AWS RoboMaker: Developer Guide
Copyright © 2020 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon’s trademarks and trade dress may not be used in connection with any product or service that is not Amazon’s, in any manner that is likely to cause confusion among customers, or in any manner that disparages or discredits Amazon. All other trademarks not owned by Amazon are the property of their respective owners, who may or may not be affiliated with, connected to, or sponsored by Amazon.
# Table of Contents

- **What Is AWS RoboMaker?** ................................................................. 1
- **Are You a First-time User of AWS RoboMaker?** .............................. 1
- **Supported Software and Versions** .................................................. 1
- **ROS 2 Dashing (Beta)** ....................................................................... 3
  - Known Differences in ROS2 .......................................................... 3
  - Cross-Compiling Applications with Colcon .................................. 3
  - Capturing log data with rosout ..................................................... 3
  - Using the ROS2 command line interface ...................................... 3
  - Using the AWS RoboMaker Simulation ROS Service ................. 4
- **Known Issues** .................................................................................... 4
- **How It Works** .................................................................................. 5
- **Robotics Development with AWS RoboMaker** ............................... 5
- **Create a ROS Development Environment** ..................................... 6
- **Create a Robot Application** ............................................................ 6
- **Develop Simulation and Testing Data** ............................................. 7
- **Fleet Management and Deployment** .............................................. 7
- **Getting Started** ............................................................................... 8
  - Important Licensing Information .................................................. 8
  - Step 1: Create an Account ............................................................... 9
    - Create an AWS Account ............................................................ 9
    - Create an IAM Administrator and Sign in ................................ 9
  - Step 2: Run the Hello World Sample Application ....................... 10
  - Step 3: Configure Environment and Build Applications ............. 10
    - Create a Development Environment ....................................... 11
    - Modify and Build Applications ................................................. 11
  - Step 4: Run Simulation ................................................................. 12
    - Bundle the Hello World Applications ..................................... 12
    - Restart the Hello World Simulation Application ..................... 12
    - Explore the Simulation ............................................................. 12
  - Step 5: Deploy Robot Application .................................................. 13
    - Prepare your TurtleBot3 Robot ................................................. 13
    - Bundle and Deploy Hello World Robot Application ................ 13
  - Step 6: Clean up ............................................................................. 15
  - Additional Sample Robots ............................................................. 15
    - Navigation and Person Recognition ....................................... 15
    - Voice Commands ..................................................................... 16
    - Robot Monitoring ..................................................................... 17
    - Object Following using Reinforcement Learning ...................... 18
    - Self-Driving using Reinforcement Learning .............................. 19
- **Sample Applications** ...................................................................... 21
  - Launching a Sample Application ............................................... 21
  - Configuring Permissions .................................................................. 22
    - Minimum permissions ......................................................... 22
    - Navigation .............................................................................. 23
    - Person detection .................................................................... 25
    - Voice commands ................................................................... 26
- **Application Versioning** ................................................................ 28
  - The $LATEST Version .................................................................. 28
  - Updating an Application Version ................................................. 28
  - Deleting an Application Version ................................................ 29
- **Developing Applications** ............................................................... 30
  - Creating a New Robotic Application ........................................... 30
    - Prerequisites ........................................................................... 31
  - Create the Robot Application Workspace .................................. 31
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create the Simulation Application Workspace</td>
<td>34</td>
</tr>
<tr>
<td>Build the Robot and Simulation Application Bundles</td>
<td>37</td>
</tr>
<tr>
<td>Building and Bundling Robotic Applications with Colcon</td>
<td>37</td>
</tr>
<tr>
<td>Installing Colcon</td>
<td>38</td>
</tr>
<tr>
<td>Using Colcon to Build and Bundle</td>
<td>38</td>
</tr>
<tr>
<td>Creating a Simulation Job</td>
<td>38</td>
</tr>
<tr>
<td>Prerequisites</td>
<td>38</td>
</tr>
<tr>
<td>Create Source and Output Amazon S3 Buckets</td>
<td>39</td>
</tr>
<tr>
<td>Create a Robot Application</td>
<td>39</td>
</tr>
<tr>
<td>Create a Simulation Application</td>
<td>39</td>
</tr>
<tr>
<td>Create a Simulation Job Role</td>
<td>40</td>
</tr>
<tr>
<td>Create a Simulation Job</td>
<td>41</td>
</tr>
<tr>
<td>AWS RoboMaker Cloud Extensions</td>
<td>42</td>
</tr>
<tr>
<td>Prerequisites</td>
<td>42</td>
</tr>
<tr>
<td>Installing AWS RoboMaker cloud extensions</td>
<td>43</td>
</tr>
<tr>
<td>Developing with AWS Cloud9</td>
<td>43</td>
</tr>
<tr>
<td>Creating a Development Environment</td>
<td>43</td>
</tr>
<tr>
<td>Working with the development environment</td>
<td>44</td>
</tr>
<tr>
<td>Deleting an Environment</td>
<td>51</td>
</tr>
<tr>
<td>Working with Robot Applications</td>
<td>52</td>
</tr>
<tr>
<td>Creating a Robot Application</td>
<td>52</td>
</tr>
<tr>
<td>Creating a Robot Application Version</td>
<td>53</td>
</tr>
<tr>
<td>Viewing a Robot Application</td>
<td>53</td>
</tr>
<tr>
<td>Updating a Robot Application</td>
<td>54</td>
</tr>
<tr>
<td>Deleting a Robot Application</td>
<td>54</td>
</tr>
<tr>
<td>Deleting a Robot Application Version</td>
<td>55</td>
</tr>
<tr>
<td>Working with Simulation Applications</td>
<td>56</td>
</tr>
<tr>
<td>Creating a Simulation Application</td>
<td>56</td>
</tr>
<tr>
<td>Creating a Simulation Application Version</td>
<td>57</td>
</tr>
<tr>
<td>Viewing a Simulation Application</td>
<td>57</td>
</tr>
<tr>
<td>Updating a Simulation Application</td>
<td>58</td>
</tr>
<tr>
<td>Deleting a Simulation Application</td>
<td>58</td>
</tr>
<tr>
<td>Deleting a Simulation Application Version</td>
<td>59</td>
</tr>
<tr>
<td>Running Simulation Jobs</td>
<td>60</td>
</tr>
<tr>
<td>VPC Settings</td>
<td>61</td>
</tr>
<tr>
<td>Configuring an AWS RoboMaker Simulation Job for Amazon VPC Access</td>
<td>61</td>
</tr>
<tr>
<td>Internet Access for Simulation Jobs</td>
<td>62</td>
</tr>
<tr>
<td>Connecting to a Simulation Job</td>
<td>62</td>
</tr>
<tr>
<td>Before You Enable Port Forwarding</td>
<td>62</td>
</tr>
<tr>
<td>Enabling Port Forwarding</td>
<td>63</td>
</tr>
<tr>
<td>Port Forwarding Examples</td>
<td>63</td>
</tr>
<tr>
<td>Accessing Simulation Job Data</td>
<td>65</td>
</tr>
<tr>
<td>Root Access and System Capabilities</td>
<td>66</td>
</tr>
<tr>
<td>Environment Variables Created by AWS RoboMaker</td>
<td>67</td>
</tr>
<tr>
<td>Managing Tags in a Simulation Job</td>
<td>67</td>
</tr>
<tr>
<td>Using tags using ROS command-line tools</td>
<td>67</td>
</tr>
<tr>
<td>Using tags from code in a simulation application</td>
<td>68</td>
</tr>
<tr>
<td>Using ROS Bags for Play Back</td>
<td>69</td>
</tr>
<tr>
<td>Example Launch File Configurations</td>
<td>69</td>
</tr>
<tr>
<td>Avoiding a Failed Simulation Job when Play Back Ends</td>
<td>72</td>
</tr>
<tr>
<td>Using Tags for Custom Status and Property Values</td>
<td>72</td>
</tr>
<tr>
<td>Canceling a Simulation Job Early</td>
<td>72</td>
</tr>
<tr>
<td>Managing Simulation Jobs</td>
<td>74</td>
</tr>
<tr>
<td>Creating a Simulation Job</td>
<td>74</td>
</tr>
<tr>
<td>Viewing a Simulation Job</td>
<td>78</td>
</tr>
<tr>
<td>Cancelling a Simulation Job</td>
<td>79</td>
</tr>
<tr>
<td>Cloning a Simulation Job</td>
<td>79</td>
</tr>
</tbody>
</table>
There are Missing Dependencies ....................................................................................... 135
Colcon Bundle Failed ........................................................................................................ 134
Colcon Build Failed ......................................................................................................... 134
API Reference ................................................................................................................................. 136
Actions .................................................................................................................................. 136
BatchDescribeSimulationJob ........................................................................................................ 138
CancelDeploymentJob ............................................................................................................... 142
CancelSimulationJob ................................................................................................................... 144
CancelSimulationJobBatch ........................................................................................................ 146
CreateDeploymentJob ............................................................................................................... 148
CreateFleet ............................................................................................................................. 154
CreateRobot ............................................................................................................................ 157
CreateRobotApplication .......................................................................................................... 161
CreateRobotApplicationVersion .............................................................................................. 165
CreateSimulationApplication .................................................................................................... 169
CreateSimulationApplicationVersion ...................................................................................... 174
CreateSimulationJob ................................................................................................................ 178
DeleteFleet .............................................................................................................................. 187
DeleteRobot ............................................................................................................................ 189
DeleteRobotApplication .......................................................................................................... 191
DeleteSimulationApplication .................................................................................................... 193
DeregisterRobot ....................................................................................................................... 195
DescribeDeploymentJob .......................................................................................................... 198
DescribeFleet ........................................................................................................................... 202
DescribeRobot ......................................................................................................................... 206
DescribeRobotApplication ....................................................................................................... 210
DescribeSimulationApplication ............................................................................................... 214
DescribeSimulationJob ............................................................................................................. 218
DescribeSimulationJobBatch .................................................................................................... 225
ListDeploymentJobs .................................................................................................................. 232
ListFleets ................................................................................................................................. 236
ListRobotApplications ............................................................................................................. 239
ListRobots ............................................................................................................................... 242
ListSimulationApplications ..................................................................................................... 245
ListSimulationJobBatches ......................................................................................................... 248
ListSimulationJobs .................................................................................................................. 251
ListTagsForResource ............................................................................................................... 254
RegisterRobot ......................................................................................................................... 256
RestartSimulationJob ............................................................................................................... 259
StartSimulationJobBatch ......................................................................................................... 261
SyncDeploymentJob ............................................................................................................... 270
TagResource ............................................................................................................................ 275
UntagResource ......................................................................................................................... 277
UpdateRobotApplication ......................................................................................................... 279
UpdateSimulationApplication ................................................................................................. 283
Data Types ............................................................................................................................ 287
BatchPolicy ............................................................................................................................ 289
DataSource ............................................................................................................................. 290
DataSourceConfig ................................................................................................................... 291
DeploymentApplicationConfig ............................................................................................... 292
DeploymentConfig .................................................................................................................. 293
DeploymentJob ......................................................................................................................... 294
DeploymentLaunchConfig ....................................................................................................... 296
FailedCreateSimulationJobRequest ....................................................................................... 298
Filter ....................................................................................................................................... 300
Fleet ....................................................................................................................................... 301
LaunchConfig .......................................................................................................................... 303
What Is AWS RoboMaker?

AWS RoboMaker is a service that makes it easy to create robotics applications at scale. AWS RoboMaker extends the Robot Operating System (ROS) framework with cloud services. This includes AWS machine learning services. It includes monitoring services. It even includes analytics services. These combine to enable a robot to do several things on its own. Stream data, navigate, communicate, comprehend, and learn. AWS RoboMaker provides a robotics application development environment. It provides a robotics simulation service, which speeds application testing. It provides a fleet management service so you can deploy and manage applications remotely.

Are You a First-time User of AWS RoboMaker?

If you are a first-time user of AWS RoboMaker, do the following.

1. **Read How It Works (p. 5)** – An overview of AWS RoboMaker. Learn the key concepts and components involved in building robot applications and simulations. Read this topic in the order presented.
2. **Read Getting Started with AWS RoboMaker (p. 8)** – A tutorial to help you build your first robot application and simulation. Explains how to run a simulation job. Use the sample code provided by AWS RoboMaker.
3. Depending on your needs, explore the following topics.
   - Learn about Gazebo, a simulator to test robots. Gazebo helps you test algorithms. It helps you design robots and train artificial intelligence. For more information, see [http://gazebosim.org/tutorials](http://gazebosim.org/tutorials). Select version 9.

Supported Software and Versions

AWS RoboMaker supports the following programs, tools, and libraries.

<table>
<thead>
<tr>
<th>Name</th>
<th>Versions Supported</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Robot Operating System (ROS)</td>
<td><strong>Kinetic</strong> (end of life April 2021)</td>
<td>Libraries and tools to help you create robot applications.</td>
</tr>
<tr>
<td></td>
<td><strong>Melodic</strong> (end of life May 2023)</td>
<td></td>
</tr>
<tr>
<td>Robot Operating System 2 (ROS 2)</td>
<td><strong>Dashing</strong> (in AWS RoboMaker beta)</td>
<td>A newer version of ROS still under development.</td>
</tr>
<tr>
<td>Colcon</td>
<td><strong>Latest</strong></td>
<td>Command line tool to bundle ROS robot and simulation applications.</td>
</tr>
<tr>
<td>Gazebo</td>
<td><strong>7, 9</strong></td>
<td>Tool to simulate robots in an environment.</td>
</tr>
<tr>
<td>rviz</td>
<td><strong>ROS rviz</strong></td>
<td>Tool to visualize sensor data and state information from ROS in 3D.</td>
</tr>
<tr>
<td></td>
<td><strong>ROS2 rviz2</strong></td>
<td></td>
</tr>
</tbody>
</table>
### Supported Software and Versions

<table>
<thead>
<tr>
<th>Name</th>
<th>Versions Supported</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rqt</td>
<td>latest</td>
<td>Framework and plugins based on Qt for ROS GUI development.</td>
</tr>
</tbody>
</table>

ROS supports the Python versions listed here.

<table>
<thead>
<tr>
<th>Name</th>
<th>Version(s) Supported</th>
<th>Ubuntu Version(s) Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROS Kinetic</td>
<td>Python 2</td>
<td>Ubuntu Xenial</td>
</tr>
<tr>
<td>ROS Melodic</td>
<td>Python 2</td>
<td>Ubuntu Bionic</td>
</tr>
<tr>
<td>ROS2 Dashing</td>
<td>Python 3</td>
<td>Ubuntu Bionic</td>
</tr>
</tbody>
</table>

The following table shows which ROS versions are supported for different AWS RoboMaker features.

<table>
<thead>
<tr>
<th>AWS RoboMaker Feature</th>
<th>ROS version(s) Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cloud Extensions</td>
<td>Melodic, Kinetic, Dashing</td>
</tr>
<tr>
<td>Development Environment (Create)</td>
<td>Melodic, Kinetic, Dashing</td>
</tr>
<tr>
<td>Simulation Job</td>
<td>Melodic, Kinetic, Dashing</td>
</tr>
<tr>
<td>Fleet Management</td>
<td>Melodic, Kinetic, Dashing</td>
</tr>
</tbody>
</table>

AWS RoboMaker simulation jobs support the combinations of ROS distribution and Gazebo listed here.

- Kinetic and Gazebo 7
- Kinetic and Gazebo 9
- Melodic and Gazebo 9
- Dashing and Gazebo 9

Gazebo supports the components listed here.

<table>
<thead>
<tr>
<th>Component</th>
<th>Version(s) Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>Physics engine</td>
<td>ODE</td>
</tr>
<tr>
<td>Rendering engine</td>
<td>OGRE</td>
</tr>
</tbody>
</table>
ROS 2 Dashing (Beta)

AWS RoboMaker now supports Robot Operating System 2 (ROS2) in beta release. ROS2 is built on top of Data Distribution Standard (DDS), an industry data connectivity standard that provides discovery, serialization, and transportation. ROS2 expands ROS's use cases with increased security, quality of service, support for embedded systems and real-time scenarios. Customers can develop and test their robot applications in AWS RoboMaker with ROS2-Dashing, through the AWS Management console, AWS SDK, AWS CLI, and the AWS API. AWS RoboMaker recommends that customers do not use this beta release of the ROS2-Dashing feature for their production workloads.

We are releasing the AWS RoboMaker ROS2 beta feature to give customers a preview of how they can build their robot applications using ROS2, specifically ROS2 Dashing, the first Long Term Support release of ROS2. There are certain known issues in ROS-2 Dashing, which the community is working to address so we do not recommend you build production applications using ROS2 Dashing. This is, however, an opportunity to develop, test and evaluate your robot applications with the AWS RoboMaker ROS2 feature and provide feedback before we announce general availability of ROS2 in AWS RoboMaker once these issues have been fully addressed. At the end of this beta program, we will deprecate support for ROS2 Dashing and will not be able to start new simulation jobs; existing simulation jobs will be allowed to run to completion. Customers will also be asked to migrate to the new release of the AWS RoboMaker ROS2 feature for managing their robot fleets.

For more information about beta terms and conditions, see AWS Service Terms.

Topics

- Known Differences in ROS2 (p. 3)
- Known Issues (p. 4)

Known Differences in ROS2

This section describes the known differences between ROS and ROS2 when using AWS RoboMaker.

Cross-Compiling Applications with Colcon

You can now cross-compile your robot applications using the cc-build plug-in for colcon. For more information about installing it in your development environment, see ros-tooling/cross-compile.

Capturing log data with rosout

Log output sent to the rosout topic is saved to a log file. The log file name format is rosout.log and will be copied to the Amazon S3 bucket specified when the simulation job was created. Specifically, it will be copied to bucket_name/simulation_id/run_id/rologs/rosout.

When a log file needs to be rotated because of size, the file is given a numerical suffix (for example, rosout.log.1) and older log files are incremented.

Using the ROS2 command line interface

To inspect ROS2 topics, messages, nodes, services and other information, use the ros2 command. For more information about how to use ros2, see Introspection with Command Line Tools.
Using the AWS RoboMaker Simulation ROS Service

For more information about using tags and cancelling a simulation with ROS2 Dashing, see github.com/aws-robotics/aws-robomaker-simulation-ros-pkgs/tree/dashing.

Known Issues

This section describes the known issues for using ROS2 with AWS RoboMaker.

- ROS bag play back is not supported. For more information about ROS bag play back, see Using ROS Bags for Play Back (p. 69). We do not automatically record rosbags for ROS2 applications when you provide an Amazon S3 output location. To learn more about recording rosbags from your ROS2 application, see the rosbag2 repository on GitHub.
- The Hello world and Robot monitoring sample applications have been updated to work with ROS2. For a description of the sample applications, see Sample Applications (p. 21).
How It Works

AWS RoboMaker is a service that allows you to quickly develop, test, and deploy robot applications. This section provides an overview of robot development. It explains how AWS RoboMaker works. If you are a first-time user of AWS RoboMaker, read the following sections in order.

Topics

• Robotics Development with AWS RoboMaker (p. 5)
• Create a ROS Development Environment (p. 6)
• Create a Robot Application (p. 6)
• Develop Simulation and Testing Data (p. 7)
• Fleet Management and Deployment (p. 7)

Robotics Development with AWS RoboMaker

This section shows a typical robot development workflow. It tells how you accomplish the tasks with AWS RoboMaker.

Robot application development usually starts after you choose your physical robot hardware. First, you create a development environment and load the tools to build an application. Next, create the robot application. Write custom logic that responds to environmental data. Next, build simulations, or models of the world that your robot will inhabit. Collect data about how your robot performs in simulation jobs. When your tests are complete, deploy your application to physical robots. Monitor them and update the software when needed.

As a robot developer, you typically perform the following activities.

1. Create a ROS development environment. To create a robot application, you need an environment configured for ROS development along with tools like Colcon to build and bundle the application. You'll also need tools to help you cross-compile the application for your physical robot. Using an integrated development environment makes it easier.

   In AWS RoboMaker, you can create an AWS Cloud9 development environment that is already configured with the tools to develop robot applications. You can also use your existing environment.

2. Create the robot application. This is where you get to write code. Build on the foundation provided by ROS and integrate functions you find elsewhere. The application you create works with your robot hardware, provides intelligence, and works with the cloud.

3. Develop simulation and testing data. In this stage, run your robot application in simulated environments. Collect sensor data and other performance data from the simulation. It can take many simulation tests to complete the robot programming.

4. Deploy an application to robot fleets. When your application performs as expected, you are ready to deploy it to a robot. In AWS RoboMaker, a robot must belong to a group of robots (a fleet) in order to receive deployed software. Each virtual robot in AWS RoboMaker represents a physical robot.

5. Monitor and update robots. Your robots are interacting in the world! Refine them by using data you collect with AWS RoboMaker cloud extensions.

The following sections explore the details of each step.
Create a ROS Development Environment

Before you create a robot application, you need a properly configured development environment. Robot development with AWS RoboMaker depends on a number of open source packages.

<table>
<thead>
<tr>
<th>Package</th>
<th>Version</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Robot Operating System (ROS)</td>
<td>Kinetic</td>
<td>Libraries and tools to help developers create robot applications.</td>
</tr>
<tr>
<td></td>
<td>Melodic</td>
<td></td>
</tr>
<tr>
<td>Robot Operating System 2 (ROS2)</td>
<td>Dashing</td>
<td>A newer version of ROS still under development. For more information about using ROS 2 with AWS RoboMaker, see ROS 2 Dashing (Beta) (p. 3).</td>
</tr>
<tr>
<td>Colcon</td>
<td>Latest</td>
<td>A command line tool for bundling ROS robot and simulation applications.</td>
</tr>
<tr>
<td>Gazebo</td>
<td>7</td>
<td>Tool for simulating robots in complex environments.</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>rviz</td>
<td>Melodic</td>
<td>Tool for visualizing sensor data and state information from ROS in 3D.</td>
</tr>
<tr>
<td></td>
<td>Dashing</td>
<td></td>
</tr>
<tr>
<td>rqt</td>
<td>Melodic</td>
<td>Framework and plugins based on Qt for ROS GUI development.</td>
</tr>
<tr>
<td></td>
<td>Dashing</td>
<td></td>
</tr>
</tbody>
</table>

You can create your own development environment or update an existing development environment to support AWS RoboMaker. Most developers use Ubuntu or other supported Linux variants. Other operating systems might be compatible.

AWS RoboMaker provides a quick and easy way to create a development environment that is already configured for robot development.

Create a Robot Application

After your development environment is configured, create a robot application. Build on the foundation ROS provides. ROS relies on a Computation Graph. It’s a collection of concurrent processes (or nodes) that perform a task like controlling wheel motors or passing messages.

You do not have to create nodes for common robotic hardware and algorithms. There are packages, which are nodes and dependent message definitions, to work with motors, lasers, actuators, lidar, and sensors of all kinds. There are also packages that consume data from other packages to create maps, find paths, and more. For a more comprehensive list, see ros.org.
Using the AWS RoboMaker cloud extensions, capture operational data to aid in troubleshooting and enhance your robot with intelligent functions. For more information, see the section called “AWS RoboMaker Cloud Extensions” (p. 42).

Develop Simulation and Testing Data

Robot application developers use simulations to help refine behavior. They use simulations to test robotics algorithms and perform regression tests. Simulations use realistic scenarios and detailed virtual environments to model the world and mimic robot behavior.

In AWS RoboMaker, a simulation application contains models for the robot, terrain, and assets organized in a scene. The simulation application is responsible for simulating the physical aspects of a robot such as its sensors, kinematics, and dynamics. Sensors may include cameras, lidars, and even GPS devices. Kinematics and dynamics are required to allow the robot to move its joints or wheels and interact physically, such as colliding, with objects in a simulated environment.

To run a simulation, pair a robot application with a simulation application in a simulation job. The simulation job can run up to 14 days. You can restart it with a new application while it is running. You can interact with a running simulation by using Gazebo, rviz, rqt, and a terminal to interact at the command line. For example, use Gazebo to see a rendered model of the robot in the environment and use the terminal to listen or send ROS messages to your robot.

The robot is unaware that it is inside a simulated environment. The simulator uses the same interfaces and data types as the robot's physical devices. This makes it possible to test the same robot software in a simulation and then deploy it to your robots.

Fleet Management and Deployment

After testing is complete, you can deploy the robot application to your robots by using an AWS IoT Greengrass over-the-air update. Before you deploy your application, set up each robot to accept updates from AWS RoboMaker and communicate its status. Next, register your robots into a fleet. A fleet is a logical group of robots. When your fleet is set up, deploy your robot application. You can control the pace of deployment. You can also control what happens before and after your application launches on the robot.

Information about the deployment is provided by AWS RoboMaker. Additional information specific to your robot and scenarios can be captured using AWS RoboMaker cloud extensions and custom code.
Getting Started with AWS RoboMaker

In this section, you learn the basics of how to structure your robot applications and simulation applications. You also learn how to edit code, build, launch new simulations, and deploy applications to robots. Hello world is an example robot application and simulation application you can use. Hello world is built for TurtleBot3. The robot rotates in place. The simulation is an empty world.

Note
This tutorial works with ROS Kinetic and ROS Melodic. For more information about ROS 2 Dashing, see ROS 2 Dashing (Beta) (p. 3).

- **Build a robot application.** A robot application is a robot operating system (ROS)-based application that runs on a physical robot. To run your application in an AWS RoboMaker simulation, you build an X86_64 architecture version of the robot application.

- **Build a simulation application.** A simulation application includes a 3D artificial world and Gazebo plugins that control the movement of a robot within that world. An X86_64 architecture version of the simulation application is required.

- **Launch an AWS RoboMaker simulation.** Simulations help you develop and test algorithms, train machine learning models, and regression-test robot applications.

- **Deploy a robot application.** You can build and deploy a robot application to a TurtleBot3 robot. Deployment is done over-the-air using AWS IoT Greengrass. For more information about TurtleBot3, see TurtleBot3.

- **Access logging and monitoring features.** You can access service metrics, data written to stdout, ROS bags, and Gazebo log files. Access can be limited using IAM.

**Topics**

- Important Licensing Information (p. 8)
- Step 1: Create an AWS Account and an Administrator (p. 9)
- Step 2: Run the Hello World Sample Application (p. 10)
- Step 3: Configure Environment and Build Applications (p. 10)
- Step 4: Run Simulation (p. 12)
- Step 5: Deploy Robot Application (p. 13)
- Step 6: Clean up (p. 15)
- Additional Sample Robots (p. 15)

**Important Licensing Information**

AWS RoboMaker sample applications include third-party software licensed under open-source licenses. The samples are provided for demonstration purposes only. Incorporation or use of AWS RoboMaker sample applications in connection with your production workloads or commercial products or devices may affect your legal rights or obligations under the applicable open-source license. Source file information is included in the *readme* file of each sample application.

- Hello World
- Navigation and Person Recognition
- Voice Commands
- Robot Monitoring
AWS RoboMaker Developer Guide
Step 1: Create an Account

• Self-Driving using Reinforcement Learning
• Object Following using Reinforcement Learning

AWS RoboMaker development environment and simulation include third-party software licensed under open-source licenses. View the source file and licensing information here:

• Simulation Environment
• Development Environment

Step 1: Create an AWS Account and an Administrator

Before you use AWS RoboMaker for the first time, complete the following tasks:

Topics
• Create an AWS Account (p. 9)
• Create an IAM Administrator and Sign in (p. 9)

Create an AWS Account

If you already have an AWS account, skip this step.

When you sign up for Amazon Web Services (AWS), your AWS account is automatically signed up for all AWS services, including AWS RoboMaker. You are charged only for the services that you use.

To create an AWS account

2. Follow the online instructions.

   Part of the sign-up procedure involves receiving a phone call and entering a verification code on the phone keypad.

Write down your AWS account ID because you’ll need it for the next task.

Create an IAM Administrator and Sign in

When you create an AWS account, you get a single sign-in identity. This allows access to all of the AWS services and resources in the account. This identity is called the AWS account root user. When you sign in to the AWS Management Console with the credentials that you used to create the account, you have access to all of the AWS resources in your account.

We strongly recommend that you not use the root user for everyday tasks, even the administrative ones. Instead, adhere to the Create Individual IAM Users. Create an AWS Identity and Access Management (IAM) user with administrator permissions. Then, securely store your root user credentials and use them to perform only a few account and service management tasks.

To create an IAM user with administrator permissions, and sign in to the console

1. Create an account with administrator permissions in your AWS account. For instructions, see Creating Your First IAM User and Administrators Group in the IAM User Guide.
Step 2: Run the Hello World Sample Application

Before you begin working with a robot application and simulation application code, run the Hello World demo application in the AWS RoboMaker console. This sets up the AWS resources the application needs, including the appropriate IAM roles and Amazon S3 bucket for loading applications and writing simulation output.

To run the Hello World demo application

2. In the AWS RoboMaker console, expand Resources on the left and then select Sample applications.
3. In the Try AWS RoboMaker sample applications page, select Hello World! and then select Launch. This opens the simulation job detail page while AWS RoboMaker launches the sample simulation.

   The sample application will use ROS Melodic and Gazebo 9 by default.

4. On the simulation job detail page, when status becomes running, select Gazebo.
5. In the AWS RoboMaker gzclient window, use the mouse or keyboard to zoom in on the TurtleBot. For more information, see Gazebo Keyboard Shortcuts.

   It is spinning clockwise. Gazebo is fully functional, so you can try out other features. For example, if you want more light on the robot, choose the sun (point light) icon. Then move the pointer around the robot to illuminate it.

6. When you are done, close Gazebo by closing the browser window.

   The Hello World simulation runs for 1 hour. In later steps, you have an opportunity to restart the simulation. If you do, the simulation job timer is reset to zero and the simulation job will run another 1 hour.

Step 3: Configure Environment and Build Applications

In this section, you create an AWS Cloud9 environment with AWS RoboMaker. You then install sample code. In the development environment, you modify the robot application, and then build the robot and simulation application.
Create a Development Environment

The AWS Cloud9 development environment provides the tools to develop robot applications and simulation applications with ROS and AWS RoboMaker.

To create a development environment:
2. On the left, expand Development, choose Development environments, and then choose Create environment.
3. In the Create AWS RoboMaker development environment page, enter HelloWorld as the environment name.
4. For ROS Distribution, select ROS Melodic.
5. Accept the default Instance type (m4.large). You can select different instances type to improve bundling performance.
6. Select a VPC. Use the default VPC.
7. Select a Subnet. Use a public Subnet.
8. Choose Create to create the AWS Cloud9 development environment.

Modify and Build Applications

In this section, you use the AWS Cloud9 development environment to modify the robot application. The steps show how to rotate the robot counter-clockwise and then build the robot and the simulation application.

To build the robot and simulation applications
2. On the left, expand Development, choose Development environments, select HelloWorld, and then choose Open environment. It might take a few minutes to prepare the development environment.
3. In the HelloWorld AWS Cloud9 development environment, choose Resources, then choose Download samples, and then select 1. Hello World.
4. On the left, in the Environment tab, expand HelloWorld, HelloWorld, robot_ws. src, hello_world_robot, and then nodes. Select the file rotate to load into the editor.
5. In the rotate tab, modify the code to make the robot turn clockwise by making the rate negative: self.twist.angular.z = -0.1. Save the file by selecting File and then Save.
6. Build the robot application. On the menu, choose Run, then choose Build, and then select HelloWorld Robot.
7. Build the simulation application. On the menu, choose Run, then choose Build, and then select HelloWorld Simulation.

Next Step

Step 4: Run Simulation (p. 12)
Step 4: Run Simulation

In this section, you bundle the robot application and simulation application into source files. A source file includes all of the dependencies you need to run the application. You use a robot application source file and a simulation application source file to create a simulation job. You use a robot application source file to create a deployment.

Topics
- Bundle the Hello World Applications (p. 12)
- Restart the Hello World Simulation Application (p. 12)
- Explore the Simulation (p. 12)

Bundle the Hello World Applications

In the previous step, you built the robot and simulation applications. In this step, you bundle each application with the resources and dependencies you need to run. For example, the simulation application needs a model and physical parameters of the TurtleBot3.

Warning
Bundling the robot application and simulation application might each take 20 minutes or more depending on your instance type.

To bundle the Hello World applications

1. In the HelloWorld AWS Cloud9 development environment, choose Run on the menu, then choose Bundle, and then select HelloWorld Robot.
2. On the menu, choose Run, then choose Bundle, and then select HelloWorld Simulation.

Restart the Hello World Simulation Application

A running simulation can be restarted with updated robot application and simulation application source. This is useful when you are debugging code and making frequent changes.

When you restart a simulation, its timer is reset to zero. It will run for the original duration until reset or canceled. For Hello World, the duration is 1 hour.

1. On the menu, choose Simulation, then select Connect. Select the simulation job from the list and then select Okay. Restart this running Hello World simulation with your new robot application.
2. Choose AWS RoboMaker Simulation, and select Restart with new bundle(s). In the Restart simulation with selected bundles dialog, type in HelloWorld/robot_ws/bundle/output.tar for Robot app bundle path and HelloWorld/simulation_ws/bundle/output.tar for Simulation app bundle path, then select OK.

AWS RoboMaker restarts the simulation with the new applications. This extends the lifetime of the simulation for another 1 hour.

Explore the Simulation

When the simulation is running, you can launch Gazebo and verify that the robot is rotating counterclockwise. You can also view the logs in the AWS Cloud9 development environment terminal.

1. The status of the simulation job is shown next to the Simulation menu entry. When it transitions to Running, you can launch applications and explore the simulation.
2. In the AWS RoboMaker console, choose Simulation jobs on the left and then select the Hello World simulation job.
3. In the Simulation details screen, in the Simulation tools section, select Gazebo.
4. In Gazebo, zoom in on the robot. It is rotating in place.
5. Log data is streamed to CloudWatch Logs and to tabs on the AWS Cloud9 development environment.

To view CloudWatch Logs, select Logs in the Simulation details screen.
6. Output files generated by the simulation job are available in the simulation job output destination. Select the link to access the Amazon S3 bucket.

Next Step

Step 5: Deploy Robot Application (p. 13)

Step 5: Deploy Robot Application

In this section, you compile the Hello World robot application for the ARMHF architecture, which is used by TurtleBot3. You also create a robot in AWS RoboMaker and then configure your TurtleBot3 robot with AWS IoT Greengrass software. Then you register the robot into a new fleet and deploy the application.

Topics
- Prepare your TurtleBot3 Robot (p. 13)
- Bundle and Deploy Hello World Robot Application (p. 13)

Prepare your TurtleBot3 Robot

Before you can deploy a robot application to a physical robot, create the robot in AWS RoboMaker. Next, configure your physical robot with AWS IoT Greengrass. After that, create a fleet and register your robot into the fleet.

To prepare your TurtleBot3 robot

1. First, create a robot in AWS RoboMaker and configure your TurtleBot3 robot. Follow the steps in Creating a Robot (p. 90).
2. Next, create a fleet. Follow the steps in Creating a Fleet (p. 93).
3. Register your robot as part of the new fleet. Follow the steps to Register your robot to your fleet in Registering and Deregistering Robots (p. 93).

Bundle and Deploy Hello World Robot Application

In this section, bundle the Hello World application for the ARMHF architecture using a Docker image. Then you copy the robot application bundle to an Amazon S3 bucket and deploy.

Bundling the robot application and simulation application might each take 20 minutes or more depending on your instance type.

Note
The AWS RoboMaker cross-compilation container uses colcon bundle to package the compiled sources. This resolves the Debian packages for Ubuntu (Bionic and Xenial). The
packages are provided in the ROS distributions. Packages that are not available in the ROS distributions, such as Debian packages for Raspbian, are not supported by the cross-compilation container.

**To bundle and deploy the Hello World robot application**

2. On the left, expand Development, choose Development environments, select HelloWorld, and then select Open IDE. It might take a few minutes to prepare the development environment.
3. In the HelloWorld AWS Cloud9 development environment, choose the bash tab at the bottom of the page. Next, run the following commands to bundle the HelloWorld robot application for ARMHF:

   **Note**
   The following commands are for ROS Kinetic and ROS Melodic applications.

   ```
   # cd /opt/robomaker/cross-compilation-dockerfile/
   # sudo bin/build_image.bash
   # cd ~/environment/HelloWorld/robot_ws
   # sudo docker run -v $(pwd):/ws -it ros-cross-compile:armhf
   # cd ws
   # apt update
   # rosdep install --from-paths src --ignore-src -r -y
   # colcon build --build-base armhf_build --install-base armhf_install
   # colcon bundle --build-base armhf_build --install-base armhf_install --bundle-base armhf_bundle --apt-sources-list /opt/cross/apt-sources.yaml
   # exit
   ```

4. Before you can deploy the robot application, copy it to an Amazon S3 bucket that AWS RoboMaker can access. A bucket was created on your behalf for the HelloWorld robot application. Find the name of the bucket using the following command:

   ```
   $ aws s3 ls | grep "robomakerhelloworld"
   ```

   Now, copy the robot application to the bucket you found above.

   ```
   $ aws s3 cp armhf_bundle/output.tar s3://robomakersampleapplicationhelloworld-bundlesbucket-##########/hello-world-robot.armhf.tar
   ```

   The ARMHF version of the robot application is hello-world-robot.armhf.tar.

5. Update the Hello World robot application created on your behalf with the ARMHF version of the robot application. In the AWS RoboMaker console, select Development, then select Robot applications, and then choose the robot application with a name beginning with roboMakerSampleApplicationHelloWorld_.

6. In the Robot application details page, in Versions, select $LATEST, and then choose Update.
7. In the Update robot application page, in Sources, specify the location of the ARMHF bundle from step 4 in the ARMHF source file field, then select Create.
8. In the Robot application details page, in Versions, select Create new version, and then choose Create.
9. You are now ready to deploy the robot application to your TurtleBot3 robot. In the AWS RoboMaker console, select Fleet management, then select Deployments, and then select Create deployment.
10. In the Create deployment page, select the Fleet you created with your robot registered, then select the Robot application and Robot application version you created.
11. In Deployment Launch config, type in hello_world_robot for Package name. Then type in deploy_rotate.launch for Launch file.
12. In **Deployment config**, accept the default values for **Concurrent deployment percentages** and **Failure threshold percentage**.
13. Select **Create**. You can track the deployment status and deployment progress detail on the **deployment details** page.

**Next Step**

Step 6: Clean up (p. 15)

**Step 6: Clean up**

To avoid extra charges, use the AWS Management Console to delete items that you created for this exercise.

1. Open the AWS RoboMaker console at https://console.aws.amazon.com/robomaker/. Choose **Simulation jobs** and then choose the **Hello World** simulation job. In the **Simulation job details** page, choose **Actions**, choose **Cancel**, and then choose **Yes, cancel**.

   **Note**
   Simulation jobs are automatically deleted after 90 days.

2. Open the AWS CloudFormation console at https://console.aws.amazon.com/cloudformation/ and delete the stack with **HelloWorld**. If there are multiple entries, choose by date and time.

3. Open the IAM console at https://console.aws.amazon.com/iam/ and delete the IAM role. If you created permissions policies, delete them.

4. Open the AWS Cloud9 console at https://console.aws.amazon.com/cloud9/. Choose the **HelloWorld** environment and then choose **Delete**. Confirm by typing **Delete** and then selecting **Delete**.

**Additional Sample Robots**

This section contains more advanced robots and simulations.

**Topics**

- Navigation and Person Recognition (p. 15)
- Voice Commands (p. 16)
- Robot Monitoring (p. 17)
- Object Following using Reinforcement Learning (p. 18)
- Self-Driving using Reinforcement Learning (p. 19)

**Navigation and Person Recognition**

In this section, explore a robot that moves between places and recognizes faces in photos.

Before you use AWS RoboMaker for the first time, complete the tasks in Create an Account (p. 9). Then, in the AWS RoboMaker console, launch the Navigation and Person Recognition sample application.

**Topics**

- View Simulated Camera Images (p. 16)
- View Recognized People (p. 16)
View Simulated Camera Images

Use rqt to view images from the robot's camera.

To view simulated camera images

2. In the AWS RoboMaker console, choose Simulation jobs on the left and then select the Navigation and Person Recognition simulation job.
3. In the Simulation details page, in the Simulation tools section, select rqt.
4. In rqt, choose Plugins, Visualization, Image View.
5. Select /camera/rgb/image_raw.

View Recognized People

Use the terminal and rostopic to view people as the robot explores the virtual space.

To view the logs

2. In the AWS RoboMaker console, choose Simulation jobs on the left and then select the Navigation and Person Recognition simulation job.
3. In the Simulation details page, in the Simulation tools section, select terminal.
4. In terminal, type in the following to set up the ROS environment and run the text command tool.

   ```
   eval $AWS_ROBOMAKER_ROBOT_APPLICATION_SETUP
   ```

5. Type in the following command.

   ```
   rostopic echo /rekognized_people
   ```

   When a person is recognized, you see output similar to this:

   ```
   Data: "I see brandon"
   ```

Voice Commands

In this section, explore a robot that accepts commands through natural language text and voice in a simulated bookstore using Amazon Lex. The robot supports the commands move [direction] [rate] or turn [direction] [rate], and stop. Each command is acknowledged and then executed.

Before you use AWS RoboMaker for the first time, complete the tasks in Create an Account (p. 9). Then, in the AWS RoboMaker console, launch the Voice Commands sample application.

Topics

- Use Natural Language Text to Move the Robot (p. 16)

Use Natural Language Text to Move the Robot

Use rqt to view images from the robot's camera.
To view simulated camera images

2. In the AWS RoboMaker console, choose Simulation jobs on the left and then select the Voice Commands simulation job.
3. In the Simulation details screen, in the Simulation tools section, select Gazebo. Zoom in on the robot. There are a series of obstacles positioned in front of it.
4. In the Simulation details screen, in the Simulation tools section, select terminal.
5. In terminal, type in the following commands to set up the robot operating system (ROS) environment and run the text command tool.

   ```
   eval $AWS_ROBOMAKER_ROBOT_APPLICATION_SETUP
   rosrun voice_interaction_robot text_input.py
   ```

6. Type in the following command to see the robot hit obstacles. The robot is moving forward at 0.4 meters per second.

   ```
   move forward 0.4
   ```

   The robot tries to move forward until it is told to stop. The faster the robot moves, the more the obstacles are displaced. Reasonable speeds are 0.3–0.5 meters per second.

   You can also move the robot backward.

   ```
   move backward 0.4
   ```

7. The robot can be commanded to turn clockwise or counterclockwise. For example:

   ```
   turn clockwise .5
   turn counterclockwise .65
   ```

   The turn rate is radians per second. Values between 0.4 and 0.78 are reasonable for the robot.

8. Type stop to stop the robot.

Robot Monitoring

This section shows how to monitor health and operational metrics for a robot in a simulated bookstore using Amazon CloudWatch Metrics and Amazon CloudWatch Logs. The streams metrics including speed, distance to nearest obstacle, distance to current goal, robot CPU utilization, and RAM usage.

Before you use AWS RoboMaker for the first time, complete the tasks in Create an Account (p. 9). Then, in the AWS RoboMaker console, launch the Robot Monitoring sample application.

Topics

- View Robot Health and Performance Metrics (p. 17)
- View CloudWatch Logs (p. 18)

View Robot Health and Performance Metrics

The Robot Monitoring sample application uses AWS RoboMaker cloud extensions to write custom health and performance metrics to Amazon CloudWatch.
To view robot health and performance metrics

2. In the CloudWatch console, select Metrics.
3. On the Metrics page, in the All metrics tab, select the Robot Monitoring example.
4. On the Metrics page, in the All metrics tab, select Robot. These are the operational metrics for the robot.
5. Select all of the metrics. Each metric will appear on the graph in a different color.
6. Hover over the graph to see values for that moment. You can also select a region on the graph to zoom in or select custom at the top of the page to select a custom time period.

View CloudWatch Logs

When a simulation job runs, logs are generated by simulation tools and the applications in the simulation job.

To view the logs

2. In the CloudWatch console, select Logs.
3. On the Log groups page, select the Robot Monitoring Log Group, and then select Turtlebot3.
4. Select an event to see details. For example, if you filter for monitor_obstacle_distance events, you can see the distance to the nearest obstacle at that moment.

Object Following using Reinforcement Learning

In this section, you teach a robot to track and follow an object through reinforcement learning in simulation using the Coach Reinforcement Learning Library. View the reward metrics in Amazon CloudWatch Metrics to explore how the machine learning model improves over time. Customize your reward function to improve the machine learning algorithm used for training.

Before you use AWS RoboMaker for the first time, complete the tasks in Create an Account (p. 9). Then, in the AWS RoboMaker console, launch the Object Following using Reinforcement Learning sample application.

Topics

- View Training Metrics (p. 18)
- View Trained Machine Learning Models (p. 19)

View Training Metrics

The Object Following using Reinforcement Learning sample application uses AWS RoboMaker cloud extensions to write training metrics to Amazon CloudWatch.

To view robot training metrics

2. In the CloudWatch console, select Metrics.
3. On the Metrics page, in the All metrics tab, select AWSRoboMakerSimulation.
4. Select the metric named ObjectTrackerRewardPerEpisode.
5. Hover over the graph to see values for that moment. This graph plots the total reward received by the robot in each trial/episode. An increase in reward with time indicates that the robot is getting better at finding and following the target object.

View Trained Machine Learning Models

When a simulation job runs, a frozen TensorFlow graph with weights is written to an Amazon S3 bucket. This file can be deployed to a robot.

To view the TensorFlow data

1. Open the Amazon Simple Storage Service console at https://console.aws.amazon.com/s3/. Open the bucket name beginning with awsrobomakerobjecttracker.
2. In the bucket, open the folder named model-store/model, and then select model.pb.

Self-Driving using Reinforcement Learning

In this section, you teach a race car to drive in a simulation. You use reinforcement learning and a coach reinforcement learning library. View the reward metrics in Amazon CloudWatch Metrics to explore how the machine learning model improves over time. Customize your reward function to improve the machine learning algorithm used for training.

Before you use AWS RoboMaker for the first time, complete the tasks in Create an Account (p. 9). Then, in the AWS RoboMaker console, launch the Self-Driving using Reinforcement Learning sample application.

Topics

- View Training Metrics (p. 19)
- View Trained Machine Learning Models (p. 19)

View Training Metrics

The Self-Driving using Reinforcement Learning sample application uses AWS RoboMaker cloud extensions to write training metrics to Amazon CloudWatch.

To view robot training metrics

2. In the CloudWatch console, select Metrics.
3. On the Metrics page, in the All metrics tab, select AWSRoboMakerSimulation.
4. Select the metric named DeepRacerRewardPerEpisode.
5. Hover over the graph to see values for that moment. This graph plots the total reward received by the robot in each trial or episode. An increased reward over time indicates that the race car is improving its performance on the race track.

View Trained Machine Learning Models

When a simulation job runs, a frozen TensorFlow graph with weights is written to an Amazon S3 bucket. This file can be deployed to a DeepRacer.
To view the TensorFlow data

1. Open the Amazon Simple Storage Service console at https://console.aws.amazon.com/s3/. Open the bucket name beginning with awsrobomakerdeepracer.
2. In the bucket, open the folder named model-store/model, and then select model.pb.
Sample Applications

AWS RoboMaker comes with sample programs you can launch in the AWS RoboMaker console. You can also download and build them on your own. Each sample has a repo on GitHub. The README file describes the sample and how to build it. The README tells how it can be run on your desktop and in AWS RoboMaker.

- **Hello world** — Learn the basics of how to create your robot applications and simulation applications. Start from a basic project template including a robot in an empty world.
- **Robot monitoring** — Monitor health and operational metrics for a robot in a simulated bookstore. Use Amazon CloudWatch Metrics and Amazon CloudWatch Logs.
- **Navigation and person recognition** — The robot navigates between goal locations in a simulated home and recognizes faces in photos. It also streams camera images to Kinesis Video Streams, receives face recognition results from Amazon Rekognition, and speaks the names of recognized people using Amazon Polly.
- **Voice commands** — Command a robot through natural language text and voice in a simulated bookstore using Amazon Lex. Default commands include "move", "turn" and "stop".
- **Self-driving using reinforcement learning** — Teach a race car to drive in a simulation through reinforcement learning in simulation using the Coach Reinforcement Learning Library. View reward metrics in Amazon CloudWatch Metrics and customize the reward function to improve the machine learning algorithm used for training.
- **Object following using reinforcement learning** — Teach a robot to track and follow an object through reinforcement learning in simulation using the Coach Reinforcement Learning Library. View reward metrics in Amazon CloudWatch Metrics and customize the reward function to improve the machine learning algorithm used for training.

In this section, you learn how to launch a sample application in the AWS RoboMaker console. You will also how to configure permissions if you want to provide your own IAM role when launching a sample application.

**Topics**
- Launching a Sample Application (p. 21)
- Configuring Permissions (p. 22)

Launching a Sample Application

The AWS RoboMaker console makes it easy to launch a pre-built AWS RoboMaker sample application. You can launch an application using default settings and let AWS RoboMaker manage permissions. You can also choose a different Gazebo version and use a custom IAM role.

**To launch a sample application**

Follow the steps under one of the following tabs.

2. In the left pane, choose Resources, and then choose Sample applications.
3. Select one of the sample applications.
4. **Optional**: Select Additional settings to view additional configuration options.
5. Optional: Select a ROS distribution. ROS Melodic is the latest version. It works with Gazebo 9. For more information about the Robot Operating System (ROS), see [www.ros.org](http://www.ros.org).
6. Optional: Select a Simulation software suite. Each sample application is built for Gazebo 7 and Gazebo 9. Gazebo 9 has additional features for robotics developers.

7. Optional: Select an IAM role. This role will be used by AWS RoboMaker to create the sample application environment and launch it. It will also be used by the sample application to access resources like Amazon Rekognition.

   **Note**
   If you are a student or educator accessing AWS RoboMaker through the AWS Educate portal, select the `robomaker_students` IAM role.

For more information about the permissions required by the sample applications, see ??? (p. 22)

### Configuring Permissions

When you launch a sample program in the AWS RoboMaker console, you can provide an IAM role to use. The permissions vary by sample program. This section describes what you need to launch each sample.

For more information about AWS Identity and Access Management roles, see Creating a Role to Delegate Permissions to an AWS Service.

#### Minimum permissions

To launch a sample application, you need a role that has:

- A trust relationship with `robomaker.amazonaws.com`.
- A trust relationship with `lambda.amazonaws.com`.
- Sample application permissions.

Use the following permissions to launch Hello world, Robot monitoring, Self-driving, and the Object-following samples. Other samples use these permissions and a set of additional permissions.

Replace `account#` with your account number.

```json
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "s3:ListBucket",
                "s3:GetObject",
                "s3:PutObject"
            ],
            "Resource": [
                "*"
            ],
            "Effect": "Allow"
        },
        {
            "Action": [
                "iam:PassRole"
            ],
            "Resource": "arn:aws:iam::account#:role/*",
            "Effect": "Allow"
        },
        {
            "Action": [
                "robomaker:*"
            ],
            "Effect": "Allow"
        }
    ]
}
```
Navigation

With the minimum permissions (p. 22), the navigation sample program requires the permissions listed here.

Replace account# with your account number.
"Action": [ "iam:PassRole" ],
  "Resource": "arn:aws:iam::account#:role/*",
  "Effect": "Allow" },
{ "Action": [ "cloudformation:DescribeStacks" ],
  "Resource": "arn:aws:cloudformation::*:account#:stack/*",
  "Effect": "Allow" },
{ "Action": [ "kinesis:DeleteStream",
  "kinesis:DescribeStream",
  "kinesis:CreateStream" ],
  "Resource": "arn:aws:kinesis::*:account#:stream/AmazonRekognitionPersonDetectionStream*",
  "Effect": "Allow" },
{ "Action": [ "kinesisvideo:CreateStream" ],
  "Resource": "*",
  "Effect": "Allow" },
{ "Action": [ "kinesisvideo:DescribeStream",
  "kinesisvideo:DeleteStream" ],
  "Resource": "arn:aws:kinesisvideo::*:account#:stream/RoboMakerPersonDetectionVideoStream*",
  "Effect": "Allow" },
{ "Action": [ "rekognition:CreateCollection",
  "rekognition:DeleteCollection",
  "rekognition:IndexFaces" ],
  "Resource": "arn:aws:rekognition::*:account#:collection/roboMakerSampleAppPersonDetectionCollection*",
  "Effect": "Allow" },
{ "Action": [ "rekognition:CreateStreamProcessor" ],
  "Resource": [ "arn:aws:rekognition::*:account#:streamprocessor/personDetectionStreamProcessor*",
  "arn:aws:rekognition::*:account#:collection/roboMakerSampleAppPersonDetectionCollection*" ],
  "Effect": "Allow" },
{ "Action": [ "rekognition:DeleteStreamProcessor",
  "rekognition:StartStreamProcessor",
  "rekognition:StopStreamProcessor" ],
Person detection

With the minimum permissions (p. 22), the person detection sample requires a trust relationship with rekognition.amazonaws.com. Attach the policy arn:aws:iam::aws:policy/service-role/AmazonRekognitionServiceRole to the role.

This also requires the permissions listed here. Replace account# with your account number.

```json
{
    "Action": [
        "iam:PassRole"
    ],
    "Resource": "arn:aws:iam::account#:role/*",
    "Effect": "Allow"
},
{
    "Action": [
        "cloudformation:DescribeStacks"
    ],
    "Resource": "arn:aws:cloudformation::account#:stack/*",
    "Effect": "Allow"
},
{
    "Action": [
        "kinesis:DeleteStream",
        "kinesis:DescribeStream",
        "kinesis:CreateStream"
    ],
    "Resource": "arn:aws:kinesis::account#:stream/AmazonRekognitionPersonDetectionStream*",
    "Effect": "Allow"
},
{
    "Action": [
        "kinesisvideo:CreateStream"
    ],
    "Resource": "*",
    "Effect": "Allow"
},
{
    "Action": [
        "kinesisvideo:DescribeStream",
        "kinesisvideo:DeleteStream"
    ],
    "Resource": "arn:aws:kinesisvideo::account#:stream/RoboMakerPersonDetectionVideoStream*",
    "Effect": "Allow"
},
{
    "Action": [
        "rekognition:CreateCollection"
    ]
}
```
Voice commands

With the minimum permissions (p. 22), the voice command sample requires the permissions listed below.

Replace \texttt{account\#} with your account number.

```
{
    "Action": [
        "lex:CreateBotVersion",
        "lex:GetBot",
        "lex:GetBotAlias",
        "lex:GetIntent",
        "lex:GetSlotType",
        "lex:GetBot",
        "lex:GetBotAlias",
        "lex:GetIntent",
        "lex:GetSlotType"
    ],
    "Resource": [
        "arn:aws:lex:*:\texttt{account\#}::*"
    ],
    "Effect": "Allow"
},
{
    "Action": [
        "s3:PutBucketNotification"
    ],
    "Resource": "+",
    "Effect": "Allow"
}
```
"lex:StartImport"
],
"Resource": "+",
"Effect": "Allow"
}
Application Versioning

AWS RoboMaker supports creating more than one version of your robot applications and simulation applications. This helps you control which code your robots and simulations use. A version is a numbered snapshot of the $LATEST version of your application. You can create a version to use in different parts of your development workflow. For example, development, beta deployment, or production.

When you version an AWS RoboMaker robot application or simulation application you create a snapshot of the application. AWS RoboMaker remembers the Amazon S3 path and ETag of the file for each version. You can use the version of the application as it existed when the version was made provided it still exists in the Amazon S3 path and has not been altered (its ETag is unchanged).

You can create a maximum of 40 versions per application.

Topics
- The $LATEST Version (p. 28)
- Updating an Application Version (p. 28)
- Deleting an Application Version (p. 29)

The $LATEST Version

When you create a version, AWS RoboMaker takes a snapshot of the $LATEST version and increments the version number by 1. AWS RoboMaker remembers the Amazon S3 path and ETag of the file. The path is used to retrieve the file. The ETag is used to confirm it has not changed. Version numbers are never reused. For example, if your latest version is 10 and you remove it and then create a new version, the new version will be version 11.

You can update the $LATEST version as you develop your application. When you select the $LATEST version, it will be retrieved from the Amazon S3 location you specify. For example, if you start a simulation job using the latest version of your robot application and simulation application, then make changes to the robot application at the Amazon S3 path, and then restart the simulation job, the updated robot application will be used.

When you deploy a robot application, you must select a specific numbered version to deploy. For more information on how to create a robot application version, see Creating a Robot Application Version (p. 53).

For more information how to create a simulation application version, see Creating a Simulation Application Version (p. 57). For more information about ETags, see Common Response Headers.

Updating an Application Version

You can update only the $LATEST version of an AWS RoboMaker application. When you do this, it is available to use in AWS RoboMaker. For example, if you restart a simulation job, the latest version of the applications will be used in the simulation.

For more information, see Updating a Robot Application (p. 54) and Updating a Simulation Application (p. 58).
Deleting an Application Version

When you no longer need an application version, delete it. For more information, see Deleting a Robot Application Version (p. 55) and Deleting a Simulation Application Version (p. 60).
Developing Applications

This section helps you get set up to develop with the Robot Operating System (ROS). You learn how to create robot applications and simulation applications with the configured AWS Cloud9 development environment. You learn about how to extend your robot application with AWS RoboMaker cloud extensions.

It also describes how to create and manage AWS RoboMaker robot applications and AWS RoboMaker simulation applications.

Topics
- Creating a New Robotic Application (p. 30)
- Building and Bundling Robotic Applications with Colcon (p. 37)
- Creating a Simulation Job (p. 38)
- AWS RoboMaker Cloud Extensions (p. 42)
- Developing with AWS Cloud9 (p. 43)

Creating a New Robotic Application

This section describes how to create a robotic application. It uses a directory structure that separates the robot application and the simulation application. This makes it easier to use in AWS RoboMaker simulations and other stages of robotics development.

Robotics applications usually include both a robot application and a simulation application. A robot application is deployed to the physical robot. Simulation applications are used to model aspects of the physical world. Using an AWS RoboMaker simulation job, robot applications can run inside of simulation applications and data can be collected and visualized.

AWS RoboMaker robotics applications usually have the following directory structure and files:

```
MyApplication
### robot_ws                               # workspace for the robot system
#   ### src
#       ### robot_app                      # ROS package for the robot application
#           ### CMakeLists.txt             # build config
#           ### launch
#           ### package.xml                # ROS package config
#           ### scripts
#           #   ### rotate.launch          # robot entrypoint, specifies running system
#           ### setup.py                   # allow ROS to find your python code
#           ### src
#               ### robot_app
#                   ### __init__.py        # python module for any .py code
#   ### simulation_ws                          # workspace for the simulation
#   ### src
#       ### simulation_app                 # ROS package for the simulation application
#           ### CMakeLists.txt             # build config
#           ### launch
#           #   ### example.launch         # simulation entrypoint, specifies world, etc
#           #   ### spawn_turtlebot.launch # launch file for spawning the simulated robot
#           ### package.xml                # ROS package config
#           ### worlds
```
### Prerequisites

You need to have a development environment configured for robotics development using AWS RoboMaker. Your development environment must have the following:

- Robot Operating System (ROS) Kinetic or Melodic
- Colcon

To create a preconfigured robotics development environment in AWS Cloud9, see [Creating a Development Environment](#).  

### Create the Robot Application Workspace

The robot application workspace contains custom ROS nodes and other assets needed by your robot application.

**To create the robot application workspace**

1. Open the command prompt.
2. Create the project directory, then move to the new directory.

```
$ mkdir MyApplication
$ cd MyApplication
```

3. Create directories for ROS launch files, ROS nodes, deployment scripts, and source folders.

```
$ mkdir -p robot_ws/src/robot_app
$ cd robot_ws/src/robot_app
$ mkdir -p launch scripts src/robot_app
```

4. Create an empty `__init__.py` file for using Python with ROS.

```
$ touch src/robot_app/__init__.py
```

5. Copy the following Python script into a file named `rotate.py` and then save it to the `scripts` directory. This sample node periodically rotates the robot. Your application will likely have more than one node and more sophisticated code.

```python
#!/usr/bin/env python
import rospy
from geometry_msgs.msg import Twist

class Rotator():
    def __init__(self):
        self._cmd_pub = rospy.Publisher('/cmd_vel', Twist, queue_size=1)
```

---

---
def rotate_forever(self):
    self.twist = Twist()
    r = rospy.Rate(10)
    while not rospy.is_shutdown():
        self.twist.angular.z = 0.1
        self._cmd_pub.publish(self.twist)
        rospy.loginfo("Rotating robot: %s", self.twist)
        r.sleep()

def main():
    rospy.init_node('rotate')
    try:
        rotator = Rotator()
        rotator.rotate_forever()
    except rospy.ROSInterruptException:
        pass
    if __name__ == '__main__':
        main()

6. Make the Python script executable so it can be found by roslaunch. roslaunch is used to start the
nodes in your application.

    $ chmod +x scripts/rotate.py

7. Copy the following XML into a file named rotate.launch and then save it to the launch
directory. The launch file is configured to start the rotate node.

    <launch>
        <!--
        Using simulation time means nodes initialized after this
        will not use the system clock for its ROS clock and
        instead wait for simulation ticks.

        See http://wiki.ros.org/Clock

        Note: set to false for deploying to a real robot.
        -->
        <arg name="use_sim_time" default="true"/>
        <param name="use_sim_time" value="$(arg use_sim_time)"/>

        <!-- Rotate the robot on launch -->
        <node pkg="robot_app" type="rotate.py" name="rotate" output="screen"/>
    </launch>

8. Copy the following into a file named CMakeLists.txt and save it to the robot_app directory. For
more information on creating make files for ROS, see CMakeLists.txt.

    # Set minimum required version of cmake, project name and compile options
    cmake_minimum_required(VERSION 2.8.3)

    # Mention your package name
    project(robot_app)

    # Find catkin packages and libraries for catkin and system dependencies
    find_package(catkin REQUIRED COMPONENTS)
Create the Robot Application Workspace

9. Copy the following XML into a file named package.xml and then save it to the robot_app directory. It includes all robot application dependencies.

```xml
<?xml version="1.0"?>
<package format="2">
  <name>robot_app</name>
  <version>1.0.0</version>
  <description>
    A custom AWS RoboMaker robot package with a rotating Turtlebot3
  </description>
  <license>MIT</license>
  <author email="aws-robomaker@amazon.com">AWS RoboMaker</author>
  <maintainer email="aws-robomaker@amazon.com">AWS RoboMaker</maintainer>
</package>
```
Create the Simulation Application Workspace

The simulation application workspace contains models for the robot and terrain. It also includes custom ROS nodes and other assets needed by your simulation application.

1. Open the command prompt and move to the project directory, then run the following commands to create simulation application directories.

```
$ mkdir -p simulation_ws/src/simulation_app
$ cd simulation_ws/src/simulation_app
$ mkdir launch worlds
```

2. Copy the following XML into a file named `example.launch` and then save it to the `launch` directory. It loads the simulated world with a Turtlebot.

```
<launch>
  <!-- Always set GUI to false for RoboMaker Simulation
       Use gui:=true on roslaunch command-line to run with a gui. -->
  <arg name="gui" default="false"/>

  <include file="$(find gazebo_ros)/launch/empty_world.launch">
    <arg name="world_name" value="$(find simulation_app)/worlds/example.world"/>
    <arg name="paused" value="false"/>
    <arg name="use_sim_time" value="true"/>
    <arg name="gui" value="$(arg gui)"/>
    <arg name="headless" value="false"/>
    <arg name="debug" value="false"/>
    <arg name="verbose" value="true"/>
  </include>

  <!-- Spawn Robot
      This must be the same robot type as the robot application -->
  <include file="$(find simulation_app)/launch/spawn_turtlebot.launch">
    <!-- Override arg parameters here e.g,
```
3. Copy the following XML into a file named `spawn_turtlebot.launch` and then save it to the `launch` directory. It spawns a Turtlebot robot into the simulation.

```xml
<launch>
<!-- Optional environment variable, default is "waffle_pi". Note that "burger" does not have a camera -->
<arg name="model" default="$(optenv TURTLEBOT3_MODEL waffle_pi)" doc="model type [burger, waffle, waffle_pi]"/>

<!-- You may override arg parameters when including this launch file -->
<arg name="x_pos" default="0.0"/>
<arg name="y_pos" default="0.0"/>
<arg name="z_pos" default="0.0"/>
<arg name="roll" default="0.0"/>
<arg name="pitch" default="0.0"/>
<arg name="yaw" default="0.0"/>

<!-- Spawn the robot into Gazebo with the turtlebot description -->
<param name="robot_description" command="$(find xacro)/xacro --inorder $(find turtlebot3_description)/urdf/turtlebot3_$(arg model).urdf.xacro"/>

<node pkg="gazebo_ros" type="spawn_model" name="spawn_urdf"
args="-urdf -param robot_description -model turtlebot3_$(arg model)
-x $(arg x_pos) -y $(arg y_pos) -z $(arg z_pos) -R $(arg roll) -P $(arg pitch) -Y $(arg yaw)"/>

<!-- Publish robot and joint states. This allows rviz to display robot data, and in the robot's coordinate frame. These nodes could go into the robot application .launch files instead. -->
</node>
</launch>
```

4. Copy the following XML into a file named `example.world` and then save it to the `worlds` directory. The world file defines the static and dynamic objects in a simulated environment. For more information about building worlds for Gazebo, see Building a World.

```xml
<?xml version="1.0" encoding="utf-8"?>
<sdf version="1.4">
<world name="default">
<gui>
  <camera name="default_camera">
    <pose>0.8 -0.75 0.35 0 0.25 2.35</pose>
  </camera>
</gui>

<include>
  <uri>model://sun</uri>
</include>

<include>
  <uri>model://ground_plane</uri>
</include>
</world>
</sdf>
```
<physics type="ode">
    <real_time_update_rate>1000.0</real_time_update_rate>
    <max_step_size>0.001</max_step_size>
    <real_time_factor>1</real_time_factor>
    <ode>
        <solver>
            <type>quick</type>
            <iters>150</iters>
            <precon_iters>0</precon_iters>
            <sor>1.400000</sor>
            <use_dynamic_moi_rescaling>1</use_dynamic_moi_rescaling>
        </solver>
        <constraints>
            <cfm>0.00001</cfm>
            <erp>0.2</erp>
            <contact_max_correcting_vel>2000.000000</contact_max_correcting_vel>
            <contact_surface_layer>0.01000</contact_surface_layer>
        </constraints>
    </ode>
</physics>

<scene>
    <ambient>0.4 0.4 0.4 1</ambient>
    <background>0.7 0.7 0.7 1</background>
    <shadows>true</shadows>
</scene>

</world>
</sdf>

5. Copy the following text into a file named CMakeLists.txt and then save it to the simulation_app directory.

```cmake
cmake_minimum_required(VERSION 2.8.3)
project(simulation_app)
find_package(catkin REQUIRED COMPONENTS
gazebo_ros
)
catkin_package(DEPENDS gazebo_ros)
install(DIRECTORY launch worlds
          DESTINATION ${CATKIN_PACKAGE_SHARE_DESTINATION})
```

6. Copy the following XML into a file named package.xml and then save it to the simulation_app directory. It includes all simulation application dependencies.

```xml
<?xml version="1.0"?>
<package format="2">
    <name>simulation_app</name>
    <version>1.0.0</version>
    <description>
        A custom AWS RoboMaker simulation package with a TurtleBot3 in an empty Gazebo world.
    </description>
    <license>MIT</license>
    <author email="aws-robomaker@amazon.com">AWS RoboMaker</author>
    <maintainer email="aws-robomaker@amazon.com">AWS RoboMaker</maintainer>
    <buildtool_depend>catkin</buildtool_depend>
    <depend>gazebo_ros</depend>
</package>
```
Build the Robot and Simulation Application Bundles

AWS RoboMaker works with robotics applications built and bundled with colcon. For more information, see Building and Bundling Robotic Applications with Colcon (p. 37).

1. Open the command line and move to the `robot_ws` directory, then run the following commands.

```bash
# rosdep update
# rosdep install --from-paths src --ignore-src -r -y
# colcon build
# colcon bundle
```

If you are launching from the command-line, run the following commands:

```bash
# source install/setup.sh
# roslaunch robot_app rotate.launch
```

Use the following to launch the robot application in AWS RoboMaker:

```bash
roslaunch robot_app rotate.launch
```

2. Move to the `simulation_ws` directory, then run the following commands.

```bash
# rosdep update
# rosdep install --from-paths src --ignore-src -r -y
# colcon build
# colcon bundle
```

If you are launching from the command-line, run the following commands:

```bash
# source install/setup.sh
# roslaunch simulation_app example.launch
```

Use the following to launch the simulation application in AWS RoboMaker:

```bash
roslaunch simulation_app example.launch
```

Building and Bundling Robotic Applications with Colcon

AWS RoboMaker works with robotics applications built and bundled with colcon. colcon is a command line tool built by the Open Source Robotics Foundation (OSRF). It automates the building and bundling of ROS and ROS2 applications. It should be a drop-in replacement for catkin_make.
For more information about colcon, see Colcon. If you experience issues while building with colcon, see colcon-ros. For problems bundling with colcon, see colcon-bundle.

Topics
- Installing Colcon (p. 38)
- Using Colcon to Build and Bundle (p. 38)

Installing Colcon

Use the following commands to install colcon:

```bash
apt-get update
apt-get install python3-pip python3-apt
pip3 install -U setuptools
pip3 install -U colcon-common-extensions colcon-ros-bundle
```

If you already have colcon installed, you can install bundling support with the following command:

```bash
pip3 install -U colcon-ros-bundle
```

Using Colcon to Build and Bundle

Use the following commands to build and then bundle your robotics application:

```bash
cd robotic-application-workspace
colcon build
colcon bundle
```

Creating a Simulation Job

This section describes how to create a simulation job from the command line.

Topics
- Prerequisites (p. 38)
- Create Source and Output Amazon S3 Buckets (p. 39)
- Create a Robot Application (p. 39)
- Create a Simulation Application (p. 39)
- Create a Simulation Job Role (p. 40)
- Create a Simulation Job (p. 41)

Prerequisites

To create an AWS RoboMaker simulation job from the command line, you need the following:

- The AWS Command Line Interface (AWS CLI). For more information about installing the AWS CLI, see Installing the AWS CLI.
- A robot application and a simulation application. They must be bundled with colcon and target the X86_64 architecture. To create a simple robotics application and simulation application from scratch, see Creating a New Robotic Application (p. 30).
Create Source and Output Amazon S3 Buckets

Before you create a simulation job, you need to create a bucket to use as source location for your applications. You can also create a bucket for output generated during the simulation job.

1. Open the command prompt.
2. Create a bucket for your application source. This bucket will be the source location for your robot and simulation applications. Select a unique bucket name.

   ```
   $ aws s3 mb s3://myapplicationsource
   ```

3. Create a bucket for output generated by the simulation job. AWS RoboMaker uploads ROS bags, ROS logs and Gazebo logs when the simulation job completes. Select a unique bucket name.

   ```
   $ aws s3 mb s3://mysimulationjoboutput
   ```

Create a Robot Application

Before you can create a simulation job, you need to create a robot application in AWS RoboMaker. It contains details like target architecture and ROS version. It can be used in a simulation job and can be deployed to physical robots in a fleet.

1. Open the command prompt.
2. Copy the robot application source bundle to your Amazon S3 bucket. It is renamed to `my-robot-application.tar.gz` during the copy. The bundle might be `.tar` or `.tar.gz` extension.

   The robot application must be built for the `X86_64` architecture.

   ```
   $ aws s3 cp robot_ws/bundle/output.tar.gz s3://myapplicationsource/my-robot-application.tar.gz
   ```

3. Create a robot application in AWS RoboMaker.

   ```
   $ aws robomaker create-robot-application --name MyRobotApplication --sources 
   s3Bucket=myapplicationsource,s3Key=my-robot-application.tar.gz,architecture=X86_64 -- 
   robot-software-suite name=ROS,version=Kinetic
   ```

   The call returns information about the newly created robot application. You will use the Amazon Resource Name (ARN) when you create the simulation job.

Create a Simulation Application

A simulation application contains all of the assets and logic needed to simulate an environment. You need to create a simulation application in AWS RoboMaker before you can create a simulation job.

1. Open the command prompt.
2. Copy the simulation application source bundle to your Amazon S3 bucket. The simulation application must be built for the `X86_64` architecture. The bundle might be `.tar` or `.tar.gz` extension.
Create a Simulation Job Role

When you create a simulation job, you need to specify an IAM role AWS RoboMaker can use to access resources like Amazon S3 buckets and Amazon CloudWatch Logs. The role will also be used by your robot application to access resource it consumes like Amazon Lex or Amazon Rekognition.

If you have already created a role, you can skip to Create a Simulation Job (p. 41).

To create the simulation job role

1. Sign in to the AWS Management Console and open the AWS Identity and Access Management console at console.aws.amazon.com/iam.
2. Create the access policy. On the left, choose Policies, then choose Create policy. Choose JSON and paste the code below:

```json
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": "s3:ListBucket",
            "Resource": [
                "arn:aws:s3:::my-input-bucket"
            ],
            "Effect": "Allow"
        },
        {
            "Action": ["s3:Get*", "s3:List*"],
            "Resource": [
                "arn:aws:s3:::my-input-bucket/*"
            ],
            "Effect": "Allow"
        },
        {
            "Action": ["s3:Put*"],
            "Resource": [
                "arn:aws:s3:::my-output-bucket/*"
            ],
            "Effect": "Allow"
        },
    ]
}
```
Create a Simulation Job

When you create a simulation job, you specify the applications in the simulation and their launch configurations, the IAM role, and simulation duration.

**To create a simulation job**

1. Open the command prompt.
2. Create the simulation job.
3. Check on the status of your simulation job using the following commands.

```bash
# aws robomaker list-simulation-jobs
```

Once the simulation has a status of Running, you can use tools like Gazebo, rqt, rviz and terminal to visualize sensors, devices, and other aspects of the simulation. To access the tools, open the AWS RoboMaker console at https://console.aws.amazon.com/robomaker/, then choose Simulation jobs, then select your simulation. Scroll down and select one of the tools.

For more information about simulation tools, see Simulation Tools (p. 83).

**AWS RoboMaker Cloud Extensions**

AWS RoboMaker cloud extensions is a collection of ROS packages you can use with your robot and simulation applications to easily access AWS. The extensions enable a number of exciting scenarios. For example you might capture video and other sensor data from your physical robot as it explores the environment. You might provide a voice for your robot and process voice commands. You might recognize faces and objects.

Available packages include the following.

- **Amazon CloudWatch Metrics** — Stream sensor data, performance metrics, and other information from your robots. View data over time and set alarms to receive alerts if data reaches certain thresholds (like low battery).
- **Amazon CloudWatch Logs** — Stream logging data from your robot fleets to a central place for easy analysis. Search data generated by hundreds of robots in one place.
- **Amazon Kinesis Video Streams** — Stream real-time video from your robot into AWS.
- **Amazon Lex** — Create a robot with engaging user experiences and lifelike conversation.
- **Amazon Polly** — Turn text into speech using lifelike voices in different languages.

To see these extensions in action, visit the AWS RoboMaker sample application page AWS RoboMaker sample applications. You can quickly and easily launch and engage with different robot simulations.

**Prerequisites**

To use the AWS RoboMaker cloud extensions, you must have:

- An AWS account. To sign up for an AWS account, see Step 1: Create an AWS Account and an Administrator (p. 9).
- AWS credentials. These credentials are used to access AWS services.
- Configure IAM role permissions to allow AWS RoboMaker cloud extensions to use AWS services. The permissions needed are included in the README files included with each extension.
Installing AWS RoboMaker cloud extensions

To install the AWS RoboMaker cloud extensions, follow the instructions provided in the README file for each package.

- Amazon Kinesis & Amazon Rekognition
- Amazon Lex
- Amazon Polly
- Amazon CloudWatch Logs
- Amazon CloudWatch Metrics

Developing with AWS Cloud9

You can create an AWS Cloud9 development environment for AWS RoboMaker and robotics development. The environment is preconfigured with ROS and integrated with other AWS RoboMaker capabilities. With it, you can manage build configurations, create simulation jobs, and explore running simulations with Gazebo, rviz, rqt and a terminal.

You access the AWS Cloud9 development environment through a web browser.

Topics
- Creating a Development Environment (p. 43)
- Working with the development environment (p. 44)
- Deleting an Environment (p. 51)

Creating a Development Environment

In this section, you create a development environment and access it from the browser.

Note
Completing these procedures might result in charges to your AWS account. These include possible charges for services such as Amazon EC2 and AWS RoboMaker. For more information, see Amazon EC2 Pricing, AWS RoboMaker Pricing, and Cloud Services Pricing.

To create a development environment

Follow these steps:

2. In the left navigation pane, choose Development, and then choose IDEs.
3. In the Create AWS RoboMaker development environment page, type a name for the environment.
4. Choose a ROS Distribution. For more information about the Robot Operating System (ROS), see www.ros.org. For more information about using ROS 2 with AWS RoboMaker, see ROS 2 Dashing (Beta) (p. 3).
5. For Instance type, choose an instance type with the amount of RAM and vCPUs you think you need for the kinds of tasks you want to do. Or leave the default choice.
Note
Choosing instance types with more RAM and vCPUs might result in additional charges to your AWS account for Amazon EC2.

6. In Networking, if your development environment needs to access resources on an Amazon VPC, select the VPC and subnets.
7. Choose Create to create the development environment.
8. On the Environment details page, choose Open environment. It might take a few moments to prepare the environment.

You can list available development environments by choosing Development in the left navigation pane, then choosing Development environments.

Working with the development environment

This section provides more information about AWS RoboMaker functionality in the AWS Cloud9 development environment.

Topics
- Manually Switching Gazebo Versions (p. 44)
- Environment Settings (p. 45)
- Creating a Workflow (p. 46)
- Menu Commands Reference (p. 49)

Manually Switching Gazebo Versions

This section describes how to change the version of Gazebo that is supported in the AWS Cloud9 development environment. This applies only to the ROS Kinetic development environment.

When you switch Gazebo versions, it affects the entire development environment. All of the robot applications and simulation applications in the development environment will be built and bundled in the switched environment.

Note
Simulation applications are dependent on a version. To change the version, you need to update the dependencies in your application.

Topics
- Switch from Gazebo 7 to Gazebo 9 (p. 44)
- Switch from Gazebo 9 to Gazebo 7 (p. 45)

Switch from Gazebo 7 to Gazebo 9

To switch from Gazebo 7 environment to Gazebo 9 environment

1. Open the command prompt.
2. Run the following code to create a shell script named `gazebo7to9.sh` in the current directory:

```bash
$ echo "Switching to Gazebo9..."
```

sudo touch /etc/apt/sources.list.d/gazebo-stable.list
sudo chmod 666 /etc/apt/sources.list.d/gazebo-stable.list
sudo echo "deb http://packages.osrfoundation.org/gazebo/ubuntu-stable 'lsb_release -cs' main" > /etc/apt/sources.list.d/gazebo-stable.list
sudo wget http://packages.osrfoundation.org/gazebo.key -O - | sudo apt-key add -
sudo apt-get update

sudo touch /etc/ros/rosdep/sources.list.d/00-gazebo9.list
sudo chmod 666 /etc/ros/rosdep/sources.list.d/00-gazebo9.list
sudo echo "yaml https://github.com/osrf/osrf-rosdep/raw/master/gazebo9/gazebo.yaml
yaml https://github.com/osrf/osrf-rosdep/raw/master/gazebo9/releases/lunar.yaml lunar"
>> /etc/ros/rosdep/sources.list.d/00-gazebo9.list

sudo apt-get remove ros-kinetic-gazebo*
sudo apt-get remove libgazebo*
sudo apt-get remove gazebo*

echo "Done switching..." > gazebo7to9.sh && chmod +x gazebo7to9.sh

You only need to create this script once in your environment.

3. On the command line, run the script you created:

$ gazebo7to9.sh

The environment will configured to build and bundle applications using Gazebo 9.

**Switch from Gazebo 9 to Gazebo 7**

**To switch from Gazebo 9 environment to Gazebo 7 environment**

1. Open the command prompt.
2. Run the following code to create a shell script named gazebo9to7.sh in the current directory:

```bash
#!/bin/bash
set -e

echo "Switching to Gazebo7..."
sudo apt-get remove ros-kinetic-gazebo*
sudo apt-get remove libgazebo*
sudo apt-get remove gazebo*
sudo rm /etc/ros/rosdep/sources.list.d/00-gazebo9.list
echo "Done switching..." > gazebo9to7.sh && chmod +x gazebo9to7.sh
```

3. On the command line, run the script you created:

$ gazebo9to7.sh

The environment will configured to build and bundle applications using Gazebo 7.

**Environment Settings**

The environment settings configure elements that apply to all of the projects in the AWS RoboMaker development environment.
Simulation Software Suite

The development environment can be configured to build and bundle with different versions of Gazebo. Only one version can be active at a time.

**Note**
The software simulation suite selected for the simulation application must match the environment setting.

Creating a Workflow

This section describes how to create an AWS RoboMaker workflow for an existing application.

A workflow is a collection of tasks that will be run sequentially in order. Tasks include building an application, bundling an application, and creating a simulation job. Workflows and tasks are created in the AWS RoboMaker AWS Cloud9 development environment. Workflows can be selected and run from the AWS RoboMaker Run menu.

Workflows are useful for performing repetitive tasks during iterative development. In this section, you create a workflow that performs the following tasks:

- Build a robot application and simulation application with colcon.
- Bundle a robot application and simulation application with colcon.
- Create a simulation job.

Prerequisites

To create an AWS RoboMaker simulation job from the command line, you need the following:

- A AWS RoboMaker development environment. For more information about creating a development environment, see Creating a Development Environment (p. 43).
- A robot application and a simulation application. To create a simple robotics application and simulation application from scratch, see Creating a New Robotic Application (p. 30).

Optionally, you can use the the AWS RoboMaker console for easy access to simulation tools like Gazebo, rqt, rviz and terminal. These tools are available on the Simulation Job Detail page.

Create Source and Output Amazon S3 Buckets

Before you create a simulation job, you need to create a bucket to use as source location for your applications. To create your buckets, follow the steps in Create Source and Output Amazon S3 Buckets (p. 39).
Create a Simulation Job Role

A simulation job requires an IAM role with access to resources like source Amazon S3 buckets and Amazon CloudWatch Logs. To create a simulation job role, follow the steps in Create a Simulation Job Role (p. 40).

Create Application Build Configurations

AWS RoboMaker uses colcon to build robot applications and simulation applications.

To create robot application and simulation application build configurations

1. Open your AWS RoboMaker development environment containing your robotic application.
2. In the AWS Cloud9 menu, choose RoboMaker Run, then choose Build, and then select Add or Edit Configurations.
3. In the RoboMaker Configuration dialog, under Create New Configuration, select Colcon build.
4. In the RoboMaker Configuration dialog, under Colcon Build Configuration, provide a name for the robot application build configuration. For example, MyRobot v3 build, bundle, and create simulation.
5. Type in the path to the robot application working directory. You can also select ... to use the file picker to select the working directory. The working directory will contain CMakeLists.txt.
6. Repeat the steps to create a simulation application build configuration. Set the Working directory to the simulation application workspace.
7. To start a build, choose RoboMaker Run, then choose Build, and then select a build configuration. A new terminal window will open at the bottom of the IDE. Use it to track build progress.

Create Application Bundle Configurations

AWS RoboMaker uses colcon to bundle robot applications and simulation applications.

To create robot application and simulation application build configurations

1. Open your AWS RoboMaker development environment containing your robotic application.
2. In the AWS Cloud9 menu, choose RoboMaker Run, then choose Bundle, and then select Add or Edit Configurations.
4. In the RoboMaker Configuration dialog, under Colcon Bundle Configuration, provide a name for the robot application bundle configuration.
5. Type in the path to the robot application working directory. You can also select ... to use the file picker to select the working directory. The working directory will contain CMakeLists.txt. For example, ./MySample/robot_ws.
6. Repeat the steps to create a simulation application bundle configuration. Set the Working directory to the simulation application workspace.
7. To start a bundle process, choose RoboMaker Run, then choose Bundle, and then select a bundle configuration. A new terminal window will open at the bottom of the IDE. Use it to track bundle progress. You must build the application before it can be bundled.

Create a Simulation Configuration

To create a simulation configuration

1. Open your AWS RoboMaker development environment containing your robotic application.
2. In the AWS Cloud9 menu, choose RoboMaker Run, then choose Launch Simulation, and then select Add or Edit Configurations.


4. In the RoboMaker Configuration dialog, under Simulation Configuration, provide a name for the robot application bundle configuration.

5. Provide a Simulation job duration in seconds. For example, to run the simulation job for an hour, type in 3600.

6. In Failure behavior, select fail.

7. Select the IAM role you created for the simulation job.

8. Select an optional Amazon S3 bucket for simulation job output.

9. Under Robot application, provide a robot application name.

10. Specify the Robot app bundle path. This is the path to the file created by the bundle process for the robot application. For example, ./MySample/robot_ws/output.tar. Optionally, select .. to locate the file.

11. For Architecture, select X86_64

12. Select the Amazon S3 bucket to use for the robot application source. The bundle will be uploaded this location with a prefix created from the path to the bundle file. For example, if your application bundle is ./MySample/robot_ws/output.tar, the filename will be output.tar and the prefix will be ./MySample/robot_ws/.

13. Specify the Launch package name and Launch file. If you are using the example project from Creating a New Robotic Application (p. 30), the launch package name is robot_app and the launch file is rotate.launch.

14. Under Simulation application, type in a simulation application name.

15. Specify the Sim app bundle path. This is the path to the file created by the bundle process for the simulation application. For example, ./MySample/simulation_ws/output.tar. Optionally, select .. to locate the file.

16. Select the Amazon S3 bucket to use for the simulation application source. The bundle will be uploaded this location with a prefix created from the path to the bundle file.

17. Specify the Launch package name and Launch file. If you are using the example project from Creating a New Robotic Application (p. 30), the launch package name is simulation_app and the launch file is example.launch.

18. Select Save to save the configuration.

19. To start a simulation job, choose RoboMaker Run, then choose Launch Simulation, and then select a simulation configuration.

Create a Workflow Configuration

To create a workflow configuration

1. Open your AWS RoboMaker development environment containing your robotic application.

2. In the AWS Cloud9 menu, choose RoboMaker Run, then choose Workflow, and then select Add or Edit Configurations.


4. In the RoboMaker Configuration dialog, under Colcon Bundle Configuration, provide a name for the workflow configuration.

5. For Action 1, select the colcon build step for your robot application.

6. Select add action, then select the colcon build step for your simulation application as the action.

7. Select add action, then select the colcon bundle step for your robot application as the action.
8. Select add action, then select the colcon bundle step for your simulation application as the action.
9. Select add action, then select the simulation step for your applicatio as the action.
10. Choose save.
11. To run a workflow, choose RoboMaker Run, then choose Workflow, and then select a workflow configuration. Use the terminal window to track workflow progress.

Use Gazebo and other simulation tools to connect to the running simulation job and explore sensor data and other information. You can find the tools in the RoboMaker Simulation menu under Applications.

**Menu Commands Reference**

The following lists describe the default menu bar commands for AWS RoboMaker in the AWS Cloud9 development environment. If the menu bar isn’t visible, choose the thin bar along the top edge of the development environment to show it.

**Topics**
- AWS RoboMaker Run (p. 49)
- AWS RoboMaker Simulation (p. 49)
- AWS RoboMaker Resources (p. 50)

**AWS RoboMaker Run**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Build</td>
<td>Build selected application or add/edit build configurations.</td>
</tr>
<tr>
<td>Bundle</td>
<td>Bundle selected application or add/edit bundle configurations.</td>
</tr>
<tr>
<td>Launch simulation</td>
<td>Start selected simulation.</td>
</tr>
<tr>
<td>Workflow</td>
<td>Run selected workflow or add/edit workflow configurations.</td>
</tr>
<tr>
<td>Add or Edit Configurations</td>
<td>Add or edit build, bundle, simulation and workflow configurations.</td>
</tr>
</tbody>
</table>

**AWS RoboMaker Simulation**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connect</td>
<td>Connect to a running simulation job.</td>
</tr>
</tbody>
</table>
### Working with the development environment

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stop</td>
<td>Stop the connected running simulation job.</td>
</tr>
<tr>
<td>Restart</td>
<td>Restart the connected simulation job. Only available if the current workflow contains a simulation step.</td>
</tr>
<tr>
<td>Restart With New Bundles</td>
<td>Restart simulation with new bundles.</td>
</tr>
<tr>
<td>Restart With Workflow</td>
<td>Restart simulation with the selected workflow.</td>
</tr>
<tr>
<td>Applications, Gazebo</td>
<td>Launches Gazebo, a tool for simulating robots in complex environments.</td>
</tr>
<tr>
<td>Applications, rviz</td>
<td>Launches rviz (ROS Visualizer), a tool for visualizing sensor data and state information from ROS in 3D.</td>
</tr>
<tr>
<td>Applications, rqt</td>
<td>Launches rqt, a Qt-based framework and plugins for ROS GUI development.</td>
</tr>
<tr>
<td>Applications, Terminal</td>
<td>Launches a terminal connected to the simulation host.</td>
</tr>
<tr>
<td>View Simulation Job Details</td>
<td>Opens a new page with simulation job details in the RoboMaker console.</td>
</tr>
<tr>
<td>Switch simulation</td>
<td>Connect to a different running simulation.</td>
</tr>
<tr>
<td>Disconnect</td>
<td>Disconnect from the simulation.</td>
</tr>
</tbody>
</table>

### AWS RoboMaker Resources

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Download Samples</td>
<td>Download selected sample application.</td>
</tr>
<tr>
<td>View Extensions</td>
<td>Download selected cloud extension.</td>
</tr>
</tbody>
</table>
Deleting an Environment

To prevent any ongoing charges to your AWS account related to an AWS Cloud9 development environment that you’re no longer using, you should delete the environment.

To delete an environment

Follow these steps:

2. In the left navigation pane, choose Development, and then choose Development environments.
3. Choose the environment you want to delete, then choose Edit.
4. In the AWS Cloud9 Environment details page, choose Delete. Type in Delete and then choose Delete to permanently delete the environment.
Working with Robot Applications

An AWS RoboMaker robot application includes one or more Amazon S3 locations with robot application bundles. The Amazon S3 resources include the ROS distribution used by the robot application.

Your robot application can be paired with a simulation application. This is called a simulation job. You can also deploy your application to physical robots.

Topics
- Creating a Robot Application (p. 52)
- Creating a Robot Application Version (p. 53)
- Viewing a Robot Application (p. 53)
- Updating a Robot Application (p. 54)
- Deleting a Robot Application (p. 54)
- Deleting a Robot Application Version (p. 55)

Creating a Robot Application

Create a robot application to use in a simulation job. When it is ready to deploy to robot hardware, you can add it to a fleet and then deploy to the fleet.

To create a robot application

Follow the steps under one of the following tabs.

Note
Amazon S3 objects must be located in the same region as AWS RoboMaker.

Using the console

2. In the left pane, choose Development, and then choose Robot applications.
3. Select Create robot application.
4. In the Create robot application page, type a Name for the robot application. Choose a name that helps you identify the robot.
5. Select the ROS distribution used by your robot application. For more information about the Robot Operating System (ROS), see www.ros.org.
6. Provide the Amazon S3 path to your bundled robot application file. If this robot application is used only in simulations, specify a bundle built for the X86_64 architecture. If you use this robot application in a fleet deployment, specify one or more bundles that represent the architectures of the robots in your fleet.

Optionally, choose Create new S3 folder to go to the Amazon Simple Storage Service AWS Management Console to create and manage buckets.

7. Optionally, under Tags, specify one or more tags for the simulation application. Tags are words or phrases that act as metadata for identifying and organizing your AWS resources. Each tag consists of a key and a value. You can manage tags for your simulation application on the Simulation application details page.

For more about tagging, see Tagging Your AWS RoboMaker Resources (p. 125).
8. Choose **Create**.

**Using the AWS CLI**

**Example**

Here’s an example AWS CLI command that performs the equivalent of the console-based create robot application on the other tab.

```bash
$ aws robomaker create-robot-application --application my-robot-application --robot-software-suite name=ROS,version=Melodic --sources architecture=X86_64,s3Bucket=my-bucket,s3Key=my-folder/cloud-watch-robot.tar
```

**Creating a Robot Application Version**

When you create a robot application version, you create a snapshot of the $LATEST version. AWS RoboMaker remembers the Amazon S3 path and ETag of the file for the version. If you change the file at the Amazon S3 path of that version, AWS RoboMaker will not be able to use that version. AWS RoboMaker does not keep a copy of the version.

You can use any version of a robot application when you create a simulation job. For deployments, you must use a numbered version. For more information about application versioning, see Application Versioning (p. 28).

**To create a robot application version**

Follow the steps under one of these tabs.

**Using the console**

2. In the left navigation pane, choose **Development**, and then choose **Robot applications**.
3. Choose the robot application **name**.
4. In the **Robot applications details** page, choose **Create new version**, and then choose **Create**.

**Using the AWS CLI**

**Example**

Here’s an example AWS CLI command that performs the equivalent of the console-based steps.

```bash
$ aws robomaker create-robot-application-version --name my-robot-application-arn
```

**Viewing a Robot Application**

View the details of your robot application including Amazon S3 location, version, and supported architectures.

**To see the details of a robot application**
Follow the steps under one of these tabs.

Using the console

2. In the left navigation pane, choose Development, then choose Robot applications.
3. Choose the Name of a robot application.

Using the AWS CLI

Example

Here's an example AWS CLI command that performs the equivalent of the console-based steps.

```bash
$ aws robomaker describe-robot-application --application my-robot-application-arn
```

Updating a Robot Application

Update a robot application.

To update a robot application

Follow the steps under one of these tabs.

Using the console

2. In the left navigation pane, choose Development, then choose Robot applications.
3. Check the box next to the robot application you want to update.
4. Choose Actions, then choose Update.
5. You can add or remove sources, but you must have at least one source robot application file.
6. Choose Update to update the robot application.

Using the AWS CLI

Example

Here's an example AWS CLI command that performs the equivalent of the console-based steps.

```bash
$ aws robomaker update-robot-application --application my-robot-application-arn --robot-software-suite name=ROS,version=Melodic --sources architecture=X86_64,s3Bucket=my-bucket,s3Key=my-folder/cloud-watch-robot.tar
```

Deleting a Robot Application

When you no longer need a AWS RoboMaker robot application and all of its versions, delete it.
To delete a robot application

Follow the steps under one of these tabs.

Using the console

2. In the left navigation pane, choose Development, then choose Robot applications.
3. Choose the Name of a robot application to see details including the time it was created and last updated.
4. In the robot application detail page, choose Delete and then choose Delete to confirm.

Using the AWS CLI

Example

Here's an example AWS CLI command that performs the equivalent of the console-based steps.

```
$ aws robomaker delete-robot-application --application my-robot-application-arn
```

Deleting a Robot Application Version

Delete robot application versions you no longer need.

To delete a robot application version

Follow the steps under one of these tabs.

Using the console

2. In the left navigation pane, choose Development, then choose Robot applications.
3. Choose the Name of the robot application to see its versions.
4. In the robot detail page, choose the Version to see version details.
5. In the robot application version details page, choose Delete, and then choose Delete to confirm.

Using the AWS CLI

Example

Here's an example AWS CLI command that performs the equivalent of the console-based steps.

```
$ aws robomaker delete-robot-application --version --application my-robot-application-arn --version 1.5
```
Working with Simulation Applications

An AWS RoboMaker simulation application includes information about its dependencies. It includes the Amazon Simple Storage Service (Amazon S3) location of a simulation application bundle for the X86_64 architecture. It also includes the name and version of the Robot Operating System (ROS) distribution and the rendering engine used.

Join the simulation with an AWS RoboMaker robot application in a simulation job to interact with your robot. Interact with tools like Gazebo and develop simulation and test data.

Topics

- Creating a Simulation Application (p. 56)
- Creating a Simulation Application Version (p. 57)
- Viewing a Simulation Application (p. 58)
- Updating a Simulation Application (p. 58)
- Deleting a Simulation Application (p. 59)
- Deleting a Simulation Application Version (p. 60)

Creating a Simulation Application

Create a simulation application to use in simulation jobs.

To create a simulation application

Follow the steps under one of the following tabs.

Note
Amazon S3 objects must be located in the same Region as AWS RoboMaker.

Using the console

2. In the left navigation pane, choose Development, and then choose simulation applications.
3. Type a name for the application. Choose a name that helps you identify the simulation. For example, Outdoor v2.
4. Select the ROS distribution used by your robot application. For more information about ROS, see www.ros.org.
5. Select the Simulation software suite used by your simulation application. For more information about ROS, see www.ros.org.
6. Select the Simulation rendering engine used by your simulation application.
7. Provide the Amazon S3 path to your bundled simulation application file built for the X86_64 architecture.

(Optional) select Create new S3 folder to go to the Amazon Simple Storage Service console. There, you can create and manage buckets.
8. (Optional) Under **Tags**, specify one or more tags for the simulation application. Tags are words or phrases that act as metadata to identify your AWS resources. Each tag consists of a key and a value. You can manage tags for your simulation on the **Simulation application details** page.

For more about tagging, see [Tagging Your AWS RoboMaker Resources](#) (p. 125).

9. Choose **Create**.

---

**Creating a Simulation Application Version**

When you create a simulation application version, you create a snapshot of the **LATEST** version. AWS RoboMaker remembers the Amazon S3 path and ETag of the file for the version. If you change the file at the Amazon S3 path of that version, AWS RoboMaker will not be able to use that version. AWS RoboMaker does not keep a copy of the version.

You can use any version of a robot simulation application when you create a simulation job. For more information about application versioning, see [Application Versioning](#) (p. 28).

**To create a simulation application version**

Follow the steps under one of the following tabs.

**Using the console**

2. In the left navigation pane, choose **Development**, and then choose **Simulation applications**.
3. Select the simulation application **name**.
4. In the **Simulation applications details** page, select **Create new version**, and then select **Create**.

**Using the AWS CLI**

**Example**

Here's an example AWS CLI command that performs the equivalent of the console-based create robot application on the other tab.

```bash
$ aws robomaker create-simulation-application-version --name my-simulation-application-arn
```
Viewing a Simulation Application

View the details of a simulation application.

**To see the details of a simulation application**

Follow the steps under one of the following tabs.

**Using the console**

2. In the left navigation pane, choose Development, then choose Simulation applications.
3. Select the Name of a simulation application to see details including the time it was created and last updated.

**Using the AWS CLI**

Use the following commands to describe a simulation application.

- **API/SDK**: DescribeSimulationApplication
- **AWS CLI**: describe-simulation-application

**Example**

Here's an example AWS CLI command that performs the equivalent of the console-based describe simulation application on the other tab.

```bash
# aws robomaker describe-simulation-application \
--job my-simulation-job-arn
```

Updating a Simulation Application

Update a simulation application.

**To update a simulation application**

Follow the steps under one of the following tabs.

**Using the console**

2. In the left navigation pane, choose Development, then choose Simulation applications.
3. Check the box next to the simulation application you want to update.
4. Select Actions, then select Update.
5. You can add or remove sources, but you must have at least one source simulation application file.
6. Select Update to update the simulation application.
Deleting a Simulation Application

When you no longer need a AWS RoboMaker simulation application and all of its versions, delete it.

To delete a simulation application

Follow the steps under one of the following tabs.

Using the console

2. In the left navigation pane, choose Development, then choose Simulation applications.
3. Select the Name of a simulation application. This shows details such as the time it was created and last updated.
4. In the simulation application detail page, choose Delete and then choose Delete to delete to confirm.

Using the AWS CLI

You can use the following commands to delete a simulation application:

• API/SDK: DeleteSimulationApplication
• AWS CLI: delete-simulation-application

Example

Here's an example AWS CLI command that performs the equivalent of the console-based delete simulation application on the other tab.

```bash
# aws robomaker delete-simulation-application --application my-robot-application-arn
```
Deleting a Simulation Application Version

You can delete simulation application versions you no longer need.

To delete a simulation application version
Follow the steps under one of the following tabs.

Using the console

2. In the left navigation pane, choose Development, then choose Simulation applications.
3. Select the Name of the simulation application to see its versions.
4. In the simulation detail page, choose Version to see details.
5. In the details page, choose Delete, and then choose Delete to confirm.

Using the AWS CLI

You can use the following commands to delete a simulation application version.

- **API/SDK:** `DeleteSimulationApplicationVersion`
- **AWS CLI:** `delete-simulation-application-version`

Example

Here's an example AWS CLI command that performs the equivalent of the console-based delete simulation application on the other tab.

```
$ aws robomaker delete-simulation-application-version \
   --application my-robot-application-arn \
   --version 1.5
```
Running Simulation Jobs

An AWS RoboMaker simulation job is a pairing of a robot application and a simulation application running in the cloud. While the simulation job is running, you can interact with it using simulation tools like Gazebo, rviz, rqt and a terminal to visualize sensor data or control components of the robot.

Topics
- Configuring an AWS RoboMaker Simulation Job to Access Resources in an Amazon VPC (p. 61)
- Connecting to a Simulation Job (p. 62)
- Accessing Simulation Job Data (p. 65)
- Root Access and System Capabilities (p. 66)
- Environment Variables Created by AWS RoboMaker (p. 67)
- Managing Tags in a Simulation Job (p. 67)
- Using ROS Bags for Play Back (p. 69)
- Managing Simulation Jobs (p. 74)
- Managing Simulation Job Batches (p. 80)
- Simulation Tools (p. 83)

Configuring an AWS RoboMaker Simulation Job to Access Resources in an Amazon VPC

When you create resources in the Amazon Virtual Private Cloud (Amazon VPC), they cannot be read through the public internet. Example resources could be Amazon Redshift data warehouses or Amazon ElastiCache clusters. They could also be your services on an Amazon Elastic Compute Cloud instance. By default, resources in an Amazon VPC are not accessible to an AWS RoboMaker simulation job.

AWS RoboMaker runs your simulation job in an Amazon VPC by default. However, to allow your job to access resources in your Amazon VPC, you must provide VPC-specific data that includes Amazon VPC subnet IDs and security group IDs. AWS RoboMaker uses this data to set up elastic network interfaces (ENIs). ENIs help your job to connect securely to other resources in your private Amazon VPC.

AWS RoboMaker does not connect to resources within dedicated tenancy VPCs. For more information, see Dedicated VPCs.

Configuring an AWS RoboMaker Simulation Job for Amazon VPC Access

Add Amazon VPC data to your AWS RoboMaker simulation job by using the VpcConfig parameter at the time you create a job (see CreateSimulationJob (p. 178)). Here is an AWS CLI example.

- The create-simulation-job CLI command specifies the --vpc-config parameter. Use it to provide VPC data at the time you create a simulation job. In this example, a public IP is assigned.

```
$ aws robomaker create-simulation-job \
--output-location s3Bucket=my-bucket,s3Prefix=my-output-folder \
--max-job-duration-in-seconds 3600 \
```
When a simulation job is configured to run in a VPC, it incurs an ENI penalty. Address resolution may be delayed when you try to connect to network resources.

### Internet Access for Simulation Jobs

AWS RoboMaker uses the VPC data you provide to set up ENIs. ENIs allow your job to access VPC resources. Each ENI is assigned a private IP address from the range in the subnets you specify. The ENI is not assigned any public IP addresses by default.

If your job requires internet access (perhaps to find AWS services that do not have VPC endpoints), you can set up a NAT inside your VPC. You can use the Amazon VPC NAT gateway. You can request RoboMaker to assign a public IP. For more information, see NAT Gateways in the Amazon VPC User Guide. You cannot use an internet gateway attached to your VPC. That requires the ENI to have public IP addresses.

To configure internet access when using public Subnets, set `assignPublicIp=true` to assign a public IP to your ENI.

### Connecting to a Simulation Job

When you want to interact with the applications in your simulation job, connect with port forwarding. When you configure port forwarding, traffic is forwarded from the simulation job port to the application port. Port forwarding makes it easy to connect with tools such as ROS Bridge and other tools. This can be useful when you want to debug or run custom tools to interact with your applications.

**Topics**
- Before You Enable Port Forwarding (p. 62)
- Enabling Port Forwarding (p. 63)
- Port Forwarding Examples (p. 63)

### Before You Enable Port Forwarding

Who requires access to your instance? A single host or a specific network that you trust such as your local computer's public IPv4 address. The security group editor in the Amazon EC2 console can automatically detect the public IPv4 address of your local computer for you. You could also search the internet for “what is my IP address” in an internet browser, or use the following service: Check IP. If you are connecting through an ISP or from behind your firewall without a static IP address, you must find out the range of IP addresses used by clients.

**Warning**
You are responsible for configuring a secure remote connection to the simulation job. We recommend that you implement a strong authentication method and encryption in transit for the ports you are opening.
For more information about security groups, see Security Groups for your VPC.

Enabling Port Forwarding

To enable port forwarding:

1. Determine which ports you need for your robot application and simulation application. A port on your application is called an **application port**.

   For example, you might want to use ROS Bridge on port 8085 in your simulation application and port 8080 for HTTP in your robot application.

2. Identify the ports on the simulation job instance you want to use as remote connection points. A port on the simulation job is called a **job port**.

   For example, you can use port 8085 for ROS Bridge and 80 for HTTP. The job port and the application ports can be different.

3. Determine which port mappings you want to enable on a public IP. A public IP address is an IPv4 address that is reachable from the internet. You can use public address for communication between your application and the internet.

   You can connect to your simulation job without enabling a public address using solutions such as Linux bastion or AWS VPN.

4. Create a simulation job (p. 74) and provide the application port mappings. For example, you can provide a simulation application mapping for ROS Bridge such as job port 8085 to application port 8085. You can also provide robot application mapping for HTTP such as job port 8080 to application port 80.

5. Configure one or more VPC security groups to enable traffic on the simulation job ports. Create rules to configure inbound traffic or outbound traffic.

   For example, if you are using HTTP on port 80, you can create rules to allow inbound and outbound traffic on port 80. You can restrict access to a single IP address, range of addresses, or use other criteria.

   For more information about working with security groups, see Working with Security Groups

Port Forwarding Examples

You can connect to your simulation job remotely using different tools. This section shows how to connect using ROS Bridge and HTTP.

Topics

- Port Forwarding with ROS Bridge (p. 63)
- Port Forwarding with HTTP Server (p. 64)

Port Forwarding with ROS Bridge

You can access ROS functionality from non-ROS programs using ROS Bridge. ROS Bridge provides a JSON API. Tools and applications that support JSON APIs can interface with ROS Bridge and access ROS functionality. For more information about ROS Bridge, see rosbridge_suite.

To connect to your application using ROS Bridge

1. Add a dependency on the ROS Bridge package in your package.xml file:
2. Update your launch file to enable ROS Bridge. It uses port 8080 by default.

```xml
<launch>
  <arg name="rosbridge_port" value="$(optenv ROSBRIDGE_PORT 8080)"/>
  <include file="$(find rosbridge_server)/launch/rosbridge_websocket.launch"
    <arg name="port" value="$(arg rosbridge_port)"/>
  </include>
</launch>
```

3. When you create your simulation job, enable remote connectivity and open port 8080 for the robot application or simulation application. When your simulation job is running, you can connect.

## Port Forwarding with HTTP Server

Copy the following code into a new Python file as a ROS Python node in your nodes directory. The code creates a node named `webserver` that hosts a web server.

For more information about ROS nodes, see [Understanding ROS Nodes](#).

1. Add a dependency on `rospy`, a Python client library for ROS in your `package.xml` file:

```xml
<package>
  ...
  <exec_depend>rospy</exec_depend>
</package>
```

2. Update your launch file to enable ROS Bridge. It uses port 8080 by default.

```xml
<launch>
  <arg name="server_port" value="$(optenv SERVER_PORT 8080)"/>
  <node pkg="python_launcher" type="run_webserver.sh" name="webserver" output="screen"
    required="true" args="$(arg server_port)" />
</launch>
```

3. Copy the following code into a new shell script named `run_webserver.sh`.

```bash
#!/usr/bin/env python
set -ex
python3 -m webserver.node $1
```

4. Copy the following code into a new Python file. The code creates a node named `webserver` that launches a simple web server.

```python
#!/usr/bin/env python
import rospy
import http.server
import socketserver
import sys

def start_server(port):
    # Start a webserver
    httpd = socketserver.TCPServer(('', int(port)),
    http.server.SimpleHTTPRequestHandler)
```

When you create your simulation job, enable port forwarding and open port 8080 for the robot application or simulation application. When your simulation job is running, you can connect using HTTP.

### Accessing Simulation Job Data

AWS RoboMaker can capture the following information from a running simulation job:

- **Information written to standard output and standard error streams.** This information is collected in Amazon CloudWatch Logs.
- **CloudWatch metric RealTimeFactor.** It is the ratio of the amount of time that was simulated versus wall clock time. If it takes an hour to simulate 30 minutes, the factor is .5. More complex simulations have a lower real time factor.
- **Gazebo log data including the state of models, links and joints.** Gazebo log data is written to the gazebo-logs folder if an Amazon Simple Storage Service bucket was specified when the simulation job was created.
- **ROS bag files containing timestamped ROS messages.** ROS bag files are written to the ros-bag folder if an Amazon Simple Storage Service bucket was specified when the simulation job was created.

Standard output and standard error information is written to Amazon CloudWatch while the simulation job is running. Gazebo logs and ROS bag files are available soon after the simulation job completes.

#### To access logs, metrics and optional output Amazon S3 bucket

1. In the AWS RoboMaker console, choose **Simulation jobs** on the left and then select the simulation job.
2. In the **Simulation details** page, select the **Configuration** tab.
3. To see ROS bags, ROS logs and Gazebo logs, select the Amazon S3 location under **Simulation job output destination** to view the Amazon S3 bucket, then select the folder beginning with `sim`, and then select the folder. If there are more than one, select by date and time.

   The folder includes gazebo-logs, ros-bags, and ros-logs.

4. To see standard error, standard output, and other information in CloudWatch Logs, select **Logs**, and then choose a simulation application or robot application log to view.
5. To see CloudWatch metrics, select **Metrics**, and then select a metric. For example, select **RealTimeFactor**.

Access CloudWatch Logs, metrics, and simulation output Amazon S3 bucket from the **Simulation job details** page in the AWS RoboMaker console.
Root Access and System Capabilities

AWS RoboMaker provides limited root (sudo) access to applications running in a simulation job. The list below contains significant (but not all) syscalls that are blocked.

- acct
- add_key
- bpf
- clock_adjtime
- clock_settime
- clone
- create_module
- delete_module
- finit_module
- get_kernel_syms
- get_mempolicy
- init_module
- ioperm
- iopl
- kcmp
- kexec_file_load
- kexec_load
- keyctl
- lookup_dcookie
- mbind
- mount
- move_pages
- name_to_handle_at
- nfsservctl
- open_by_handle_at
- perf_event_open
- personality
- pivot_root
- process_vm_readv
- process_vm_writev
- ptrace
- query_module
- quotactl
- reboot
- request_key
- set_mempolicy
- setns
- settimeofday
- stime
- swapon
- swapoff
Environment Variables Created by AWS RoboMaker

AWS RoboMaker defines these simulation job environment variables.

- AWS_ROBOMAKER_SIMULATION_JOB_ID
- AWS_ROBOMAKER_SIMULATION_JOB_ARN
- AWS_ROBOMAKER_SIMULATION_RUN_ID

You can get these variables from your application or from the command line. For example, to get the current simulation job Amazon Resource Name (ARN) in Python, use `os.environ.get("AWS_ROBOMAKER_SIMULATION_JOB_ARN")`.

If you specified an Amazon Simple Storage Service output bucket for the simulation job, you can use the environment variables to find the output path. AWS RoboMaker writes output to `s3://bucket-name/AWS_ROBOMAKER_SIMULATION_JOB_ID/AWS_ROBOMAKER_SIMULATION_RUN_ID`. Use this to manage objects in Amazon S3 from code or the command line.

Managing Tags in a Simulation Job

To help you manage your simulation jobs, you can optionally assign your own metadata. For example, you can categorize simulation jobs by the type of terrain simulated, test result, or robot hardware configuration. This helps you organize and track simulation job results.

This section includes information about using ROS command-line tools and code. For more information about managing tags with the AWS RoboMaker API, see TagResource. For more information about tagging, see Tagging Your AWS RoboMaker Resources (p. 125).

You must have permissions to tag, untag, and list tags for a simulation job. For more information, see Permissions Required to use Tags from a ROS Application or ROS Command Line (p. 104).

**Note**
This topic applies to ROS Kinetic and ROS Melodic. For more information about ROS 2 Dashing, see ROS 2 Dashing (Beta) (p. 3).

Using tags using ROS command-line tools

You can use the ROS command-line tool `rosservice` to add, list, and remove tags for the simulation job. The following example adds the tags "status" and "name", lists the tags, and then removes both of the tags.
The following example adds the tags "status" and "pass" to the current simulation job:

**Note**
You must have permissions to call `TagResource`, `UntagResource`, and `ListTagsForResource`. For more information about permissions, see Authentication and Access Control for AWS RoboMaker (p. 99).

```bash
roservice call /robomaker/job/add_tags "[{key: status, value: pass}, {key: name, value: my_test}]"
```

success: True
message: ''

The following example lists all tags for the simulation job:

```bash
roservice call /robomaker/job/list_tags
```

success: True
message: ''
tags:
- key: "status"
  value: "pass"
- key: "name"
  value: "my_test"

The following example removes the tags "status" and "pass" from the current simulation job:

```bash
roservice call /robomaker/job/remove_tags "[status, name]"
```

success: True
message: ''

**Using tags from code in a simulation application**

The following example provides a Python function you can use to manage tags in your simulation application:

```python
#!/usr/bin/env python
# before using, add a dependency on the AWS RoboMaker package
# in package.xml to access the service (.srv) types:
#<depend>aws_robomaker_simulation_ros_pkgs</depender>
# See the repo at https://github.com/aws-robotics/aws-robomaker-simulation-ros-pkgs.
import rospy
from robomaker_simulation_msgs.msg import Tag
from robomaker_simulation_msgs.srv import AddTags

# add a list of Tag(key,value) to the simulation job
def add_tags(tags):
    # ensure the service is ready
    rospy.wait_for_service('/robomaker/job/add_tags', timeout=30)
    requestAddTags = rospy.ServiceProxy('/robomaker/job/add_tags', AddTags)
    response = requestAddTags(tags)
    if response.success:
        rospy.loginfo("Successfully added tags: %s", tags)
    else:
        rospy.logerr("Add tags request failed for tags (%s): %s", tags, response.message)
```

68
For more information about

**Using ROS Bags for Play Back**

**ROS bags** are files that contain timestamped, serialized message data from ROS topics. AWS RoboMaker can play back messages in ROS bags to ROS applications running in a simulation job. AWS RoboMaker uses `rosbag` to play back messages. Messages are played back based on their original timestamp and include the original payload. The messages from the ROS bags replace inputs and other observations from the real world and virtual simulation.

ROS bags are recorded using `rosbag record`. ROS bags can be created from ROS applications designed for simulation. They can also be created by robots operating in the real world. ROS bags are used to test new robotic applications, troubleshoot existing applications, and develop new functionality.

ROS bags specified in data sources are copied to the `/opt/robomaker/datasources/` directory in the simulation environment.

**Topics**
- Example Launch File Configurations (p. 69)
- Avoiding a Failed Simulation Job when Play Back Ends (p. 72)
- Using Tags for Custom Status and Property Values (p. 72)
- Canceling a Simulation Job Early (p. 72)

**Example Launch File Configurations**

A simulation job using ROS bag playback requires at least one ROS bag data source. The simulation application must also have a launch file node configured to play back the ROS bag data source. This section contains example launch configurations.

The examples below use a node that plays ROS bag files. The BAG files are specified in arguments. The path to the ROS bag file is prefixed with the mount path for data sources. For example, in the following:

```
args="/opt/robomaker/datasources/mybaggroup/myS3prefix/log_0.bag"/
```

Where:
- `myS3prefix/log_0.bag` is the full Amazon S3 key path to the bag file
- `mybaggroup` is the name of the bag group
- `/opt/robomaker/datasources/` is the path where the log file is mounted

**Play a Single ROS bag File**

In the following example, the messages in ROS bag file `log_0.bag` are played back ordered by synchronized time. The bag file is part of the ROS bag group `mybaggroup`. The ROS bag file itself is located in `myS3prefix`, not in `mybaggroup`.

```
<launch>
  <!-- ROS bag files are copied to /opt/robomaker/datasources/ -->
  <!-- ROS bag files are copied from myS3prefix/ -->
  <node pkg="rosbag" type="play" name="rosbag_play" output="screen" required="true"
    args="/opt/robomaker/datasources/mybaggroup/myS3prefix/log_0.bag"/>
</launch>
```

69
Play Multiple ROS Bag Files

In the following example, the messages in the ROS bag files log_0.bag and log_1.bag are played back. The ROS bags are in the same ROS bag group mybaggroup. Messages from both bag files are ordered by synchronized time.

```xml
<launch>
  <node pkg="rosbag" type="play" name="rosbag_play" output="screen" required="true"
    args="/opt/robomaker/datasources/mybaggroup/myS3prefix/log_0.bag
    /opt/robomaker/datasources/mybaggroup/myS3prefix/log_1.bag"/>
</launch>
```

The time between messages are simulated. For example, if there are two ROS bags with a 30-minute gap between them, the simulation ticks for 30 (simulation) minutes.

Use `--skip-empty=SEC` to skip regions where there are no messages for SEC seconds. In the following example, any gaps more than 60 seconds are skipped.

```xml
<launch>
  <node pkg="rosbag" type="play" name="rosbag_play" output="screen" required="true"
    args="--skip-empty=60 /opt/robomaker/datasources/mybaggroup/myS3prefix/log_0.bag
    /opt/robomaker/datasources/mybaggroup/myS3prefix/log_1.bag"/>
</launch>
```

Play ROS Bag Files from Different Bag Groups

In the following example, the messages in the ROS bag files log_0.bag and log_1.bag are played back ordered by synchronized time. The files are from different ROS bag groups mybaggroup_1 and mybaggroup_2.

```xml
<launch>
  <node pkg="rosbag" type="play" name="rosbag_play" output="screen" required="true"
    args="/opt/robomaker/datasources/mybaggroup_1/myS3prefix/log_0.bag
    /opt/robomaker/datasources/mybaggroup_2/myS3prefix/log_0.bag"/>
</launch>
```

Use a Different Starting Offset and Duration

You can control when playback begins by specifying a time offset in seconds. In the following example, the messages in the ROS bag file log_0.bag begin playback at 10 seconds. It ends when the end of the ROS bag is reached.

```xml
<launch>
  <node pkg="rosbag" type="play" name="rosbag_play" output="screen" required="true"
    args="--start 10 /opt/robomaker/datasources/mybaggroup_1/myS3prefix/log_0.bag"/>
</launch>
```

You can also play back a range a subset of messages by specifying `start` and `duration` times. In the following example, the messages in the ROS bag file log_0.bag begin playback at 10 seconds. It ends after 100 seconds.

```xml
<launch>
  <node pkg="rosbag" type="play" name="rosbag_play" output="screen" required="true"
    args="--start 10 --duration 100 /opt/robomaker/datasources/mybaggroup_1/myS3prefix/log_0.bag"/>
</launch>
```
To play a subset of the messages starting at the beginning of the ROS bag, specify only a duration.

```xml
<launch>
  <node pkg="rosbag" type="play" name="rosbag_play" output="screen" required="true"
    args="--duration 100 /opt/robomaker/datasources/mybaggroup_1/myS3prefix/log_0.bag"/>
</launch>
```

Pausing and Playing Messages Interactively

ROS bag messages begin playback automatically. You can manually start and stop message playback. Use `--pause` to start ROS bag message playback as paused. You can then interact with playback using tools such as rviz and rqt. This is helpful when debugging your robot application.

For more information, see rosservice command line tool.

In the following launch file, playback begins in a paused state.

```xml
<launch>
  <node pkg="rosbag" type="play" name="rosbag_play" output="screen" required="true"
    args="--pause /opt/robomaker/datasources/mybaggroup_1/myS3prefix/log_0.bag"/>
</launch>
```

From the terminal, use the following commands to resume message playback and pause message playback.

```bash
robomaker@9cc6d11dfa46:~$ rosservice call /rosbag_play/pause_playback '{data: True}'
success: True
message: "Playback is now paused"
robomaker@9cc6d11dfa46:~$ rosservice call /rosbag_play/pause_playback '{data: False}'
success: True
message: "Playback is now resumed"
```

Using Environment Variables to Configure Play Back

You can use environment variables as arguments in the launch.xml file. Environment variables can be defined when creating a simulation job. Use them to make it easier to specify the location of ROS bags, playback arguments like start, and other details.

In the following launch file, two arguments `bags` and `args` are created. Their values are captured from the environment variables BAGS and ARGS defined when you create the simulation job.

```xml
<launch>
  <arg name="bags" default="$(optenv BAGS)" doc="space separated list of bag files"/>
  <arg name="args" default="$(optenv ARGS)" doc="rosbag play args"/>
  <node pkg="rosbag" type="play" name="rosbag_play" output="screen" required="true"
    args="$(arg bags) $(arg args)"/>
</launch>
```

When you create the simulation job, create the BAGS and ARGS environment variables. For example, if you create environment variables with the values below, playback uses two ROS bag files. Playback begins at 10 seconds and lasts for 100 seconds.
Avoiding a Failed Simulation Job when Play Back Ends

When the playback node exits and it is required, the simulation application closes. The simulation job status is set to Failed and the failure reason is Simulation application exited abnormally (segfault, etc.). Avoid this by using the keep-alive option.

In the following example, the simulation job runs for the simulation job duration time specified. If playback has not completed at the end of the simulation job, it is stopped. In both cases, the simulation job status is set to Completed.

```xml
<launch>
  <!-- Other ROS launch nodes... -->
  <node pkg="rosbag" type="play" name="rosbag_play" output="screen" required="true" args="--keep-alive /opt/robomaker/datasources/mybaggroup_1/myS3prefix/log_0.bag"/>
</launch>
```

You can also cancel a simulation job early. For more information, see Canceling a Simulation Job Early (p. 72).

Using Tags for Custom Status and Property Values

To help you manage your simulation jobs, you can optionally assign your own metadata. For more information, see Managing Tags in a Simulation Job (p. 67).

Canceling a Simulation Job Early

You can choose to cancel your simulation job early with custom application logic or manually from the terminal. When you cancel, the simulation job status is set to Cancelled. This can be helpful when reviewing simulation jobs. If a simulation job is Cancelled, it was done intentionally. If a simulation job has a job status of Failed, it exited unexpectedly.

A simulation job can be cancelled manually from the terminal or programmatically within your simulation application.

**Note**

You must have permissions to call CancelSimulationJob to cancel a simulation job. For more information about permissions, see Authentication and Access Control for AWS RoboMaker (p. 99).

Canceling Manually from the Terminal

The following example shows how to use rosservice to manually cancel a simulation job.

**Note**

The connection is closed when the simulation job terminates.
Canceling from Python

To cancel using Python in your simulation application:

1. To access the service types (.srv), add a dependency on the AWS RoboMaker package in your package.xml.

   ```xml
   <package>
   ...
   <depend>aws_robomaker_simulation_ros_pkgs</depend>
   </package>
   ```

2. Copy the following code into a new Python file as a ROS Python node. The code uses service proxies to add a tag and cancel the simulation job based on elapsed time. Your cancel conditions might depend on sensor data, robot conditions, or other information.

   For more information about ROS nodes, see Understanding ROS Nodes.

   ```python
   #!/usr/bin/env python

   import rospy
   from rosgraph_msgs.msg import Clock
   from robomaker_simulation_msgs.msg import Tag
   from robomaker_simulation_msgs.srv import Cancel, AddTags
   
   is_cancelled = False

   # cancels the simulation job
   def cancel_job():
       # proxy for simulation job cancel
       requestCancel = rospy.ServiceProxy('/robomaker/job/cancel', Cancel)
       response = requestCancel()
       if response.success:
           global is_cancelled
           is_cancelled = True
           rospy.loginfo("Successfully requested cancel job")
       else:
           rospy.logerr("Cancel request failed: %s", response.message)

   # adds the key=value tag to the simulation job
   def add_tags(tags):
       requestAddTags = rospy.ServiceProxy('/robomaker/job/add_tags', AddTags)
       response = requestAddTags(tags)
       if response.success:
           rospy.loginfo("Successfully added tags: %s", tags)
       else:
           rospy.logerr("Add tags request failed for tags (%s): %s", tags, response.message)

   # checks for simulation job cancel conditions
   def check_complete(msg):
       # when clock time is 30 seconds, add a tag indicating results
       # and cancel the simulation job
       global is_cancelled
       if msg.clock.secs > 30.0 and not is_cancelled:
           add_tags(key="result", value="pass")
           is_cancelled = cancel_job()

   # timer callback that cancels the job
   def cancel_timeout(timer):
       global is_cancelled
       if not is_cancelled:
           rospy.loginfo("Timed out, canceling job")
   ```
Managing Simulation Jobs

In this section, learn how to create and manage simulation jobs.

**Topics**

- Creating a Simulation Job (p. 74)
- Viewing a Simulation Job (p. 78)
- Cancelling a Simulation Job (p. 79)
- Cloning a Simulation Job (p. 79)
- Restarting a Simulation Job (p. 80)

**Creating a Simulation Job**

Create a simulation job when you want to run your robot application in a virtual world using Gazebo or using previously recorded ROS messages stored in ROS bags. You select the software suite name when you specify the simulation application. If you choose Gazebo, your robot application will interact with the objects, terrain, and other physical aspects modelled in the simulation application. If you choose RosbagPlay, your robot application will consume ROS messages published from the ROS bags you provide as data sources.

For more information about configuring your ROS launch file to play back ROS bag messages, see Using ROS Bags for Play Back (p. 69).

**Note**

After 90 days, simulation jobs expire and will be deleted. They will no longer be accessible.

**Topics**

- Create a Simulation Job using Gazebo (p. 74)
- Create a Simulation Job using ROS bags (p. 77)

**Create a Simulation Job using Gazebo**

You will create a simulation job using Gazebo when you want to run your robot application in the simulation you created. Once running, you can interact with the simulation. For example, you can use rviz to see images from visual sensors on your robot.
To create a simulation job using Gazebo

Follow the steps under one of the following tabs:

**Note**
If you are using an existing Amazon S3 bucket or creating a new bucket, it must be located in the same region as AWS RoboMaker.

Using the console

2. In the left navigation pane, choose Simulations, and then choose Simulation jobs.
3. Choose Create simulation job.
4. On the Simulation configuration page, select a simulation job duration. Select any value between 5 minutes and 14 days.

**Important**
To learn more about how you are charged for AWS RoboMaker see AWS RoboMaker Pricing.

5. Select a Failure behavior. Choose fail to terminate the host instance if the simulation job fails. Choose continue to keep the host instance so you can connect and investigate.

If you specify an optional S3 folder below, it will contain simulation data. It is available independent of the selected failure behavior.

6. For IAM Role, select a role or select Create new role to create one. AWS RoboMaker will use this role to access resources on your behalf. It is also used by your application to access AWS resources like Amazon Rekognition or Amazon Lex.

7. Optional: In Output destination, type in a Amazon S3 folder name where simulation job output will be stored. Optionally, select Create new S3 folder to create a new Amazon S3 folder.

8. Optional: In Networking, if your robot application or simulation application accesses resources on an Amazon VPC, select the VPC, subnets and security groups. If you want to access the simulation job from outside of the VPC, select Assign public IP.

9. Optional: To connect to your simulation application or robot application remotely, select Enable connectivity to simulation, then specify port mappings for the robot application and simulation application.

**Warning**
You are responsible for configuring a secure remote connection to the simulation job. We recommend you implement a strong authentication method and encryption in transit for the ports you are opening. For more information about remote connectivity, see Connecting to a Simulation Job (p. 62).

10. Optionally, under Tags, specify one or more tags for the simulation job. Tags are words or phrases that act as metadata for identifying and organizing your AWS resources. Each tag consists of a key and a value. You can manage tags for your simulation job on the Simulation Job details page.

For more about tagging, see Using Cost Allocation Tags in the AWS Billing and Cost Management User Guide.

11. Choose Next.

12. On the Specify robot application page, under Robot application, select Create new application. Optionally, you can select Choose existing application to use a robot application that you have already created.

13. Type a name for the robot application.

14. Under Sources, specify the Amazon S3 location for the X86_64 robot application source. AWS RoboMaker simulation jobs require an X86_64 source to run the simulation.
Optionally, if you plan on deploying the robot application to robots in a fleet, you can provide ARMHF and ARM64 robot application source files. You can also update the robot application to include additional source files. For more information, see Updating a Robot Application (p. 54).

15. In **Robot application configuration**, provide the roslaunch **Launch package name** for your robot application.

16. Specify the roslaunch **Launch file**. A launch file contains configuration information about which nodes to start up as well as other initialization parameters for roslaunch.

To learn more about roslaunch, see [roslaunch](https://wiki.ros.org/roslaunch).

17. **Optional**: If your application includes a graphical user interface, select **Run with streaming session**. AWS RoboMaker will configure a connection so you can interact with your application as it is running in the simulation. You can connect by selecting **Robot Application** under **Simulation tools** on the simulation job detail page.

18. **Optional**: If your robot application uses environment variables, specify the **Name** and **Value** pairs. Environment variable names must start with A-Z or underscore and consist of A-Z, 0-9 and underscore. Names beginning with "AWS" are reserved.

Select **Add environment variable** to add additional variables.

You can read environment variables in a launch file using roslaunch substitution args.

19. Choose **Next**.

20. On the **Specify simulation application** page, select **Create new application**. Optionally, you can select **Choose existing application** use a simulation application that you have already created.

21. Type a **name** for the simulation application.

22. Select the **Simulation software suite** and **Simulation rendering version** used by your simulation application.

23. Under **Sources**, specify the Amazon S3 location for the **X86_64** simulation application source. AWS RoboMaker simulation jobs require an **X86_64** source to run the simulation.

24. In **Simulation application configuration**, provide the roslaunch **Launch package name** and the roslaunch **Launch file** for your simulation application.

25. **Optional**: If your application includes a graphical user interface, select **Run with streaming session**. AWS RoboMaker will configure a connection so you can interact with your application as it is running in the simulation. You can connect by selecting **Simulation Application** under **Simulation tools** on the simulation job detail page.

26. **Optional**: If your simulation application uses environment variables, specify the **Name** and **Value** pairs. Select **Add environment variable** to add additional variables.

27. Choose **Next**.

28. Select **Create** to create the simulation job.

---

Using the AWS CLI

**Example**

Here's an example AWS CLI command that performs the equivalent of the console-based simulation job creation on the other tab.

```bash
$ aws robomaker create-simulation-job --max-job-duration-in-seconds 3600
--iam-role arn:aws:iam::111111111111:role/MyRole
--robot-applications
--simulation-applications application=arn:aws:robomaker:us-
```
Create a Simulation Job using ROS bags

You must configure your ROS launch file to play back the ROS bag files you use as data sources. For more information about configuring your ROS launch file to play back ROS bag messages, see Using ROS Bags for Play Back (p. 69).

To create a simulation job using RosbagPlay

Follow the steps below:

**Note**
If you are using an existing Amazon S3 bucket or creating a new bucket, it must be located in the same region as AWS RoboMaker.

2. In the left navigation pane, choose Simulations, and then choose Simulation jobs.
3. Choose Create simulation job.
4. On the Simulation configuration page, select a simulation job duration. Select any value between 5 minutes and 14 days.

**Important**
To learn more about how you are charged for AWS RoboMaker see AWS RoboMaker Pricing.

5. Select a Failure behavior. Choose fail to terminate the host instance if the simulation job fails. Choose continue to keep the host instance so you can connect and investigate.

   If you specify an optional S3 folder below, it will contain simulation data. It is available independent of the selected failure behavior.

6. For IAM Role, select a role or select Create new role to create one. AWS RoboMaker will use this role to access resources on your behalf. It is also used by your application to access AWS resources like Amazon Rekognition or Amazon Lex.

7. **Optional:** In Output destination, type in a Amazon S3 folder name where simulation job output will be stored. Optionally, select Create new S3 folder to create a new Amazon S3 folder.

8. **Optional:** In Networking, if your robot application or simulation application access resources on an Amazon VPC, select the VPC, subnets and security groups. If you want to access the simulation job from outside of the VPC, select Assign public IP.

9. Optionally, under Tags, specify one or more tags for the simulation job. Tags are words or phrases that act as metadata for identifying and organizing your AWS resources. Each tag consists of a key and a value. You can manage tags for your simulation job on the Simulation Job details page.

   For more about tagging, see Tagging Your AWS RoboMaker Resources (p. 125).

10. Choose Next.
11. On the Specify robot application page, under Robot application, select Create new application. Optionally, you can select Choose existing application to use a robot application that you have already created.
12. Type a name for the robot application.
13. Under Sources, specify the Amazon S3 location for the X86_64 robot application source. AWS RoboMaker simulation jobs require an X86_64 source to run the simulation.

   Optionally, if you plan on deploying the robot application to robots in a fleet, you can provide ARMHF and ARM64 robot application source files. You can also update the robot application to include additional source files. For more information, see Updating a Robot Application (p. 54).
14. In **Robot application configuration**, provide the roslaunch **Launch package name** for your robot application.

15. Specify the roslaunch **Launch file**. A launch file contains configuration information about which nodes to start up as well as other initialization parameters for roslaunch.

To learn more about roslaunch, see roslaunch.

16. **Optional:** If your application includes a graphical user interface, select **Run with streaming session**. AWS RoboMaker will configure a connection so you can interact with your application as it is running in the simulation. You can connect by selecting **Robot Application** under **Simulation tools** on the simulation job detail page.

17. **Optional:** If your robot application uses environment variables, specify the **Name** and **Value** pairs. Environment variable names must start with A-Z or underscore and consist of A-Z, 0-9 and underscore. Names beginning with “AWS” are reserved.

Select **Add environment variable** to add additional variables.

You can read environment variables in a launch file using roslaunch substituion args.

18. Choose **Next**.

19. On the **Specify simulation application** page, select **Create new application**. Optionally, you can select **Choose existing application** use a simulation application that you have already created.

20. Type a **name** for the simulation application.

21. For **Software suite name**, select **RosbagPlay**, Kinetic will automatically be selected as the **Software suite version**.

22. Select the **Browse S3**, and then specify the path to your simulation application.

23. In **Simulation application configuration**, provide the roslaunch **Launch package name** and the roslaunch **Launch file** for your simulation application.

24. **Optional:** If your application includes a graphical user interface, select **Run with streaming session**. AWS RoboMaker will configure a connection so you can interact with your application as it is running in the simulation. You can connect by selecting **Simulation Application** under **Simulation tools** on the simulation job detail page.

25. **Optional:** If your simulation application uses environment variables, specify the **Name** and **Value** pairs. Select **Add environment variable** to add additional variables.

26. In **Data source configuration**, provide a **ROS bag group name**, then choose **Browse S3** to select **ROS bag files**. The files should contain ROS messages in the same format used by rosbag record. Choose **Add group** to add additional groups of data files.

   **Note**
   You can select up 100 files with a combined size less than 25 GB across all groups. Performance may be impacted as the combined size of the data files increase.

27. Choose **Next**.

28. Select **Create** to create the simulation job.

---

**Viewing a Simulation Job**

You can view information about a simulation job and, if the job is running, launch Gazebo, rviz, rqt, or a terminal to interact with the simulation. You can also view details about the simulation job and manage tags.

**To see the details of a simulation job**

Follow the steps under one of the following tabs:

Using the console
2. In the left navigation pane, choose Simulations, then choose Simulation jobs.
3. Select the Id of a simulation job to view its details including the time it was created and launch commands for the robot application and simulation application.

Using the AWS CLI

**Example**

Here's an example AWS CLI command that performs the equivalent of the console-based view simulation job on the other tab.

```
$ aws robomaker list-simulation-jobs
$ aws robomaker describe-simulation-job --job my-simulation-job-arn
```

**Cancelling a Simulation Job**

A simulation job can be cancelled if it is running and no longer needed.

**To cancel a simulation job**

Follow the steps under one of the following tabs:

Using the console

2. In the left navigation pane, choose Simulations, then choose Simulation jobs.
3. Select the Id of the simulation job you want to cancel.
4. In the Simulation job detail page, under Actions, choose Cancel.
5. In the Cancel simulation job page, select Yes, cancel.

Using the AWS CLI

**Example**

Here's an example AWS CLI command that performs the equivalent of the console-based cancel simulation job on the other tab.

```
$ aws robomaker list-simulation-jobs
$ aws robomaker cancel-simulation-job --job my-simulation-job-arn
```

**Cloning a Simulation Job**

You can create a new simulation job from an existing simulation job by cloning it from the Simulation job detail page.

Restarting a Simulation Job

Running simulation jobs can be restarted. When restarted, the simulation job will use the robot application and simulation application source files in the Amazon S3 location and all other configuration settings specified when the simulation job was created.

To restart a simulation job

Follow the steps under one of the following tabs:

Using the console

2. In the left navigation pane, choose Simulations, then choose Simulation jobs.
3. Select the Id of a running simulation job you would like to restart.
4. In the Simulation job detail page, under Actions, choose Clone.
5. In the Review and create simulation job, select Edit to make changes and then select Create.

Using the AWS CLI

Example

Here's an example AWS CLI command that performs the equivalent of the console-based restart simulation job on the other tab. The simulation job must be running.

```
$ aws robomaker restart-simulation-job --job my-simulation-job-arn
```

Managing Simulation Job Batches

This section provides information about how you can start and manage simulation job batches. Using a simulation job batch, you can launch and run many simulations using a single API call. This makes it easy to perform regression testing, parameter optimization, machine learning model training, and synthetic data generation.

Note

Simulation job batches can only be started using the AWS RoboMaker SDK or AWS CLI. You can view, clone, and cancel simulation batches using the AWS RoboMaker console.

Topics

- Starting a Simulation Job Batch (p. 81)
- Viewing a Simulation Job Batch (p. 82)
- Cancelling a Simulation Job Batch (p. 82)
Starting a Simulation Job Batch

Simulation job batches are started from the AWS SDK or AWS CLI. A simulation job batch includes one or more simulation job requests. Each simulation job request identifies which applications to use in each simulation, the maximum duration of the job, and other information. You can apply tags to the simulation job batch and each simulation job request.

To start a simulation job batch, you must do the following:

1. Install the AWS Command Line Interface. For more information about installing the AWS CLI, see Installing the AWS CLI.
2. Copy the following JSON into a file named `startsimjobbatch.json`. Modify the file to match your desired configuration, and then save it.

```json
{
    "batchPolicy": {
        "timeoutInSeconds": 400,
        "maxConcurrency": 2
    },
    "createSimulationJobRequests": [
        {
            "maxJobDurationInSeconds": 300,
            "iamRole": "arn:aws:iam::111111111111:role/MyRole",
            "failureBehavior": "Fail",
            "robotApplications": [
                {
                    "launchConfig": {
                        "packageName": "hello_world_robot",
                        "launchFile": "rotate.launch"
                    }
                }
            ],
            "simulationApplications": [
                {
                    "application": "arn:aws:robomaker:us-east-1:111111111111:simulation-application/MySimulationApplicationArn",
                    "launchConfig": {
                        "packageName": "hello_world_simulation",
                        "launchFile": "simulation.launch"
                    }
                }
            ],
            "tags": {
                "myRequestTagKey": "myRequestTagValue"
            }
        },
        {
            "maxJobDurationInSeconds": 200,
            "iamRole": "arn:aws:iam::111111111111:role/MyRole",
            "failureBehavior": "Fail",
            "simulationApplications": [
                {
                    "application": "arn:aws:robomaker:us-east-1:111111111111:simulation-application/MySimulationApplicationArn",
                    "launchConfig": {
                        "packageName": "hello_world_simulation",
                        "launchFile": "simulation.launch"
                    }
                }
            ]
        }
    }
}
```
3. Open a command prompt, then run the following AWS CLI command:

```
$ aws robomaker start-simulation-job-batch start-simulation-job-batch --cli-input-json file://startsimjobbatch.json
```

To view the simulation job batch, see Viewing a Simulation Job Batch (p. 82).

**Viewing a Simulation Job Batch**

You can view information about a simulation job batch including details about simulation job requests in the batch.

**To see the details of a simulation job batch**

Follow the steps under one of the following tabs:

**Using the console**

2. In the left navigation pane, choose **Simulations**, then choose **Simulation job batches**.
3. Select the **Id** of a simulation job batch to view its details.

**Using the AWS CLI**

**Example**

Here’s an example AWS CLI command that performs the equivalent of the console-based view simulation job on the other tab.

```
$ aws robomaker list-simulation-job-batches
$ aws robomaker describe-simulation-job-batch --job my-simulation-job-batch-arn
```

**Cancelling a Simulation Job Batch**

A simulation job can be cancelled if it is running and no longer needed.

**To cancel a simulation job**

Follow the steps under one of the following tabs:

**Using the console**
2. In the left navigation pane, choose Simulations, then choose Simulation job batches.
3. Select the Id of the simulation job batch you want to cancel.
4. In the Simulation job batch detail page, under Batch actions, choose Cancel batch.
5. In the Cancel simulation job batch page, select Cancel.

Using the AWS CLI

Example

Here's an example AWS CLI command that performs the equivalent of the console-based cancel simulation job batch on the other tab.

```bash
# aws robomaker list-simulation-job-batches
# aws robomaker cancel-simulation-job-batch --job my-simulation-job-batch-arn
```

Cloning a Simulation Job Batch

You can start a new simulation job batch by cloning an existing batch. When you clone, you can include all of the simulation job requests or select a subset of requests.

To clone a simulation job batch:

2. In the left navigation pane, choose Simulations, then choose Simulation job batches.
3. Select the Id of the simulation job batch you want to clone.
4. To clone the entire batch, in the Simulation job batch detail page, under Batch actions, choose Clone batch.

   To clone specific simulation job requests from the batch, under Simulation job requests, check the simulation job requests you want to clone, then select Request actions and choose Clone request.

5. In the Clone simulation job batch page, select Submit.

Simulation Tools

AWS RoboMaker provides Gazebo, rqt, rviz and terminal access to interact with running simulation jobs.

Common tasks include:

- Pause a Running Simulation (p. 84)
- View Node Graph (p. 86)
- View Robot Sensor Data (p. 87)
- Inspect ROS Topics and Messages (p. 87)

The IAM user or role used to create simulation will automatically have permission to access the simulation tools. If it is a different user or role, it should have the robomaker:CreateSimulationJob privilege.

Topics
Pause a Running Simulation

You can pause a running simulation in Gazebo by selecting the pause icon. It is located under the rendering of the world on the left.

When a running simulation is paused, it is paused in other simulation tools like rqt and rviz. This is useful for investigating simulation data at a moment in time. For example, using rqt to examine image data from a video camera mounted on a robot.

View Robot and Objects in the Simulation

When you open Gazebo, it presents a view of the simulated world. The initial perspective is configured by the simulation application developer.

1. **In Gazebo**, use the mouse or keyboard to explore the world. Zoom in, pan out, and move the world around.
2. Switch to an orthographic (or perspective) camera angle. In the menu, select **Camera** and then choose **Orthographic** (or **Perspective**). Reset the camera by choosing **Reset View Angle**.
3. Select an alternate view to see objects differently. For example, select **View** and then choose **Wireframe** to see the world rendered as a wireframe.
4. It is easy to reset the world to its original configuration. Select **Edit** and then choose **Reset World**. Select **Reset Model Poses** to revert changes to model poses.

Add and Move Objects in the Simulation

Gazebo includes a collection of models that can be used to create an environment. Objects can be placed in the environment, moved, and posed to meet the needs of the scenario.

1. **In Gazebo**, on the left, select the **Insert tab**.
2. In the **Insert** tab, choose **Bookshelf**, then move the cursor to the room. As you move into the room, you will see the bookshelf model. Click the left mouse button to place it in the room.

3. Move the bookshelf by selecting Translation mode. Choose the multi-arrow plus icon in the menu or use the keyboard shortcut `Control-T`. Select the bookshelf, then move it to a new location and click the mouse button.

4. Press **Escape** to exit Translation mode. Select the bookshelf and then in the **World** tab, expand **Pose** to see different pose settings. Select a value and then change one increment at a time using the up and down selectors. Gazebo updates the world after each click.

### Apply Forces to Robots and Objects

Things do not always go as planned in the physical world. A robot might be subjected to unexpected forces and disturbances during operation. Objects might tumble, spin, and interact with neighboring objects or the robot itself. Using Gazebo, you can create disturbances by applying force and/or torque to models during simulation.

This example uses the **Navigation and Person Recognition** (p. 15) sample. For more information, see AWS RoboMaker sample applications. The principles apply to robots and objects that are not static. Entities marked as static only have collision geometry.

1. In **Gazebo**, verify the simulation is running. The simulation must be running to see how an object responds to force and torque.

2. In the **World** tab on the left, expand **turtlebot3_waffle_pi**. Right click **wheel_left_link** and then choose **Apply Force/Torque**.

3. Under **Force**, specify an X value of **1000**. Use the mouse or keyboard to move the underlying view so the robot is in view, then choose **Apply Force**. The robot will mostly rotate counter-clockwise.

   Select **Clear** and then choose **Apply Force** to remove the force.

4. Now apply enough torque on the Y-axis to tumble the robot upsidedown. Under **Torque**, specify a Y value of **400**. Make sure the robot is in view, then choose **Apply Torque**. The robot will flip upsidedown. Choose **Apply Torque** and it will tumble upright.

   Select **Clear** and then choose **Apply Torque** to remove the torque. Select **Cancel** to close the dialog box.

5. No try applying force to an object. In the **World** tab on the left, right click **ChairA_01_001** and then choose **Apply Force**. Use the mouse or keyboard to make sure the chair is in view.

6. Under **Force**, specify an Z value of **50000**, then choose **Apply Force**. The chair will launch off the ground and then return to rest.

   Select **Clear** and then choose **Apply Force** to remove the force. Select **Cancel** to close the dialog box.

### rqt

rqt hosts a number of different plugins for visualizing ROS information. Multiple plugins can be displayed on a custom dashboard, providing a unique view of your robot. rqt includes many useful plugins and provides a framework to write custom plugins.

To perform the tasks below, rqt must be open and connected to a running simulation job. You can open rqt from the **Simulation jobs detail** page of a running simulation job.

**Topics**

- View Image Data from Robot (p. 86)
- View Node Graph (p. 86)
• View Currently Advertised Topics (p. 86)

View Image Data from Robot

rqt provides a plugin to help visualize image data from a robot. The image data is updated as the robot moves in the simulated world.

1. In rqt, choose Plugins, Visualization, Image View.
2. In the Image View view, choose /camera/rgb/image_raw from the dropdown. To pause the simulation, see Pause a Running Simulation (p. 84).

View Node Graph

The node graph is a visual representation of all of the ROS nodes and topics in your application. Directional arrows indicate which nodes (ovals) are advertising or subscribing to a topic (squares). You can filter the graph to show all topics, active topics, or nodes only. There are also options to hide or display group, topic and node information.

The graph node view is helpful for verifying which nodes are running and to confirm that nodes and topics are connected as expected.

To view the node graph

1. In rqt, select Plugins, then select Introspection, and then choose Node Graph.
2. Use the mouse or keyboard to zoom in on the graph. Select a node or a topic to see subscriptions (purple arrows) and advertisements (green arrows).
3. The filter defaults to Nodes only. Choose Nodes/Topics (all) to see all of the nodes and topics. Toggle checkboxes to display additional information such as unreachable nodes or dead sinks.

View Currently Advertised Topics

The Topic Monitor plugin makes it easy to view information about ROS topics including publishers, subscribers and publishing rate, and ROS messages. It is also helpful for validating message data, identifying nodes that are not publishing messages, and look for bandwidth issues.

To monitor topics

1. In rqt, select Plugins, then select Topics, and then choose Topic Monitor.
2. Use the mouse or keyboard to scroll the list of currently advertised topics. Expand a topic to see message details.
3. Select the checkbox to the left of the expanded topic to subscribe to its messages. As new message arrive, messages data, message bandwidth, and publication frequency are updated.

rviz

rviz is a 3d visualization tool for ROS applications. It provides a view of your robot model, capture sensor information from robot sensors, and replay captured data. It can display data from camera, lasers, from 3D and 2D devices including pictures and point clouds.

To perform the tasks below, rviz must be open and connected to a running simulation job. You can open rviz from the Simulation jobs detail page of a running simulation job.

Topics
View Robot Sensor Data

Robots typically have sensors to gather data from the world. For example, a robot might use a sensor to detect a collision with an object or a laser scanner to learn about objects in the surrounding environment.

1. In rviz, select File and then choose Open Config. In the dialog box, navigate to robomaker/workspace/bundle-store/GUID/opt/ros/$ROS_DISTRO/share/turtlebot3_description/rviz. Select model.rviz and then choose Open. It does not matter which GUID directory you choose.

   If prompted about unsaved changes, select Discard.
2. In the display, laser scan data appears as red dots. Walls and other objects can be identified by lines of red dots centered on the robot. The laser scan will only be visible if there are objects in the virtual world.
3. Open Gazebo. Compare the location and orientation of the robot in Gazebo to what the laser scan detects.
4. In Gazebo, drop a sphere next to the robot. In rviz, the laser scan detects part of the object.

Terminal

The Terminal provides access to a command-line on the simulation job host.

To perform the tasks below, the Terminal must be open and connected to a running simulation job. You can open the Terminal from the Simulation jobs detail page of a running simulation job.

   Note
   Launching GUI applications in the terminal window is unsupported.

Topics

- Inspect ROS Topics and Messages (p. 87)
- Inspect ROS Nodes and Services (p. 88)
- View Log Files in Real Time (p. 88)

Inspect ROS Topics and Messages

Use rostopic to display information about ROS topics. For more information about rostopic, see http://wiki.ros.org/rostopic.

1. In terminal, type in the following commands to set up the ROS environment:

   eval $AWS_ROBOMAKER_ROBOT_APPLICATION_SETUP

2. Type the following command to see a list of available topics:

   rostopic list

3. Use the following command to view messages associated with a listed topic:

   rostopic echo /topic_name
Inspect ROS Nodes and Services

Use `rosnodes` to display information about ROS nodes and services. For more information about `rosnodes`, see [http://wiki.ros.org/rosnodes](http://wiki.ros.org/rosnodes).

1. In terminal, type in the following commands to set up the ROS environment:
   
   ```bash
   eval $AWS_ROBOMAKER_ROBOT_APPLICATION_SETUP
   ```

2. Type the following command to see a list of available topics:
   
   ```bash
   rosnode list
   ```

3. Use the following command to view messages associated with a listed topic:
   
   ```bash
   rostopic info /node_name
   ```

View Log Files in Real Time

Log files are written to `tmp/robot-logs/stdout_and_stderr` and `tmp/simulation-logs/stdout_and_stderr`. For example, to view the last part of the robot application log, use the following command:

```bash
tail -f /tmp/robot-logs/stdout_and_stderr
```
Fleet Management

This section contains information on working with robots, fleets, and deployments.

Topics

- Deploying a Robot Application (p. 89)
- Managing Robots (p. 90)
- Managing Fleets (p. 93)
- Managing Deployments (p. 95)

Deploying a Robot Application

To deploy a robot application to a physical robot, the physical robot must be configured to receive deployments and belong to a fleet. At a minimum, to deploy, do this:

1. Create a robot application with sources for the architectures of your robots. Supported architectures are X86_64, ARM64 and ARMHF.

2. Create the robot in AWS RoboMaker and configure it with AWS IoT Greengrass so it can receive deployments. AWS RoboMaker uses AWS IoT Greengrass to deploy the robot application. Each robot has one AWS IoT Greengrass group which internally has one AWS IoT Greengrass core.

If your robot application uses AWS RoboMaker cloud extensions or other AWS services, then grant permissions for the robot application to access them.

3. Register the robot in a fleet. A fleet is logical grouping of robots with shared functionality defined by the robot application.

4. Create a deployment to install the robot application to the fleet. Pick the robot application version, customize the launch configuration (including pre- and post-launch actions), and specify how the robot application is deployed.

5. Monitor the deployment. You can track the progress of your deployment and other information in deployment details. Customize your robot application to provide additional information by using AWS RoboMaker cloud extensions.

You can learn how to do these steps in Getting Started with AWS RoboMaker (p. 8).

How Robot Applications are Deployed

When a robot application is deployed to a physical robot, AWS RoboMaker does the following:

1. AWS RoboMaker creates or updates a custom Lambda in your account. The Lambda contains the logic needed for deployment. This includes robot application bundle download, ROS launch, pre- and post-checks, and other logic.

2. AWS RoboMaker begins deployment to the fleet using the parallelization specified in the deployment configuration.

3. AWS RoboMaker notifies AWS IoT Greengrass to run the custom Lambda on the target robot. The daemon running on the robot receives the command and runs the Lambda. If a Lambda is running when the command is received, it and all ROS process on the robot are terminated.
Managing Robots

In this section, you learn how to create and delete robots.

Topics
- Creating a Robot (p. 90)
- Deleting a Robot (p. 92)

Creating a Robot

Before you can deploy a robot application, you must configure your robot hardware. When you create a robot, you select the hardware architecture and an IAM role for AWS IoT Greengrass. After the robot is created, download the AWS IoT Greengrass core and security resources, then configure your robot hardware.

Topics
- Create deployment role (p. 90)
- Create a robot (p. 91)

Create deployment role

Before you create a robot, create an IAM role for robot application deployment. The role will also be used by AWS RoboMaker to access resources like Amazon S3 (where your robot application is placed prior to deployment) and by your robot application to access resources it consumes like Amazon Lex or Amazon Rekognition.

If you have already created these roles, you can skip to Create a robot (p. 91).

To create the AWS IoT Greengrass role

1. Sign in to the AWS Management Console and open the AWS Identity and Access Management console at console.aws.amazon.com/iam.
2. Create the access policy. On the left, choose Policies, then choose Create policy. Choose JSON and paste the code below:

```json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "robomaker:UpdateRobotDeployment"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
```
To create a robot:

2. In the left navigation pane, choose Fleet Management, and then choose Robots.
3. Choose Create robot.
4. In the Create robot page, type a name for the robot.
5. Select the Architecture of the robot.
6. Under AWS IoT Greengrass group defaults, select a Create new to create a new AWS IoT Greengrass group for the robot. Optionally, you can select an existing AWS IoT Greengrass group. Each robot must have its own AWS IoT Greengrass group.

   If you use an existing AWS IoT Greengrass group, it must have an IAM role associated with it. To create the role, see Create deployment role (p. 90).

7. Optionally, modify the Greengrass prefix. This string is prepended to AWS IoT Greengrass objects created on your behalf.
8. Select an IAM role to assign to the AWS IoT Greengrass group created for the robot. It grants permissions for AWS IoT Greengrass to access your robot application in Amazon S3 and read update status from AWS RoboMaker.

9. Optionally, under Tags, specify one or more tags for the robot. Tags are words or phrases that act as metadata for identifying and organizing your AWS resources. Each tag consists of a key and a value. You can manage tags for your robot on the Robot details page.

For more about tagging, see Tagging Your AWS RoboMaker Resources (p. 125).

10. Choose Create.

11. In the Download your Core device page, choose Download to download and store your robot's security resources.

12. Download AWS IoT Greengrass core software matching the architecture of your physical robot. To configure and run the AWS IoT Greengrass core software, follow the steps in Module 1: Environment Setup for Greengrass. Then follow the steps in Start AWS Greengrass on the Core Device.

Use the following command to unzip your security resources:

```
$ sudo unzip RobotName-setup.zip -d /greengrass
```

# Deleting a Robot

When you no longer need a robot, you can delete it. You can delete a robot that is unregistered or registered as part of a fleet, even if there is an active deployment to the fleet.

The AWS IoT Greengrass group and other assets created for the robot by AWS RoboMaker are not deleted. You can create a new robot and reuse the group. To delete AWS IoT Greengrass resources, use the https://console.aws.amazon.com/iot/.

**To delete a robot**

Follow the steps under one of the following tabs:

**Using the console**

2. In the left navigation pane, choose Fleet Management, and then choose Robots.
3. Select the robot you want to delete, then choose Delete.

**Note**

To delete the underlying AWS IoT Greengrass group and resources, use the https://console.aws.amazon.com/iot/.

**Using the AWS CLI**

**Example**

Here's an example AWS CLI command that performs the equivalent of the console-based delete robot on the other tab.

**Note**

To delete the underlying AWS IoT Greengrass group and resources, use the https://console.aws.amazon.com/iot/.
Managing Fleets

A fleet is a group of robots. When you are ready to deploy a robot application to your robots, they must be part of a fleet.

**Topics**
- Creating a Fleet (p. 93)
- Registering and Deregistering Robots (p. 93)
- Deleting a Fleet (p. 94)

**Creating a Fleet**

To create a fleet, follow the steps under one of the following tabs:

Using the console

2. In the left navigation pane, choose **Fleet Management**, and then choose **fleets**.
3. Select **Create fleet**.
4. In the **Create fleet** page, type a **name** for the fleet.
5. Optionally, under **Tags**, specify one or more tags for the fleet. Tags are words or phrases that act as metadata for identifying and organizing your AWS resources. Each tag consists of a key and a value. You can manage tags for your fleet on the **Fleet details** page.
   
   For more about tagging, see Tagging Your AWS RoboMaker Resources (p. 125).
6. Click **Create** to create the deployment job.

Using the AWS CLI

**Example**

Here's an example AWS CLI command that performs the equivalent of the console-based fleet creation on the other tab.

```
$ aws robomaker create-fleet --name my-fleet
```

**Registering and Deregistering Robots**

You can register (add) and deregister (remove) robots from fleets. This is useful if you want to remove a robot from a fleet for maintenance or move a robot from one fleet to another fleet.

You can register a robot to a single fleet only.

**Register a robot**

To register a robot to a fleet, follow the steps under one of the following tabs:
Deleting a Fleet

Using the console

2. In the left navigation pane, choose Fleet Management, and then choose Fleets.
3. Select the Name of the fleet you want to modify.
4. In the Fleet details page, select Register.
5. In the Register robots page, select the robot you want to register, then select Register robots.

Using the AWS CLI

Example

Here's an example AWS CLI command that performs the equivalent of the console-based robot registration on the other tab.

```
$ aws robomaker register-robot --fleet my-fleet-arn --robot my-robot-arn
```

Deregister a robot

To deregister a robot, follow the steps under one of the following tabs:

Using the console

2. In the left navigation pane, choose Fleet Management, and then choose Fleets.
3. Select the Name of the fleet you want to modify.
4. In the Fleet details page, select the robot you want to deregister, then select Deregister.

Using the AWS CLI

Example

Here's an example AWS CLI command that performs the equivalent of the console-based robot deregistration on the other tab.

```
$ aws robomaker deregister-robot --fleet my-fleet-arn --robot my-robot-arn
```

Deleting a Fleet

When you no longer need a fleet, you can delete it. When you do this, robots registered to the fleet are deregistered.

To delete a fleet

Follow these steps under one of the following tabs:

Using the console
2. In the left navigation pane, choose Fleet Management, and then choose Fleets.
3. Select the fleet you want to delete, then select Delete.

Using the AWS CLI

**Example**

Here's an example AWS CLI command that performs the equivalent of the console-based fleet deletion on the other tab.

```
$ aws robomaker delete-fleet --fleet my-fleet-arn
```

Managing Deployments

In AWS RoboMaker, a robot application is delivered and installed onto a fleet of physical robots using deployment job. The robot application must be build for the architecture supported by the physical robot (for example, ARMHF).

**Topics**

- Conditional Deployment (p. 95)
- Creating a deployment job (p. 96)
- Viewing a Deployment Job (p. 97)

Conditional Deployment

A robot application is downloaded and installed onto robots in a fleet using a deployment job. Any tasks the robot is performing will halt as the new version of the robot application is installed. You can verify your robot is ready to download and install the new robot application using a download condition file.

The **download condition file** is a script run on the robot prior to downloading the new deployment. If the script exits with 0, verification succeeded and the deployment can proceed on the robot. If the script exits with 1, the deployment will not be downloaded and installation will fail.

**Note**

To use a download condition file, you must have AWS IoT Greengrass Core version 1.9.4 or newer installed on your robots.

Use the following script as a template for your download condition file:

```
#!/bin/bash
# sample command as condition result
# for example, you could check to see if the robot is in a charging station or other suitable spot for a deployment
conditionalScriptPass="<Condition_Verification_Commands>
if [[ ! -z "$conditionalScriptPass" ]]; then
    #condition pass
    echo succeeded
    exit 0
else
    #condition failed
    echo failed
```
Creating a deployment job

Create a deployment job to install a unique version of a robot application on robots in a fleet. You can define custom environment variables and run a script before and after your application launches on the robot to perform additional configuration.

For more details about how AWS RoboMaker deploys a robot application, see How Robot Applications are Deployed (p. 89).

Note
After 90 days, deployment jobs expire and will be deleted. They will no longer be accessible.

Create a deployment job

Using the console

To create a deployment job:

2. In the left navigation pane, choose Fleet Management, and then choose Deployments.
3. Click Create deployment.
4. In the Create deployment page, under Configuration, select a Fleet.
5. Select a Robot application.
6. Select the Robot application version to deploy. The robot application must have a numbered applicationVersion for consistency reasons. If there are no versions listed, or to create a new version, see Creating a Robot Application Version (p. 53).
7. Under Deployment launch config, specify the Package name.
8. Specify the Launch file.
9. Optionally, specify the Prelaunch file for your application. This is a script that is run before the ROS launch file. It can be used to check the robot environment or other tasks. A non-zero exit from the script causes the robot deployment to fail.

   The script should be copied into $CATKIN_GLOBAL_SHARE_DESTINATION.

   For example, add the following configuration to your CMakeLists.txt:

   ```
   install(FILES deploymentScripts/post_launch_script.sh
   DESTINATION $CATKIN_GLOBAL_SHARE_DESTINATION)
   ```

10. Optionally, specify the Postlaunch file for your application. This is a script that is run after launching ROS processes. It can be used to check the robot environment or other tasks. A non-zero exit from the script causes the robot deployment to fail.

   The script should be copied into $CATKIN_GLOBAL_SHARE_DESTINATION.
11. Optionally, under **Environment variables**, type in an environment **Name** and **Value**. Environment variable names must start with A-Z or underscore and consist of A-Z, 0-9 and underscore. Names beginning with "AWS" are reserved.

Select **Add environment variable** to create additional environment variables.

12. Under **Deployment config**, specify a **Concurrent deployment percentage**. AWS RoboMaker will deploy the robot application concurrently to a percentage of the fleet. If you have 200 robots in the fleet and choose 10%, deployment will be attempted on 20 robots simultaneously.

13. Specify a **Failure threshold percentage**. Deployment will halt if this percentage of your fleet experiences deployment failure.

   **Warning**
   
   Specify a failure threshold percentage larger than concurrent deployment percentage to ensure deployment halts at the threshold. If the value is smaller, the threshold can be exceeded up to the concurrent deployment percentage.

14. Specify a **Robot deployment timeout**. Deployment to an individual robot will stop if it does not complete before the amount of time specified.

15. Optionally, provide a **Download condition file in S3**. The file is a script you can use to verify the robot is ready to download and install the deployment. For example, you can check to see if the robot is in a charging station and not performing a task (like flying or moving objects).

16. Optionally, you can **lock S3 file to the latest etag**. The entity tag is a hash of the Amazon S3 object and reflects changes to the contents of the file, not its metadata. When selected, AWS RoboMaker will ensure that version is used during deployment.

17. Optionally, under **Tags**, specify one or more tags for the deployment. Tags are words or phrases that act as metadata for identifying and organizing your AWS resources. Each tag consists of a key and a value. You can manage tags for your deployment on the **Deployment details** page.

   For more about tagging, see Tagging Your AWS RoboMaker Resources.

18. Click **Create** to create the deployment job.

### Using the AWS CLI

#### Example

Here's an example AWS CLI command to create a deployment job.

```bash
$ aws robomaker create-deployment-job --fleet=my-fleet-arn --deployment-application-configs application=my-robotarn,applicationVersion="$LATEST",launchConfig={packageName="cloudwatch_robot",launchFile="cloudwatch_deploy.launch"}
--deployment-config concurrentDeploymentPercentage="100",failureThresholdPercentage="100"
```

### Viewing a Deployment Job

Once a deployment job is created, you can view its details and track the deployment status.

**To see the details of a deployment job**

Follow the steps under one of the following tabs:

#### Using the console

2. In the left navigation pane, choose **Fleet management**, then choose **Deployments**.
3. Click on the **Id** of a deployment job to see details about the job including the time it was created and robot application version, deployment status, and the status of each robot in the fleet.

Using the AWS CLI

**Example**

Here's an example AWS CLI command that performs the equivalent of the console-based view deployment job on the other tab.

```
# aws robomaker list-deployment-jobs
# aws robomaker describe-deployment-job --job my-deployment-job-arn
```
Security

This section provides guidelines for securing different aspects of AWS RoboMaker.

Topics

• Data Protection in AWS RoboMaker (p. 99)
• Authentication and Access Control for AWS RoboMaker (p. 99)
• Logging and Monitoring in AWS RoboMaker (p. 121)
• Security Compliance (p. 123)
• Resilience in AWS RoboMaker (p. 123)
• Infrastructure Security in AWS RoboMaker (p. 124)

Data Protection in AWS RoboMaker

AWS RoboMaker conforms to the AWS shared responsibility model, which includes regulations and guidelines for data protection. AWS is responsible for protecting the global infrastructure that runs all the AWS services. AWS maintains control over data hosted on this infrastructure, including the security configuration controls for handling customer content and personal data. AWS customers and APN partners, acting either as data controllers or data processors, are responsible for any personal data that they put in the AWS Cloud.

For data protection purposes, we recommend that you protect AWS account credentials and set up individual user accounts with AWS Identity and Access Management (IAM), so that each user is given only the permissions necessary to fulfill their job duties. We also recommend that you secure your data in the following ways:

• Use multi-factor authentication (MFA) with each account.
• Use SSL/TLS to communicate with AWS resources.
• Set up API and user activity logging with AWS CloudTrail.
• Use AWS encryption solutions, along with all default security controls within AWS services.
• Use advanced managed security services such as Amazon Macie, which assists in discovering and securing personal data that is stored in Amazon S3.

We strongly recommend that you never put sensitive identifying information, such as your customers' account numbers, into free-form fields such as a Name field. This includes when you work with AWS RoboMaker or other AWS services using the console, API, AWS CLI, or AWS SDKs. Any data that you enter into AWS RoboMaker or other services might get picked up for inclusion in diagnostic logs. When you provide a URL to an external server, don’t include credentials information in the URL to validate your request to that server.

For more information about data protection, see the AWS Shared Responsibility Model and GDPR blog post on the AWS Security Blog.

Authentication and Access Control for AWS RoboMaker

AWS Identity and Access Management (IAM) is an AWS service that helps an administrator securely control access to AWS RoboMaker resources. Administrators use IAM to control who is authenticated
(signed in) and authorized (has permissions) to use AWS RoboMaker resources. IAM is a feature of your AWS account offered at no additional charge.

**Important**
To get started quickly, review the introductory information on this page and then see Getting Started with IAM (p. 114). You can optionally learn more about authentication and access control by viewing What is Authentication? (p. 106), What is Access Control? (p. 107), and What are Policies? (p. 109).

**Topics**
- Introduction to Authorization and Access Control (p. 100)
- Permissions Required (p. 101)
- Understanding How AWS RoboMaker Works with IAM (p. 104)
- Troubleshooting Authentication and Access Control (p. 105)

**Introduction to Authorization and Access Control**

**Authentication** – To sign in to AWS, you must use IAM user credentials, temporary credentials using IAM roles, or root user credentials (not recommended). To learn more about these entities, see What is Authentication? (p. 106).

**Access Control** – AWS administrators use policies to control access to AWS resources, such as the AWS RoboMaker robot application. To learn more, see What is Access Control? (p. 107) and What are Policies? (p. 109).

**Important**
All resources in an account are owned by the account, regardless of who created those resources. You must be granted access to create a resource. However, just because you created a resource does not mean that you automatically have full access to that resource. An administrator must explicitly grant permissions for each action that you want to perform. That administrator can also revoke your permissions at any time.

To help you understand the basics of how IAM works, review the following terms:

- **Resources** – AWS services, such as AWS RoboMaker and IAM, are made up of objects called resources. You can create, manage, and delete these resources from the service. IAM resources include users, groups, roles, and policies.
- **Users** – An IAM user represents the person or application who uses its credentials to interact with AWS. A user consists of a name, a password to sign into the AWS Management Console, and up to two access keys that can be used with the AWS CLI or AWS API.
- **Groups** – An IAM group is a collection of IAM users. You can use groups to specify permissions for its member users. This makes it easier for you to manage permissions for multiple users.
- **Roles** – An IAM role does not have any long-term credentials (password or access keys) associated with it. A role can be assumed by anyone who needs it and has permissions. An IAM user can assume a role to temporarily take on different permissions for a specific task. Federated users can assume a role by using an external identity provider that is mapped to the role. Some AWS services can assume a service role to access AWS resources on your behalf.
- **Policies** – Policies are JSON policy documents that define the permissions for the object to which they are attached. AWS supports identity-based policies that you attach to identities (users, groups, or roles). Some AWS services allow you to attach resource-based policies to resources to control what a principal (person or application) can do to that resource. AWS RoboMaker does not support resource-based policies.
- **Identities** – Identities are IAM resources for which you can define permissions. These include users, groups, and roles.
- **Entities** – Entities are IAM resources that you use for authentication. These include users and roles.
• **Principals** – In AWS, a principal is a person or application that uses an entity to sign in and make requests to AWS. As a principal, you can use the AWS Management Console, the AWS CLI, or the AWS API to perform an operation (such as deleting a robot application). This creates a request for that operation. Your request specifies the action, resource, principal, principal account, and any additional information about your request. All of this information provides AWS with context for your request. AWS checks all the policies that apply to the context of your request. AWS authorizes the request only if each part of your request is allowed by the policies.

To view a diagram of the authentication and access control process, see Understanding How IAM Works in the IAM User Guide. For details about how AWS determines whether a request is allowed, see Policy Evaluation Logic in the IAM User Guide.

### Permissions Required

To use AWS RoboMaker or to manage authorization and access control for yourself or others, you must have the correct permissions.

#### Permissions Required to Use the AWS RoboMaker Console

To access the AWS RoboMaker console, you must have a minimum set of permissions that allows you to list and view details about the AWS RoboMaker resources in your AWS account. If you create an identity-based permissions policy that is more restrictive than the minimum required permissions, the console won't function as intended for entities with that policy.

For full access to the AWS RoboMaker console, use the `AWSRoboMakerFullAccess` policy.

For read-only access to the AWS RoboMaker console, use the `AWSRoboMakerReadOnlyAccess` policy.

If an IAM user wants to create a simulation job, you need to grant `iam:PassRole` permission to that user. For more information about passing a role, see Granting a User Permissions to Pass a Role to an AWS Service.

For example, you can attach the following policy to a user. It provides permission to create a simulation job:

```json
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "arn:aws:iam::123456789012:role/S3AndCloudWatchAccess"
        }
    ]
}
```

You don't need to allow minimum console permissions for users that are making calls only to the AWS CLI or the AWS API. Instead, you need only the permissions that match the API operation you're trying to perform.

#### Permissions Required to Use the AWS RoboMaker Simulation Tools

The IAM user or role used to create simulation will automatically have permission to access the simulation tools. If it is a different user or role, it should have the `robomaker:CreateSimulationJob` privilege.
Permissions Required for Authentication Management

To manage your own credentials, such as your password, access keys, and multi-factor authentication (MFA) devices, your administrator must grant you the required permissions. To view the policy that includes these permissions, see Allow Users to Self-Manage Their Credentials (p. 117).

As an AWS administrator, you need full access to IAM so that you can create and manage users, groups, roles, and policies in IAM. You should use the AdministratorAccess AWS managed policy that includes full access to all of AWS. This policy does not provide access to the AWS Billing and Cost Management console or allow tasks that require root user credentials. For more information, see AWS Tasks That Require AWS Account Root User Credentials in the AWS General Reference.

**Warning**
Only an administrator user should have full access to AWS. Anyone with this policy has permission to fully manage authentication and access control, in addition to modifying every resource in AWS. To learn how to create this user, see Create your IAM Admin User (p. 114).

Permissions Required for Access Control

If your administrator provided you with IAM user credentials, they attached policies to your IAM user to control what resources you can access. To view the policies attached to your user in the AWS Management Console, you must have the following permissions:

```json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ViewOwnUserInfo",
      "Effect": "Allow",
      "Action": [
        "iam:GetUserPolicy",
        "iam:ListGroupsForUser",
        "iam:ListAttachedUserPolicies",
        "iam:ListUserPolicies",
        "iam:GetUser"
      ],
      "Resource": [
        "arn:aws:iam::*:user/${aws:username}"
      ]
    },
    {
      "Sid": "ListUsersViewGroupsAndPolicies",
      "Effect": "Allow",
      "Action": [
        "iam:GetGroupPolicy",
        "iam:GetPolicyVersion",
        "iam:GetPolicy",
        "iam:ListAttachedGroupPolicies",
        "iam:ListGroupPolicies",
        "iam:ListPolicyVersions",
        "iam:ListPolicies",
        "iam:ListUsers"
      ],
      "Resource": "*"
    }
  ]
}
```

If you need additional permissions, ask your administrator to update your policies to allow you to access the actions that you require.
Permissions Required for a Simulation Job

A simulation job, when it is created, must have an IAM role with the permissions below. Replace my-input-bucket with the name of the bucket containing the robot and simulation application bundles. Replace my-output-bucket to point to the bucket were AWS RoboMaker will write output files. Replace account# with your account number.

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Action": "s3:ListBucket",
         "Resource": ["arn:aws:s3:::my-input-bucket"],
         "Effect": "Allow"
      },
      {
         "Action": ["s3:Get*", "s3:List*"]
         "Resource": ["arn:aws:s3:::my-input-bucket/*"],
         "Effect": "Allow"
      },
      {
         "Action": "s3:Put*",
         "Resource": ["arn:aws:s3:::my-output-bucket/*"],
         "Effect": "Allow"
      },
      {
         "Action": ["logs:CreateLogGroup",
                     "logs:CreateLogStream",
                     "logs:PutLogEvents",
                     "logs:DescribeLogStreams"],
         "Resource": ["arn:aws:logs:*:account#log-group:/aws/robomaker/SimulationJobs*"],
         "Effect": "Allow"
      }
   ]
}
```

The policy must be attached to a role with the following trust policy:

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Principal": {
            "Service": "robomaker.amazonaws.com"
         },
         "Action": "sts:AssumeRole"
      }
   ]
}
```
Permissions Required to use Tags from a ROS Application or ROS Command Line

You can tag, untag, and list tags in your simulation job from the ROS command-line or in your ROS application while it is running. You must have an IAM role with the permissions below. Replace account# with your account number.

For more information, see Managing Tags in a Simulation Job (p. 67)

```json
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Action": [
            "robomaker:TagResource",
            "robomaker:UntagResource",
            "robomaker:ListTagsForResource",
         ],
         "Resource": [
            "arn:aws:robomaker:*:account#:simulation-job*"
         ],
         "Effect": "Allow"
      }
   ]
}
```

The policy must be attached to a role with the following trust policy:

```json
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Principal": {
            "Service": "robomaker.amazonaws.com"
         },
         "Action": "sts:AssumeRole"
      }
   ]
}
```

Understanding How AWS RoboMaker Works with IAM

Services can work with IAM in several ways:

- **Actions** – AWS RoboMaker supports using actions in a policy. This allows an administrator to control whether an entity can complete an operation in AWS RoboMaker. For example, to allow an entity to view a policy by performing the GetPolicy AWS API operation, an administrator must attach a policy that allows the iam:GetPolicy action.

- **Resource-level permissions** – AWS RoboMaker does not support resource-level permissions. Resource-level permissions allow you to use ARNs to specify individual resources in the policy. Because AWS RoboMaker does not support this feature, then you must choose All resources in the policy visual editor. In a JSON policy document, you must use * in the Resource element.

- **Resource-based policies** – AWS RoboMaker does not support resource-based policies. Resource-based policies allow you to attach a policy to a resource within the service. Resource-based policies include a Principal element to specify which IAM identities can access that resource.
• **Authorization based on tags** – AWS RoboMaker does support authorization based tags. This feature allows you to use resource tags in the condition of a policy.

• **Temporary credentials** – AWS RoboMaker supports temporary credentials. This feature allows you to sign in with federation, assume an IAM role, or to assume a cross-account role. You obtain temporary security credentials by calling AWS STS API operations such as AssumeRole or GetFederationToken.

• **Service-linked roles** – AWS RoboMaker supports service roles. This feature allows a service to assume a service-linked role on your behalf. This role allows the service to access resources in other services to complete an action on your behalf. Service-linked roles appear in your IAM account, and are owned by the service. An IAM administrator can view, but not edit the permissions for service-linked roles.

• **Service roles** – AWS RoboMaker supports service roles. This feature allows a service to assume a service role on your behalf. This role allows the service to access resources in other services to complete an action on your behalf. Service roles appear in your IAM account, and are owned by the account. This means that an IAM administrator can change the permissions for this role. However, this might break the functionality of the service.

## Troubleshooting Authentication and Access Control

Use the following information to help you diagnose and fix common issues that you might encounter when working with IAM.

**Topics**

- I am not authorized to perform an action in AWS RoboMaker (p. 105)
- I'm an administrator and want to allow others to access AWS RoboMaker (p. 105)
- I want to understand IAM without becoming an expert (p. 105)

### I am not authorized to perform an action in AWS RoboMaker

If you receive an error in the AWS Management Console that tells you that you're not authorized to perform an action, then you must contact the administrator that provided you with your user name and password.

The following example error occurs when an IAM user named my-user-name tries to use the console to perform the CreateRobotApplication action, but does not have permissions.

```plaintext
User: arn:aws:iam::123456789012:user/my-user-name is not authorized to perform: aws-robomaker:CreateRobotApplication on resource: my-example-robot-application
```

For this example, ask your administrator to update your policies to allow you to access the my-example-robot-application resource using the aws-robomaker:CreateRobotApplication action.

### I'm an administrator and want to allow others to access AWS RoboMaker

To allow others to access AWS RoboMaker you must create an IAM entity (user or role) for the person or application that needs access. They will use the credentials for that entity to access AWS. You must then attach a policy to the entity that grants them the correct permissions in AWS RoboMaker.

To get started right away, see Getting Started with IAM (p. 114).

### I want to understand IAM without becoming an expert

To learn more about IAM terms, concepts, and procedures, see the following pages:
What is Authentication?

Authentication is how you sign in to AWS using your credentials.

Note
To get started quickly, you can ignore this page. First, review the introductory information on Authentication and Access Control for AWS RoboMaker (p. 99) and then see Getting Started with IAM (p. 114).

As a principal, you must be authenticated (signed in to AWS) using an entity (root user, IAM user, or IAM role) to send a request to AWS. An IAM user can have long-term credentials such as a user name and password or a set of access keys. When you assume an IAM role, you are given temporary security credentials.

To authenticate from the AWS Management Console as a user, you must sign in with your user name and password. To authenticate from the AWS CLI or AWS API, you must provide your access key and secret key or temporary credentials. AWS provides SDK and CLI tools to cryptographically sign your request using your credentials. If you don't use AWS tools, you must sign the request yourself. Regardless of the authentication method that you use, you might also be required to provide additional security information. For example, AWS recommends that you use multi-factor authentication (MFA) to increase the security of your account.

As a principal, you can sign in to AWS using the following entities (users or roles):

- AWS account root user – When you first create an AWS account, you begin with a single sign-in identity that has complete access to all AWS services and resources in the account. This identity is called the AWS account root user and is accessed by signing in with the email address and password that you used to create the account. We strongly recommend that you do not use the root user for your everyday tasks, even the administrative ones. Instead, adhere to the best practice of using the root user only to create your first IAM user. Then securely lock away the root user credentials and use them to perform only a few account and service management tasks.

- IAM user – An IAM user is an entity within your AWS account that has specific permissions. AWS RoboMaker supports Signature Version 4, a protocol for authenticating inbound API requests. For more information about authenticating requests, see Signature Version 4 Signing Process in the AWS General Reference.

- IAM role – An IAM role is an IAM identity that you can create in your account that has specific permissions. An IAM role is similar to an IAM user in that it is an AWS identity with permissions policies that determine what the identity can and cannot do in AWS. However, instead of being uniquely associated with one person, a role is intended to be assumable by anyone who needs it. Also, a role does not have standard long-term credentials such as a password or access keys associated with it. Instead, when you assume a role, it provides you with temporary security credentials for your role session. IAM roles with temporary credentials are useful in the following situations:

  - Federated user access – Instead of creating an IAM user, you can use existing identities from AWS Directory Service, your enterprise user directory, or a web identity provider. These are known as federated users. AWS assigns a role to a federated user when access is requested through an identity provider. For more information about federated users, see Federated Users and Roles in the IAM User Guide.

  - Temporary user permissions – An IAM user can assume a role to temporarily take on different permissions for a specific task.

  - Cross-account access – You can use an IAM role to allow a trusted principal in a different account to access resources in your account. Roles are the primary way to grant cross-account access. However,
with some AWS services, you can attach a policy directly to a resource (instead of using a role as a proxy). AWS RoboMaker does not support these resource-based policies. For more information about choosing whether to use a role or a resource-based policy to allow cross-account access, see Controlling Access to Principals in a Different Account (p. 109).

- **AWS service access** – A service role is an IAM role that a service assumes to perform actions in your account on your behalf. When you set up some AWS service environments, you must define a role for the service to assume. This service role must include all the permissions that are required for the service to access the AWS resources that it needs. Service roles vary from service to service, but many allow you to choose your permissions as long as you meet the documented requirements for that service. Service roles provide access only within your account and cannot be used to grant access to services in other accounts. You can create, modify, and delete a service role from within IAM. For example, you can create a role that allows Amazon Redshift to access an Amazon S3 bucket on your behalf and then load data from that bucket into an Amazon Redshift cluster. For more information, see Creating a Role to Delegate Permissions to an AWS Service in the IAM User Guide.

- **Applications running on Amazon EC2** – You can use an IAM role to manage temporary credentials for applications that are running on an EC2 instance and making AWS CLI or AWS API requests. This is preferable to storing access keys within the EC2 instance. To assign an AWS role to an EC2 instance and make it available to all of its applications, you create an instance profile that is attached to the instance. An instance profile contains the role and enables programs that are running on the EC2 instance to get temporary credentials. For more information, see Using an IAM Role to Grant Permissions to Applications Running on Amazon EC2 Instances in the IAM User Guide.

### What is Access Control?

After you sign in (are authenticated) to AWS, your access to AWS resources and operations is controlled using policies. Access control is also known as authorization.

**Note**

To get started quickly, you can ignore this page. First, review the introductory information on Authentication and Access Control for AWS RoboMaker (p. 99) and then see Getting Started with IAM (p. 114).

During authorization, AWS uses values from the request context to check for policies that apply. It then uses the policies to determine whether to allow or deny the request. Most policies are stored in AWS as JSON documents and specify the permissions that are allowed or denied for principals. For more information about the structure and contents of JSON policy documents, see What are Policies? (p. 109).

Policies let an administrator specify who has access to AWS resources, and what actions they can perform on those resources. Every IAM entity (user or role) starts with no permissions. In other words, by default, users can do nothing, not even view their own access keys. To give a user permission to do something, an administrator must attach a permissions policy to a user. Or they can add the user to a group that has the intended permissions. When an administrator then give permissions to a group, all users in that group get those permissions.

You might have valid credentials to authenticate your requests, but unless an administrator grants you permissions you cannot create or access AWS RoboMaker resources. For example, you must have explicit permissions to create an AWS RoboMaker robot application.

As an administrator, you can write a policy to control access to the following:

- **AWS for Principals (p. 108)** – Control what the person making the request (the principal) is allowed to do.
- **IAM Identities (p. 108)** – Control which IAM identities (groups, users, and roles) can be accessed and how.
- **IAM Policies (p. 108)** – Control who can create, edit, and delete customer managed policies, and who can attach and detach all managed policies.
What is Access Control?

- **AWS Resources (p. 108)** – Control who has access to resources using an identity-based policy or a resource-based policy.
- **AWS Accounts (p. 109)** – Control whether a request is allowed only for members of a specific account.

### Controlling Access for Principals

Permissions policies control what you, as a principal, are allowed to do. An administrator must attach an identity-based permissions policy to the identity (user, group, or role) that provides your permissions. Permissions policies allow or deny access to AWS. Administrators can also set a permissions boundary for an IAM entity (user or role) to define the maximum permissions that the entity can have. Permissions boundaries are an advanced IAM feature. For more information about permissions boundaries, see Permissions Boundaries for IAM Identities in the *IAM User Guide*.

For more information and an example of how to control AWS access for principals, see Controlling Access for Principals in the *IAM User Guide*.

### Controlling Access to Identities

Administrators can control what you can do to an IAM identity (user, group, or role) by creating a policy that limits what can be done to an identity, or who can access it. Then attach that policy to the identity that provides your permissions.

For example, an administrator might allow you to reset the password for three specific users. To do this, they attach a policy to your IAM user that allows you to reset the password for only yourself and users with the ARN of the three specified users. This allows you to reset the password of your team members but not other IAM users.

For more information and an example of using a policy to control AWS access to identities, see Controlling Access to Identities in the *IAM User Guide*.

### Controlling Access to Policies

Administrators can control who can create, edit, and delete customer managed policies, and who can attach and detach all managed policies. When you review a policy, you can view the policy summary that includes a summary of the access level for each service within that policy. AWS categorizes each service action into one of four access levels based on what each action does: List, Read, Write, or Permissions management. You can use these access levels to determine which actions to include in your policies. For more information, see Understanding Access Level Summaries Within Policy Summaries in the *IAM User Guide*.

**Warning**

You should limit Permissions Management access level permissions in your account. Otherwise your account members can create policies for themselves with more permissions than they should have. Or they can create separate users with full access to AWS.

For more information and an example for how to control AWS access to policies, see Controlling Access to Policies in the *IAM User Guide*.

### Controlling Access to Resources

Administrators can control access to resources using an identity-based policy or a resource-based policy. In an identity-based policy, you attach the policy to an identity and specify what resources that identity can access. In a resource-based policy, you attach a policy to the resource that you want to control. In the policy, you specify which principals can access that resource.

For more information, see Controlling Access to Resources in the *IAM User Guide*.
Resource Creators Do Not Automatically Have Permissions

All resources in an account are owned by the account, regardless of who created those resources. The AWS account root user is the account owner, and therefore has permission to perform any action on any resource in the account.

**Important**

We strongly recommend that you do not use the root user for your everyday tasks, even the administrative ones. Instead, adhere to the best practice of using the root user only to create your first IAM user. Then securely lock away the root user credentials and use them to perform only a few account and service management tasks. To view the tasks that require you to sign in as the root user, see AWS Tasks That Require Root User.

Entities (users or roles) in your account must be granted access to create a resource. But just because they create a resource does not mean they automatically have full access to that resource. You must explicitly grant permissions for each action. Additionally, you can revoke those permissions at any time, as long as you have access to manage user and role permissions.

Controlling Access to Principals in a Different Account

Administrators can use AWS resource-based policies, IAM cross-account roles, or the AWS Organizations service to allow principals in another account to access resources in your account.

For some AWS services, you can grant cross-account access to your resources. To do this, you attach a policy directly to the resource that you want to share, instead of using a role as a proxy. If the service supports this policy type, then the resource that you want to share must also support resource-based policies. Unlike a user-based policy, a resource-based policy specifies who (in the form of a list of AWS account ID numbers) can access that resource. AWS RoboMaker does not support resource-based policies.

Cross-account access with a resource-based policy has some advantages over a role. With a resource that is accessed through a resource-based policy, the principal (person or application) still works in the trusted account and does not have to give up his or her user permissions in place of the role permissions. In other words, the principal has access to resources in the trusted account and in the trusting account at the same time. This is useful for tasks such as copying information from one account to another. For more information about using cross-account roles, see Providing Access to an IAM User in Another AWS Account That You Own in the IAM User Guide.

AWS Organizations offers policy-based management for multiple AWS accounts that you own. With Organizations, you can create groups of accounts, automate account creation, apply and manage policies for those groups. Organizations enables you to centrally manage policies across multiple accounts, without requiring custom scripts and manual processes. Using AWS Organizations, you can create Service Control Policies (SCPs) that centrally control AWS service use across AWS accounts. For more information, see What Is AWS Organizations? in the AWS Organizations User Guide.

What are Policies?

You control access in AWS by creating policies and attaching them to IAM identities or AWS resources.

**Note**

To get started quickly, you can ignore this page. First, review the introductory information on Authentication and Access Control for AWS RoboMaker (p. 99) and then see Getting Started with IAM (p. 114).

A policy is an object in AWS that, when associated with an entity or resource, defines their permissions. AWS evaluates these policies when a principal, such as a user, makes a request. Permissions in the policies determine whether the request is allowed or denied. Most policies are stored in AWS as JSON documents.
IAM policies define permissions for an action regardless of the method that you use to perform the operation. For example, if a policy allows the GetUser action, then a user with that policy can get user information from the AWS Management Console, the AWS CLI, or the AWS API. When you create an IAM user, you can set up the user to allow console or programmatic access. The IAM user can sign in to the console using a user name and password. Or they can use access keys to work with the CLI or API.

The following policy types, listed in order of frequency, can affect whether a request is authorized. For more details, see Policy Types in the IAM User Guide.

- **Identity-based policies** – You can attach managed and inline policies to IAM identities (users, groups to which users belong, and roles).
- **Resource-based policies** – You can attach inline policies to resources in some AWS services. The most common examples of resource-based policies are Amazon S3 bucket policies and IAM role trust policies. AWS RoboMaker does not support resource-based policies.
- **Organizations SCPs** – You can use an AWS Organizations service control policy (SCP) to apply a permissions boundary to an AWS Organizations organization or organizational unit (OU). Those permissions are applied to all entities within the member accounts.
- **Access control lists (ACLs)** – You can use ACLs to control what principals can access a resource. ACLs are similar to resource-based policies, although they are the only policy type that does not use the JSON policy document structure. AWS RoboMaker does not support ACLs.

These policies types can be categorized as permissions policies or permissions boundaries.

- **Permissions policies** – You can attach permissions policies to a resource in AWS to define the permissions for that object. Within a single account, AWS evaluates all permissions policies together. Permissions policies are the most common policies. You can use the following policy types as permissions policies:
  - **Identity-based policies** – When you attach a managed or inline policy to an IAM user, group, or role, the policy defines the permissions for that entity.
  - **Resource-based policies** – When you attach a JSON policy document to a resource, you define the permissions for that resource. The service must support resource-based policies.
  - **Access control lists (ACLs)** – When you attach an ACL to a resource, you define a list of principals with permission to access that resource. The resource must support ACLs.

- **Permissions boundaries** – You can use policies to define the permissions boundary for an entity (user or role). A permissions boundary controls the maximum permissions that an entity can have. Permissions boundaries are an advanced AWS feature. When more than one permissions boundaries applies to a request, AWS evaluates each permissions boundary separately. You can apply a permissions boundary in the following situations:
  - **Organizations** – You can use an AWS Organizations service control policy (SCP) to apply a permissions boundary to an AWS Organizations organization or organizational unit (OU).
  - **IAM users or roles** – You can use a managed policy for a user or role's permissions boundary. For more information, see Permissions Boundaries for IAM Entities in the IAM User Guide.

**Topics**
- Identity-Based Policies (p. 110)
- Resource-Based Policies (p. 111)
- Policy Access Level Classifications (p. 111)

**Identity-Based Policies**

You can attach policies to IAM identities. For example, you can do the following:
What are Policies?

- **Attach a permissions policy to a user or a group in your account** – To grant a user permissions to create an AWS RoboMaker resource, such as a robot applications, you can attach a permissions policy to a user or a group to which the user belongs.

- **Attach a permissions policy to a role (grant cross-account permissions)** – You can attach an identity-based permissions policy to an IAM role to grant cross-account permissions. For example, the administrator in account A can create a role to grant cross-account permissions to another AWS account (for example, account B) or an AWS service as follows:
  1. Account A administrator creates an IAM role and attaches a permissions policy to the role that grants permissions on resources in account A.
  2. Account A administrator attaches a trust policy to the role identifying account B as the principal who can assume the role.
  3. Account B administrator can then delegate permissions to assume the role to any users in account B. Doing this allows users in account B to create or access resources in account A. The principal in the trust policy can also be an AWS service principal if you want to grant an AWS service permissions to assume the role.

For more information about using IAM to delegate permissions, see [Access Management](#) in the IAM User Guide.

For more information about users, groups, roles, and permissions, see [Identities (Users, Groups, and Roles)](#) in the IAM User Guide.

### Resource-Based Policies

Resource-based policies are JSON policy documents that you attach to a resource. These policies allow you to specify what actions a specified principal can perform on that resource and under what conditions. The most commonly-known resource-based policy is an Amazon S3 bucket. Resource-based policies are inline policies that exist only on the resource. There are no managed resource-based policies.

Granting permissions to members of other AWS accounts using a resource-based policy has some advantages over an IAM role. For more information, see [How IAM Roles Differ from Resource-based Policies](#) in the IAM User Guide.

AWS RoboMaker does not support resource-based policies.

### Policy Access Level Classifications

In the IAM console, actions are grouped using the following access level classifications:

- **List** – Provide permission to list resources within the service to determine whether an object exists. Actions with this level of access can list objects but cannot see the contents of a resource. Most actions with the List access level cannot be performed on a specific resource. When you create a policy statement with these actions, you must specify All resources ("*").

- **Read** – Provide permission to read but not edit the contents and attributes of resources in the service. For example, the Amazon S3 actions GetObject and GetBucketLocation have the Read access level.

- **Write** – Provide permission to create, delete, or modify resources in the service. For example, the Amazon S3 actions CreateBucket, DeleteBucket and PutObject have the Write access level.

- **Permissions management** – Provide permission to grant or modify resource permissions in the service. For example, most IAM and AWS Organizations policy actions have the Permissions management access level.

  **Tip**
  To improve the security of your AWS account, restrict or regularly monitor policies that include the Permissions management access level classification.
• **Tagging** – Provide permission to create, delete, or modify tags that are attached to a resource in the service. For example, the Amazon EC2 CreateTags and DeleteTags actions have the **Tagging** access level.

**Using Service-Linked Roles for AWS RoboMaker**

AWS RoboMaker uses AWS Identity and Access Management (IAM) service-linked roles. A service-linked role is a unique type of IAM role that is linked directly to AWS RoboMaker. Service-linked roles are predefined by AWS RoboMaker and include all the permissions that the service requires to call other AWS services on your behalf.

A service-linked role makes setting up AWS RoboMaker easier because you don't have to manually add the necessary permissions. AWS RoboMaker defines the permissions of its service-linked roles, and unless defined otherwise, only AWS RoboMaker can assume its roles. The defined permissions include the trust policy and the permissions policy, and that permissions policy cannot be attached to any other IAM entity.

You can delete a service-linked role only after first deleting their related resources. This protects your AWS RoboMaker resources because you can't inadvertently remove permission to access the resources.

For information about other services that support service-linked roles, see AWS Services That Work with IAM and look for the services that have Yes in the Service-Linked Role column. Choose a Yes with a link to view the service-linked role documentation for that service.

**Service-Linked Role Permissions for AWS RoboMaker**

AWS RoboMaker uses the service-linked role named **AWSServiceRoleForRoboMaker** – Allows RoboMaker to access EC2, Greengrass, and Lambda resources on your behalf.

The AWSServiceRoleForRoboMaker service-linked role trusts the following services to assume the role:

- robomaker.amazonaws.com

The role permissions policy allows AWS RoboMaker to complete the following actions on the specified resources:

- Create and cancel a simulation job created as part of a simulation job batch
- Manage Amazon EC2 networking resources
- Manage AWS IoT Greengrass deployments

You must configure permissions to allow an IAM entity (such as a user, group, or role) to create, edit, or delete a service-linked role. For more information, see Service-Linked Role Permissions in the IAM User Guide.

**Creating a Service-Linked Role for AWS RoboMaker**

You don't need to manually create a service-linked role. When you SimulationJob or DeploymentJob in the AWS Management Console, the AWS CLI, or the AWS API, AWS RoboMaker creates the service-linked role for you.

If you delete this service-linked role, and then need to create it again, you can use the same process to recreate the role in your account. When you create a SimulationJob, SimulationJobBatch, or DeploymentJob, AWS RoboMaker creates the service-linked role for you again.

You can also use the IAM console to create a service-linked role with the RoboMaker use case. In the AWS CLI or the AWS API, create a service-linked role with the robomaker.amazonaws.com service
name. For more information, see Creating a Service-Linked Role in the IAM User Guide. If you delete this service-linked role, you can use this same process to create the role again.

**Editing a Service-Linked Role for AWS RoboMaker**

AWS RoboMaker does not allow you to edit the AWSServiceRoleForRoboMaker service-linked role. After you create a service-linked role, you cannot change the name of the role because various entities might reference the role. However, you can edit the description of the role using IAM. For more information, see Editing a Service-Linked Role in the IAM User Guide.

**Deleting a Service-Linked Role for AWS RoboMaker**

If you no longer need to use a feature or service that requires a service-linked role, we recommend that you delete that role. That way you don’t have an unused entity that is not actively monitored or maintained. However, you must clean up the resources for your service-linked role before you can manually delete it.

**Note**
If the AWS RoboMaker service is using the role when you try to delete the resources, then the deletion might fail. If that happens, wait for a few minutes and try the operation again.

**To manually delete the service-linked role using IAM**

Use the IAM console, the AWS CLI, or the AWS API to delete the AWSServiceRoleForRoboMaker service-linked role. For more information, see Deleting a Service-Linked Role in the IAM User Guide.

**Supported Regions for AWS RoboMaker Service-Linked Roles**

AWS RoboMaker supports using service-linked roles in all of the regions where the service is available. For more information, see AWS Regions and Endpoints.

AWS RoboMaker does not support using service-linked roles in every region where the service is available. You can use the AWSServiceRoleForRoboMaker role in the following regions.

<table>
<thead>
<tr>
<th>Region name</th>
<th>Region identity</th>
<th>Support in AWS RoboMaker</th>
</tr>
</thead>
<tbody>
<tr>
<td>US East (N. Virginia)</td>
<td>us-east-1</td>
<td>Yes</td>
</tr>
<tr>
<td>US East (Ohio)</td>
<td>us-east-2</td>
<td>Yes</td>
</tr>
<tr>
<td>US West (N. California)</td>
<td>us-west-1</td>
<td>Yes</td>
</tr>
<tr>
<td>US West (Oregon)</td>
<td>us-west-2</td>
<td>Yes</td>
</tr>
<tr>
<td>Asia Pacific (Mumbai)</td>
<td>ap-south-1</td>
<td>Yes</td>
</tr>
<tr>
<td>Asia Pacific (Osaka-Local)</td>
<td>ap-northeast-3</td>
<td>Yes</td>
</tr>
<tr>
<td>Asia Pacific (Seoul)</td>
<td>ap-northeast-2</td>
<td>Yes</td>
</tr>
<tr>
<td>Asia Pacific (Singapore)</td>
<td>ap-southeast-1</td>
<td>Yes</td>
</tr>
<tr>
<td>Asia Pacific (Sydney)</td>
<td>ap-southeast-2</td>
<td>Yes</td>
</tr>
<tr>
<td>Asia Pacific (Tokyo)</td>
<td>ap-northeast-1</td>
<td>Yes</td>
</tr>
<tr>
<td>Canada (Central)</td>
<td>ca-central-1</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Getting Started with IAM

AWS Identity and Access Management (IAM) is an AWS service that allows you to securely manage access to services and resources. IAM is included with your AWS account at no additional charge.

**Note**
Before you start with IAM, review the introductory information on Authentication and Access Control for AWS RoboMaker (p. 99).

When you first create an AWS account, you begin with a single sign-in identity that has complete access to all AWS services and resources in the account. This identity is called the AWS account root user and is accessed by signing in with the email address and password that you used to create the account. We strongly recommend that you do not use the root user for your everyday tasks, even the administrative ones. Instead, adhere to the best practice of using the root user only to create your first IAM user. Then, securely lock away the root user credentials and use them to perform only a few account and service management tasks.

Create your IAM Admin User

To create an administrator user for yourself and add the user to an administrators group (console)

1. Use your AWS account email address and password to sign in as the AWS account root user to the IAM console at https://console.aws.amazon.com/iam/.

   **Note**
   We strongly recommend that you adhere to the best practice of using the Administrator IAM user below and securely lock away the root user credentials. Sign in as the root user only to perform a few account and service management tasks.

2. In the navigation pane, choose Users and then choose Add user.

3. For **User name**, enter Administrator.

4. Select the check box next to **AWS Management Console access**. Then select **Custom password**, and then enter your new password in the text box.

5. (Optional) By default, AWS requires the new user to create a new password when first signing in. You can clear the check box next to **User must create a new password at next sign-in** to allow the new user to reset their password after they sign in.

6. Choose **Next: Permissions**.

7. Under **Set permissions**, choose **Add user to group**.

8. Choose **Create group**.

9. In the **Create group** dialog box, for **Group name** enter **Administrators**.

---

<table>
<thead>
<tr>
<th>Region name</th>
<th>Region identity</th>
<th>Support in AWS RoboMaker</th>
</tr>
</thead>
<tbody>
<tr>
<td>Europe (Frankfurt)</td>
<td>eu-central-1</td>
<td>Yes</td>
</tr>
<tr>
<td>Europe (Ireland)</td>
<td>eu-west-1</td>
<td>Yes</td>
</tr>
<tr>
<td>Europe (London)</td>
<td>eu-west-2</td>
<td>Yes</td>
</tr>
<tr>
<td>Europe (Paris)</td>
<td>eu-west-3</td>
<td>Yes</td>
</tr>
<tr>
<td>South America (São Paulo)</td>
<td>sa-east-1</td>
<td>Yes</td>
</tr>
<tr>
<td>AWS GovCloud (US)</td>
<td>us-gov-west-1</td>
<td>No</td>
</tr>
</tbody>
</table>
10. Choose Filter policies, and then select AWS managed -job function to filter the table contents.
11. In the policy list, select the check box for AdministratorAccess. Then choose Create group.

   **Note**
   You must activate IAM user and role access to Billing before you can use the AdministratorAccess permissions to access the AWS Billing and Cost Management console. To do this, follow the instructions in step 1 of the tutorial about delegating access to the billing console.

12. Back in the list of groups, select the check box for your new group. Choose Refresh if necessary to see the group in the list.
13. Choose Next: Tags.
14. (Optional) Add metadata to the user by attaching tags as key-value pairs. For more information about using tags in IAM, see Tagging IAM Entities in the IAM User Guide.
15. Choose Next: Review to see the list of group memberships to be added to the new user. When you are ready to proceed, choose Create user.

You can use this same process to create more groups and users and to give your users access to your AWS account resources. To learn about using policies that restrict user permissions to specific AWS resources, see Access Management and Example Policies.

### Create Delegated Users for AWS RoboMaker

To support multiple users in your AWS account, you must delegate permission to allow other people to perform only the actions you want to allow. To do this, create an IAM group with the permissions those people need and then add IAM users to the necessary groups as you create them. You can use this process to set up the groups, users, and permissions for your entire AWS account. This solution is best used by small and medium organizations where an AWS administrator can manually manage the users and groups. For large organizations, you can use custom IAM roles, federation, or single sign-on.

In the following task, you will create three users named arnav, carlos, and martha and attach a policy that grants permission to create a robot application named my-example-robot-application, but only within the next 30 days. You can use the steps provided here to add users with different permissions.

#### To create a delegated user for someone else (console)

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation pane, choose Users and then choose Add user.
3. For User name, enter arnav.
4. Choose Add another user and enter carlos for the second user. Then choose Add another user and enter martha for the third user.
5. Select the check box next to AWS Management Console access and select Autogenerated password.
6. Clear the check box next to User must create a new password at next sign-in to allow the new user to reset their password after they sign in.
7. Choose Next: Permissions.
8. Choose Attach existing policies directly. You will create a new managed policy for the users.
9. Choose Create policy.

   The Create policy wizard opens in a new tab or browser window.
10. On the Visual editor tab, choose Choose a service. Then choose AWS RoboMaker. You can use the search box at the top to limit the results in the list of services.
The Service section closes and the Actions section opens automatically.

11. Choose the AWS RoboMaker actions that you want to allow. For example, to grants permission to create a robot application, enter `CreateRobotApplication` in the Filter actions text box. When the list of AWS RoboMaker actions is filtered, choose the check box next to `CreateRobotApplication`.

The AWS RoboMaker actions are grouped by access level classification to make it easy for you to quickly determine the level of access that each action provides. For more information, see Policy Access Level Classifications (p. 111).

12. If the actions that you selected in the previous steps do not support choosing specific resources, then All resources is selected for you. In that case, you cannot edit this section.

If you chose one or more actions that support resource-level permissions, then the visual editor lists those resource types in the Resources section. Choose You chose actions that require the robot application resource type to choose whether you want to enter a specific robot application for your policy.

13. If you want to allow the `CreateRobotApplication` action for all resources, choose All resources.

If you want to specify a resource, choose Add ARN. Specify the region and account ID (or account ID) (or choose Any), and then enter `my-example-robot-application` for the resource. Then choose Add.

14. Choose Specify request conditions (optional).

15. Choose Add condition to grants permission to create a robot application within the next 7 days. Assume that today's date is January 1, 2019.

16. For Condition Key, choose `aws:CurrentTime`. This condition key checks the date and time that the user makes the request. It returns true (and therefore allows the `CreateRobotApplication` action only if the date and time are within the specified range.

17. For Qualifier, leave the default value.

18. To specify the start of the allowed date and time range, for Operator, choose DateGreaterThan. Then for Value, enter `2019-01-01T00:00:00Z`.

19. Choose Add to save your condition.

20. Choose Add another condition to specify the end date.

21. Follow similar steps to specify the end of the allowed date and time range. For Condition Key, choose `aws:CurrentTime`. For Operator, choose DateLessThan. For Value, enter `2019-01-06T23:59:59Z`, seven days after the first date. Then choose Add to save your condition.

22. (Optional) To see the JSON policy document for the policy you are creating, choose the JSON tab. You can switch between the Visual editor and JSON tabs any time. However, if you make changes or choose Review policy in the Visual editor tab, IAM might restructure your policy to optimize it for the visual editor. For more information, see Policy Restructuring in the IAM User Guide.

23. When you are finished, choose Review policy.

24. On the Review policy page, for Name, enter `CreateRobotApplicationPolicy` and for the Description, enter Policy to grants permission to create a robot application. Review the policy summary to make sure that you have granted the intended permissions, and then choose Create policy to save your new policy.

25. Return to the original tab or window, and refresh your list of policies.

26. In the search box, enter `CreateRobotApplicationPolicy`. Select the check box next to your new policy. Then choose Next Step.

27. Choose Next: Review to preview your new users. When you are ready to proceed, choose Create users.

28. Download or copy the passwords for your new users and deliver them to the users securely. Separately, provide your users with a link to your IAM user console page and the user names you just created.
Allow Users to Self-Manage Their Credentials

You must have physical access to the hardware that will host the user's virtual MFA device in order to configure MFA. For example, you might configure MFA for a user who will use a virtual MFA device running on a smartphone. In that case, you must have the smartphone available in order to finish the wizard. Because of this, you might want to let users configure and manage their own virtual MFA devices. In that case, you must grant users the permissions to perform the necessary IAM actions.

To create a policy to allow credential self-management (console)

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation pane, choose Policies, and then choose Create policy.
3. Choose the JSON tab and copy the text from the following JSON policy document. Paste this text into the JSON text box.

   Important
   This example policy does not allow users to reset their password while signing in. New users and users with an expired password might try to do so. You can allow this by adding iam:ChangePassword and iam:CreateLoginProfile to the statement BlockMostAccessUnlessSignedInWithMFA. However, IAM does not recommend this.

   ```json
   { 
   "Version": "2012-10-17",
   "Statement": [
     {
       "Sid": "AllowAllUsersToListAccounts",
       "Effect": "Allow",
       "Action": [
         "iam:ListAccountAliases",
         "iam:ListUsers",
         "iam:ListVirtualMFADevices",
         "iam:GetAccountPasswordPolicy",
         "iam:GetAccountSummary"
       ],
       "Resource": "*"
     },
     {
       "Sid": "AllowIndividualUserToSeeAndManageOnlyTheirOwnAccountInformation",
       "Effect": "Allow",
       "Action": [
         "iam:ChangePassword",
         "iam:CreateAccessKey",
         "iam:CreateLoginProfile",
         "iam:DeleteAccessKey",
         "iam:DeleteLoginProfile",
         "iam:GetLoginProfile",
         "iam:ListAccessKeys",
         "iam:UpdateAccessKey",
         "iam:UpdateLoginProfile",
         "iam:ListSigningCertificates",
         "iam:DeleteSigningCertificate",
         "iam:UpdateSigningCertificate",
         "iam:UploadSigningCertificate",
         "iam:ListSSHPublicKeys",
         "iam:GetSSHPublicKey",
         "iam:DeleteSSHPublicKey",
         "iam:UpdateSSHPublicKey",
         "iam:UploadSSHPublicKey"
       ],
       "Resource": "arn:aws:iam::*:user/${aws:username}"
     }
   ]
   }
   ```
What does this policy do?

- The `AllowAllUsersToListAccounts` statement enables the user to see basic information about the account and its users in the IAM console. These permissions must be in their own
statement because they do not support or do not need to specify a specific resource ARN, and instead specify "Resource" : "*".

- The AllowIndividualUserToSeeAndManageOnlyTheirOwnAccountInformation statement enables the user to manage his or her own user, password, access keys, signing certificates, SSH public keys, and MFA information in the IAM console. It also allows users to sign in for the first time in an administrator requires them to set a first-time password. The resource ARN limits the use of these permissions to only the user’s own IAM user entity.

- The AllowIndividualUserToViewAndManageTheirOwnMFA statement enables the user to view or manage his or her own MFA device. Notice that the resource ARNs in this statement allow access to only an MFA device or user that has the same name as the currently signed-in user. Users can’t create or alter any MFA device other than their own.

- The AllowIndividualUserToDeactivateOnlyTheirOwnMFAOnlyWhenUsingMFA statement allows the user to deactivate only his or her own MFA device, and only if the user signed in using MFA. This prevents others with only the access keys (and not the MFA device) from deactivating the MFA device and accessing the account.

- The BlockMostAccessUnlessSignedInWithMFA statement uses a combination of "Deny" and "NotAction" to deny access to all but a few actions in IAM and other AWS services if the user is not signed-in with MFA. For more information about the logic for this statement, see NotAction with Deny in the IAM User Guide. If the user is signed-in with MFA, then the "Condition" test fails and the final "deny" statement has no effect and other policies or statements for the user determine the user’s permissions. This statement ensures that when the user is not signed-in with MFA, they can perform only the listed actions and only if another statement or policy allows access to those actions.

The ...IfExists version of the Bool operator ensures that if the aws:MultiFactorAuthPresent key is missing, the condition returns true. This means that a user accessing an API with long-term credentials, such as an access key, is denied access to the non-IAM API operations.

4. When you are finished, choose Review policy.

5. On the Review page, type Force_MFA for the policy name. For the policy description, type This policy allows users to manage their own passwords and MFA devices but nothing else unless they authenticate with MFA. Review the policy Summary to see the permissions granted by your policy, and then choose Create policy to save your work.

The new policy appears in the list of managed policies and is ready to attach.

To attach the policy to a user (console)

1. In the navigation pane, choose Users.
2. Choose the name (not the check box) of the user you want to edit.
3. On the Permissions tab, and choose Add permissions.
4. Choose Attach existing policies directly.
5. In the search box, enter Force, and then select the check box next to Force_MFA in the list. Then choose Next: Review.
6. Review your changes and choose Add permissions.

Enable MFA for Your IAM User

For increased security, we recommend that all IAM users configure multi-factor authentication (MFA) to help protect your AWS RoboMaker resources. MFA adds extra security because it requires users to provide unique authentication from an AWS-supported MFA device in addition to their regular sign-in credentials. The most secure AWS MFA device is the U2F security key. If your company already has U2F
If you don’t already have a U2F device, you can get started quickly and at a low cost by enabling a virtual MFA device. This requires that you install a software app on an existing phone or other mobile device. The device generates a six-digit numeric code based upon a time-synchronized one-time password algorithm. When the user signs in to AWS, they are prompted to enter a code from the device. Each virtual MFA device assigned to a user must be unique. A user cannot enter a code from another user’s virtual MFA device to authenticate. For a list of a few supported apps that you can use as virtual MFA devices, see Multi-Factor Authentication.

**Note**
You must have physical access to the mobile device that will host the user’s virtual MFA device in order to configure MFA for an IAM user.

**To enable a virtual MFA device for an IAM user (console)**

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation pane, choose Users.
3. In the User Name list, choose the name of the intended MFA user.
5. In the Manage MFA Device wizard, choose Virtual MFA device, and then choose Continue.

IAM generates and displays configuration information for the virtual MFA device, including a QR code graphic. The graphic is a representation of the “secret configuration key” that is available for manual entry on devices that do not support QR codes.

6. Open your virtual MFA app.

For a list of apps that you can use for hosting virtual MFA devices, see Multi-Factor Authentication. If the virtual MFA app supports multiple accounts (multiple virtual MFA devices), choose the option to create a new account (a new virtual MFA device).

7. Determine whether the MFA app supports QR codes, and then do one of the following:

   • From the wizard, choose Show QR code, and then use the app to scan the QR code. For example, you might choose the camera icon or choose an option similar to Scan code, and then use the device’s camera to scan the code.

   • In the Manage MFA Device wizard, choose Show secret key, and then enter the secret key into your MFA app.

When you are finished, the virtual MFA device starts generating one-time passwords.

8. In the Manage MFA Device wizard, in the MFA code 1 box, enter the one-time password that currently appears in the virtual MFA device. Wait up to 30 seconds for the device to generate a new one-time password. Then enter the second one-time password into the MFA code 2 box. Choose Assign MFA.

**Important**
Submit your request immediately after generating the codes. If you generate the codes and then wait too long to submit the request, the MFA device successfully associates with the user but the MFA device is out of sync. This happens because time-based one-time passwords (TOTP) expire after a short period of time. If this happens, you can resync the device. For more information, see Resynchronizing Virtual and Hardware MFA Devices in the IAM User Guide.

The virtual MFA device is now ready for use with AWS.
Logging and Monitoring in AWS RoboMaker

Monitoring is an important part of maintaining the reliability, availability, and performance of AWS RoboMaker and your AWS solutions. You should collect monitoring data from all of the parts of your AWS solution so that you can more easily debug a multi-point failure if one occurs.

Topics
- Monitoring AWS RoboMaker with Amazon CloudWatch (p. 121)
- Logging AWS RoboMaker API Calls with AWS CloudTrail (p. 122)

Monitoring AWS RoboMaker with Amazon CloudWatch

You can monitor AWS RoboMaker simulation jobs using Amazon CloudWatch, which collects information from your simulation job and creates readable, near real-time metrics. Information is provided at 1-minute frequency.

Metrics exist only in the region in which they are created. Metrics cannot be deleted, but they automatically expire after 15 months if no new data is published to them.

For more information about Amazon CloudWatch, see the Amazon CloudWatch User Guide. For more information about pricing, see Amazon CloudWatch Pricing.

AWS RoboMaker Metrics

The following metrics are available in the SimulationJobId dimension.

<table>
<thead>
<tr>
<th>Metric</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RealTimeFactor</td>
<td>The ratio of the amount of time that was simulated versus wall clock time.</td>
</tr>
<tr>
<td></td>
<td>For example, if it takes an hour to simulate 30 minutes, the factor is .5.</td>
</tr>
<tr>
<td></td>
<td>More complex simulations have a lower real time factor.</td>
</tr>
<tr>
<td>vCPU*</td>
<td>Number of virtual CPU cores used by the simulation job</td>
</tr>
<tr>
<td></td>
<td>Unit: Count</td>
</tr>
<tr>
<td>Memory*</td>
<td>Amount of Memory, in GB, used by the SimulationJob</td>
</tr>
<tr>
<td></td>
<td>Unit: GB</td>
</tr>
<tr>
<td>SimulationUnit*</td>
<td>SimulationUnit is calculated based on vCPU and memory consumption of the</td>
</tr>
<tr>
<td></td>
<td>Simulation Job</td>
</tr>
<tr>
<td></td>
<td>Unit: Count</td>
</tr>
</tbody>
</table>

121
Logging AWS RoboMaker API Calls with AWS CloudTrail

AWS RoboMaker is integrated with AWS CloudTrail, a service that provides a record of actions taken by a user, role, or an AWS service in AWS RoboMaker. CloudTrail captures all API calls for AWS RoboMaker as events. The calls captured include calls from the AWS RoboMaker console and code calls to the AWS RoboMaker API operations. If you create a trail, you can enable continuous delivery of CloudTrail events to an Amazon S3 bucket, including events for AWS RoboMaker. If you don't configure a trail, you can still view the most recent events in the CloudTrail console in Event history. Using the information collected by CloudTrail, you can determine the request that was made to AWS RoboMaker, the IP address from which the request was made, who made the request, when it was made, and additional details.

To learn more about CloudTrail, see the AWS CloudTrail User Guide.

AWS RoboMaker Information in CloudTrail

CloudTrail is enabled on your AWS account when you create the account. When activity occurs in AWS RoboMaker, that activity is recorded in a CloudTrail event along with other AWS service events in Event history. You can view, search, and download recent events in your AWS account. For more information, see Viewing Events with CloudTrail Event History.

For an ongoing record of events in your AWS account, including events for AWS RoboMaker, create a trail. A trail enables CloudTrail to deliver log files to an Amazon S3 bucket. By default, when you create a trail in the console, the trail applies to all AWS Regions. The trail logs events from all Regions in the AWS partition and delivers the log files to the Amazon S3 bucket that you specify. Additionally, you can configure other AWS services to further analyze and act upon the event data collected in CloudTrail logs. For more information, see the following:

- Overview for Creating a Trail
- CloudTrail Supported Services and Integrations
- Configuring Amazon SNS Notifications for CloudTrail
- Receiving CloudTrail Log Files from Multiple Regions and Receiving CloudTrail Log Files from Multiple Accounts

All AWS RoboMaker actions are logged by CloudTrail and are documented in the AWS RoboMaker API Reference. For example, calls to the CreateSimulationJob, RegisterRobot and UpdateRobotApplication actions generate entries in the CloudTrail log files.

Every event or log entry contains information about who generated the request. The identity information helps you determine the following:

- Whether the request was made with root or AWS Identity and Access Management (IAM) user credentials.
- Whether the request was made with temporary security credentials for a role or federated user.
- Whether the request was made by another AWS service.

For more information, see the CloudTrail userIdentity Element.
Understanding AWS RoboMaker Log File Entries

A trail is a configuration that enables delivery of events as log files to an Amazon S3 bucket that you specify. CloudTrail log files contain one or more log entries. An event represents a single request from any source and includes information about the requested action, the date and time of the action, request parameters, and so on. CloudTrail log files aren't an ordered stack trace of the public API calls, so they don't appear in any specific order.

The following example shows a CloudTrail log entry that demonstrates the DescribeRobot action.

```json
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "my-principal-id",
    "arn": "my-arn",
    "accountId": "my-account-id",
    "accessKeyId": "my-access-key",
    "userName": "my-user-name"
  },
  "eventTime": "2018-12-07T00:28:03Z",
  "eventSource": "robomaker.amazonaws.com",
  "eventName": "DescribeRobot",
  "awsRegion": "us-west-2",
  "sourceIPAddress": "my-ip-address",
  "userAgent": "aws-internal/3 aws-sdk-java/1.11.455 Linux/4.4.83-0.1.fm.327.54.326.metall1.x86_64 OpenJDK_64-Bit_Server_VM/25.192-b12 java/1.8.0_192",
  "requestParameters": {
    "robot": "my-robot-arn"
  },
  "responseElements": null,
  "requestID": "f54cdf8b-f9b6-11e8-8883-c3f04579eca3",
  "eventID": "affb0303-ff48-4f65-af8e-d7d19710bac3",
  "readOnly": true,
  "eventType": "AwsApiCall",
  "recipientAccountArn": "my-recipient-account-id"
}
```

Security Compliance

The AWS HIPAA Compliance program includes AWS RoboMaker as a HIPAA Eligible Service. The AWS PCI DSS Compliance program includes AWS RoboMaker as a PCI-compliant service.

For general information about AWS Cloud and HIPAA compliance, see the following:

- HIPAA Compliance
- Architecting for HIPAA Security and Compliance on Amazon Web Services

Resilience in AWS RoboMaker

The AWS global infrastructure is built around AWS Regions and Availability Zones. AWS Regions provide multiple physically separated and isolated Availability Zones, which are connected with low-latency, high-throughput, and highly redundant networking. With Availability Zones, you can design and operate applications and databases that automatically fail over between zones without interruption. Availability
Zones are more highly available, fault tolerant, and scalable than traditional single or multiple data center infrastructures.

For more information about AWS Regions and Availability Zones, see AWS Global Infrastructure.

In addition to the AWS global infrastructure, AWS RoboMaker offers several features to help support your data resiliency and backup needs.

**Infrastructure Security in AWS RoboMaker**

As a managed service, AWS RoboMaker is protected by the AWS global network security procedures that are described in the Amazon Web Services: Overview of Security Processes whitepaper.

You use AWS published API calls to access AWS RoboMaker through the network. Clients must support Transport Layer Security (TLS) 1.0 or later. We recommend TLS 1.2 or later. Clients must also support cipher suites with perfect forward secrecy (PFS) such as Ephemeral Diffie-Hellman (DHE) or Elliptic Curve Ephemeral Diffie-Hellman (ECDHE). Most modern systems such as Java 7 and later support these modes.

Additionally, requests must be signed by using an access key ID and a secret access key that is associated with an IAM principal. Or you can use the AWS Security Token Service (AWS STS) to generate temporary security credentials to sign requests.
Tagging Your AWS RoboMaker Resources

To help you manage and organize your fleets, robots, robot applications, simulation applications, simulation jobs and deployments you can optionally assign your own metadata to each of these resources in the form of tags. This section describes tags and shows you how to create them.

Tag Basics

Tags enable you to categorize your AWS RoboMaker resources in different ways, for example, by purpose, owner, or environment. This is useful when you have many resources of the same type — you can quickly identify a specific resource based on the tags you've assigned to it. Each tag consists of a key and optional value, both of which you define. For example, you could define a set of tags for your robots that helps you track devices by function. We recommend that you create a set of tag keys that meets your needs for each kind of resource. Using a consistent set of tag keys makes it easier for you to manage your resources.

You can search for and filter resources based on the tags you add or apply. You can also use tags to control access to your resources as described in Using Tags with IAM Policies (p. ___).

For ease of use, the Tag Editor in the AWS Management Console provides a central, unified way to create and manage your tags. For more information, see Working with Tag Editor in Working with the AWS Management Console.

You can also work with tags using the AWS CLI and the AWS RoboMaker API. You can associate tags with thing groups, thing types, topic rules, jobs, security profiles, and billing groups when you create them by using the "Tags" field in the following commands:

- CreateDeploymentJob
- CreateFleet
- CreateRobot
- CreateRobotApplication
- CreateSimulationApplication
- CreateSimulationJob
- StartSimulationJobBatch

You can add, modify, or delete tags for existing resources that support tagging by using the following commands:

- TagResource
- ListTagsForResource
- UntagResource

You can edit tag keys and values, and you can remove tags from a resource at any time. You can set the value of a tag to an empty string, but you can't set the value of a tag to null. If you add a tag that has the same key as an existing tag on that resource, the new value overwrites the old value. If you delete a resource, any tags associated with the resource are also deleted.
Additional information is available in AWS Tagging Strategies.

Tag Restrictions and Limitations

The following basic restrictions apply to tags:

- Maximum number of tags per resource — 50
- Maximum key length — 127 Unicode characters in UTF-8
- Maximum value length — 255 Unicode characters in UTF-8
- Tag keys and values are case-sensitive.
- Do not use the "aws:" prefix in your tag names or values because it’s reserved for AWS use. You can’t edit or delete tag names or values with this prefix. Tags with this prefix don’t count against your tags per resource limit.
- If your tagging schema is used across multiple services and resources, remember that other services may have restrictions on allowed characters. Generally, allowed characters are: letters, spaces, and numbers representable in UTF-8, and the following special characters: + - = . _ : / @.

Using Tags with IAM Policies

You can apply tag-based resource-level permissions in the IAM policies you use for AWS RoboMaker API actions. This gives you better control over what resources a user can create, modify, or use. You use the `Condition` element (also called the `Condition` block) with the following condition context keys and values in an IAM policy to control user access (permissions) based on a resource’s tags:

- Use `aws:ResourceTag/tag-key: tag-value` to allow or deny user actions on resources with specific tags.
- Use `aws:RequestTag/tag-key: tag-value` to require that a specific tag be used (or not used) when making an API request to create or modify a resource that allows tags.
- Use `aws:TagKeys: [tag-key, ...]` to require that a specific set of tag keys be used (or not used) when making an API request to create or modify a resource that allows tags.

**Note**
The condition context keys and values in an IAM policy apply only to those AWS RoboMaker actions where an identifier for a resource capable of being tagged is a required parameter. For example, the use of `ListFleets` will not be allowed or denied on the basis of condition context keys and values because no taggable resource (fleet, robot, robot application, simulation application, simulation job, deployment job) is referenced in this request.

Controlling Access Using Tags in the AWS Identity and Access Management User Guide has additional information on using tags. The IAM JSON Policy Reference section of that guide has detailed syntax, descriptions, and examples of the elements, variables, and evaluation logic of JSON policies in IAM.

The following example policy applies two tag-based restrictions. An IAM user restricted by this policy:

- Cannot give a resource the tag "env=prod" (in the example, see the line "aws:RequestTag/env" : "prod"
- Cannot modify or access a resource that has an existing tag "env=prod" (in the example, see the line "aws:ResourceTag/env" : "prod").

```json
{
    "Version" : "2012-10-17",
    "Statement" : [
```
You can also specify multiple tag values for a given tag key by enclosing them in a list, like this:

```
"StringEquals" : {
    "aws:ResourceTag/env" : ["dev", "test"]
}
```

**Note**

If you allow or deny users access to resources based on tags, you must consider explicitly denying users the ability to add those tags to or remove them from the same resources. Otherwise, it's possible for a user to circumvent your restrictions and gain access to a resource by modifying its tags.
Troubleshooting

Solve common problems you might find when developing robotics applications with AWS RoboMaker.

Topics

- Troubleshooting Simulation Jobs (p. 128)
- Troubleshooting Development Environments (p. 131)
- Troubleshooting Deployments (p. 131)
- Troubleshooting Colcon Build and Bundle (p. 134)

Troubleshooting Simulation Jobs

This section can help you fix issues with AWS RoboMaker simulation jobs.

Simulation Job Failed

If your simulation job failed, see the following common solutions.

Are Your Amazon S3 Resources in the Same Region as AWS RoboMaker?

Your robot application, simulation application, and output locations must be in the same Region as AWS RoboMaker. Verify your application sources and simulation job output locations.

Did Your Robot Application Exit Abnormally?

There was a problem setting up your robot application for simulation. Review the robot application logs for the simulation job in Amazon CloudWatch.

Logs are accessed from the simulation job detail screen. Select Logs, and then select a log stream. To look for specific issues, use the filter. For example, WARNING or ERROR.

Is Your Application Missing an .so File?

If your application crashed, it might be missing a dependent shared object (.so) file. Extract your application bundle in your environment and verify that the shared object libraries you need are in /usr/lib or /usr/local/lib. Make sure the dependency is added to your package .xml file.

Did You Use the ARN of your Role with the CLI?

When you call create-simulation-job from the AWS CLI, use the full Amazon Resource Name (ARN) of the role, not just the role name.

Does Your Role Have a Trust Policy for AWS RoboMaker?

If you are passing the full Amazon Resource Name (ARN) of the IAM role when you call create-simulation-job from the AWS CLI, your trust policy might have insufficient privileges. Check the role to make sure it has a trust relationship with robomaker.amazonaws.com.
Does Your Role Have Permissions to Publish to Amazon S3?

If you specify an output S3 bucket for a simulation job, your role must have write permissions to the bucket. Update your trust policy to include write permissions. The example trust policy below adds read, list, and write permissions to an S3 bucket.

```
{
   "Action": "s3:ListBucket",
   "Resource": [
     "my-bucket"
   ],
   "Effect": "Allow"
},
{
   "Action": [
     "s3:Get*",
     "s3:List*"
   ],
   "Resource": [
     "my-bucket"
   ],
   "Effect": "Allow"
},
{
   "Action": "s3:Put*",
   "Resource": [
     "my-bucket"
   ],
   "Effect": "Allow"
}
```

Does Your Role Have Permission to Publish to Amazon CloudWatch?

Update the permissions policies of your IAM role with CloudWatch access.

```
{
   "Effect": "Allow",
   "Action": [
     "logs:CreateLogGroup",
     "logs:CreateLogStream",
     "logs:PutLogEvents",
     "logs:DescribeLogStreams"
   ],
   "Resource": "*"
}
```

Does your Application Have a Mismatched Entity Tag?

The entity tag (ETag) is a hash of the Amazon S3 object provided while creating the simulation. The ETag reflects changes only to the contents of an object, not its metadata. If you change the content of the robot application or simulation bundle in Amazon S3 before AWS RoboMaker has consumed it, there will be a version mismatch.

To resolve this, create a new robot application or simulation application version and provide the key location for the updated application bundle. For more information, see Creating a Robot Application Version (p. 53) or Creating a Simulation Application Version (p. 53).
Is Your Subnet ENI Limit Exceeded?

AWS RoboMaker uses one elastic network interface (ENI) for each concurrent simulation job in the subnet in which the simulation job is run. Each of these must be assigned an IP address. To resolve this, you can:

- Delete unused ENIs to free up IP addresses in the subnet. To delete an unused ENI, see Deleting a Network Interface.
- Request a service limit increase for ENIs in a specific AWS Region.

Is the Launch Command Properly Configured?

Gazebo can take a few minutes to launch if your simulation is complex. If AWS RoboMaker spends more than 10 minutes preparing the simulation job, there might be a problem with the launch command.

Cancel the job and then create a new simulation job. If the problem persists, contact AWS Support.

It's also possible that one of the ROS nodes did not start or experienced problems. Check the simulation logs for errors. You can also use the terminal simulation to connect and troubleshoot the running simulation job.

Are Your Subnets in Zones AWS RoboMaker Supports?

Provide subnets in two of the AWS Availability Zones supported by AWS RoboMaker. API response contains a list of supported AWS Availability Zones.

Are the Launch File and Package Name Correct?

Use CloudWatch Logs to verify the package name and launch file used by the simulation job. Filter to roslaunch events, and then expand each event to look for issues similar to the following.

```
[launch_file.launch] is neither a launch file in package [package_name] nor is [package_name] a launch file name
```

Is the Node Package Named Correctly in the Launch File?

Use CloudWatch Logs to verify the node package name used by the simulation job. Filter to cannot launch node events, and then expand each event to look for issues similar to the following.

```
ERROR: cannot launch node of type [node_package_name/node_type]: node_package_name
```

Did you Include an Incorrect Launch File?

Use CloudWatch Logs to check if the launch file was not found. Filter to roslaunch events, and then expand each event to look for issues similar to the following.

```
while processing directory/path/to/launch/launch_file
Invalid roslaunch XML syntax: [Errno 2] No such file or directory: 'directory/path/to/launch/launch_file'
```

Are Your World File Model References Correct?

Use CloudWatch Logs to verify all of the models in your world file are correct. If a model could not be located, you see information like the following.
Troubleshooting Development Environments

This section helps you fix issues with creating applications in the AWS RoboMaker environment.

There is a Problem with the Sample Code

If there is a problem with sample code, see the following solutions.

Did the Sample Fail to Install?

The sample might not install if AWS Cloud9 is running background updates. Issues can arise when the IDE is trying to update dependencies for a repository. Wait a few minutes and try again.

Troubleshooting Deployments

This section can help you fix issues when deploying a robot application to a fleet.

My Deployment Failed

See the following topics for common solutions.

Is Your Robot Part of a Fleet?

A robot must be part of a fleet to receive a deployment. To check the status of your robots in the AWS RoboMaker console, expand Fleet management and then choose Robots. Robots that are registered to a fleet will include the Fleet name.

Is AWS IoT Greengrass Running on Your Robot?

To configure and run the AWS IoT Greengrass core software, follow the steps in Module 1: Environment Setup for Greengrass, then follow the steps in Start AWS Greengrass on the Core Device.

Is a Resource Missing?

In the deployment details page, review the Failure reason. It will list the missing resource. Verify that the resource exists. For example, if the robot application is missing, it might have been deleted from the Amazon S3 location. Also, the Amazon S3 ETag information might be incorrect.

Did the AWS IoT Greengrass Deployment Encounter a Problem?

In the deployment details page, review the Failure reason. It will contain more details.
If AWS IoT Greengrass has problems starting or restarting, look at the AWS IoT Greengrass system logs located on the robot at /greengrass/ggc/var/log/system/runtime.log.

To troubleshoot, see Troubleshooting AWS IoT Greengrass.

**Do you get Error x509: Certificate Signed by Unknown Authority?**

This error might occur if you recently updated your certification or configuration files from AWS RoboMaker or AWS IoT Greengrass in your robot. The error appears in the log, /greengrass/ggc/var/log/system/runtime.log. To resolve, upgrade root.ca.pem in your robot. To upgrade, follow step 5 in Start AWS IoT Greengrass on Core Device.

**Was the Failure Threshold Breached?**

Deployment stops if the failure threshold is exceeded. You can raise the threshold to attempt to deploy to more of your fleet.

**Is the Deployment Taking Longer Than Expected?**

Deployment time depends on the size of the robot application package. It also depends on the robot network conditions. If you have many robots and are deploying few concurrently, deployment might take longer.

Once started, a single robot deployment timeout is five hours. Use the deployment detail page to identify robots that have an active deployment. Use SSH to connect. Use the command ps aux | grep 'greengrass' to verify AWS IoT Greengrass is running. To troubleshoot, see Troubleshooting AWS IoT Greengrass.

**Did Your Robot Receive the Deployment Request?**

Your robot might not receive the deployment request if AWS IoT Greengrass is not properly configured and running.

First, verify that your robot received the deployment request. SSH to the robot. Once connected, use ps aux | grep 'greengrass' to see if AWS IoT Greengrass is running. check for errors in the log located at /greengrass/ggc/var/log/user/region/account/aws-robomaker-deployment-function-robot architecture_DO_NOT_DELETE.log.

If there are no errors in the log, make sure you have AWS IoT Greengrass version 1.7.0 or later installed. See Module 1: Environment Setup for Greengrass for more information.

Next, make sure AWS IoT Greengrass is running with the following command.

```bash
ps aux | grep 'greengrass'
```

If it is running, look at the AWS IoT Greengrass system logs located on the robot at /greengrass/ggc/var/log/system/runtime.log. See Troubleshooting AWS IoT Greengrass for additional troubleshooting information.

**Will an Offline Robot get the Newest Deployment or the Last Deployment to Succeed?**

If you received the RobotAgentResponseTimeoutException in a new deployment because a robot is unavailable, when it becomes available it downloads the latest (timed out) deployment. The robot status is updated after the robot finishes deployment. The status changes from NoResponse to InSync.
Are you Trying to Override Environment Variables Sourced in Bundle Setup Scripts?

Environment variables that are sourced from setup bundle scripts can override similar variables that are defined when a job is created. To avoid this issue, define a new, unique environment variable or change values used in the setup scripts.

ROS did not Restart on Device Reboot

Failure to restart happens when the AWS IoT Greengrass daemon is not configured to run on restart. To modify the device init system to run the AWS IoT Greengrass daemon, see Configure the Init System to Start the Greengrass Daemon.

ROS Needs Root Privilege

You can grant the ggc_user root privilege permission by configuring the Linux user profile in /etc/passwd. You can set the user_id and group_id to 0. Use the following command.

```
named:password:userid:groupid:gecos:home directory:shell ggc_user:x:0:0::/home/ggc_user:/bin/false
```

Managing the Robot Application and Default ROS HOME Directories

AWS IoT Greengrass currently downloads the robot application to the /home/ggc_user folder if it was created when you created the ggc_user. Otherwise, it downloads to /tmp/roboMakerDeploymentPackage. The /tmp directory may be cleaned up on reboot.

If you don’t want the bundle to be removed after the reboot, ensure the directory /home/ggc_user exists and ggc_user has read and write permissions.

The default ROS_HOME directory is /home/ggc_user/ros/home/deployment-id. If the ggc_user home directory does not exist, the default directory is /tmp/ros/home/deployment-id. You can also specify the ROS_HOME directory by adding it into the environment variables when you create a deployment job. The ggc_user must have read and write permissions to the folder.

If /tmp/roboMakerDeploymentPackage was added to tmpfiles.d configuration file to persist it to the tmp folder, remove it.

**Note**
AWS RoboMaker deployment cleans the robot application folder. It does not clean the ROS_HOME folder. Use a pre-deployment or post-deployment script to manage the directory.

How do I Use Local Libraries on Robot?

You can override or link the deployed bundle to use local ROS libraries. To do this, provide environment variables when you create the deployment job. For example, set ROS_ROOT to the local ROS.

```
"ROS_ROOT":"/opt/ros/kinetic/share/ros"
```

Did you get a Robot Application version ETag mismatch?

This happens when the source file (Amazon S3 object) of the robot application version you selected was modified since the version was created and the Etag no longer matches. When you create a robot
application version, AWS RoboMaker remembers the Amazon S3 path and the ETag of the version. You must not remove or modify the version.

To resolve the issue, locate an unmodified version of the source file, use a different version or create a new version. For more information about application versioning, see Application Versioning (p. 28).

Troubleshooting Colcon Build and Bundle

Help with building and bundling applications with colcon. For more information about the bundle format and other technical details, see Building and Bundling a ROS Application for AWS RoboMaker on the AWS Open Source Blog.

Colcon Build Failed

See the following topics for common solutions.

Are There CMakeLists.txt Files in Nested Folders?

If you are building a ROS1 application that is built with catkin_make, colcon may not properly enumerate all of the packages in the workspace. This is usually caused by nested folder structures with CMakeLists.txt in one or more intermediate directories. colcon supports nested folder structures and finds your packages automatically.

For example, the CMakeLists.txt in the intermediate_directory is not required.

```
src/  ### package_1/  # okay
    |  ### package.xml
    |  ### CMakeLists.txt  # okay
    ### intermediate_directory/  # okay
        |  ### package_2
        |      |  ### package.xml
        |      |  ### CMakeLists.txt  # okay
        |  ### package_3
        |      |  ### package.xml
        |  ### CMakeLists.txt  # okay
        |  ### CMakeLists.txt  # okay
```

Are There Missing Install Directives in CMakeLists.txt?

If you are building a ROS1 application built with catkin_make, the devel directory and its setup.sh adds all local packages into the search paths for ROS tools. colcon behaves differently. It installs the targets you specify into the install directory, so all of your cmake install() directives are executed.

If you are experiencing errors like [my_launchfile] is neither a launch file in package [my_package] nor is [my_package] a launch file name or [rosrun] Couldn't find executable named my_node below /opt/ros/$ROS_DISTRO/share/my_package, then adding calls to install() in your CMakeLists.txt might fix the issue.

For more examples on how to fix this issue, see the ROS Wiki.

Colcon Bundle Failed

See the following topics for common solutions.
Cannot Locate rosdep Definition for [package_name]

Ensure that you are using the correct ROS package name for the package you want to depend on. For example, the package might be named ros-kinetic-packagename in apt, but in your package.xml it should be packagename. Search the ROS distribution GitHub repo to see if the package is in the existing rosdep database. If it is missing, add the dependency to rosdep. For more information about adding dependencies to rosdep, see the tutorials.

There are Missing Dependencies

Common error messages for missing dependencies include:

- Could not load libxyz.so
- No such file or directory some_script.py
- Could not load module 'python_dependency'

To solve this problem, add the dependency to package.xml of the packages that require it. Retry the bundle command.

If your application uses dependencies from your own apt or pip repository, you need to include those repositories when you invoke colcon bundle. To resolve this issue, try the following.

- Override the sources.lst used by the apt installer for colcon bundle with the --apt-sources-list argument. To avoid resolution errors for base operating system and ROS packages, we recommend that you include all the sources we currently use. For more information, see examplesources.list on GitHub.
- To override pip or pip3 sources, use --pip-args and --pip3-args arguments. The string after these arguments is passed directly to pip. For example, --extra-index-url https://my-custom-pip-repo/index.
This section contains the API Reference documentation.

**Actions**

The following actions are supported:

- `BatchDescribeSimulationJob` (p. 138)
- `CancelDeploymentJob` (p. 142)
- `CancelSimulationJob` (p. 144)
- `CancelSimulationJobBatch` (p. 146)
- `CreateDeploymentJob` (p. 148)
- `CreateFleet` (p. 154)
- `CreateRobot` (p. 157)
- `CreateRobotApplication` (p. 161)
- `CreateRobotApplicationVersion` (p. 165)
- `CreateSimulationApplication` (p. 169)
- `CreateSimulationApplicationVersion` (p. 174)
- `CreateSimulationJob` (p. 178)
- `DeleteFleet` (p. 187)
- `DeleteRobot` (p. 189)
- `DeleteRobotApplication` (p. 191)
- `DeleteSimulationApplication` (p. 193)
- `DeregisterRobot` (p. 195)
- `DescribeDeploymentJob` (p. 198)
- `DescribeFleet` (p. 202)
- `DescribeRobot` (p. 206)
- `DescribeRobotApplication` (p. 210)
- `DescribeSimulationApplication` (p. 214)
- `DescribeSimulationJob` (p. 218)
- `DescribeSimulationJobBatch` (p. 225)
- `ListDeploymentJobs` (p. 232)
- `ListFleets` (p. 236)
- `ListRobotApplications` (p. 239)
- `ListRobots` (p. 242)
- `ListSimulationApplications` (p. 245)
- `ListSimulationJobBatches` (p. 248)
- `ListSimulationJobs` (p. 251)
- `ListTagsForResource` (p. 254)
- `RegisterRobot` (p. 256)
- `RestartSimulationJob` (p. 259)
- `StartSimulationJobBatch` (p. 261)
• SyncDeploymentJob (p. 270)
• TagResource (p. 275)
• UntagResource (p. 277)
• UpdateRobotApplication (p. 279)
• UpdateSimulationApplication (p. 283)
BatchDescribeSimulationJob

Describes one or more simulation jobs.

Request Syntax

```
POST /batchDescribeSimulationJob HTTP/1.1
Content-type: application/json

{   "jobs": [ "string" ]
}
```

URI Request Parameters

The request does not use any URI parameters.

Request Body

The request accepts the following data in JSON format.

**jobs (p. 138)**

A list of Amazon Resource Names (ARNs) of simulation jobs to describe.

- Type: Array of strings
- Array Members: Minimum number of 1 item. Maximum number of 100 items.
- Pattern: arn:. *
- Required: Yes

Response Syntax

```
HTTP/1.1 200
Content-type: application/json

{   "jobs": [   {   "arn": "string",   "clientRequestToken": "string",   "dataSources": [   {   "name": "string",   "s3Bucket": "string",   "s3Keys": [   {   "etag": "string",   "s3Key": "string"   }   ]   }   ]   }

```
"failureBehavior": "string",
"failureCode": "string",
"failureReason": "string",
"iamRole": "string",
"lastStartedAt": number,
"lastUpdatedAt": number,
"loggingConfig": {
   "recordAllRosTopics": boolean
},
"maxJobDurationInSeconds": number,
"name": "string",
"networkInterface": {
   "networkInterfaceId": "string",
   "privateIpAddress": "string",
   "publicIpAddress": "string"
},
"outputLocation": {
   "s3Bucket": "string",
   "s3Prefix": "string"
},
"robotApplications": [
{
   "application": "string",
   "applicationVersion": "string",
   "launchConfig": {
      "environmentVariables": {
         "string": "string"
      },
      "launchFile": "string",
      "packageName": "string",
      "portForwardingConfig": {
         "portMappings": [
            {
               "applicationPort": number,
               "enableOnPublicIp": boolean,
               "jobPort": number
            }
         ]
      },
      "streamUI": boolean
   }
}
],
"simulationApplications": [
{
   "application": "string",
   "applicationVersion": "string",
   "launchConfig": {
      "environmentVariables": {
         "string": "string"
      },
      "launchFile": "string",
      "packageName": "string",
      "portForwardingConfig": {
         "portMappings": [
            {
               "applicationPort": number,
               "enableOnPublicIp": boolean,
               "jobPort": number
            }
         ]
      },
      "streamUI": boolean
   }
}
],

"simulationTimeMillis": number,
"status": "string",
"tags": {
  "string": "string"
},
"vpcConfig": {
  "assignPublicIp": boolean,
  "securityGroups": [ "string" ],
  "subnets": [ "string" ],
  "vpcId": "string"
}
],
"unprocessedJobs": [ "string" ]

Response Elements

If the action is successful, the service sends back an HTTP 200 response. The following data is returned in JSON format by the service.

jobs (p. 138)
A list of simulation jobs.
Type: Array of SimulationJob (p. 326) objects

unprocessedJobs (p. 138)
A list of unprocessed simulation job Amazon Resource Names (ARNs).
Type: Array of strings
Array Members: Minimum number of 1 item. Maximum number of 100 items.
Pattern: arn:.*

Errors
For information about the errors that are common to all actions, see Common Errors (p. 342).

InternalServerException
AWS RoboMaker experienced a service issue. Try your call again.
HTTP Status Code: 500

InvalidParameterException
A parameter specified in a request is not valid, is unsupported, or cannot be used. The returned message provides an explanation of the error value.
HTTP Status Code: 400

ResourceNotFoundException
The specified resource does not exist.
HTTP Status Code: 400
ThrottlingException

AWS RoboMaker is temporarily unable to process the request. Try your call again.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3
CancelDeploymentJob

Cancels the specified deployment job.

Request Syntax

```
POST /cancelDeploymentJob HTTP/1.1
Content-type: application/json
{
    "job": "string"
}
```

URI Request Parameters

The request does not use any URI parameters.

Request Body

The request accepts the following data in JSON format.

**job (p. 142)**

The deployment job ARN to cancel.

Type: String


Pattern: arn:.*

Required: Yes

Response Syntax

```
HTTP/1.1 200
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response with an empty HTTP body.

Errors

For information about the errors that are common to all actions, see Common Errors (p. 342).

**InternalServerException**

AWS RoboMaker experienced a service issue. Try your call again.

HTTP Status Code: 500

**InvalidParameterException**

A parameter specified in a request is not valid, is unsupported, or cannot be used. The returned message provides an explanation of the error value.
HTTP Status Code: 400
**ResourceNotFoundException**

The specified resource does not exist.

HTTP Status Code: 400
**ThrottlingException**

AWS RoboMaker is temporarily unable to process the request. Try your call again.

HTTP Status Code: 400

**See Also**

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3
CancelSimulationJob

Cancels the specified simulation job.

Request Syntax

POST /cancelSimulationJob HTTP/1.1
Content-type: application/json

{  "job": "string" }

URI Request Parameters

The request does not use any URI parameters.

Request Body

The request accepts the following data in JSON format.

job (p. 144)

   The simulation job ARN to cancel.

   Type: String


   Pattern: arn:.*

   Required: Yes

Response Syntax

HTTP/1.1 200

Response Elements

If the action is successful, the service sends back an HTTP 200 response with an empty HTTP body.

Errors

For information about the errors that are common to all actions, see Common Errors (p. 342).

InternalServerException

   AWS RoboMaker experienced a service issue. Try your call again.

   HTTP Status Code: 500

InvalidParameterException

   A parameter specified in a request is not valid, is unsupported, or cannot be used. The returned message provides an explanation of the error value.
HTTP Status Code: 400

ResourceNotFoundException

The specified resource does not exist.

HTTP Status Code: 400

ThrottlingException

AWS RoboMaker is temporarily unable to process the request. Try your call again.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3
CancelSimulationJobBatch

Cancels a simulation job batch. When you cancel a simulation job batch, you are also cancelling all of the active simulation jobs created as part of the batch.

Request Syntax

POST /cancelSimulationJobBatch HTTP/1.1
Content-type: application/json

{
  "batch": "string"
}

URI Request Parameters

The request does not use any URI parameters.

Request Body

The request accepts the following data in JSON format.

batch (p. 146)

The id of the batch to cancel.

Type: String


Pattern: arn:.*

Required: Yes

Response Syntax

HTTP/1.1 200

Response Elements

If the action is successful, the service sends back an HTTP 200 response with an empty HTTP body.

Errors

For information about the errors that are common to all actions, see Common Errors (p. 342).

InternalServerException

AWS RoboMaker experienced a service issue. Try your call again.

HTTP Status Code: 500

InvalidParameterException

A parameter specified in a request is not valid, is unsupported, or cannot be used. The returned message provides an explanation of the error value.
HTTP Status Code: 400

**ResourceNotFoundException**

The specified resource does not exist.

HTTP Status Code: 400

**ThrottlingException**

AWS RoboMaker is temporarily unable to process the request. Try your call again.

HTTP Status Code: 400

**See Also**

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3
CreateDeploymentJob

Deploys a specific version of a robot application to robots in a fleet.

The robot application must have a numbered `applicationVersion` for consistency reasons. To create a new version, use `CreateRobotApplicationVersion` or see Creating a Robot Application Version.

**Note**

After 90 days, deployment jobs expire and will be deleted. They will no longer be accessible.

**Request Syntax**

```json
POST /createDeploymentJob HTTP/1.1
Content-type: application/json

{
"clientRequestToken": "string",
"deploymentApplicationConfigs": [
{
"application": "string",
"applicationVersion": "string",
"launchConfig": {
"environmentVariables": {
"string": "string"
},
"launchFile": "string",
"packageName": "string",
"postLaunchFile": "string",
"preLaunchFile": "string"
}
},
"deploymentConfig": {
"concurrentDeploymentPercentage": number,
"downloadConditionFile": {
"bucket": "string",
"etag": "string",
"key": "string"
},
"failureThresholdPercentage": number,
"robotDeploymentTimeoutInSeconds": number
},
"fleet": "string",
"tags": {
"string": "string"
}
}
```

**URI Request Parameters**

The request does not use any URI parameters.

**Request Body**

The request accepts the following data in JSON format.

**clientRequestToken (p. 148)**

Unique, case-sensitive identifier that you provide to ensure the idempotency of the request.

Type: String
Length Constraints: Minimum length of 1. Maximum length of 64.
Pattern: [a-zA-Z0-9_\-\=]*
Required: Yes

deploymentApplicationConfigs (p. 148)
The deployment application configuration.
Type: Array of DeploymentApplicationConfig (p. 292) objects
Array Members: Fixed number of 1 item.
Required: Yes

deploymentConfig (p. 148)
The requested deployment configuration.
Type: DeploymentConfig (p. 293) object
Required: No

fleet (p. 148)
The Amazon Resource Name (ARN) of the fleet to deploy.
Type: String
Pattern: arn:.*
Required: Yes

tags (p. 148)
A map that contains tag keys and tag values that are attached to the deployment job.
Type: String to string map
Key Length Constraints: Minimum length of 1. Maximum length of 128.
Key Pattern: [a-zA-Z0-9_\-\.\+/=]*
Value Length Constraints: Minimum length of 0. Maximum length of 256.
Value Pattern: [a-zA-Z0-9_\-\.\+/=]*
Required: No

Response Syntax

HTTP/1.1 200
Content-type: application/json

{  
  "arn": "string",
  "createdAt": number,
  "deploymentApplicationConfigs": [
  
    {  
      "application": "string",
      "applicationVersion": "string",

```
"launchConfig": {
  "environmentVariables": {
    "string": "string"
  },
  "launchFile": "string",
  "packageName": "string",
  "postLaunchFile": "string",
  "preLaunchFile": "string"
}
}
"deploymentConfig": {
  "concurrentDeploymentPercentage": number,
  "downloadConditionFile": {
    "bucket": "string",
    "etag": "string",
    "key": "string"
  },
  "failureThresholdPercentage": number,
  "robotDeploymentTimeoutInSeconds": number
},
"failureCode": "string",
"failureReason": "string",
"fleet": "string",
"status": "string",
"tags": {
  "string": "string"
}
}

**Response Elements**

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

**arn (p. 149)**

The Amazon Resource Name (ARN) of the deployment job.

Type: String


Pattern: arn:.*

**createdAt (p. 149)**

The time, in milliseconds since the epoch, when the fleet was created.

Type: Timestamp

**deploymentApplicationConfigs (p. 149)**

The deployment application configuration.

Type: Array of DeploymentApplicationConfig (p. 292) objects

Array Members: Fixed number of 1 item.

**deploymentConfig (p. 149)**

The deployment configuration.

Type: DeploymentConfig (p. 293) object
failureCode (p. 149)

The failure code of the simulation job if it failed:

**BadPermissionError**

AWS Greengrass requires a service-level role permission to access other services. The role must include the `AWSGreengrassResourceAccessRolePolicy` managed policy.

**ExtractingBundleFailure**

The robot application could not be extracted from the bundle.

**FailureThresholdBreached**

The percentage of robots that could not be updated exceeded the percentage set for the deployment.

**GreengrassDeploymentFailed**

The robot application could not be deployed to the robot.

**GreengrassGroupVersionDoesNotExist**

The AWS Greengrass group or version associated with a robot is missing.

**InternalServer**

An internal error has occurred. Retry your request, but if the problem persists, contact us with details.

**MissingRobotApplicationArchitecture**

The robot application does not have a source that matches the architecture of the robot.

**MissingRobotDeploymentResource**

One or more of the resources specified for the robot application are missing. For example, does the robot application have the correct launch package and launch file?

**PostLaunchFileFailure**

The post-launch script failed.

**PreLaunchFileFailure**

The pre-launch script failed.

**ResourceNotFound**

One or more deployment resources are missing. For example, do robot application source bundles still exist?

**RobotDeploymentNoResponse**

There is no response from the robot. It might not be powered on or connected to the internet.

Type: String

**failureReason (p. 149)**

The failure reason of the deployment job if it failed.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 1024.

Pattern: . *

**fleet (p. 149)**

The target fleet for the deployment job.

Type: String


Pattern: arn: . *

**status (p. 149)**

The status of the deployment job.

Type: String

Valid Values: Pending | Preparing | InProgress | Failed | Succeeded | Canceled

**tags (p. 149)**

The list of all tags added to the deployment job.

Type: String to string map

Key Length Constraints: Minimum length of 1. Maximum length of 128.

Key Pattern: [a-zA-Z0-9 _-\.\+/=:] *

Value Length Constraints: Minimum length of 0. Maximum length of 256.

Value Pattern: [a-zA-Z0-9 _-\.\+/=:] *

**Errors**

For information about the errors that are common to all actions, see Common Errors (p. 342).

**ConcurrentDeploymentException**

The failure percentage threshold percentage was met.

HTTP Status Code: 400

**IdempotentParameterMismatchException**

The request uses the same client token as a previous, but non-identical request. Do not reuse a client token with different requests, unless the requests are identical.

HTTP Status Code: 400

**InternalServerException**

AWS RoboMaker experienced a service issue. Try your call again.

HTTP Status Code: 500
InvalidParameterException

A parameter specified in a request is not valid, is unsupported, or cannot be used. The returned message provides an explanation of the error value.

HTTP Status Code: 400

LimitExceededException

The requested resource exceeds the maximum number allowed, or the number of concurrent stream requests exceeds the maximum number allowed.

HTTP Status Code: 400

ResourceNotFoundException

The specified resource does not exist.

HTTP Status Code: 400

ThrottlingException

AWS RoboMaker is temporarily unable to process the request. Try your call again.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3
CreateFleet

Creates a fleet, a logical group of robots running the same robot application.

Request Syntax

```
POST /createFleet HTTP/1.1
Content-type: application/json

{
  "name": "string",
  "tags": {
    "string" : "string"
  }
}
```

URI Request Parameters

The request does not use any URI parameters.

Request Body

The request accepts the following data in JSON format.

**name (p. 154)**

The name of the fleet.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 255.

Pattern: [a-zA-Z0-9\-_]*

Required: Yes

**tags (p. 154)**

A map that contains tag keys and tag values that are attached to the fleet.

Type: String to string map

Key Length Constraints: Minimum length of 1. Maximum length of 128.

Key Pattern: [a-zA-Z0-9\-_\/+\=:]*

Value Length Constraints: Minimum length of 0. Maximum length of 256.

Value Pattern: [a-zA-Z0-9\-_\/+\=:]*

Required: No

Response Syntax

```
HTTP/1.1 200
Content-type: application/json
```

154
Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

**arn (p. 154)**

The Amazon Resource Name (ARN) of the fleet.

Type: String


Pattern: `arn:`.*

**createdAt (p. 154)**

The time, in milliseconds since the epoch, when the fleet was created.

Type: Timestamp

**name (p. 154)**

The name of the fleet.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 255.

Pattern: `[a-zA-Z0-9_\-]*`

**tags (p. 154)**

The list of all tags added to the fleet.

Type: String to string map

Key Length Constraints: Minimum length of 1. Maximum length of 128.

Key Pattern: `[a-zA-Z0-9_\-\./+=:]*`

Value Length Constraints: Minimum length of 0. Maximum length of 256.

Value Pattern: `[a-zA-Z0-9_\-\./+=:]*`

Errors

For information about the errors that are common to all actions, see Common Errors (p. 342).

**InternalServerException**

AWS RoboMaker experienced a service issue. Try your call again.
HTTP Status Code: 500

InvalidParameterException

A parameter specified in a request is not valid, is unsupported, or cannot be used. The returned message provides an explanation of the error value.

HTTP Status Code: 400

LimitExceededException

The requested resource exceeds the maximum number allowed, or the number of concurrent stream requests exceeds the maximum number allowed.

HTTP Status Code: 400

ThrottlingException

AWS RoboMaker is temporarily unable to process the request. Try your call again.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3
CreateRobot

Creates a robot.

Request Syntax

POST /createRobot HTTP/1.1
Content-type: application/json

{
  "architecture": "string",
  "greengrassGroupId": "string",
  "name": "string",
  "tags": {
    "string" : "string"
  }
}

URI Request Parameters

The request does not use any URI parameters.

Request Body

The request accepts the following data in JSON format.

architecture (p. 157)

The target architecture of the robot.

Type: String

Valid Values: X86_64 | ARM64 | ARMHF

Required: Yes

greengrassGroupId (p. 157)

The Greengrass group id.

Type: String


Pattern: . *

Required: Yes

name (p. 157)

The name for the robot.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 255.

Pattern: [a-zA-Z0-9_\-]*

Required: Yes
tags (p. 157)

A map that contains tag keys and tag values that are attached to the robot.

Type: String to string map

Key Length Constraints: Minimum length of 1. Maximum length of 128.

Key Pattern: [a-zA-Z0-9 _.-/+=:]*

Value Length Constraints: Minimum length of 0. Maximum length of 256.

Value Pattern: [a-zA-Z0-9 _.-/+=:]*

Required: No

Response Syntax

HTTP/1.1 200
Content-type: application/json

```
{
  "architecture": "string",
  "arn": "string",
  "createdAt": number,
  "greengrassGroupId": "string",
  "name": "string",
  "tags": {
    "string" : "string"
  }
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

architecture (p. 158)

The target architecture of the robot.

Type: String

Valid Values: X86_64 | ARM64 | ARMHF

arn (p. 158)

The Amazon Resource Name (ARN) of the robot.

Type: String


Pattern: arn:.*

createdAt (p. 158)

The time, in milliseconds since the epoch, when the robot was created.

Type: Timestamp
greengrassGroupId (p. 158)

The Amazon Resource Name (ARN) of the Greengrass group associated with the robot.

Type: String


Pattern: .*

name (p. 158)

The name of the robot.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 255.

Pattern: [a-zA-Z0-9_\-]*

tags (p. 158)

The list of all tags added to the robot.

Type: String to string map

Key Length Constraints: Minimum length of 1. Maximum length of 128.

Key Pattern: [a-zA-Z0-9 _\-]*

Value Length Constraints: Minimum length of 0. Maximum length of 256.

Value Pattern: [a-zA-Z0-9 _\-\+/=:]*

Errors

For information about the errors that are common to all actions, see Common Errors (p. 342).

InternalServerErrorException

AWS RoboMaker experienced a service issue. Try your call again.

HTTP Status Code: 500

InvalidParameterValue

A parameter specified in a request is not valid, is unsupported, or cannot be used. The returned message provides an explanation of the error value.

HTTP Status Code: 400

LimitExceededException

The requested resource exceeds the maximum number allowed, or the number of concurrent stream requests exceeds the maximum number allowed.

HTTP Status Code: 400

ResourceAlreadyExistsException

The specified resource already exists.

HTTP Status Code: 400
**ThrottlingException**

AWS RoboMaker is temporarily unable to process the request. Try your call again.

HTTP Status Code: 400

**See Also**

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3
CreateRobotApplication

Creates a robot application.

Request Syntax

POST /createRobotApplication HTTP/1.1
Content-type: application/json

{  
  "name": "string",
  "robotSoftwareSuite": {  
    "name": "string",
    "version": "string"
  },
  "sources": [
    {  
      "architecture": "string",
      "s3Bucket": "string",
      "s3Key": "string"
    }
  ],
  "tags": {
    "string": "string"
  }
}

URI Request Parameters

The request does not use any URI parameters.

Request Body

The request accepts the following data in JSON format.

name (p. 161)

The name of the robot application.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 255.

Pattern: [a-zA-Z0-9_-]*

Required: Yes

robotSoftwareSuite (p. 161)

The robot software suite (ROS distribution) used by the robot application.

Type: RobotSoftwareSuite (p. 320) object

Required: Yes

sources (p. 161)

The sources of the robot application.

Type: Array of SourceConfig (p. 339) objects
Required: Yes

tags (p. 161)

A map that contains tag keys and tag values that are attached to the robot application.

Type: String to string map

Key Length Constraints: Minimum length of 1. Maximum length of 128.

Key Pattern: [a-zA-Z0-9 _.-/+=:*]*

Value Length Constraints: Minimum length of 0. Maximum length of 256.

Value Pattern: [a-zA-Z0-9 _.-/+=:*]*

Required: No

Response Syntax

HTTP/1.1 200
Content-type: application/json

{
    "arn": "string",
    "lastUpdatedAt": number,
    "name": "string",
    "revisionId": "string",
    "RobotSoftwareSuite": {
        "name": "string",
        "version": "string"
    },
    "sources": [
        {
            "architecture": "string",
            "etag": "string",
            "s3Bucket": "string",
            "s3Key": "string"
        }
    ],
    "tags": {
        "string": "string"
    },
    "version": "string"
}

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

arn (p. 162)

The Amazon Resource Name (ARN) of the robot application.

Type: String


Pattern: arn:*
**lastUpdatedAt (p. 162)**

The time, in milliseconds since the epoch, when the robot application was last updated.

Type: Timestamp

**name (p. 162)**

The name of the robot application.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 255.

Pattern: `[a-zA-Z0-9_\-]*`

**revisionId (p. 162)**

The revision id of the robot application.

Type: String


Pattern: `[a-zA-Z0-9_.\-]*`

**robotSoftwareSuite (p. 162)**

The robot software suite (ROS distribution) used by the robot application.

Type: `RobotSoftwareSuite (p. 320)` object

**sources (p. 162)**

The sources of the robot application.

Type: Array of `Source (p. 338)` objects

**tags (p. 162)**

The list of all tags added to the robot application.

Type: String to string map

Key Length Constraints: Minimum length of 1. Maximum length of 128.

Key Pattern: `[a-zA-Z0-9_.\-\+/=]*`

Value Length Constraints: Minimum length of 0. Maximum length of 256.

Value Pattern: `[a-zA-Z0-9_.\-\+/=]*`

**version (p. 162)**

The version of the robot application.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 255.

Pattern: `(\$LATEST) | [0-9]*`

**Errors**

For information about the errors that are common to all actions, see Common Errors (p. 342).
IdempotentParameterMismatchException

The request uses the same client token as a previous, but non-identical request. Do not reuse a client token with different requests, unless the requests are identical.

HTTP Status Code: 400

InternalServerException

AWS RoboMaker experienced a service issue. Try your call again.

HTTP Status Code: 500

InvalidParameterException

A parameter specified in a request is not valid, is unsupported, or cannot be used. The returned message provides an explanation of the error value.

HTTP Status Code: 400

LimitExceededException

The requested resource exceeds the maximum number allowed, or the number of concurrent stream requests exceeds the maximum number allowed.

HTTP Status Code: 400

ResourceAlreadyExistsException

The specified resource already exists.

HTTP Status Code: 400

ThrottlingException

AWS RoboMaker is temporarily unable to process the request. Try your call again.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3
CreateRobotApplicationVersion

Creates a version of a robot application.

Request Syntax

POST /createRobotApplicationVersion HTTP/1.1
Content-type: application/json

{
    "application": "string",
    "currentRevisionId": "string"
}

URI Request Parameters

The request does not use any URI parameters.

Request Body

The request accepts the following data in JSON format.

application (p. 165)

The application information for the robot application.

Type: String


Pattern: arn:.*

Required: Yes

currentRevisionId (p. 165)

The current revision id for the robot application. If you provide a value and it matches the latest revision ID, a new version will be created.

Type: String


Pattern: [a-zA-Z0-9_.\-]*

Required: No

Response Syntax

HTTP/1.1 200
Content-type: application/json

{
    "arn": "string",
    "lastUpdatedAt": number,
    "name": "string",
    "revisionId": "string",
    "robotSoftwareSuite": {

CreateRobotApplicationVersion

```json

"name": "string",
"version": "string"
}
"sources": [
{
"architecture": "string",
"etag": "string",
"s3Bucket": "string",
"s3Key": "string"
}
],
"version": "string"
}

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

**arn (p. 165)**

The Amazon Resource Name (ARN) of the robot application.

Type: String


Pattern: `arn:.*`

**lastUpdatedAt (p. 165)**

The time, in milliseconds since the epoch, when the robot application was last updated.

Type: Timestamp

**name (p. 165)**

The name of the robot application.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 255.

Pattern: `[a-zA-Z0-9-_]*`

**revisionId (p. 165)**

The revision id of the robot application.

Type: String


Pattern: `[a-zA-Z0-9-_]*`

**robotSoftwareSuite (p. 165)**

The robot software suite (ROS distribution) used by the robot application.

Type: `RobotSoftwareSuite (p. 320)` object

**sources (p. 165)**

The sources of the robot application.
Type: Array of **Source** (p. 338) objects

**version** (p. 165)

The version of the robot application.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 255.

Pattern: \($LATEST\) | \([0-9]^{*}\)

**Errors**

For information about the errors that are common to all actions, see *Common Errors* (p. 342).

**IdempotentParameterMismatchException**

The request uses the same client token as a previous, but non-identical request. Do not reuse a client token with different requests, unless the requests are identical.

HTTP Status Code: 400

**InternalServerException**

AWS RoboMaker experienced a service issue. Try your call again.

HTTP Status Code: 500

**InvalidParameterException**

A parameter specified in a request is not valid, is unsupported, or cannot be used. The returned message provides an explanation of the error value.

HTTP Status Code: 400

**LimitExceededException**

The requested resource exceeds the maximum number allowed, or the number of concurrent stream requests exceeds the maximum number allowed.

HTTP Status Code: 400

**ThrottlingException**

AWS RoboMaker is temporarily unable to process the request. Try your call again.

HTTP Status Code: 400

**See Also**

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3
CreateSimulationApplication

Creates a simulation application.

Request Syntax

```
POST /createSimulationApplication HTTP/1.1
Content-type: application/json

{
  "name": "string",
  "renderingEngine": {
    "name": "string",
    "version": "string"
  },
  "robotSoftwareSuite": {
    "name": "string",
    "version": "string"
  },
  "simulationSoftwareSuite": {
    "name": "string",
    "version": "string"
  },
  "sources": [
    {
      "architecture": "string",
      "s3Bucket": "string",
      "s3Key": "string"
    }
  ],
  "tags": {
    "string": "string"
  }
}
```

URI Request Parameters

The request does not use any URI parameters.

Request Body

The request accepts the following data in JSON format.

**name (p. 169)**

The name of the simulation application.

- **Type:** String
- **Length Constraints:** Minimum length of 1. Maximum length of 255.
- **Pattern:** `[a-zA-Z0-9-_\-]*`
- **Required:** Yes

**renderingEngine (p. 169)**

The rendering engine for the simulation application.

- **Type:** RenderingEngine (p. 312) object
**CreateSimulationApplication**

- **Required**: No

**robotSoftwareSuite (p. 169)**

The robot software suite (ROS distribution) used by the simulation application.

Type: [RobotSoftwareSuite (p. 320)](p. 320) object

- **Required**: Yes

**simulationSoftwareSuite (p. 169)**

The simulation software suite used by the simulation application.

Type: [SimulationSoftwareSuite (p. 337)](p. 337) object

- **Required**: Yes

**sources (p. 169)**

The sources of the simulation application.

Type: Array of [SourceConfig (p. 339)](p. 339) objects

- **Required**: Yes

**tags (p. 169)**

A map that contains tag keys and tag values that are attached to the simulation application.

Type: String to string map

Key Length Constraints: Minimum length of 1. Maximum length of 128.

Key Pattern: `[a-zA-Z0-9 _.-\-/+=:]*`

Value Length Constraints: Minimum length of 0. Maximum length of 256.

Value Pattern: `[a-zA-Z0-9 _.-\-/+=:]*`

- **Required**: No

**Response Syntax**

```
HTTP/1.1 200
Content-type: application/json

{
    "arn": "string",
    "lastUpdatedAt": number,
    "name": "string",
    "renderingEngine": {
        "name": "string",
        "version": "string"
    },
    "revisionId": "string",
    "robotSoftwareSuite": {
        "name": "string",
        "version": "string"
    },
    "simulationSoftwareSuite": {
        "name": "string",
        "version": "string"
    }
}
```
"sources": [
  {
    "architecture": "string",
    "etag": "string",
    "s3Bucket": "string",
    "s3Key": "string"
  }
],
"tags": {
  "string": "string"
},
"version": "string"

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

arn (p. 170)

The Amazon Resource Name (ARN) of the simulation application.

Type: String


Pattern: arn:.*

lastUpdatedAt (p. 170)

The time, in milliseconds since the epoch, when the simulation application was last updated.

Type: Timestamp

name (p. 170)

The name of the simulation application.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 255.

Pattern: [a-zA-Z0-9\-\_\-]*

renderingEngine (p. 170)

The rendering engine for the simulation application.

Type: RenderingEngine (p. 312) object

revisionId (p. 170)

The revision id of the simulation application.

Type: String


Pattern: [a-zA-Z0-9\-\_\-]*

robotSoftwareSuite (p. 170)

Information about the robot software suite (ROS distribution).

171
CreateSimulationApplication

Type: `RobotSoftwareSuite (p. 320)` object

`simulationSoftwareSuite (p. 170)`

The simulation software suite used by the simulation application.

Type: `SimulationSoftwareSuite (p. 337)` object

`sources (p. 170)`

The sources of the simulation application.

Type: Array of `Source (p. 338)` objects

`tags (p. 170)`

The list of all tags added to the simulation application.

Type: String to string map

Key Length Constraints: Minimum length of 1. Maximum length of 128.

Key Pattern: `[a-zA-Z0-9 _-.\+/=:]*`

Value Length Constraints: Minimum length of 0. Maximum length of 256.

Value Pattern: `[a-zA-Z0-9 _-.\+/=:]*`

`version (p. 170)`

The version of the simulation application.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 255.

Pattern: `(\$LATEST)|[0-9]*`

**Errors**

For information about the errors that are common to all actions, see [Common Errors (p. 342)](#).

**IdempotentParameterMismatchException**

The request uses the same client token as a previous, but non-identical request. Do not reuse a client token with different requests, unless the requests are identical.

HTTP Status Code: 400

**InternalServerException**

AWS RoboMaker experienced a service issue. Try your call again.

HTTP Status Code: 500

**InvalidParameterException**

A parameter specified in a request is not valid, is unsupported, or cannot be used. The returned message provides an explanation of the error value.

HTTP Status Code: 400

**LimitExceededException**

The requested resource exceeds the maximum number allowed, or the number of concurrent stream requests exceeds the maximum number allowed.
HTTP Status Code: 400

**ResourceAlreadyExistsException**

The specified resource already exists.

HTTP Status Code: 400

**ThrottlingException**

AWS RoboMaker is temporarily unable to process the request. Try your call again.

HTTP Status Code: 400

**See Also**

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3
CreateSimulationApplicationVersion

Creates a simulation application with a specific revision id.

Request Syntax

```
POST /createSimulationApplicationVersion HTTP/1.1
Content-type: application/json

{
  "application": "string",
  "currentRevisionId": "string"
}
```

URI Request Parameters

The request does not use any URI parameters.

Request Body

The request accepts the following data in JSON format.

application (p. 174)

The application information for the simulation application.

Type: String


Pattern: arn:.*

Required: Yes

currentRevisionId (p. 174)

The current revision id for the simulation application. If you provide a value and it matches the latest revision ID, a new version will be created.

Type: String


Pattern: [a-zA-Z0-9_.\-]*

Required: No

Response Syntax

```
HTTP/1.1 200
Content-type: application/json

{
  "arn": "string",
  "lastUpdatedAt": number,
  "name": "string",
  "renderingEngine": {
    "name": "string",
  }
}
```
"version": "string",
"revisionId": "string",
"robotSoftwareSuite": {
  "name": "string",
  "version": "string"
},
"simulationSoftwareSuite": {
  "name": "string",
  "version": "string"
},
"sources": [
  {
    "architecture": "string",
    "etag": "string",
    "s3Bucket": "string",
    "s3Key": "string"
  }
],
"version": "string"

### Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

**arn (p. 174)**

The Amazon Resource Name (ARN) of the simulation application.

Type: String


Pattern: arn:.*

**lastUpdatedAt (p. 174)**

The time, in milliseconds since the epoch, when the simulation application was last updated.

Type: Timestamp

**name (p. 174)**

The name of the simulation application.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 255.

Pattern: [a-zA-Z0-9_-]*

**renderingEngine (p. 174)**

The rendering engine for the simulation application.

Type: RenderingEngine (p. 312) object

**revisionId (p. 174)**

The revision ID of the simulation application.

Type: String
Pattern: \[a-zA-Z0-9_.\-]*

**robotSoftwareSuite (p. 174)**

Information about the robot software suite (ROS distribution).
Type: `RobotSoftwareSuite (p. 320)` object

**simulationSoftwareSuite (p. 174)**

The simulation software suite used by the simulation application.
Type: `SimulationSoftwareSuite (p. 337)` object

**sources (p. 174)**

The sources of the simulation application.
Type: Array of `Source (p. 338)` objects

**version (p. 174)**

The version of the simulation application.
Type: String

Length Constraints: Minimum length of 1. Maximum length of 255.
Pattern: \((\$LATEST) | [0-9]*\)

## Errors

For information about the errors that are common to all actions, see [Common Errors (p. 342)].

**IdempotentParameterMismatchException**

The request uses the same client token as a previous, but non-identical request. Do not reuse a client token with different requests, unless the requests are identical.

HTTP Status Code: 400

**InternalServerException**

AWS RoboMaker experienced a service issue. Try your call again.

HTTP Status Code: 500

**InvalidParameterException**

A parameter specified in a request is not valid, is unsupported, or cannot be used. The returned message provides an explanation of the error value.

HTTP Status Code: 400

**LimitExceededException**

The requested resource exceeds the maximum number allowed, or the number of concurrent stream requests exceeds the maximum number allowed.

HTTP Status Code: 400

**ThrottlingException**

AWS RoboMaker is temporarily unable to process the request. Try your call again.
HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3
CreateSimulationJob

Creates a simulation job.

**Note**
After 90 days, simulation jobs expire and will be deleted. They will no longer be accessible.

**Request Syntax**

```
POST /createSimulationJob HTTP/1.1
Content-type: application/json

{
  "clientRequestToken": "string",
  "dataSources": [
    { "name": "string",
      "s3Bucket": "string",
      "s3Keys": [ "string" ]
    },
  ],
  "failureBehavior": "string",
  "iamRole": "string",
  "loggingConfig": {
    "recordAllRosTopics": boolean,
  },
  "maxJobDurationInSeconds": number,
  "outputLocation": {
    "s3Bucket": "string",
    "s3Prefix": "string"
  },
  "robotApplications": [
    { "application": "string",
      "applicationVersion": "string",
      "launchConfig": {
        "environmentVariables": {
          "string": "string"
        },
        "launchFile": "string",
        "packageName": "string",
        "portForwardingConfig": {
          "portMappings": [
            { "applicationPort": number,
              "enableOnPublicIp": boolean,
              "jobPort": number
            }
          ],
        },
        "streamUI": boolean
      }
    },
  ],
  "simulationApplications": [
    { "application": "string",
      "applicationVersion": "string",
      "launchConfig": {
        "environmentVariables": {
          "string": "string"
        },
        "launchFile": "string",
      }
    }]
}
```
CreateSimulationJob

```

"packageName": "string",
"portForwardingConfig": {
    "portMappings": [
      {
        "applicationPort": number,
        "enableOnPublicIp": boolean,
        "jobPort": number
      }
    ],

    "streamUI": boolean
  },

"tags": {
    ": ": 
  },

"vpcConfig": {
    "assignPublicIp": boolean,
    "securityGroups": [ "string" ],
    "subnets": [ "string" ]
}
```

**URI Request Parameters**

The request does not use any URI parameters.

**Request Body**

The request accepts the following data in JSON format.

**clientRequestToken (p. 178)**

Unique, case-sensitive identifier that you provide to ensure the idempotency of the request.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 64.

Pattern: [a-zA-Z0-9-_\-=]*

Required: No

**dataSources (p. 178)**

Specify data sources to mount read-only files from S3 into your simulation. These files are available under /opt/robomaker/datasources/data_source_name.

**Note**

There is a limit of 100 files and a combined size of 25GB for all DataSourceConfig objects.

Type: Array of DataSourceConfig (p. 291) objects

Array Members: Minimum number of 1 item. Maximum number of 5 items.

Required: No

**failureBehavior (p. 178)**

The failure behavior the simulation job.
Continue

Restart the simulation job in the same host instance.

Fail

Stop the simulation job and terminate the instance.

Type: String

Valid Values: Fail | Continue

Required: No

`iamRole (p. 178)`

The IAM role name that allows the simulation instance to call the AWS APIs that are specified in its associated policies on your behalf. This is how credentials are passed in to your simulation job.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 255.

Pattern: `arn:aws:iam::\w+:role/.*`

Required: Yes

`loggingConfig (p. 178)`

The logging configuration.

Type: `LoggingConfig (p. 305)` object

Required: No

`maxJobDurationInSeconds (p. 178)`

The maximum simulation job duration in seconds (up to 14 days or 1,209,600 seconds. When `maxJobDurationInSeconds` is reached, the simulation job will status will transition to Completed.

Type: Long

Required: Yes

`outputLocation (p. 178)`

Location for output files generated by the simulation job.

Type: `OutputLocation (p. 307)` object

Required: No

`robotApplications (p. 178)`

The robot application to use in the simulation job.

Type: Array of `RobotApplicationConfig (p. 315)` objects

Array Members: Fixed number of 1 item.

Required: No

`simulationApplications (p. 178)`

The simulation application to use in the simulation job.

Type: Array of `SimulationApplicationConfig (p. 323)` objects
Array Members: Fixed number of 1 item.

Required: No

**tags (p. 178)**

A map that contains tag keys and tag values that are attached to the simulation job.

Type: String to string map

Key Length Constraints: Minimum length of 1. Maximum length of 128.

Key Pattern: [a-zA-Z0-9 _\-\./+=:]*

Value Length Constraints: Minimum length of 0. Maximum length of 256.

Value Pattern: [a-zA-Z0-9 _\-\./+=:]*

Required: No

**vpcConfig (p. 178)**

If your simulation job accesses resources in a VPC, you provide this parameter identifying the list of security group IDs and subnet IDs. These must belong to the same VPC. You must provide at least one security group and one subnet ID.

Type: VPCConfig (p. 340) object

Required: No

**Response Syntax**

```json
HTTP/1.1 200
Content-type: application/json

{
  "arn": "string",
  "clientRequestToken": "string",
  "dataSources": [
    {
      "name": "string",
      "s3Bucket": "string",
      "s3Keys": [
        {
          "etag": "string",
          "s3Key": "string"
        }
      ]
    }
  ],
  "failureBehavior": "string",
  "failureCode": "string",
  "iamRole": "string",
  "lastStartedAt": number,
  "lastUpdatedAt": number,
  "loggingConfig": {
    "recordAllRosTopics": boolean
  },
  "maxJobDurationInSeconds": number,
  "outputLocation": {
    "s3Bucket": "string",
    "s3Prefix": "string"
  }
}
```
"robotApplications": [
    {
        "application": "string",
        "applicationVersion": "string",
        "launchConfig": {
            "environmentVariables": {
                "string": "string"
            },
            "launchFile": "string",
            "packageName": "string",
            "portForwardingConfig": {
                "portMappings": [
                    {
                        "applicationPort": number,
                        "enableOnPublicIp": boolean,
                        "jobPort": number
                    }
                ],
                "streamUI": boolean
            }
        }
    },
    "simulationApplications": [
        {
            "application": "string",
            "applicationVersion": "string",
            "launchConfig": {
                "environmentVariables": {
                    "string": "string"
                },
                "launchFile": "string",
                "packageName": "string",
                "portForwardingConfig": {
                    "portMappings": [
                        {
                            "applicationPort": number,
                            "enableOnPublicIp": boolean,
                            "jobPort": number
                        }
                    ],
                    "streamUI": boolean
                }
            }
        },
        "simulationTimeMillis": number,
        "status": "string",
        "tags": {
            "string": "string"
        }
    },
    "vpcConfig": {
        "assignPublicIp": boolean,
        "securityGroups": [ "string" ],
        "subnets": [ "string" ],
        "vpcId": "string"
    }
]}

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.
arn (p. 181)

The Amazon Resource Name (ARN) of the simulation job.
Type: String
Pattern: arn:.*

clientRequestToken (p. 181)

Unique, case-sensitive identifier that you provide to ensure the idempotency of the request.
Type: String
Length Constraints: Minimum length of 1. Maximum length of 64.
Pattern: [a-zA-Z0-9_-=]*

dataSources (p. 181)

The data sources for the simulation job.
Type: Array of DataSource (p. 290) objects

failureBehavior (p. 181)

the failure behavior for the simulation job.
Type: String
Valid Values: Fail | Continue

failureCode (p. 181)

The failure code of the simulation job if it failed:
InternalServiceError
    Internal service error.
RobotApplicationCrash
    Robot application exited abnormally.
SimulationApplicationCrash
    Simulation application exited abnormally.
BadPermissionsRobotApplication
    Robot application bundle could not be downloaded.
BadPermissionsSimulationApplication
    Simulation application bundle could not be downloaded.
BadPermissionsS3Output
    Unable to publish outputs to customer-provided S3 bucket.
BadPermissionsCloudwatchLogs
    Unable to publish logs to customer-provided CloudWatch Logs resource.
SubnetIpLimitExceeded
    Subnet IP limit exceeded.
ENILimitExceeded

ENI limit exceeded.

BadPermissionsUserCredentials

Unable to use the Role provided.

InvalidBundleRobotApplication

Robot bundle cannot be extracted (invalid format, bundling error, or other issue).

InvalidBundleSimulationApplication

Simulation bundle cannot be extracted (invalid format, bundling error, or other issue).

RobotApplicationVersionMismatchedEtag

Etag for RobotApplication does not match value during version creation.

SimulationApplicationVersionMismatchedEtag

Etag for SimulationApplication does not match value during version creation.

Type: String

Valid Values: InternalServiceError | RobotApplicationCrash | SimulationApplicationCrash | BadPermissionsRobotApplication | BadPermissionsSimulationApplication | BadPermissionsS3Object | BadPermissionsS3Output | BadPermissionsCloudwatchLogs | SubnetIpLimitExceeded | ENILimitExceeded | BadPermissionsUserCredentials | InvalidBundleRobotApplication | InvalidBundleSimulationApplication | InvalidS3Resource | LimitExceeded | MismatchedEtag | RobotApplicationVersionMismatchedEtag | SimulationApplicationVersionMismatchedEtag | ResourceNotFoundException | RequestThrottled | BatchTimedOut | BatchCanceled | InvalidInput | WrongRegionS3Bucket | WrongRegionS3Output | WrongRegionRobotApplication | WrongRegionSimulationApplication

iamRole (p. 181)

The IAM role that allows the simulation job to call the AWS APIs that are specified in its associated policies on your behalf.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 255.

Pattern: arn:aws:iam::\w+:role/.*

lastStartedAt (p. 181)

The time, in milliseconds since the epoch, when the simulation job was last started.

Type: Timestamp

lastUpdatedAt (p. 181)

The time, in milliseconds since the epoch, when the simulation job was last updated.

Type: Timestamp

loggingConfig (p. 181)

The logging configuration.

Type: LoggingConfig (p. 305) object
maxJobDurationInSeconds (p. 181)

The maximum simulation job duration in seconds.

Type: Long

outputLocation (p. 181)

Simulation job output files location.

Type: OutputLocation (p. 307) object

robotApplications (p. 181)

The robot application used by the simulation job.

Type: Array of RobotApplicationConfig (p. 315) objects

Array Members: Fixed number of 1 item.

simulationApplications (p. 181)

The simulation application used by the simulation job.

Type: Array of SimulationApplicationConfig (p. 323) objects

Array Members: Fixed number of 1 item.

simulationTimeMillis (p. 181)

The simulation job execution duration in milliseconds.

Type: Long

status (p. 181)

The status of the simulation job.

Type: String

Valid Values: Pending | Preparing | Running | Restarting | Completed | Failed | RunningFailed | Terminating | Terminated | Canceled

tags (p. 181)

The list of all tags added to the simulation job.

Type: String to string map

Key Length Constraints: Minimum length of 1. Maximum length of 128.

Key Pattern: \[a-zA-Z0-9 _-\-\+/=:]*

Value Length Constraints: Minimum length of 0. Maximum length of 256.

Value Pattern: \[a-zA-Z0-9 _-\-\+/=:]*

vpcConfig (p. 181)

Information about the vpc configuration.

Type: VPCCfgResponse (p. 341) object

Errors

For information about the errors that are common to all actions, see Common Errors (p. 342).
IdempotentParameterMismatchException

The request uses the same client token as a previous, but non-identical request. Do not reuse a client token with different requests, unless the requests are identical.

HTTP Status Code: 400

InternalServerException

AWS RoboMaker experienced a service issue. Try your call again.

HTTP Status Code: 500

InvalidParameterException

A parameter specified in a request is not valid, is unsupported, or cannot be used. The returned message provides an explanation of the error value.

HTTP Status Code: 400

LimitExceededException

The requested resource exceeds the maximum number allowed, or the number of concurrent stream requests exceeds the maximum number allowed.

HTTP Status Code: 400

ResourceNotFoundException

The specified resource does not exist.

HTTP Status Code: 400

ServiceUnavailableException

The request has failed due to a temporary failure of the server.

HTTP Status Code: 503

ThrottlingException

AWS RoboMaker is temporarily unable to process the request. Try your call again.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3
DeleteFleet

Deletes a fleet.

Request Syntax

```
POST /deleteFleet HTTP/1.1
Content-type: application/json
{
  "fleet": "string"
}
```

URI Request Parameters

The request does not use any URI parameters.

Request Body

The request accepts the following data in JSON format.

fleet (p. 187)

  The Amazon Resource Name (ARN) of the fleet.
  
  Type: String
  
  
  Pattern: arn:.*
  
  Required: Yes

Response Syntax

```
HTTP/1.1 200
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response with an empty HTTP body.

Errors

For information about the errors that are common to all actions, see Common Errors (p. 342).

InternalServerException

  AWS RoboMaker experienced a service issue. Try your call again.
  
  HTTP Status Code: 500

InvalidParameterException

  A parameter specified in a request is not valid, is unsupported, or cannot be used. The returned message provides an explanation of the error value.
HTTP Status Code: 400

**ThrottlingException**

AWS RoboMaker is temporarily unable to process the request. Try your call again.

HTTP Status Code: 400

**See Also**

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)
DeleteRobot

Deletes a robot.

Request Syntax

POST /deleteRobot HTTP/1.1
Content-type: application/json

{   "robot": "string"
}

URI Request Parameters

The request does not use any URI parameters.

Request Body

The request accepts the following data in JSON format.

robot (p. 189)

The Amazon Resource Name (ARN) of the robot.

Type: String


Pattern: arn:.*

Required: Yes

Response Syntax

HTTP/1.1 200

Response Elements

If the action is successful, the service sends back an HTTP 200 response with an empty HTTP body.

Errors

For information about the errors that are common to all actions, see Common Errors (p. 342).

InternalServerException

AWS RoboMaker experienced a service issue. Try your call again.

HTTP Status Code: 500

InvalidParameterException

A parameter specified in a request is not valid, is unsupported, or cannot be used. The returned message provides an explanation of the error value.
HTTP Status Code: 400

**ThrottlingException**

AWS RoboMaker is temporarily unable to process the request. Try your call again.

HTTP Status Code: 400

**See Also**

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3
DeleteRobotApplication

Deletes a robot application.

Request Syntax

```plaintext
POST /deleteRobotApplication HTTP/1.1
Content-type: application/json

{
  "application": "string",
  "applicationVersion": "string"
}
```

URI Request Parameters

The request does not use any URI parameters.

Request Body

The request accepts the following data in JSON format.

application (p. 191)

The Amazon Resource Name (ARN) of the the robot application.

Type: String


Pattern: arn:.*

Required: Yes

applicationVersion (p. 191)

The version of the robot application to delete.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 255.

Pattern: (\$LATEST)|[0-9]*

Required: No

Response Syntax

HTTP/1.1 200

Response Elements

If the action is successful, the service sends back an HTTP 200 response with an empty HTTP body.

Errors

For information about the errors that are common to all actions, see Common Errors (p. 342).
InternalServerException

AWS RoboMaker experienced a service issue. Try your call again.

HTTP Status Code: 500

InvalidParameterException

A parameter specified in a request is not valid, is unsupported, or cannot be used. The returned message provides an explanation of the error value.

HTTP Status Code: 400

ThrottlingException

AWS RoboMaker is temporarily unable to process the request. Try your call again.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3
DeleteSimulationApplication

Deletes a simulation application.

Request Syntax

```
POST /deleteSimulationApplication HTTP/1.1
Content-type: application/json
{
  "application": "string",
  "applicationVersion": "string"
}
```

URI Request Parameters

The request does not use any URI parameters.

Request Body

The request accepts the following data in JSON format.

**application** (p. 193)

The application information for the simulation application to delete.

Type: String


Pattern: arn:.*

Required: Yes

**applicationVersion** (p. 193)

The version of the simulation application to delete.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 255.

Pattern: (\$LATEST)|[0-9]*

Required: No

Response Syntax

```
HTTP/1.1 200
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response with an empty HTTP body.

Errors

For information about the errors that are common to all actions, see Common Errors (p. 342).
**InternalServerException**

AWS RoboMaker experienced a service issue. Try your call again.

HTTP Status Code: 500

**InvalidParameterException**

A parameter specified in a request is not valid, is unsupported, or cannot be used. The returned message provides an explanation of the error value.

HTTP Status Code: 400

**ThrottlingException**

AWS RoboMaker is temporarily unable to process the request. Try your call again.

HTTP Status Code: 400

**See Also**

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3
DeregisterRobot

Deregisters a robot.

Request Syntax

POST /deregisterRobot HTTP/1.1
Content-type: application/json

{
  "fleet": "string",
  "robot": "string"
}

URI Request Parameters

The request does not use any URI parameters.

Request Body

The request accepts the following data in JSON format.

fleet (p. 195)

The Amazon Resource Name (ARN) of the fleet.

Type: String


Pattern: arn:.*

Required: Yes

robot (p. 195)

The Amazon Resource Name (ARN) of the robot.

Type: String


Pattern: arn:.*

Required: Yes

Response Syntax

HTTP/1.1 200
Content-type: application/json

{
  "fleet": "string",
  "robot": "string"
}
Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

fleet (p. 195)

The Amazon Resource Name (ARN) of the fleet.

Type: String


Pattern: arn:.*

robot (p. 195)

The Amazon Resource Name (ARN) of the robot.

Type: String


Pattern: arn:.*

Errors

For information about the errors that are common to all actions, see Common Errors (p. 342).

InternalServerException

AWS RoboMaker experienced a service issue. Try your call again.

HTTP Status Code: 500

InvalidParameterException

A parameter specified in a request is not valid, is unsupported, or cannot be used. The returned message provides an explanation of the error value.

HTTP Status Code: 400

ResourceNotFoundException

The specified resource does not exist.

HTTP Status Code: 400

ThrottlingException

AWS RoboMaker is temporarily unable to process the request. Try your call again.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
• AWS SDK for C++
• AWS SDK for Go
• AWS SDK for Java
• AWS SDK for JavaScript
• AWS SDK for PHP V3
• AWS SDK for Python
• AWS SDK for Ruby V3
DescribeDeploymentJob

Describes a deployment job.

Request Syntax

POST /describeDeploymentJob HTTP/1.1
Content-type: application/json

{
  "job": "string"
}

URI Request Parameters

The request does not use any URI parameters.

Request Body

The request accepts the following data in JSON format.

job (p. 198)

The Amazon Resource Name (ARN) of the deployment job.

Type: String


Pattern: arn:.*

Required: Yes

Response Syntax

HTTP/1.1 200
Content-type: application/json

{
  "arn": "string",
  "createdAt": number,
  "deploymentApplicationConfigs": [
    {
      "application": "string",
      "applicationVersion": "string",
      "launchConfig": {
        "environmentVariables": {
          "string": "string"
        },
        "launchFile": "string",
        "packageName": "string",
        "postLaunchFile": "string",
        "preLaunchFile": "string"
      }
    }
  ],
  "deploymentConfig": {
    "concurrentDeploymentPercentage": number,
  }
}
"downloadConditionFile": {  
  "bucket": "string",  
  "etag": "string",  
  "key": "string"  
},  
"failureThresholdPercentage": number,  
"robotDeploymentTimeoutInSeconds": number  
},  
"failureCode": "string",  
"failureReason": "string",  
"fleet": "string",  
"robotDeploymentSummary": [  
  {  
    "arn": "string",  
    "deploymentFinishTime": number,  
    "deploymentStartTime": number,  
    "failureCode": "string",  
    "failureReason": "string",  
    "progressDetail": {  
      "currentProgress": "string",  
      "estimatedTimeRemainingSeconds": number,  
      "percentDone": number,  
      "targetResource": "string"  
    },  
    "status": "string"  
  }  
],  
"status": "string",  
"tags": {  
  "string": "string"  
}  
}

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

**arn (p. 198)**

The Amazon Resource Name (ARN) of the deployment job.

Type: String


Pattern: arn:.*

**createdAt (p. 198)**

The time, in milliseconds since the epoch, when the deployment job was created.

Type: Timestamp

**deploymentApplicationConfigs (p. 198)**

The deployment application configuration.

Type: Array of DeploymentApplicationConfig (p. 292) objects

Array Members: Fixed number of 1 item.

**deploymentConfig (p. 198)**

The deployment configuration.
**Type: **DeploymentConfig (p. 293) object

**failureCode (p. 198)**

The deployment job failure code.

Type: String


**failureReason (p. 198)**

A short description of the reason why the deployment job failed.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 1024.

Pattern: .*

**fleet (p. 198)**

The Amazon Resource Name (ARN) of the fleet.

Type: String


Pattern: arn:.*

**robotDeploymentSummary (p. 198)**

A list of robot deployment summaries.

Type: Array of RobotDeployment (p. 318) objects

**status (p. 198)**

The status of the deployment job.

Type: String

Valid Values: Pending | Preparing | InProgress | Failed | Succeeded | Canceled

**tags (p. 198)**

The list of all tags added to the specified deployment job.

Type: String to string map

Key Length Constraints: Minimum length of 1. Maximum length of 128.

Key Pattern: [a-zA-Z0-9 _.-\/%+=:]*

Value Length Constraints: Minimum length of 0. Maximum length of 256.

Value Pattern: [a-zA-Z0-9 _.-\/+=:]*
Errors

For information about the errors that are common to all actions, see Common Errors (p. 342).

InternalServerException

AWS RoboMaker experienced a service issue. Try your call again.

HTTP Status Code: 500

InvalidParameterException

A parameter specified in a request is not valid, is unsupported, or cannot be used. The returned message provides an explanation of the error value.

HTTP Status Code: 400

ResourceNotFoundException

The specified resource does not exist.

HTTP Status Code: 400

ThrottlingException

AWS RoboMaker is temporarily unable to process the request. Try your call again.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3
DescribeFleet

Describes a fleet.

Request Syntax

POST /describeFleet HTTP/1.1
Content-type: application/json

{  "fleet": "string"
}

URI Request Parameters

The request does not use any URI parameters.

Request Body

The request accepts the following data in JSON format.

fleet (p. 202)

The Amazon Resource Name (ARN) of the fleet.

Type: String


Pattern: arn:.*

Required: Yes

Response Syntax

HTTP/1.1 200
Content-type: application/json

{  "arn": "string",
   "createdAt": number,
   "lastDeploymentJob": "string",
   "lastDeploymentStatus": "string",
   "lastDeploymentTime": number,
   "name": "string",
   "robots": [
      {
         "architecture": "string",
         "arn": "string",
         "createdAt": number,
         "fleetArn": "string",
         "greenGrassGroupId": "string",
         "lastDeploymentJob": "string",
         "lastDeploymentTime": number,
         "name": "string",
         "status": "string"
      }
   ],"string"
Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

**arn (p. 202)**

The Amazon Resource Name (ARN) of the fleet.

Type: String


Pattern: arn:.*

**createdAt (p. 202)**

The time, in milliseconds since the epoch, when the fleet was created.

Type: Timestamp

**lastDeploymentJob (p. 202)**

The Amazon Resource Name (ARN) of the last deployment job.

Type: String


Pattern: arn:.*

**lastDeploymentStatus (p. 202)**

The status of the last deployment.

Type: String

Valid Values: Pending | Preparing | InProgress | Failed | Succeeded | Canceled

**lastDeploymentTime (p. 202)**

The time of the last deployment.

Type: Timestamp

**name (p. 202)**

The name of the fleet.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 255.

Pattern: [a-zA-Z0-9-\_\-]*

**robots (p. 202)**

A list of robots.

Type: Array of Robot (p. 313) objects
Array Members: Minimum number of 0 items. Maximum number of 1000 items.

tags (p. 202)

The list of all tags added to the specified fleet.

Type: String to string map

Key Length Constraints: Minimum length of 1. Maximum length of 128.

Key Pattern: [a-zA-Z0-9 \-_\.\-\+/=\:]*

Value Length Constraints: Minimum length of 0. Maximum length of 256.

Value Pattern: [a-zA-Z0-9 \-_\.\-\+/=\:]*

Errors

For information about the errors that are common to all actions, see Common Errors (p. 342).

InternalServerException

AWS RoboMaker experienced a service issue. Try your call again.

HTTP Status Code: 500

InvalidParameterException

A parameter specified in a request is not valid, is unsupported, or cannot be used. The returned message provides an explanation of the error value.

HTTP Status Code: 400

ResourceNotFoundException

The specified resource does not exist.

HTTP Status Code: 400

ThrottlingException

AWS RoboMaker is temporarily unable to process the request. Try your call again.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3
DescribeRobot

Describes a robot.

Request Syntax

POST /describeRobot HTTP/1.1
Content-type: application/json

{  "robot": "string"
}

URI Request Parameters

The request does not use any URI parameters.

Request Body

The request accepts the following data in JSON format.

robot (p. 206)

The Amazon Resource Name (ARN) of the robot to be described.

Type: String


Pattern: arn:.*

Required: Yes

Response Syntax

HTTP/1.1 200
Content-type: application/json

{  "architecture": "string",
  "arn": "string",
  "createdAt": number,
  "fleetArn": "string",
  "greengrassGroupId": "string",
  "lastDeploymentJob": "string",
  "lastDeploymentTime": number,
  "name": "string",
  "status": "string",
  "tags": {  
    "string" : "string"
  }
}

Response Elements

If the action is successful, the service sends back an HTTP 200 response.
The following data is returned in JSON format by the service.

**architecture (p. 206)**

The target architecture of the robot application.

Type: String

Valid Values: X86_64 | ARM64 | ARMHF

**arn (p. 206)**

The Amazon Resource Name (ARN) of the robot.

Type: String


Pattern:arn:.*

**createdAt (p. 206)**

The time, in milliseconds since the epoch, when the robot was created.

Type: Timestamp

**fleetArn (p. 206)**

The Amazon Resource Name (ARN) of the fleet.

Type: String


Pattern:arn:.*

**greengrassGroupId (p. 206)**

The Greengrass group id.

Type: String


Pattern:. *

**lastDeploymentJob (p. 206)**

The Amazon Resource Name (ARN) of the last deployment job.

Type: String


Pattern:arn:.*

**lastDeploymentTime (p. 206)**

The time of the last deployment job.

Type: Timestamp

**name (p. 206)**

The name of the robot.

Type: String
Length Constraints: Minimum length of 1. Maximum length of 255.
Pattern: [a-zA-Z0-9_\-]*

**status (p. 206)**

The status of the fleet.
Type: String
Valid Values: Available | Registered | PendingNewDeployment | Deploying | Failed | InSync | NoResponse

**tags (p. 206)**

The list of all tags added to the specified robot.
Type: String to string map
Key Length Constraints: Minimum length of 1. Maximum length of 128.
Key Pattern: [a-zA-Z0-9 _\-\.\_/\+=:\]*
Value Length Constraints: Minimum length of 0. Maximum length of 256.
Value Pattern: [a-zA-Z0-9 _\-\.\_/\+=:\]*

**Errors**

For information about the errors that are common to all actions, see Common Errors (p. 342).

**InternalServerException**

AWS RoboMaker experienced a service issue. Try your call again.
HTTP Status Code: 500

**InvalidParameterException**

A parameter specified in a request is not valid, is unsupported, or cannot be used. The returned message provides an explanation of the error value.
HTTP Status Code: 400

**ResourceNotFoundException**

The specified resource does not exist.
HTTP Status Code: 400

**ThrottlingException**

AWS RoboMaker is temporarily unable to process the request. Try your call again.
HTTP Status Code: 400

**See Also**

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
• AWS SDK for C++
• AWS SDK for Go
• AWS SDK for Java
• AWS SDK for JavaScript
• AWS SDK for PHP V3
• AWS SDK for Python
• AWS SDK for Ruby V3
DescribeRobotApplication

Describes a robot application.

Request Syntax

POST /describeRobotApplication HTTP/1.1
Content-type: application/json

{
  "application": "string",
  "applicationVersion": "string"
}

URI Request Parameters

The request does not use any URI parameters.

Request Body

The request accepts the following data in JSON format.

application (p. 210)

The Amazon Resource Name (ARN) of the robot application.

Type: String


Pattern: arn:.*

Required: Yes

applicationVersion (p. 210)

The version of the robot application to describe.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 255.

Pattern: (\$LATEST)|[0-9]*

Required: No

Response Syntax

HTTP/1.1 200
Content-type: application/json

{
  "arn": "string",
  "lastUpdatedAt": number,
  "name": "string",
  "revisionId": "string",
  "robotSoftwareSuite": {


"name": "string",
"version": "string"
},
"sources": [
{
"architecture": "string",
"etag": "string",
"s3Bucket": "string",
"s3Key": "string"
}
],
"tags": {
"string": "string"
},
"version": "string"
}

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

**arn (p. 210)**

The Amazon Resource Name (ARN) of the robot application.

Type: String


Pattern: arn:.*

**lastUpdatedAt (p. 210)**

The time, in milliseconds since the epoch, when the robot application was last updated.

Type: Timestamp

**name (p. 210)**

The name of the robot application.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 255.

Pattern: [a-zA-Z0-9_\-]*

**revisionId (p. 210)**

The revision id of the robot application.

Type: String


Pattern: [a-zA-Z0-9_\-]*

**robotSoftwareSuite (p. 210)**

The robot software suite (ROS distribution) used by the robot application.

Type: RobotSoftwareSuite (p. 320) object
sources (p. 210)

The sources of the robot application.

Type: Array of Source (p. 338) objects

tags (p. 210)

The list of all tags added to the specified robot application.

Type: String to string map

Key Length Constraints: Minimum length of 1. Maximum length of 128.

Key Pattern: [a-zA-Z0-9 \-_\/:]+*

Value Length Constraints: Minimum length of 0. Maximum length of 256.

Value Pattern: [a-zA-Z0-9 \-_\/:]+*

version (p. 210)

The version of the robot application.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 255.

Pattern: (\$LATEST)|[0-9]*

Errors

For information about the errors that are common to all actions, see Common Errors (p. 342).

InternalServerException

AWS RoboMaker experienced a service issue. Try your call again.

HTTP Status Code: 500

InvalidParameterException

A parameter specified in a request is not valid, is unsupported, or cannot be used. The returned message provides an explanation of the error value.

HTTP Status Code: 400

ResourceNotFoundException

The specified resource does not exist.

HTTP Status Code: 400

ThrottlingException

AWS RoboMaker is temporarily unable to process the request. Try your call again.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:
• AWS Command Line Interface
• AWS SDK for .NET
• AWS SDK for C++
• AWS SDK for Go
• AWS SDK for Java
• AWS SDK for JavaScript
• AWS SDK for PHP V3
• AWS SDK for Python
• AWS SDK for Ruby V3
DescribeSimulationApplication

Describes a simulation application.

Request Syntax

```
POST /describeSimulationApplication HTTP/1.1
Content-type: application/json

{
    "application": "string",
    "applicationVersion": "string"
}
```

URI Request Parameters

The request does not use any URI parameters.

Request Body

The request accepts the following data in JSON format.

application (p. 214)

The application information for the simulation application.

Type: String


Pattern: arn:.*

Required: Yes

applicationVersion (p. 214)

The version of the simulation application to describe.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 255.

Pattern: (\$LATEST)|[0-9]*

Required: No

Response Syntax

```
HTTP/1.1 200
Content-type: application/json

{
    "arn": "string",
    "lastUpdatedAt": number,
    "name": "string",
    "renderingEngine": {
        "name": "string",
        "version": "string"
    }
}
```
DescribeSimulationApplication

},
"revisionId": "string",
"robotSoftwareSuite": {
   "name": "string",
   "version": "string"
},
"simulationSoftwareSuite": {
   "name": "string",
   "version": "string"
},
"sources": [
   {
      "architecture": "string",
      "etag": "string",
      "s3Bucket": "string",
      "s3Key": "string"
   }
],
"tags": {
   "string": "string"
},
"version": "string"

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

**arn (p. 214)**

The Amazon Resource Name (ARN) of the robot simulation application.

Type: String


Pattern: arn:.*

**lastUpdatedAt (p. 214)**

The time, in milliseconds since the epoch, when the simulation application was last updated.

Type: Timestamp

**name (p. 214)**

The name of the simulation application.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 255.

Pattern: [a-zA-Z0-9_\-]*

**renderingEngine (p. 214)**

The rendering engine for the simulation application.

Type: RenderingEngine (p. 312) object

**revisionId (p. 214)**

The revision id of the simulation application.
DescribeSimulationApplication

Type: String


Pattern: [a-zA-Z0-9_.\-]*

**robotSoftwareSuite (p. 214)**

Information about the robot software suite (ROS distribution).

Type: `RobotSoftwareSuite (p. 320)` object

**simulationSoftwareSuite (p. 214)**

The simulation software suite used by the simulation application.

Type: `SimulationSoftwareSuite (p. 337)` object

**sources (p. 214)**

The sources of the simulation application.

Type: Array of `Source (p. 338)` objects

**tags (p. 214)**

The list of all tags added to the specified simulation application.

Type: String to string map

Key Length Constraints: Minimum length of 1. Maximum length of 128.

Key Pattern: [a-zA-Z0-9 _\-./+=:]*

Value Length Constraints: Minimum length of 0. Maximum length of 256.

Value Pattern: [a-zA-Z0-9 _\-./+=:]*

**version (p. 214)**

The version of the simulation application.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 255.

Pattern: (\$LATEST) | [0-9]*

**Errors**

For information about the errors that are common to all actions, see `Common Errors (p. 342)`.

**InternalServerException**

AWS RoboMaker experienced a service issue. Try your call again.

HTTP Status Code: 500

**InvalidParameterValue**

A parameter specified in a request is not valid, is unsupported, or cannot be used. The returned message provides an explanation of the error value.

HTTP Status Code: 400
ResourceNotFoundException

The specified resource does not exist.

HTTP Status Code: 400

ThrottlingException

AWS RoboMaker is temporarily unable to process the request. Try your call again.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3
DescribeSimulationJob

Describes a simulation job.

Request Syntax

POST /describeSimulationJob HTTP/1.1
Content-type: application/json

{  "job": "string"
}

URI Request Parameters

The request does not use any URI parameters.

Request Body

The request accepts the following data in JSON format.

job (p. 218)

The Amazon Resource Name (ARN) of the simulation job to be described.

Type: String


Pattern: arn:*

Required: Yes

Response Syntax

HTTP/1.1 200
Content-type: application/json

{  "arn": "string",
   "clientRequestToken": "string",
   "dataSources": [  {
      "name": "string",
      "s3Bucket": "string",
      "s3Keys": [  {
         "etag": "string",
         "s3Key": "string"
      }
    ]
  },
  "failureBehavior": "string",
  "failureCode": "string",
  "failureReason": "string",
  "iamRole": "string"}
"lastStartedAt": number,
"lastUpdatedAt": number,
"loggingConfig": {
   "recordAllRosTopics": boolean
 },
"maxJobDurationInSeconds": number,
"name": "string",
"networkInterface": {
   "networkInterfaceId": "string",
   "privateIpAddress": "string",
   "publicIpAddress": "string"
 },
"outputLocation": {
   "s3Bucket": "string",
   "s3Prefix": "string"
 },
"robotApplications": [
   {
      "application": "string",
      "applicationVersion": "string",
      "launchConfig": {
         "environmentVariables": {
            "string": "string"
         },
         "launchFile": "string",
         "packageName": "string",
         "portForwardingConfig": {
            "portMappings": [
               {
                  "applicationPort": number,
                  "enableOnPublicIp": boolean,
                  "jobPort": number
               }
            ],
            "streamUI": boolean
         }
      }
   }
},
"simulationApplications": [
   {
      "application": "string",
      "applicationVersion": "string",
      "launchConfig": {
         "environmentVariables": {
            "string": "string"
         },
         "launchFile": "string",
         "packageName": "string",
         "portForwardingConfig": {
            "portMappings": [
               {
                  "applicationPort": number,
                  "enableOnPublicIp": boolean,
                  "jobPort": number
               }
            ],
            "streamUI": boolean
         }
      }
   },
   {
      "application": "string",
      "applicationVersion": "string",
      "launchConfig": {
         "environmentVariables": {
            "string": "string"
         },
         "launchFile": "string",
         "packageName": "string",
         "portForwardingConfig": {
            "portMappings": [
               {
                  "applicationPort": number,
                  "enableOnPublicIp": boolean,
                  "jobPort": number
               }
            ],
            "streamUI": boolean
         }
      }
   }
},
"simulationTimeMillis": number,
"status": "string",
"tags": {
   "string": "string"
"vpcConfig": {
    "assignPublicIp": boolean,
    "securityGroups": [ "string" ],
    "subnets": [ "string" ],
    "vpcId": "string"
}

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

**arn (p. 218)**

The Amazon Resource Name (ARN) of the simulation job.

Type: String


Pattern: arn:.*

**clientRequestToken (p. 218)**

Unique, case-sensitive identifier that you provide to ensure the idempotency of the request.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 64.

Pattern: [a-zA-Z0-9-_\-=]*

**dataSources (p. 218)**

The data sources for the simulation job.

Type: Array of DataSource (p. 290) objects

**failureBehavior (p. 218)**

The failure behavior for the simulation job.

Type: String

Valid Values: Fail | Continue

**failureCode (p. 218)**

The failure code of the simulation job if it failed:

- InternalServiceError
  - Internal service error.
- RobotApplicationCrash
  - Robot application exited abnormally.
- SimulationApplicationCrash
  - Simulation application exited abnormally.
BadPermissionsRobotApplication
  Robot application bundle could not be downloaded.
BadPermissionsSimulationApplication
  Simulation application bundle could not be downloaded.
BadPermissionsS3Output
  Unable to publish outputs to customer-provided S3 bucket.
BadPermissionsCloudwatchLogs
  Unable to publish logs to customer-provided CloudWatch Logs resource.
SubnetIpLimitExceeded
  Subnet IP limit exceeded.
ENILimitExceeded
  ENI limit exceeded.
BadPermissionsUserCredentials
  Unable to use the Role provided.
InvalidBundleRobotApplication
  Robot bundle cannot be extracted (invalid format, bundling error, or other issue).
InvalidBundleSimulationApplication
  Simulation bundle cannot be extracted (invalid format, bundling error, or other issue).
RobotApplicationVersionMismatchedEtag
  Etag for RobotApplication does not match value during version creation.
SimulationApplicationVersionMismatchedEtag
  Etag for SimulationApplication does not match value during version creation.

Type: String

Valid Values: InternalServiceError | RobotApplicationCrash | SimulationApplicationCrash | BadPermissionsRobotApplication | BadPermissionsSimulationApplication | BadPermissionsS3Object | BadPermissionsS3Output | BadPermissionsCloudwatchLogs | SubnetIpLimitExceeded | ENILimitExceeded | BadPermissionsUserCredentials | InvalidBundleRobotApplication | InvalidBundleSimulationApplication | InvalidS3Resource | LimitExceeded | MismatchedEtag | RobotApplicationVersionMismatchedEtag | SimulationApplicationVersionMismatchedEtag | ResourceNotFound | RequestThrottled | BatchTimedOut | BatchCanceled | InvalidInput | WrongRegionS3Bucket | WrongRegionS3Output | WrongRegionRobotApplication | WrongRegionSimulationApplication

failureReason (p. 218)

Details about why the simulation job failed. For more information about troubleshooting, see Troubleshooting.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 1024.

Pattern: . *
iamRole (p. 218)

The IAM role that allows the simulation instance to call the AWS APIs that are specified in its associated policies on your behalf.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 255.


lastStartedAt (p. 218)

The time, in milliseconds since the epoch, when the simulation job was last started.

Type: Timestamp

lastUpdatedAt (p. 218)

The time, in milliseconds since the epoch, when the simulation job was last updated.

Type: Timestamp

loggingConfig (p. 218)

The logging configuration.

Type: LoggingConfig (p. 305) object

maxJobDurationInSeconds (p. 218)

The maximum job duration in seconds. The value must be 8 days (691,200 seconds) or less.

Type: Long

name (p. 218)

The name of the simulation job.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 255.

Pattern: [a-zA-Z0-9-\_\-]*

networkInterface (p. 218)

The network interface information for the simulation job.

Type: NetworkInterface (p. 306) object

outputLocation (p. 218)

Location for output files generated by the simulation job.

Type: OutputLocation (p. 307) object

robotApplications (p. 218)

A list of robot applications.

Type: Array of RobotApplicationConfig (p. 315) objects

Array Members: Fixed number of 1 item.

simulationApplications (p. 218)

A list of simulation applications.
Type: Array of `SimulationApplicationConfig` objects

Array Members: Fixed number of 1 item.

`simulationTimeMillis` (p. 218)

The simulation job execution duration in milliseconds.

Type: Long

`status` (p. 218)

The status of the simulation job.

Type: String

Valid Values: Pending | Preparing | Running | Restarting | Completed | Failed | RunningFailed | Terminating | Terminated | Canceled

`tags` (p. 218)

The list of all tags added to the specified simulation job.

Type: String to string map

Key Length Constraints: Minimum length of 1. Maximum length of 128.

Key Pattern: `[a-zA-Z0-9\-_\./+=:]*`

Value Length Constraints: Minimum length of 0. Maximum length of 256.

Value Pattern: `[a-zA-Z0-9\-_\./+=:]*`

`vpcConfig` (p. 218)

The VPC configuration.

Type: `VPCConfigResponse` object

Errors

For information about the errors that are common to all actions, see `Common Errors` (p. 342).

`InternalServerException`

AWS RoboMaker experienced a service issue. Try your call again.

HTTP Status Code: 500

`InvalidParameterException`

A parameter specified in a request is not valid, is unsupported, or cannot be used. The returned message provides an explanation of the error value.

HTTP Status Code: 400

`ResourceNotFoundException`

The specified resource does not exist.

HTTP Status Code: 400

`ThrottlingException`

AWS RoboMaker is temporarily unable to process the request. Try your call again.
HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3
DescribeSimulationJobBatch

Describes a simulation job batch.

Request Syntax

POST /describeSimulationJobBatch HTTP/1.1
Content-type: application/json

{  "batch": "string"}

URI Request Parameters

The request does not use any URI parameters.

Request Body

The request accepts the following data in JSON format.

batch (p. 225)

The id of the batch to describe.

Type: String


Pattern: arn:.*

Required: Yes

Response Syntax

HTTP/1.1 200
Content-type: application/json

{  "arn": "string",
   "batchPolicy": {
      "maxConcurrency": number,
      "timeoutInSeconds": number
   },
   "clientRequestToken": "string",
   "createdAt": number,
   "createdRequests": [
      {  "arn": "string",
         "dataSourceNames": [ "string" ],
         "lastUpdatedAt": number,
         "name": "string",
         "robotApplicationNames": [ "string" ],
         "simulationApplicationNames": [ "string" ],
         "status": "string"
      }
   ],
"failedRequests": [
  {
    "failedAt": number,
    "failureCode": "string",
    "failureReason": "string",
    "request": {
      "dataSources": [
        {
          "name": "string",
          "s3Bucket": "string",
          "s3Keys": [ "string" ]
        }
      ],
      "failureBehavior": "string",
      "iamRole": "string",
      "loggingConfig": {
        "recordAllRosTopics": boolean
      },
      "maxJobDurationInSeconds": number,
      "outputLocation": {
        "s3Bucket": "string",
        "s3Prefix": "string"
      },
      "robotApplications": [
        {
          "application": "string",
          "applicationVersion": "string",
          "launchConfig": {
            "environmentVariables": {
              "string": "string"
            },
            "launchFile": "string",
            "packageName": "string",
            "portForwardingConfig": {
              "portMappings": [
                {
                  "applicationPort": number,
                  "enableOnPublicIp": boolean,
                  "jobPort": number
                }
              ],
              "streamUI": boolean
            }
          }
        }
      ],
      "simulationApplications": [
        {
          "application": "string",
          "applicationVersion": "string",
          "launchConfig": {
            "environmentVariables": {
              "string": "string"
            },
            "launchFile": "string",
            "packageName": "string",
            "portForwardingConfig": {
              "portMappings": [
                {
                  "applicationPort": number,
                  "enableOnPublicIp": boolean,
                  "jobPort": number
                }
              ],
              "streamUI": boolean
            }
          }
        }
      ]
    }
  ]
{ "jobIds": [ { "jobId": "string", "failureCode": "string", "failureReason": "string", "lastUpdatedAt": number, "pendingRequests": [ { "dataSources": [ { "name": "string", "s3Bucket": "string", "s3Keys": [ "string" ] } ], "failureBehavior": "string", "iamRole": "string", "loggingConfig": { "recordAllRosTopics": boolean }, "maxJobDurationInSeconds": number, "outputLocation": { "s3Bucket": "string", "s3Prefix": "string" }, "robotApplications": [ { "application": "string", "applicationVersion": "string", "launchConfig": { "environmentVariables": { "string": "string" } }, "launchFile": "string", "packageName": "string", "portForwardingConfig": { "portMappings": [ { "applicationPort": number, "enableOnPublicIp": boolean, "jobPort": number } ] }, "streamUI": boolean } ] }, "simulationApplications": [ { "application": "string", "applicationVersion": "string", "launchConfig": { "environmentVariables": { ... } } ] } ] }, "tags": { "string": "string" }, "useDefaultApplications": boolean, "vpcConfig": { "assignPublicIp": boolean, "securityGroups": [ "string" ], "subnets": [ "string" ] } } }
"string" : "string"
},
"launchFile": "string",
"packageName": "string",
"portForwardingConfig": {
  "portMappings": [
    {
      "applicationPort": number,
      "enableOnPublicIp": boolean,
      "jobPort": number
    }
  ]
},
"streamUI": boolean
}
],
"tags": {
  "string" : "string"
},
"useDefaultApplications": boolean,
"vpcConfig": {
  "assignPublicIp": boolean,
  "securityGroups": [ "string" ],
  "subnets": [ "string" ]
}
],
"status": "string",
"tags": {
  "string" : "string"
}
}

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

arn (p. 225)

The Amazon Resource Name (ARN) of the batch.

Type: String


Pattern: arn:.*

batchPolicy (p. 225)

The batch policy.

Type: BatchPolicy (p. 289) object

clientRequestToken (p. 225)

Unique, case-sensitive identifier that you provide to ensure the idempotency of the request.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 64.
DescribeSimulationJobBatch

**Pattern:** [a-zA-Z0-9_\-\=\*]*

**createdAt (p. 225)**

The time, in milliseconds since the epoch, when the simulation job batch was created.

Type: Timestamp

**createdRequests (p. 225)**

A list of created simulation job summaries.

Type: Array of `SimulationJobSummary (p. 335)` objects

Array Members: Minimum number of 0 items. Maximum number of 100 items.

**failedRequests (p. 225)**

A list of failed create simulation job requests. The request failed to be created into a simulation job. Failed requests do not have a simulation job ID.

Type: Array of `FailedCreateSimulationJobRequest (p. 298)` objects

**failureCode (p. 225)**

The failure code of the simulation job batch.

Type: String

Valid Values: InternalServiceError

**failureReason (p. 225)**

The reason the simulation job batch failed.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 1024.

Pattern: .*

**lastUpdatedAt (p. 225)**

The time, in milliseconds since the epoch, when the simulation job batch was last updated.

Type: Timestamp

**pendingRequests (p. 225)**

A list of pending simulation job requests. These requests have not yet been created into simulation jobs.

Type: Array of `SimulationJobRequest (p. 332)` objects

Array Members: Minimum number of 1 item.

**status (p. 225)**

The status of the batch.

Pending

The simulation job batch request is pending.

InProgress

The simulation job batch is in progress.
DescribeSimulationJobBatch

Failed

The simulation job batch failed. One or more simulation job requests could not be completed due to an internal failure (like InternalServiceError). See failureCode and failureReason for more information.

Completed

The simulation batch job completed. A batch is complete when (1) there are no pending simulation job requests in the batch and none of the failed simulation job requests are due to InternalServiceError and (2) when all created simulation jobs have reached a terminal state (for example, Completed or Failed).

Canceled

The simulation batch job was cancelled.

Canceling

The simulation batch job is being cancelled.

Completing

The simulation batch job is completing.

TimingOut

The simulation job batch is timing out.

If a batch timing out, and there are pending requests that were failing due to an internal failure (like InternalServiceError), the batch status will be Failed. If there are no such failing request, the batch status will be TimedOut.

TimedOut

The simulation batch job timed out.

Type: String

Valid Values: Pending | InProgress | Failed | Completed | Canceled | Canceling | Completing | TimingOut | TimedOut

tags (p. 225)

A map that contains tag keys and tag values that are attached to the simulation job batch.

Type: String to string map

Key Length Constraints: Minimum length of 1. Maximum length of 128.

Key Pattern: [a-zA-Z0-9_\-\./+=:]*

Value Length Constraints: Minimum length of 0. Maximum length of 256.

Value Pattern: [a-zA-Z0-9_\-\+/+=:]*

Errors

For information about the errors that are common to all actions, see Common Errors (p. 342).

InternalServerException

AWS RoboMaker experienced a service issue. Try your call again.

HTTP Status Code: 500
InvalidParameterException

A parameter specified in a request is not valid, is unsupported, or cannot be used. The returned message provides an explanation of the error value.

HTTP Status Code: 400

ResourceNotFoundException

The specified resource does not exist.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3
ListDeploymentJobs

Returns a list of deployment jobs for a fleet. You can optionally provide filters to retrieve specific deployment jobs.

Request Syntax

```
POST /listDeploymentJobs HTTP/1.1
Content-type: application/json

{
  "filters": [
    {
      "name": "string",
      "values": [ "string" ]
    }
  ],
  "maxResults": number,
  "nextToken": "string"
}
```

URI Request Parameters

The request does not use any URI parameters.

Request Body

The request accepts the following data in JSON format.

filters (p. 232)

Optional filters to limit results.

The filter names status and fleetName are supported. When filtering, you must use the complete value of the filtered item. You can use up to three filters, but they must be for the same named item. For example, if you are looking for items with the status InProgress or the status Pending.

Type: Array of Filter (p. 300) objects

Array Members: Fixed number of 1 item.

Required: No

maxResults (p. 232)

When this parameter is used, ListDeploymentJobs only returns maxResults results in a single page along with a nextToken response element. The remaining results of the initial request can be seen by sending another ListDeploymentJobs request with the returned nextToken value. This value can be between 1 and 200. If this parameter is not used, then ListDeploymentJobs returns up to 200 results and a nextToken value if applicable.

Type: Integer

Required: No

nextToken (p. 232)

The nextToken value returned from a previous paginated ListDeploymentJobs request where maxResults was used and the results exceeded the value of that parameter. Pagination continues from the end of the previous results that returned the nextToken value.
Type: String

Length Constraints: Minimum length of 1. Maximum length of 2048.

Pattern: [a-zA-Z0-9_.\-\+/=]*

Required: No

Response Syntax

HTTP/1.1 200
Content-type: application/json

{
  "deploymentJobs": [
    {
      "arn": "string",
      "createdAt": number,
      "deploymentApplicationConfigs": [
        {
          "application": "string",
          "applicationVersion": "string",
          "launchConfig": {
            "environmentVariables": {
              "string": "string"
            },
            "launchFile": "string",
            "packageName": "string",
            "postLaunchFile": "string",
            "preLaunchFile": "string"
          }
        }
      ],
      "deploymentConfig": {
        "concurrentDeploymentPercentage": number,
        "downloadConditionFile": {
          "bucket": "string",
          "etag": "string",
          "key": "string"
        },
        "failureThresholdPercentage": number,
        "robotDeploymentTimeoutInSeconds": number
      },
      "failureCode": "string",
      "failureReason": "string",
      "fleet": "string",
      "status": "string"
    }
  ],
  "nextToken": "string"
}

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

**deploymentJobs (p. 233)**

A list of deployment jobs that meet the criteria of the request.
Type: Array of DeploymentJob (p. 294) objects

Array Members: Minimum number of 0 items. Maximum number of 200 items.

nextToken (p. 233)

The nextToken value to include in a future ListDeploymentJobs request. When the results of a ListDeploymentJobs request exceed maxResults, this value can be used to retrieve the next page of results. This value is null when there are no more results to return.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2048.

Pattern: [a-zA-Z0-9_.-/+=]*

Errors

For information about the errors that are common to all actions, see Common Errors (p. 342).

InternalServerErrorException

AWS RoboMaker experienced a service issue. Try your call again.

HTTP Status Code: 500

InvalidParameterException

A parameter specified in a request is not valid, is unsupported, or cannot be used. The returned message provides an explanation of the error value.

HTTP Status Code: 400

ResourceNotFoundException

The specified resource does not exist.

HTTP Status Code: 400

ThrottlingException

AWS RoboMaker is temporarily unable to process the request. Try your call again.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3
ListFleets

Returns a list of fleets. You can optionally provide filters to retrieve specific fleets.

Request Syntax

POST /listFleets HTTP/1.1
Content-type: application/json

{
   "filters": [
      {
         "name": "string",
         "values": [ "string" ]
      }
   ],
   "maxResults": number,
   "nextToken": "string"
}

URI Request Parameters

The request does not use any URI parameters.

Request Body

The request accepts the following data in JSON format.

filters (p. 236)

Optional filters to limit results.

The filter name name is supported. When filtering, you must use the complete value of the filtered item. You can use up to three filters.

Type: Array of Filter (p. 300) objects

Array Members: Fixed number of 1 item.

Required: No

maxResults (p. 236)

When this parameter is used, ListFleets only returns maxResults results in a single page along with a nextToken response element. The remaining results of the initial request can be seen by sending another ListFleets request with the returned nextToken value. This value can be between 1 and 200. If this parameter is not used, then ListFleets returns up to 200 results and a nextToken value if applicable.

Type: Integer

Required: No

nextToken (p. 236)

The nextToken value returned from a previous paginated ListFleets request where maxResults was used and the results exceeded the value of that parameter. Pagination continues from the end of the previous results that returned the nextToken value.
Note
This token should be treated as an opaque identifier that is only used to retrieve the next items in a list and not for other programmatic purposes.

Type: String
Length Constraints: Minimum length of 1. Maximum length of 2048.
Pattern: [a-zA-Z0-9_.\-/+=]*
Required: No

Response Syntax

HTTP/1.1 200
Content-type: application/json
{
  "fleetDetails": [
    {
      "arn": "string",
      "createdAt": number,
      "lastDeploymentJob": "string",
      "lastDeploymentStatus": "string",
      "lastDeploymentTime": number,
      "name": "string"
    }
  ],
  "nextToken": "string"
}

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

fleetDetails (p. 237)
A list of fleet details meeting the request criteria.
Type: Array of Fleet (p. 301) objects
Array Members: Minimum number of 0 items. Maximum number of 200 items.

nextToken (p. 237)
The nextToken value to include in a future ListDeploymentJobs request. When the results of a ListFleets request exceed maxResults, this value can be used to retrieve the next page of results. This value is null when there are no more results to return.
Type: String
Length Constraints: Minimum length of 1. Maximum length of 2048.
Pattern: [a-zA-Z0-9_.\-/+=]*

Errors
For information about the errors that are common to all actions, see Common Errors (p. 342).
InternalServerException

AWS RoboMaker experienced a service issue. Try your call again.

HTTP Status Code: 500

InvalidParameterException

A parameter specified in a request is not valid, is unsupported, or cannot be used. The returned message provides an explanation of the error value.

HTTP Status Code: 400

ResourceNotFoundException

The specified resource does not exist.

HTTP Status Code: 400

ThrottlingException

AWS RoboMaker is temporarily unable to process the request. Try your call again.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3
ListRobotApplications

Returns a list of robot application. You can optionally provide filters to retrieve specific robot applications.

Request Syntax

```json
POST /listRobotApplications HTTP/1.1
Content-type: application/json

{
   "filters": [
      {
         "name": "string",
         "values": [ "string" ]
      }
   ],
   "maxResults": number,
   "nextToken": "string",
   "versionQualifier": "string"
}
```

URI Request Parameters

The request does not use any URI parameters.

Request Body

The request accepts the following data in JSON format.

`filters (p. 239)`

Optional filters to limit results.

The filter name `name` is supported. When filtering, you must use the complete value of the filtered item. You can use up to three filters.

Type: Array of `Filter (p. 300)` objects

Array Members: Fixed number of 1 item.

Required: No

`maxResults (p. 239)`

When this parameter is used, `ListRobotApplications` only returns `maxResults` results in a single page along with a `nextToken` response element. The remaining results of the initial request can be seen by sending another `ListRobotApplications` request with the returned `nextToken` value. This value can be between 1 and 100. If this parameter is not used, then `ListRobotApplications` returns up to 100 results and a `nextToken` value if applicable.

Type: Integer

Required: No

`nextToken (p. 239)`

The `nextToken` value returned from a previous paginated `ListRobotApplications` request where `maxResults` was used and the results exceeded the value of that parameter. Pagination continues from the end of the previous results that returned the `nextToken` value.
Type: String
Length Constraints: Minimum length of 1. Maximum length of 2048.
Pattern: [a-zA-Z0-9_.\-\+/=]*

Required: No

versionQualifier (p. 239)
The version qualifier of the robot application.
Type: String
Length Constraints: Minimum length of 1. Maximum length of 255.
Pattern: ALL
Required: No

Response Syntax

```
HTTP/1.1 200
Content-type: application/json

{
   "nextToken": "string",
   "robotApplicationSummaries": [
      {
         "arn": "string",
         "lastUpdatedAt": number,
         "name": "string",
         "robotSoftwareSuite": {
            "name": "string",
            "version": "string"
         },
         "version": "string"
      }
   ]
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

nextToken (p. 240)
The `nextToken` value to include in a future ListRobotApplications request. When the results of a ListRobotApplications request exceed maxResults, this value can be used to retrieve the next page of results. This value is null when there are no more results to return.

Type: String
Length Constraints: Minimum length of 1. Maximum length of 2048.
Pattern: [a-zA-Z0-9_.\-\+/=]*

robotApplicationSummaries (p. 240)
A list of robot application summaries that meet the criteria of the request.
Type: Array of $\text{RobotApplicationSummary (p. 316)}$ objects

Array Members: Minimum number of 0 items. Maximum number of 100 items.

Errors

For information about the errors that are common to all actions, see Common Errors (p. 342).

**InternalServerException**

AWS RoboMaker experienced a service issue. Try your call again.

HTTP Status Code: 500

**InvalidParameterException**

A parameter specified in a request is not valid, is unsupported, or cannot be used. The returned message provides an explanation of the error value.

HTTP Status Code: 400

**ThrottlingException**

AWS RoboMaker is temporarily unable to process the request. Try your call again.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3
ListRobots

Returns a list of robots. You can optionally provide filters to retrieve specific robots.

Request Syntax

```
POST /listRobots HTTP/1.1
Content-type: application/json

{
  "filters": [
    {
      "name": "string",
      "values": [ "string" ]
    }
  ],
  "maxResults": number,
  "nextToken": "string"
}
```

URI Request Parameters

The request does not use any URI parameters.

Request Body

The request accepts the following data in JSON format.

**filters (p. 242)**

Optional filters to limit results.

The filter names status and fleetName are supported. When filtering, you must use the complete value of the filtered item. You can use up to three filters, but they must be for the same named item. For example, if you are looking for items with the status Registered or the status Available.

Type: Array of Filter (p. 300) objects

Array Members: Fixed number of 1 item.

Required: No

**maxResults (p. 242)**

When this parameter is used, ListRobots only returns maxResults results in a single page along with a nextToken response element. The remaining results of the initial request can be seen by sending another ListRobots request with the returned nextToken value. This value can be between 1 and 200. If this parameter is not used, then ListRobots returns up to 200 results and a nextToken value if applicable.

Type: Integer

Required: No

**nextToken (p. 242)**

The nextToken value returned from a previous paginated ListRobots request where maxResults was used and the results exceeded the value of that parameter. Pagination continues from the end of the previous results that returned the nextToken value.
Type: String
Length Constraints: Minimum length of 1. Maximum length of 2048.
Pattern: [a-zA-Z0-9_.-\/=]*
Required: No

Response Syntax

HTTP/1.1 200
Content-type: application/json
{
  "nextToken": "string",
  "robots": [
    {
      "architecture": "string",
      "arn": "string",
      "createdAt": number,
      "fleetArn": "string",
      "greenGrassGroupId": "string",
      "lastDeploymentJob": "string",
      "lastDeploymentTime": number,
      "name": "string",
      "status": "string"
    }
  ]
}

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

**nextToken (p. 243)**

The `nextToken` value to include in a future `ListRobots` request. When the results of a `ListRobots` request exceed `maxResults`, this value can be used to retrieve the next page of results. This value is null when there are no more results to return.

Type: String
Length Constraints: Minimum length of 1. Maximum length of 2048.
Pattern: [a-zA-Z0-9_.-\/=]*

**robots (p. 243)**

A list of robots that meet the criteria of the request.

Type: Array of `Robot (p. 313)` objects
Array Members: Minimum number of 0 items. Maximum number of 1000 items.

Errors

For information about the errors that are common to all actions, see `Common Errors (p. 342)`.
**InternalServerException**

AWS RoboMaker experienced a service issue. Try your call again.

HTTP Status Code: 500

**InvalidParameterException**

A parameter specified in a request is not valid, is unsupported, or cannot be used. The returned message provides an explanation of the error value.

HTTP Status Code: 400

**ResourceNotFoundException**

The specified resource does not exist.

HTTP Status Code: 400

**ThrottlingException**

AWS RoboMaker is temporarily unable to process the request. Try your call again.

HTTP Status Code: 400

**See Also**

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3
ListSimulationApplications

Returns a list of simulation applications. You can optionally provide filters to retrieve specific simulation applications.

Request Syntax

POST /listSimulationApplications HTTP/1.1
Content-type: application/json

{
  "filters": [
    {
      "name": "string",
      "values": [ "string" ]
    },
    "maxResults": number,
    "nextToken": "string",
    "versionQualifier": "string"
  ]
}

URI Request Parameters

The request does not use any URI parameters.

Request Body

The request accepts the following data in JSON format.

filters (p. 245)

Optional list of filters to limit results.

The filter name name is supported. When filtering, you must use the complete value of the filtered item. You can use up to three filters.

Type: Array of Filter (p. 300) objects

Array Members: Fixed number of 1 item.

Required: No

maxResults (p. 245)

When this parameter is used, ListSimulationApplications only returns maxResults results in a single page along with a nextToken response element. The remaining results of the initial request can be seen by sending another ListSimulationApplications request with the returned nextToken value. This value can be between 1 and 100. If this parameter is not used, then ListSimulationApplications returns up to 100 results and a nextToken value if applicable.

Type: Integer

Required: No

nextToken (p. 245)

The nextToken value returned from a previous paginated ListSimulationApplications request where maxResults was used and the results exceeded the value of that parameter. Pagination continues from the end of the previous results that returned the nextToken value.
Response Syntax

HTTP/1.1 200
Content-type: application/json

{  
  "nextToken": "string",
  "simulationApplicationSummaries": [  
    {  
      "arn": "string",
      "lastUpdatedAt": number,
      "name": "string",
      "robotSoftwareSuite": {  
        "name": "string",
        "version": "string"
      },
      "simulationSoftwareSuite": {  
        "name": "string",
        "version": "string"
      },
      "version": "string"
    }
  ]
}

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

nextToken (p. 246)

The nextToken value to include in a future ListSimulationApplications request. When the results of a ListRobot request exceed maxResults, this value can be used to retrieve the next page of results. This value is null when there are no more results to return.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2048.
Pattern: [a-zA-Z0-9_.\-_\+\=]*

**simulationApplicationSummaries (p. 246)**

A list of simulation application summaries that meet the criteria of the request.

Type: Array of SimulationApplicationSummary (p. 324) objects

Array Members: Minimum number of 0 items. Maximum number of 100 items.

**Errors**

For information about the errors that are common to all actions, see Common Errors (p. 342).

**InternalServerException**

AWS RoboMaker experienced a service issue. Try your call again.

HTTP Status Code: 500

**InvalidParameterException**

A parameter specified in a request is not valid, is unsupported, or cannot be used. The returned message provides an explanation of the error value.

HTTP Status Code: 400

**ThrottlingException**

AWS RoboMaker is temporarily unable to process the request. Try your call again.

HTTP Status Code: 400

**See Also**

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3
ListSimulationJobBatches

Returns a list simulation job batches. You can optionally provide filters to retrieve specific simulation batch jobs.

Request Syntax

```json
POST /listSimulationJobBatches HTTP/1.1
Content-type: application/json

{
    "filters": [
        {
            "name": "string",
            "values": [ "string" ]
        }
    ],
    "maxResults": number,
    "nextToken": "string"
}
```

URI Request Parameters

The request does not use any URI parameters.

Request Body

The request accepts the following data in JSON format.

**filters (p. 248)**

Optional filters to limit results.

Type: Array of Filter (p. 300) objects

Array Members: Fixed number of 1 item.

Required: No

**maxResults (p. 248)**

When this parameter is used, ListSimulationJobBatches only returns maxResults results in a single page along with a nextToken response element. The remaining results of the initial request can be seen by sending another ListSimulationJobBatches request with the returned nextToken value.

Type: Integer

Required: No

**nextToken (p. 248)**

The nextToken value returned from a previous paginated ListSimulationJobBatches request where maxResults was used and the results exceeded the value of that parameter. Pagination continues from the end of the previous results that returned the nextToken value.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2048.

Pattern: [a-zA-Z0-9_.\-\+/=]*
ListSimulationJobBatches

Required: No

Response Syntax

HTTP/1.1 200
Content-type: application/json
{
    "nextToken": "string",
    "simulationJobBatchSummaries": [
    {
        "arn": "string",
        "createdAt": number,
        "createdRequestCount": number,
        "failedRequestCount": number,
        "lastUpdatedAt": number,
        "pendingRequestCount": number,
        "status": "string"
    }
    ]
}

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

nextToken (p. 249)

The nextToken value to include in a future ListSimulationJobBatches request. When the results of a ListSimulationJobBatches request exceed maxResults, this value can be used to retrieve the next page of results. This value is null when there are no more results to return.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2048.

Pattern: [a-zA-Z0-9_.-\+/=]*

simulationJobBatchSummaries (p. 249)

A list of simulation job batch summaries.

Type: Array of SimulationJobBatchSummary (p. 330) objects

Errors

For information about the errors that are common to all actions, see Common Errors (p. 342).

InternalServerException

AWS RoboMaker experienced a service issue. Try your call again.

HTTP Status Code: 500

InvalidParameterException

A parameter specified in a request is not valid, is unsupported, or cannot be used. The returned message provides an explanation of the error value.
HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3
ListSimulationJobs

Returns a list of simulation jobs. You can optionally provide filters to retrieve specific simulation jobs.

Request Syntax

```
POST /listSimulationJobs HTTP/1.1
Content-type: application/json

{
  "filters": [
    {
      "name": "string",
      "values": [ "string" ]
    },
    "maxResults": number,
    "nextToken": "string"
  }
}
```

URI Request Parameters

The request does not use any URI parameters.

Request Body

The request accepts the following data in JSON format.

filters (p. 251)

Optional filters to limit results.

The filter names status and simulationApplicationName and robotApplicationName are supported. When filtering, you must use the complete value of the filtered item. You can use up to three filters, but they must be for the same named item. For example, if you are looking for items with the status Preparing or the status Running.

Type: Array of Filter (p. 300) objects

Array Members: Fixed number of 1 item.

Required: No

maxResults (p. 251)

When this parameter is used, ListSimulationJobs only returns maxResults results in a single page along with a nextToken response element. The remaining results of the initial request can be seen by sending another ListSimulationJobs request with the returned nextToken value. This value can be between 1 and 1000. If this parameter is not used, then ListSimulationJobs returns up to 1000 results and a nextToken value if applicable.

Type: Integer

Required: No

nextToken (p. 251)

The nextToken value returned from a previous paginated ListSimulationJobs request where maxResults was used and the results exceeded the value of that parameter. Pagination continues from the end of the previous results that returned the nextToken value.
**Note**
This token should be treated as an opaque identifier that is only used to retrieve the next items in a list and not for other programmatic purposes.

Type: String
Length Constraints: Minimum length of 1. Maximum length of 2048.
Pattern: [a-zA-Z0-9_.\-/+=]*
Required: No

**Response Syntax**

HTTP/1.1 200
Content-type: application/json

```json
{
    "nextToken": "string",
    "simulationJobSummaries": [
        {
            "arn": "string",
            "dataSourceNames": [ "string" ],
            "lastUpdatedAt": number,
            "name": "string",
            "robotApplicationNames": [ "string" ],
            "simulationApplicationNames": [ "string" ],
            "status": "string"
        }
    ]
}
```

**Response Elements**

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

**nextToken (p. 252)**

The `nextToken` value to include in a future `ListSimulationJobs` request. When the results of a `ListRobot` request exceed `maxResults`, this value can be used to retrieve the next page of results. This value is `null` when there are no more results to return.

Type: String
Length Constraints: Minimum length of 1. Maximum length of 2048.
Pattern: [a-zA-Z0-9_.\-/+=]*

**simulationJobSummaries (p. 252)**

A list of simulation job summaries that meet the criteria of the request.

Type: Array of `SimulationJobSummary (p. 335)` objects
Array Members: Minimum number of 0 items. Maximum number of 100 items.

**Errors**

For information about the errors that are common to all actions, see `Common Errors (p. 342)`. 
InternalServerException

AWS RoboMaker experienced a service issue. Try your call again.

HTTP Status Code: 500

InvalidParameterException

A parameter specified in a request is not valid, is unsupported, or cannot be used. The returned message provides an explanation of the error value.

HTTP Status Code: 400

ThrottlingException

AWS RoboMaker is temporarily unable to process the request. Try your call again.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3
ListTagsForResource

Lists all tags on a AWS RoboMaker resource.

Request Syntax

GET /tags/resourceArn HTTP/1.1

URI Request Parameters

The request requires the following URI parameters.

resourceArn (p. 254)

The AWS RoboMaker Amazon Resource Name (ARN) with tags to be listed.


Pattern: arn:.*

Request Body

The request does not have a request body.

Response Syntax

HTTP/1.1 200
Content-type: application/json
{
    "tags": {
        "String" : "string"
    }
}

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

tags (p. 254)

The list of all tags added to the specified resource.

Type: String to string map

Key Length Constraints: Minimum length of 1. Maximum length of 128.

Key Pattern: [a-zA-Z0-9 _.-\+/=]*

Value Length Constraints: Minimum length of 0. Maximum length of 256.

Value Pattern: [a-zA-Z0-9 _.-\+/=]*
Errors

For information about the errors that are common to all actions, see Common Errors (p. 342).

InternalServerException

AWS RoboMaker experienced a service issue. Try your call again.

HTTP Status Code: 500

InvalidParameterException

A parameter specified in a request is not valid, is unsupported, or cannot be used. The returned message provides an explanation of the error value.

HTTP Status Code: 400

ResourceNotFoundException

The specified resource does not exist.

HTTP Status Code: 400

ThrottlingException

AWS RoboMaker is temporarily unable to process the request. Try your call again.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3
RegisterRobot

Registers a robot with a fleet.

Request Syntax

POST /registerRobot HTTP/1.1
Content-type: application/json

{
  "fleet": "string",
  "robot": "string"
}

URI Request Parameters

The request does not use any URI parameters.

Request Body

The request accepts the following data in JSON format.

fleet (p. 256)

The Amazon Resource Name (ARN) of the fleet.

Type: String


Pattern: arn:.*

Required: Yes

robot (p. 256)

The Amazon Resource Name (ARN) of the robot.

Type: String


Pattern: arn:.*

Required: Yes

Response Syntax

HTTP/1.1 200
Content-type: application/json

{
  "fleet": "string",
  "robot": "string"
}
Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

fleet (p. 256)

The Amazon Resource Name (ARN) of the fleet that the robot will join.

Type: String


Pattern: arn:.*

robot (p. 256)

Information about the robot registration.

Type: String


Pattern: arn:.*

Errors

For information about the errors that are common to all actions, see Common Errors (p. 342).

InternalServerErrorException

AWS RoboMaker experienced a service issue. Try your call again.

HTTP Status Code: 500

InvalidParameterException

A parameter specified in a request is not valid, is unsupported, or cannot be used. The returned message provides an explanation of the error value.

HTTP Status Code: 400

LimitExceededException

The requested resource exceeds the maximum number allowed, or the number of concurrent stream requests exceeds the maximum number allowed.

HTTP Status Code: 400

ResourceNotFoundException

The specified resource does not exist.

HTTP Status Code: 400

ThrottlingException

AWS RoboMaker is temporarily unable to process the request. Try your call again.

HTTP Status Code: 400
See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3
RestartSimulationJob

Restarts a running simulation job.

Request Syntax

POST /restartSimulationJob HTTP/1.1
Content-type: application/json

{
  "job": "string"
}

URI Request Parameters

The request does not use any URI parameters.

Request Body

The request accepts the following data in JSON format.

job (p. 259)

The Amazon Resource Name (ARN) of the simulation job.

Type: String


Pattern: arn:.*

Required: Yes

Response Syntax

HTTP/1.1 200

Response Elements

If the action is successful, the service sends back an HTTP 200 response with an empty HTTP body.

Errors

For information about the errors that are common to all actions, see Common Errors (p. 342).

InternalServerException

AWS RoboMaker experienced a service issue. Try your call again.

HTTP Status Code: 500

InvalidParameterException

A parameter specified in a request is not valid, is unsupported, or cannot be used. The returned message provides an explanation of the error value.
HTTP Status Code: 400

**LimitExceeded Exception**

The requested resource exceeds the maximum number allowed, or the number of concurrent stream requests exceeds the maximum number allowed.

HTTP Status Code: 400

**ResourceNotFoundException**

The specified resource does not exist.

HTTP Status Code: 400

**ThrottlingException**

AWS RoboMaker is temporarily unable to process the request. Try your call again.

HTTP Status Code: 400

**See Also**

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3
StartSimulationJobBatch

Starts a new simulation job batch. The batch is defined using one or more SimulationJobRequest objects.

Request Syntax

```json
POST /startSimulationJobBatch HTTP/1.1
Content-type: application/json

{
  "batchPolicy": {
    "maxConcurrency": number,
    "timeoutInSeconds": number
  },
  "clientRequestToken": "string",
  "createSimulationJobRequests": [
    {
      "dataSources": [
        {
          "name": "string",
          "s3Bucket": "string",
          "s3Keys": [ "string" ]
        }
      ],
      "failureBehavior": "string",
      "iamRole": "string",
      "loggingConfig": {
        "recordAllRosTopics": boolean
      },
      "maxJobDurationInSeconds": number,
      "outputLocation": {
        "s3Bucket": "string",
        "s3Prefix": "string"
      },
      "robotApplications": [
        {
          "application": "string",
          "applicationVersion": "string",
          "launchConfig": {
            "environmentVariables": {
              "string": "string"
            },
            "launchFile": "string",
            "packageName": "string",
            "portForwardingConfig": {
              "portMappings": [
                {
                  "applicationPort": number,
                  "enableOnPublicIp": boolean,
                  "jobPort": number
                }
              ],
              "streamUI": boolean
            }
          }
        }
      ],
      "simulationApplications": [
        {
          "application": "string",
          "applicationVersion": "string",
          "launchConfig": {
```
"environmentVariables": {
"string": "string"
},
"launchFile": "string",
"packageName": "string",
"portForwardingConfig": {
"portMappings": [
{
"applicationPort": number,
"enableOnPublicIp": boolean,
"jobPort": number
}
],
"streamUI": boolean
},
"tags": {
"string": "string"
},
"useDefaultApplications": boolean,
"vpcConfig": {
"assignPublicIp": boolean,
"securityGroups": [ "string" ],
"subnets": [ "string" ]
}
},
"tags": {
"string": "string"
}

**URI Request Parameters**

The request does not use any URI parameters.

**Request Body**

The request accepts the following data in JSON format.

**batchPolicy (p. 261)**

The batch policy.

Type: BatchPolicy (p. 289) object

Required: No

**clientRequestToken (p. 261)**

Unique, case-sensitive identifier that you provide to ensure the idempotency of the request.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 64.

Pattern: [ a-zA-Z0-9_\-\=]*

Required: No

**createSimulationJobRequests (p. 261)**

A list of simulation job requests to create in the batch.
Type: Array of `SimulationJobRequest (p. 332)` objects

Array Members: Minimum number of 1 item.

Required: Yes

tags (p. 261)

A map that contains tag keys and tag values that are attached to the deployment job batch.

Type: String to string map

Key Length Constraints: Minimum length of 1. Maximum length of 128.

Key Pattern: `[a-zA-Z0-9 _\-\/]\+\*`*

Value Length Constraints: Minimum length of 0. Maximum length of 256.

Value Pattern: `[a-zA-Z0-9 _\-\/]\+\*`*

Required: No

Response Syntax

```json
HTTP/1.1 200
Content-type: application/json

{
  "arn": "string",
  "batchPolicy": {
    "maxConcurrency": number,
    "timeoutInSeconds": number
  },
  "clientRequestToken": "string",
  "createdAt": number,
  "createdRequests": [
    {
      "arn": "string",
      "dataSourceNames": [ "string" ],
      "lastUpdatedAt": number,
      "name": "string",
      "robotApplicationNames": [ "string" ],
      "simulationApplicationNames": [ "string" ],
      "status": "string"
    }
  ],
  "failedRequests": [
    {
      "failedAt": number,
      "failureCode": "string",
      "failureReason": "string",
      "request": {,
        "dataSources": [ {
          "name": "string",
          "s3Bucket": "string",
          "s3Keys": [ "string" ]
        }
      ],
      "failureBehavior": "string",
      "iamRole": "string",
      "loggingConfig": {,
        "recordAllRosTopics": boolean
      }
    }
  ]
}```
"maxJobDurationInSeconds": number,
"outputLocation": {
  "s3Bucket": "string",
  "s3Prefix": "string"
},
"robotApplications": [
  {
    "application": "string",
    "applicationVersion": "string",
    "launchConfig": {
      "environmentVariables": {
        "string": "string"
      },
      "launchFile": "string",
      "packageName": "string",
      "portForwardingConfig": {
        "portMappings": [
          {
            "applicationPort": number,
            "enableOnPublicIp": boolean,
            "jobPort": number
          }
        ],
        "streamUI": boolean
      }
    }
  }
],
"simulationApplications": [
  {
    "application": "string",
    "applicationVersion": "string",
    "launchConfig": {
      "environmentVariables": {
        "string": "string"
      },
      "launchFile": "string",
      "packageName": "string",
      "portForwardingConfig": {
        "portMappings": [
          {
            "applicationPort": number,
            "enableOnPublicIp": boolean,
            "jobPort": number
          }
        ],
        "streamUI": boolean
      }
    }
  }
],
"tags": {
  "string": "string"
},
"useDefaultApplications": boolean,
"vpcConfig": {
  "assignPublicIp": boolean,
  "securityGroups": [ "string" ],
  "subnets": [ "string" ]
}
"failureCode": "string",
"failureReason": "string"
"pendingRequests": [
  {
    "dataSources": [
      {
        "name": "string",
        "s3Bucket": "string",
        "s3Keys": [ "string" ]
      }
    ],
    "failureBehavior": "string",
    "iamRole": "string",
    "loggingConfig": {
      "recordAllRosTopics": boolean
    },
    "maxJobDurationInSeconds": number,
    "outputLocation": {
      "s3Bucket": "string",
      "s3Prefix": "string"
    },
    "robotApplications": [
      {
        "application": "string",
        "applicationVersion": "string",
        "launchConfig": {
          "environmentVariables": {
            "string": "string"
          },
          "launchFile": "string",
          "packageName": "string",
          "portForwardingConfig": {
            "portMappings": [
              {
                "applicationPort": number,
                "enableOnPublicIp": boolean,
                "jobPort": number
              }
            ]
          },
          "streamUI": boolean
        }
      }
    ],
    "simulationApplications": [
      {
        "application": "string",
        "applicationVersion": "string",
        "launchConfig": {
          "environmentVariables": {
            "string": "string"
          },
          "launchFile": "string",
          "packageName": "string",
          "portForwardingConfig": {
            "portMappings": [
              {
                "applicationPort": number,
                "enableOnPublicIp": boolean,
                "jobPort": number
              }
            ]
          },
          "streamUI": boolean
        }
      }
    ],
    "tags": {
      "tagKey": "string",
      "tagValue": "string"
    }
  }
]
"string" : "string"
},
"useDefaultApplications": boolean,
"vpcConfig": {
  "assignPublicIp": boolean,
  "securityGroups": [ "string" ],
  "subnets": [ "string" ]
}
]
,
"status": "string",
"tags": {
  "string" : "string"
}
}

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

**arn (p. 263)**

The Amazon Resource Name (arn) of the batch.

Type: String


Pattern: arn:.*

**batchPolicy (p. 263)**

The batch policy.

Type: BatchPolicy (p. 289) object

**clientRequestToken (p. 263)**

Unique, case-sensitive identifier that you provide to ensure the idempotency of the request.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 64.

Pattern: [a-zA-Z0-9-_]*

**createdAt (p. 263)**

The time, in milliseconds since the epoch, when the simulation job batch was created.

Type: Timestamp

**createdRequests (p. 263)**

A list of created simulation job request summaries.

Type: Array of SimulationJobSummary (p. 335) objects

Array Members: Minimum number of 0 items. Maximum number of 100 items.

**failedRequests (p. 263)**

A list of failed simulation job requests. The request failed to be created into a simulation job. Failed requests do not have a simulation job ID.
Type: Array of `FailedCreateSimulationJobRequest (p. 298)` objects

`failureCode (p. 263)`

The failure code if the simulation job batch failed.

Type: String

Valid Values: `InternalServerError`

`failureReason (p. 263)`

The reason the simulation job batch failed.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 1024.

Pattern: .*

`pendingRequests (p. 263)`

A list of pending simulation job requests. These requests have not yet been created into simulation jobs.

Type: Array of `SimulationJobRequest (p. 332)` objects

Array Members: Minimum number of 1 item.

`status (p. 263)`

The status of the simulation job batch.

Pending

The simulation job batch request is pending.

InProgress

The simulation job batch is in progress.

Failed

The simulation job batch failed. One or more simulation job requests could not be completed due to an internal failure (like `InternalServerError`). See `failureCode` and `failureReason` for more information.

Completed

The simulation batch job completed. A batch is complete when (1) there are no pending simulation job requests in the batch and none of the failed simulation job requests are due to `InternalServerError` and (2) when all created simulation jobs have reached a terminal state (for example, `Completed` or `Failed`).

Canceled

The simulation batch job was cancelled.

Canceling

The simulation batch job is being cancelled.

Completing

The simulation batch job is completing.

TimingOut

The simulation job batch is timing out.
If a batch timing out, and there are pending requests that were failing due to an internal failure (like `InternalServiceError`), the batch status will be `Failed`. If there are no such failing request, the batch status will be `TimedOut`.

`TimedOut`

The simulation batch job timed out.

Type: String

Valid Values: Pending | InProgress | Failed | Completed | Canceled | Canceling | Completing | TimingOut | TimedOut

`tags` (p. 263)

A map that contains tag keys and tag values that are attached to the deployment job batch.

Type: String to string map

Key Length Constraints: Minimum length of 1. Maximum length of 128.

Key Pattern: `[a-zA-Z0-9 _.-\+/=:]*`

Value Length Constraints: Minimum length of 0. Maximum length of 256.

Value Pattern: `[a-zA-Z0-9 _.-\+/=:]*`

**Errors**

For information about the errors that are common to all actions, see `Common Errors` (p. 342).

`IdempotentParameterMismatchException`

The request uses the same client token as a previous, but non-identical request. Do not reuse a client token with different requests, unless the requests are identical.

HTTP Status Code: 400

`InternalServerException`

AWS RoboMaker experienced a service issue. Try your call again.

HTTP Status Code: 500

`InvalidParameterException`

A parameter specified in a request is not valid, is unsupported, or cannot be used. The returned message provides an explanation of the error value.

HTTP Status Code: 400

`LimitExceededException`

The requested resource exceeds the maximum number allowed, or the number of concurrent stream requests exceeds the maximum number allowed.

HTTP Status Code: 400

`ThrottlingException`

AWS RoboMaker is temporarily unable to process the request. Try your call again.

HTTP Status Code: 400
See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3
SyncDeploymentJob

Syncrhonizes robots in a fleet to the latest deployment. This is helpful if robots were added after a deployment.

Request Syntax

```
POST /syncDeploymentJob HTTP/1.1
Content-type: application/json

{
  "clientRequestToken": "string",
  "fleet": "string"
}
```

URI Request Parameters

The request does not use any URI parameters.

Request Body

The request accepts the following data in JSON format.

clientRequestToken (p. 270)

- Unique, case-sensitive identifier that you provide to ensure the idempotency of the request.
- Type: String
- Length Constraints: Minimum length of 1. Maximum length of 64.
- Pattern: `[a-zA-Z0-9-_\-=]*`
- Required: Yes

fleet (p. 270)

- The target fleet for the synchronization.
- Type: String
- Pattern: `arn:` + `.*`
- Required: Yes

Response Syntax

```
HTTP/1.1 200
Content-type: application/json

{
  "arn": "string",
  "createdAt": number,
  "deploymentApplicationConfigs": [
  ]
}
```
"application": "string",
"applicationVersion": "string",
"launchConfig": {
    "environmentVariables": {
        "string": "string"
    },
    "launchFile": "string",
    "packageName": "string",
    "postLaunchFile": "string",
    "preLaunchFile": "string"
}
],
"deploymentConfig": {
    "concurrentDeploymentPercentage": number,
    "downloadConditionFile": {
        "bucket": "string",
        "etag": "string",
        "key": "string"
    },
    "failureThresholdPercentage": number,
    "robotDeploymentTimeoutInSeconds": number
},
"failureCode": "string",
"failureReason": "string",
"fleet": "string",
"status": "string"

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

**arn (p. 270)**

The Amazon Resource Name (ARN) of the synchronization request.

Type: String


Pattern: arn:.*

**createdAt (p. 270)**

The time, in milliseconds since the epoch, when the fleet was created.

Type: Timestamp

**deploymentApplicationConfigs (p. 270)**

Information about the deployment application configurations.

Type: Array of DeploymentApplicationConfig (p. 292) objects

Array Members: Fixed number of 1 item.

**deploymentConfig (p. 270)**

Information about the deployment configuration.

Type: DeploymentConfig (p. 293) object
failureCode (p. 270)

The failure code if the job fails:

- **InternalServiceError**
  - Internal service error.
- **RobotApplicationCrash**
  - Robot application exited abnormally.
- **SimulationApplicationCrash**
  - Simulation application exited abnormally.
- **BadPermissionsRobotApplication**
  - Robot application bundle could not be downloaded.
- **BadPermissionsSimulationApplication**
  - Simulation application bundle could not be downloaded.
- **BadPermissionsS3Output**
  - Unable to publish outputs to customer-provided S3 bucket.
- **BadPermissionsCloudwatchLogs**
  - Unable to publish logs to customer-provided CloudWatch Logs resource.
- **SubnetIpLimitExceeded**
  - Subnet IP limit exceeded.
- **ENILimitExceeded**
  - ENI limit exceeded.
- **BadPermissionsUserCredentials**
  - Unable to use the Role provided.
- **InvalidBundleRobotApplication**
  - Robot bundle cannot be extracted (invalid format, bundling error, or other issue).
- **InvalidBundleSimulationApplication**
  - Simulation bundle cannot be extracted (invalid format, bundling error, or other issue).
- **RobotApplicationVersionMismatchedEtag**
  - Etag for RobotApplication does not match value during version creation.
- **SimulationApplicationVersionMismatchedEtag**
  - Etag for SimulationApplication does not match value during version creation.

Type: String

Valid Values:
- ResourceNotFound
- EnvironmentSetupError
- EtagMismatch
- FailureThresholdBreached
- RobotDeploymentAborted
- RobotDeploymentNoResponse
- RobotAgentConnectionTimeout
- GreengrassDeploymentFailed
- InvalidGreengrassGroup
- MissingRobotArchitecture
- MissingRobotApplicationArchitecture
- MissingRobotDeploymentResource
- GreengrassGroupVersionDoesNotExist
- LambdaDeleted
- ExtractingBundleFailure
- PreLaunchFileFailure
- PostLaunchFileFailure
- BadPermissionError
- DownloadConditionFailed
- InternalServerError
failureReason (p. 270)

The failure reason if the job fails.

Type: String
Length Constraints: Minimum length of 0. Maximum length of 1024.
Pattern: .*

fleet (p. 270)

The Amazon Resource Name (ARN) of the fleet.

Type: String
Pattern: arn:.*

status (p. 270)

The status of the synchronization job.

Type: String
Valid Values: Pending | Preparing | InProgress | Failed | Succeeded | Canceled

Errors

For information about the errors that are common to all actions, see Common Errors (p. 342).

ConcurrentDeploymentException

The failure percentage threshold percentage was met.

HTTP Status Code: 400

IdempotentParameterMismatchException

The request uses the same client token as a previous, but non-identical request. Do not reuse a client token with different requests, unless the requests are identical.

HTTP Status Code: 400

InternalServerException

AWS RoboMaker experienced a service issue. Try your call again.

HTTP Status Code: 500

InvalidParameterException

A parameter specified in a request is not valid, is unsupported, or cannot be used. The returned message provides an explanation of the error value.

HTTP Status Code: 400

LimitExceededError

The requested resource exceeds the maximum number allowed, or the number of concurrent stream requests exceeds the maximum number allowed.

HTTP Status Code: 400
ResourceNotFoundException

The specified resource does not exist.

HTTP Status Code: 400

ThrottlingException

AWS RoboMaker is temporarily unable to process the request. Try your call again.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3
TagResource

Adds or edits tags for a AWS RoboMaker resource.

Each tag consists of a tag key and a tag value. Tag keys and tag values are both required, but tag values can be empty strings.

For information about the rules that apply to tag keys and tag values, see User-Defined Tag Restrictions in the AWS Billing and Cost Management User Guide.

Request Syntax

```
POST /tags/resourceArn HTTP/1.1
Content-type: application/json
{
    "tags": {
        "String" : "string"
    }
}
```

URI Request Parameters

The request requires the following URI parameters.

resourceArn (p. 275)

- The Amazon Resource Name (ARN) of the AWS RoboMaker resource you are tagging.
- Pattern: arn:.*

Request Body

The request accepts the following data in JSON format.

tags (p. 275)

- A map that contains tag keys and tag values that are attached to the resource.
- Type: String to string map
- Key Pattern: [a-zA-Z0-9_.\-\+/=]*
- Value Length Constraints: Minimum length of 0. Maximum length of 256.
- Value Pattern: [a-zA-Z0-9_.\-\+/=]*
- Required: Yes

Response Syntax

```
HTTP/1.1 200
```
Response Elements

If the action is successful, the service sends back an HTTP 200 response with an empty HTTP body.

Errors

For information about the errors that are common to all actions, see Common Errors (p. 342).

InternalServerException

AWS RoboMaker experienced a service issue. Try your call again.

HTTP Status Code: 500

InvalidParameterException

A parameter specified in a request is not valid, is unsupported, or cannot be used. The returned message provides an explanation of the error value.

HTTP Status Code: 400

ResourceNotFoundException

The specified resource does not exist.

HTTP Status Code: 400

ThrottlingException

AWS RoboMaker is temporarily unable to process the request. Try your call again.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3
UntagResource

Removes the specified tags from the specified AWS RoboMaker resource.

To remove a tag, specify the tag key. To change the tag value of an existing tag key, use TagResource.

Request Syntax

```
DELETE /tags/resourceArn?tagKeys=tagKeys HTTP/1.1
```

URI Request Parameters

The request requires the following URI parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>resourceArn</td>
<td>The Amazon Resource Name (ARN) of the AWS RoboMaker resource you are removing tags.</td>
</tr>
<tr>
<td></td>
<td>Pattern: arn:.*</td>
</tr>
<tr>
<td>tagKeys</td>
<td>A map that contains tag keys and tag values that will be unattached from the resource.</td>
</tr>
<tr>
<td></td>
<td>Pattern: [a-zA-Z0-9_.-/+=:]*</td>
</tr>
</tbody>
</table>

Request Body

The request does not have a request body.

Response Syntax

```
HTTP/1.1 200
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response with an empty HTTP body.

Errors

For information about the errors that are common to all actions, see Common Errors.

**InternalServerException**

AWS RoboMaker experienced a service issue. Try your call again.

HTTP Status Code: 500

**InvalidParameterException**

A parameter specified in a request is not valid, is unsupported, or cannot be used. The returned message provides an explanation of the error value.
HTTP Status Code: 400

**ResourceNotFoundException**

The specified resource does not exist.

HTTP Status Code: 400

**ThrottlingException**

AWS RoboMaker is temporarily unable to process the request. Try your call again.

HTTP Status Code: 400

**See Also**

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3
UpdateRobotApplication

Updates a robot application.

Request Syntax

```
POST /updateRobotApplication HTTP/1.1
Content-type: application/json

{
  "application": "string",
  "currentRevisionId": "string",
  "robotSoftwareSuite": {
    "name": "string",
    "version": "string"
  },
  "sources": [
    {
      "architecture": "string",
      "s3Bucket": "string",
      "s3Key": "string"
    }
  ]
}
```

URI Request Parameters

The request does not use any URI parameters.

Request Body

The request accepts the following data in JSON format.

**application (p. 279)**

The application information for the robot application.

- **Type:** String
- **Length Constraints:** Minimum length of 1. Maximum length of 1224.
- **Pattern:** arn:.*
- **Required:** Yes

**currentRevisionId (p. 279)**

The revision id for the robot application.

- **Type:** String
- **Length Constraints:** Minimum length of 1. Maximum length of 40.
- **Pattern:** [a-zA-Z0-9_.\-]*
- **Required:** No

**robotSoftwareSuite (p. 279)**

The robot software suite (ROS distribution) used by the robot application.
Type: **RobotSoftwareSuite (p. 320)** object

Required: Yes

**sources (p. 279)**

The sources of the robot application.

Type: Array of **SourceConfig (p. 339)** objects

Required: Yes

**Response Syntax**

```json
HTTP/1.1 200
Content-type: application/json

{
    "arn": "string",
    "lastUpdatedAt": number,
    "name": "string",
    "revisionId": "string",
    "robotSoftwareSuite": {
        "name": "string",
        "version": "string"
    },
    "sources": [
        {
            "architecture": "string",
            "etag": "string",
            "s3Bucket": "string",
            "s3Key": "string"
        }
    ],
    "version": "string"
}
```

**Response Elements**

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

**arn (p. 280)**

The Amazon Resource Name (ARN) of the updated robot application.

Type: String


Pattern: arn::*

**lastUpdatedAt (p. 280)**

The time, in milliseconds since the epoch, when the robot application was last updated.

Type: Timestamp

**name (p. 280)**

The name of the robot application.
Type: String
Length Constraints: Minimum length of 1. Maximum length of 255.
Pattern: [a-zA-Z0-9_\-]*

**revisionId (p. 280)**

The revision id of the robot application.

Type: String
Pattern: [a-zA-Z0-9_.\-]*

**robotSoftwareSuite (p. 280)**

The robot software suite (ROS distribution) used by the robot application.

Type: RobotSoftwareSuite (p. 320) object

**sources (p. 280)**

The sources of the robot application.

Type: Array of Source (p. 338) objects

**version (p. 280)**

The version of the robot application.

Type: String
Length Constraints: Minimum length of 1. Maximum length of 255.
Pattern: (\$LATEST)|[0-9]*

---

**Errors**

For information about the errors that are common to all actions, see Common Errors (p. 342).

**InternalServerException**

AWS RoboMaker experienced a service issue. Try your call again.

HTTP Status Code: 500

**InvalidParameterException**

A parameter specified in a request is not valid, is unsupported, or cannot be used. The returned message provides an explanation of the error value.

HTTP Status Code: 400

**LimitExceededException**

The requested resource exceeds the maximum number allowed, or the number of concurrent stream requests exceeds the maximum number allowed.

HTTP Status Code: 400

**ResourceNotFoundException**

The specified resource does not exist.
HTTP Status Code: 400

**ThrottlingException**

AWS RoboMaker is temporarily unable to process the request. Try your call again.

HTTP Status Code: 400

**See Also**

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3
# UpdateSimulationApplication

Updates a simulation application.

## Request Syntax

```
POST /updateSimulationApplication HTTP/1.1
Content-type: application/json

{
  "application": "string",
  "currentRevisionId": "string",
  "renderingEngine": {
    "name": "string",
    "version": "string"
  },
  "robotSoftwareSuite": {
    "name": "string",
    "version": "string"
  },
  "simulationSoftwareSuite": {
    "name": "string",
    "version": "string"
  },
  "sources": [
    {
      "architecture": "string",
      "s3Bucket": "string",
      "s3Key": "string"
    }
  ]
}
```

## URI Request Parameters

The request does not use any URI parameters.

## Request Body

The request accepts the following data in JSON format.

### application (p. 283)

The application information for the simulation application.

- **Type:** String
- **Length Constraints:** Minimum length of 1. Maximum length of 1224.
- **Pattern:** `arn:*`
- **Required:** Yes

### currentRevisionId (p. 283)

The revision id for the robot application.

- **Type:** String
- **Length Constraints:** Minimum length of 1. Maximum length of 40.
Pattern: [a-zA-Z0-9_.\-]*

Required: No

**renderingEngine (p. 283)**

The rendering engine for the simulation application.

Type: RenderingEngine (p. 312) object

Required: No

**robotSoftwareSuite (p. 283)**

Information about the robot software suite (ROS distribution).

Type: RobotSoftwareSuite (p. 320) object

Required: Yes

**simulationSoftwareSuite (p. 283)**

The simulation software suite used by the simulation application.

Type: SimulationSoftwareSuite (p. 337) object

Required: Yes

**sources (p. 283)**

The sources of the simulation application.

Type: Array of SourceConfig (p. 339) objects

Required: Yes

**Response Syntax**

```json
HTTP/1.1 200
Content-type: application/json

{
  "arn": "string",
  "lastUpdatedAt": number,
  "name": "string",
  "renderingEngine": {
    "name": "string",
    "version": "string"
  },
  "revisionId": "string",
  "robotSoftwareSuite": {
    "name": "string",
    "version": "string"
  },
  "simulationSoftwareSuite": {
    "name": "string",
    "version": "string"
  },
  "sources": [
    {
      "architecture": "string",
      "etag": "string",
      "s3Bucket": "string",
      "s3Key": "string"
    }
  ]
}
```
Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

**arn (p. 284)**

The Amazon Resource Name (ARN) of the updated simulation application.

Type: String


Pattern: `arn:`.*

**lastUpdatedAt (p. 284)**

The time, in milliseconds since the epoch, when the simulation application was last updated.

Type: Timestamp

**name (p. 284)**

The name of the simulation application.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 255.

Pattern: `[a-zA-Z0-9_\-]*`

**renderingEngine (p. 284)**

The rendering engine for the simulation application.

Type: RenderingEngine (p. 312) object

**revisionId (p. 284)**

The revision id of the simulation application.

Type: String


Pattern: `[a-zA-Z0-9_]\-]*`

**robotSoftwareSuite (p. 284)**

Information about the robot software suite (ROS distribution).

Type: RobotSoftwareSuite (p. 320) object

**simulationSoftwareSuite (p. 284)**

The simulation software suite used by the simulation application.

Type: SimulationSoftwareSuite (p. 337) object
sources (p. 284)
The sources of the simulation application.
Type: Array of Source (p. 338) objects

version (p. 284)
The version of the robot application.
Type: String
Length Constraints: Minimum length of 1. Maximum length of 255.
Pattern: (\$LATEST) | [0-9]*

Errors
For information about the errors that are common to all actions, see Common Errors (p. 342).

InternalServerException
AWS RoboMaker experienced a service issue. Try your call again.
HTTP Status Code: 500

InvalidParameterException
A parameter specified in a request is not valid, is unsupported, or cannot be used. The returned message provides an explanation of the error value.
HTTP Status Code: 400

LimitExceededException
The requested resource exceeds the maximum number allowed, or the number of concurrent stream requests exceeds the maximum number allowed.
HTTP Status Code: 400

ResourceNotFoundException
The specified resource does not exist.
HTTP Status Code: 400

ThrottlingException
AWS RoboMaker is temporarily unable to process the request. Try your call again.
HTTP Status Code: 400

See Also
For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
Data Types

The following data types are supported:

- BatchPolicy (p. 289)
- DataSource (p. 290)
- DataSourceConfig (p. 291)
- DeploymentApplicationConfig (p. 292)
- DeploymentConfig (p. 293)
- DeploymentJob (p. 294)
- DeploymentLaunchConfig (p. 296)
- FailedCreateSimulationJobRequest (p. 298)
- Filter (p. 300)
- Fleet (p. 301)
- LaunchConfig (p. 303)
- LoggingConfig (p. 305)
- NetworkInterface (p. 306)
- OutputLocation (p. 307)
- PortForwardingConfig (p. 308)
- PortMapping (p. 309)
- ProgressDetail (p. 310)
- RenderingEngine (p. 312)
- Robot (p. 313)
- RobotApplicationConfig (p. 315)
- RobotApplicationSummary (p. 316)
- RobotDeployment (p. 318)
- RobotSoftwareSuite (p. 320)
- S3KeyOutput (p. 321)
- S3Object (p. 322)
- SimulationApplicationConfig (p. 323)
- SimulationApplicationSummary (p. 324)
- SimulationJob (p. 326)
- SimulationJobBatchSummary (p. 330)
- SimulationJobRequest (p. 332)
- SimulationJobSummary (p. 335)
- SimulationSoftwareSuite (p. 337)
- Source (p. 338)
- SourceConfig (p. 339)
- VPCCConfig (p. 340)
- VPCCConfigResponse (p. 341)
BatchPolicy

Information about the batch policy.

Contents

maxConcurrency

The number of active simulation jobs created as part of the batch that can be in an active state at the same time.

Active states include: Pending, Preparing, Running, Restarting, RunningFailed and Terminating. All other states are terminal states.

Type: Integer

Required: No

timeoutInSeconds

The amount of time, in seconds, to wait for the batch to complete.

If a batch times out, and there are pending requests that were failing due to an internal failure (like InternalServiceError), they will be moved to the failed list and the batch status will be Failed. If the pending requests were failing for any other reason, the failed pending requests will be moved to the failed list and the batch status will be TimedOut.

Type: Long

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for Ruby V3
**DataSource**

Information about a data source.

**Contents**

**name**

The name of the data source.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 255.

Pattern: `[a-zA-Z0-9\-_]*`

Required: No

**s3Bucket**

The S3 bucket where the data files are located.

Type: String


Pattern: `[a-z0-9][a-z0-9.\-]*[a-z0-9]`

Required: No

**s3Keys**

The list of S3 keys identifying the data source files.

Type: Array of [S3KeyOutput](p. 321) objects

Required: No

**See Also**

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for Ruby V3
**DataSourceConfig**

Information about a data source.

**Contents**

**name**

The name of the data source.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 255.

Pattern: [a-zA-Z0-9_\-]*

Required: Yes

**s3Bucket**

The S3 bucket where the data files are located.

Type: String


Pattern: [a-z0-9][a-z0-9.\-]*[a-z0-9]

Required: Yes

**s3Keys**

The list of S3 keys identifying the data source files.

Type: Array of strings

Array Members: Minimum number of 1 item. Maximum number of 100 items.


Pattern: .*

Required: Yes

**See Also**

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for Ruby V3
DeploymentApplicationConfig

Information about a deployment application configuration.

Contents

application

The Amazon Resource Name (ARN) of the robot application.

Type: String


Pattern: arn:.*

Required: Yes

applicationVersion

The version of the application.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 255.

Pattern: [0-9]*

Required: Yes

launchConfig

The launch configuration.

Type: DeploymentLaunchConfig (p. 296) object

Required: Yes

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for Ruby V3
DeploymentConfig

Information about a deployment configuration.

Contents

concurrentDeploymentPercentage

The percentage of robots receiving the deployment at the same time.

Type: Integer

Valid Range: Minimum value of 1. Maximum value of 100.

Required: No

downloadConditionFile

The download condition file.

Type: S3Object (p. 322) object

Required: No

failureThresholdPercentage

The percentage of deployments that need to fail before stopping deployment.

Type: Integer

Valid Range: Minimum value of 1. Maximum value of 100.

Required: No

robotDeploymentTimeoutInSeconds

The amount of time, in seconds, to wait for deployment to a single robot to complete. Choose a time between 1 minute and 7 days. The default is 5 hours.

Type: Long

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for Ruby V3
DeploymentJob

Information about a deployment job.

Contents

arn

The Amazon Resource Name (ARN) of the deployment job.

Type: String


Pattern: arn:.*

Required: No

createdAt

The time, in milliseconds since the epoch, when the deployment job was created.

Type: Timestamp

Required: No

deploymentApplicationConfigs

The deployment application configuration.

Type: Array of DeploymentApplicationConfig (p. 292) objects

Array Members: Fixed number of 1 item.

Required: No

deploymentConfig

The deployment configuration.

Type: DeploymentConfig (p. 293) object

Required: No

failureCode

The deployment job failure code.

Type: String


Required: No
**failureReason**
A short description of the reason why the deployment job failed.

Type: String
Length Constraints: Minimum length of 0. Maximum length of 1024.
Pattern: . *
Required: No

**fleet**
The Amazon Resource Name (ARN) of the fleet.

Type: String
Pattern: arn:.*
Required: No

**status**
The status of the deployment job.

Type: String
Valid Values: Pending | Preparing | InProgress | Failed | Succeeded | Canceled
Required: No

**See Also**
For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for Ruby V3
DeploymentLaunchConfig

Configuration information for a deployment launch.

Contents

environmentVariables

An array of key/value pairs specifying environment variables for the robot application.

Type: String to string map

Key Length Constraints: Minimum length of 1. Maximum length of 1024.

Key Pattern: [A-Z_.][A-Z0-9_.]*


Value Pattern: . *

Required: No

launchFile

The launch file name.

Type: String


Pattern: [a-zA-Z0-9_.\-]*

Required: Yes

packageName

The package name.

Type: String


Pattern: [a-zA-Z0-9_.\-]*

Required: Yes

postLaunchFile

The deployment post-launch file. This file will be executed after the launch file.

Type: String


Pattern: . *

Required: No

preLaunchFile

The deployment pre-launch file. This file will be executed prior to the launch file.

Type: String

Pattern: . *

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for Ruby V3
FailedCreateSimulationJobRequest

Information about a failed create simulation job request.

Contents

failedAt

The time, in milliseconds since the epoch, when the simulation job batch failed.

Type: Timestamp

Required: No

failureCode

The failure code.

Type: String

Valid Values: InternalServiceError | RobotApplicationCrash | SimulationApplicationCrash | BadPermissionsRobotApplication | BadPermissionsSimulationApplication | BadPermissionsS3Object | BadPermissionsS3Output | BadPermissionsCloudwatchLogs | SubnetIpLimitExceeded | ENILimitExceeded | BadPermissionsUserCredentials | InvalidBundleRobotApplication | InvalidBundleSimulationApplication | InvalidS3Resource | LimitExceeded | MismatchedEtag | RobotApplicationVersionMismatchedEtag | SimulationApplicationVersionMismatchedEtag | ResourceNotFound | RequestThrottled | BatchTimedOut | BatchCanceled | InvalidInput | WrongRegionS3Bucket | WrongRegionS3Output | WrongRegionRobotApplication | WrongRegionSimulationApplication

Required: No

failureReason

The failure reason of the simulation job request.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 1024.

Pattern: .*

Required: No

request

The simulation job request.

Type: SimulationJobRequest (p. 332) object

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
• AWS SDK for Go
• AWS SDK for Java
• AWS SDK for Ruby V3
Filter

Information about a filter.

Contents

name

The name of the filter.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 255.

Pattern: [a-zA-Z0-9_-]*

Required: No

values

A list of values.

Type: Array of strings

Array Members: Fixed number of 1 item.

Length Constraints: Minimum length of 1. Maximum length of 255.

Pattern: [a-zA-Z0-9_-]*

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for Ruby V3
Fleet

Information about a fleet.

Contents

**arn**

The Amazon Resource Name (ARN) of the fleet.

Type: String


Pattern: `arn:`.*

Required: No

**createdAt**

The time, in milliseconds since the epoch, when the fleet was created.

Type: Timestamp

Required: No

**lastDeploymentJob**

The Amazon Resource Name (ARN) of the last deployment job.

Type: String


Pattern: `arn:`.*

Required: No

**lastDeploymentStatus**

The status of the last fleet deployment.

Type: String

Valid Values: Pending | Preparing | InProgress | Failed | Succeeded | Canceled

Required: No

**lastDeploymentTime**

The time of the last deployment.

Type: Timestamp

Required: No

**name**

The name of the fleet.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 255.
Pattern: [a-zA-Z0-9_]*

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for Ruby V3
LaunchConfig

Information about a launch configuration.

Contents

environmentVariables

The environment variables for the application launch.

Type: String to string map

Key Length Constraints: Minimum length of 1. Maximum length of 1024.

Key Pattern: \[A-Z_]*\[A-Z0-9_]*


Value Pattern: .*

Required: No

launchFile

The launch file name.

Type: String


Pattern: \[a-zA-Z0-9_.\-]*

Required: Yes

packageName

The package name.

Type: String


Pattern: \[a-zA-Z0-9_.\-]*

Required: Yes

portForwardingConfig

The port forwarding configuration.

Type: PortForwardingConfig (p. 308) object

Required: No

streamUI

Boolean indicating whether a streaming session will be configured for the application. If True, AWS RoboMaker will configure a connection so you can interact with your application as it is running in the simulation. You must configure and launch the component. It must have a graphical user interface.

Type: Boolean

Required: No
See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for Ruby V3
LoggingConfig

The logging configuration.

Contents

recordAllRosTopics

A boolean indicating whether to record all ROS topics.

Type: Boolean

Required: Yes

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for Ruby V3
NetworkInterface

Describes a network interface.

Contents

**networkInterfaceId**

The ID of the network interface.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 1024.

Pattern: . *

Required: No

**privateIpAddress**

The IPv4 address of the network interface within the subnet.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 1024.

Pattern: . *

Required: No

**publicIpAddress**

The IPv4 public address of the network interface.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 1024.

Pattern: . *

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for Ruby V3
OutputLocation

The output location.

Contents

s3Bucket

The S3 bucket for output.

Type: String


Pattern: [a-z0-9][a-z0-9.\-]*[a-z0-9]

Required: No

s3Prefix

The S3 folder in the s3Bucket where output files will be placed.

Type: String


Pattern: .*

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for Ruby V3
PortForwardingConfig

Configuration information for port forwarding.

Contents

portMappings

The port mappings for the configuration.

Type: Array of PortMapping (p. 309) objects

Array Members: Minimum number of 0 items. Maximum number of 10 items.

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for Ruby V3
PortMapping

An object representing a port mapping.

Contents

applicationPort

The port number on the application.

Type: Integer


Required: Yes

enableOnPublicIp

A Boolean indicating whether to enable this port mapping on public IP.

Type: Boolean

Required: No

jobPort

The port number on the simulation job instance to use as a remote connection point.

Type: Integer


Required: Yes

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for Ruby V3
ProgressDetail

Information about the progress of a deployment job.

Contents

currentProgress

The current progress status.
Validating

Validating the deployment.
DownloadingExtracting

Downloading and extracting the bundle on the robot.
ExecutingPreLaunch

Executing pre-launch script(s) if provided.
Launching

Launching the robot application.
ExecutingPostLaunch

Executing post-launch script(s) if provided.
Finished

Deployment is complete.

Type: String

Valid Values: Validating | DownloadingExtracting | ExecutingDownloadCondition | ExecutingPreLaunch | Launching | ExecutingPostLaunch | Finished

Required: No

estimatedTimeRemainingSeconds

Estimated amount of time in seconds remaining in the step. This currently only applies to the Downloading/Extracting step of the deployment. It is empty for other steps.

Type: Integer

Required: No

percentDone

Percentage of the step that is done. This currently only applies to the Downloading/Extracting step of the deployment. It is empty for other steps.

Type: Float

Valid Range: Minimum value of 0.0. Maximum value of 100.0.

Required: No

targetResource

The Amazon Resource Name (ARN) of the deployment job.

Type: String
Length Constraints: Minimum length of 0. Maximum length of 1024.

Pattern: . *

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for Ruby V3
RenderingEngine

Information about a rendering engine.

Contents

name

The name of the rendering engine.
Type: String
Valid Values: OGRE
Required: No

version

The version of the rendering engine.
Type: String
Pattern: 1.x
Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for Ruby V3
Robot
Information about a robot.

Contents

architecture
The architecture of the robot.
Type: String
Valid Values: X86_64 | ARM64 | ARMHF
Required: No

arn
The Amazon Resource Name (ARN) of the robot.
Type: String
Pattern: arn:.*
Required: No

createdAt
The time, in milliseconds since the epoch, when the robot was created.
Type: Timestamp
Required: No

fleetArn
The Amazon Resource Name (ARN) of the fleet.
Type: String
Pattern: arn:.*
Required: No

greenGrassGroupId
The Greengrass group associated with the robot.
Type: String
Pattern: .*
Required: No

lastDeploymentJob
The Amazon Resource Name (ARN) of the last deployment job.
Type: String
Pattern: arn:.*
Required: No

lastDeploymentTime

The time of the last deployment.
Type: Timestamp
Required: No

name

The name of the robot.
Type: String
Length Constraints: Minimum length of 1. Maximum length of 255.
Pattern: [a-zA-Z0-9\-_]*
Required: No

status

The status of the robot.
Type: String
Valid Values: Available | Registered | PendingNewDeployment | Deploying | Failed | InSync | NoResponse
Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for Ruby V3
RobotApplicationConfig

Application configuration information for a robot.

Contents

application

The application information for the robot application.
Type: String
Pattern: arn:.*
Required: Yes

applicationVersion

The version of the robot application.
Type: String
Length Constraints: Minimum length of 1. Maximum length of 255.
Pattern: (\$LATEST) | \[0-9]*
Required: No

launchConfig

The launch configuration for the robot application.
Type: LaunchConfig (p. 303) object
Required: Yes

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for Ruby V3
RobotApplicationSummary

Summary information for a robot application.

Contents

**arn**

The Amazon Resource Name (ARN) of the robot.

Type: String


Pattern: `arn:*`

Required: No

**lastUpdatedAt**

The time, in milliseconds since the epoch, when the robot application was last updated.

Type: Timestamp

Required: No

**name**

The name of the robot application.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 255.

Pattern: `[^\x00-\x1f\x7f\-\_\ ]*`

Required: No

**robotSoftwareSuite**

Information about a robot software suite (ROS distribution).

Type: `RobotSoftwareSuite (p. 320)` object

Required: No

**version**

The version of the robot application.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 255.

Pattern: `(\$LATEST)|[0-9]*`

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:
• AWS SDK for C++
• AWS SDK for Go
• AWS SDK for Java
• AWS SDK for Ruby V3
RobotDeployment

Information about a robot deployment.

Contents

**arn**

The robot deployment Amazon Resource Name (ARN).

Type: String


Pattern: arn:.*

Required: No

**deploymentFinishTime**

The time, in milliseconds since the epoch, when the deployment finished.

Type: Timestamp

Required: No

**deploymentStartTime**

The time, in milliseconds since the epoch, when the deployment was started.

Type: Timestamp

Required: No

**failureCode**

The robot deployment failure code.

Type: String


Required: No

**failureReason**

A short description of the reason why the robot deployment failed.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 1024.

Pattern: .*

Required: No
**progressDetail**

Information about how the deployment is progressing.

Type: **ProgressDetail (p. 310) object**

Required: No

**status**

The status of the robot deployment.

Type: String

Valid Values: Available | Registered | PendingNewDeployment | Deploying | Failed | InSync | NoResponse

Required: No

**See Also**

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for Ruby V3
RobotSoftwareSuite

Information about a robot software suite (ROS distribution).

Contents

name

The name of the robot software suite (ROS distribution).

Type: String

Valid Values: ROS | ROS2

Required: No

version

The version of the robot software suite (ROS distribution).

Type: String

Valid Values: Kinetic | Melodic | Dashing

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for Ruby V3
S3KeyOutput

Information about S3 keys.

Contents

etag

The etag for the object.

Type: String

Required: No

s3Key

The S3 key.

Type: String


Pattern: .*

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for Ruby V3
S3Object

Information about an S3 object.

Contents

bucket

The bucket containing the object.

Type: String


Pattern: [a-z0-9][a-z0-9.\-]*[a-z0-9]

Required: Yes

etag

The etag of the object.

Type: String

Required: No

key

The key of the object.

Type: String


Pattern: .*

Required: Yes

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for Ruby V3
SimulationApplicationConfig

Information about a simulation application configuration.

Contents

application

The application information for the simulation application.

Type: String


Pattern: arn:.*

Required: Yes

applicationVersion

The version of the simulation application.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 255.

Pattern: (\$LATEST)|([0-9]*)

Required: No

launchConfig

The launch configuration for the simulation application.

Type: LaunchConfig (p. 303) object

Required: Yes

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for Ruby V3
SimulationApplicationSummary

Summary information for a simulation application.

Contents

arn

The Amazon Resource Name (ARN) of the simulation application.

Type: String


Pattern: arn:.*

Required: No

lastUpdatedAt

The time, in milliseconds since the epoch, when the simulation application was last updated.

Type: Timestamp

Required: No

name

The name of the simulation application.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 255.

Pattern: [a-zA-Z0-9_\-]*

Required: No

robotSoftwareSuite

Information about a robot software suite (ROS distribution).

Type: RobotSoftwareSuite (p. 320) object

Required: No

simulationSoftwareSuite

Information about a simulation software suite.

Type: SimulationSoftwareSuite (p. 337) object

Required: No

version

The version of the simulation application.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 255.

Pattern: (\$LATEST)|[0-9]*
Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for Ruby V3
**SimulationJob**

Information about a simulation job.

**Contents**

**arn**

The Amazon Resource Name (ARN) of the simulation job.

Type: String


Pattern: `arn:*`

Required: No

**clientRequestToken**

A unique identifier for this SimulationJob request.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 64.

Pattern: `[a-zA-Z0-9_-=]*`

Required: No

**dataSources**

The data sources for the simulation job.

Type: Array of `DataSource` objects

Required: No

**failureBehavior**

The failure behavior the simulation job.

- Continue
  
  Restart the simulation job in the same host instance.

- Fail
  
  Stop the simulation job and terminate the instance.

Type: String

Valid Values: Fail | Continue

Required: No

**failureCode**

The failure code of the simulation job if it failed.

Type: String

Valid Values: `InternalServiceError` | `RobotApplicationCrash` | `SimulationApplicationCrash` | `BadPermissionsRobotApplication` | `BadPermissionsSimulationApplication` | `BadPermissionsS3Object`
SimulatedJob

| BadPermissionsS3Output | BadPermissionsCloudWatchLogs
| SubnetIpLimitExceeded | ENILimitExceeded |
| BadPermissionsUserCredentials | InvalidBundleRobotApplication |
| InvalidBundleSimulationApplication | InvalidS3Resource | LimitExceeded |
| MismatchedEtag | RobotApplicationVersionMismatchedEtag |
| SimulationApplicationVersionMismatchedEtag | ResourceNotFind |
| RequestThrottled | BatchTimedOut | BatchCanceled | InvalidInput |
| WrongRegionS3Bucket | WrongRegionS3Output | WrongRegionRobotApplication |
| WrongRegionSimulationApplication

- **failureReason**
  The reason why the simulation job failed.
  
  *Type: String*
  
  *Length Constraints: Minimum length of 0. Maximum length of 1024.*
  
  *Pattern: .*
  
  *Required: No*

- **iamRole**
  The IAM role that allows the simulation instance to call the AWS APIs that are specified in its associated policies on your behalf. This is how credentials are passed in to your simulation job.
  
  *Type: String*
  
  
  *Pattern: arn:aws:iam::\w+:role/.*
  
  *Required: No*

- **lastStartedAt**
  The time, in milliseconds since the epoch, when the simulation job was last started.
  
  *Type: Timestamp*
  
  *Required: No*

- **lastUpdatedAt**
  The time, in milliseconds since the epoch, when the simulation job was last updated.
  
  *Type: Timestamp*
  
  *Required: No*

- **loggingConfig**
  The logging configuration.
  
  *Type: LoggingConfig (p. 305) object*
  
  *Required: No*

- **maxJobDurationInSeconds**
  The maximum simulation job duration in seconds. The value must be 8 days (691,200 seconds) or less.
SimulationJob

Type: Long
Required: No

name
The name of the simulation job.
Type: String
Length Constraints: Minimum length of 1. Maximum length of 255.
Pattern: [a-zA-Z0-9_\-]*
Required: No

networkInterface
Information about a network interface.
Type: NetworkInterface (p. 306) object
Required: No

outputLocation
Location for output files generated by the simulation job.
Type: OutputLocation (p. 307) object
Required: No

robotApplications
A list of robot applications.
Type: Array of RobotApplicationConfig (p. 315) objects
Array Members: Fixed number of 1 item.
Required: No

simulationApplications
A list of simulation applications.
Type: Array of SimulationApplicationConfig (p. 323) objects
Array Members: Fixed number of 1 item.
Required: No

simulationTimeMillis
The simulation job execution duration in milliseconds.
Type: Long
Required: No

status
Status of the simulation job.
Type: String
Valid Values: Pending | Preparing | Running | Restarting | Completed | Failed | RunningFailed | Terminating | Terminated | Canceled
Required: No

**tags**

A map that contains tag keys and tag values that are attached to the simulation job.

Type: String to string map

Key Length Constraints: Minimum length of 1. Maximum length of 128.

Key Pattern: [a-zA-Z0-9 _.-\-/+/=:]*

Value Length Constraints: Minimum length of 0. Maximum length of 256.

Value Pattern: [a-zA-Z0-9 _.-\-/+/=:]*

Required: No

**vpcConfig**

VPC configuration information.

Type: VPCConfigResponse (p. 341) object

Required: No

**See Also**

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for Ruby V3
SimulationJobBatchSummary

Information about a simulation job batch.

Contents

arn

The Amazon Resource Name (ARN) of the batch.

Type: String


Pattern: arn:.*

Required: No

createdAt

The time, in milliseconds since the epoch, when the simulation job batch was created.

Type: Timestamp

Required: No

createdRequestCount

The number of created simulation job requests.

Type: Integer

Required: No

failedRequestCount

The number of failed simulation job requests.

Type: Integer

Required: No

lastUpdatedAt

The time, in milliseconds since the epoch, when the simulation job batch was last updated.

Type: Timestamp

Required: No

pendingRequestCount

The number of pending simulation job requests.

Type: Integer

Required: No

status

The status of the simulation job batch.

Pending

The simulation job batch request is pending.
In Progress

The simulation job batch is in progress.

Failed

The simulation job batch failed. One or more simulation job requests could not be completed due to an internal failure (like InternalServiceError). See failureCode and failureReason for more information.

Completed

The simulation batch job completed. A batch is complete when (1) there are no pending simulation job requests in the batch and none of the failed simulation job requests are due to InternalServiceError and (2) when all created simulation jobs have reached a terminal state (for example, Completed or Failed).

Canceled

The simulation batch job was cancelled.

Canceling

The simulation batch job is being cancelled.

Completing

The simulation batch job is completing.

Timing Out

The simulation job batch is timing out.

If a batch timing out, and there are pending requests that were failing due to an internal failure (like InternalServiceError), the batch status will be Failed. If there are no such failing request, the batch status will be TimedOut.

Timed Out

The simulation batch job timed out.

Type: String

Valid Values: Pending | InProgress | Failed | Completed | Canceled | Canceling | Completing | Timing Out | Timed Out

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for Ruby V3
**SimulationJobRequest**

Information about a simulation job request.

**Contents**

**dataSources**

Specify data sources to mount read-only files from S3 into your simulation. These files are available under `/opt/robomaker/datasources/data_source_name`.

**Note**

There is a limit of 100 files and a combined size of 25GB for all `DataSourceConfig` objects.

Type: Array of `DataSourceConfig (p. 291)` objects

Array Members: Minimum number of 1 item. Maximum number of 5 items.

Required: No

**failureBehavior**

The failure behavior the simulation job.

*Continue*

Restart the simulation job in the same host instance.

*Fail*

Stop the simulation job and terminate the instance.

Type: String

Valid Values: Fail | Continue

Required: No

**iamRole**

The IAM role name that allows the simulation instance to call the AWS APIs that are specified in its associated policies on your behalf. This is how credentials are passed in to your simulation job.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 255.

Pattern: `arn:aws:iam::\w+:role/*`

Required: No

**loggingConfig**

The logging configuration.

Type: `LoggingConfig (p. 305)` object

Required: No

**maxJobDurationInSeconds**

The maximum simulation job duration in seconds. The value must be 8 days (691,200 seconds) or less.
Type: Long
Required: Yes

**outputLocation**

The output location.

Type: `OutputLocation (p. 307)` object

Required: No

**robotApplications**

The robot applications to use in the simulation job.

Type: `Array of RobotApplicationConfig (p. 315)` objects

Array Members: Fixed number of 1 item.

Required: No

**simulationApplications**

The simulation applications to use in the simulation job.

Type: `Array of SimulationApplicationConfig (p. 323)` objects

Array Members: Fixed number of 1 item.

Required: No

**tags**

A map that contains tag keys and tag values that are attached to the simulation job request.

Type: String to string map

Key Length Constraints: Minimum length of 1. Maximum length of 128.

Key Pattern: `[^a-zA-Z0-9 _\-\./+=:]*`

Value Length Constraints: Minimum length of 0. Maximum length of 256.

Value Pattern: `[^a-zA-Z0-9 _\-\./+=:]*`

Required: No

**useDefaultApplications**

Boolean indicating whether to use default simulation tool applications.

Type: Boolean

Required: No

**vpcConfig**

If your simulation job accesses resources in a VPC, you provide this parameter identifying the list of security group IDs and subnet IDs. These must belong to the same VPC. You must provide at least one security group and two subnet IDs.

Type: `VPCConfig (p. 340)` object

Required: No
See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for Ruby V3
SimulationJobSummary

Summary information for a simulation job.

Contents

**arn**

The Amazon Resource Name (ARN) of the simulation job.

Type: String


Pattern: `arn:`.*

Required: No

**dataSourceNames**

The names of the data sources.

Type: Array of strings

Length Constraints: Minimum length of 1. Maximum length of 255.

Pattern: `[a-zA-Z0-9_\-]*`

Required: No

**lastUpdatedAt**

The time, in milliseconds since the epoch, when the simulation job was last updated.

Type: Timestamp

Required: No

**name**

The name of the simulation job.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 255.

Pattern: `[a-zA-Z0-9_\-]*`

Required: No

**robotApplicationNames**

A list of simulation job robot application names.

Type: Array of strings

Length Constraints: Minimum length of 1. Maximum length of 255.

Pattern: `[a-zA-Z0-9_\-]*`

Required: No

**simulationApplicationNames**

A list of simulation job simulation application names.
Type: Array of strings

Length Constraints: Minimum length of 1. Maximum length of 255.

Pattern: \[a-zA-Z0-9_\-]*

Required: No

**status**

The status of the simulation job.

Type: String

Valid Values: Pending | Preparing | Running | Restarting | Completed | Failed | RunningFailed | Terminating | Terminated | Canceled

Required: No

**See Also**

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for Ruby V3
SimulationSoftwareSuite

Information about a simulation software suite.

Contents

name

The name of the simulation software suite.

Type: String

Valid Values: Gazebo | RosbagPlay

Required: No

version

The version of the simulation software suite.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 1024.

Pattern: 7|9|Kinetic|Melodic|Dashing

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for Ruby V3
Source

Information about a source.

Contents

architecture

The taget processor architecture for the application.

Type: String

Valid Values: X86_64 | ARM64 | ARMHF

Required: No
eTag

A hash of the object specified by s3Bucket and s3Key.

Type: String

Required: No

s3Bucket

The s3 bucket name.

Type: String


Pattern: [a-z0-9][a-z0-9.\-]*[a-z0-9]

Required: No

s3Key

The s3 object key.

Type: String


Pattern: .*

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for Ruby V3
SourceConfig

Information about a source configuration.

Contents

architecture

The target processor architecture for the application.

Type: String

Valid Values: X86_64 | ARM64 | ARMHF

Required: No

s3Bucket

The Amazon S3 bucket name.

Type: String


Pattern: [a-z0-9][a-zA-Z0-9._-]*[a-z0-9]

Required: No

s3Key

The s3 object key.

Type: String


Pattern: .*

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for Ruby V3
VPCCfg

If your simulation job accesses resources in a VPC, you provide this parameter identifying the list of security group IDs and subnet IDs. These must belong to the same VPC. You must provide at least one security group and two subnet IDs.

Contents

assignPublicIp

A boolean indicating whether to assign a public IP address.

Type: Boolean

Required: No

securityGroups

A list of one or more security groups IDs in your VPC.

Type: Array of strings

Array Members: Minimum number of 1 item. Maximum number of 5 items.

Length Constraints: Minimum length of 1. Maximum length of 255.

Pattern: .+

Required: No

subnets

A list of one or more subnet IDs in your VPC.

Type: Array of strings

Array Members: Minimum number of 1 item. Maximum number of 16 items.

Length Constraints: Minimum length of 1. Maximum length of 255.

Pattern: .+

Required: Yes

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for Ruby V3
VPCConfigResponse

VPC configuration associated with your simulation job.

Contents

assignPublicIp

A boolean indicating if a public IP was assigned.

Type: Boolean

Required: No

securityGroups

A list of security group IDs associated with the simulation job.

Type: Array of strings

Array Members: Minimum number of 1 item. Maximum number of 5 items.

Length Constraints: Minimum length of 1. Maximum length of 255.

Pattern: .*

Required: No

subnets

A list of subnet IDs associated with the simulation job.

Type: Array of strings

Array Members: Minimum number of 1 item. Maximum number of 16 items.

Length Constraints: Minimum length of 1. Maximum length of 255.

Pattern: .*

Required: No

vpcId

The VPC ID associated with your simulation job.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 1024.

Pattern: .*

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
Common Errors

This section lists the errors common to the API actions of all AWS services. For errors specific to an API action for this service, see the topic for that API action.

**AccessDeniedException**
You do not have sufficient access to perform this action.
HTTP Status Code: 400

**IncompleteSignature**
The request signature does not conform to AWS standards.
HTTP Status Code: 400

**InternalFailure**
The request processing has failed because of an unknown error, exception or failure.
HTTP Status Code: 500

**InvalidAction**
The action or operation requested is invalid. Verify that the action is typed correctly.
HTTP Status Code: 400

**InvalidClientTokenId**
The X.509 certificate or AWS access key ID provided does not exist in our records.
HTTP Status Code: 403

**InvalidParameterCombination**
Parameters that must not be used together were used together.
HTTP Status Code: 400

**InvalidParameterValue**
An invalid or out-of-range value was supplied for the input parameter.
HTTP Status Code: 400

**InvalidQueryParameter**
The AWS query string is malformed or does not adhere to AWS standards.
HTTP Status Code: 400

**MalformedQueryString**
The query string contains a syntax error.
HTTP Status Code: 404

**MissingAction**
The request is missing an action or a required parameter.
HTTP Status Code: 400
MissingAuthenticationToken
The request must contain either a valid (registered) AWS access key ID or X.509 certificate.
HTTP Status Code: 403

MissingParameter
A required parameter for the specified action is not supplied.
HTTP Status Code: 400

OptInRequired
The AWS access key ID needs a subscription for the service.
HTTP Status Code: 403

RequestExpired
The request reached the service more than 15 minutes after the date stamp on the request or more than 15 minutes after the request expiration date (such as for pre-signed URLs), or the date stamp on the request is more than 15 minutes in the future.
HTTP Status Code: 400

ServiceUnavailable
The request has failed due to a temporary failure of the server.
HTTP Status Code: 503

ThrottlingException
The request was denied due to request throttling.
HTTP Status Code: 400

ValidationError
The input fails to satisfy the constraints specified by an AWS service.
HTTP Status Code: 400

Common Parameters

The following list contains the parameters that all actions use for signing Signature Version 4 requests with a query string. Any action-specific parameters are listed in the topic for that action. For more information about Signature Version 4, see Signature Version 4 Signing Process in the Amazon Web Services General Reference.

Action
The action to be performed.
Type: string
Required: Yes

Version
The API version that the request is written for, expressed in the format YYYY-MM-DD.
Type: string
**Required: Yes**

**X-Amz-Algorithm**

The hash algorithm that you used to create the request signature.

Condition: Specify this parameter when you include authentication information in a query string instead of in the HTTP authorization header.

Type: string

Valid Values: AWS4-HMAC-SHA256

**Required: Conditional**

**X-Amz-Credential**

The credential scope value, which is a string that includes your access key, the date, the region you are targeting, the service you are requesting, and a termination string ("aws4_request"). The value is expressed in the following format: access_key/YYYYMMDD/region/service/aws4_request.

For more information, see Task 2: Create a String to Sign for Signature Version 4 in the Amazon Web Services General Reference.

Condition: Specify this parameter when you include authentication information in a query string instead of in the HTTP authorization header.

Type: string

**Required: Conditional**

**X-Amz-Date**

The date that is used to create the signature. The format must be ISO 8601 basic format (YYYYMMDD'T'HHMMSS'Z'). For example, the following date time is a valid X-Amz-Date value: 20120325T120000Z.

Condition: X-Amz-Date is optional for all requests; it can be used to override the date used for signing requests. If the Date header is specified in the ISO 8601 basic format, X-Amz-Date is not required. When X-Amz-Date is used, it always overrides the value of the Date header. For more information, see Handling Dates in Signature Version 4 in the Amazon Web Services General Reference.

Type: string

**Required: Conditional**

**X-Amz-Security-Token**

The temporary security token that was obtained through a call to AWS Security Token Service (AWS STS). For a list of services that support temporary security credentials from AWS Security Token Service, go to AWS Services That Work with IAM in the IAM User Guide.

Condition: If you're using temporary security credentials from the AWS Security Token Service, you must include the security token.

Type: string

**Required: Conditional**

**X-Amz-Signature**

Specifies the hex-encoded signature that was calculated from the string to sign and the derived signing key.
Condition: Specify this parameter when you include authentication information in a query string instead of in the HTTP authorization header.

Type: string

Required: Conditional

**X-Amz-SignedHeaders**

Specifies all the HTTP headers that were included as part of the canonical request. For more information about specifying signed headers, see Task 1: Create a Canonical Request For Signature Version 4 in the Amazon Web Services General Reference.

Condition: Specify this parameter when you include authentication information in a query string instead of in the HTTP authorization header.

Type: string

Required: Conditional
Limits and Supported Regions

For AWS RoboMaker service limits, see AWS RoboMaker Limits.

For information about requesting limit increases for AWS resources, see AWS Service Limits.

For a list of the AWS Regions supporting AWS RoboMaker, see AWS RoboMaker Regions.
Document History for AWS RoboMaker

The following table describes the documentation for this release of AWS RoboMaker.

<table>
<thead>
<tr>
<th>Change</th>
<th>Description</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>New service and guide</td>
<td>This is the initial release of AWS RoboMaker and the AWS RoboMaker Developer Guide.</td>
<td>11/07/2018</td>
</tr>
<tr>
<td>Support for tags</td>
<td>Added support for tags to many AWS RoboMaker resources.</td>
<td>1/24/2019</td>
</tr>
</tbody>
</table>
For the latest AWS terminology, see the AWS Glossary in the AWS General Reference.