Linux Academy

Amazon Web Services

Certified DevOps Engineer – Professional Level

Auto Scaling

Auto Scaling can play a major role in deployments, so understanding how it behaves with different methods of deployment is very important

When using Auto Scaling, even with other services, it's important to avoid the downtime of an instance during a deployment process

Auto Scaling

Considerations when launching an application using Auto Scaling:

1. How long does it take to deploy the code and configure an instance?
   - A healthy instance may not necessarily indicate that the application is ready to serve traffic

2. How do you test new launch configurations?

3. How would you deploy new launch configurations while phasing out older ones?

Auto Scaling

Important terms to know:

- Scale out:
  - Refers to when Auto Scaling responds to an event by launching new instances
- Scale in:
  - Refers to when Auto Scaling responds to an event by terminating instances


Linux Academy
Amazon Web Services
Certified DevOps Engineer – Professional Level
Linux Academy

Auto Scaling Termination Policies

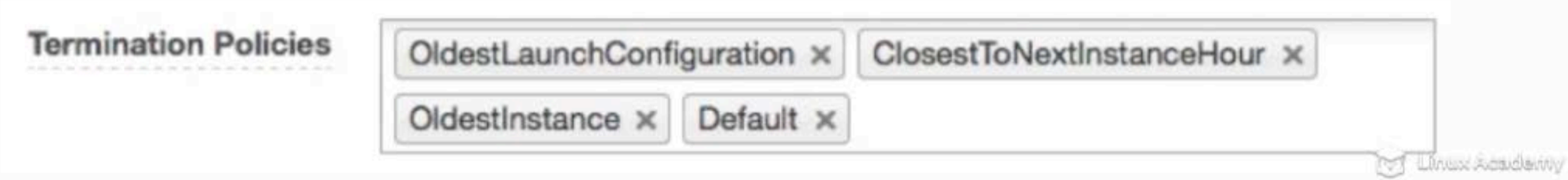
Termination policies are used to determine which instances should be terminated when there is a scale in event. This makes them important to understand for deploying or updating our environment.

Termination policies answer the questions: **Which instance should be terminated, and why?**

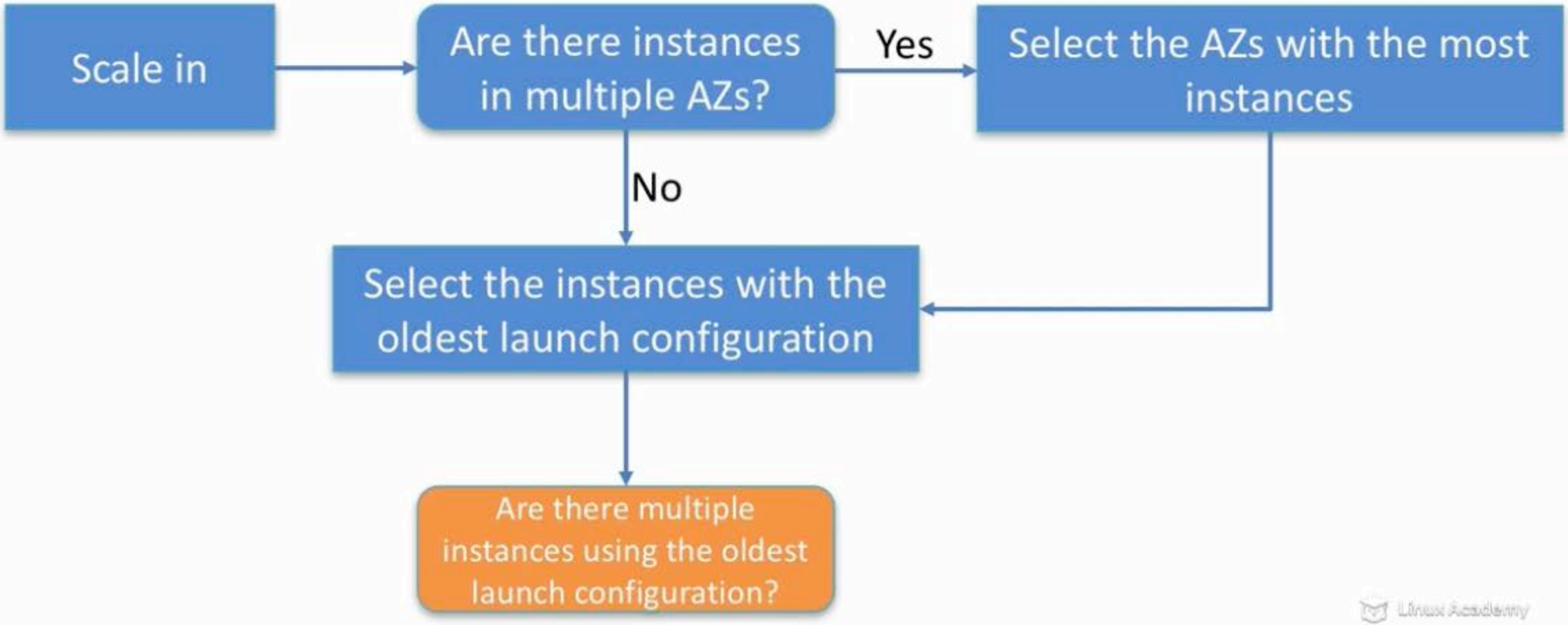
Configuring Termination Policies

Termination policies are configured in the Auto Scaling group configuration settings

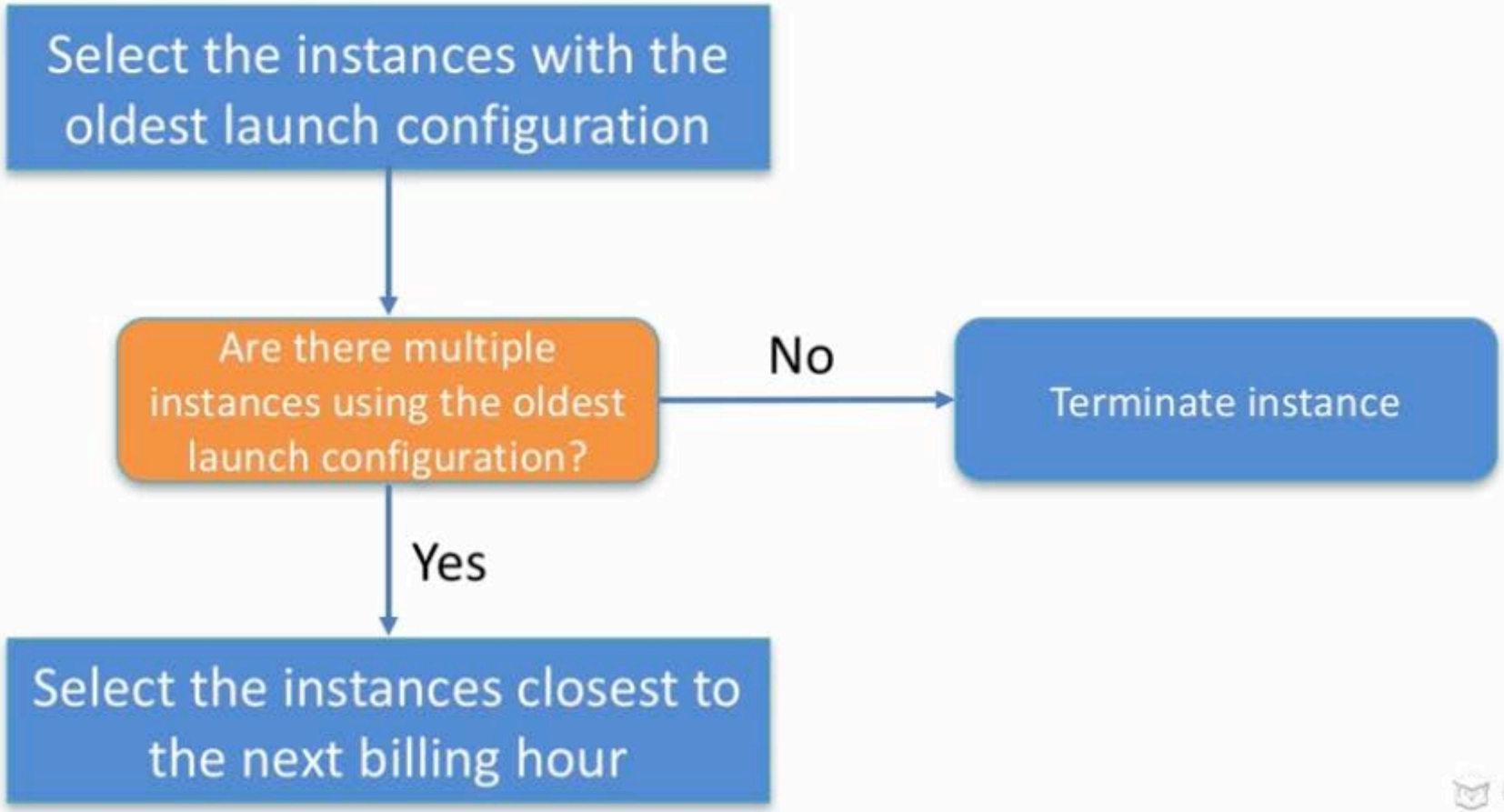
- **Note:**
  - Policies are executed in order
  - If the evaluated policy does not find a suitable instance to terminate, the next policy in line gets evaluated until a match is found



Default Termination Policy



Default Termination Policy



What is a Launch Configuration?

A Launch Configuration is a template used by an Auto Scaling group to launch EC2 instances

- This is where we specify:
  - An AMI
  - Instance type
  - Key pair
  - Security Groups
  - Block device mapping

Which Termination Policy should we use?

- OldestInstance
  - Useful when changing to a different instance type
- NewestInstances
  - Useful when testing an instance or a new launch configuration
- OldestLaunchConfiguration
  - This option is useful to phase out an older launch configuration
- ClosestToNextInstanceHour
  - Helps reduce costs

Important Termination Policy tips

- Protecting instances from scale in prevents them from being terminated, unless:
  - We manually terminate the instance
  - The instance is marked as unhealthy and removed from the group

- Scale in protection can be set on:
  - An entire Auto Scaling group
  - An individual instance


Linux Academy
Amazon Web Services
Certified DevOps Engineer – Professional Level
Linux Academy

Suspending Auto Scaling Processes

- Auto Scaling processes:
  - Launch
  - Terminate
  - HealthCheck
  - ReplaceUnhealthy
  - AZRebalance
  - AlarmNotification
  - ScheduledActions
  - AddToLoadBalancer

Auto Scaling gives us the option to suspend processes that Auto Scaling can perform

Suspending Auto Scaling Processes

- Launch process
  - Adds a new EC2 instance to the Auto Scaling group to increase capacity
  - This process could get executed to match desired capacity or as a response to a scale out event
  - Suspending this process disrupts other processes

- Terminate process
  - Removes an EC2 instance from the group to decrease capacity
  - This process could get executed to match desired capacity or as a response to a scale out event

Suspending Auto Scaling Processes

- HealthCheck process
  - Checks the health of an instance and marks it as unhealthy if Auto Scaling or the ELB reports it as unhealthy
  - Overrides manually setting the health check

- ReplaceUnhealthy process
  - Works with the HealthCheck process and uses the Terminate and Launch processes to replace unhealthy instances

- AZRebalance process
  - Balances the number of EC2 instances across Availability Zones

Suspending Auto Scaling Processes

- AlarmNotification process
  - Receives notifications from CloudWatch alarms
  - Suspending this removes the ability to execute policies that normally get triggered by alarms (like scale out or scale in event triggers)

- ScheduledActions
  - This process performs scheduled actions that we create
  - ie: Scale out Tuesday through Thursday, and scale in Friday through Monday

Suspending Auto Scaling Processes

- AddToLoadBalancer process
  - Adds instances to the load balancer (or target group) when they are launched
  - Suspending this can be useful for testing new instances before sending traffic to them, while still keeping them in our Auto Scaling group
  - Note: Resuming this process does not add the instances to the Elastic Load Balancer unless we manually add them

Suspending Auto Scaling Processes

- How can you suspend Auto Scaling Processes?
  - API, SDK, or CLI calls – or even from the console
  - You can suspend one or more processes at a time

Creating custom health checks

- You can create custom health checks via the API
- Using custom health checks can give more accurate checks for your application
- We can update the health status of an instance to Unhealthy, and Auto Scaling will clean it out of rotation


Linux Academy
Amazon Web Services
Certified DevOps Engineer – Professional Level
Linux Academy

Auto Scaling Lifecycle Hooks

Deploying code and applications can often take a lot of time. You do not want instances serving traffic until the application is ready.

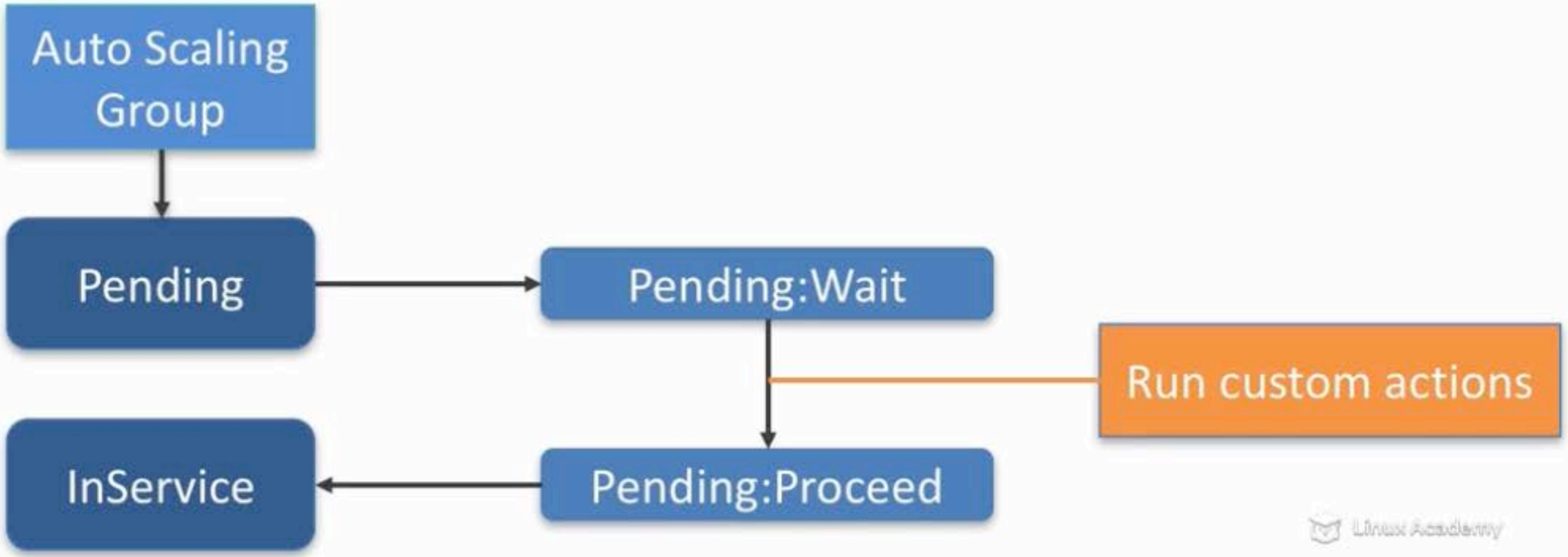
What we know:

- If an instance is seen as healthy, then the load balancer will send it traffic
- An instance does not necessarily wait for the application to be ready before registering as healthy

How can we solve this problem? Lifecycle Hooks

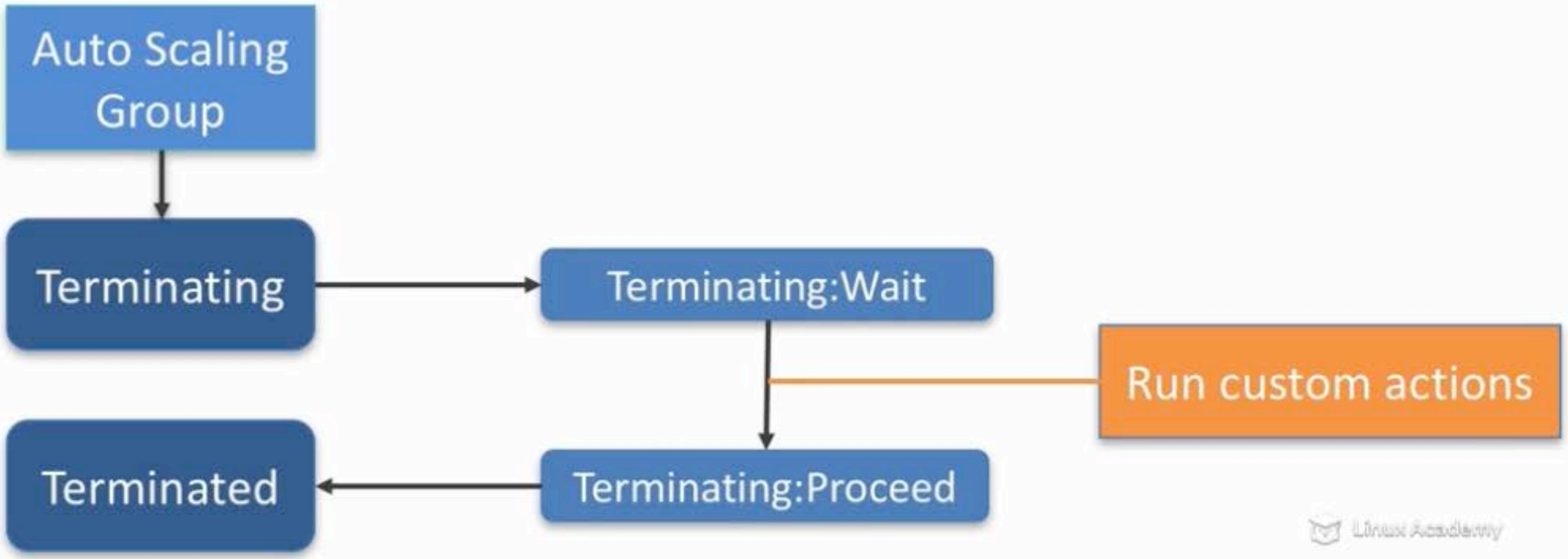
What are Lifecycle Hooks?

Lifecycle hooks give us the ability to perform actions before an instance is in the *inService* state or before an instance is in the *Terminated* state.



What are Lifecycle Hooks?

Lifecycle hooks give us the ability to perform actions before an instance is in the *inService* state or before an instance is in the *Terminated* state.

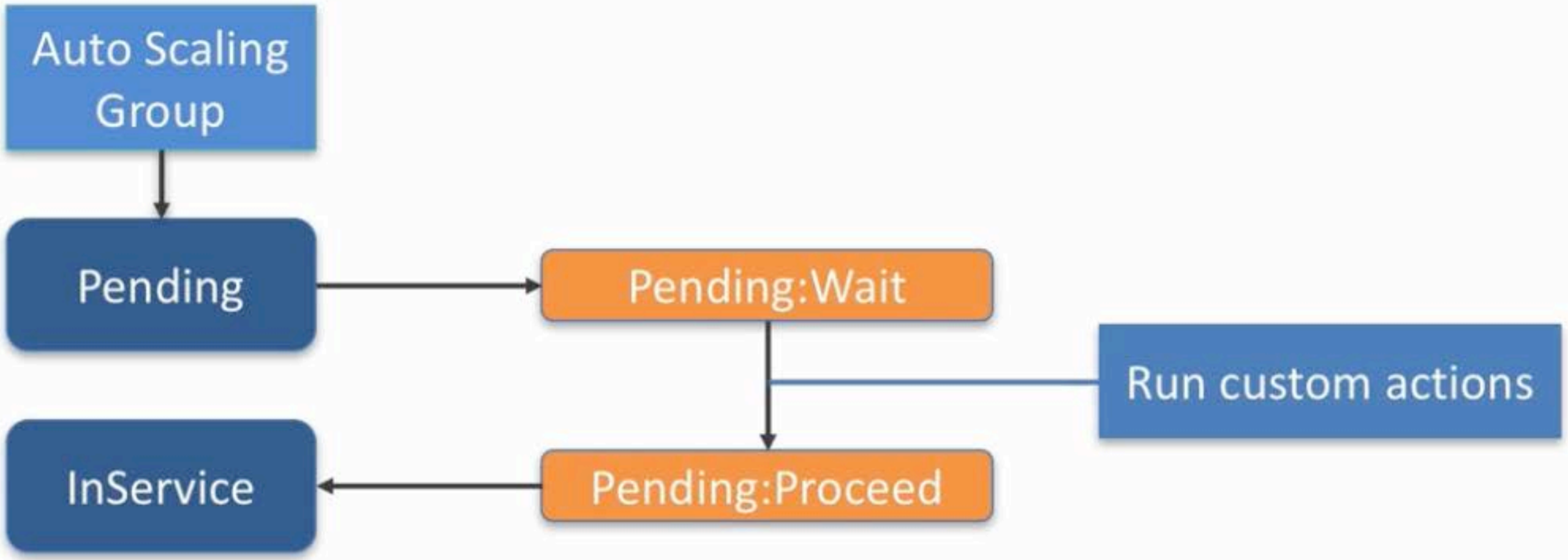


What Lifecycle Hooks options do we have?

1. Using CloudWatch events to invoke a Lambda function
   - Auto Scaling submits an event to CloudWatch events
   - The event invokes a Lambda function
   - The Lambda function gets information about the instance and a special token to control the Lifecycle action

2. Using a notification target for the Lifecycle Hook
   - Useful for sending messages to endpoints like Amazon SNS or SQS

3. Run a script on the instance as the instance starts
   - The script receives the instance ID to control the Lifecycle action



The Lifecycle Wait State

- The default wait state is 60 minutes
- If the action finishes sooner, you can complete it so that the instance can move on to the next step
  - **complete-lifecycle-action** CLI call
- If we need more time to complete the action than the wait state timeout gives us, we can submit a *Heartbeat*
  - **record-lifecycle-action-heartbeat** CLI call

Cooldowns with Lifecycle Hooks

Using Lifecycle Hooks can increase the time it takes for an instance to be ready to serve traffic

This can cause Auto Scaling to think it needs to scale in or scale out again to reach the desired capacity, which is not what we want

Use cooldown settings to solve this problem