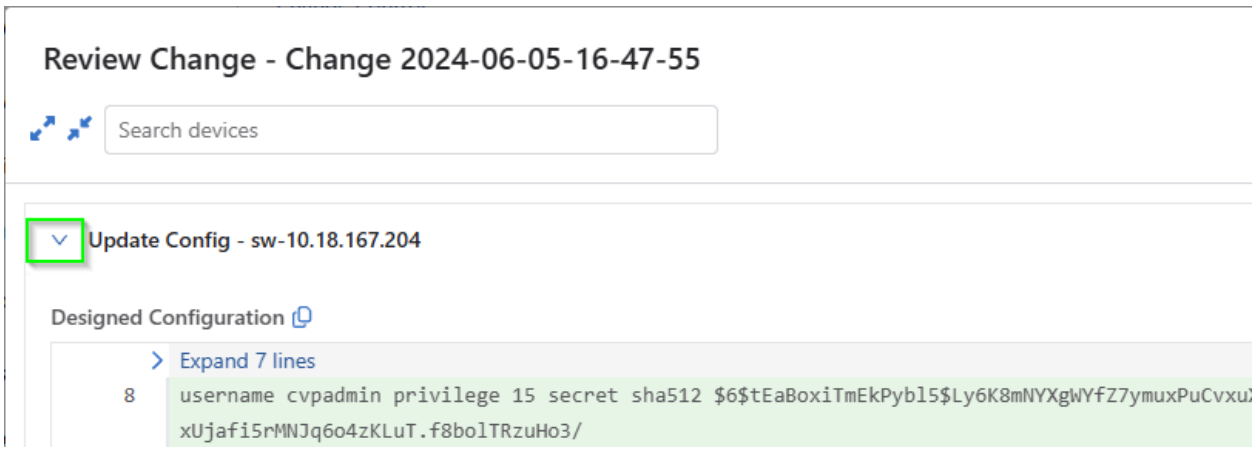

3. We're going to change the name of the Change Control to something more descriptive. This will allow us to quickly locate it later if necessary. Click on the pencil icon next to the name of the Change Control and name it, "**Adding Devices to DC1 Fabric**"



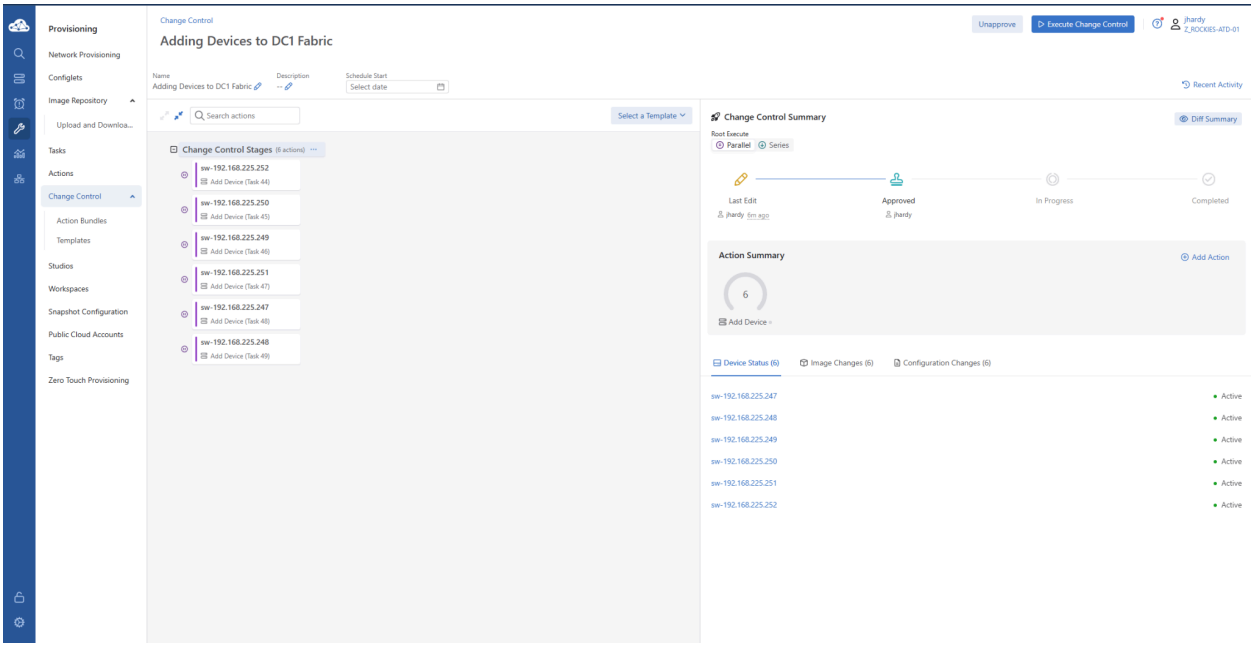

4. In the upper right-hand corner, click on "**Review and Approve**". Review the changes that are being applied to each device and then click "**Approve**" if you're satisfied with the proposed changes




While reviewing the Change Control, you'll notice that it looks very similar to what we saw when Validating the config changes on each of the devices. Again, you can see how many "**New Lines**", "**Mismatch Lines**", and "**Lines To Reconcile**" are being proposed for each device. You can also click on the ⌄ next to each device to collapse its changes - which will provide you with a more summarized view.

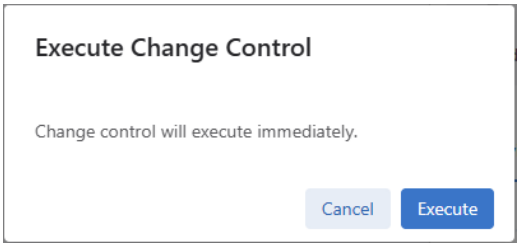



5. Once Approved, our Change Control should look something like this:

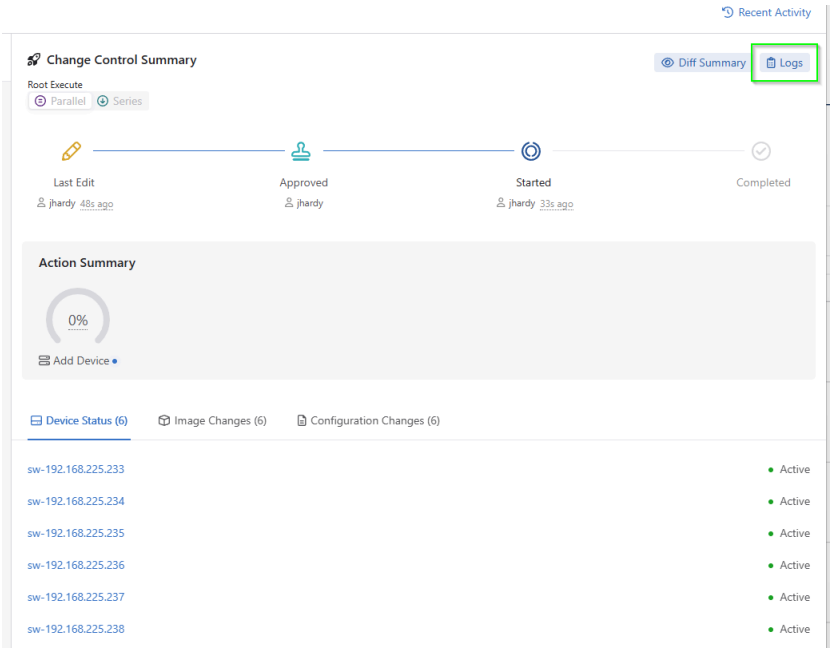


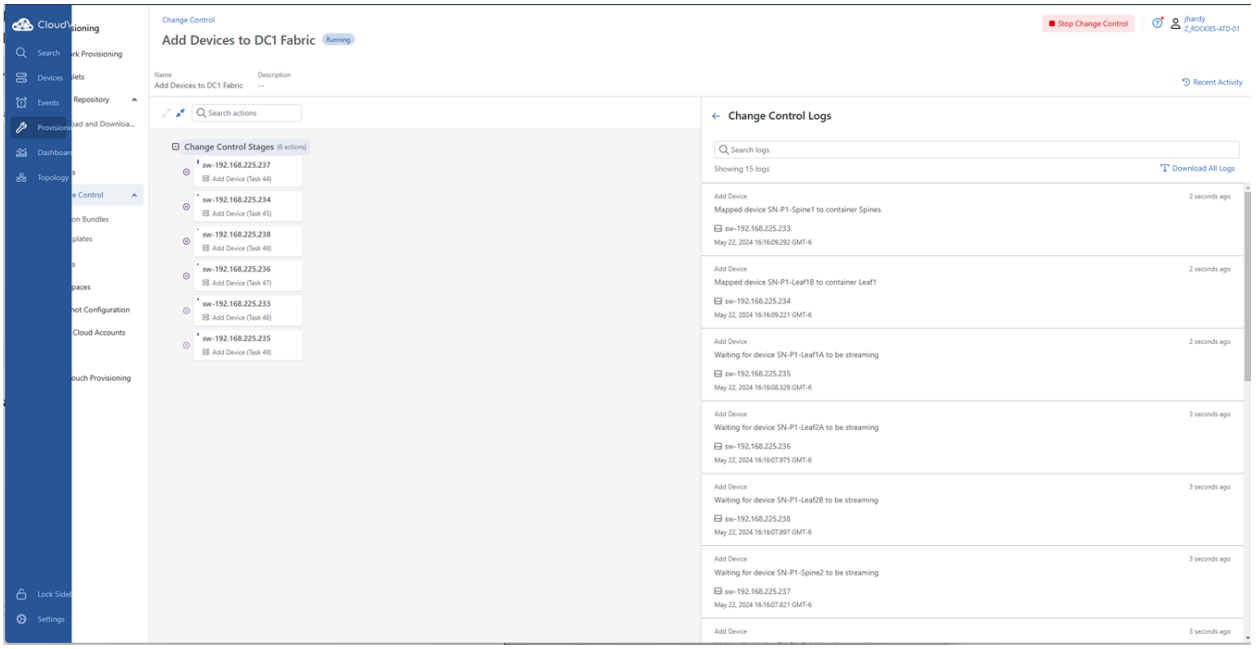

**NOTE**: No changes have actually been applied to the devices up to this point. The only way to push the changes is through the execution of the Change Control.

6. Now it's time to execute our Change Control. Click on "**Execute Change Control**" in the upper right corner. In the pop-up window, we'll click on "**Execute**".



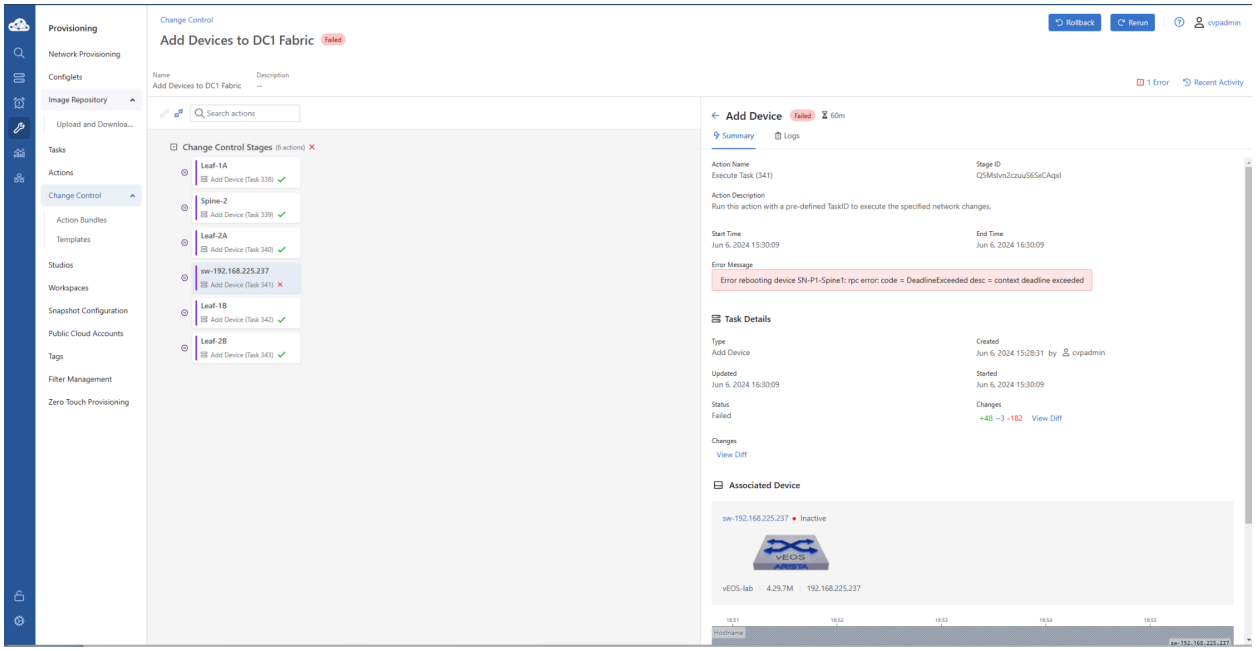
While the Change Control is running, you can click on “**Logs**” to get real-time updates as to what actions are being executed within the Change Control.



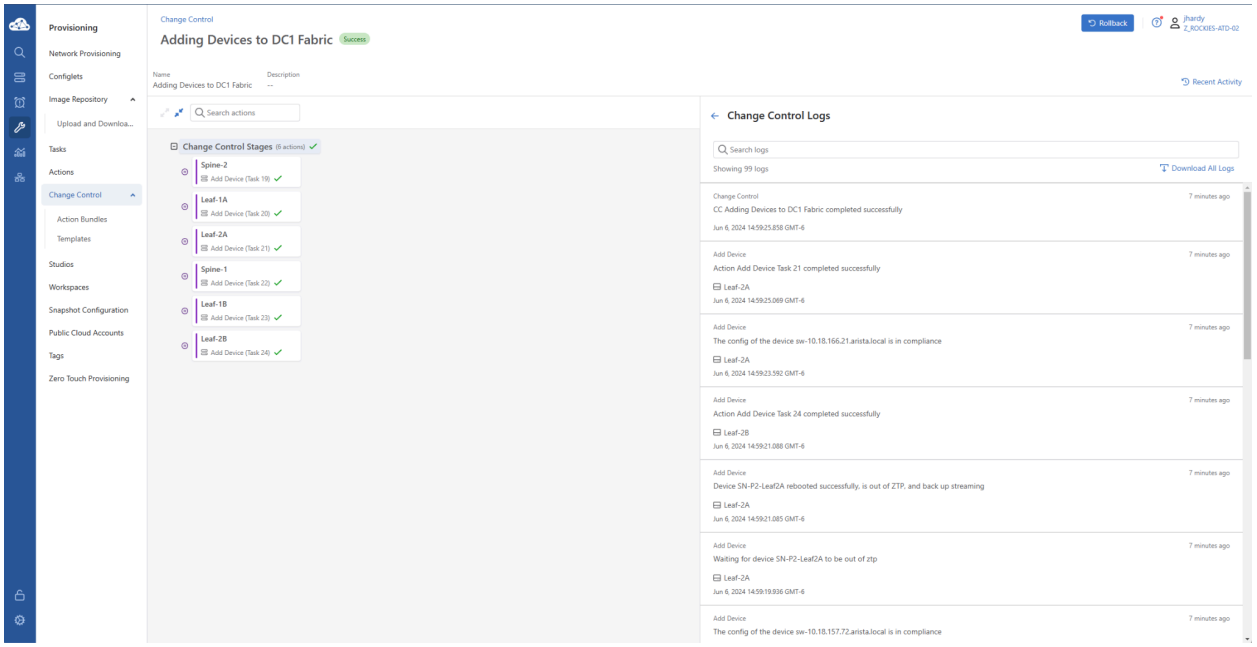




If any task in the Change Control fails, you’ll see a Red “X” next to the task. You can click on the failed task and get more details as to why the action failed.

In the example below, we can see that switch **SN-P1-Spine1** exceeded the timeout after it was rebooted during the provisioning process.




If the Change Control completes with no errors or failed Tasks, then **Congratulations**, you have successfully deployed all of the switches in your DC1 fabric!




**** If your Change Control failed for any reason, please reach out to an Arista team member for assistance. ****

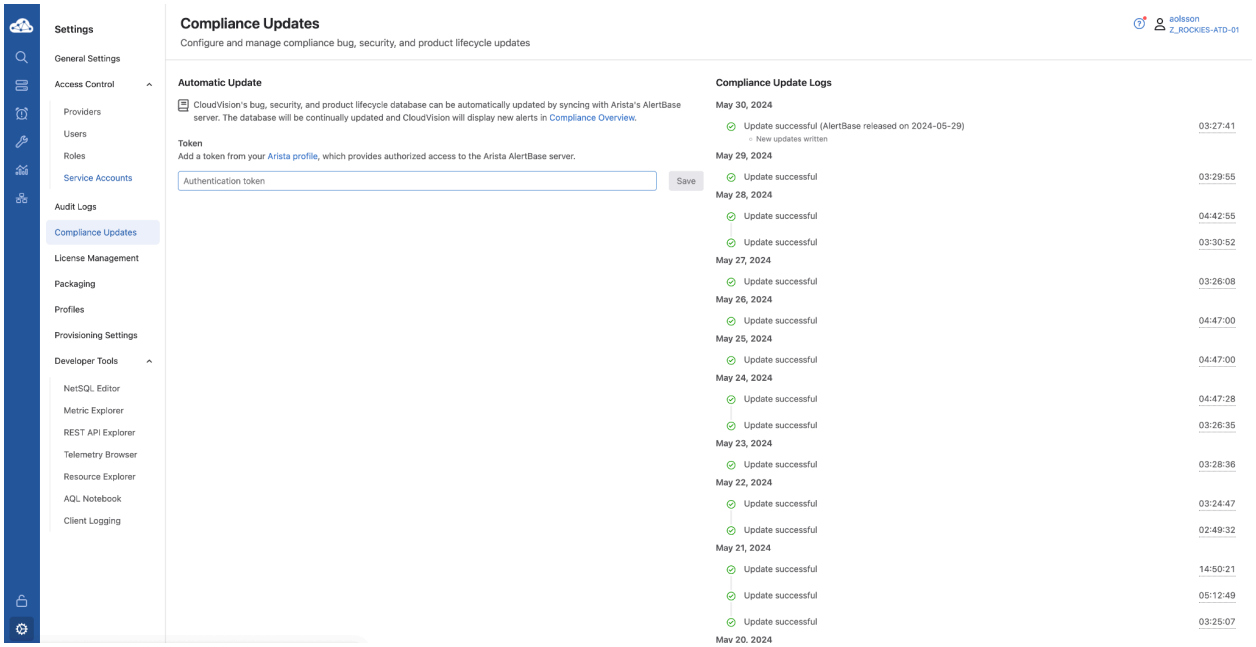
# Compliance Updates and Token

Compliance updates automatically keep the compliance settings within CloudVision up to date. These Compliance Updates include:

- Bug Database
- Software End-of-Life
- Hardware End-of-Life

Let's start by configuring the token for Compliance Updates so that the items above stay current.

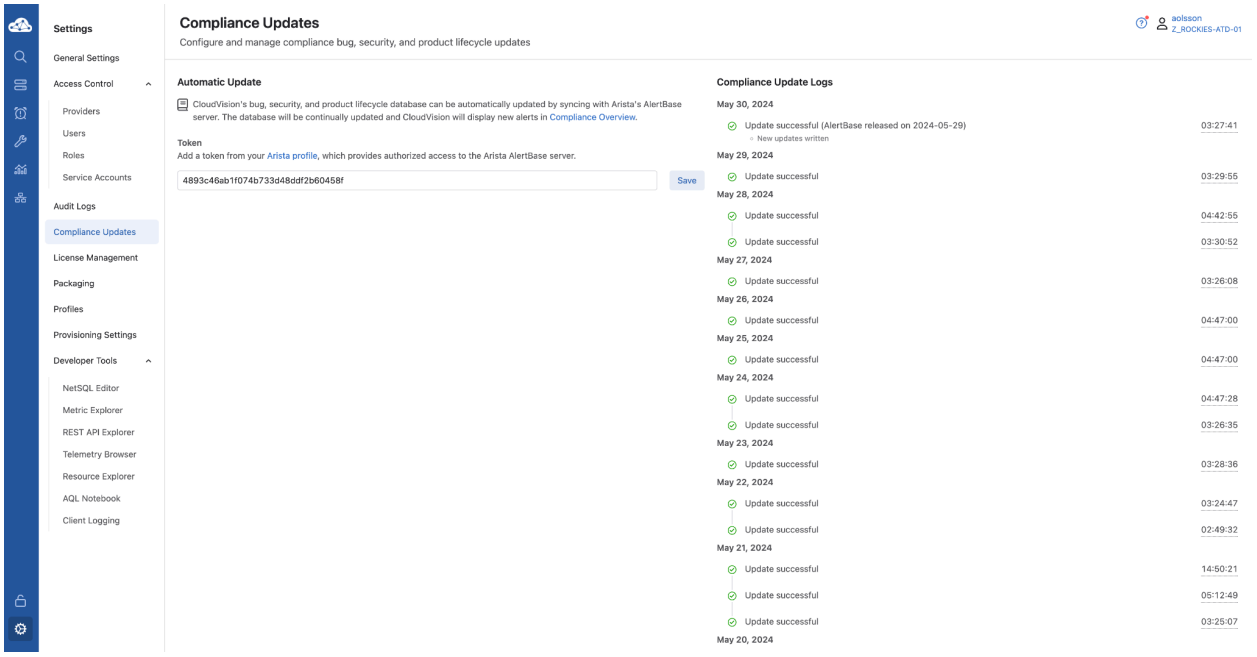
1. Navigate to **Settings** > **Compliance Updates** (Settings is the gear icon in the bottom left corner).




2. In the "**Authentication Token**" box, enter the following token

```
4893c46ab1f074b733d48ddf2b60458f
```

3. Select "**Save**" to save the Token.




**NOTE**: This token can be generated for customer environments by following the blue hyperlink to your Arista profile. Each token has a lifespan of one year, so keep in mind that it will need to be periodically updated.

Now that the Authentication Token has been supplied, CloudVision will periodically check for compliance updates daily.

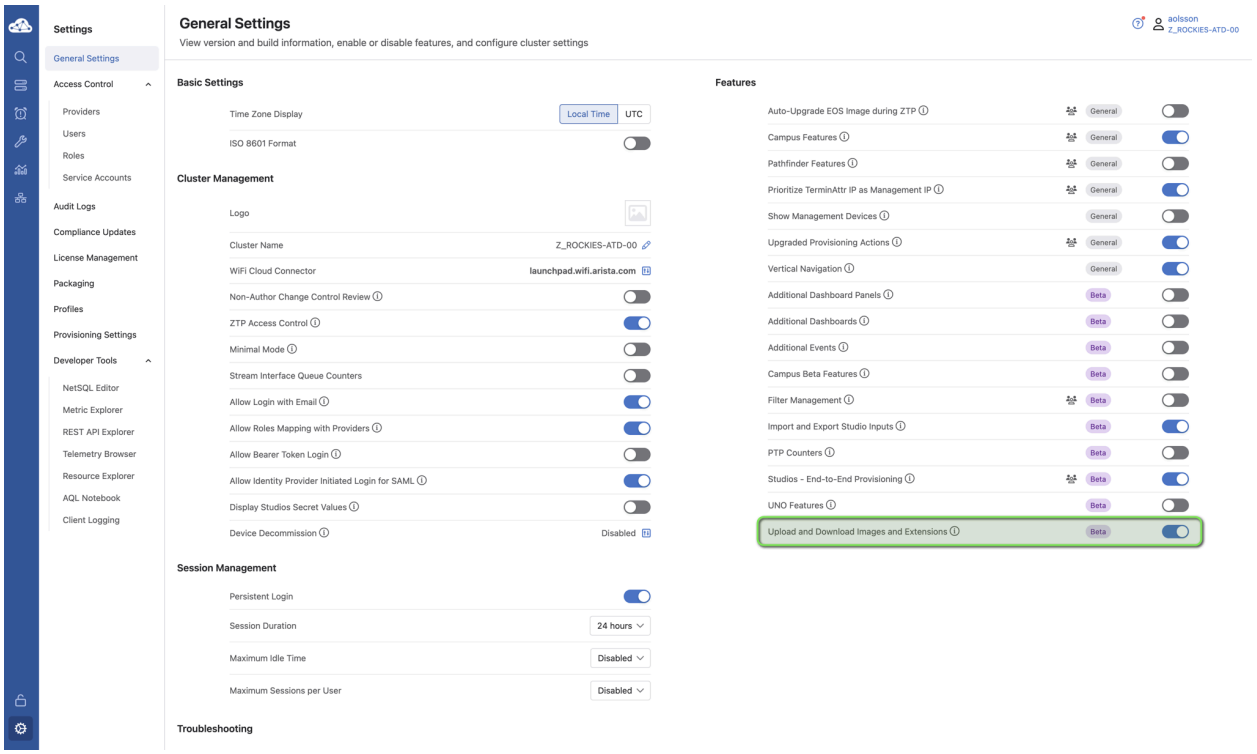
## Images and Image Bundles

CloudVision simplifies image management for all network devices. A more recent CloudVision feature now allows users to download Arista images and extensions directly from within CloudVision, minimizing the steps required to deploy images to network devices.

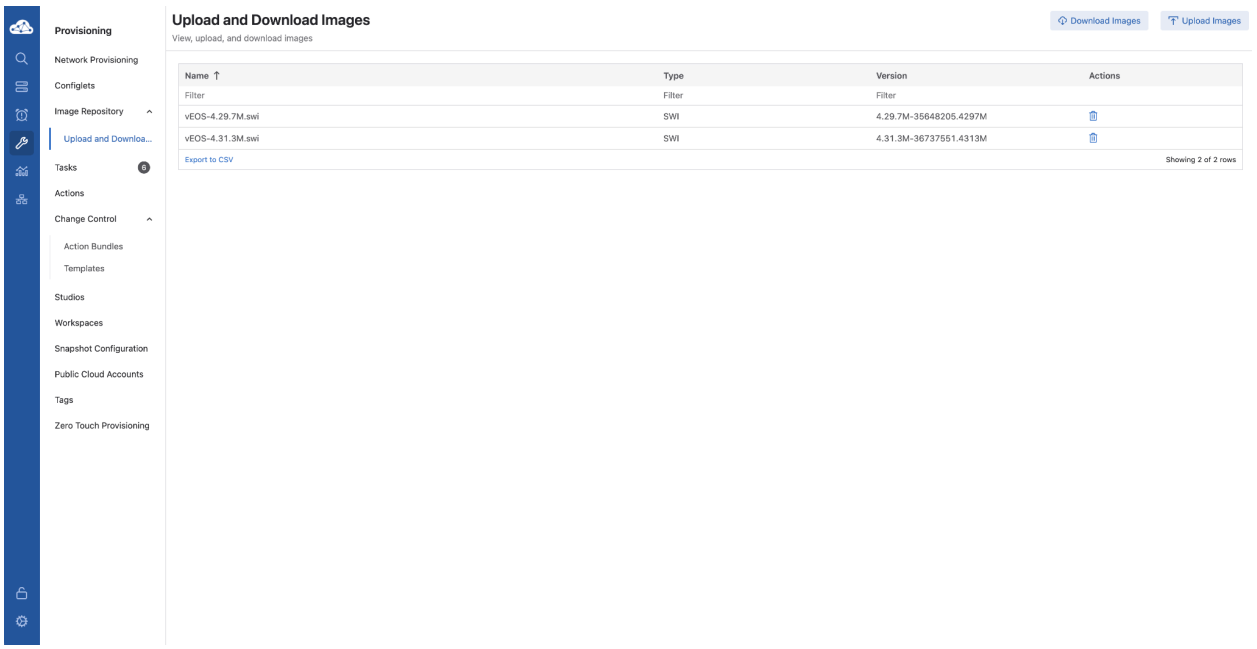
**NOTE**: The Compliance Token added in the previous section is required to be able to download images from within CloudVision

First, we must enable the feature that allows images to be downloaded from within CloudVision.

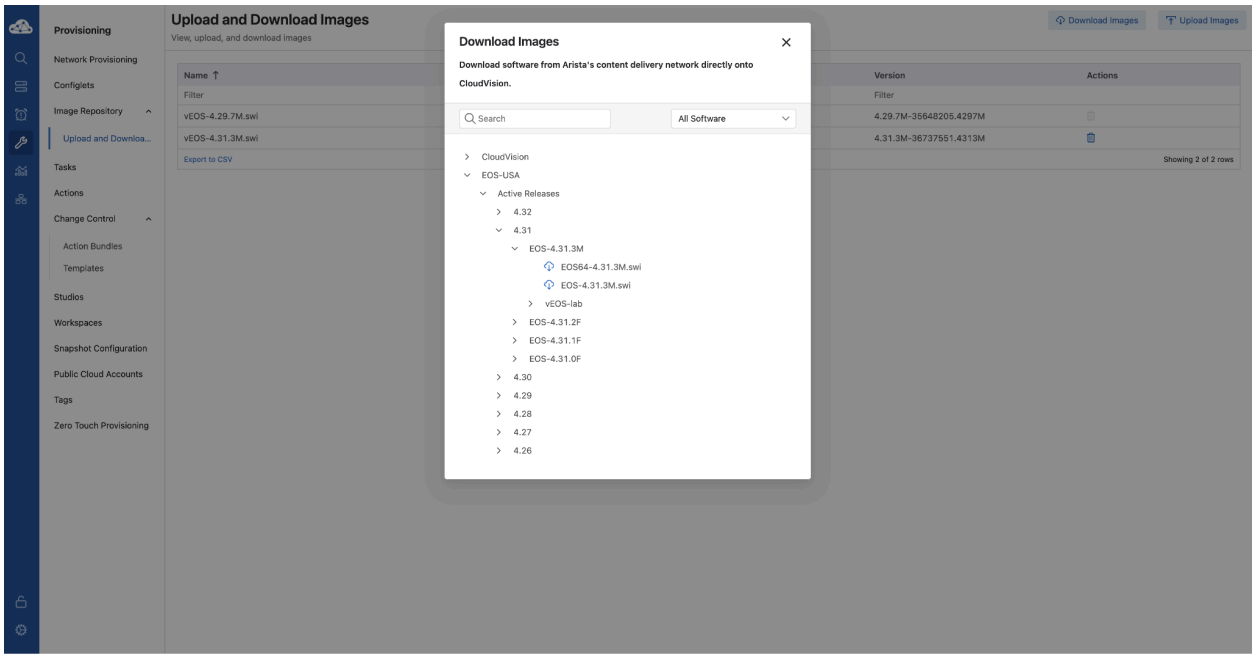
1. Let's navigate to **Settings** > **General Settings**
2. The General Settings page has a section called "**Features**" On the right side of your screen. Let's enable the "**Upload and Download Images and Extensions**" feature.




3. Now let's Navigate to **Provisioning** > **Image Repository** > **Upload and Download Images**



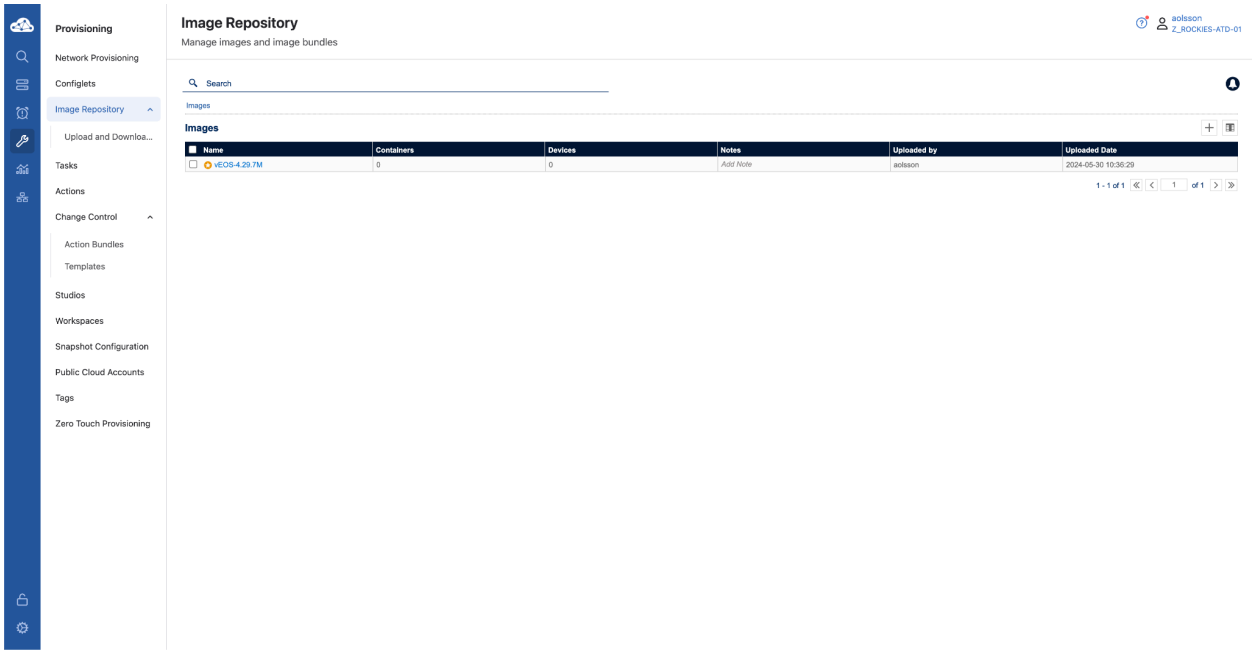
4. In the upper right corner, select “**Download Images**”
5. A pop-up window will appear showing the available images to be downloaded.
6. Expand **EOS-USA** > **Active Releases** > **4.31** > **EOS-4.31.3M**



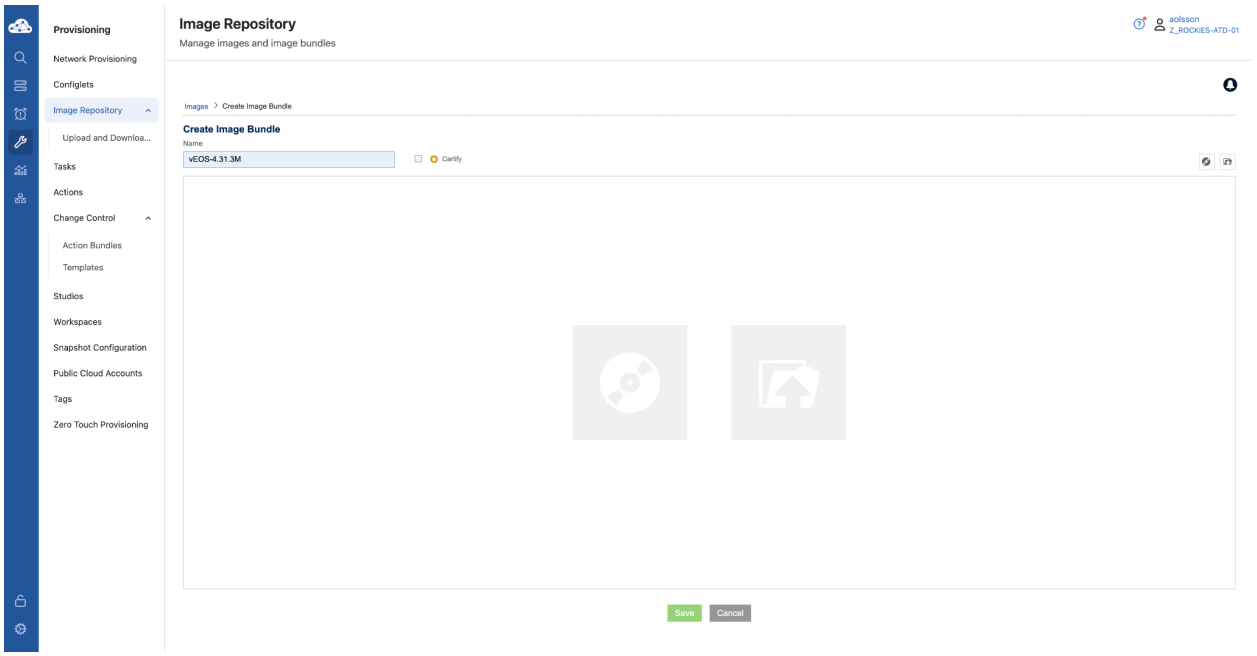

7. Typically, you would click on the blue cloud icon to download an image, but we don’t need to download any images for this lab.

**NOTE**: The images required for this lab are unique and have already been uploaded to CloudVision. **Please use the provided images in the steps going forward!**

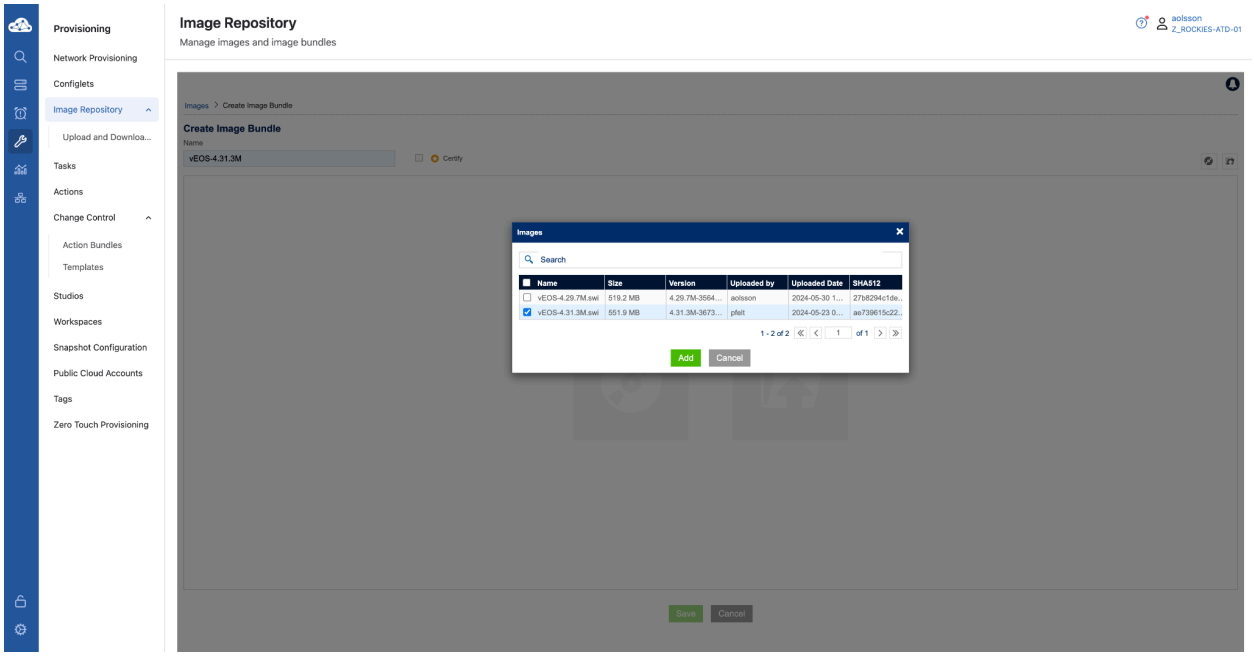
8. Now that the image is downloaded, we need to create the Image Bundle that can be assigned to devices. Head to **Provisioning** > **Image Repository**.
9. Click on the “**+**” icon near the top of the right side of the screen.



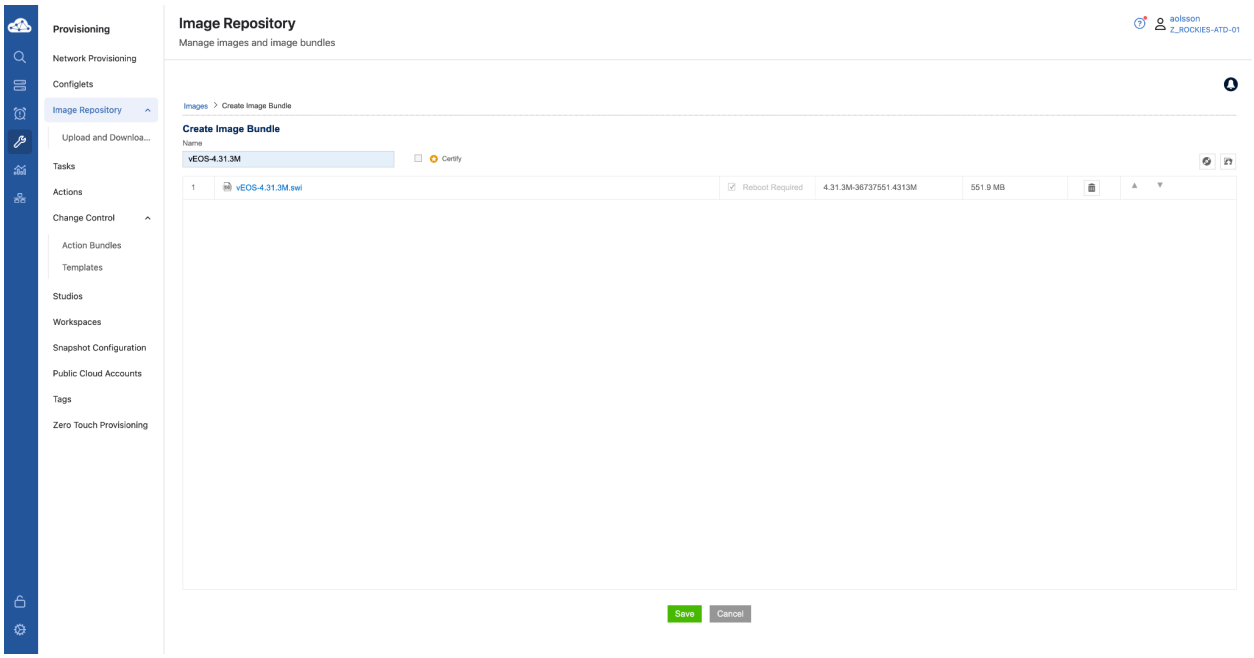

10. In the "**Name**" field, enter a meaningful name for the image. For this lab, we'll use "**vEOS-4.31.3M**".



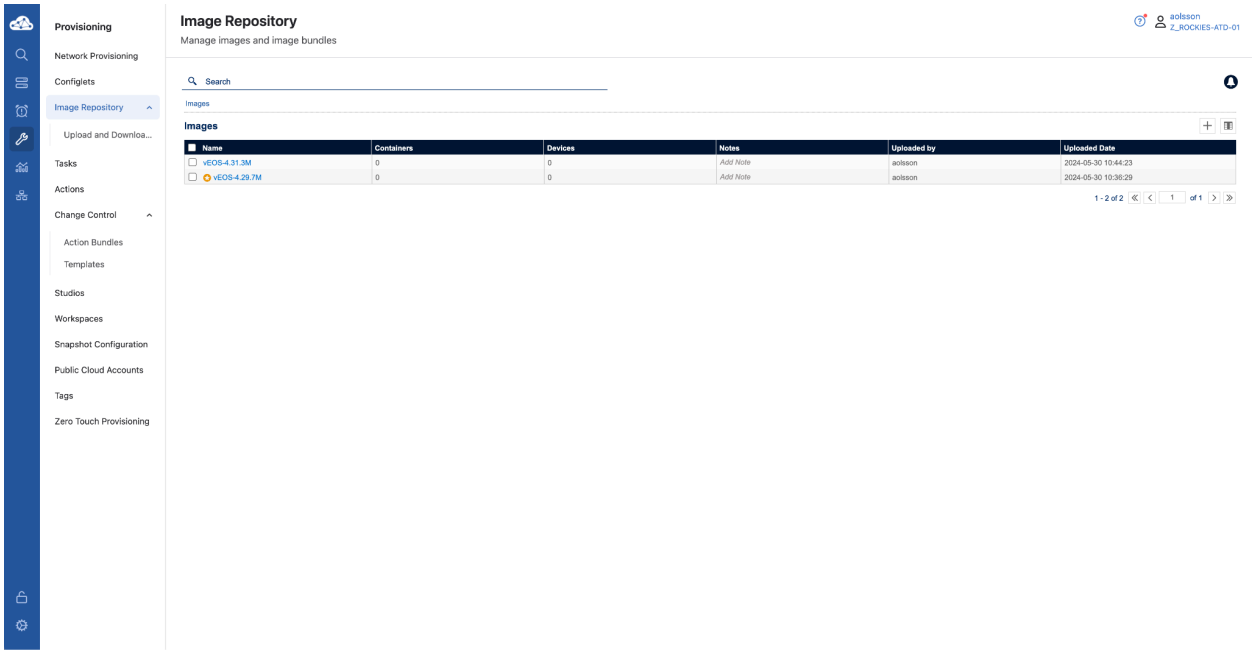

11. To add the Image to the Image Bundle, you can select either the large disk icon in the center of the page or the smaller disk icon on the upper right side of the screen.
12. An "**Images**" dialog box will appear. Select **vEOS-4.31.3M.swi** and then select the "**Add**" button.




13. You'll be returned to the "**Create Image Bundle**" page and should see the Image you just added. Click the "**Save**" button to save the image bundle.



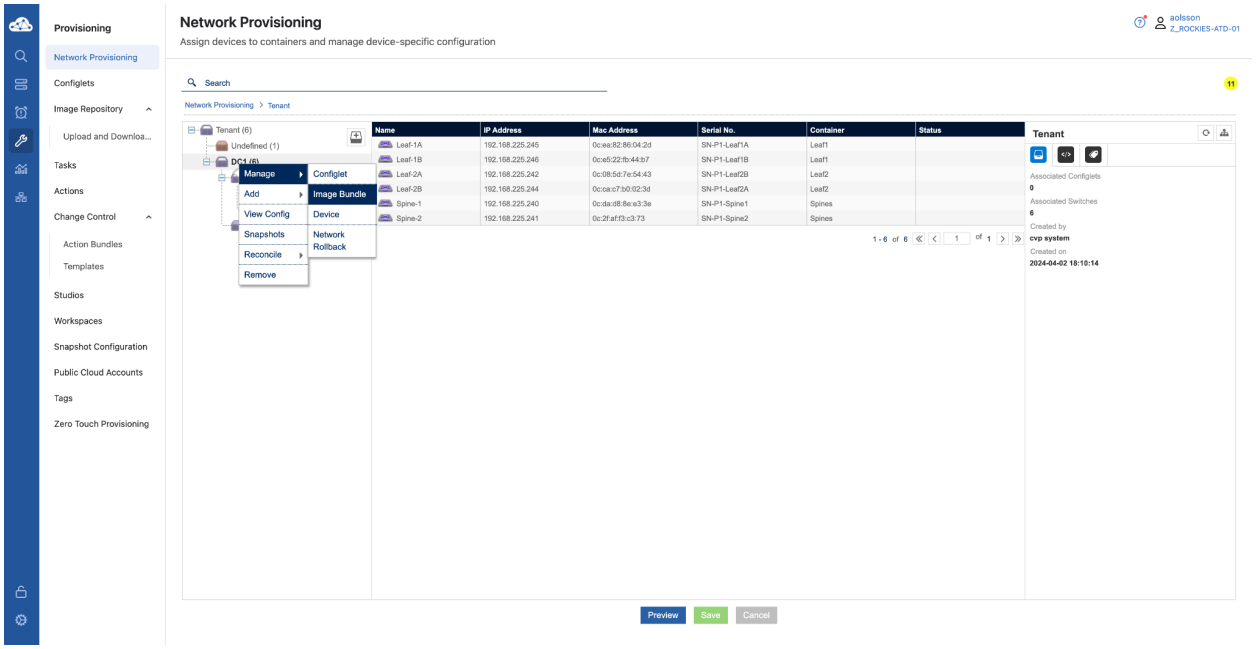

14. The **vEOS-4.31.3M** image bundle will now appear in the Image Repository



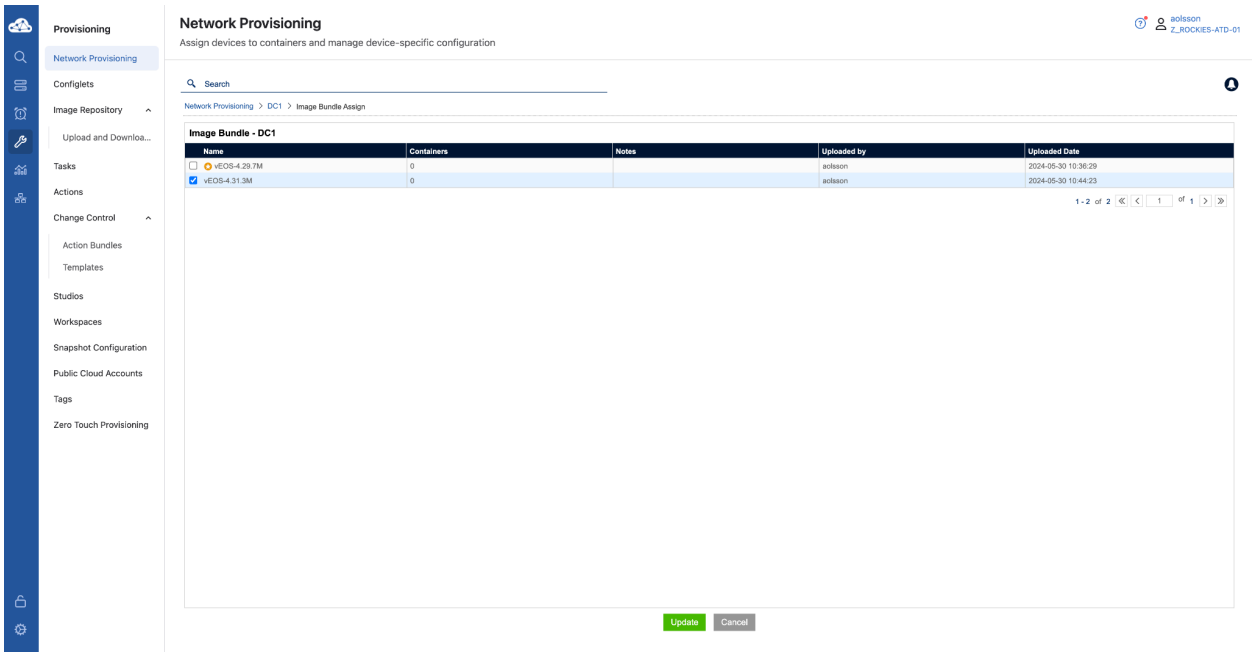

# Image Assignment

Now that an Image Bundle has been created, we can assign it to the devices in the topology.

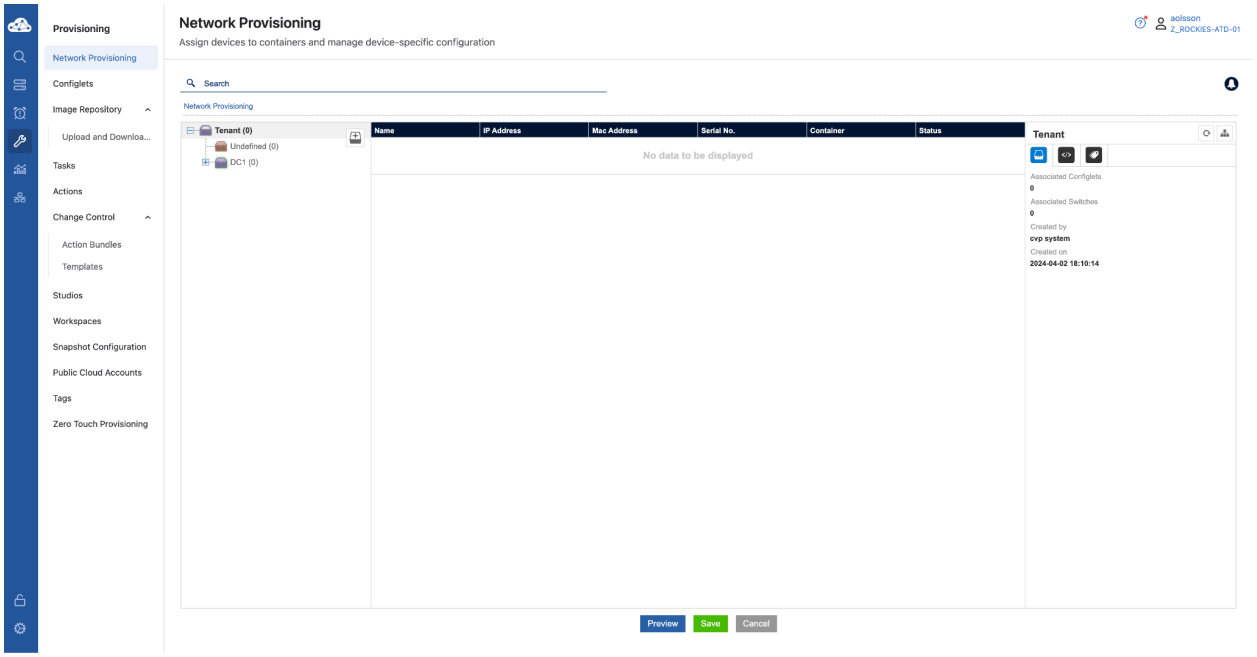
1. Let's start by navigating to **Provisioning** > **Network Provisioning**
2. Right-click on the "**DC1**" container and select **Manage** > **Image Bundle**



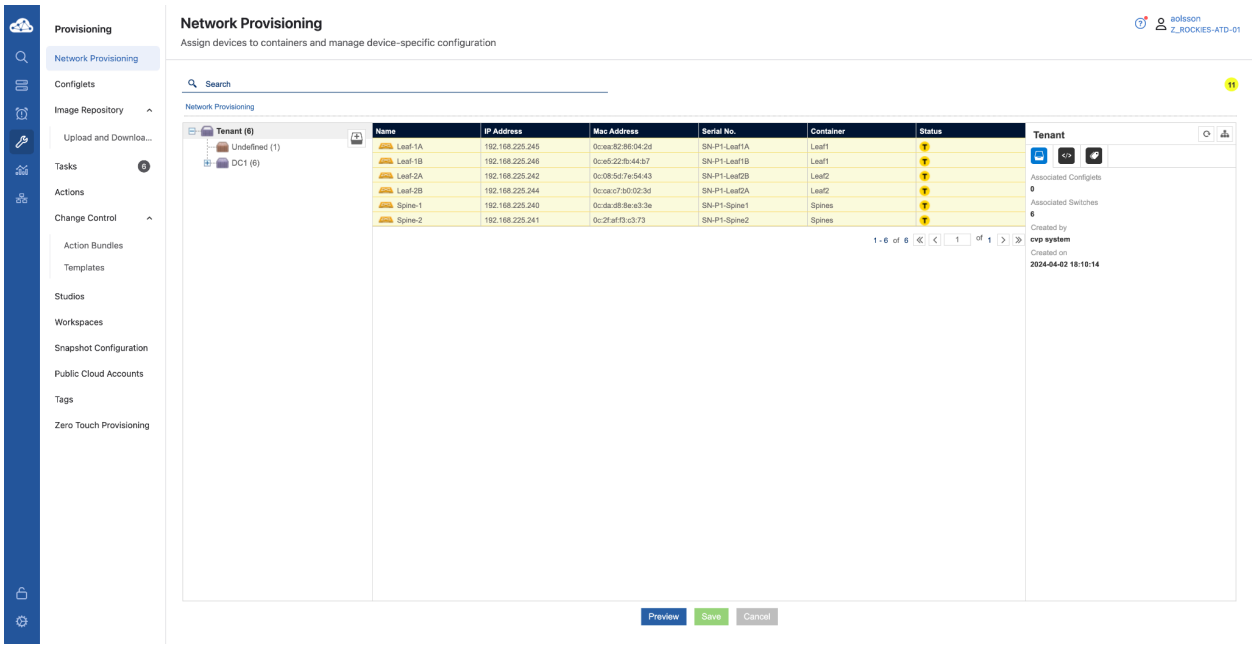
3. Select the one created in the previous step, **vEOS-4.31.3M**, from the available image bundles.
4. Select the "**Update**" button at the bottom of the screen to assign the Image Bundle to all devices in the DC1 container.



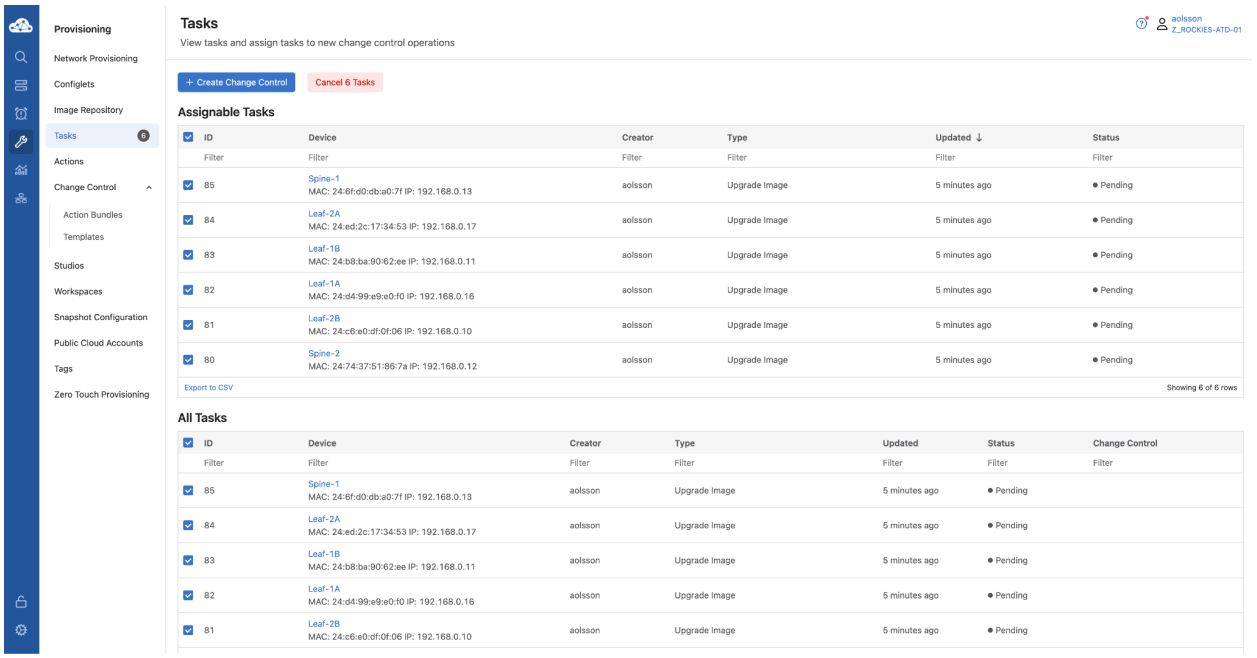

5. You will now see a green hue surrounding the "**DC1**" container and the switches under the "**DC1**" hierarchy, indicating that changes must be saved. Click the "**Save**" button at the bottom of the screen to save the image assignment changes.



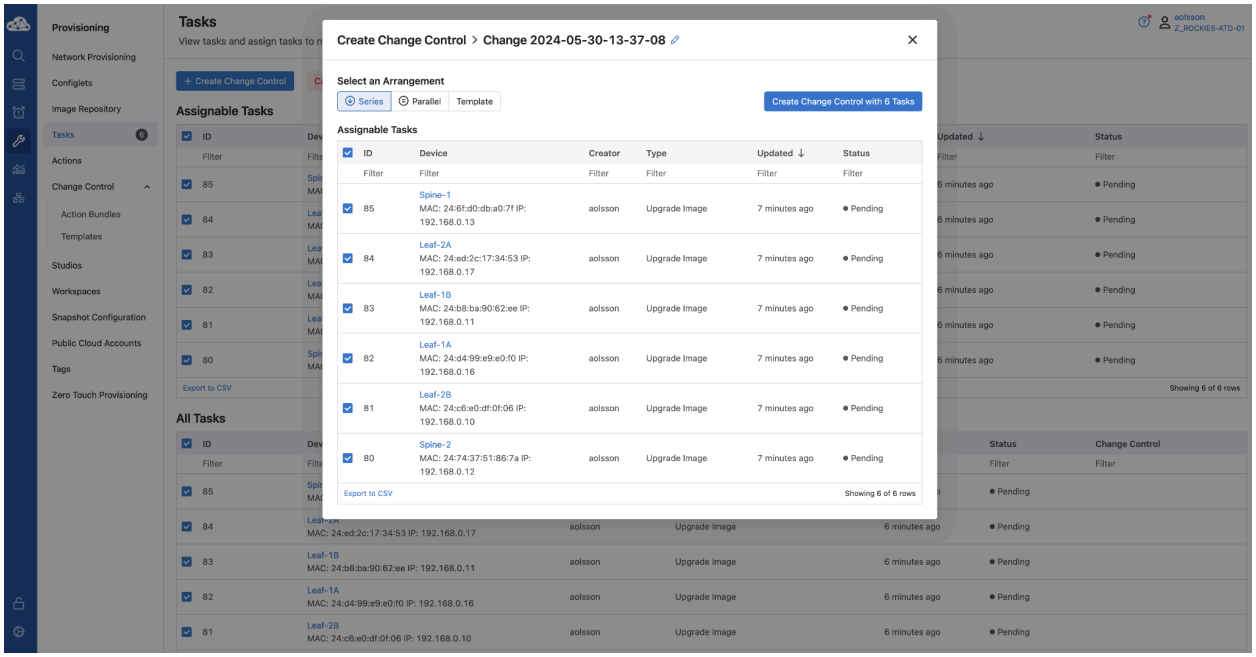
6. You'll now see the switches are all shown in yellow with a "**T**" in the status section, indicating that these devices have available tasks.



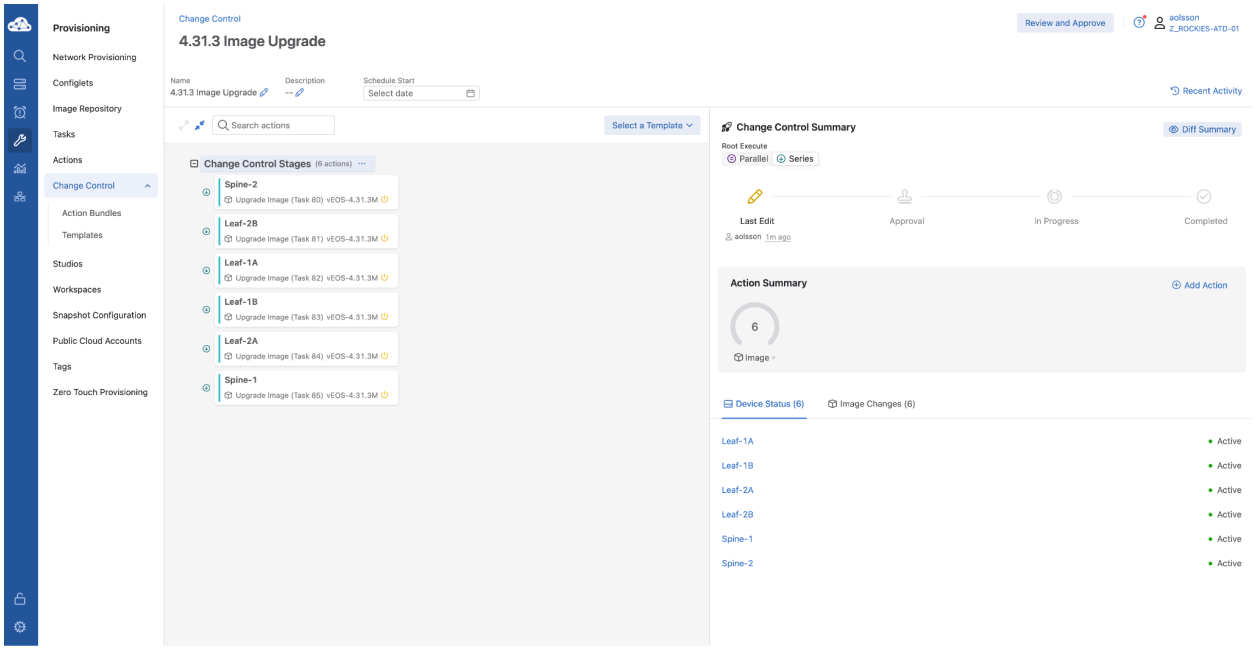

7. On the left side of the screen, select "**Tasks**." You should see 6 pending tasks, one for each device in the topology.
8. Select the 6 "**Upgrade Image**" tasks in the "**Assignable Tasks**" section.



9. Now, we need to create a change control with those tasks, so select the "**+ Create Change Control**" button at the top of the screen.
10. The "**Create Change Control**" screen should pop up. Ensure all 6 tasks are selected and that "**Series**" is chosen. Then, choose "**Create Change Control with 6 Tasks**" on the top left of the pop-up.




11. You'll then be taken to the change control screen. Let's continue by giving the change control a meaningful name. At the top of the change control where it says "**Name,**" click the little pencil icon and enter "**4.31.3 Image Upgrade**" and press enter.




As you can see in the change control, every device in our topology will be upgraded in series. While that might look good, it's not an optimal upgrade process for our environment. For example, how do we know that MLAG Health is in a good state so we don't have an outage during the upgrade? How do we know that the MLAG timers have been completed and the switch is back online before taking down its peer for the upgrade?  Let's leave the change control as is for now and return to it once we've completed the following steps.

# ACTION BUNDLES

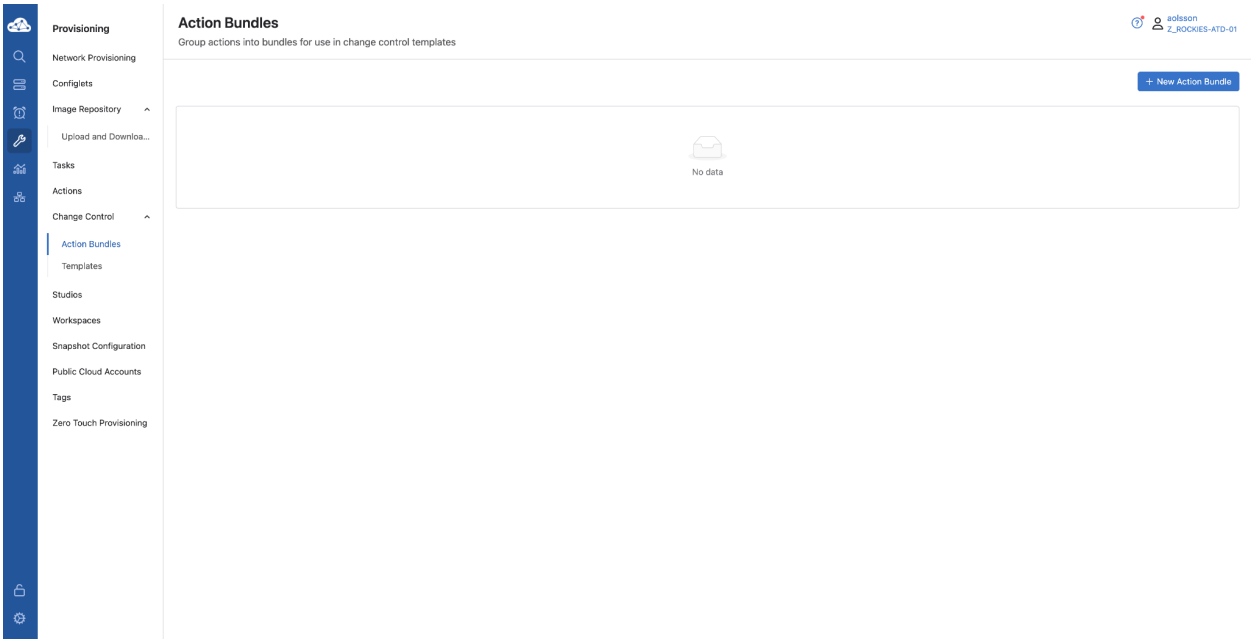
An action bundle is a collection of actions that are applied to the stage rule(s) of a change control template. The template is then used in a change control action, to organize the tasks into logical steps as administratively defined.where the actions contained in its action bundles are executed

Each action bundle can contain up to one task action and an unlimited number of non-task actions. You can apply the same action bundle to multiple change control templates.

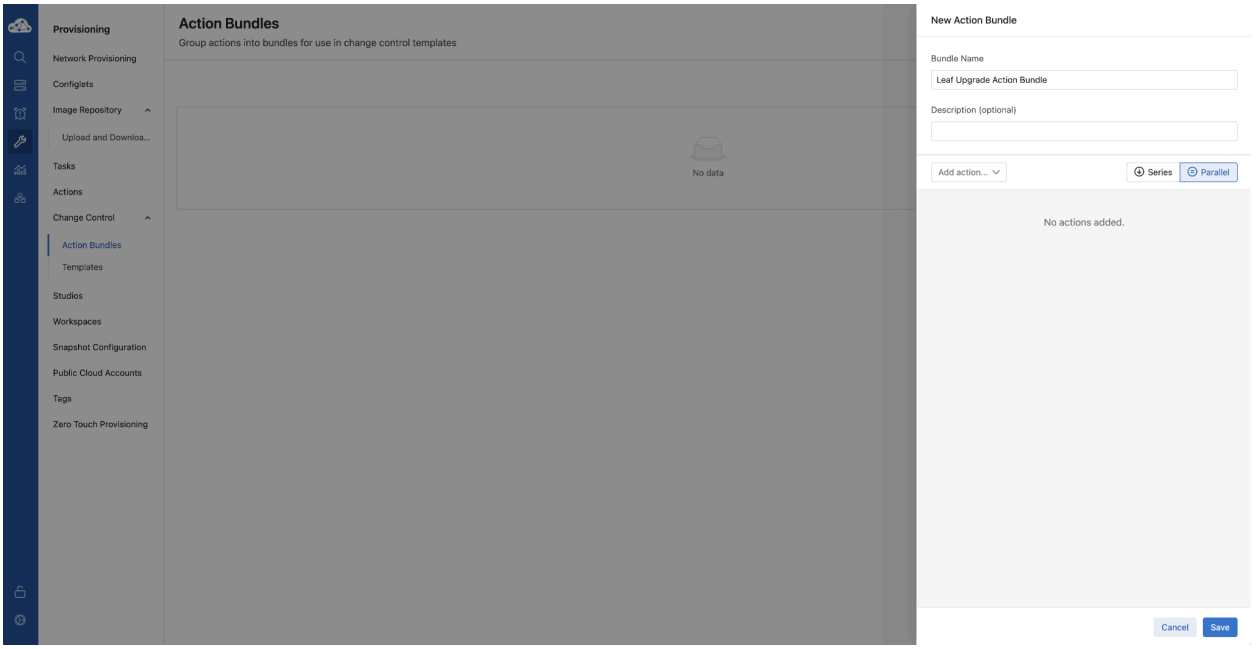
## Create an Action Bundle for the Leafs

Let's create a new action bundle for our Leafs that can help us do MLAG validation as part of the upgrade process.

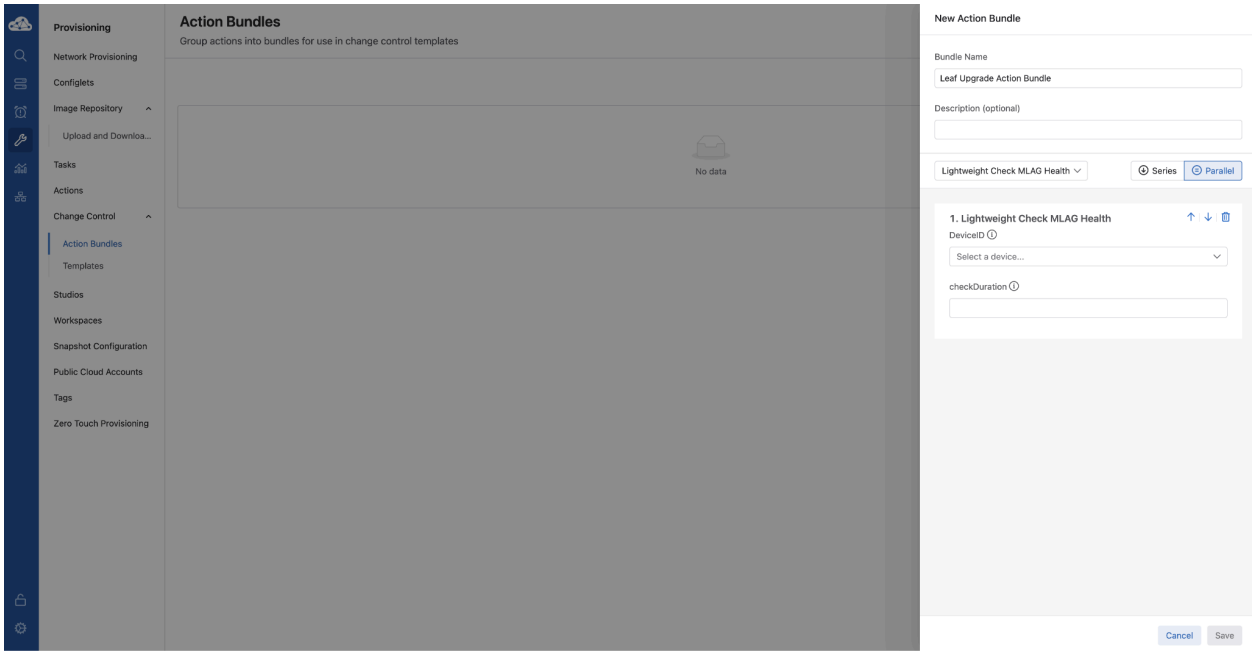
1. Start by navigating to **Provisioning** > **Action Bundles**.
2. Select “**New Action Bundle**”.



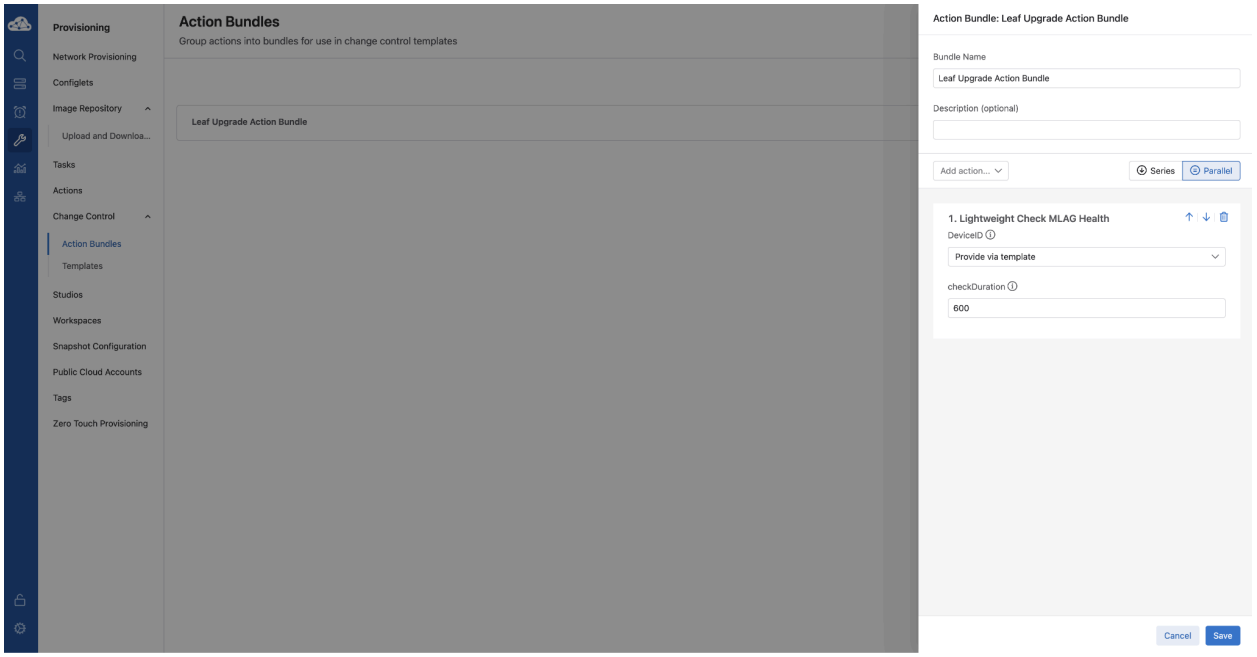

3. In the “**Bundle Name**” field, enter a meaningful description for this action bundle. This Action Bundle will be used for Leaf upgrades, so we’ll use the name “**Leaf Upgrade Action Bundle.**”




4. Select the “**Add action**” dropdown box and choose the “**Lightweight Check MLAG Health**” action.

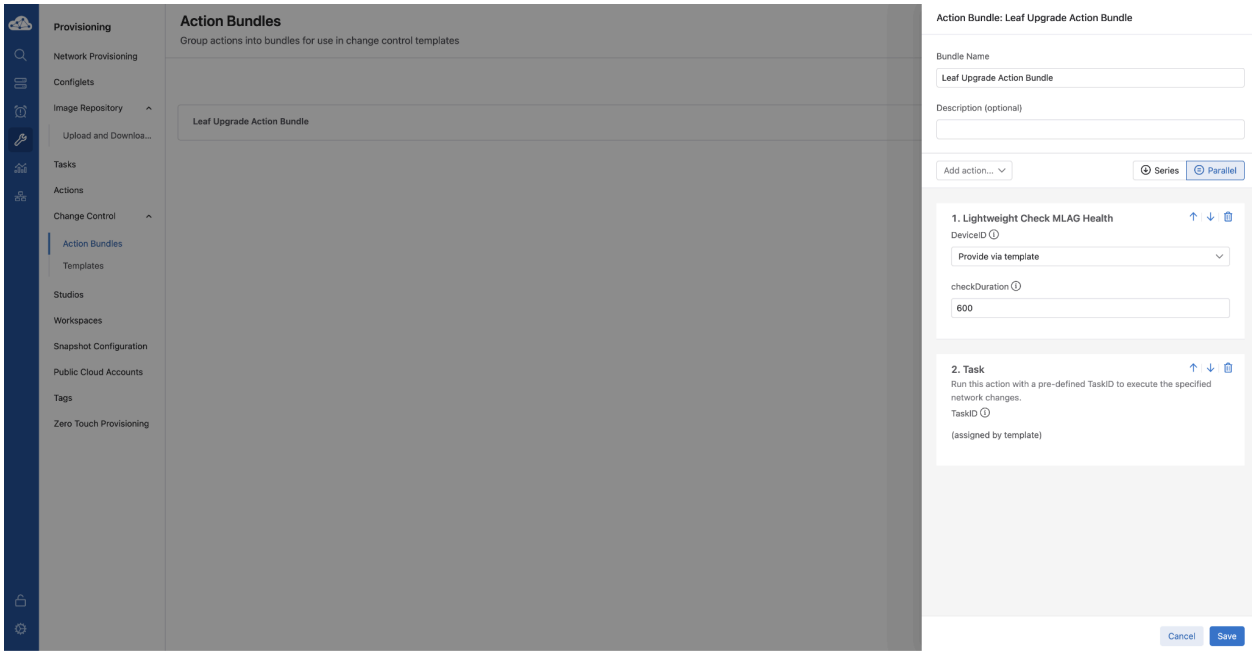



5. In the "**DeviceID**" section, choose the "**Provide via template**" option and set the "**checkDuration**" to 600 (5 Minutes).

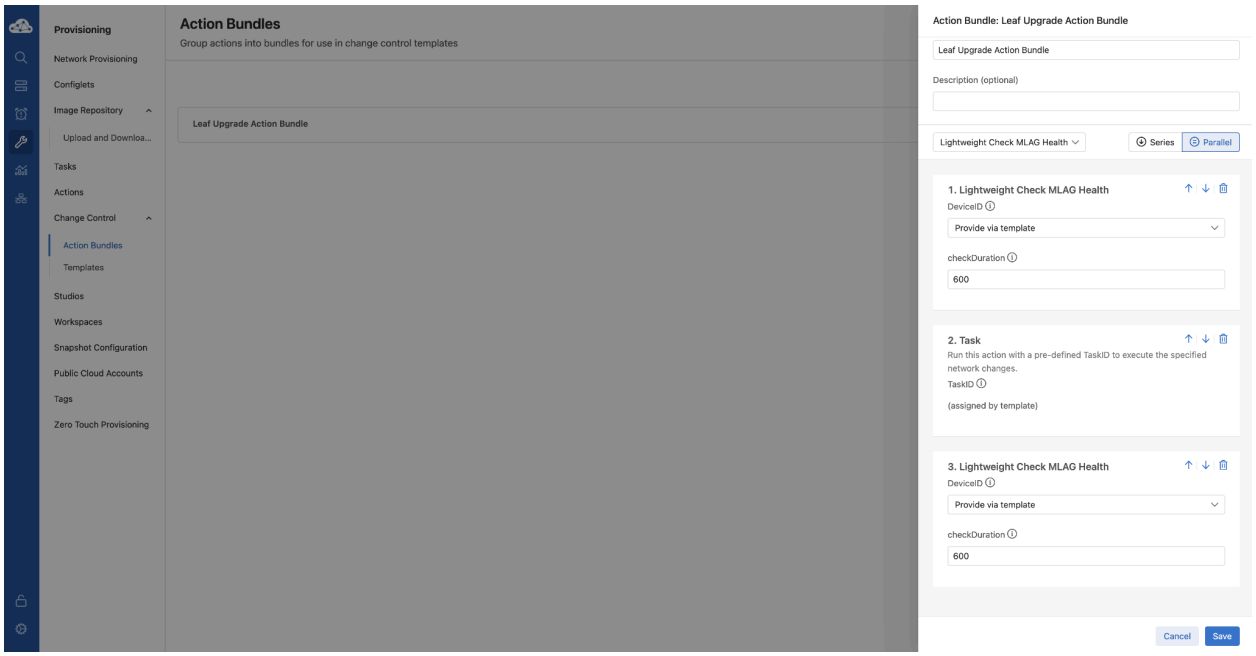



**NOTE**: The "**Add action**" dropdown may stay populated with the name of the last action that was chosen

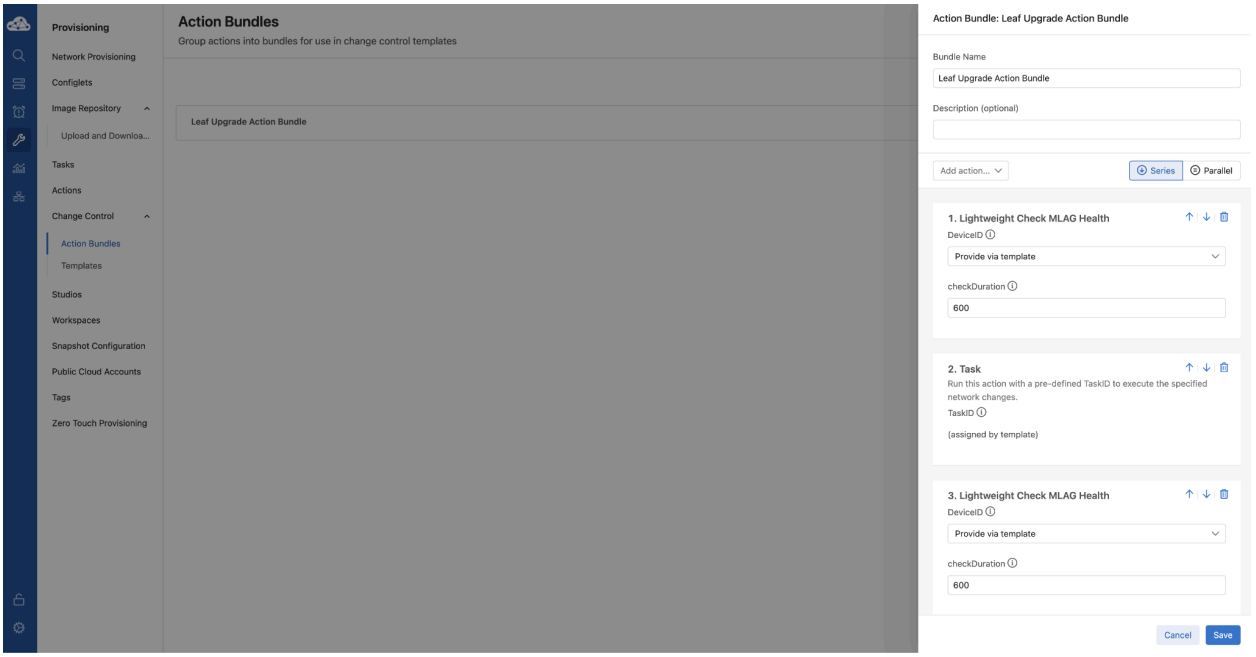
6. Select the “**Add action**” dropdown box and select the “**Execute Task**” action.




7. Select the “**Add action**” dropdown box again and select the “**Lightweight Check MLAG Health**” action.
8. In the “**DeviceID**” section, choose the “**Provide via template**” option and set the “**checkDuration**” to 600 (5 Minutes).



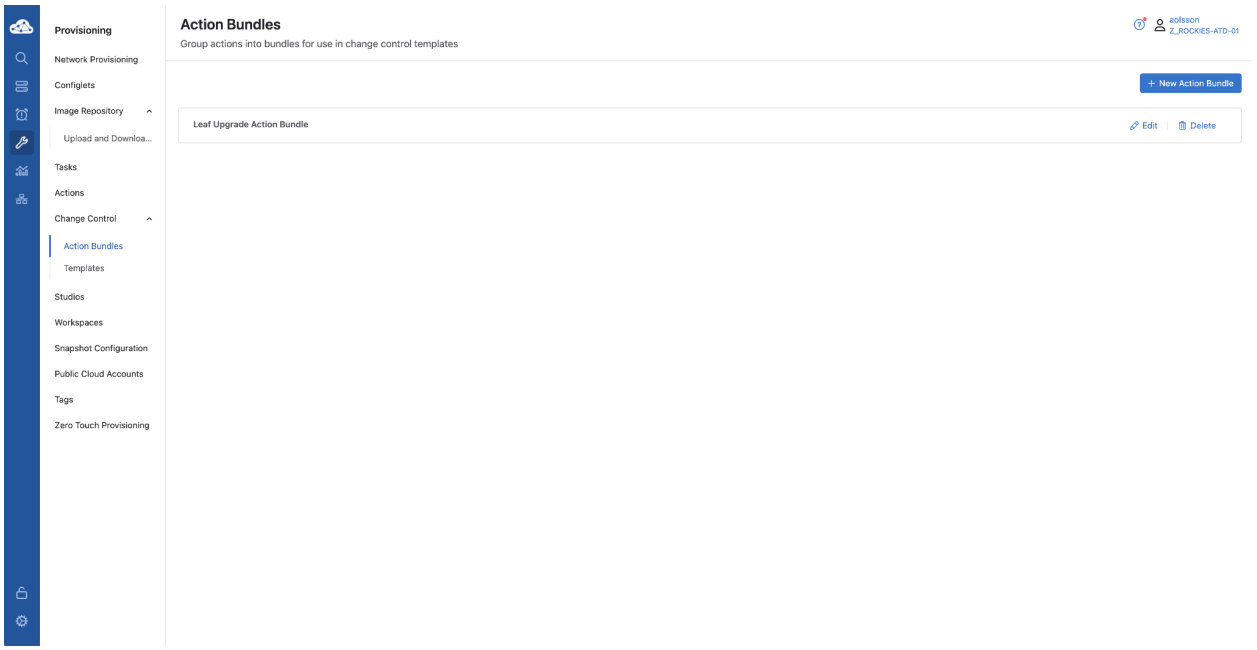
9. Select the “**Series**” option so that the actions in this Action Bundle are in series.
10. Select “**Save**” once the Action Bundle looks like the one in the screenshot below.



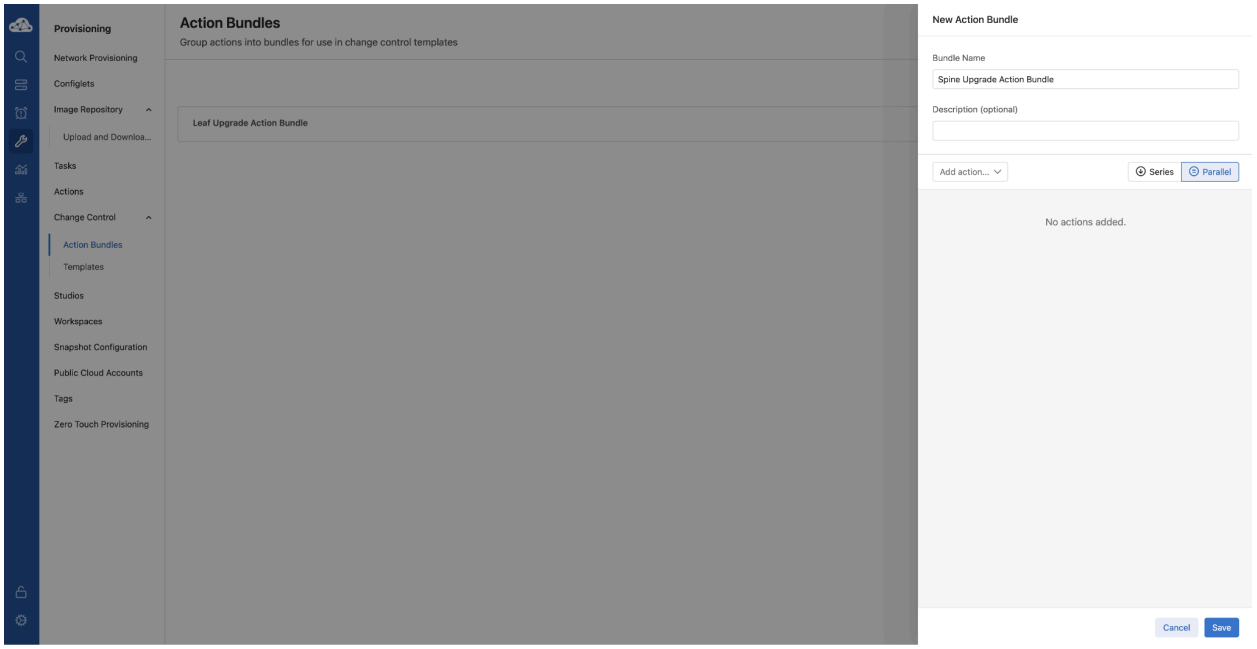

## Create an Action Bundle for the Spines

Now, let's repeat the process we just went through with a few changes tailored to the Spines. Since the Spines aren’t in an MLAG pair, we don’t need to do the MLAG Health Check, but we can use BGP Maintenance Mode since all the peers are running BGP and support the BGP GSHUT community.

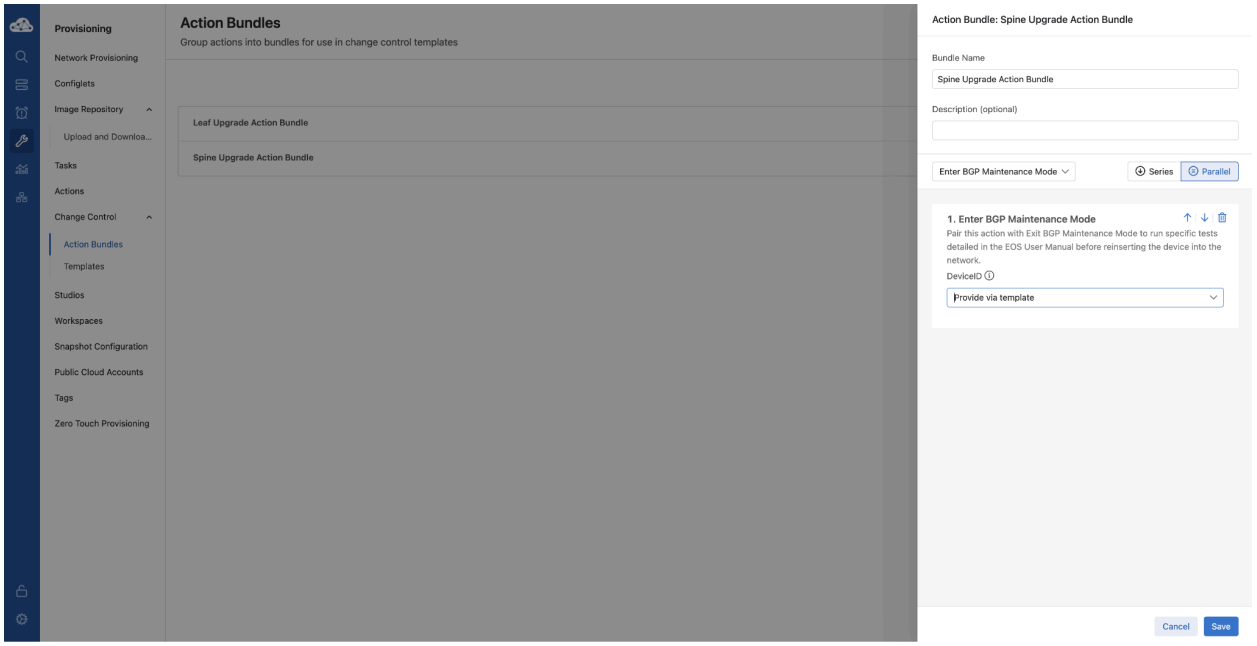
1. Let’s start by creating a new Action Bundle.  Click “**New Action Bundle**” on the top right of the screen.




2. In the "**Bundle Name**" field, enter a meaningful description for this action bundle. This Action Bundle will be used for Spine upgrades, so we'll use the name "**Spine Upgrade Action Bundle**."

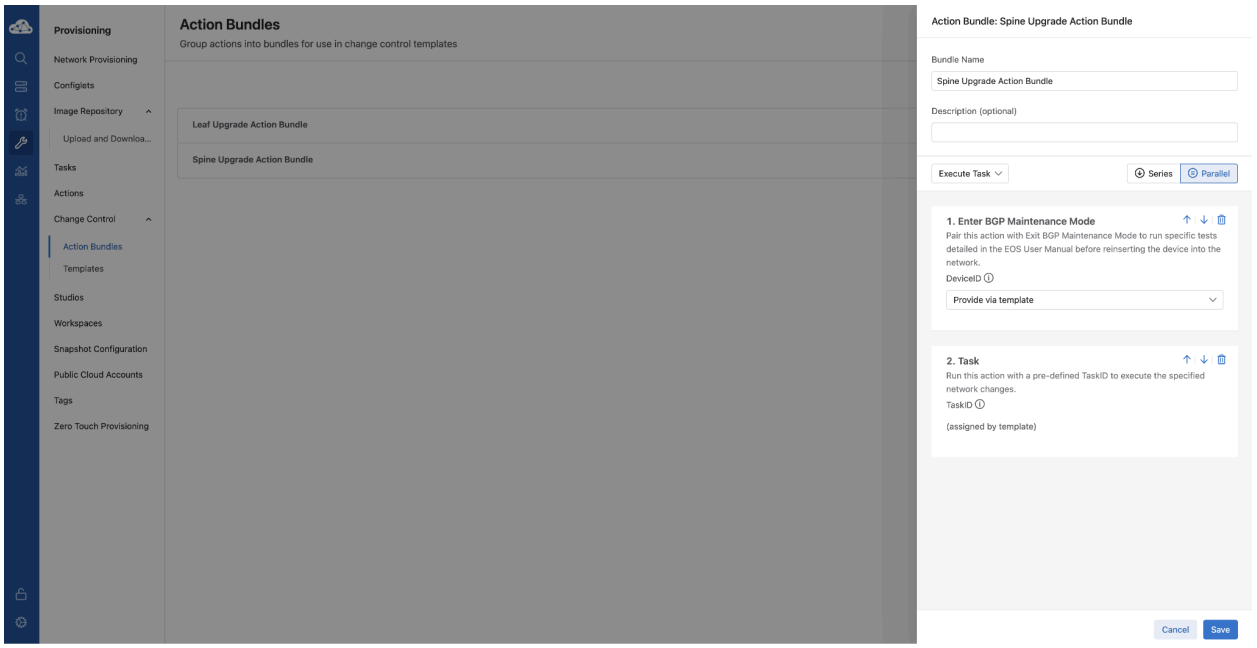



3. Select the "**Add action**" dropdown box and select the "**Enter BGP Maintenance Mode**" action.
4. In the "**DeviceID**" section, choose the "**Provide via template**" option.



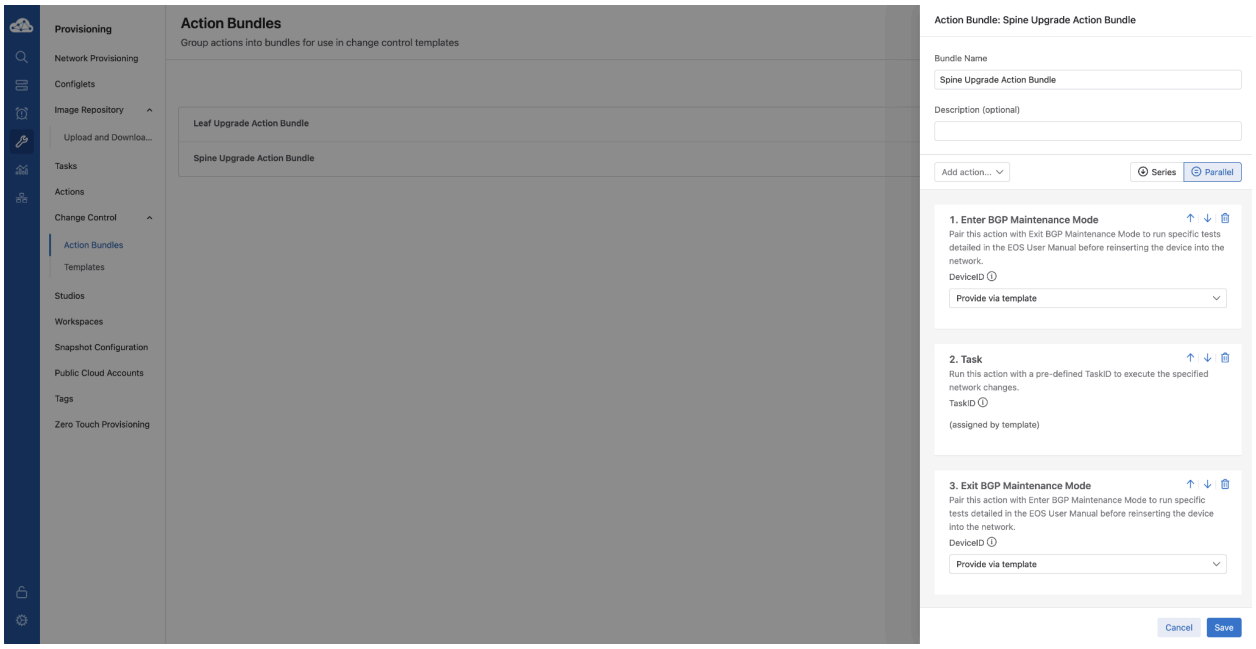

**NOTE**: The "**Add action**" dropdown may stay populated with the name of the last action that was chosen

5. Select the "**Add action**" dropdown box and select the "**Execute Task**" action.

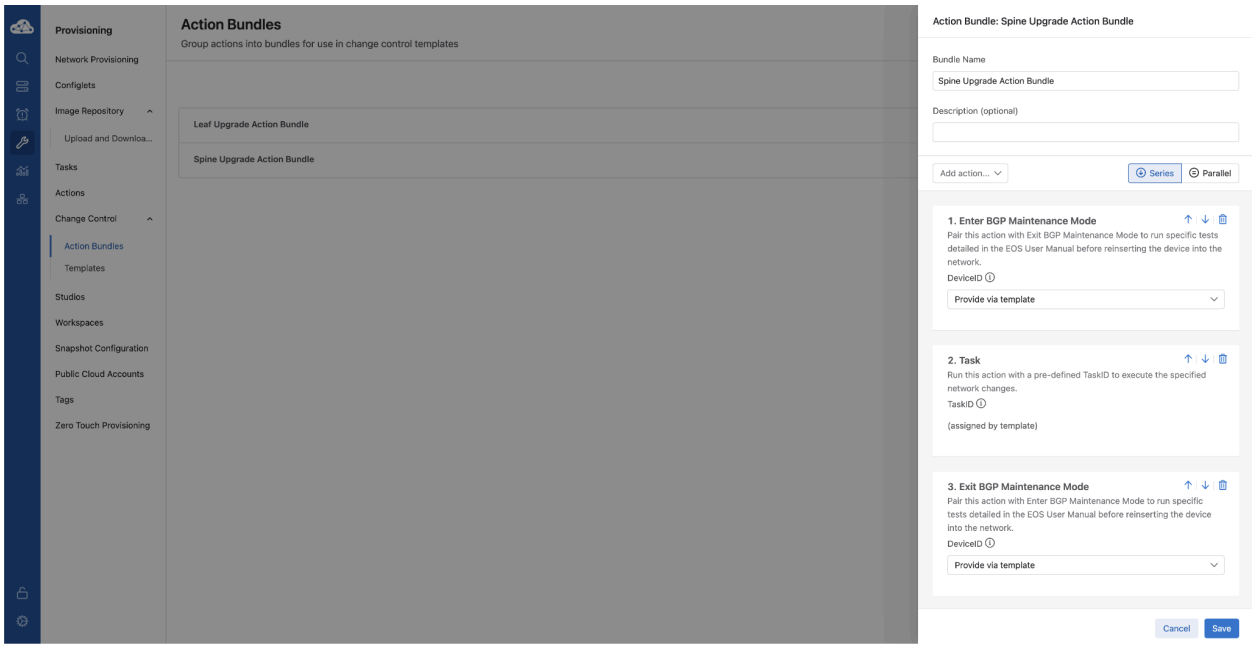



6. Select the "**Add action**" dropdown box again and select the "**Exit BGP Maintenance Mode**" action.

7. In the “**DeviceID**” section, choose the “**Provide via template**” option.



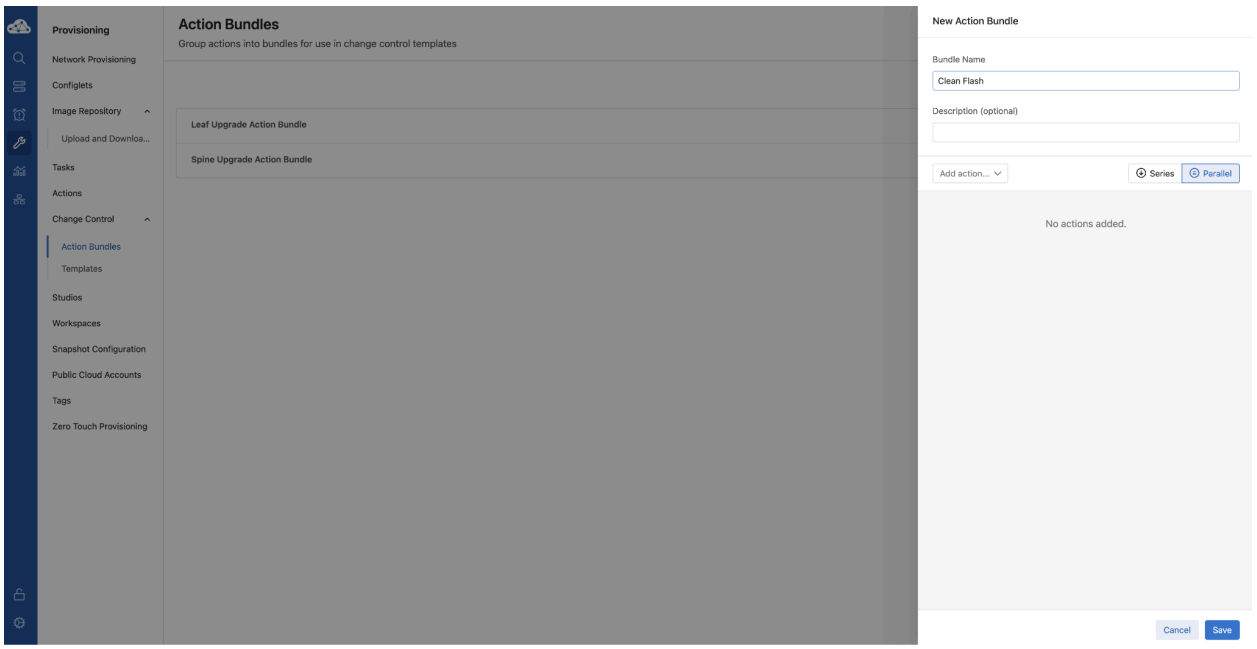

8. Select “**Series**” so that the actions in this Action Bundle are in series.
9. Select “**Save**.”



# Create an Action Bundle to Cleanup the Flash

The final action bundle that we'll create will remove the old images from Flash so we can keep the Flash nice and tidy and not have old images hanging around forever.

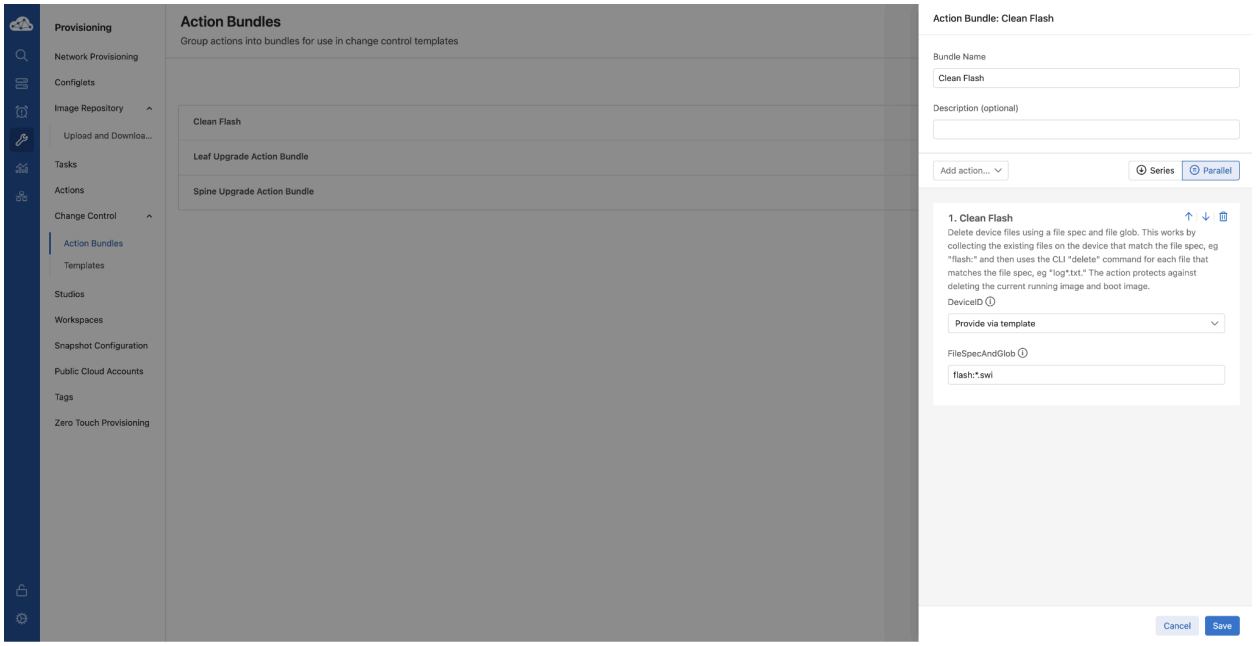
1. Select "**New Action Bundle**"
2. In the "**Bundle Name**" field, enter a meaningful description for this action bundle. This Action Bundle will be used to clean up the flash, so we'll use "**Clean Flash**."



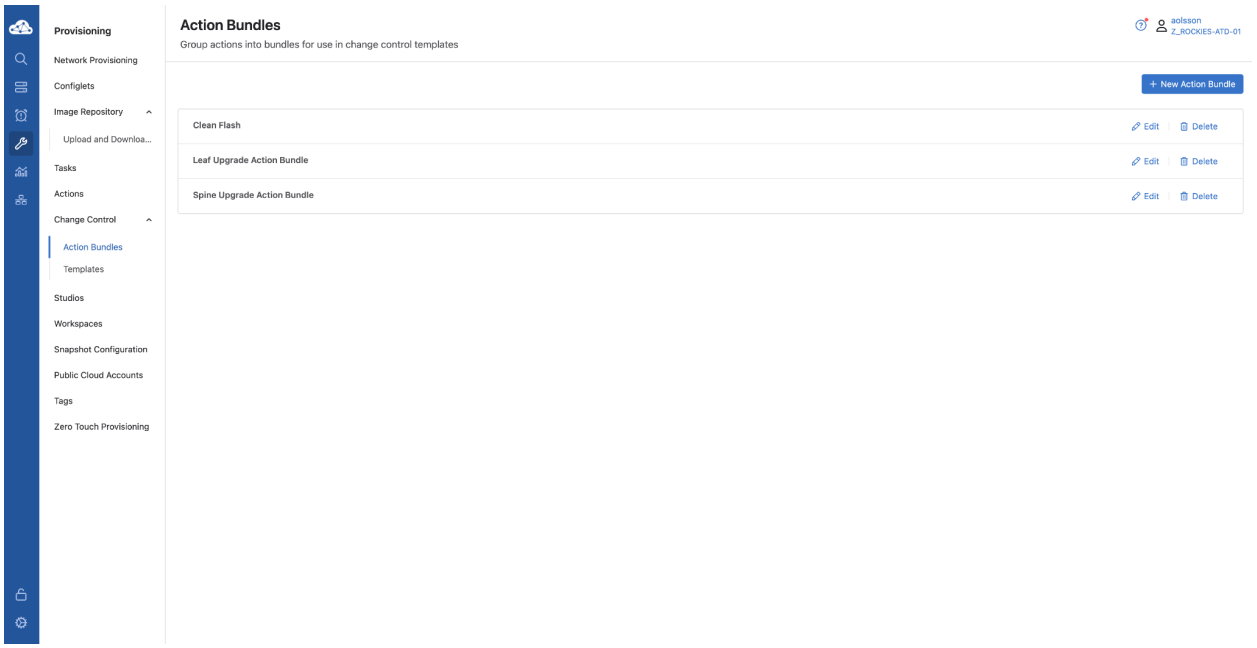

3. Select the "**Add action**" dropdown box and select the "**Clean Flash**" action.
4. In the "**DeviceID**" section, choose the "**Provide via template**" option.
5. In the "**FileSpecAndGlob**" section, enter:

```
flash:*.swi
```

6. Select "**Save**."




7. You should now see the three Action Bundles that have been created.



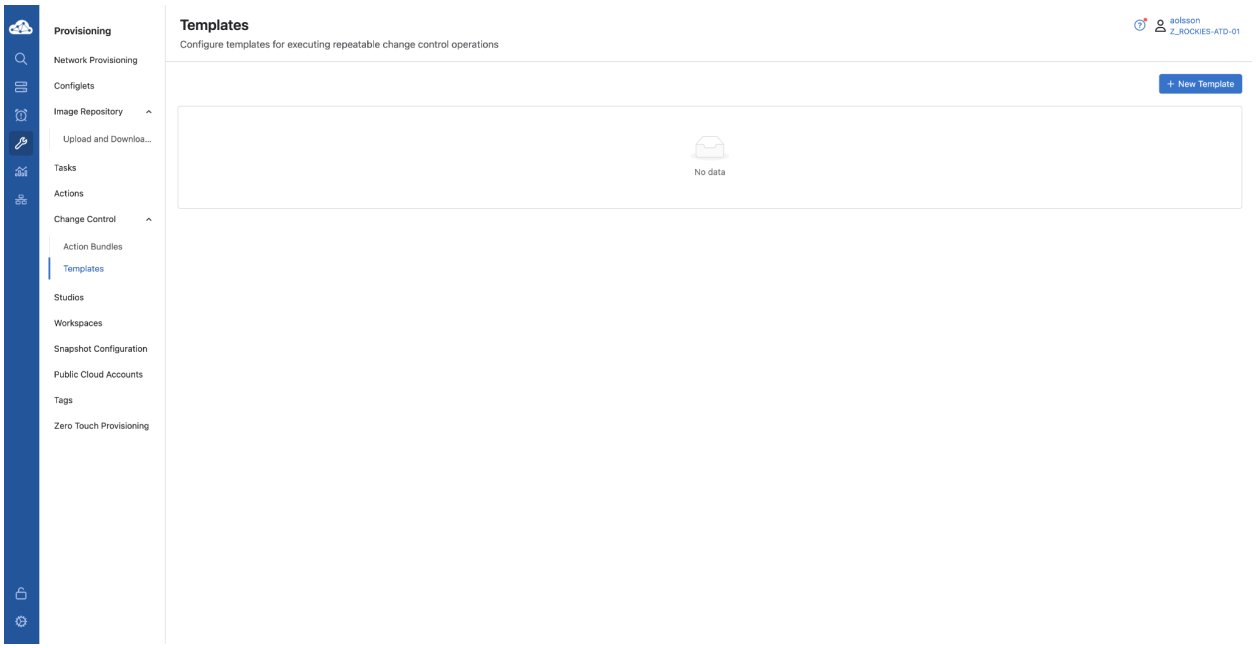
# TEMPLATES

A change control template is used as a structure for repeatable change control operations. It enables you to complete common and frequent changes in your network without needing to configure the details of the change control operation each time.

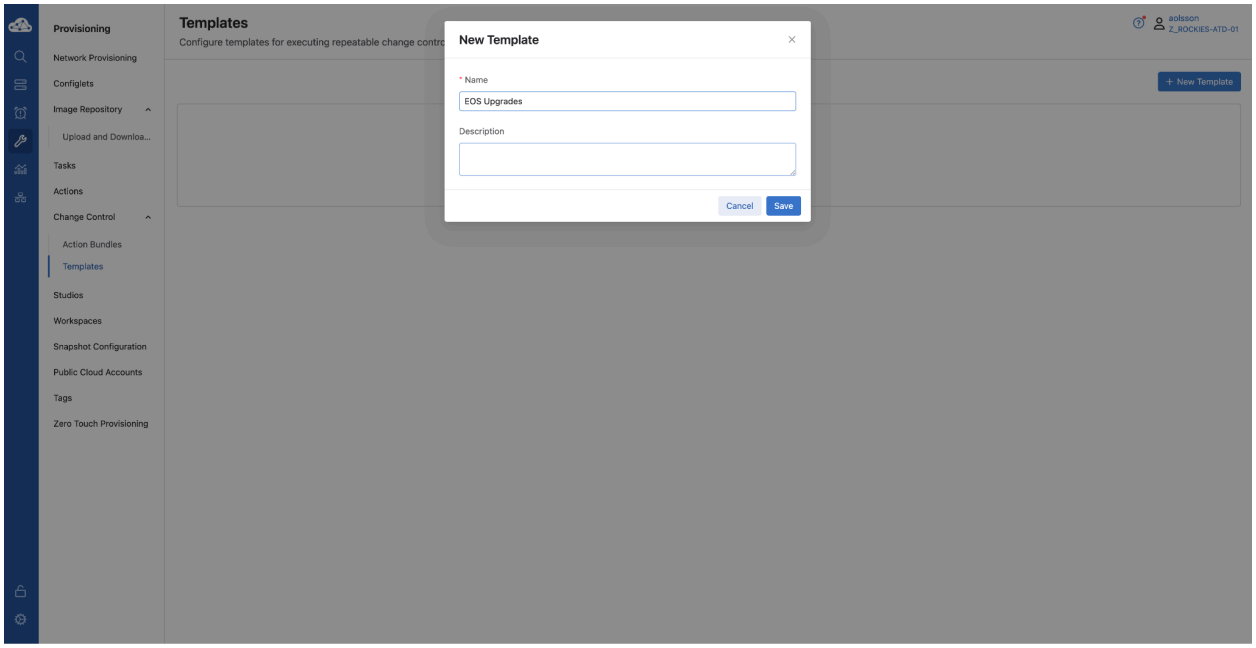
Two elements are used to construct a change control template, action bundles, and templates. Once a template has been created, you can then use it in all future change control operations.

## Create a Template

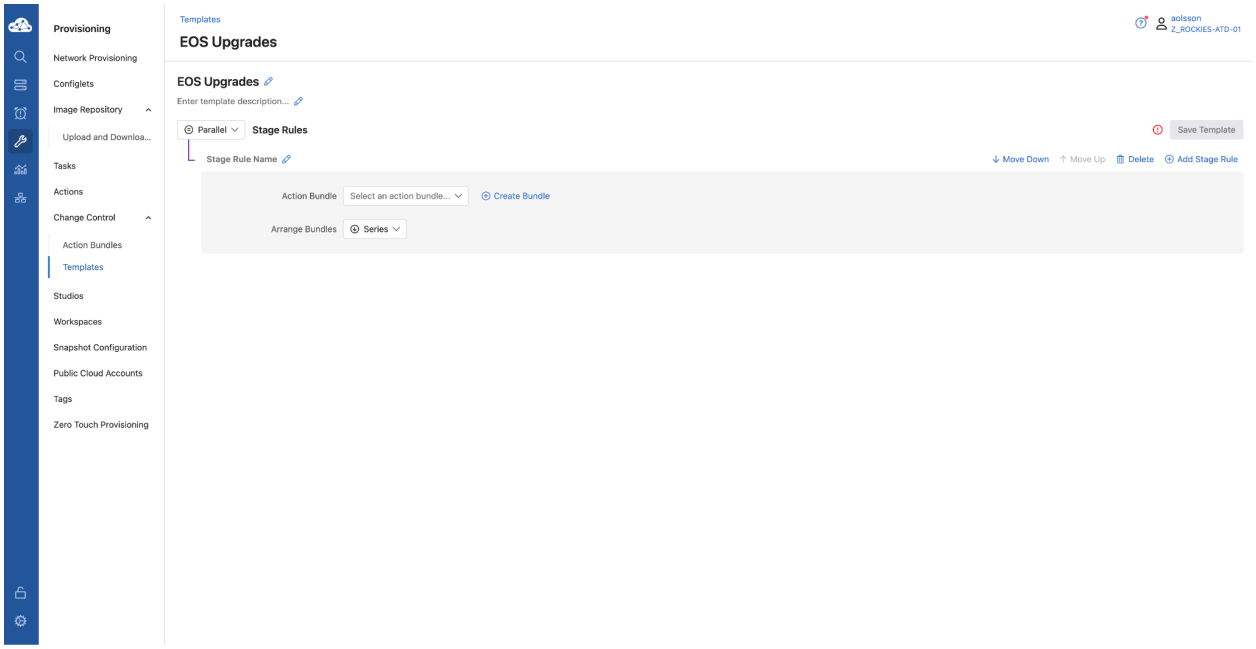
1. The first thing we need to do is navigate to **Provisioning** > **Templates**.
2. Create a new template by selecting "**+ New Template**".




3. The "**New Template**" dialog box should appear. Enter a name for the new template. Since this template will be for upgrades, let's call it "**EOS Upgrades**."
4. Click "**Save**."




5. Next, you'll be taken to a page where you can define the template options.



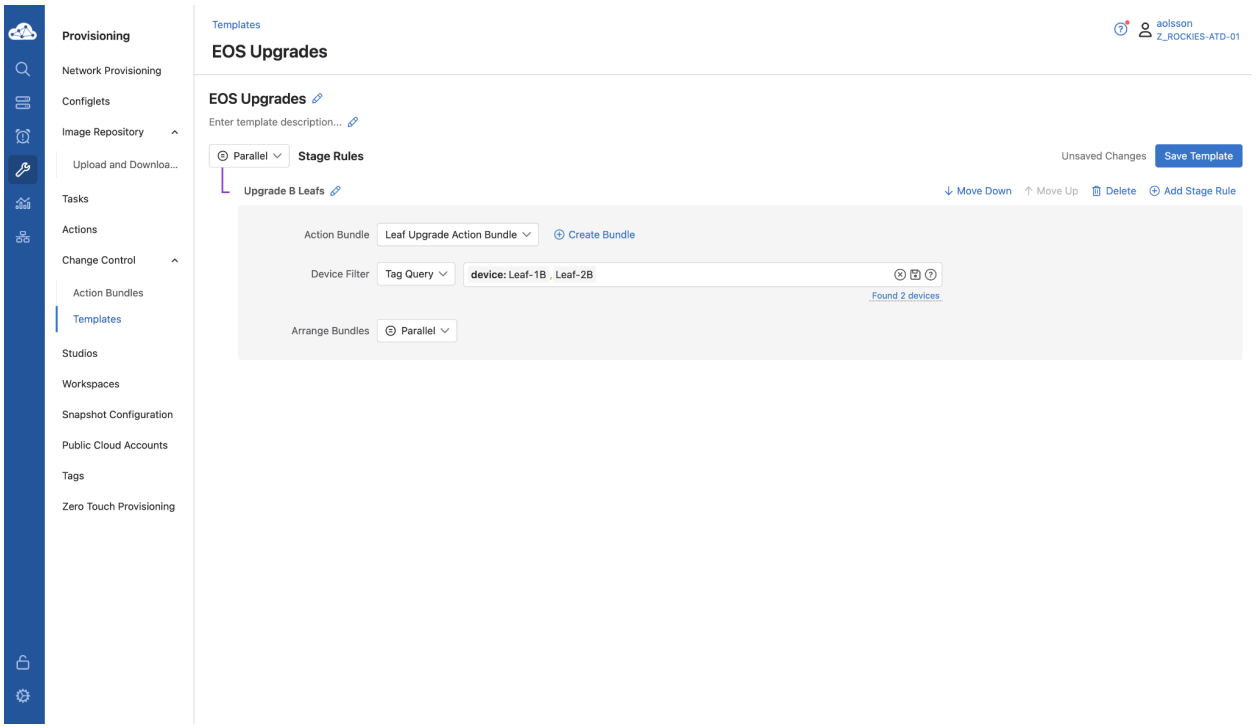

6. Now, let's add the first stage to the Template. On the right side of the page, select "**+ Add Stage Rule**".

**NOTE**: The Stage rules need to match the order they are in the lab guide. Ensure you are clicking the "**Add Stage Rule**" of the last Stage in the Template.

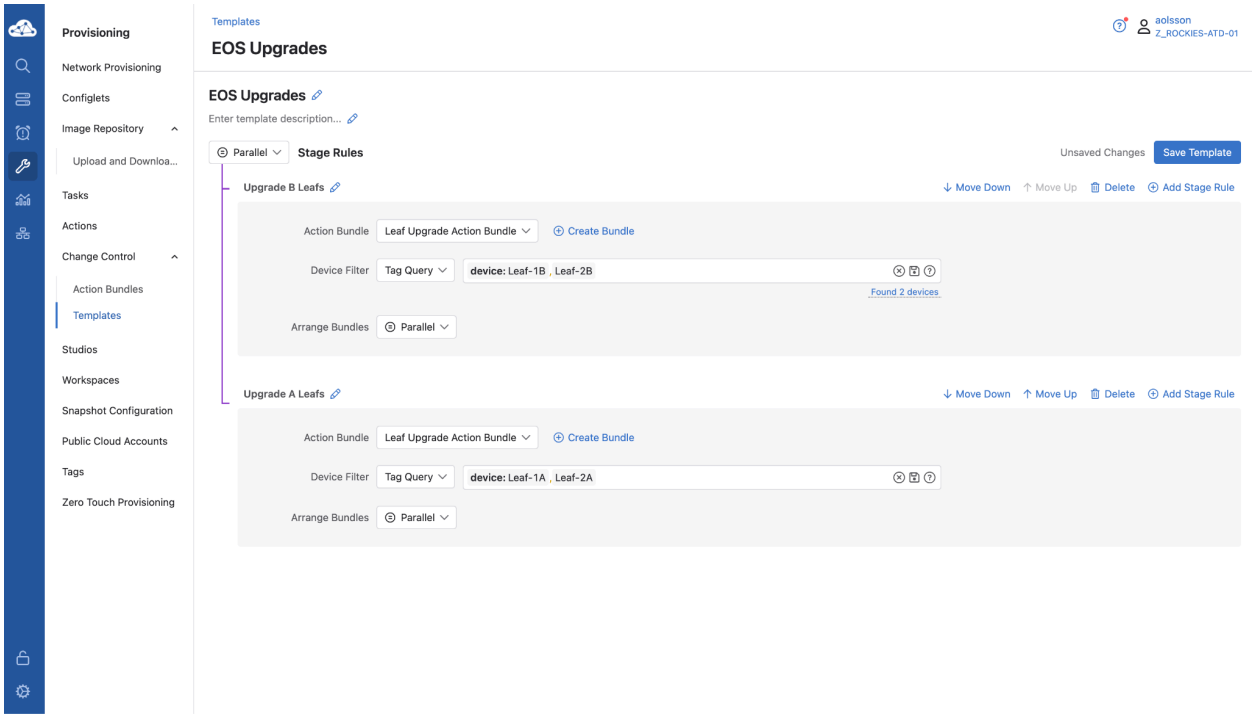
7. Click on the pencil icon next to the "**Stage Rule Name**" of the first stage rule and name it "**Upgrade B Leafs**." As you can probably tell from the name, this stage will only be used to upgrade the B Leafs.
8. Since we'll upgrade the Leafs in this Template stage, choose "**Leaf Upgrade Action Bundle**" in the "**Action Bundle**" dropdown.
9. In the "**Device Filter**" dropdown, change the selection to "**Tag Query**."
10. In the box just to the right of the "**Tag Query**" dropdown, we must select Leaf-1B and Leaf-2B. To do this, type "**Device:**" and choose Leaf-1B from the dropdown followed by a "**,**" then select Leaf-2B.

**NOTE**: Keep in mind that the Tag Key:Value pairs are case sensitive.

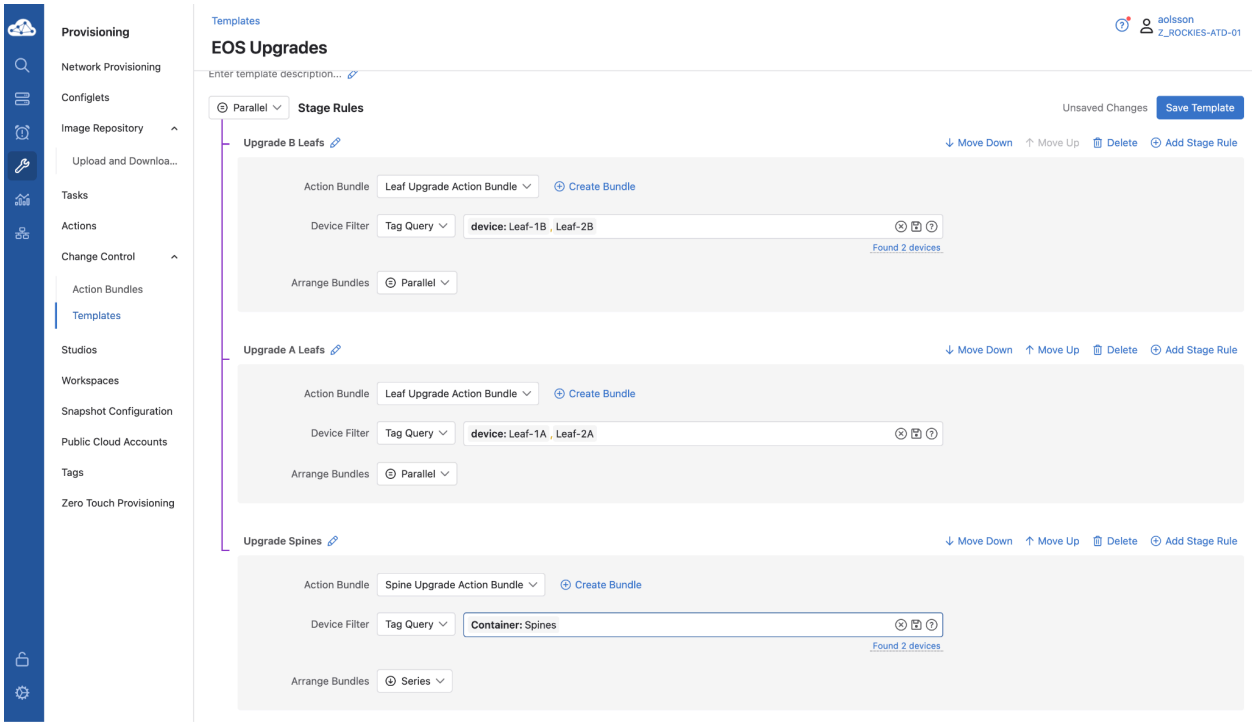
11. Change the "**Arrange Bundles**" dropdown from "**Series**" to "**Parallel**" so that this action can be run on both B switches at the same time.



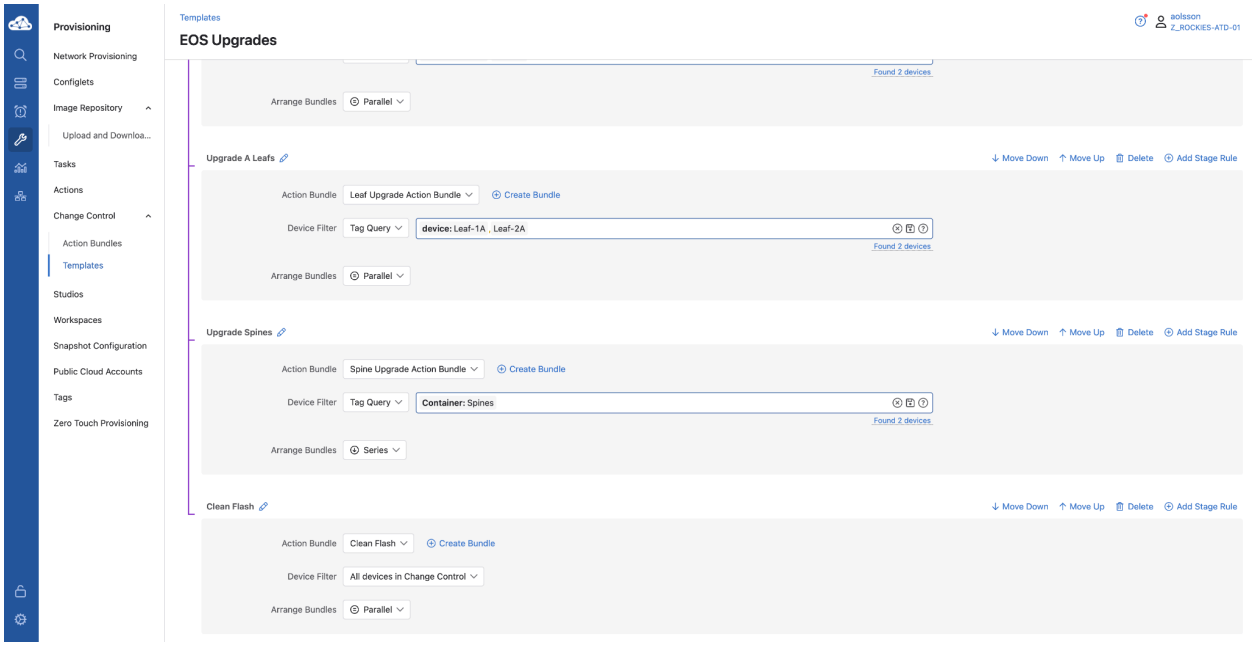

12. Repeat the steps completed for the "**Upgrade B Leafs**" stage, but for "**Upgrade A Leafs**." Be sure to select both Leaf-1A and Leaf-2A in the Tag Query.



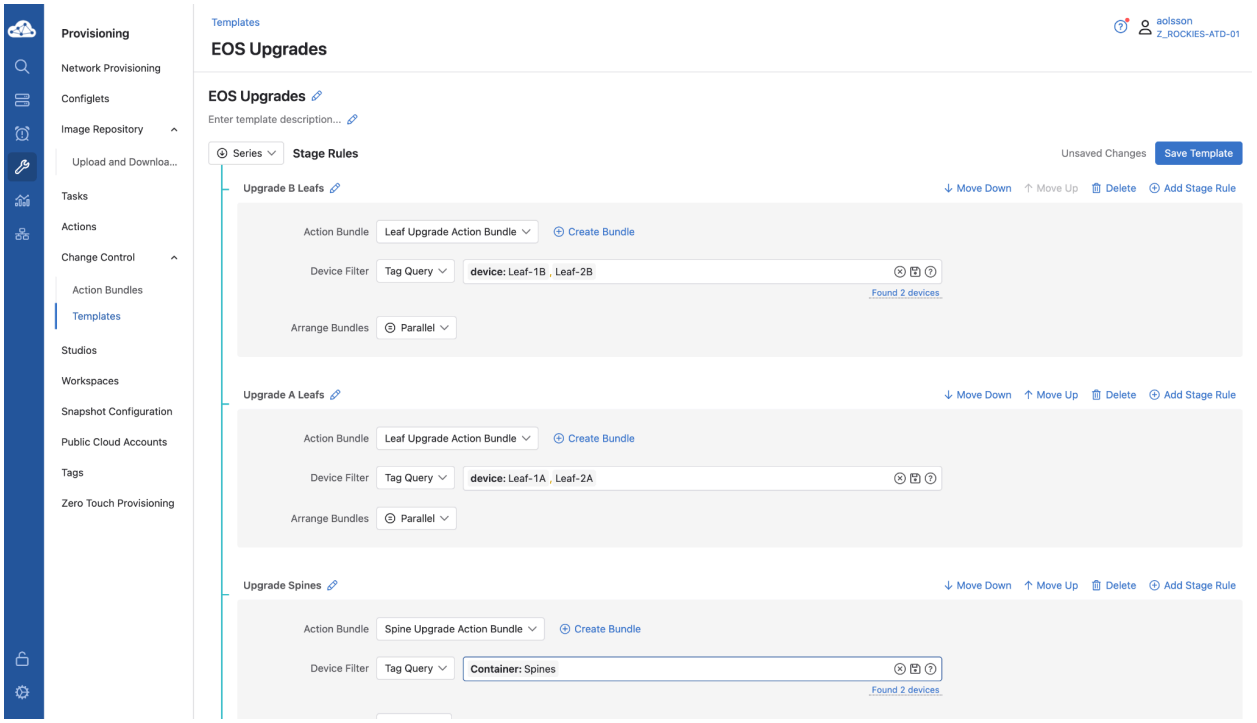

13. Now that the stages for the Leafs are complete, we can create the next stage, which will be used to upgrade the spines. Click on "**+ Add Stage Rule**" once again.
14. Change the name of the stage to "**Upgrade Spines**."
15. For the "**Action Bundle**," we need to use the "**Spine Upgrade Action Bundle**".
16. The Tag Query will be a bit different, we will use the "**Container**" tag for the Spines since both Spines are in the same container in Network Provisioning. Enter "**Container: Spines**" in the Tag Query field.
17. We'll leave the Arrange Bundles dropdown as "**Series**" because we want to upgrade one Spine at a time.




18. Add a final stage to the Template to clean the flash. We'll name it "**Clean Flash**". This will remove the old EOS Image after all the upgrades are completed. Select the "**Clean Flash**" action bundle created in the previous section in the Action Bundle dropdown.
19. The "**Device Filter**" can remain unchanged because we want to clean the flash on all the devices in the change control as part of this stage in the Template.
20. Change the "**Arrange Bundles**" dropdown from "**Series**" to "**Parallel**" so that this action can be run on all devices simultaneously.

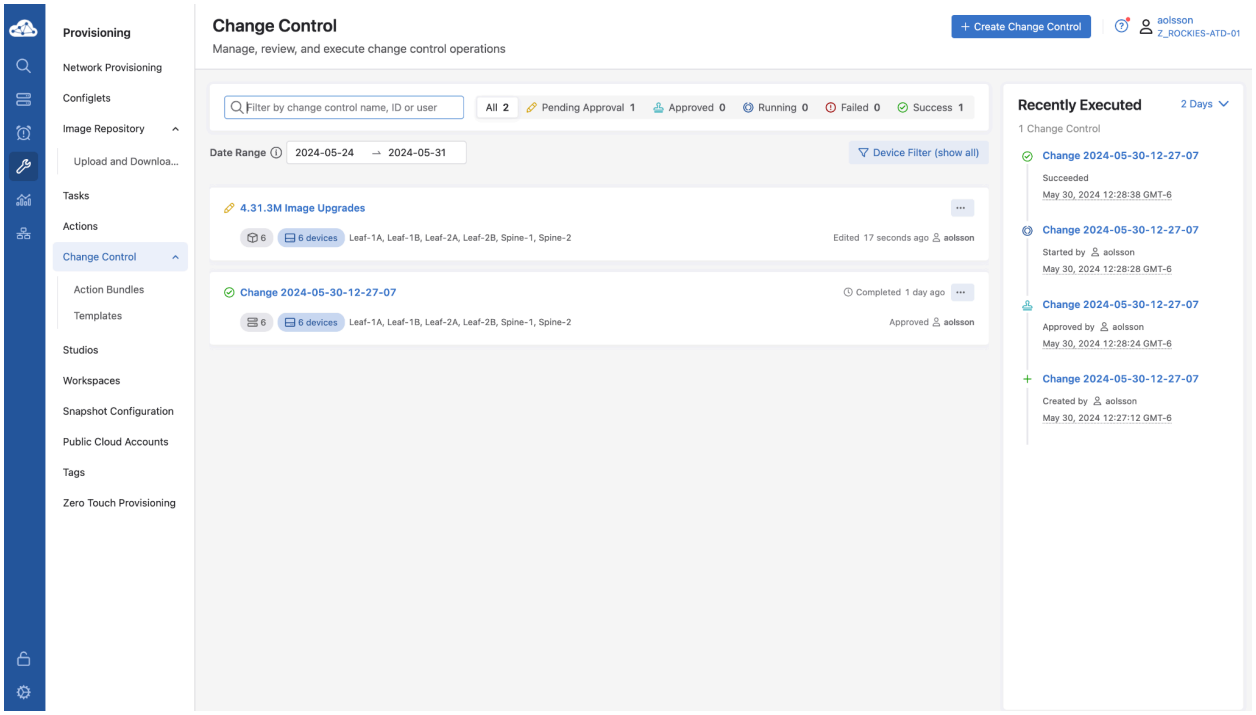



21. Change the “**Stage Rules**” section at the top left of the page to “**Series**” so all stages in the template happen in series.
22. Click on “**Save Template**” on the top right of the screen.

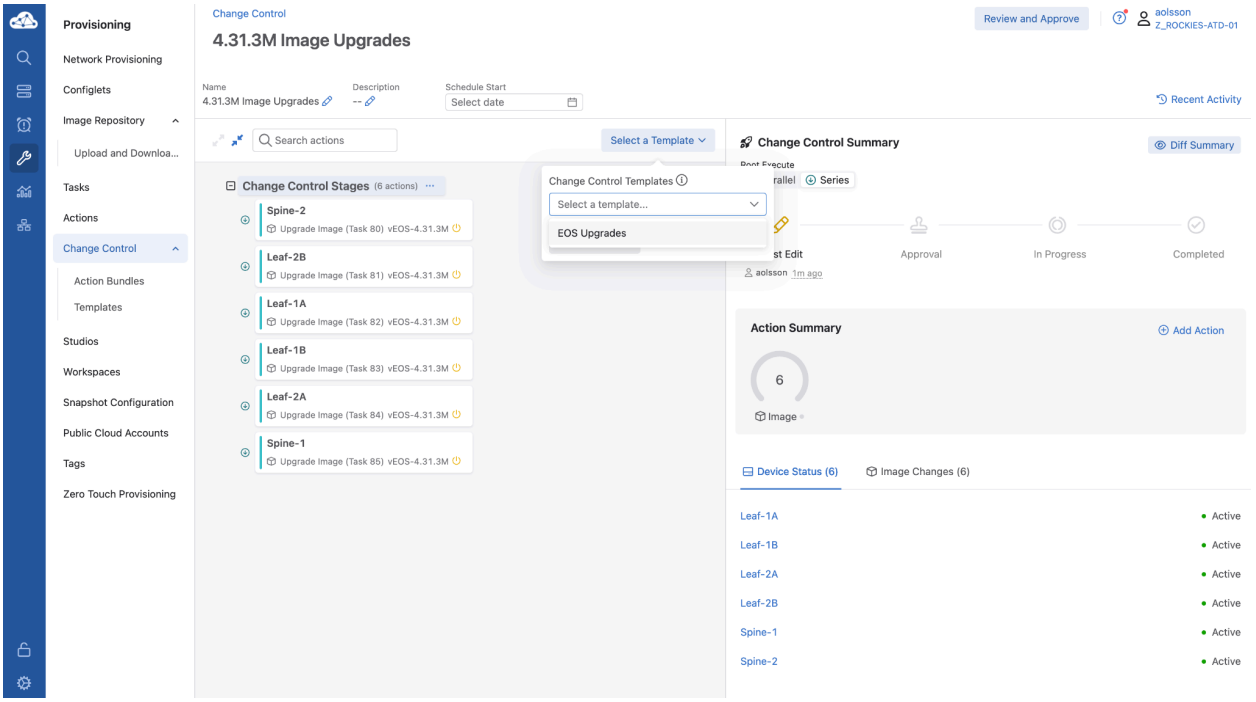



Now that the Action Bundles and Upgrade Template have been created, they can be added to the change control created previously.

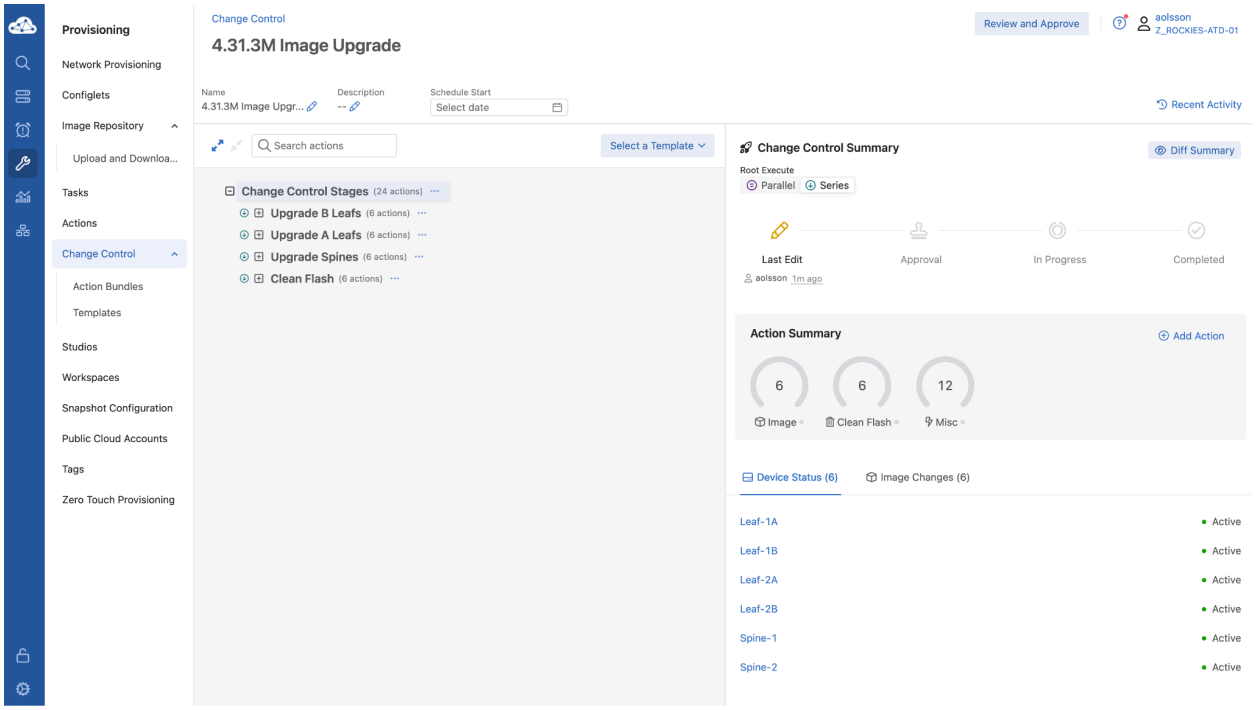
23. Head over to **Provisioning** > **Change Control**.
24. Select the “**4.31.3 Image Upgrade**” Change Control.



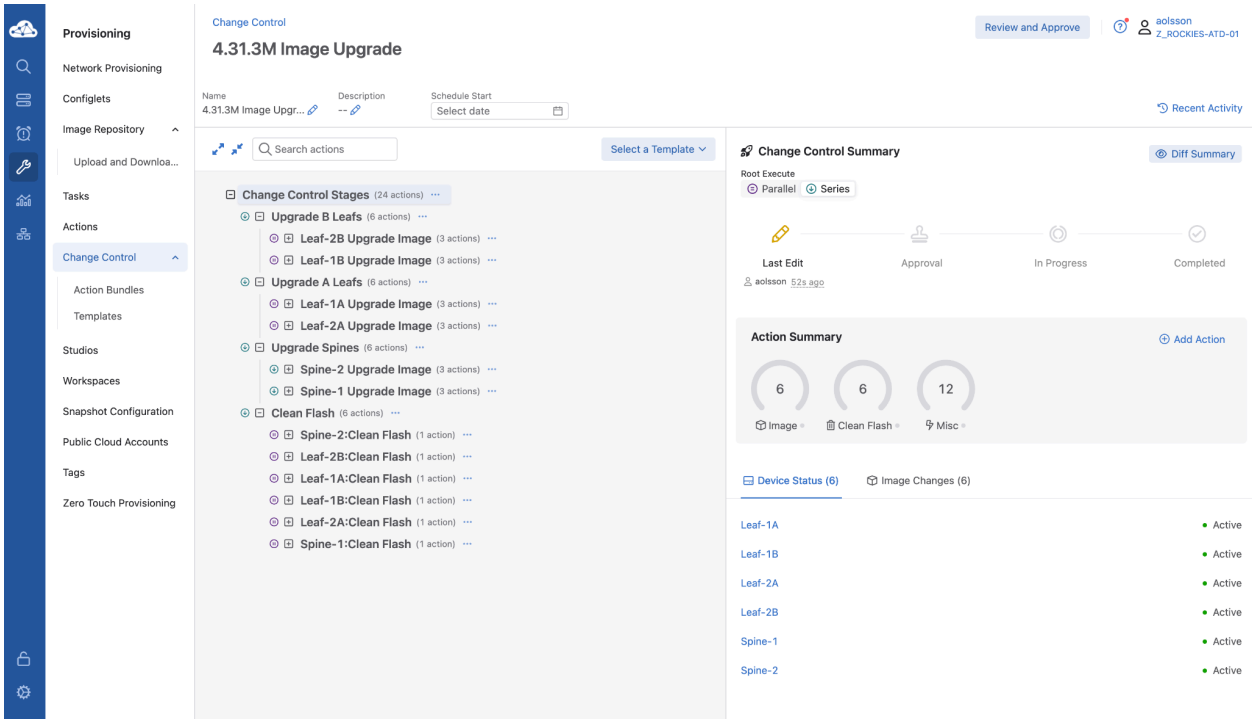

25. Once in the change control, click the blue “**Select a Template**” dropdown box.
26. In the “**Change Control Template**” dialog box, select the “**EOS Upgrades**” Template created in the previous section.
27. Select “**Apply Template**”.




The change control will reflect what was defined in the upgrade template. While expanding the newly added stages and looking at the change control, remember that the green circle with the arrow in the center means that the action will be completed in series, and the purple circle with the two parallel lines means that those actions will be completed in parallel.

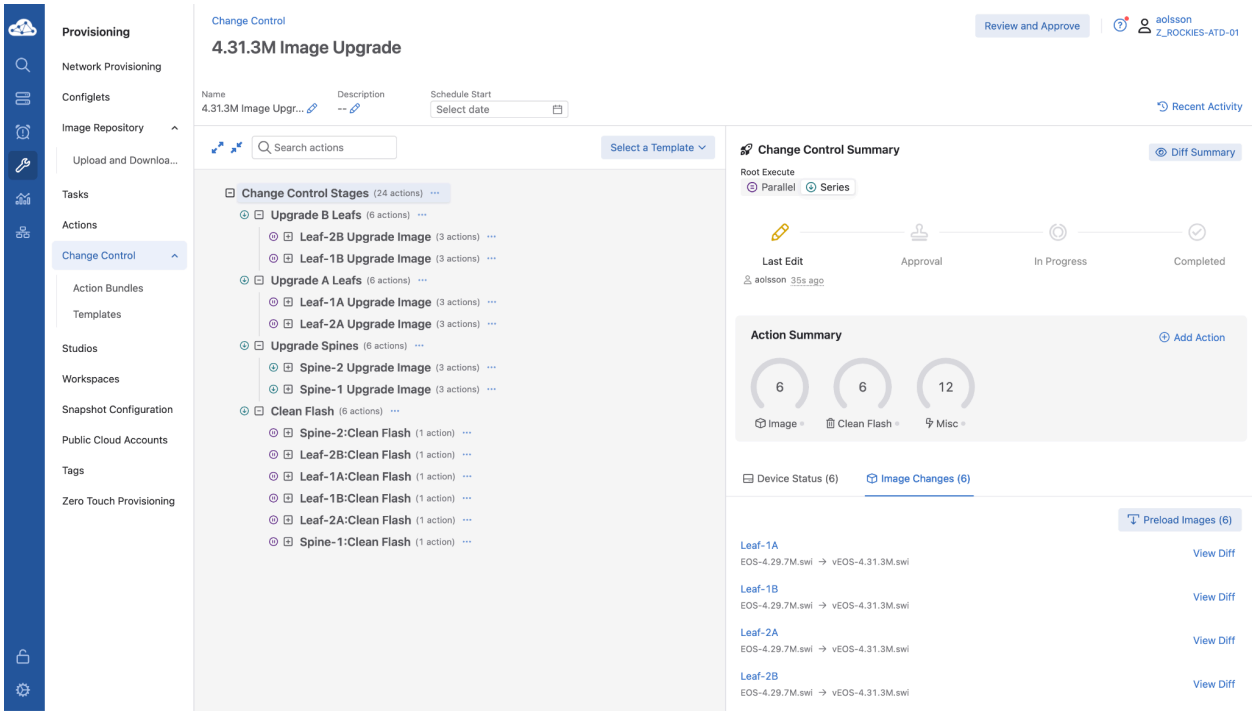



28. Expand each stage of the change control by Clicking the “**+**” next to each stage.
29. Validate that your change control looks similar to the screenshot below.

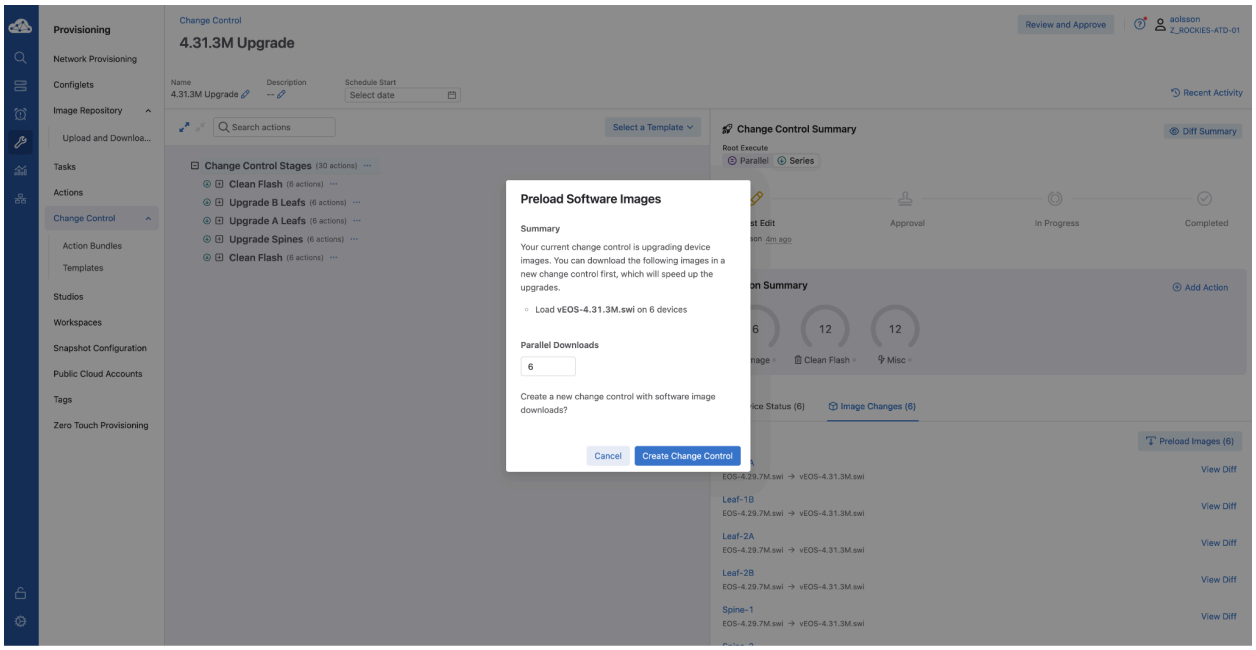


Now that the change control has been updated using the template, it can be executed as is. However, there's a process that can be used to preload the images on the devices to minimize upgrade times during a maintenance window.

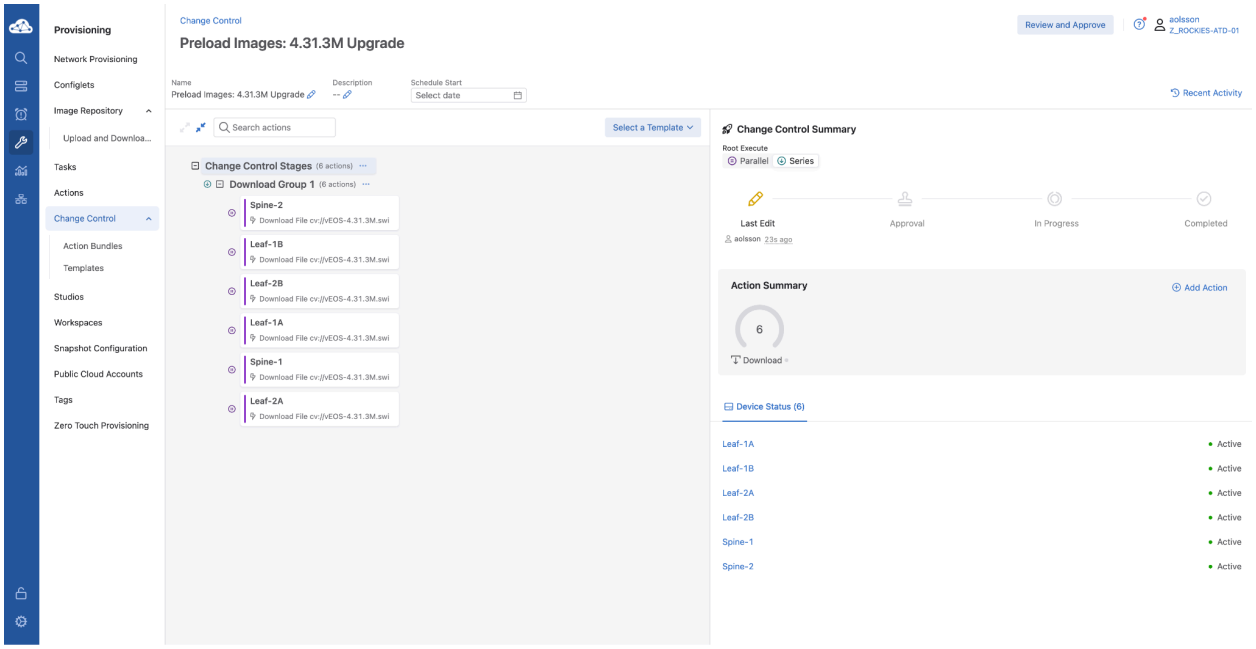
30. On the bottom right section of the change control, click on "**Image Changes**"



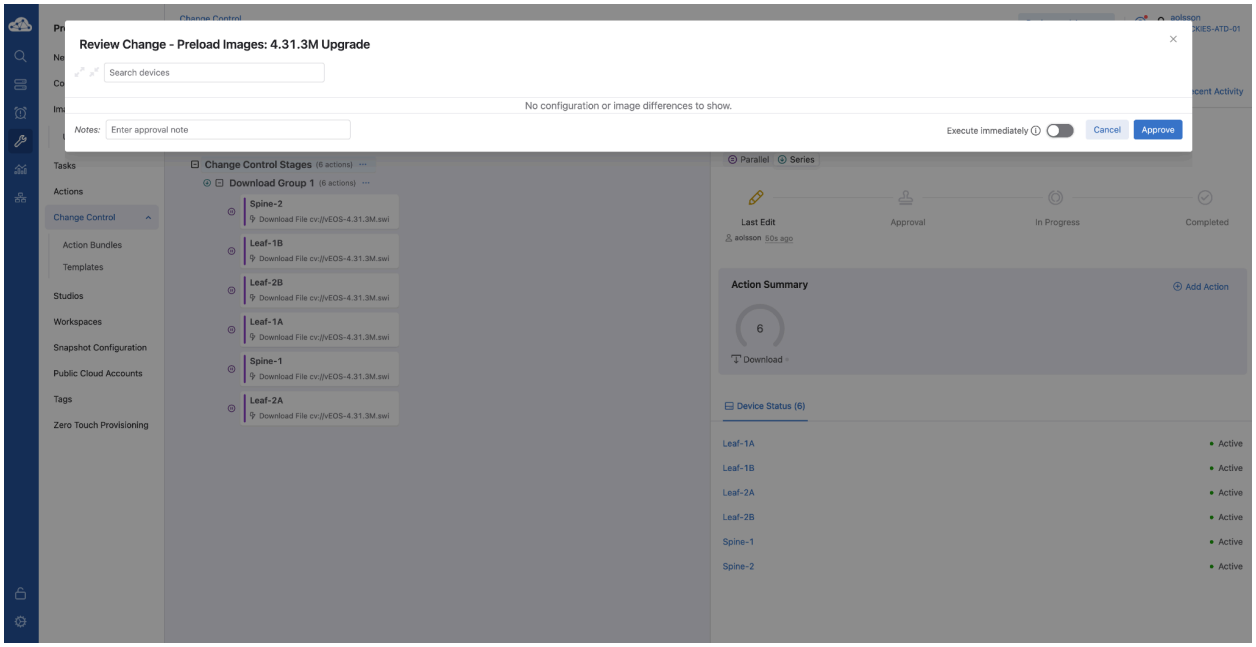

31. Select the blue "**Preload Images**" button.
32. A "**Preload Software Images**" dialog box appears. From here, the number of Parallel Downloads, which can be 6 for the lab, can be specified.
33. Click "**Create Change Control**" to generate a new change control to preload the images on all devices.



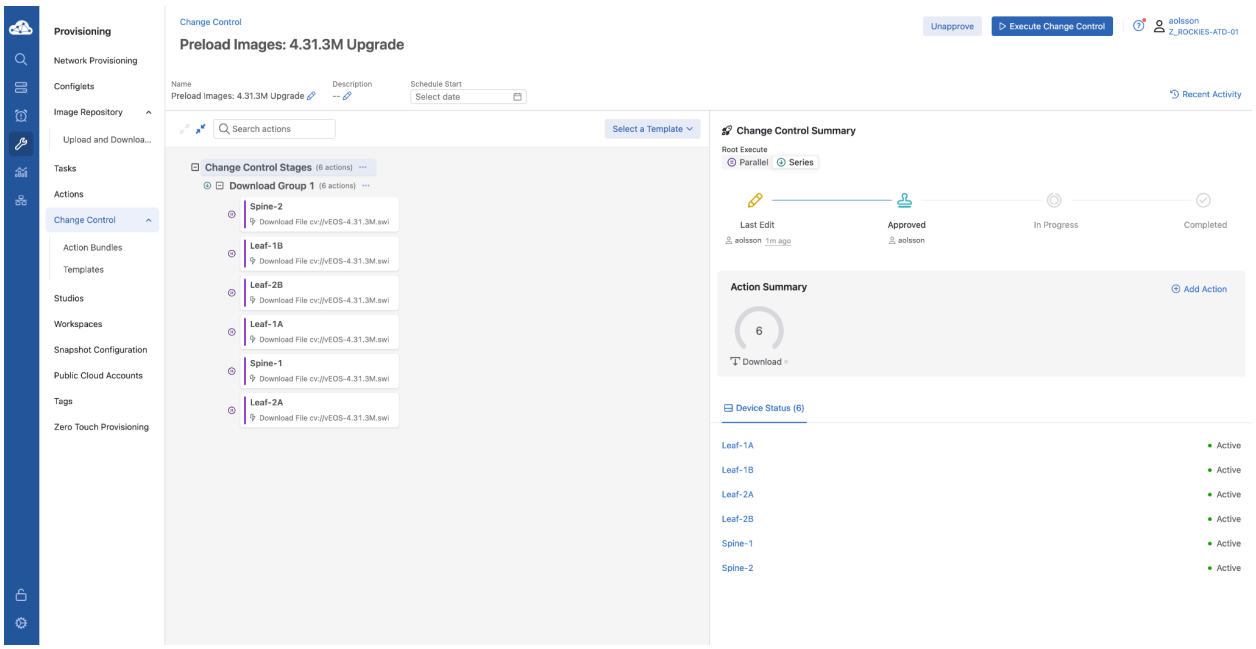

34. CloudVision will redirect you to the new "**Preload Images: 4.31.3M Upgrade**" change control.
35. Select "**Review and Approve**" in the upper right corner to review the change control.



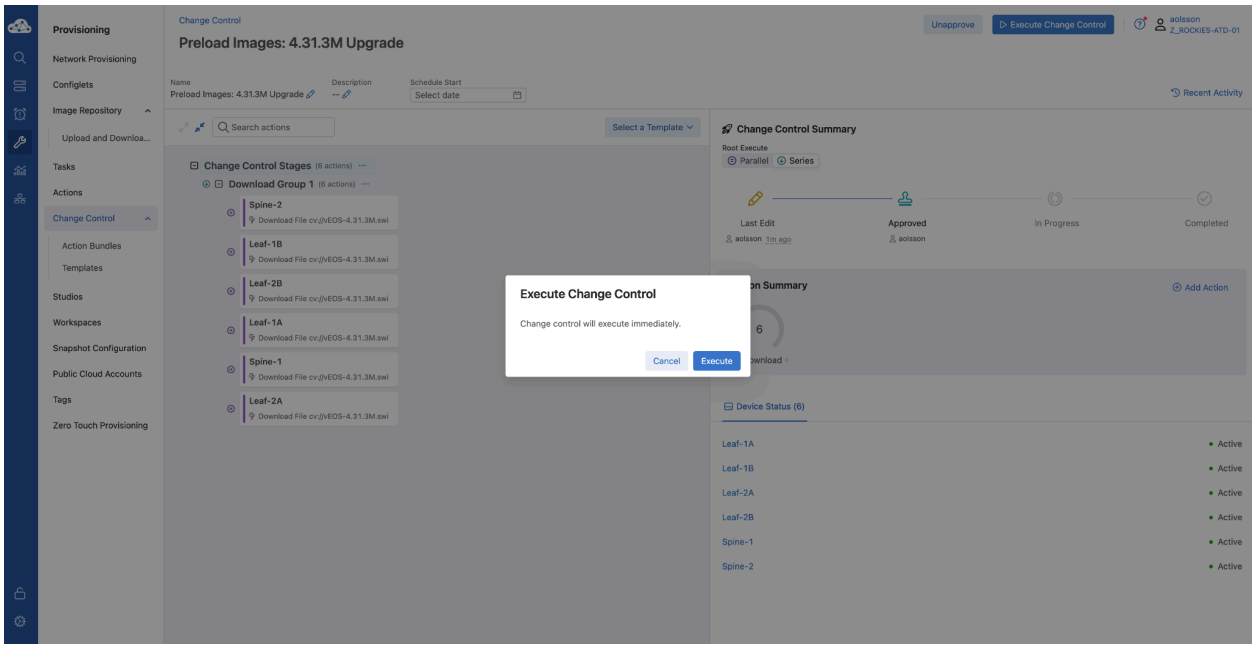

36. The "**Review Change**" pop-up will appear and show no changes to the device are being made. This is expected because we're just transferring the EOS image to the device, so no actual changes are being made.
37. Once satisfied with the change control review, select "**Approve**."



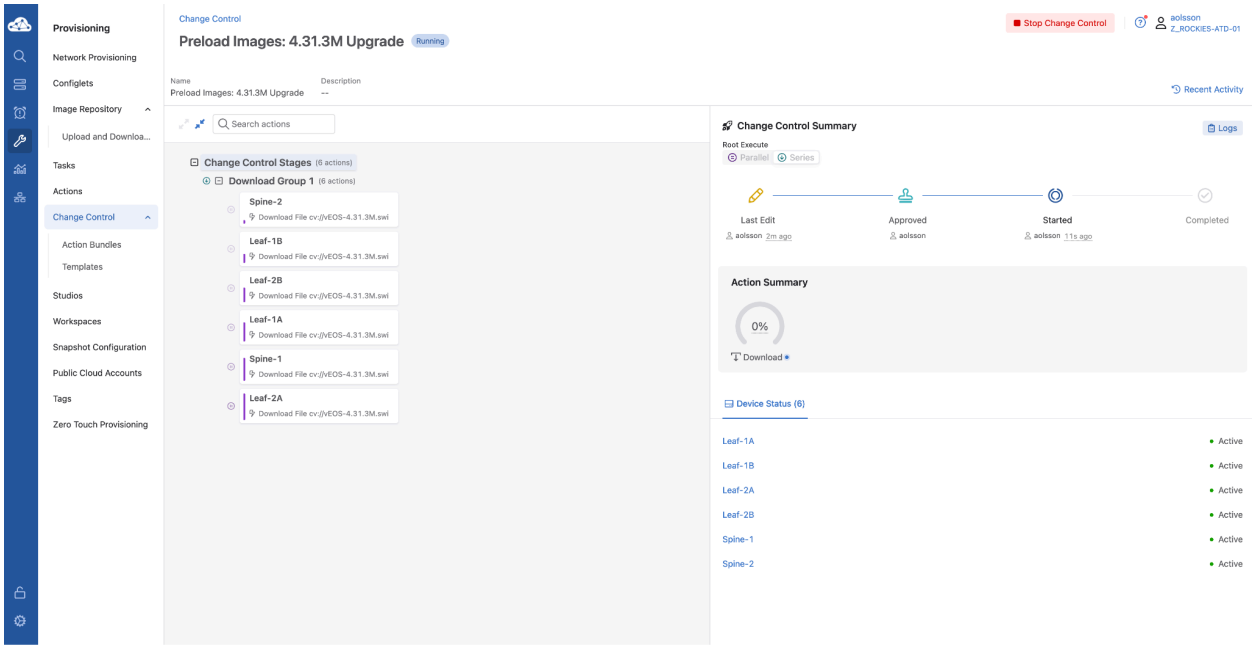

38. Select the blue “**Execute Change Control**” button.



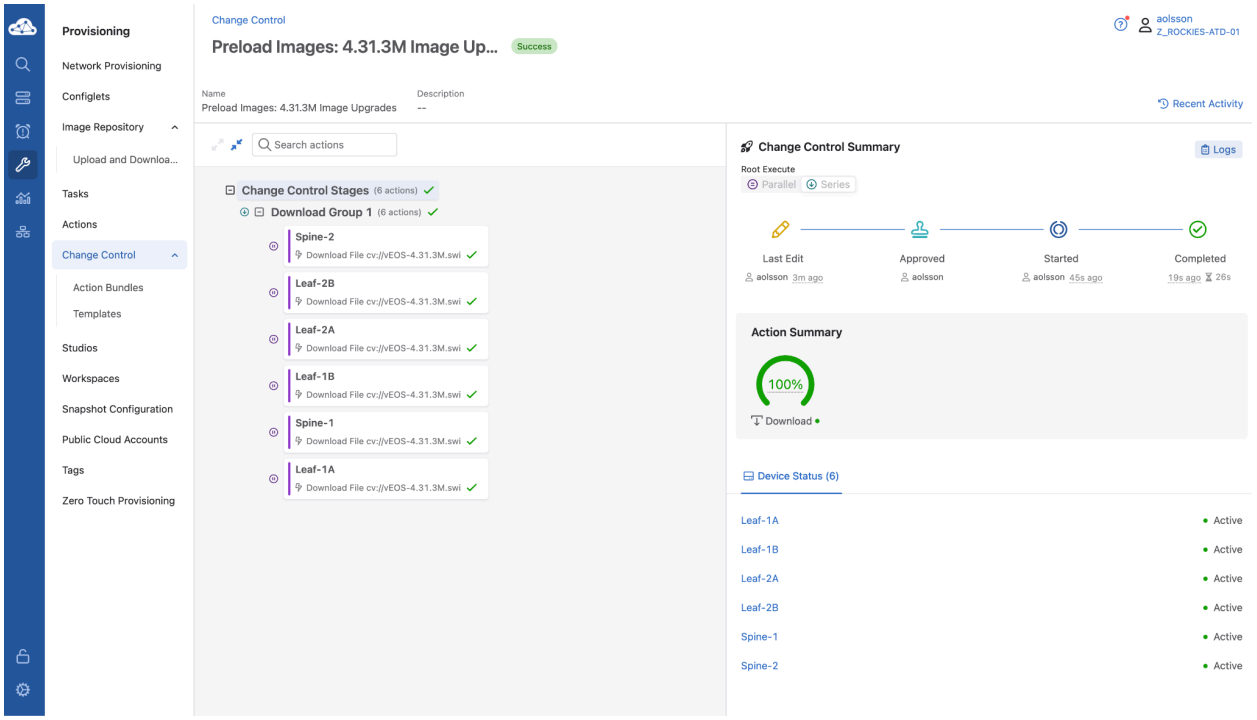

39. An “**Execute Change Control**” pop-up will appear. Select **Execute** to begin the change control execution.



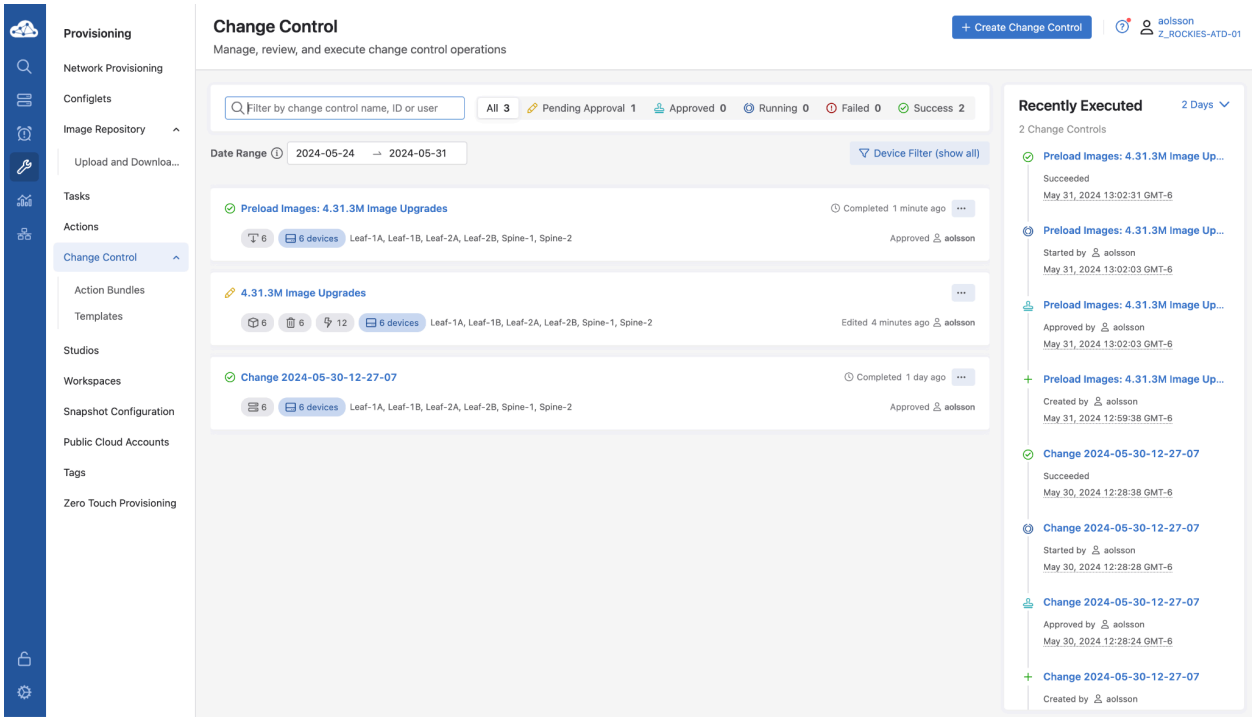

40. The change control animations will appear next to each task in the change control, showing that all tasks are being executed.



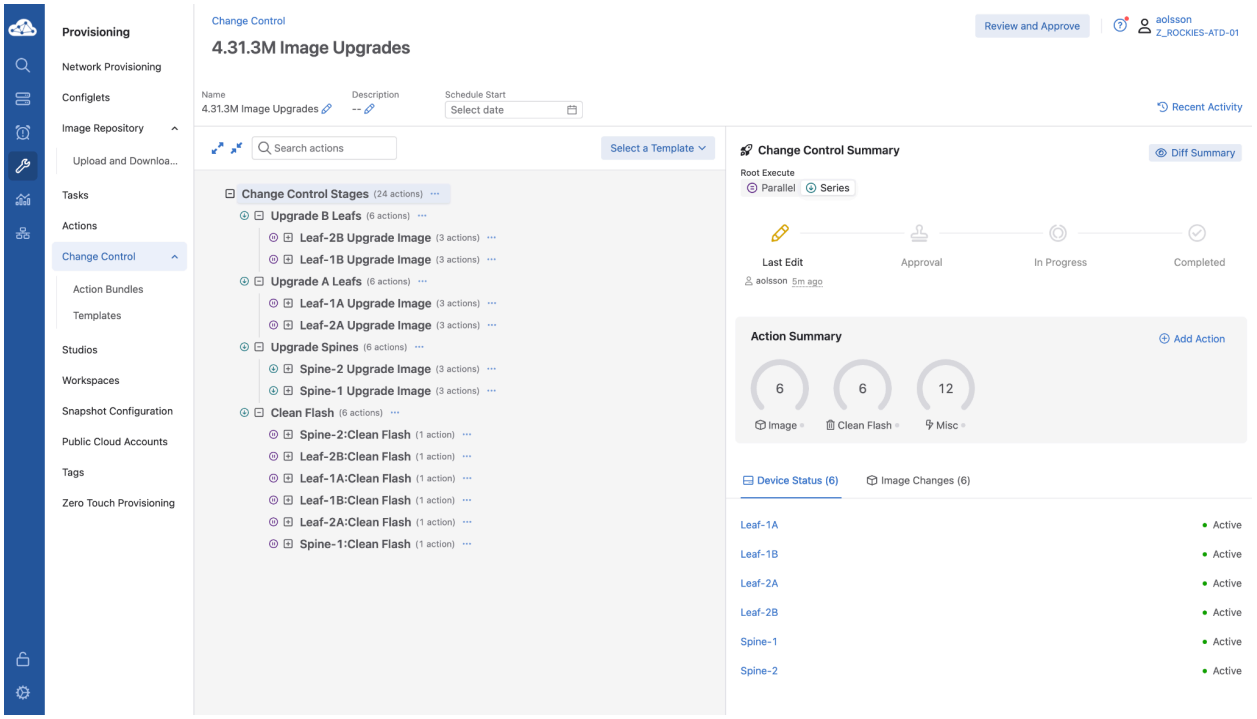

41. As the tasks are complete, a green check will appear at the right of each task. Once the change control has been completed, a green “**Success**” message/label will appear to the right of the change control name.



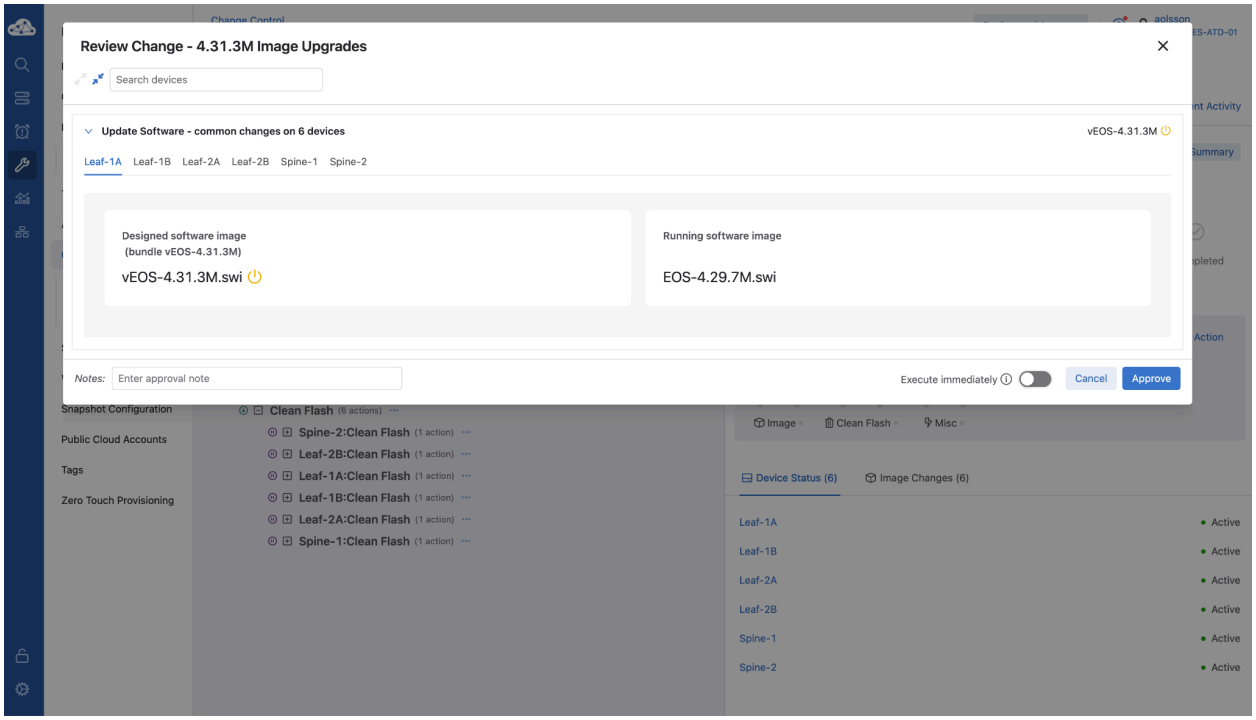

42. The upgrade change control can be executed now that the EOS images have been preloaded onto all devices. Head to **Provisioning** > **Change Control** and select the upgrade change control, which is named "**4.31.3M Image Upgrades**."



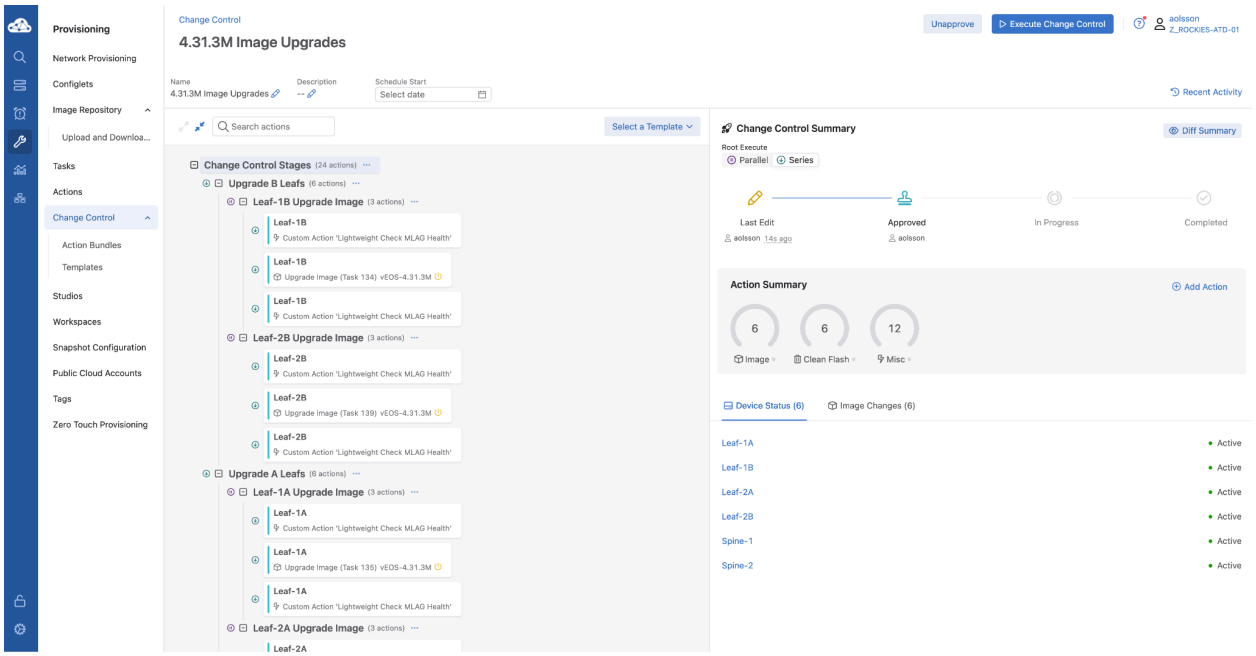

43. Select the blue "**Review and Approve**" button in the upper right corner.



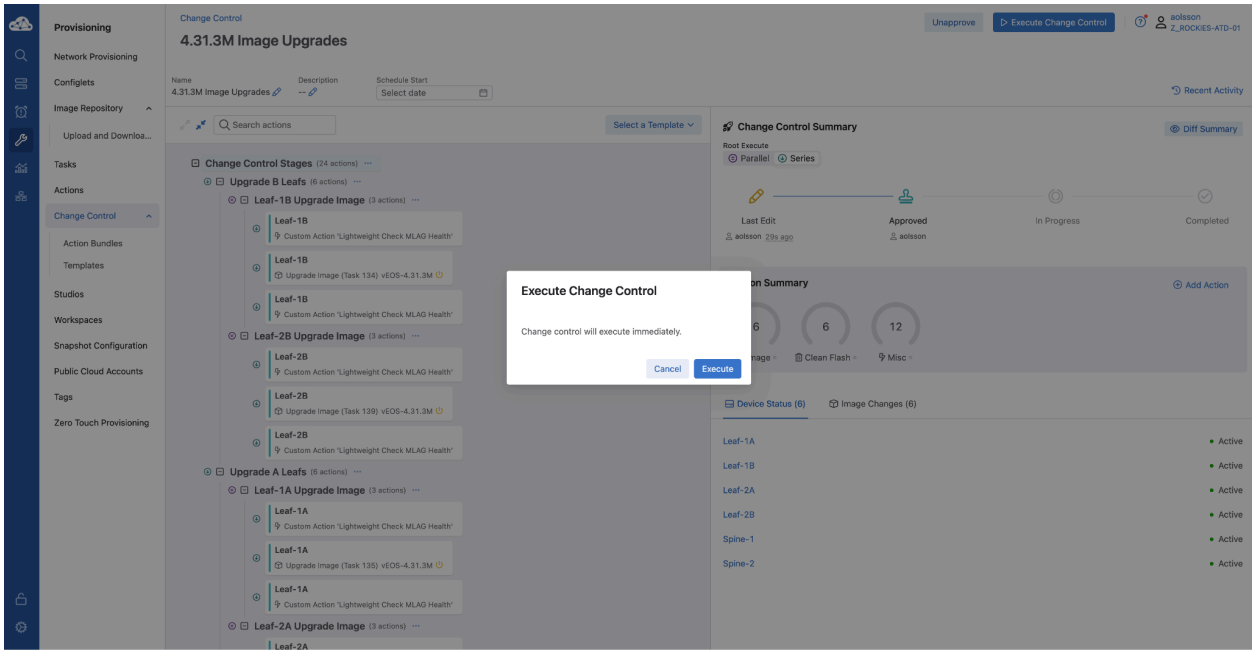

44. The “**Review Change”** pop-up will appear for this change control and should indicate that EOS is being upgraded. The changes for each of the 6 devices should be grouped together because all 6 devices are running the same version of EOS and are being upgraded to the same version of EOS. Select the blue “**Approve**” button to approve the changes.



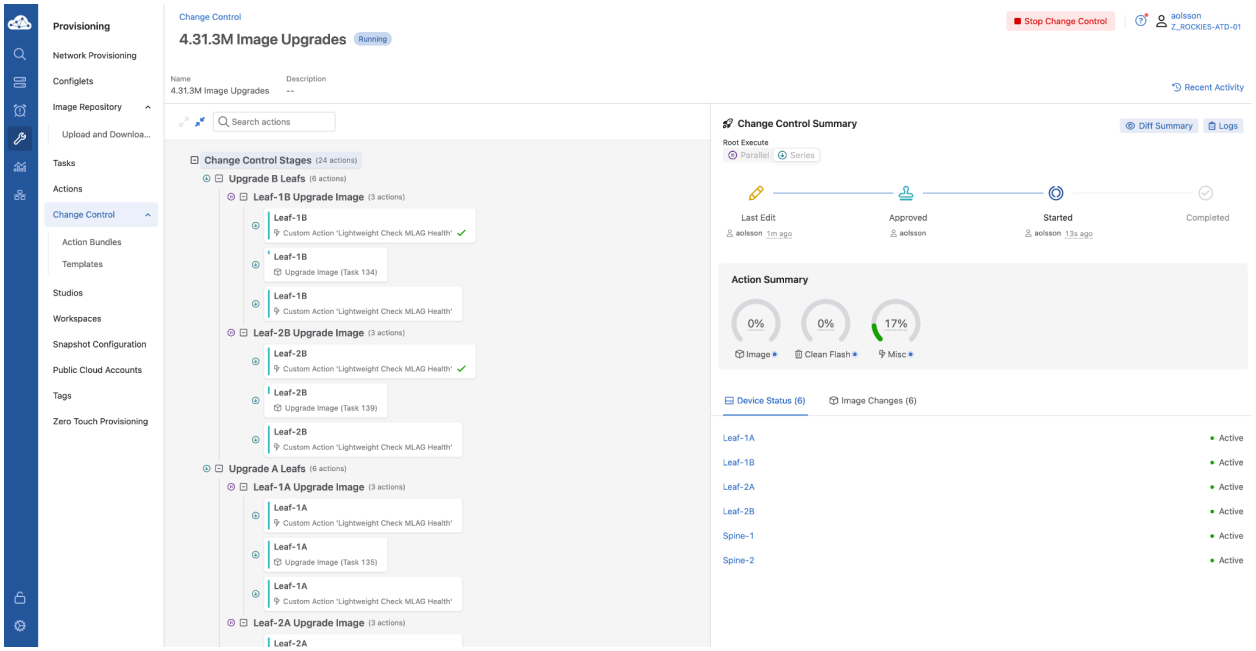

45. Select the blue "**Execute Change Control**" button in the upper right corner.



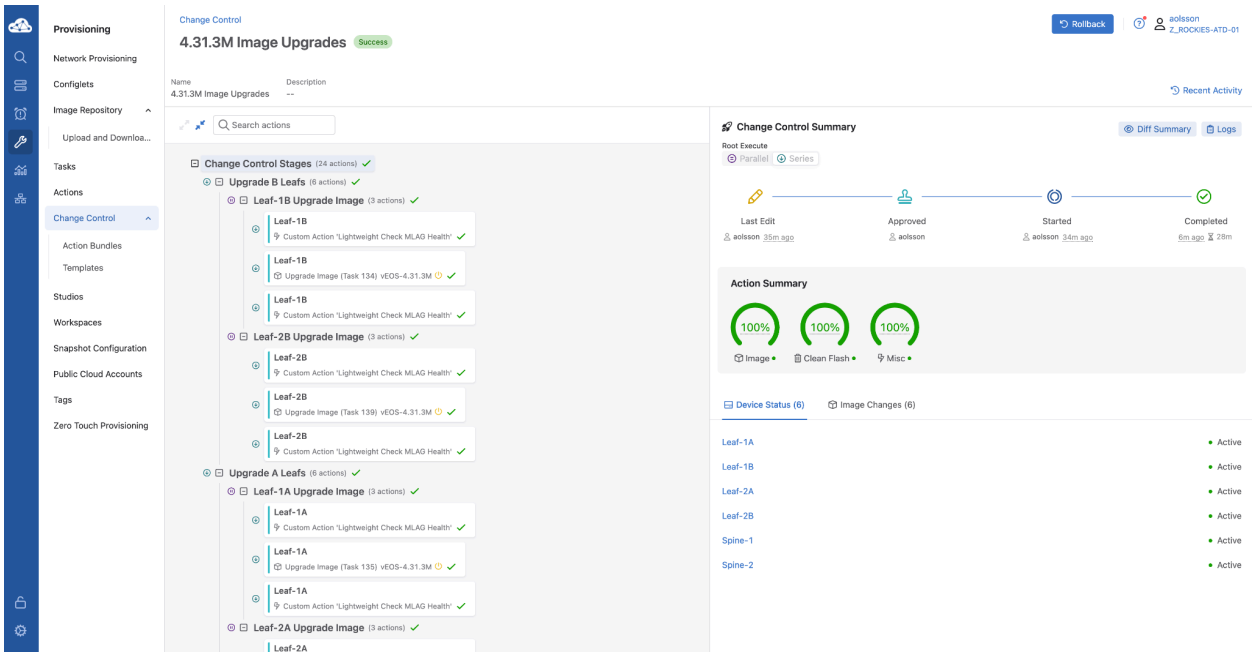

46. An "**Execute Change Control**" pop-up will appear. Select **Execute** to begin the change control execution.




47. Once again, the change control animations will appear next to each task in the change control as it runs. A green check mark will appear just to the right of each task when it is completed.




48. When the change control has been successfully completed, a green “**Success**” message/label will appear next to the change control name.



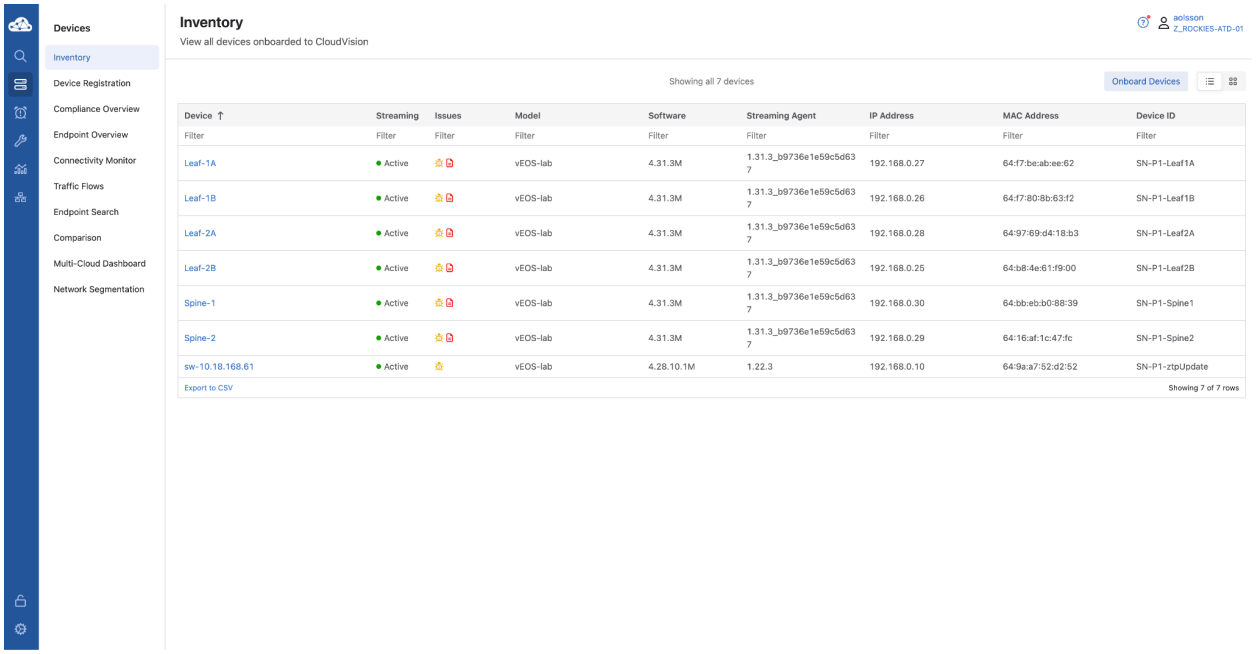

# Compliance - Device Configuration

CloudVision continuously computes image and configuration compliances. If a device is either configuration, image, or extension non-compliant, CVP automatically generates a non-compliant event on the Compliance dashboard and flags the device as non-compliant on the Inventory screen.

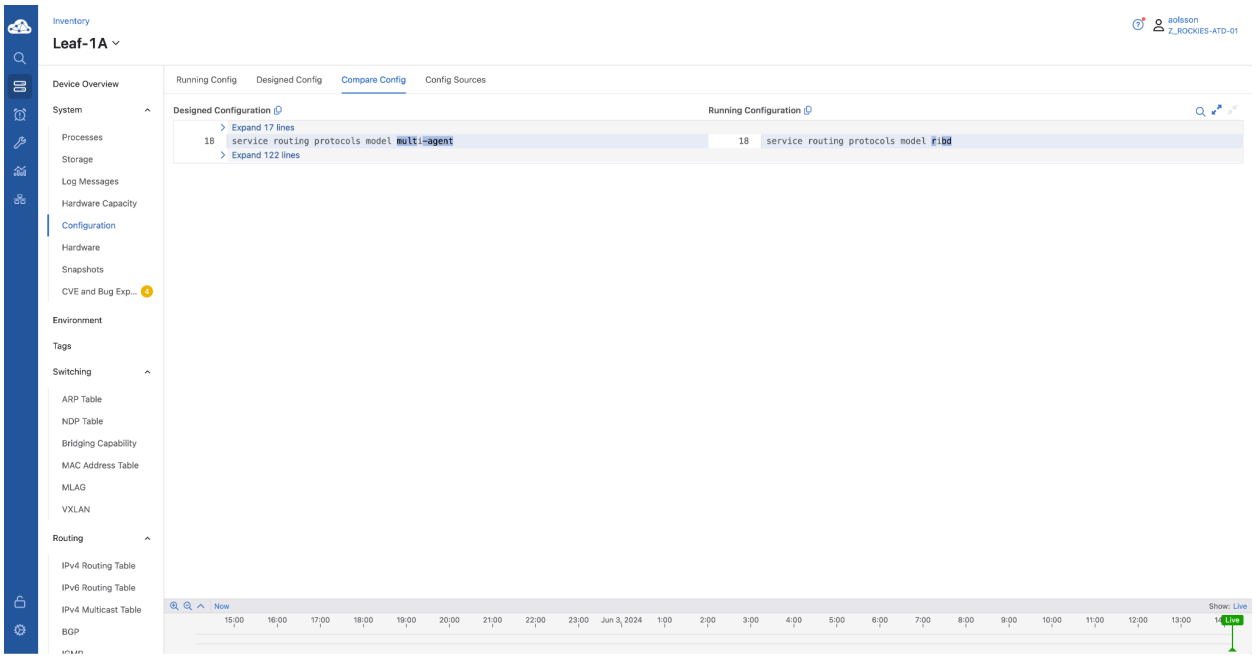
1. Navigate to **Devices** > **Inventory**.

   The Inventory page will show that all devices are now running 4.31.3M but will also show a red “**Page**” icon in the “**Issues**” column that wasn’t there before the upgrade. This icon indicates that the running configuration on the devices differs from what CloudVision thinks it should be running. This happened because “**ribd**” was the default routing model before EOS 4.30, and “**multi-agent**” was the default for EOS 4.30 and later.

**NOTE**: Changing the routing model protocol from “**ribd**” to “**multi-agent**” requires a reboot for the changes to take effect. This lab will skip the reboot due to time limitations.

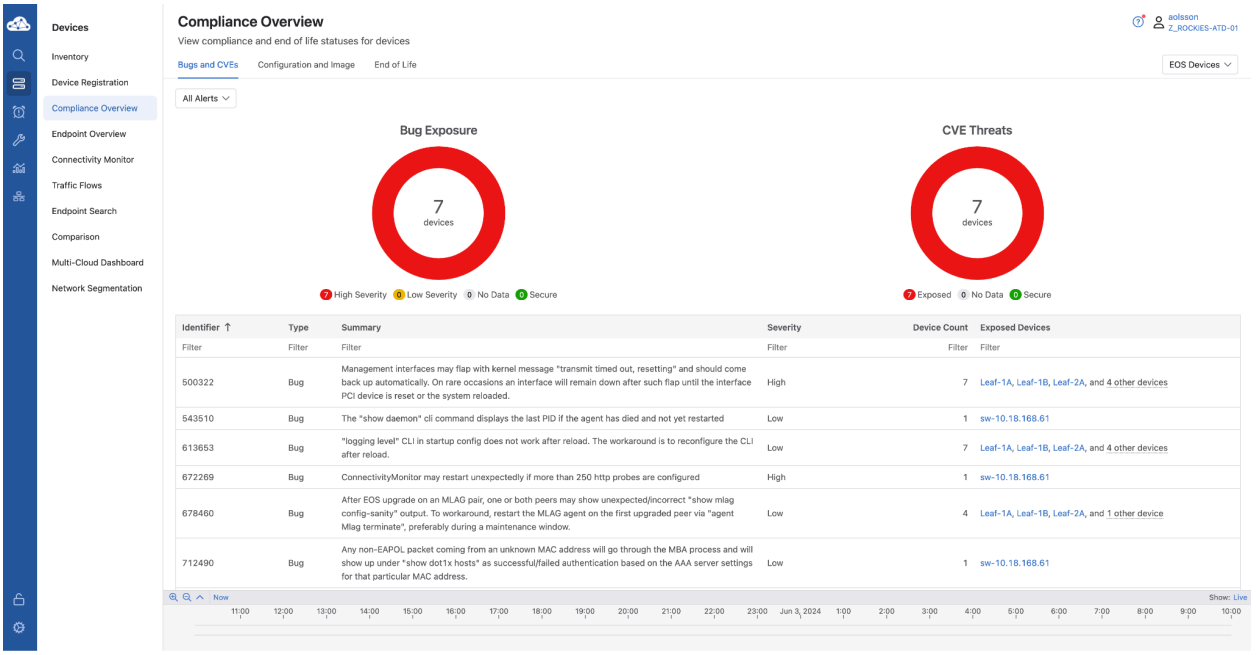



2. Click on the red “**Page**” icon next to one of the devices. This will take you to that device’s configuration “**Compare Config**” section.

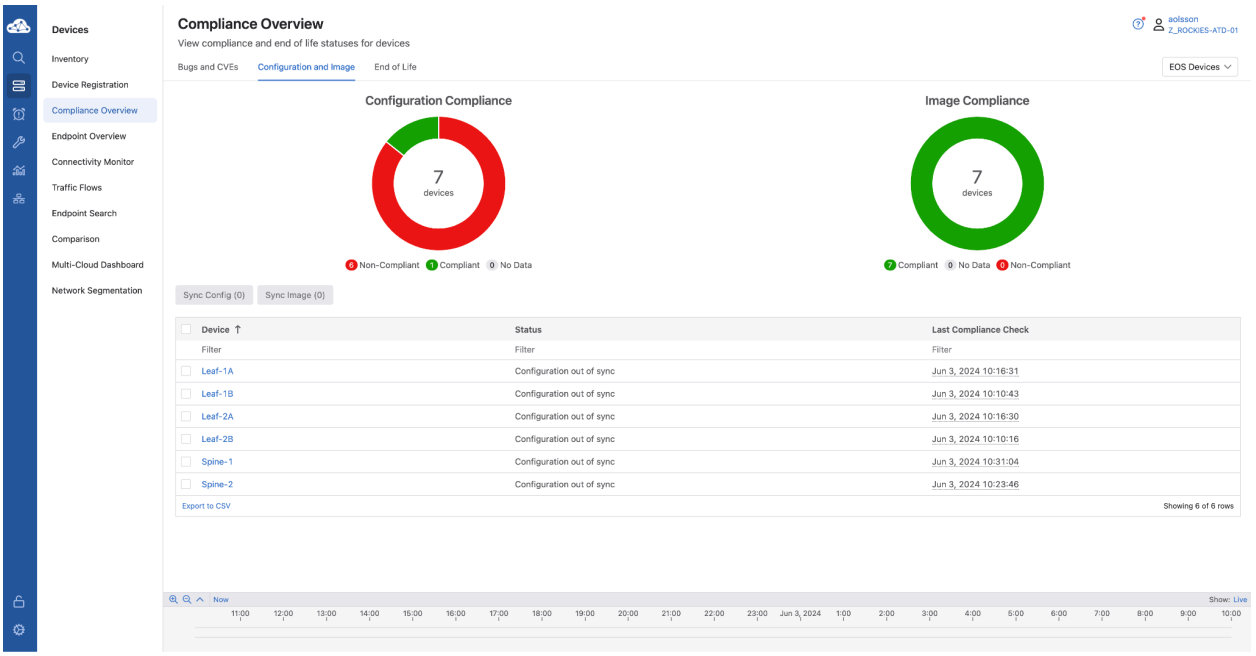



**NOTE**: It’s always a good idea to validate device configuration compliance within CloudVision after an upgrade in case EOS default values have changed.

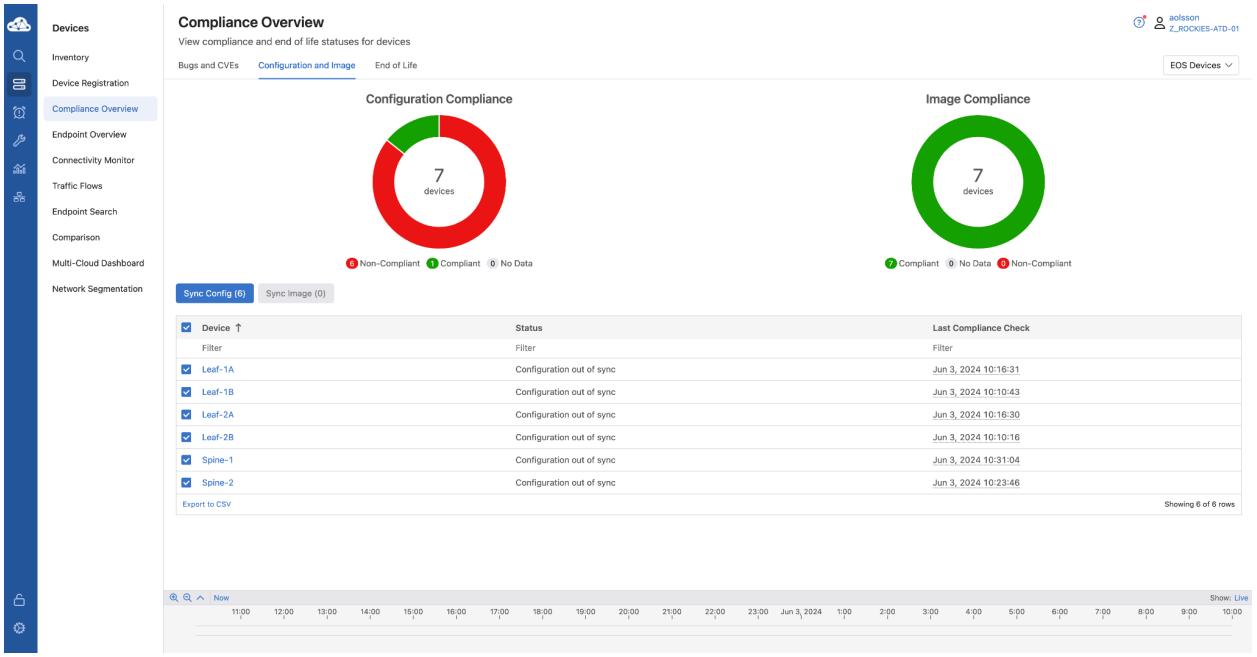
3. To bring the device configuration back into compliance, head to **Devices** > **Compliance Overview**



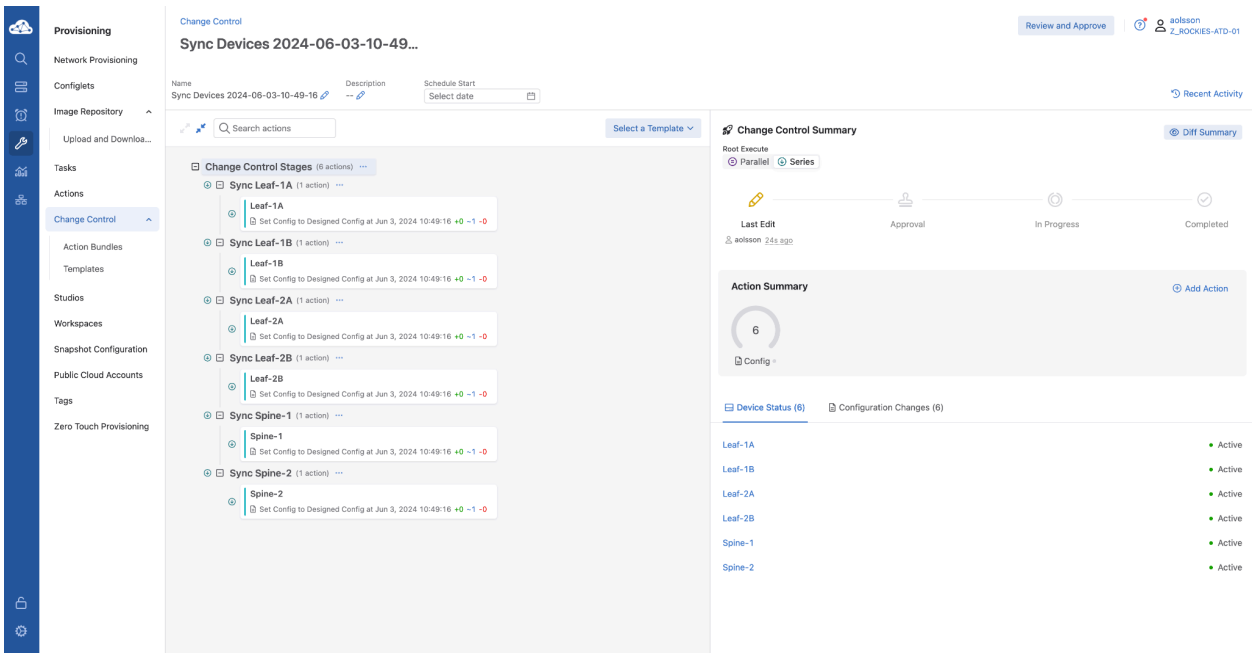

4. Select the “**Configuration and Image**” tab near the top.




5. Select all 6 devices that need to be brought back into compliance
6. Click the blue “**Sync Config**” button



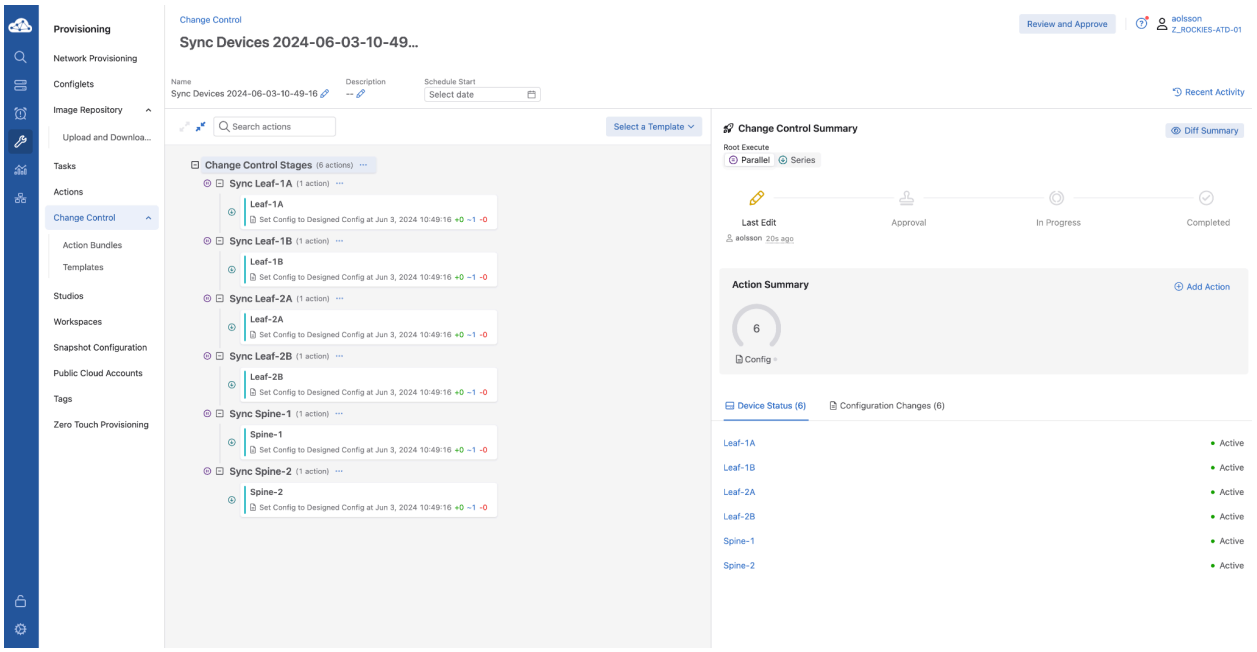

7. You'll be redirected to the change control overview screen for the change control created to bring the devices back into compliance.



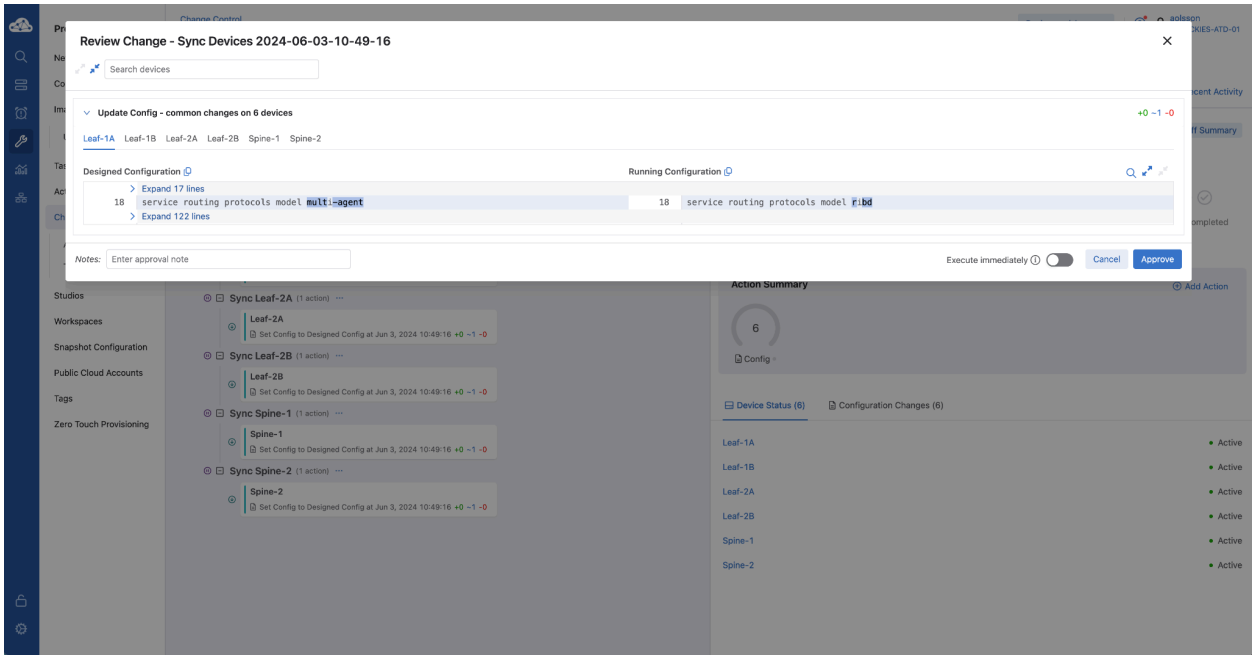

8. By default, all change controls created using the "**Compliance Overview**" page will be in series. For this lab, we want to run all changes in parallel, so select the "**Change Control Stages**" line item at the top of the screen in the section showing the tasks to be

executed. Once the “**Change Control Stages**” line is highlighted, select the “**Parallel**” button on the right side of the screen just under the “**Change Control Summary**.”

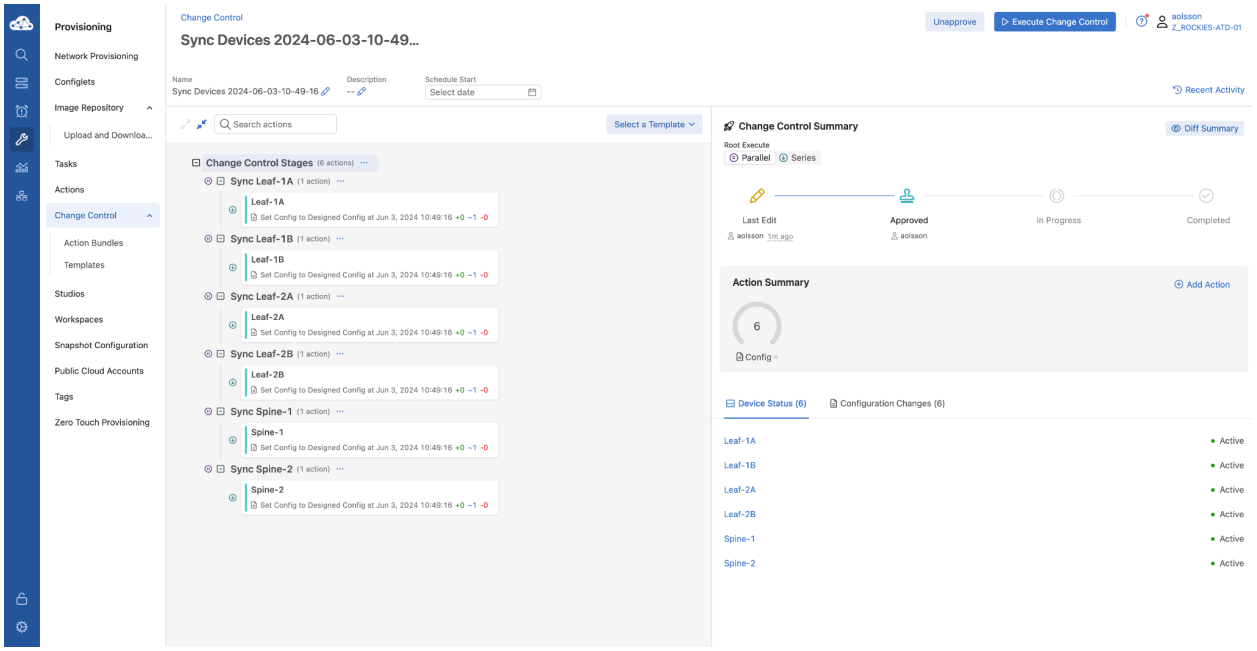
9. Select the blue “**Review and Approve**” button on the top right of the screen.



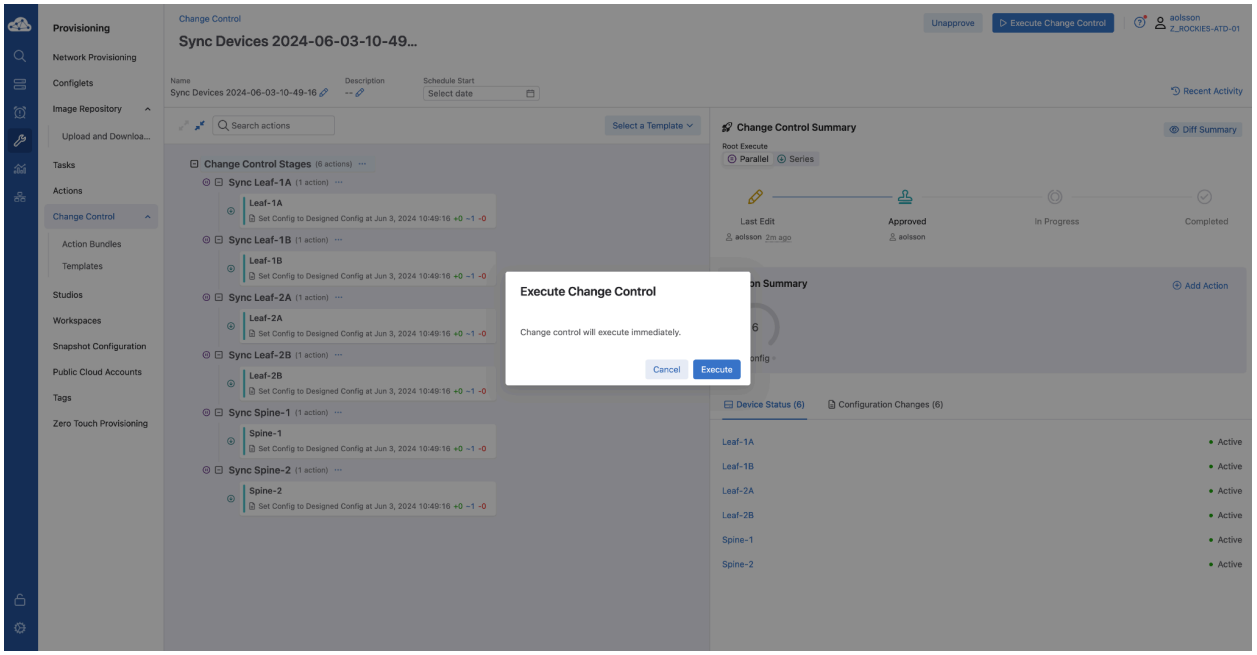

10. The “**Review Change**” pop-up will appear, showing a single configuration change on all 6 devices. Select the blue “**Approve**” button to approve the change control.



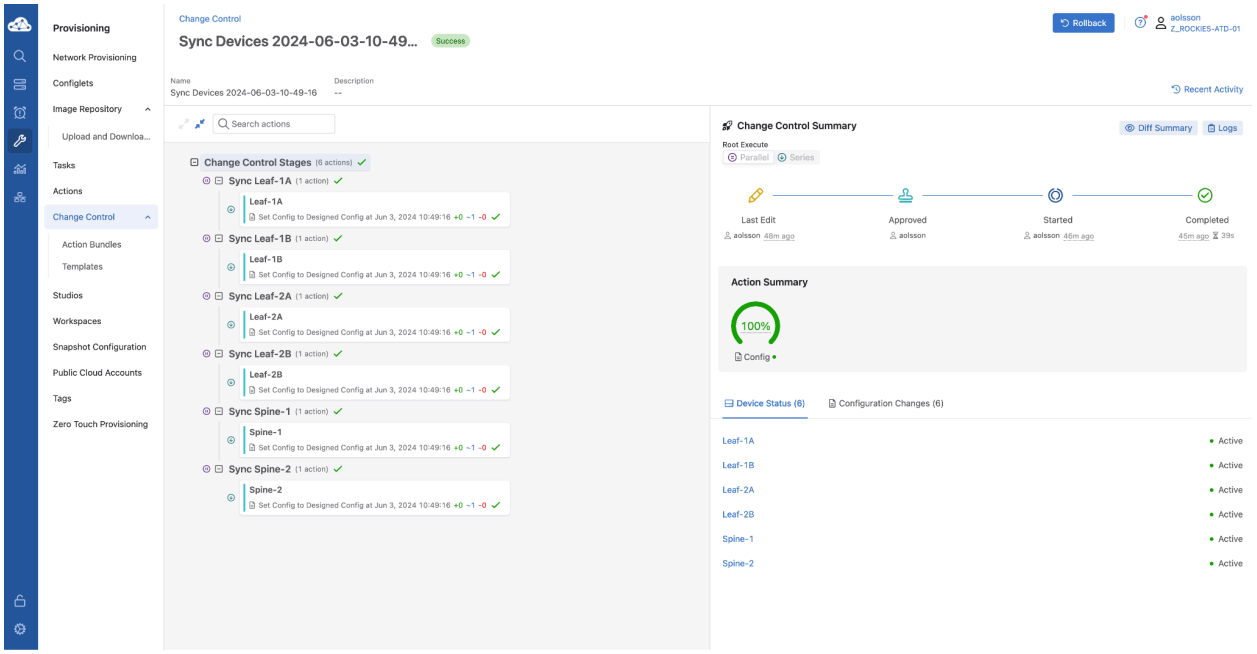

11. Select the blue “**Execute Change Control**” button on the top right.




12. The “**Execute Change Control**” pop-up will appear. Select the “**Execute**” button to continue.




13. Once completed, green check marks will appear next to each task, and the green “**Success**” icon will appear to the right of the change control name.



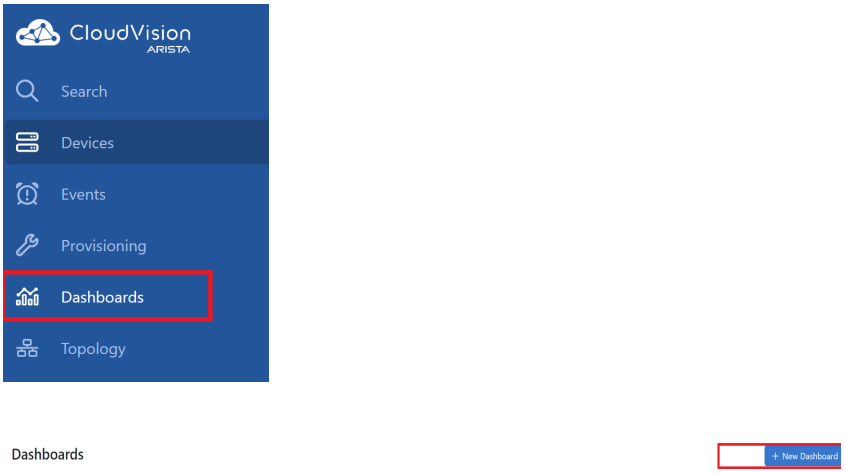

# Dashboards

CloudVision allows you to create customizable dashboards consisting of multiple metrics across various datasets in different views. You can quickly resize and drag widgets on the grid to accommodate various custom layouts views. Data is gathered from the real-time telemetry data that the Arista switches are streaming to CVP.

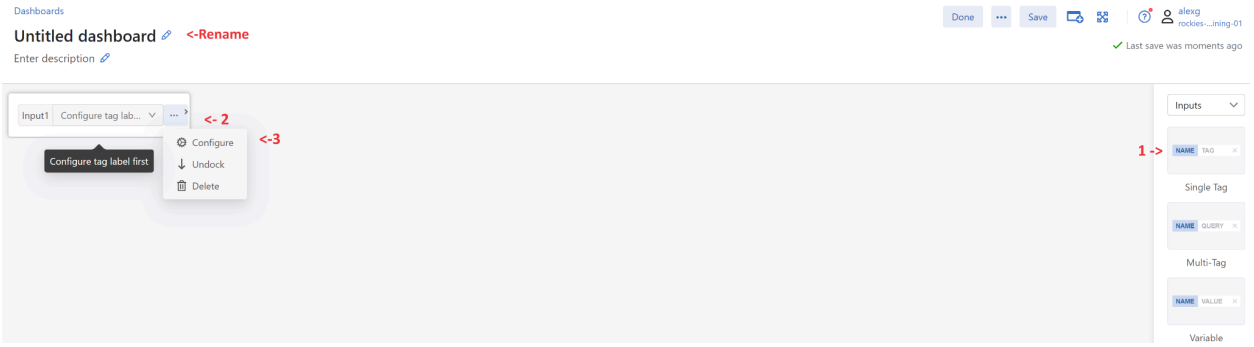
## Creating Custom Dashboards

In this lab we will be creating a custom dashboard that will be used as an overall network status page that can be used in a NOC as a visual representation for the overall state of the network.

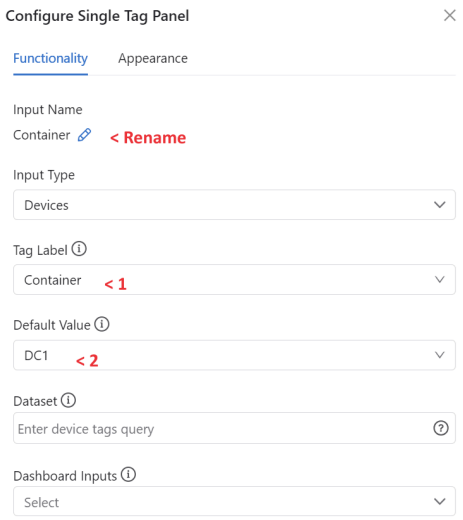
1. Navigate to the Dashboards section of CloudVision, and click on the **“New Dashboard”** button.




2. Rename your dashboard to "<**FirstName_LastInitial_NOC**". Select **Inputs** > **Single tag** from the dropdown menu on the right. Then click on the "**3 dots (...) > Configure"** for your input field.

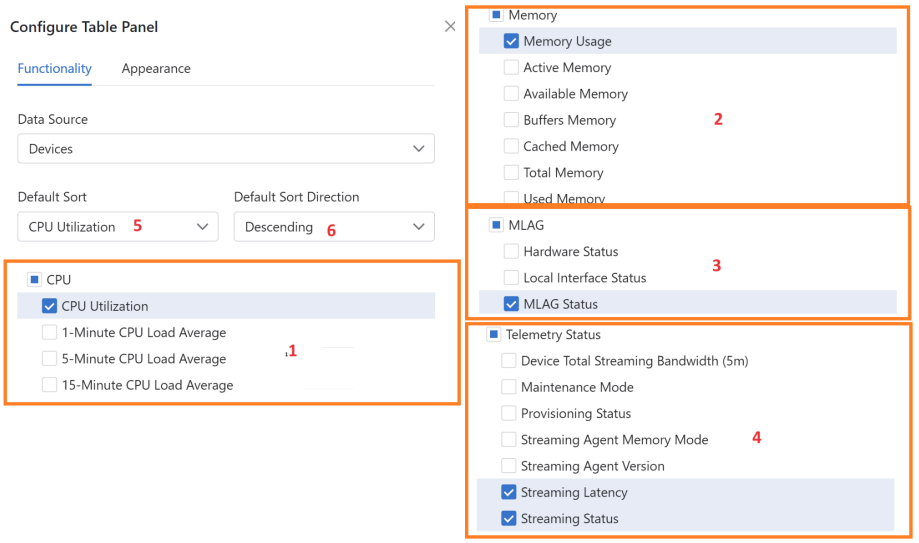



3. Configure the Single Tag Panel with the name "**Container**". Select "**Container**" as the "**Tag Label**", and select "**DC1**" as the default value, close the configuration panel.

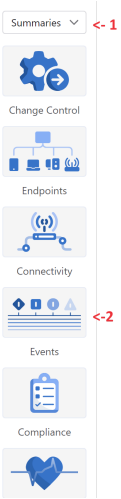



4. Insert a new graph by selecting “**Metrics**” from the drop down on the right, then select “**Table**”.


5. Configure this graph with the “**CPU > CPU Utilization**”, “**Memory > Memory Usage**”, “**MLAG > MLAG Status**”, and “**Telemetry > Streaming Latency + Streaming Status**” options in the picture below. Then sort by “**CPU**” > “**Descending**”, close the configuration panel.



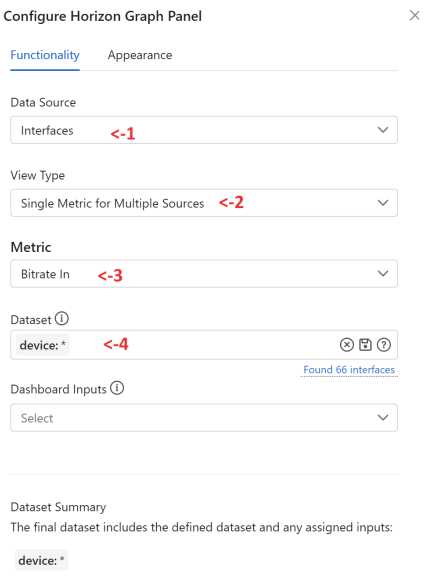

6. Insert a new graph by selecting “**Summaries**” and then “**Events**”.




7. The preconfigured values for the “**Events**” dashboard are already correct.

8. Insert a new graph by selecting “**Metrics**” then “**Horizon Graph**”.

9. Select “**Interfaces**” as your Data Source, “**Single Metric for Multiple Sources**” as the View type, “**RX Traffic Rate** > **Bitrate In**” as your Metric, and “**Device** > *****” as your Dataset, close the configuration panel.




10. Insert a new graph by selecting "**Summaries**" then "**Compliance**".

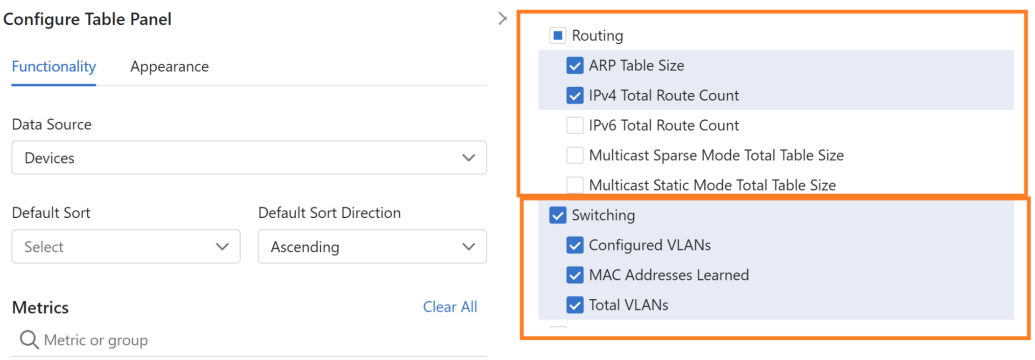
11. Select "**Configuration Compliance**" for your Compliance Metric, close the configuration panel.

12. Insert another Compliance graph by selecting "**Summaries**" then "**Compliance**".

13. Select "**Bug Exposure**" for your Compliance Metric, close the configuration panel.

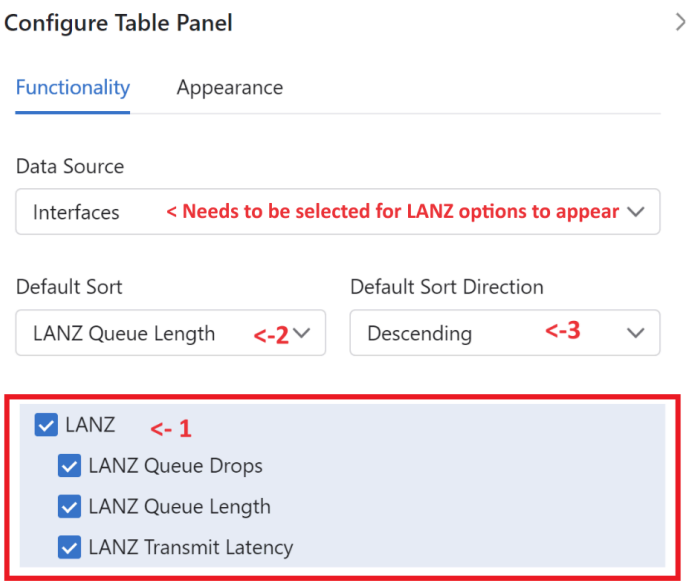
14. Insert a new graph by selecting "**Metrics**" then "**Table**"

15. Select the "**Routing** > **ARP Table Size** + **IPv4 Total Route Count**", "**Switching** > **Configured VLANs**", "**MAC Addresses Learned**", and "**Total VLANs**", close the configuration panel




16. Insert a new graph by selecting "**Metric**" then "**Table**"

17. Select “**Interfaces**” as the Data Source, then select “**LANZ**” and all suboptions, the sort by “**LANZ Queue Length**” and “**Descending**”, close the configuration panel



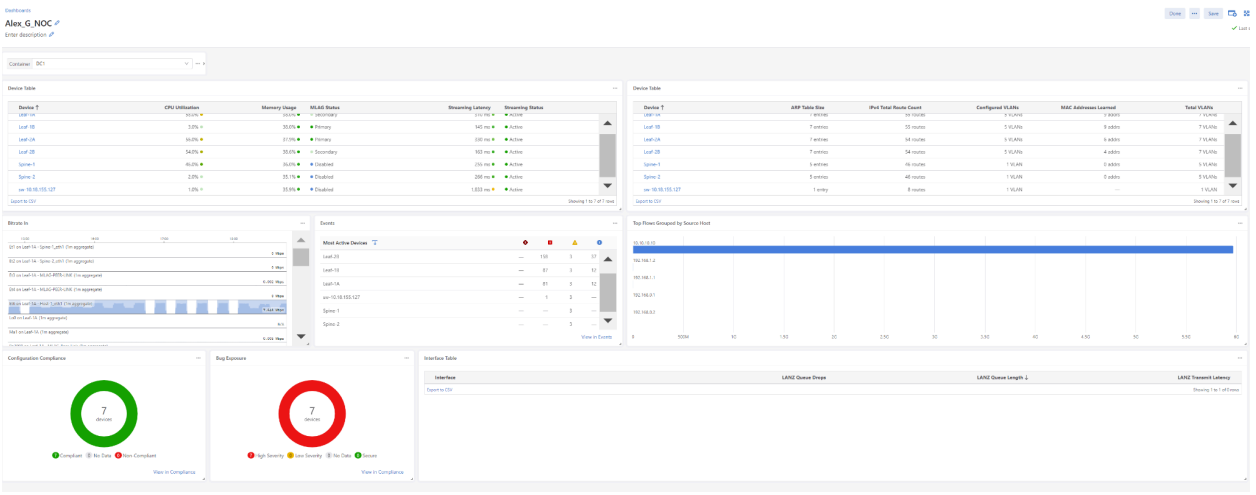

18. The prior “**LANZ**” Dashboard is for instruction and will not be functional due to the lab running on **vEOS**, as this feature can only be run on hardware. We will demonstrate the **LANZ** feature in a later section of the Workshop.

19. Insert a new graph by selecting “**Summaries**” and then “**Traffic Flows**”

20. The default values for the “**Traffic Flows**” dashboard are already correct

21. Organize the dashboard panes as you see fit by dragging and dropping them on the page.

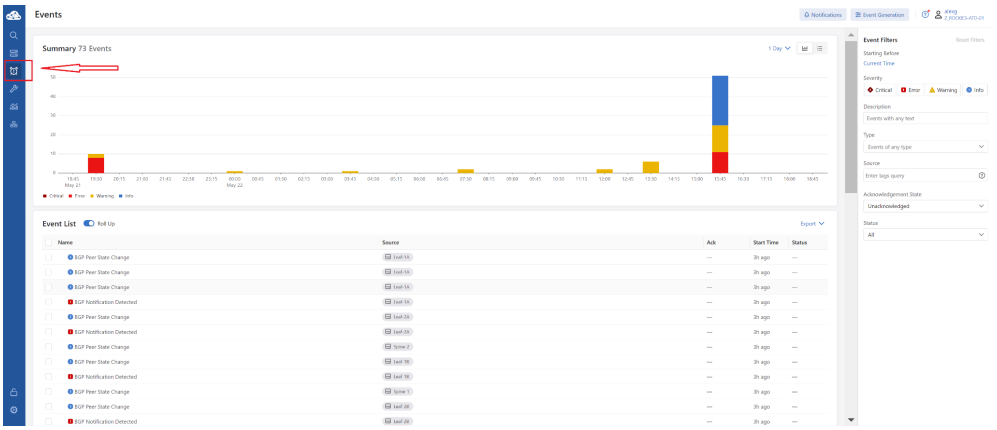
22. Your Dashboard should appear similar to the example below:.



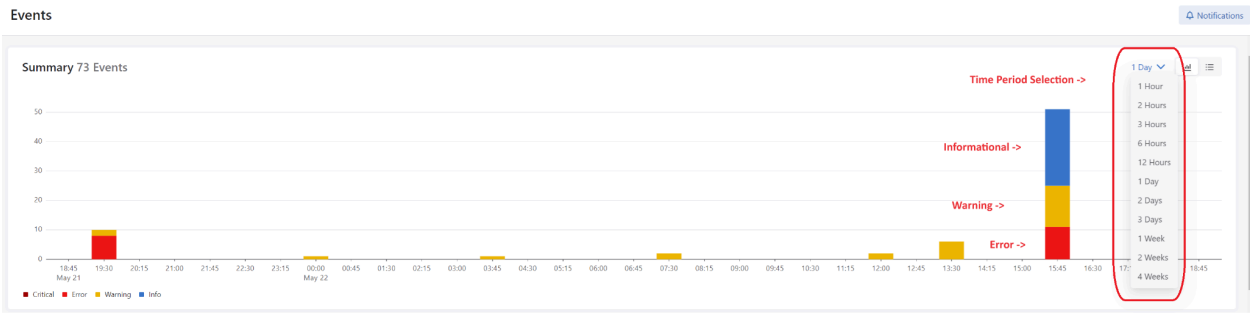
# Events and Notifications

CloudVision events and notifications allow users to configure alerts for network events and receive notifications about them. Users can access the events screens by clicking the Events tab, which provides an overview of network events and allows users to change the time range and filter events. Users can also click on an event to view more details, acknowledge it to hide it, or export the events to a CSV file.

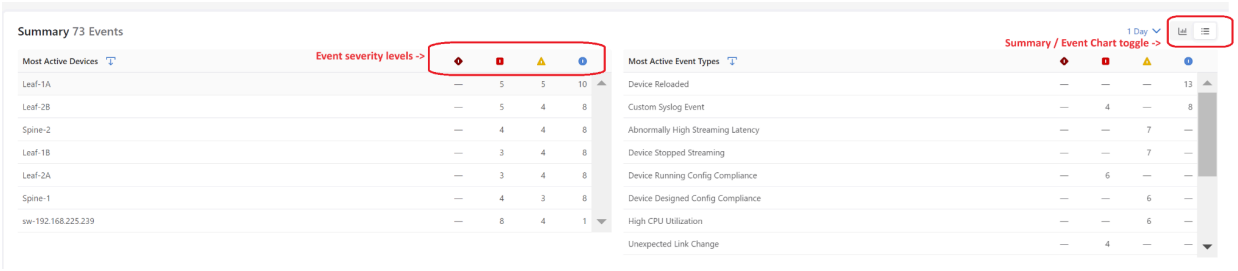
1. Navigate to the “**Events**” section of CloudVision




By default, CloudVision “**Events**” are graphically displayed for the selected time period using colors to denote the severity of the event. The time period selection can be changed using the drop down near the top right corner of the graph.

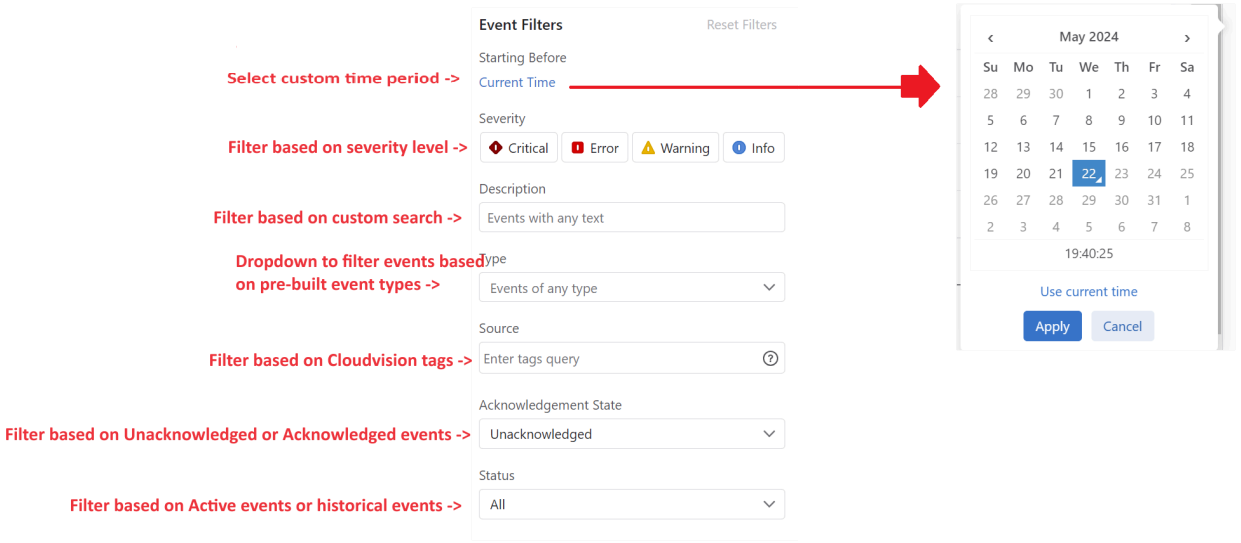



Swap views from the "**Event Chart"** to the "**Summary Tables**" using the toggle in the top right corner of the "**Summary Events"** pane. This allows you to view events by device and severity level for this time period, instead of viewing events by time and severity level.

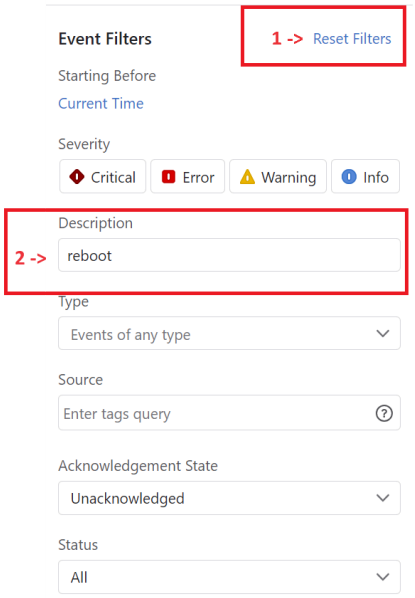



There are many ways to filter the viewable events in CloudVision by using the "**Event Filters**" on the right side of the screen:

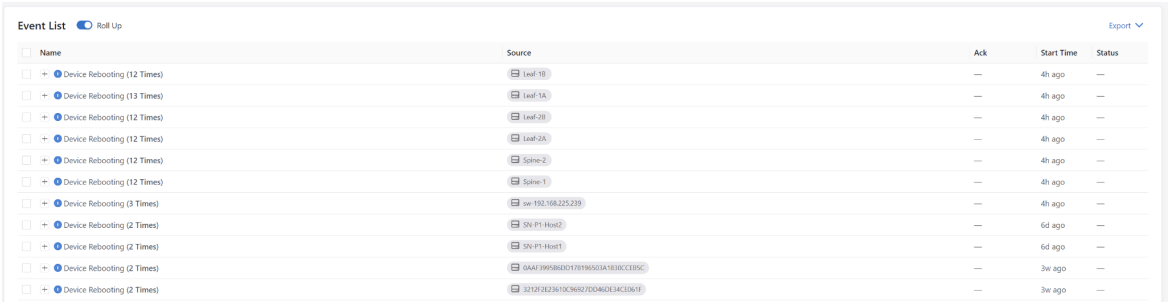
**Time**: Custom date and time periods can be selected instead of using only the current time.
**Severity**: Events can be filtered based on severity level.
**Description**: Events can be filtered based on matching text.
**Type**: Events can be filtered based on the pre-built event types within CloudVision.
**Tags**: Events can be filtered based on pre-built or custom user created Tags.
**State**: Events can be filtered based on whether they are Acknowledged or Unacknowledged.
**Status**: Events can be filtered based on whether they are currently Active or Ended.




2. Reset any existing filters, then filter based on "**reboot**" using the "**Description"** field.

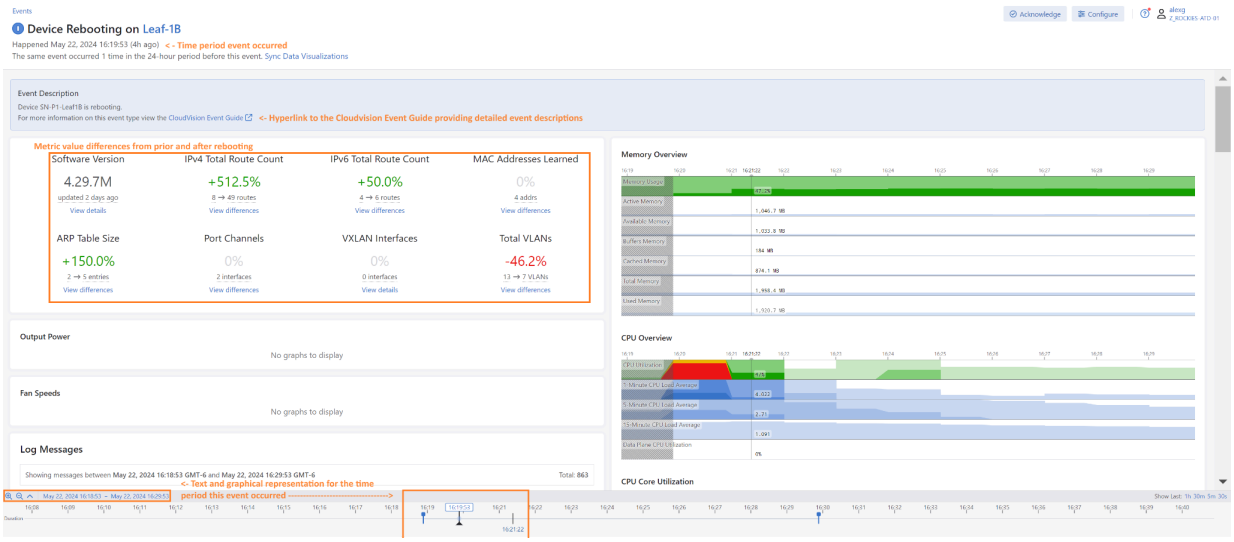


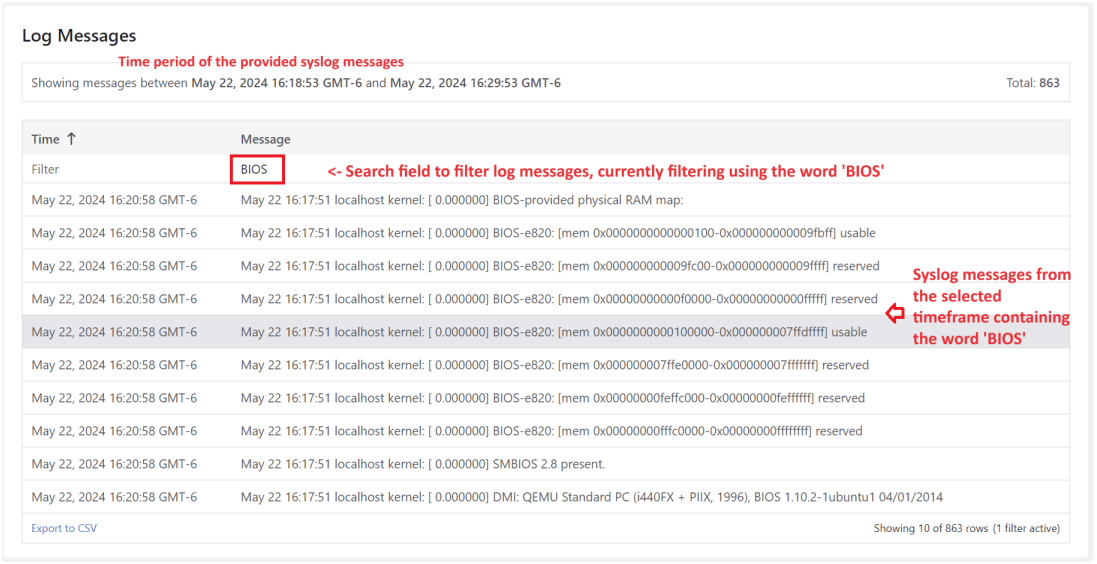

3. Click on one of the "**Device Rebooting**" events in the "**Event List**". (If your lab doesn't contain a Device Rebooting event yet, you can follow along using the screenshots in the next section)




Clicking on an event provides custom details based on the event type:

**Time period**: The exact time the event occurred is located at the top left. The overall time period from the event becoming active to when it ended is also displayed at the bottom of the screen.
**Metrics**: Based on the event type, specific metrics and graphs will be displayed providing more details
**Logs**: Syslog information from the device will be displayed and can be filtered based on search



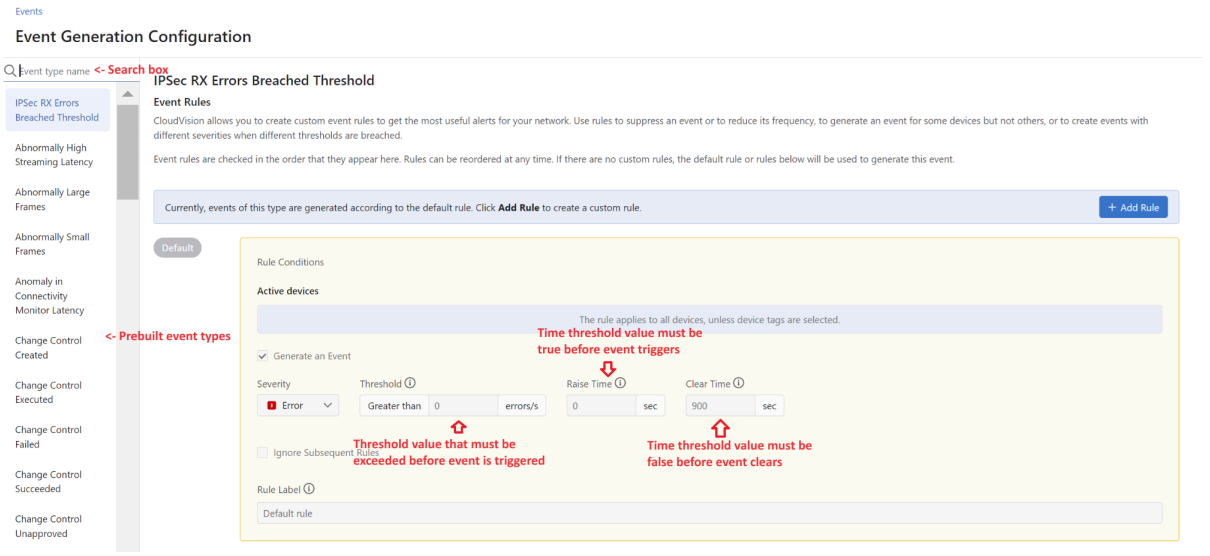



# CloudVision Event Generation

1. Navigate to “**Events**”, then click on the “**Event Generation**” button near the top right corner


The “**Event Generation**” page allows you to edit any of the pre-built events types within CloudVision. Let’s review some of the important fields:

**Threshold**: The value that must be reached before the event will trigger
**Raise Time**: The time period the threshold value must be met before the event will trigger
**Clear Time**: The time period the threshold value must be false before an Active event will end

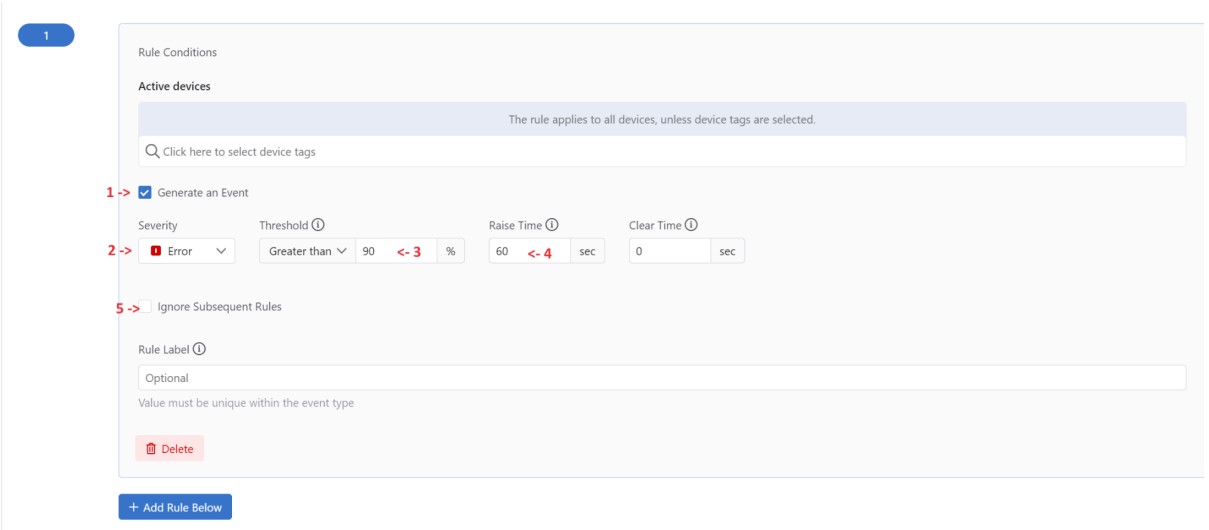



2. In the search field for “**Event type name**”, type “**CPU**” then click on “**CPU Utilization Breached Threshold**” event type

   The pre-built “**CPU Utilization Breached Threshold**” event type is set to trigger a **Warning** event if the CPU Utilization exceeds 85% for more than 30 seconds. We’re going to create an additional rule for this event that triggers an **Error** event for more serious situations where the CPU Utilization exceeds 90% for more than 60 seconds.

3. Click on “**Add Rule**”, then perform the following steps:

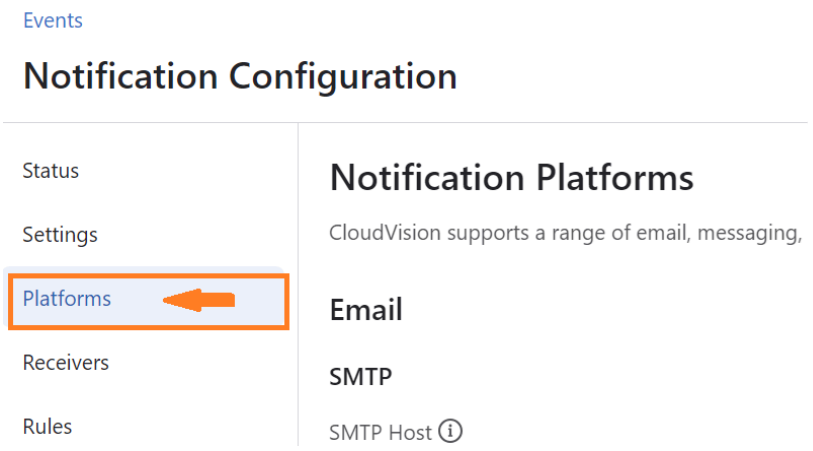
   a. Check the box for “**Generate an Event**”.
   b. Change the “**Severity**” level to “**Error**”.
   c. Set the “**Threshold**” value to “**90**”.
   d. Set the “**Raise Time**” value to “**60**”.
   e. Uncheck the box for “**Ignore Subsequent Rules**”.
   f. Click “**Save**” in the top right corner.




## CloudVision Event Notifications

1. Go back to the "**Events**" page, then click on the "**Notifications**" button near the top right


2. Select the "**Platforms**" option for "**Notification Configuration**" on the left hand side of your screen.



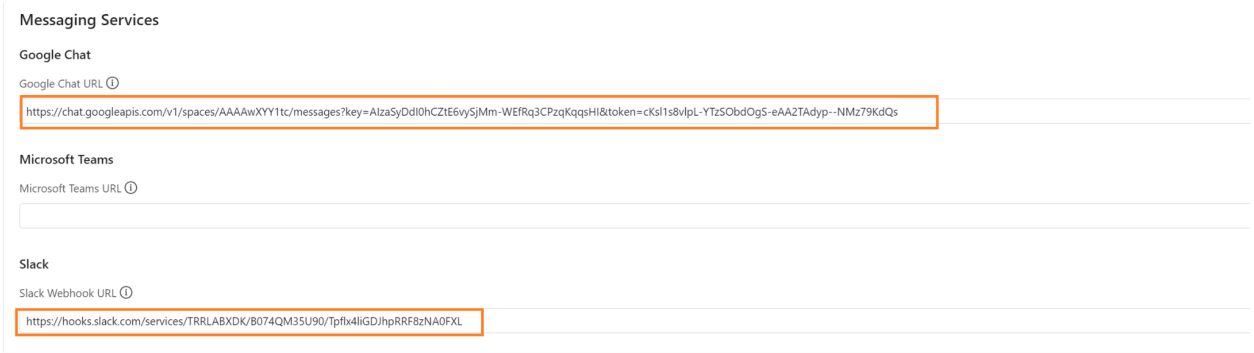

The "**Platforms**" configuration is where you can add additional notification platforms for CloudVision to send event notification alerts to when events are triggered. CloudVision

supports many different platforms such as **Email**, **Slack**, **Zoom**, and **PagerDuty**. Additional platform options are regularly added with each new CloudVision update.

Most of these can be added fairly easily via the webhook URL for the specific application platform in question. Let's add two additional platforms this way.

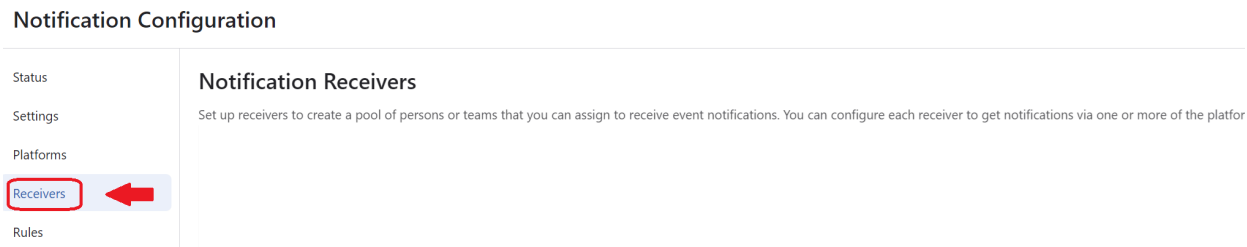
3. Add the following URL for the "**Google Chat**" webhook: https://chat.googleapis.com/v1/spaces/AAAAwXYY1tc/messages?key=AIzaSyDdI0hCZtE6vySjMm-WEfRq3CPzqKqqsHI&token=cKsl1s8vlpL-YTzSObdOgS-eAA2TAdyp--NMz79KdQs

4. Add the following URL for the "**Slack**" webhook: https://hooks.slack.com/services/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

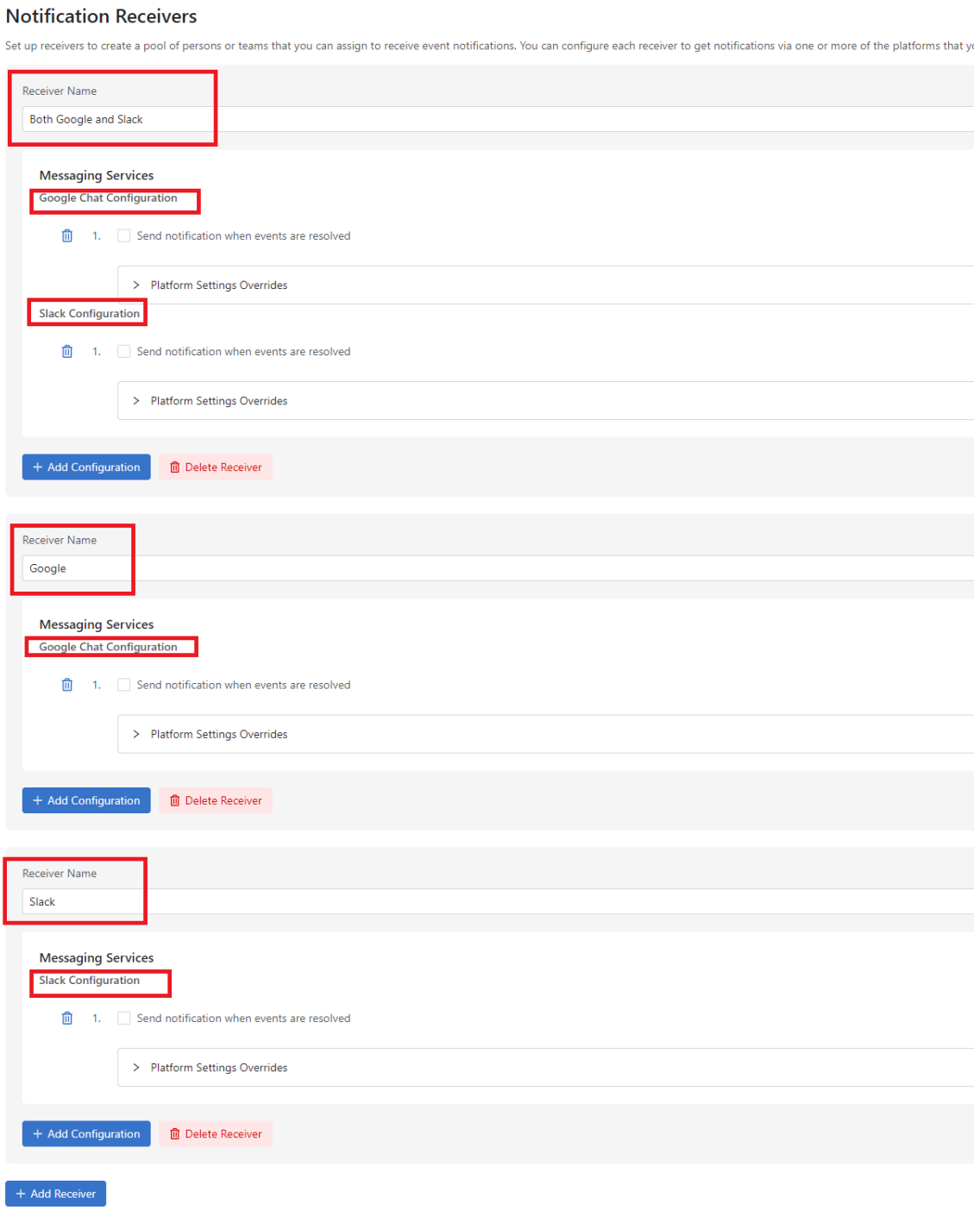



Now that we have added two new platforms for event notifications, we need to configure the "**Notification Receivers**" that we want to use for our event alerting.

5. Navigate to the "**Receivers**" sections in "**Notification Configuration**".

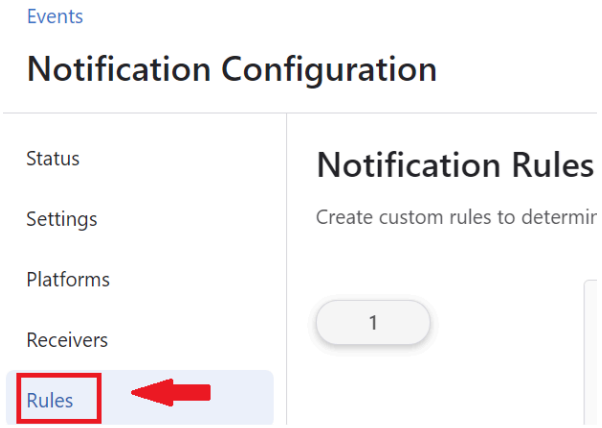


6. Click on “**Add Receiver**” and configure three different “**Notification Receivers**”. One for “**Google Chat**”, another for “**Slack**”, and a final one that includes **Both**. Click “**Add Configuration**” and select the corresponding items from the drop down menu for each receiver.



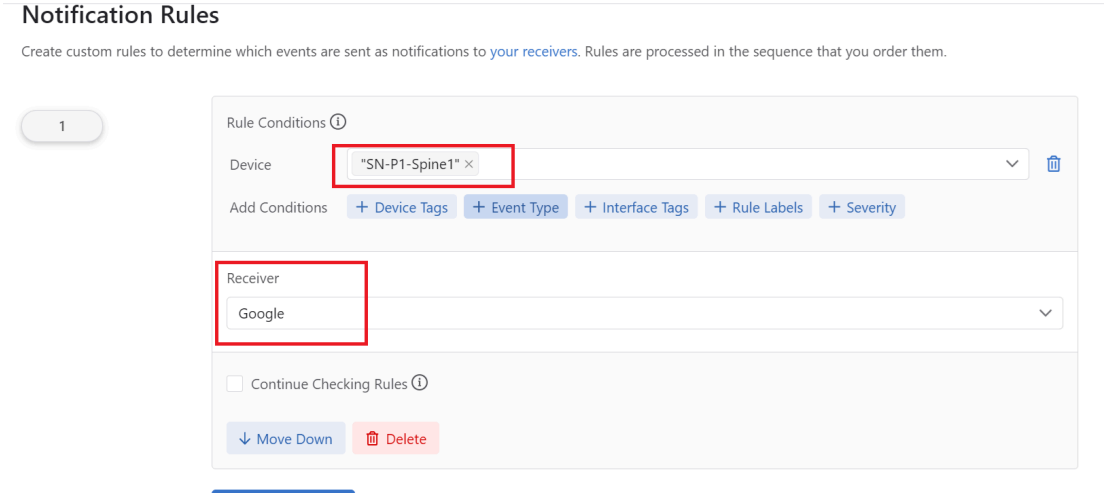
You've now added two new event "**Platforms**" - Google Chat and Slack, and you've created three different groups of "**Receivers**" - Google Chat, Slack, and both Google Chat + Slack. The final step is to add the event "Rules" for which "**Receiver**" groups will be used for different events.

7. Navigate to the "**Rules**" section of "**Notification Configuration**".

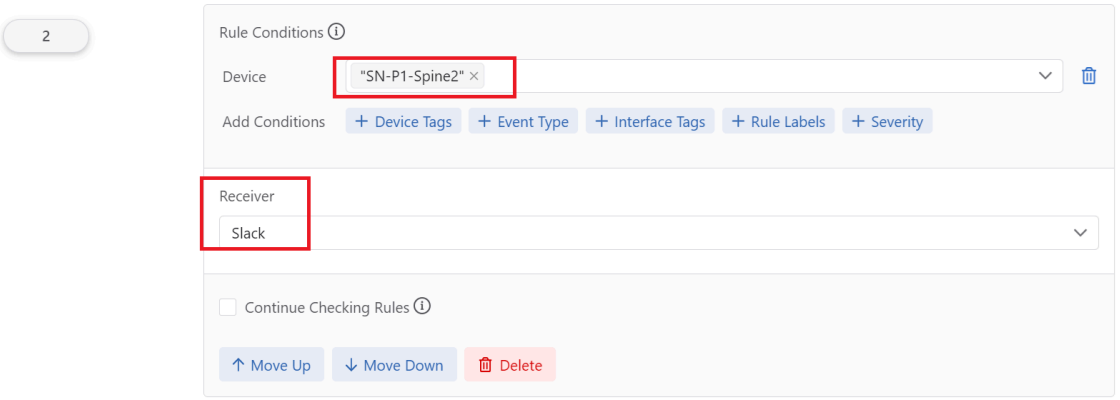



We will now configure three different "**Rules**" that use the "**Receivers**" we just created to highlight how this process works during testing later.

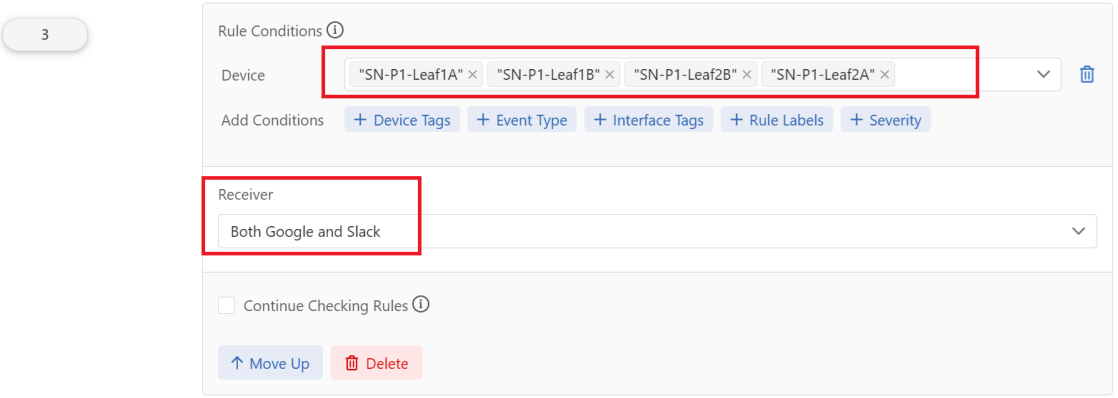
8. For the first rule, we are going to select "**Spine-1**" as our "**Device**", and "**Google**" as our "**Receiver"**.



9. For the second rule, we are going to select “**Spine-2**” as our “**Device**”, and “**Slack**” as our “**Receiver**”.



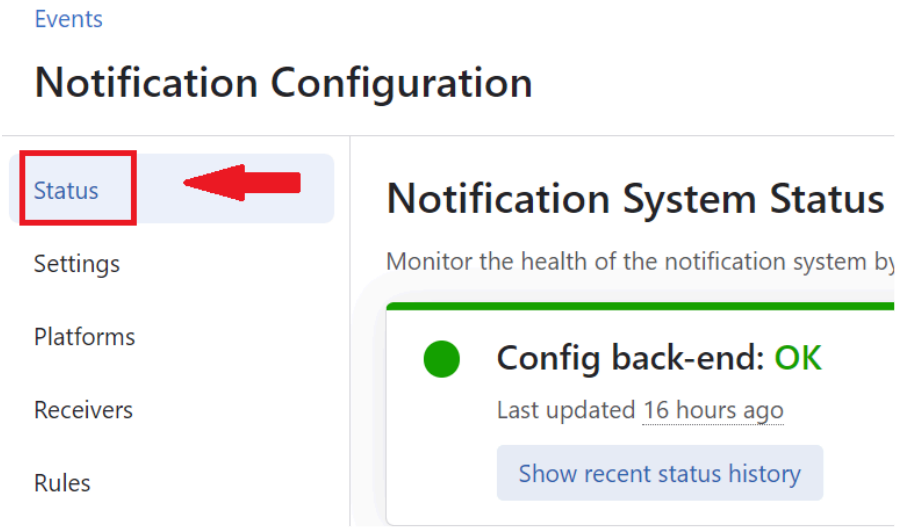

10. For the third and last rule, we are going to select all Leafs, and “**Both Google and Slack**” as our “**Receiver”**.



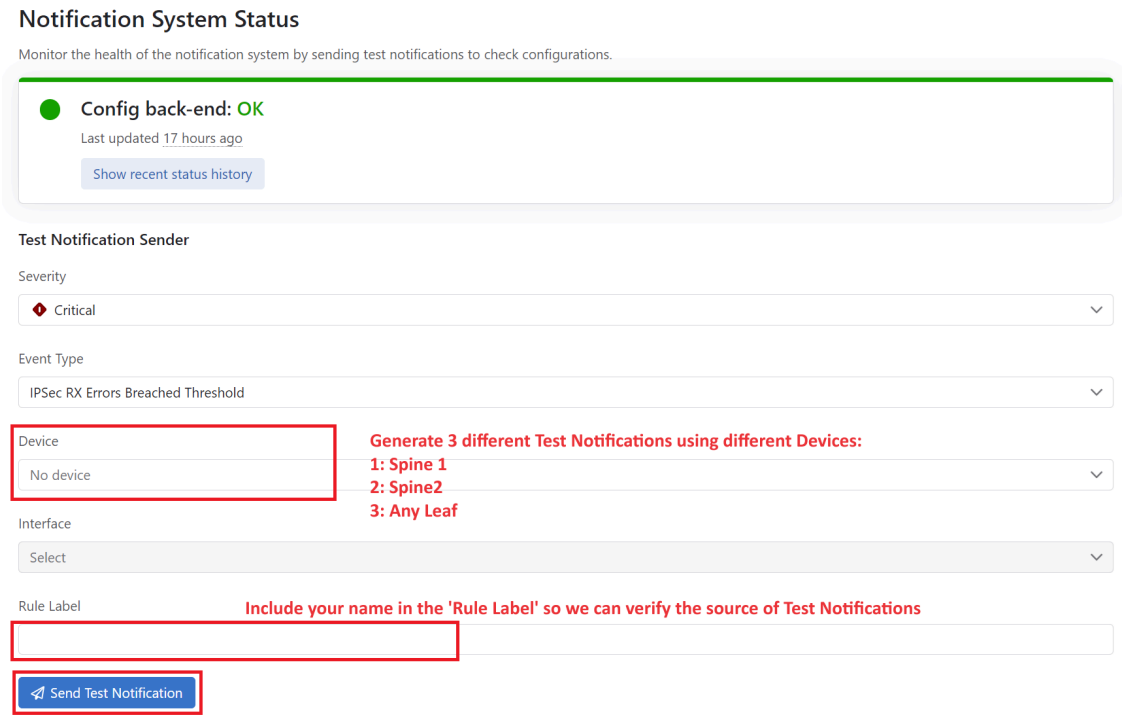

11. Click “**Save**” in the upper right hand corner.

    We’ve set up everything required for event notifications. We can now test if our notifications are working correctly using the “**Test Notification**” tool in the “**Status**” section of “**Notifications**”.

12. Navigate to the “**Status**” section of “**Notifications**”.




13. We will now generate 3 different test notifications for: “**Spine-1**”, “**Spine-2**”, and any **Leaf**. Include your “**<FirstName_LastInitial>**” in the “**Rule Label**’ field so you can identify your test notifications later in **Slack & Google Chat**.




14. The **Google Chat** and **Slack** notifications will be displayed by your instructors.

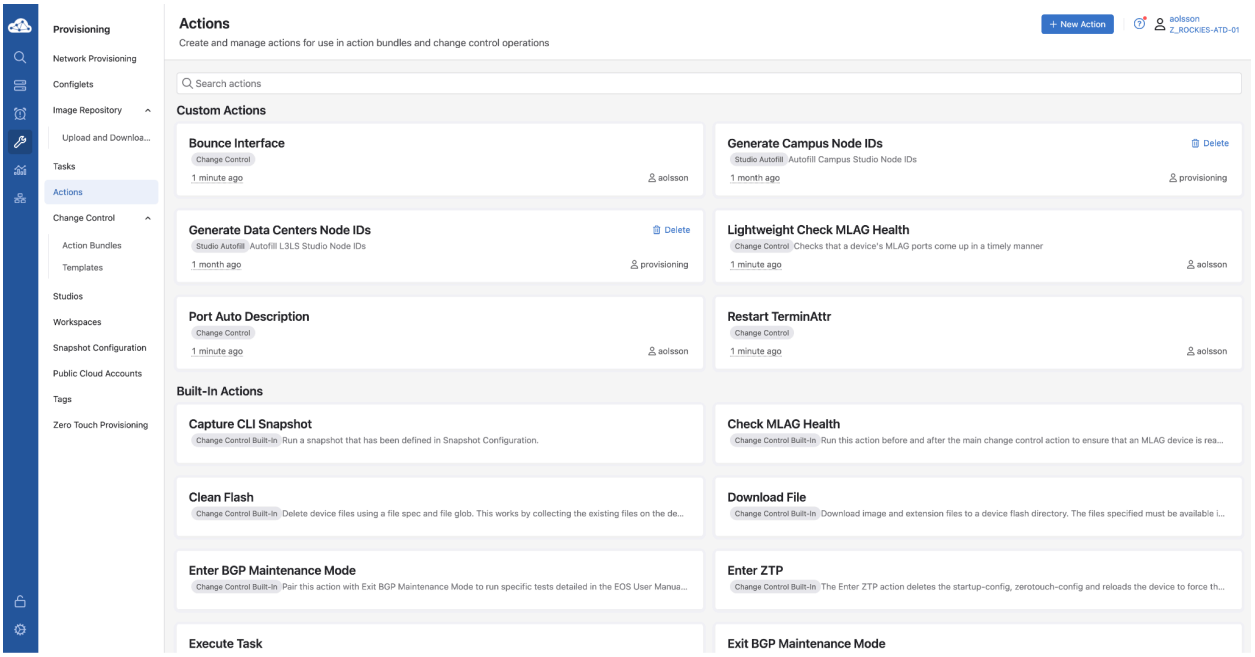
# ACTIONS

Actions can be used within a change control to run a series of commands on devices. There are a number of built-in actions that can be selected when configuring a change control operation or creating a change control template. You can also create custom actions by creating arguments and writing a Python 3 script to manipulate the arguments.

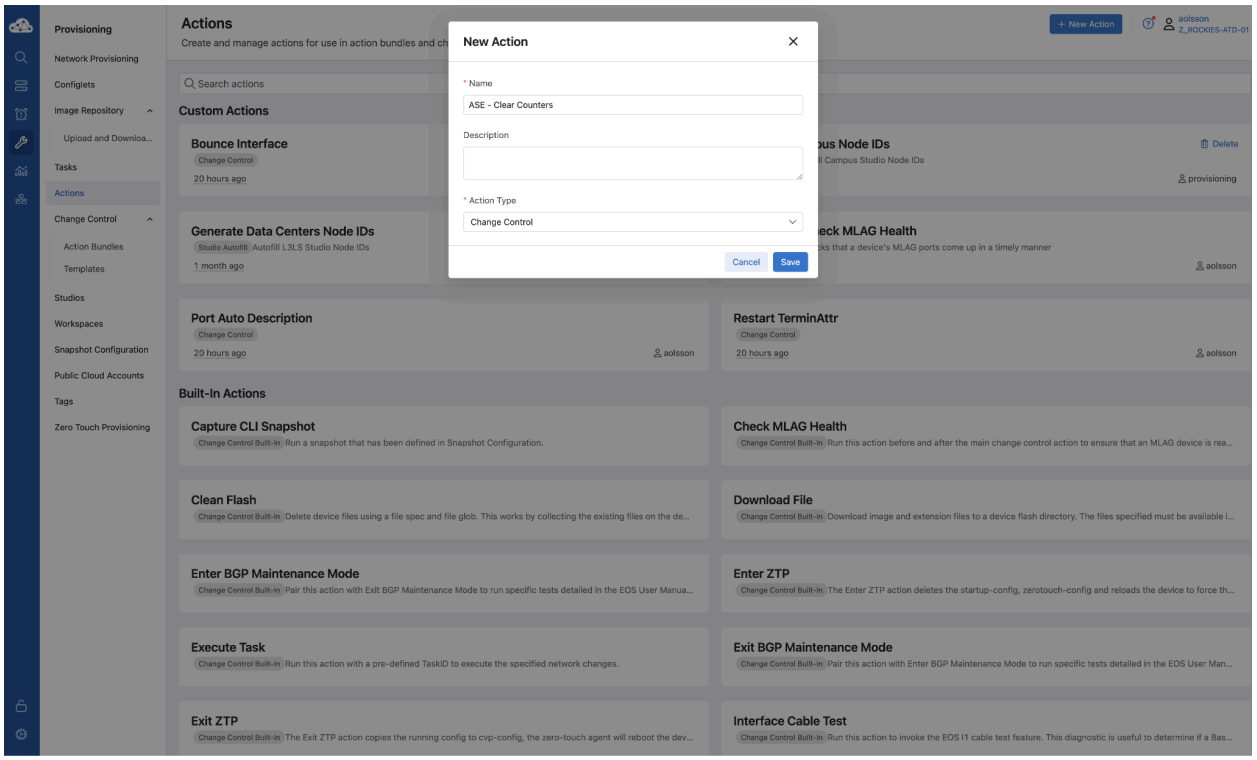
When configuring a change control, you'll select the dynamic values of an argument, like the device the action will run against and the image to install. When creating a custom action, you'll use dynamic arguments, like DeviceID, to enable users to select values when configuring a change control.

## Create an Action

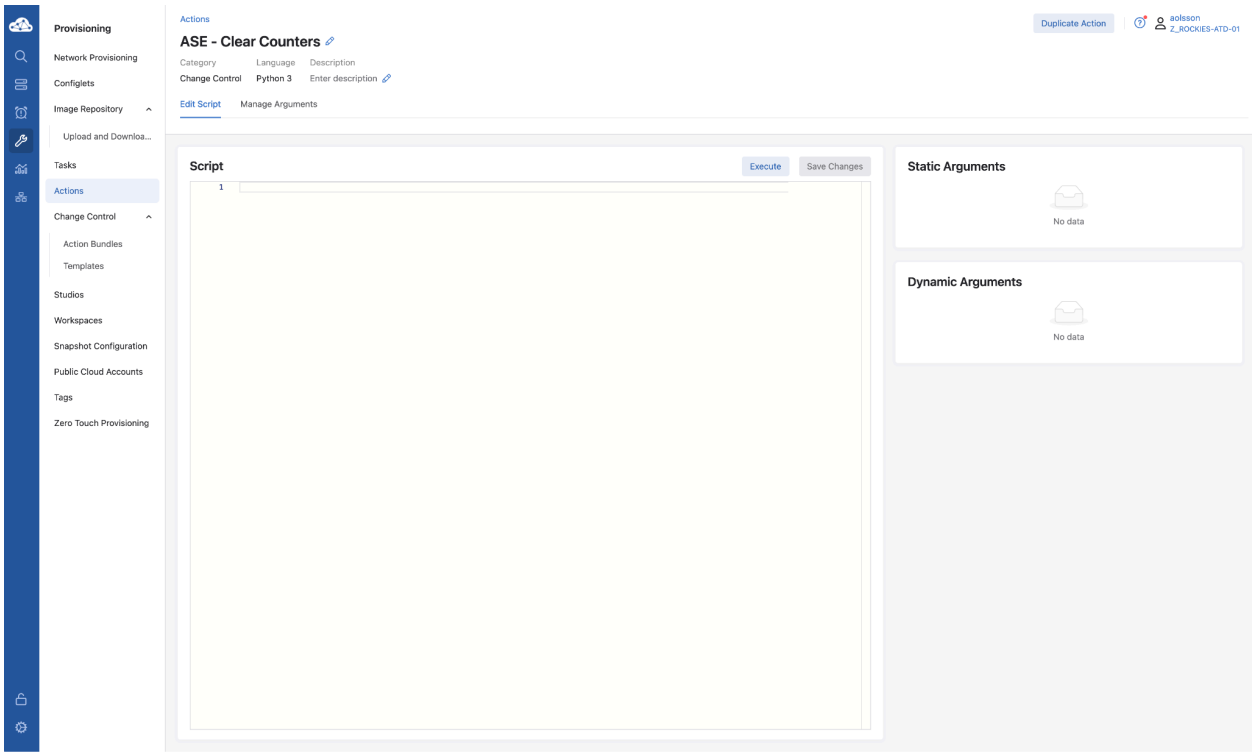
1. Navigate to **Provisioning** > **Actions.**
2. To create a new action, click on the "**+ New Action**" button at the top of the screen.



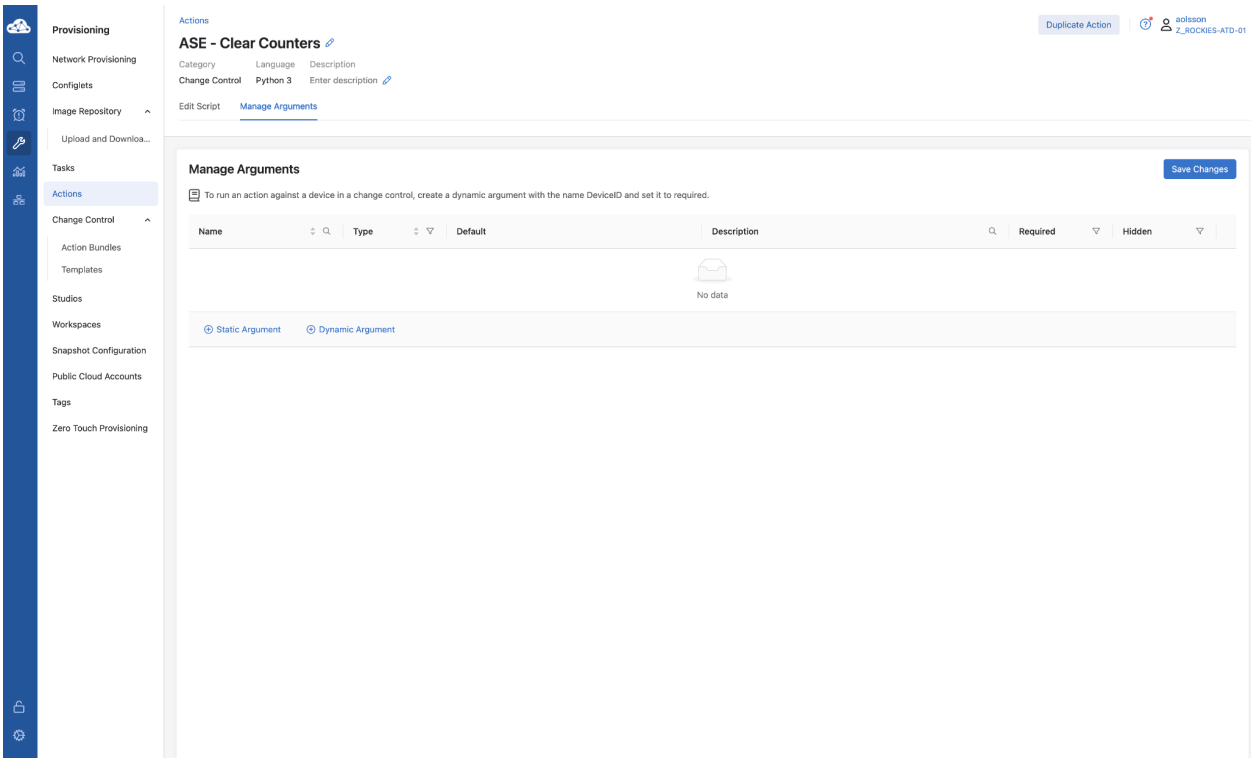

3. A "**New Action**" pop-up will appear. In the "**Name**" field, type your initials followed by "**- Clear Counters.**"
4. Click the blue "**Save**" button to save the new action.




5. You'll then be redirected to the action configuration page for the new action.



6. We'll first want to create some arguments for the Device and Interface for the action to be run against. To do this, click the "**Manage Arguments**" button just below the action's name

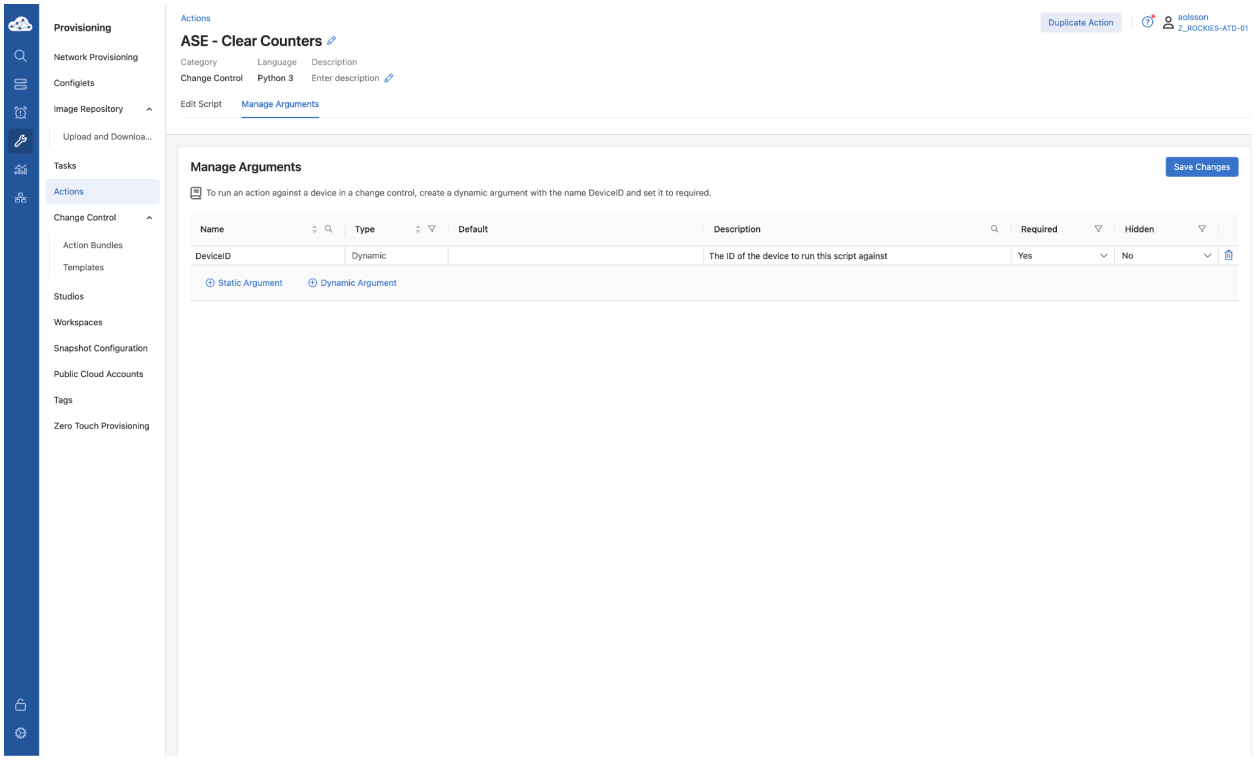



7. Create a new argument by selecting the "**+ Dynamic Argument**" at the bottom of the "**Manage Arguments**" section.

**NOTE**: Keep an eye on the Arguments and Variables as they are case sensitive.

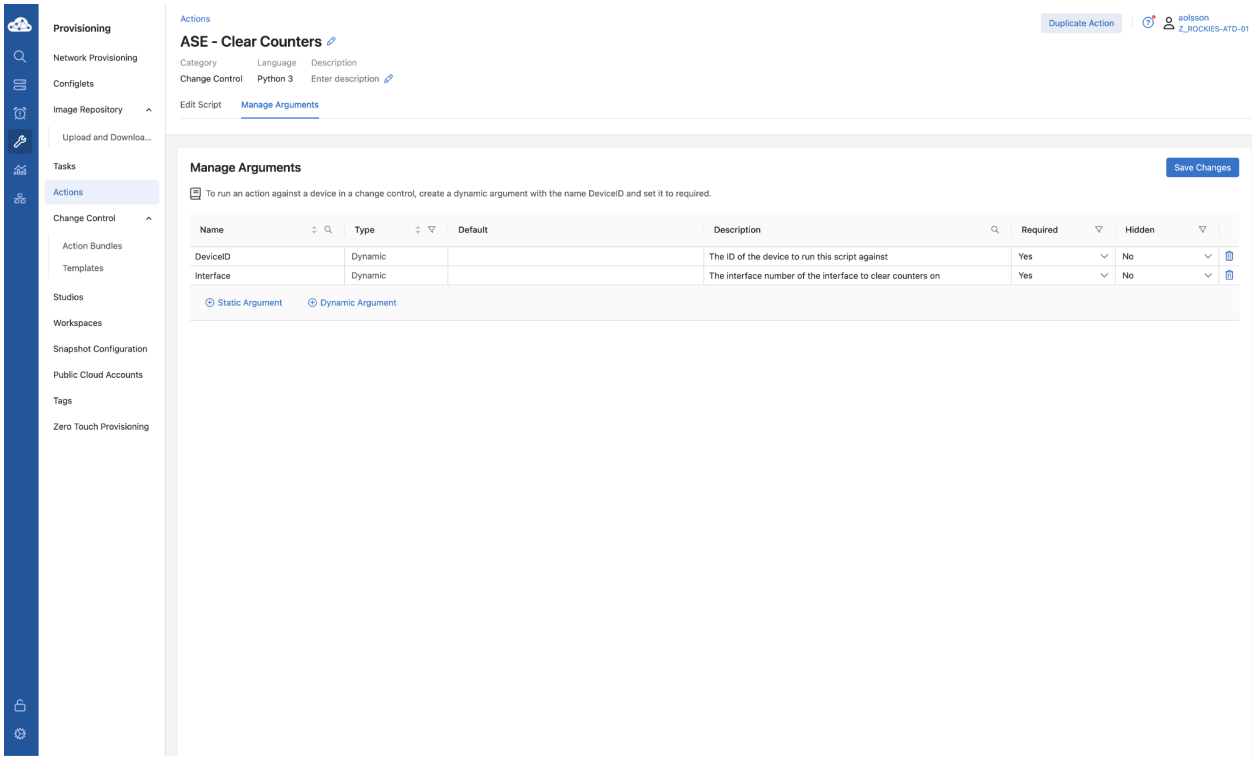
8. A new line for the new argument will appear in the "**Manage Arguments**" section. Enter the following into the new line:

```
Name : DeviceID
Type : Dynamic
Default : Leave Blank
Description : The ID of the device to run this script against
Required : Yes
Hidden : No
```

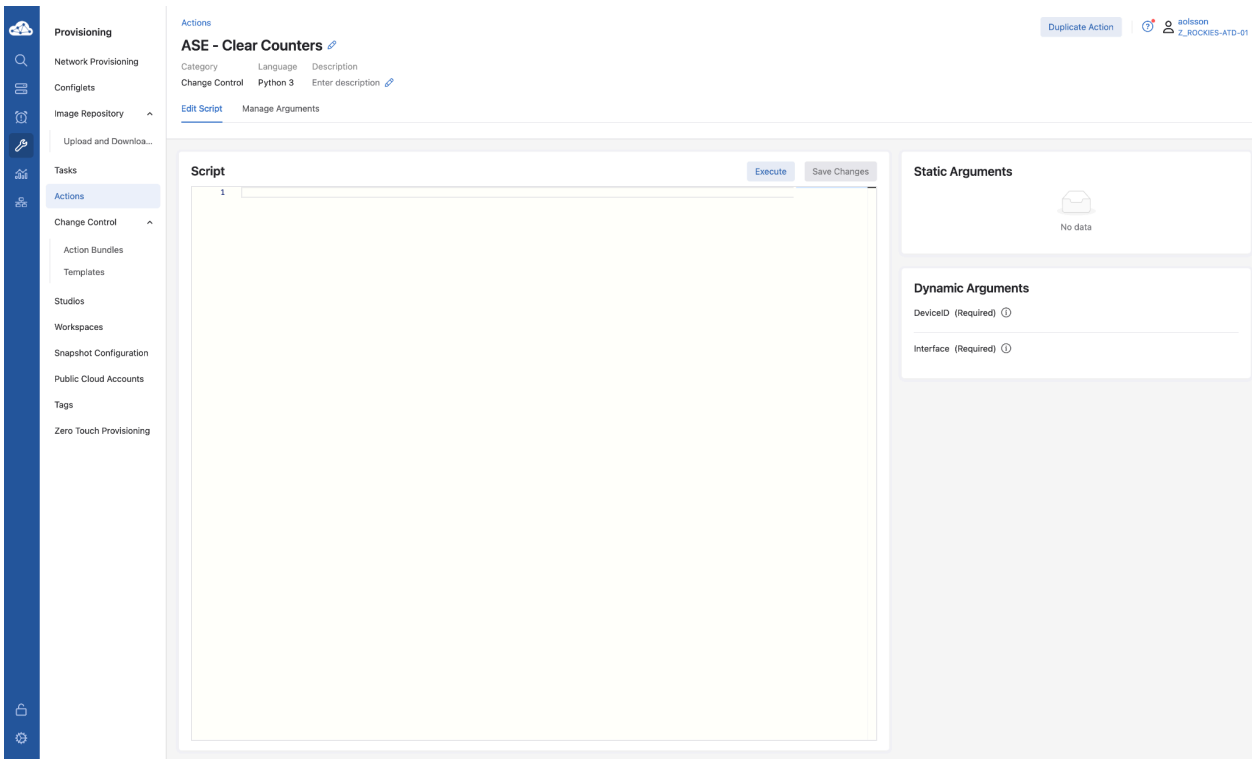



9. Repeat steps 7 & 8 to create the interface argument using the following values:

```
Name : Interface
Type : Dynamic
Default : Leave Blank
Description : The interface number of the interface to clear counters on
Required : Yes
Hidden : No
```




10. Click the blue “**Save Changes**” button on the right side of the screen.
11. Now that the Arguments are created, we can head to the Script section to begin writing the action. Click on the “**Edit Script**” button below the Action name.



12. On line 1 of the script we'll want to identify who wrote the script. Go ahead and enter the following on line 1, replacing "your-name-here" with your name:

```
# Written by your-name-here
```

13. The next thing we need to do is create a variable in the script that references the "**Interface**" argument that was created. To do that, we skip a line an then add the following on lines 3 & 4.

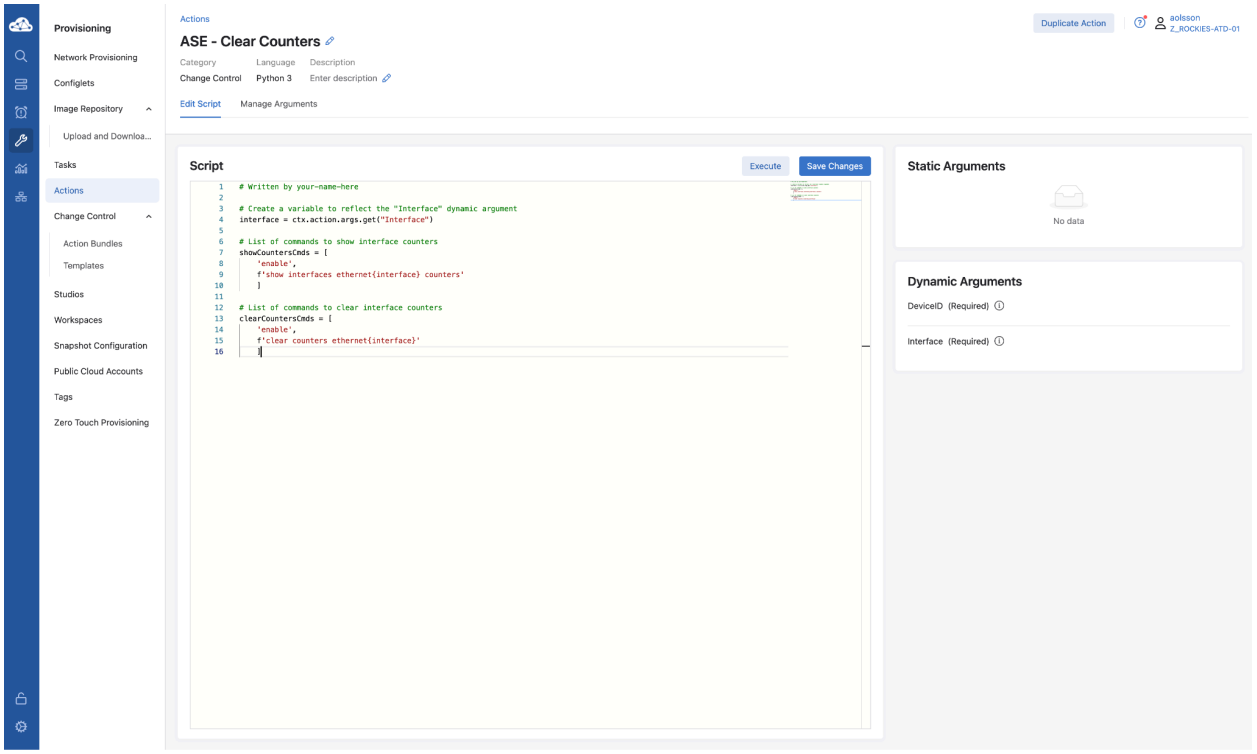
```
# Create a variable to reflect the "Interface" dynamic argument
interface = ctx.action.args.get("Interface")
```

14. Now, we need to add two different lists of commands that will need to be run on the switch. The first list will show the interface's existing counters, and the second will clear the interface counters. Skip a line and then paste the following into lines 6 - 16:

```
# List of commands to show interface counters
showCountersCmds = [
   'enable',
   f'show interfaces ethernet{interface} counters'
   ]

# List of commands to clear interface counters
clearCountersCmds = [
   'enable',
   f'clear counters ethernet{interface}'
   ]
```

15. Validate that the script section matches the following screenshot.




16. Let's continue editing the script by adding some logging stating that we're getting the values of the current interface counters. Skip another line and then add the following to lines 18 & 19:

```
# Create log entry
ctx.alog(f'Getting counters for Ethernet{interface}')
```

17. Now, we can proceed with running the commands listed in the "showCountersCmds" list on the switch. To do that, add another blank line and the following to lines 21 and 22.

```
# Run the list of commands to get counters
cmdResponse = ctx.runDeviceCmds(showCountersCmds)
```

18. The commands have now been run on the switch and the response has been stored in the variable "**cmdResponse.**" Let's create a log entry that outputs the response. On lines 24 - 25 add the following:

```
# Create a log entry of the counters
ctx.alog(f'The currect interface counters are : {cmdResponse[1]["response"]}')
```

**NOTE**: The cmdResponse value returned above is similar to the following:

```
[{'error': '', 'response': {}},
{'error': '', 'response': {
   'interfaces': {
      'Ethernet1': {
         'inBroadcastPkts': 1,
         'inDiscards': 0,
         'inMulticastPkts': 38,
         'inOctets': 7529,
         'inUcastPkts': 0,
         'lastUpdateTimestamp': 1717697811.8986592,
         'outBroadcastPkts': 836, 'outDiscards': 0,
         'outMulticastPkts': 37,
         'outOctets': 45227, 'outUcastPkts': 0}
      }
   }
}]
```

The cmdResponse list contains a dictionary for each command run on the device. Since we're interested in the "show interfaces ethernet counters" command response value, we need to specify "**[1]**" to indicate the second item in the list (since list values start at the 0th item in Python) and "**["response"]**" for the response data.

19. Now that we have logged what the counters were, let's proceed with clearing the counters. Let's create a blank line, then create another log entry stating that we will clear the counters. On lines 27 - 28 enter the following:

```
# Create log entry before clearing the counters
ctx.alog(f'Clearing counters for Ethernet{interface}')
```

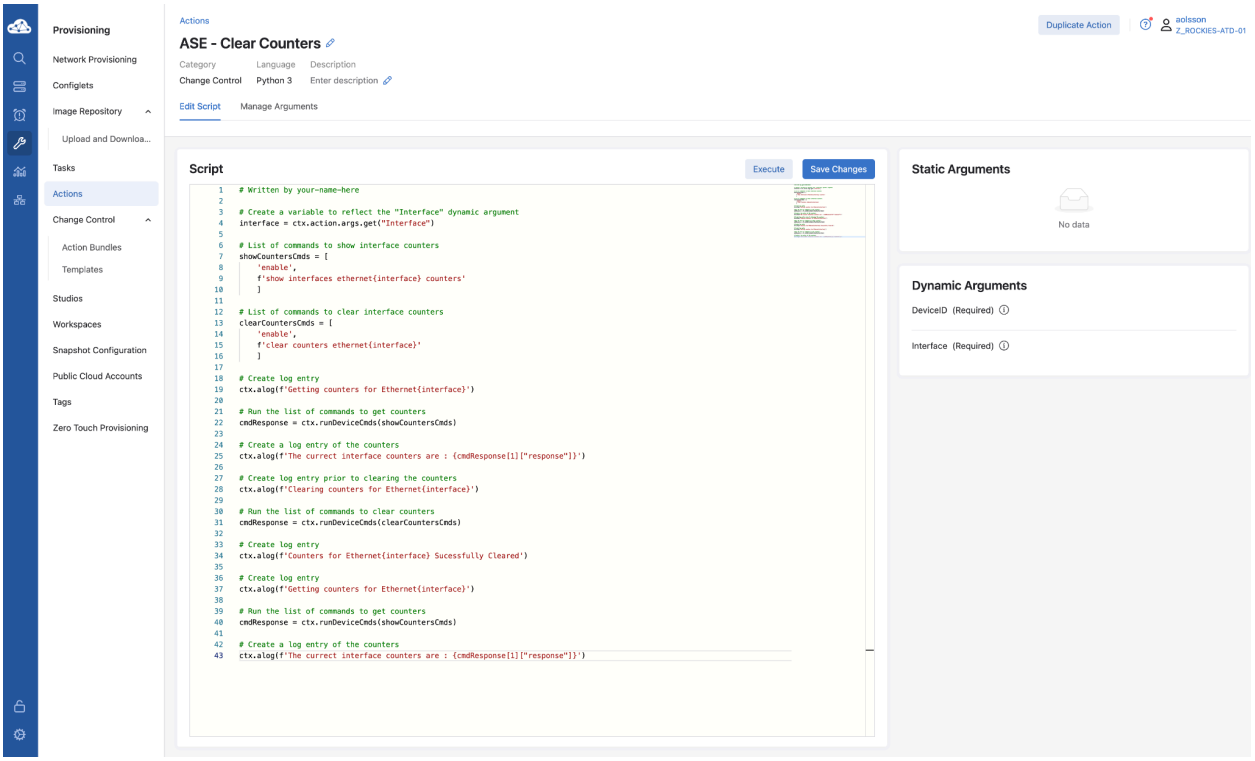
20. Let's actually clear the counters now. Let's skip another line and then enter the following on lines 30 - 31:

```
# Run the list of commands to clear counters
cmdResponse = ctx.runDeviceCmds(clearCountersCmds)
```

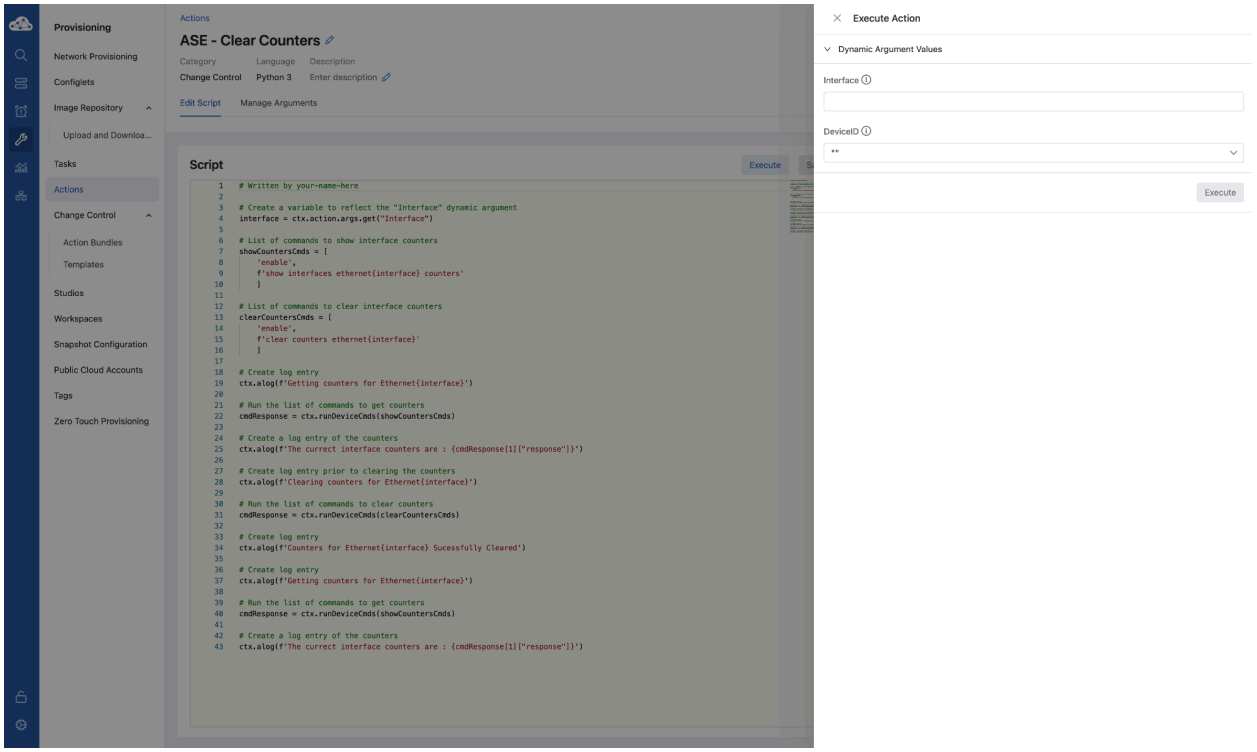
21. Now that the counters are cleared let's create another log entry stating that the counters have been successfully cleared. To do that, let's skip another line and add the following content to lines 33 - 34:

```
# Create log entry
ctx.alog(f'Counters for Ethernet{interface} Sucessfully Cleared')
```

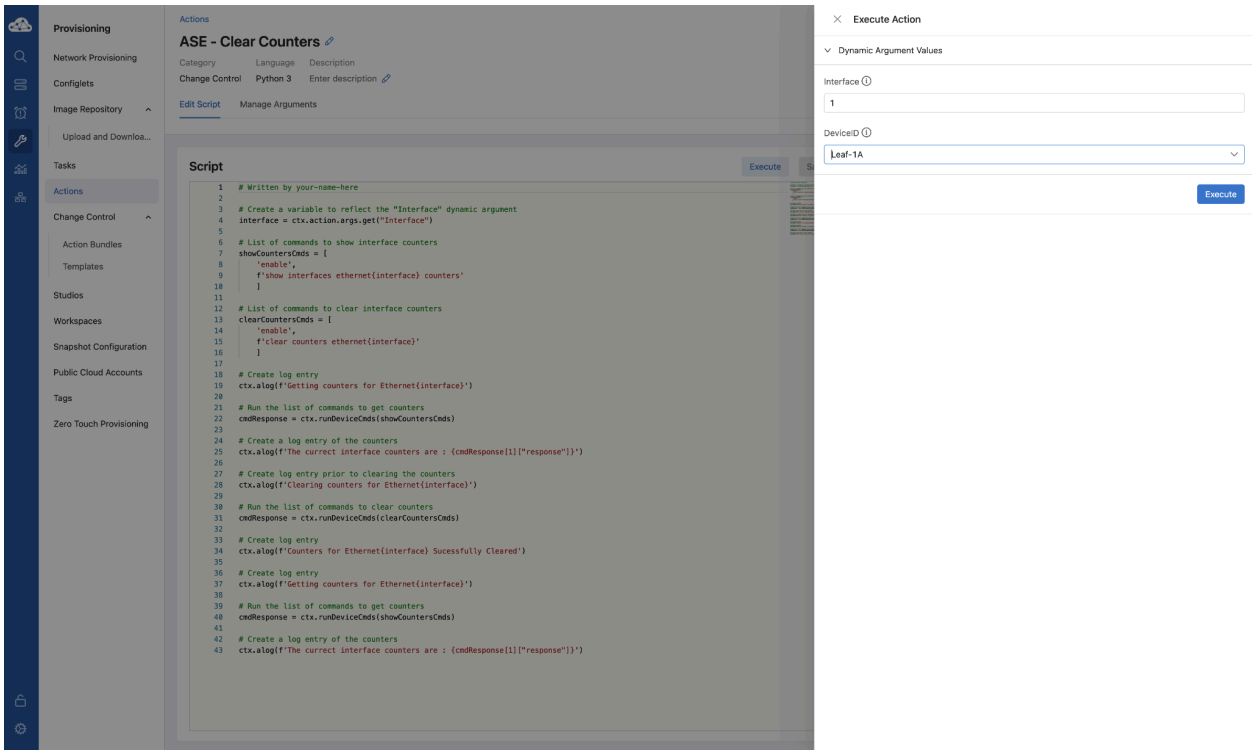
22. For the final part of the script, let's repeat steps 16 - 18 to get the interface's current counter values and log them.
23. Validate that the script matches the screenshot below, then click the blue "**Save Changes**" button to save the action.



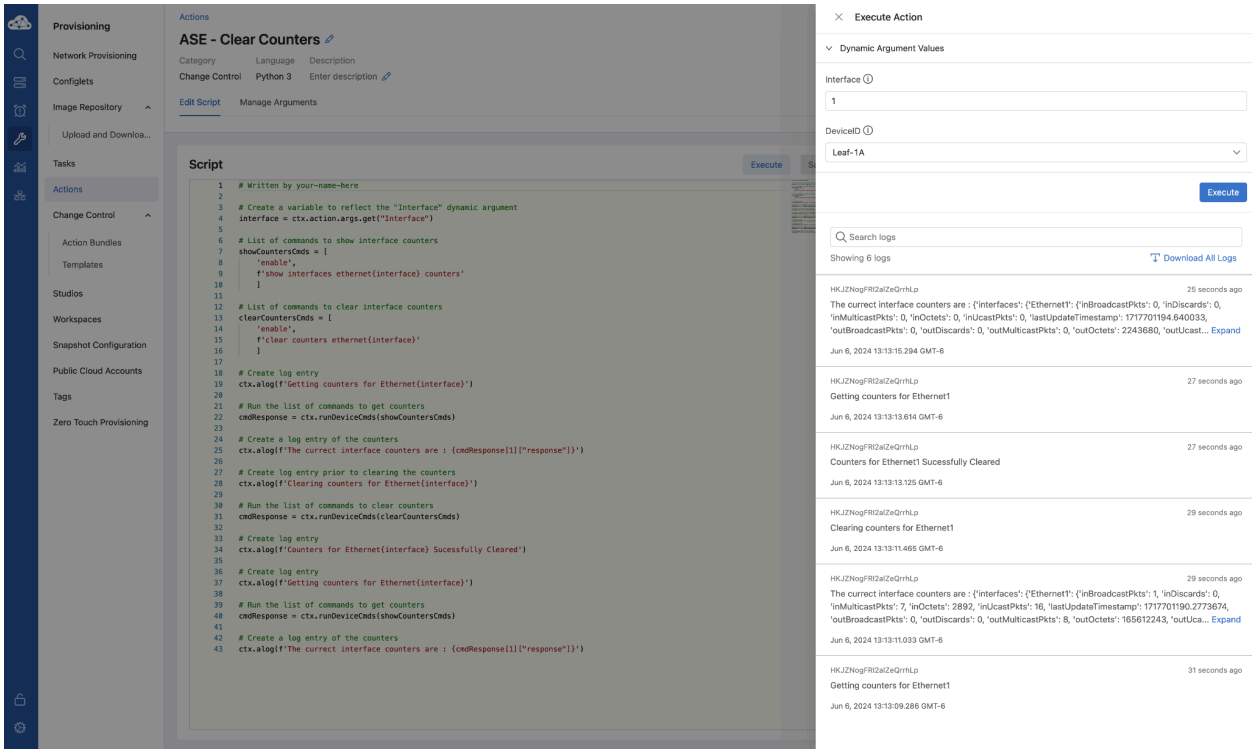

24. Now, we can proceed with testing our Action. To do so, click on the light blue "**Execute**" button.
25. An "**Execute Action**" dialog box will appear on the right side of the screen. Click "**Dynamic Argument Values**" to expand the arguments section.




26. Let's use Ethernet1 for this test, so specify "**1**" for the "**Interface**."
27. In the DeviceID dropdown, select "**Leaf-1A**"
28. Next, click the blue "**Execute**" button.



29. As the action executes, it will provide the logging output in the pane below the "**Execute**" button. The oldest log entry will be at the bottom, so you should read the logs from the bottom up. You'll be able to see that the current interface counters were displayed, they were then cleared, and the new counters were displayed.




30. Now, your custom action is available to be used within a change control.

**NOTE**: Check out this document for more information on custom actions. https://www.arista.com/en/support/toi/cvp-2021-3-0/14901-ui-for-custom-action-scripts
Arista also has a repo with some custom actions you might find useful. That repo can be accessed here: https://github.com/aristanetworks/cloudvision-python-actions