Linux Academy

Live! Lab

Auto Scaling Lifecycle Hooks

Contents

Related Courses

AWS Certified DevOps Engineer - Professional Level

Related Videos

Auto Scaling Lifecycle Hooks

Deployment Concepts with Auto Scaling

Hands-On: Auto Scaling Lifecycle Hooks

Need Help?

Linux Academy Community

... and you can always send in a support ticket on our website to talk to an instructor!

Lab Connection Information

- Labs may take up to five minutes to build
- Access to an AWS Console is provided on the Live! Lab page, along with your login credentials
- Ensure you are using the N. Virginia region
- Labs will automatically end once the alloted amount of time finishes

Auto Scaling lifecycle hooks can be used to bootstrap instances on AWS. In this lab, we are configuring a simple HTML application within an Auto Scaling group using lifecycle hooks — these hooks send signals to endpoints and can be used to run scripts during the *Pending:Wait* state of an instance.

Create a Launch Configuration and Auto Scaling Group

Launch Configuration

Open the **EC2 Dashboard** and select **Auto Scaling Groups**. **Create Auto Scaling group**. First, **Create launch configuration**.

Select the *Amazon Linux AMI* and keep the *t2.micro* instance type. From here, we can give our launch configuration a **Name**. We called ours *devops-pro-configuration*. Set the **IAM role** to the role containing *auto-scaling* and open the **Advanced Details** section. This allows us to input Bash commands to run during the initialization period. Copy the following script into **User data**:

```
#!/bin/bash
yum update -y && \
yum install -y httpd && \
service httpd start && \
chkconfig httpd on && \
echo “<h1>Welcome to your application</h1>” > /var/www/html/index.html
&& \
chmod 644 /var/www/html/index.html && \
chown root:root /var/www/html/index.html && \
INSTANCE_ID=”`wget -q -O - http://instance-data/latest/meta-data/
instance-id`” && \
aws autoscaling complete-lifecycle-action --lifecycle-action-result
CONTINUE --instance-id $INSTANCE_ID --lifecycle-hook-name devops-pro-
hook --auto-scaling-group-name devops-pro --region us-east-1 || \
aws autoscaling complete-lifecycle-action --lifecycle-action-result
ABANDON --instance-id $INSTANCE_ID --lifecycle-hook-name devops-pro-hook
--auto-scaling-group-name devops-pro --region us-east-1
```

This script updates the packages, installs and starts Apache, sets an *index.html* file, changes permissions and owners on the file and then confirms that the initial script succeeds. If successful, the instance is then put into the *InService* state. Should the code fail, the process exits.

Set **IP Address Type** to *Assign a public IP address to every instance*. **Next: Add Storage**.

Do not change any settings until reaching the **Security Groups** page. Add *SSH* and *HTTP* access, setting the **Source** to *Anywhere* (0.0.0.0). **Review**. **Create Launch Configuration**.

Create a new key pair. We set our **Key pair name** to *devops-pro*. **Download** and **Create launch configuration**.

Auto Scaling Group

We now need to create the Auto Scaling group itself. Set the **Group name** to *devops-pro*, and set the **Group size** to *0* — we want to use lifecycle hooks to call for instance creation. Select the appropriate VPC and subnets. **Next: Configure scaling policies**. *Keep this group at its initial size*. **Next: Configure Notifications**. Progress forward with no changes to the default settings until the group creation is complete.

Create Lifecycle Hooks

Using the credentials provided on the Live! Lab page, SSH into the given server. Configure the CLI:

```
linuxacademy@ip-10-0-0-120:~$ aws configure
AWS Access Key ID [None]:
AWS Secret Access Key [None]:
Default region name [None]: us-east-1
Default output format [None]:
```

Leave all settings with the default options, except for the **Default region name** which needs to be set to *us-east-1*.

Add a lifecycle hook:

```
linuxacademy@ip-10-0-0-120:~$ aws autoscaling put-lifecycle-hook
--lifeycle-hook-name devops-pro-hook --auto-scaling-group-name devops-
pro --lifecycle-transition autoscaling:EC2_INSTANCE_LAUNCHING
```

Confirm its success:

```
linuxacademy@ip-10-0-0-120:~$ aws autoscaling describe-lifecycle-hooks
--auto-scaling-group-name devops-pro
{
    “LifecycleHooks”: [
        {
            “GlobalTimeout”: 172800,
            “HeartbeatTimeout”: 3600,
            “AutoScalingGroupName”: “devops-pro”,
            “LifecycleHookName”: “devops-pro-hook”,
            “DefaultResult”: “ABANDON”,
            “LifecycleTransition”: “autoscaling:EC2_INSTANCE_LAUNCHING”
        }
      ]
  }
```

This hook stops an instance in its pending state until it reaches its timeout period, or it receives a notification that it finished provisioning, as in the Bash script provided.

Launch an Instance

Earlier in the lab, we set the minimum and maximum instance number for the Auto Scaling group to zero. Return to the **EC2 Dashboard**, **Auto Scaling Groups**. **Edit** your group, and set the **Desired**, **Min** and **Max** values to *1*. **Save**.

Open the **Instances** tab and refresh until the instance launches. Refresh to monitor the **Lifecycle** status as it goes from *Pending* to *Pending:Wait* to *InService*, denoting the instance has launched and is healthy according to the Auto Scaling group.

Navigate to the **Instances** page and access the website through the provided public DNS. You should see the “Welcome to your application” message set up in the Bash script.

Additionally, you can SSH into the instance using the key pair downloaded earlier to further confirm the success of the Bash script.