AWS RoboMaker

Developer Guide

AWS RoboMaker: Developer Guide

Table of Contents

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. You can easily create hundreds of new worlds from templates you define using Simulation WorldForge 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. 4)** – 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. 7)** – 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. **Read Getting Started with Simulation WorldForge (p. 13)** – A tutorial to help you create a simulation world template and generate worlds to use in your simulation.
4. Depending on your needs, explore the following topics.
   - Learn about the Robot Operating System (ROS). In AWS RoboMaker, you build applications and simulations based on ROS. For more information, see http://wiki.ros.org/ROS/Tutorials.
   - 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. Select version 9.

Supported Software and Versions

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

Name	Versions Supported	Description
Robot Operating System (ROS)	Melodic (end of life May 2023)	Libraries and tools to help you create robot applications.
Robot Operating System 2 (ROS 2)	Foxy	Released June 5, 2020.
Colcon	Latest	Command line tool to bundle ROS robot and simulation applications.

Name	Versions Supported	Description
Gazebo	9, 11	Tool to simulate robots in an environment.
rviz	ROS rviz ROS2 rviz2	Tool to visualize sensor data and state information from ROS in 3D.
rqt	latest	Framework and plugins based on Qt for ROS GUI development.

ROS supports the Python and Ubuntu versions listed below.

Name	Version(s) Supported	Ubuntu Version(s) Supported
ROS Melodic	Python 2	Ubuntu Bionic
ROS2 Foxy	Python 3	Ubuntu Focal

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

AWS RoboMaker Feature	ROS version(s) Supported
Cloud Extensions	Melodic
Development Environment (Create)	Melodic, Foxy
Simulation Job	Melodic, Foxy
Fleet Management	Melodic, Foxy

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

- Melodic and Gazebo 9
- Foxy and Gazebo 11

Gazebo supports the following components:

Component	Version(s) Supported
Physics engine	ODE
Rendering engine	OGRE

Software Support Policy

AWS RoboMaker supports specific open-source software versions of ROS, Gazebo and other dependent components. AWS RoboMaker will deprecate open-source software versions that are approaching end of life (EOL). In general, EOL means software will no longer be supported, receive software updates or security updates from its maintainers.

AWS RoboMaker will notify you of upcoming deprecation(s) of EOL open-source software. If your version will be deprecated, you should migrate your AWS RoboMaker resources to a supported ROS distribution so that you continue to benefit from the latest security updates and performance upgrades.

If your version will be deprecated, you should migrate your resources to a supported ROS distribution. For more information about ROS distributions, see Distributions (ROS 1) and Distributions (ROS 2).

Support ended April 2021

Starting April 30, 2021, you will no longer be able to create new ROS Kinetic, Gazebo 7.1, ROS Dashing or Ubuntu 16.04 resources in AWS RoboMaker. However, any existing AWS RoboMaker resources will remain in your account. If you do not upgrade, functionality of your ROS Kinetic, Gazebo 7.1, ROS Dashing and Ubuntu 16.04 resources within AWS RoboMaker features will change or even break.

The following software suite combinations will be deprecated:

- ROS Kinetic, Gazebo 7.1, Ubuntu 16.04
- ROS Kinetic, Gazebo 9, Ubuntu 16.04
- ROS Dashing, Gazebo 9, Ubuntu 16.04

The deprecation affected the following areas:

- **AWS Cloud9 integrated development environments (IDEs)**
  - You will have access to all existing ROS Kinetic and ROS Dashing based IDEs. You can continue to work within the IDE. The successful execution of the build and bundle process is not guaranteed.
  - You will be unable to create new ROS Kinetic and ROS Dashing based IDEs.
- **Robot and simulation applications**
  - You will be unable to create new ROS Kinetic and ROS Dashing based robot applications.
  - You will be unable to create new simulation applications with ROS Kinetic with Gazebo 7.1, ROS Kinetic with Gazebo 9, or ROS Dashing with Gazebo 9.
  - You will be unable to create a new versions of existing robot or simulation applications using the deprecated ROS and Gazebo versions.
- **Simulation jobs and simulation batches**
  - You will not be able to create new simulation jobs with robot applications and simulation applications using Kinetic, Dashing or Gazebo 7.1.

    Simulation jobs launched prior to the deprecation date and whose duration extends past the deprecation date will continue to run successfully until completed. With a maximum simulation job duration of 14 days, these jobs can run for a maximum of 14 days post deprecation.
- **Deployment jobs**
  - You will be not be able to create a deployment job for Kinetic or Dashing based robot applications.
- **Sample applications and cloud extensions**
  - The cloud extensions will no longer be supported in ROS Kinetic and ROS Dashing based applications. While you can install the cloud extensions into ROS Kinetic and ROS Dashing workspaces, they may or may not work.
  - You can no longer select ROS Kinetic or ROS Dashing as the ROS distribution to launch a sample application. The sample applications may still be downloaded into existing ROS Kinetic and ROS Dashing IDEs, however, they are no longer supported and may break.

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

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.

Package	Version	Description	
Robot Operating System (ROS)	Melodic	Libraries and tools to help developers create robot applications.	
Robot Operating System 2 (ROS2)	Foxy	A newer version of ROS.	
Colcon	Latest	A command line tool for bundling ROS robot and simulation applications.	
Gazebo	9 11	Tool for simulating robots in complex environments.	
rviz	Melodic Foxy	Tool for visualizing sensor data and state information from ROS in 3D.	
rqt	Melodic Foxy	Framework and plugins based on Qt for ROS GUI development.	

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.

A robot application includes information about the *robot software suite* required by the application. The robot software suite specifies the name and the version of a supported ROS distribution. For example, `ROS2` and `Foxy`.

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. 69).

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.

A simulation application includes information about the *simulation software suite* and robot software suite required by the simulation application. The simulation software suite specifies the name and the version of a supported simulation software suite. For example, you can specify `Gazebo` version `9` or `RosbagPlay` for `Melodic`.

You can generate hundreds of unique worlds from world templates using Simulation WorldForge. Each world template defines the parameters like room types and furnishing to use when generating worlds. When your world template is complete, you generate worlds using a world generation job. You can use the worlds in your simulation job or export them to use in other applications.

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. If default tools are enabled, 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.

When you create a simulation job, you specify a robot application configuration and a simulation application configuration. These specify launch configuration, tools, upload configurations, and for simulation applications, the world configuration. For example, you can configure custom tools to interact with the running simulation.

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.

Application 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

This section walks through running your first robotic application using AWS RoboMaker simulation and creating your first worlds with Simulation WorldForge. You use robotic applications and sample simulation world templates provided by AWS RoboMaker.

Topics

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, and run your applications. 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.

- **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 the robot and simulation application.** Use ROS to launch the applications. Explore the running simulation using rqt, Gazebo and other tools.

Topics

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
- 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

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

1. Open https://portal.aws.amazon.com/billing/signup.
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*.

> **Note**
> We assume that you use administrator-level credentials for the exercises and procedures in this guide. If you choose to create and use another IAM user, grant that user minimum permissions.

2. Sign in to the AWS Management Console.

   To sign in to the AWS Management Console as a IAM user, you must use a special URL. For more information, see How Users Sign In to Your Account in the *IAM User Guide*.

For more information about IAM, see the following:

- AWS Identity and Access Management (IAM)
- Getting started
- IAM User Guide

Step 2: 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.

Topics

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:

1. Sign in to the AWS RoboMaker console at https://console.aws.amazon.com/robomaker/.
2. On the left, expand **Development**, choose **Development environments**, and then choose **Create development 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 (c4.xlarge**). 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. You then 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.

To build the robot and simulation applications

1. Sign in to the AWS RoboMaker console at https://console.aws.amazon.com/robomaker/.
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, on the **Welcome** page, choose **Clone Hello world sample**.
4. On the left, in the **Environment** tab, expand **aws-robomaker-sample-application-helloworld**, **robot_ws**. **src**, **hello_world_robot**, and then **nodes**. Select the file **rotate.py** 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 **Window**, and then choose **New Terminal**. In the terminal window, use the following commands to build the application:

```
$ sudo apt update
$ sudo apt install python3-vcstool -y
$ cd  aws-robomaker-sample-application-helloworld/robot_ws/
$ vcs import < .rosinstall
$ rosdep install --from-paths src --ignore-src -r -y
$ colcon build
```

Note
Note: When using the “sudo apt update” command, if you receive a GPG error follow these instructions: ROS GPG Key Error (p. 231).

7. Build the simulation application. In the terminal window, change to the simulation application workspace and then use colcon to build the application:

```
$ cd ../simulation_ws
$ vcs import < .rosinstall
$ rosdep install --from-paths src --ignore-src -r -y
$ colcon build
```

Step 3: Run Robot and Simulation Applications

In this section, you run the Hello World application in the AWS Cloud9 environment.

Topics

Launch Applications

Launch the Hello World application in the AWS Cloud9 environment.

1. In the menu, navigate to **Virtual Desktop** and select **Launch Virtual Desktop**. It may take a few moments to launch in a new browser tab.

2. Navigate back to the **HelloWorld** AWS Cloud9 development environment, and in the terminal change directories to the `robot_ws` folder.
3. Launch the robot application:

```
$ cd ../robot_ws
$ source install/setup.bash
$ roslaunch hello_world_robot rotate.launch
```

   After the robot application launches, you will see the following information continuously updated in the terminal:

```
[INFO] [1616095476.465313, 1350.786000]: Rotating robot: linear:
  x: 0.0
  y: 0.0
  z: 0.0
angular:
  x: 0.0
  y: 0.0
  z: -0.1
```

4. Open a terminal in the `simulation_ws` folder.
5. Launch the simulation application:

```
$ cd aws-robomaker-sample-application-helloworld/simulation_ws
$ source install/local_setup.bash
$ export DISPLAY=:0
$ roslaunch hello_world_simulation empty_world.launch gui:=true
```

Note
When you specify `gui:=true`, roslaunch launches the Gazebo server (gzserver) and Gazebo client (gzclient). You should specify `gui:=false` when running an AWS RoboMaker simulation job. You can start and configure Gazebo and other simulation tools when you create a simulation job. For more information, see Simulation Tools (p. 178). You can stop zombie Gazebo processs by using `killall gzserver` in any terminal.

View and Explore a Simulation

When the simulation is running, you can use the virtual desktop to launch Gazebo and verify that the robot is rotating counter-clockwise.

1. In the **HelloWorld** AWS Cloud9 development environment, navigate to the previously opened **Virtual Desktop**.
2. In the **Virtual Desktop**, in the **Gazebo** application, zoom in on the robot. It is rotating in place.

Step 4: Run a Simulation Job

Once you have a working simulation you can bundle it, upload to Amazon S3, and create a simulation job in AWS RoboMaker. 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.

Note
Before running a simulation job in AWS RoboMaker simulation, check the gui argument in the launch file. The launch file starts the Gazebo server, and setting gui:=true will also launch the Gazebo client that is used for viewing the simulation on the Ubuntu desktop. However, when running an AWS RoboMaker simulation job, the Gazebo server and client are started separately and you should ensure that your launch files do not set the gui argument to true. You can start and configure the Gazebo client in the AWS RoboMaker simulation service using the simulation tool configuration.

Step 5: Clean up

1. Once you have finished using the development environment, select the robot symbol from the menu and then open the AWS RoboMaker console at `https://console.aws.amazon.com/robomaker/`.
2. You can either leave it for use another time OR you can delete it by choosing Development environments from the RoboMaker menu, choosing your environment from the list (e.g.: Helloworld), and then selecting **Delete**.

Additional Sample Robots

This section contains more advanced robots and simulations.

Topics

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. 8). Then, in the AWS RoboMaker console, launch the Robot Monitoring sample application.

Topics

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

1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
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

1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
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.

Getting Started with Simulation WorldForge

In this section, you learn Simulation WorldForge concepts and how to create a simulation world template describing the parameters of your world including floor plan, rooms, and furnishings. You also learn how to use a world generation job to generate worlds and then use them in a simulation application.

Topics

Step 1: Complete the Prerequisites

Complete the following setup steps before starting the walkthrough. Setup includes signing up for an AWS account, configuring a user, and then running the Hello World sample application. You use the Hello World robot application and simulation application with the worlds you create in this walkthrough.

If you are already using AWS RoboMaker and have a robotic application that meets the criteria described in Using generated worlds in your simulation (p. 67), you can skip this step.

To complete the prerequisites

1. Follow the instructions in Step 1: Create an AWS Account and an Administrator (p. 8).
2. Run the Hello World sample application as described in Run the Hello World Sample Application (p. 13).

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

1. Sign in to the AWS RoboMaker console at https://console.aws.amazon.com/robomaker/.
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 2: Create a Simulation World Template

In this section, you create a simulation world template. A simulation world template contains parameters that Simulation WorldForge uses to generate random worlds. Simulation WorldForge provides sample templates to help you get started.

To create a simulation world template

1. Sign in to the AWS RoboMaker console at https://console.aws.amazon.com/robomaker/.
2. On the AWS RoboMaker console, expand **Simulation WorldForge** on the left and then choose **World templates**.
3. On the **World templates** page, choose **Create template**.
4. On the **Create a world template** page, under **Sample templates**, choose **Browse and select**.
5. On the **Templates** page, under **Small house**, choose **Start**.
6. On the **Template detail** page, choose **Save and exit** to save the simulation world template. You will use it in the next section to generate a simulation world.

Step 3: Generate a Simulation World from the Simulation World Template

In this section, you create a world generation job using the simulation world template you created. A world generation job generates the worlds using the specified template parameters. When you create the generation job, you specify the number of floor plans and interior variations per floor plan in the worlds.

To create a simulation world from a simulation world template

1. Sign in to the AWS RoboMaker console at https://console.aws.amazon.com/robomaker/.
2. On the AWS RoboMaker console, expand **Simulation WorldForge** on the left and then choose **World templates**.
3. On the **World templates** page, choose the sample template you created, and then choose **Generate worlds**.
4. On the **Generate worlds** page, specify `1` for **Number of floor plans** and **Interior variations per floor plan**. Simulation WorldForge generates 1 world, or the `number of floor plans * interior variations per floor plan`. You can generate up to 50 worlds.

5. Choose **Generate** to begin world generation.
6. On the **World generation job details** page, you can track world generation progress. Generated worlds appear under **Completed worlds**.

Step 4: Use the Generated World in the Hello World Simulation

In this section, you create a simulation job that runs the Hello World application with the new world you generated.

To use the generated world in the Hello World simulation

1. Sign in to the AWS RoboMaker console at https://console.aws.amazon.com/robomaker/.
2. On the AWS RoboMaker console, expand **Simulations** on the left and then choose **Simulation jobs**.
3. On the **Simulation jobs** page, choose the Hello World simulation job. It is the simulation job with a robot application named **AWSRoboMakerHelloWorld**.
4. On the **Simulation jobs detail** page, choose **Actions** and then choose **Clone**.
5. On the **Review and create simulation job** page, next to **Step 3: Specify simulation application**, choose **Edit**.
6. On the **Specify simulation application** page, change **Launch file** to `worldforge_world.launch`.
7. On the **Specify simulation application** page, expand **Import world from WorldForge**, and then choose **Browse worlds**.
8. On the **Choose worlds** page, under **World**, choose the world you previously created, and then choose **Choose world**.
9. On the **Specify simulation application** page, choose **Next**.
10. On the **Review and create simulation job** page, choose **Create**.
11. You can explore the simulation using simulation tools. For more information about simulation tools, see Simulation Tools (p. 178). For other ways to explore the simulation, see View and Explore a Simulation (p. 11).

Step 5: Clean Up Resources

To avoid extra charges, clean up the example by following the steps in Step 5: Clean up (p. 12).

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.

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

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.

1. Sign in to the AWS RoboMaker console at `https://console.aws.amazon.com/robomaker/`.
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.
6. Optional: Select a **Simulation software suite**. Each sample application is built for Gazebo 9 and 11.
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. 17)

Configuring Permissions

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

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` and `Robot monitoring`.

Replace `account#` with your account number. Replace `simulation-job-role-id` with the IAM role ID. For more information, see `iamRole` under CreateSimulationJob (p. 287).

```
{

    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "s3:ListBucket",
                "s3:GetObject",
                "s3:PutObject"
            ],
            "Resource": [
                "*"
            ],
            "Effect": "Allow"
        },
        {
            "Action": [
                "iam:PassRole"
            ],
            "Resource": "arn:aws:iam::account#:role/simulation-job-role-id",
            "Effect": "Allow"
        },
        {
            "Action": [
                "robomaker:*"
            ],
            "Resource": "*",
            "Effect": "Allow"
        },
        {
            "Action": [
                "ec2:AssociateRouteTable",
                "ec2:AttachInternetGateway",
                "ec2:CreateInternetGateway",
                "ec2:CreateSubnet",
                "ec2:CreateVpc",
                "ec2:CreateTags",
                "ec2:CreateRoute",
                "ec2:CreateRouteTable",
                "ec2:CreateSecurityGroup",
                "ec2:DeleteSubnet",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeSubnets",
                "ec2:DescribeVpcs"
            ],
            "Resource": "*",
            "Effect": "Allow"
        },
        {
            "Action": [
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:PutLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:account#:log-group:/aws/lambda/*:*"
            ],
            "Effect": "Allow"
        },
        {
            "Condition": {
                "StringEquals": {
                    "iam:AWSServiceName": [
                        "robomaker.amazonaws.com"
                    ]
                }
            },
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "*",
            "Effect": "Allow"
        },
        {
            "Action": [
                "cloudformation:DescribeStacks"
            ],
            "Resource": "arn:aws:cloudformation:*:account#:stack/*",
            "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.

If you're using colcon to build and bundle your applications, 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).

If you're using container images for your applications, you upload your images to Amazon ECR. Amazon ECR uses image digests to indicate the version of your application. AWS RoboMaker remembers the image digest for each version.

If you have the image uploaded to Amazon ECR and you haven't altered the image digest, you can access and use that version of your application.

You can create a maximum of 40 versions per application.

Topics

Application Versioning With Bundles

You can update the $LATEST version as you develop your application. When you select the $LATEST version, you get it from the Amazon S3 location that you specify.

For example, you can start a simulation job using the latest version of your robot application and simulation application. If make changes to the robot application at the Amazon S3 path [what does at the S3 path mean?] and restart the simulation job, you use the updated version of the robot application.

You can only make updates to the $LATEST version of an AWS RoboMaker application. When you update the application to the $LATEST version, AWS RoboMaker uses that version. For example, if you restart a simulation job, AWS RoboMaker uses the latest version of the applications that you use in the simulation job.

Application Versioning With Images

You can update the $LATEST version of your container image as you develop your application. When you select the $LATEST version, get it from the Amazon ECR location that you specify.

When you create an image, you can also apply tags to them. You can specify the value of the tag field as `"latest"` for the $LATEST version. These values are distinct from each other.

There are two ways that an image gets the `"latest"` tag:

- If you specified a tag with the value of `"latest"`.
- If you're pushing an image that doesn't have tags, Amazon ECR updates the image with the `"latest"` tag.

In AWS RoboMaker, when you specify a tag for an image, that image is always picked as the $LATEST version. For example, if you create a robot application with the image name `"myImage"`, the tag `"xyz"`, and the image digest `"123"`, the $LATEST version is `myImage:xyz` with the digest `"123"`.

The following are the scenarios when you want to add a tag:

- When you want to update the $LATEST version to use a new tag. For example, you can have the image `"myImage"`. You can update your image with the tag `"abc"`. The $LATEST version of the image points to `myImage:abc`.
- When you want to update the image and retag it. For example, you can make changes to an image that has the tag `"abc"`. You can use the tag `"xyz"` after you update it. The $LATEST version points to `myImage:xyz`.

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. 75).

For more information how to create a simulation application version, see Creating a Simulation Application Version (p. 80). 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. 76) and Updating a Simulation Application (p. 81).

Deleting an Application Version

When you no longer need an application version, delete it. For more information, see Deleting a Robot Application Version (p. 77) and Deleting a Simulation Application Version (p. 83).

Application Versioning for Bundles

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

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
#           #   ### rotate.launch              # robot entrypoint, specifies running system
#           ### package.xml                    # ROS package config
#           ### scripts
#           #   ### rotate.py                  # custom ROS node, loaded at launch
#           ### 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
                ### example.world          # world description
```

Topics

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) Melodic or ROS2.
- Colcon

To create a preconfigured robotics development environment in AWS Cloud9, see Creating a Development Environment (p. 71).

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.

```
#!/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
  rospy
  std_msgs
)

################################################################################
# Setup for python modules and scripts
################################################################################
catkin_python_setup()

################################################################################
# Declare ROS messages, services and actions
################################################################################

################################################################################
# Declare ROS dynamic reconfigure parameters
################################################################################

################################################################################
# Declare catkin specific configuration to be passed to dependent projects
################################################################################
catkin_package(
  CATKIN_DEPENDS
    rospy
    std_msgs
)

################################################################################
# Build
################################################################################
include_directories(
  include
  ${catkin_INCLUDE_DIRS}
)

################################################################################
# Install
################################################################################
# Add your custom nodes here.
catkin_install_python(PROGRAMS
  scripts/rotate.py
  DESTINATION ${CATKIN_PACKAGE_BIN_DESTINATION}
)

install(DIRECTORY launch
    DESTINATION ${CATKIN_PACKAGE_SHARE_DESTINATION}
)

################################################################################
# Test
################################################################################
```

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 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>
  <buildtool_depend>catkin</buildtool_depend>
  <depend>rospy</depend>
  <depend>std_msgs</depend>
  <depend>turtlebot3_msgs</depend>
  <exec_depend>message_runtime</exec_depend>
  <exec_depend>turtlebot3_bringup</exec_depend>
</package>
```

10. Create a `setup.py` file and then save it to the `robot_app` directory. It helps ROS find your Python nodes. To learn more abot ROS Python makefiles, see Writing a ROS Python Makefile.

```
## ! DO NOT MANUALLY INVOKE THIS setup.py, USE CATKIN INSTEAD
## See http://ros.org/doc/api/catkin/html/user_guide/setup_dot_py.html

from distutils.core import setup
from catkin_pkg.python_setup import generate_distutils_setup

# fetch values from package.xml
setup_args = generate_distutils_setup(
    packages=['robot_app'],
    package_dir={'': 'src'}
)

setup(**setup_args)
```

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,
          <arg name="x_pos" default="10.0"/>
          <arg name="y_pos" default="5.0"/>
     -->
  </include>
</launch>
```

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.

```
<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 parmaters 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 pkg="robot_state_publisher" type="robot_state_publisher"
 name="robot_state_publisher" output="screen"/>
  <node name="joint_state_publisher" pkg="joint_state_publisher"
 type="joint_state_publisher" />
</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 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>
    <!-- A global light source -->
    <include>
      <uri>model://sun</uri>
    </include>

    <!-- A ground plane -->
    <include>
      <uri>model://ground_plane</uri>
    </include>

    <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_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 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>
  <depend>gazebo_plugins</depend>
  <depend>turtlebot3_description</depend>
  <depend>turtlebot3_gazebo</depend>
  <depend>turtlebot3_simulations</depend>
  <exec_depend>gazebo</exec_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. 30).

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

   ```
   $ 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:

   ```
   $ source install/setup.sh
   $ roslaunch robot_app rotate.launch
   ```

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

   ```
   roslaunch robot_app rotate.launch
   ```

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

   ```
   $ 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:

   ```
   $ source install/setup.sh
   $ roslaunch simulation_app example.launch
   ```

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

   ```
   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

Use the following commands to install `colcon` and `colcon-bundle`:

```
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:

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

Using Colcon to Build and Bundle

If you are migrating from `catkin`, verify the following before you use colcon to build and bundle your robotic application:

- All of the folders you want to include in the application are defined. For example, the following code includes common ROS folders `launch` and `nodes` in the install directory created by `colcon`. You can replace replace `launch` and `nodes` with the folders used by your application. Make sure this block is defined in your `cMakeLists.txt` file:

  ```
  install(DIRECTORY launch nodes
    DESTINATION ${CATKIN_PACKAGE_SHARE_DESTINATION}
  )
  ```

  For more information on `CMakeList.txt` and `catkin`, see catkin / CMakeLists.txt on ROS.org.
- All of your application dependencies are installed through `rosdep` or `pip`. Make sure all of your dependencies are added to `package.xml` and/or an included `requirements.txt` file. If a dependency is not available in the public rosdep sources lists, you can create a custom sources list. To do this, follow the instructions on ROS.org. For more information about troubleshooting, see Troubleshooting Colcon Build and Bundle (p. 235).
- All of the arguments you are pathing through your launch file are defined as environment variables in the AWS RoboMaker simulation job. You can define environment variables using the console or the AWS RoboMaker SDK. For an example using an environment variable, see ??? (p. 163).

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

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

You can test your application prior to bundling and uploading with the following:

```
source install/setup.sh
roslaunch package_name launch_file
```

Using Images to Develop Your AWS RoboMaker Applications

You can use one or more container images to develop and run your simulation and robot applications. For information about images, see Docker basics for Amazon ECS. The images that you use must meet the requirements listed in Requirements for AWS RoboMaker Compatible Containers (p. 31).

You can use your own images with AWS RoboMaker if you use one of the development environments that we support. For more information, see Create a ROS Development Environment (p. 5).

There are multiple ways that you can use container images to develop your applications. To see examples of how to develop your applications, see Creating Images to Run the Hello World Sample Application (p. 47).

After you've used images to develop your applications, you can test them. To test whether your applications work, you can visualize them on your local Linux machine.

After you've tested that your simulation works, you can push your images to Amazon ECR and run simulation jobs to see how your robot would interact in a virtual environment.

Topics

Requirements for AWS RoboMaker Compatible Containers

You must meet a set of requirements to run a **Robomaker Compatible Container (container image)** and to start a simulation successfully. If you've met these requirements, and you're still having trouble running the simulation, see Troubleshooting Simulation Jobs (p. 227) and Troubleshooting Simulation WorldForge (p. 230).

Robot Operating System (ROS) Requirements

Your container image:

- **MUST** be Open Container Initiative (OCI) complaint.

- **MUST** be built for the X86_64 architecture. If it's built for a different architecture, AWS RoboMaker **WILL** fail the simulation with a `4XX` error code.
- **MUST** be less than or equal to 40 GB in size uncompressed. If your container image is greater than 40 GB uncompressed, AWS RoboMaker **WILL** fail the simulation with a `4XX` error code.
- **MUST** use a base image that is based on Linux. If you don't use a base image that is based on Linux, AWS RoboMaker **WILL** fail the simulation with a `4XX` error code.
- **MUST** use a Robot Operating System (ROS) development environment and operating system that are compatible with each other. The following are examples of compatible combinations of ROS development environments and operating systems:
  - Robot Operating System (ROS) Melodic – ubuntu:bionic
  - Robot Operating System (ROS) 2 Foxy – ubuntu:focal

  If you don't use a compatible combination of robotics framework and operating system, your simulation might show unexpected behavior. For more information on development environments, see Create a ROS Development Environment (p. 5).

The following are the binary requirements for your container image:

- It **MUST** have an ROS environment set up at `/opt/ros/`*`distro`*`/setup.bash`. AWS RoboMaker sources the `setup.bash` to run implicit health checks.
- It **MUST** have access to `ros/ros2topic`. If it doesn't have access to this directory, your simulation **WILL** fail.
- It **MUST** have `roslaunch` installed. AWS RoboMaker sources `/opt/ros/`*`distro`*`/setup.bash` to find `roslaunch`. If `roslaunch` isn't installed, AWS RoboMaker **WILL** fail your simulation with a `4XX` error code.
- An image running as a simulation application with Gazebo Suite **MUST** have `roslaunch` installed and have a Gazebo environment set up at `/usr/share/`*`gazebo-version`*`/setup.bash`. RoboMaker sources `/opt/ros/`*`distro`*`/setup.bash && /usr/share/`*`gazebo-version`*`/setup.bash`. AWS RoboMaker **WILL** fail the simulation if these binaries are not present with a `4XX` error code.
- If your container image is running a simulation application with Gazebo Suite, it **MUST** have access to *`gz-topic`*. If your container image doesn't have access to *`gz-topic`*, AWS RoboMaker **WILL** fail your simulation.
- If your container image is running a simulation application with Gazebo Suite, it **MUST** have the Gazebo environment set up at `/usr/share/`*`gazebo-version`*. AWS RoboMaker sources `/usr/share/`*`gazebo-version`*`/setup.bash` to run implicit health checks.

To support GUI streaming, we recommend installing and sourcing the following binaries:

- `devilspie`

We recommend that your container image use absolute paths for its executables. We also recommend that you test whether the executables inside the containers run correctly. Your simulation **WILL** fail if it can't find the path to your executables.

Simulation Runtime Requirements

Your container image can't use `VOLUME` in the Dockerfile. If `VOLUME` is in the Dockerfile, your simulation **WILL** fail with a `4XX` error code.

Your container image can't use `EXPOSE` in the Dockerfile. If `EXPOSE` is in the Dockerfile, AWS RoboMaker **WILL** fail your simulation with a `4XX` error code.

You can't specify `CMD` in your Dockerfile. If you do, AWS RoboMaker overwrites it with the package name and launch file. Instead, you can use the `command` parameter in the `launchConfig` of each simulation application or robot application within your `CreateSimulationJob` request to provide a list of launch commands. This will be set as `CMD` in the simulation job. An example `command` is `["/bin/bash", "-c", "sleep 365d"]`.

If you want to add tools to your simulation job, you **MUST** install `bash` to your container image. Your tools will be launched with `["/bin/bash", "-c", "<command>"]`.

You **MUST** give your container image permission to communicate over ROS with your robot application and your simulation application. Your container image communicates with your containers using the following robotics frameworks:

- ROS Master
- Gazebo Master
- ROS IP

You can't customize the `/etc/resolv.conf` file in your container. AWS RoboMaker overwrites the file with its own file.

If you're running your Dockerfile on AWS, you can't **MOUNT** the image. If you specify `Mount` in the Dockerfile, AWS RoboMaker **WILL** fail your simulation with a `4XX` error code.

Your container image can't use system calls that are blocked by the default Docker `seccomp` profile. For information on blocked system calls, see Seccomp security profiles.

To specify a user that runs an image, you can specify a `USER` keyword in the Dockerfile. If you don't specify a user, AWS RoboMaker uses the root user in the container.

In your container image, you can specify either the `USER` as either a name or a `UID:GID`. If your container image doesn't have a UID, it has a default value of `1000`.

Your container image can't store data in `/opt/amazon/robomaker` or in any of its subfolders. Only AWS RoboMaker can use that directory. Your simulation might not behave properly if you use that directory.

Your container image **MUST** have ROS master running inside the Robot Application Container for ROS Software Suite. If you don't have ROS master running, AWS RoboMaker **WILL** fail your simulation with a `4XX` error code.

If your container image doesn't run ROS, you can define your health checks for your container.

If your container image is running ROS, AWS RoboMaker runs the following health checks on both the Robot Application Container and Simulation Application container at ten second intervals.

HEALTH CHECK	Robot Application Container	Simulation Application Container with Gazebo Suite	Simulation Application Container with RosbagPlay Suite	FAILURE CONDITION
`/opt/ros/<distro>/setup.bash; \rostopic info /rosout OR /opt/ros/<distro>/setup.bash; ros2 topic info /rosout`	YES	YES	YES	Return non-0 error code

`/usr/share/<gazebo version>/setup.bash; gz topic list`	NO	YES	NO	Return non-0 error code

If your container image doesn't pass the preceding health checks, your simulation **WILL** fail.

The following runtime configurations are not supported.

	Docker Run Argument	Description
1	`-\-add-host`	Add a custom host-to-IP mapping (host:ip)
2	`-\-attach , -a`	Attach to STDIN, STDOUT or STDERR
3	`-\-blkio-weight`	Block IO (relative weight), between 10 and 1000, or 0 to disable (default 0)
4	`-\-blkio-weight-devi ce`	Block IO weight (relative device weight)
5	`-\-cap-add`	Add Linux capabilities
6	`-\-cap-drop`	Drop Linux capabilities
7	`-\-cgroup-parent`	Optional parent cgroup for the container
8	`-\-cgroupns`	API 1.41+ <https://docs.d ocker.com/engine/api/ v1.41/ >__Cgroup namespace to use (host|private) 'host': Run the container in the Docker host's cgroup namespace 'private': Run the container in its own private cgroup namespace '': Use the cgroup namespace as configured by the default-cgroupns-mode option on the daemon (default)
9	`-\-cidfile`	Write the container ID to the file
10	`-\-cpu-count`	CPU count (Windows only)
11	`-\-cpu-percent`	CPU percent (Windows only)
12	`-\-cpu-period`	Limit CPU CFS (Completely Fair Scheduler) period
13	`-\-cpu-quota`	Limit CPU CFS (Completely Fair Scheduler) quota
14	`-\-cpu-rt-period`	API 1.25+ <https://docs.d ocker.com/engine/api/ v1.25/

	Docker Run Argument	Description
		>__Limit CPU real-time period in microseconds
15	`-\-cpu-rt-runtime`	API 1.25+ <https://docs.d ocker.com/engine/api/ v1.25/ >__Limit CPU real-time runtime in microseconds
16	`-\-cpu-shares , -c`	CPU shares (relative weight)
17	`-\-cpus`	API 1.25+ <https://docs.d ocker.com/engine/api/ v1.25/ >__Number of CPUs
18	`-\-cpuset-cpus`	CPUs in which to allow execution (0-3, 0,1)
19	`-\-cpuset-mems`	MEMs in which to allow execution (0-3, 0,1)
20	`-\-detach , -d`	Run container in background and print container ID
21	`-\-detach-keys`	Override the key sequence for detaching a container
22	`-\-device`	Add a host device to the container
23	`-\-device-cgroup-rul e`	Add a rule to the cgroup allowed devices list
24	`-\-device-read-bps`	Limit read rate (bytes per second) from a device
25	`-\-device-read-iops `	Limit read rate (IO per second) from a device
26	`-\-device-write-bps `	Limit write rate (bytes per second) to a device
27	`-\-device-write-iops`	Limit write rate (IO per second) to a device
28	`-\-disable-content-t rust`	Skip image verification
29	`-\-dns`	Set custom DNS servers
30	`-\-dns-opt`	Set DNS options
31	`-\-dns-option`	Set DNS options
32	`-\-dns-search`	Set custom DNS search domains
33	`-\-domainname`	Container NIS domain name

	Docker Run Argument	Description
34	`-\-gpus`	API 1.40+ <https://docs.d ocker.com/engine/api/ v1.40/ >__GPU devices to add to the container ('all' to pass all GPUs)
35	`-\-group-add`	Add additional groups to join
36	`-\-health-cmd`	Command to run to check health
37	`-\-health-interval`	Time between running the check (msm|h) (default 0s)
38	`-\-health-retries`	Consecutive failures needed to report unhealthy
39	`-\-health-start-peri od`	API 1.29+ <https://docs.d ocker.com/engine/api/ v1.29/ >__Start period for the container to initialize before starting health-retries countdown (msm| h) (default 0s)
40	`-\-health-timeout`	Maximum time to allow one check to run (msm|h) (default 0s)
41	`-\-help`	Print usage
42	`-\-hostname , -h`	Container host name
43	`-\-init`	API 1.25+ <https://docs.d ocker.com/engine/api/ v1.25/ >__Run an init inside the container that forwards signals and reaps processes
44	`-\-interactive , -i `	Keep STDIN open even if not attached
45	`-\-io-maxbandwidth`	Maximum IO bandwidth limit for the system drive (Windows only)
46	`-\-io-maxiops`	Maximum IOps limit for the system drive (Windows only)
47	`-\-ip`	IPv4 address (e.g., 172.30.100.104)
48	`-\-ip6`	IPv6 address (e.g., 2001:db8::33)
49	`-\-ipc`	IPC mode to use
50	`-\-isolation`	Container isolation technology
51	`-\-kernel-memory`	Kernel memory limit
52	`-\-label , -l`	Set meta data on a container

	Docker Run Argument	Description
53	`-\-label-file`	Read in a line delimited file of labels
54	`-\-link`	Add link to another container
55	`-\-link-local-ip`	Container IPv4/IPv6 link-local addresses
56	`-\-log-driver`	Logging driver for the container
57	`-\-log-opt`	Log driver options
58	`-\-mac-address`	Container MAC address (e.g., 92:d0:c6:0a:29:33)
59	`-\-memory , -m`	Memory limit
60	`-\-memory-reservation`	Memory soft limit
61	`-\-memory-swap`	Swap limit equal to memory plus swap: '-1' to enable unlimited swap
62	`-\-memory-swappiness`	Tune container memory swappiness (0 to 100)
63	`-\-name`	Assign a name to the container
64	`-\-net`	Connect a container to a network
65	`-\-net-alias`	Add network-scoped alias for the container
66	`-\-network`	Connect a container to a network
67	`-\-network-alias`	Add network-scoped alias for the container
68	`-\-no-healthcheck`	Disable any container-specified HEALTHCHECK
69	`-\-oom-kill-disable `	Disable OOM Killer
70	`-\-oom-score-adj`	Tune host's OOM preferences (-1000 to 1000)
71	`-\-pid`	PID namespace to use
72	`-\-pids-limit`	Tune container pids limit (set -1 for unlimited)
73	`-\-platform`	API 1.32+ <https://docs.d ocker.com/engine/api/ v1.32/ >__Set platform if server is multi-platform capable

	Docker Run Argument	Description
74	`-\-privileged`	Give extended privileges to this container
75	`-\-publish , -p`	Publish a container's port(s) to the host
76	`-\-publish-all , -P `	Publish all exposed ports to random ports
77	`-\-pull`	Pull image before running ("always" " never")
78	`-\-read-only`	Mount the container's root filesystem as read only
79	`-\-restart`	Restart policy to apply when a container exits
80	`-\-rm`	Automatically remove the container when it exits
81	`-\-runtime`	Runtime to use for this container
82	`-\-security-opt`	Security Options
83	`-\-shm-size`	Size of /dev/shm
84	`-\-sig-proxy`	Proxy received signals to the process
85	`-\-stop-timeout`	API 1.25+ <https://docs.d ocker.com/engine/api/ v1.25/ >__Timeout (in seconds) to stop a container
86	`-\-storage-opt`	Storage driver options for the container
87	`-\-sysctl`	Sysctl options
88	`-\-tmpfs`	Mount a tmpfs directory
89	`-\-tty , -t`	Allocate a pseudo-TTY
90	`-\-ulimit`	Ulimit options
91	`-\-userns`	User namespace to use
92	`-\-uts`	UTS namespace to use
93	`-\-volume , -v`	Bind mount a volume
94	`-\-volume-driver`	Optional volume driver for the container
95	`-\-volumes-from`	Mount volumes from the specified container(s)

If you run a simulation job with the preceding runtime configurations, AWS RoboMaker **WILL** fail your simulation with a `4XX` error code.

Metadata Requirements

Your container image:

- **MUST** be Open Container Initiative (OCI) complaint.
- **MUST** be built for the X86_64 architecture. If it's built for a different architecture, AWS RoboMaker **WILL** fail the simulation with a `4XX` error code.
- **MUST** be less than or equal to 40 GB in size uncompressed. If your container image is greater than 40 GB uncompressed, AWS RoboMaker **WILL** fail the simulation with a `4XX` error code.
- **MUST** use a base image that is based on Linux. If you don't use a base image that is based on Linux, AWS RoboMaker **WILL** fail the simulation with a `4XX` error code.
- **MUST** use a development environment and operating system that are compatible with each other. The following are examples of compatible combinations of development environments and operating systems:
  - Robot Operating System (ROS) Melodic – ubuntu:bionic
  - Robot Operating System (ROS) 2 Foxy – ubuntu:focal

  If you don't use a compatible combination of robotics framework and operating system, your simulation might show unexpected behavior. For more information on development environments, see Create a ROS Development Environment (p. 5).

Binary Requirements

The following are the binary requirements for your container image:

- It **MUST** have an ROS environment set up at `/opt/ros/`*`distro`*`/setup.bash`. AWS RoboMaker sources the `setup.bash` to run implicit health checks.
- It **MUST** have access to `ros/ros2topic`. If it doesn't have access to this directory, your simulation **WILL** fail.
- It **MUST** have `roslaunch` installed. AWS RoboMaker sources `/opt/ros/`*`distro`*`/setup.bash` to find `roslaunch`. If `roslaunch`, AWS RoboMaker **WILL** fail your simulation with a `4XX` error code.
- If your container image is running a simulation application with Gazebo Suite, it **MUST** have access to *`gz-topic`*. If your container image doesn't have access to *`gz-topic`*, AWS RoboMaker **WILL** fail your simulation.
- If your container image is running a simulation application with Gazebo Suite, it **MUST** have the Gazebo environment set up at `/usr/share/`*`gazebo-version`*. AWS RoboMaker sources `/usr/share/`*`gazebo-version`*`/setup.bash` to run implicit health checks.

To support GUI streaming, we recommend installing and sourcing the following binaries:

- `devilspie`

We recommend that your container image use absolute paths for its executables. We also recommend that the executable inside the container runs correctly. Your simulation **WILL** fail if it can't find the path to your executables.

GPU Requirements

Your container image:

- **MUST** have glvnd installed if using OpenGL in your applications.

- **MUST** have NVIDIA CUDA 11.2 or lower if using CUDA in your applications.
- **MUST** have OpenGL version 4.6 or lower if using OpenGL in your applications.
- **MUST** have Vulkan version 1.2 or lower if using Vulkan APIs in your applications.
- **MUST** have OpenCL version 1.2 or lower if using OpenCL in your applications.

Note
AWS RoboMaker supports Vulkan only for offscreen rendering and is not operational in GUI displays. So, streamUI should be set to `false` if using Vulkan.

For detailed instructions on how GPU images can be created, see Creating Images to Run GPU Applications (p. 46).

Dockerfile and Environment Variable Requirements

A container image **MUST** provide an entrypoint script for sourcing. The entrypoint script **MUST** have `exec "${@:1}"` as the last line so that AWS RoboMaker can run the entrypoint script. Running the entrypoint script gives you the ability to use the `roslaunch` *`package-name`* command. *`launch-file`* command to run the containers.

Your container image can't use `VOLUME` in the Dockerfile. If `VOLUME` is in the Dockerfile, your simulation **WILL** fail with a `4XX` error code.

The `EXPOSE` keyword in your Dockerfile is ignored by AWS RoboMaker. Any ports exposed by the `EXPOSE` keyword are not automatically exposed by the system. If you would like to expose ports on your simulation, you can use AWS RoboMaker port forwarding configuration.

AWS RoboMaker uses the following environment variables. If you run your simulation on AWS, AWS RoboMaker overwrites any value that you specify for these environment variables:

- `ROBOMAKER*`
- `ROS_MASTER_URI`
- `ROS_IP`
- `GAZEBO_MASTER_URI`
- `DCV_VIRTUAL_SESSION`
- `XDG_SESSION_ID`
- `DCV_SESSION_ID`
- `XDG_SESSION_TYPE`
- `XDG_RUNTIME_DIR`
- `SHLVL`
- `DISPLAY`
- `XAUTHORITY`

You can't specify `CMD` in your Dockerfile. If you do, AWS RoboMaker overwrites with the package name and launch file.

Network, Mount, Security and User Requirements

You must give your container image permission to communicate over ROS with your robot application and your simulation application. Your container image communicates with your containers using the following robotics frameworks:

- ROS Master
- Gazebo Master
- ROS IP

You can't customize the `/etc/resolv.conf` file in your container. AWS RoboMaker overwrites the file with its own file.

If you're running your Dockerfile on AWS, you can't **MOUNT** the image. If you specify `Mount` in the Dockerfile, AWS RoboMaker **WILL** fail your simulation with a `4XX` error code.

Your container image can't use system calls that are blocked by the default Docker `seccomp` profile. For information on blocked system calls, see Seccomp security profiles.

To specify a user that runs an image, you can specify a `USER` keyword in the Dockerfile. If you don't specify a user, AWS RoboMaker uses the root user in the container.

In your container image, you can specify the `USER` as either a name or a `UID:GID`. If your container image doesn't have a UID, it has a default value of `1000`.

Other Requirements

Your container image can't store data in `/opt/amazon/robomaker` or in any of its subfolders. Only AWS RoboMaker can use that directory. Your simulation might not behave properly if you use that directory.

Your container image **MUST** have `gzserver` running inside the Simulation Application Container for Gazebo Software Suite. If it doesn't have `gzserver` running, AWS RoboMaker **WILL** fail your simulation with a `4XX` error code.

Your container image **MUST** have ROS master running inside the Robot Application Container for ROS Software Suite. If you don't have ROS master running, AWS RoboMaker **WILL** fail your simulation with a `4XX` error code.

AWS RoboMaker runs the following health checks on both the Robot Application Container and Simulation Application container at ten second intervals.

HEALTH CHECK	Robot Application Container	Simulation Application Container with Gazebo Suite	Simulation Application Container with RosbagPlay Suite	FAILURE CONDITION
`/opt/ros/<distro>/setup.bash; \rostopic info /rosout OR /opt/ros/<distro>/setup.bash; ros2 topic info /rosout`	YES	YES	YES	Return non-0 error code
`/usr/share/<gazebo version>/setup.bash; gz topic list`	NO	YES	NO	Return non-0 error code

If your container image doesn't pass the preceding health checks, your simulation **WILL** fail.

The following runtime configurations are not supported.

	Docker Run Argument	Description
1	`--add-host`	Add a custom host-to-IP mapping (host:ip)
2	`--attach , -a`	Attach to STDIN, STDOUT or STDERR
3	`--blkio-weight`	Block IO (relative weight), between 10 and 1000, or 0 to disable (default 0)
4	`--blkio-weight-devi ce`	Block IO weight (relative device weight)
5	`--cap-add`	Add Linux capabilities
6	`--cap-drop`	Drop Linux capabilities
7	`--cgroup-parent`	Optional parent cgroup for the container
8	`--cgroupns`	API 1.41+ <https://docs.d ocker.com/engine/api/ v1.41/ >__Cgroup namespace to use (host|private) 'host': Run the container in the Docker host's cgroup namespace 'private': Run the container in its own private cgroup namespace '': Use the cgroup namespace as configured by the default-cgroupns-mode option on the daemon (default)
9	`--cidfile`	Write the container ID to the file
10	`--cpu-count`	CPU count (Windows only)
11	`--cpu-percent`	CPU percent (Windows only)
12	`--cpu-period`	Limit CPU CFS (Completely Fair Scheduler) period
13	`--cpu-quota`	Limit CPU CFS (Completely Fair Scheduler) quota
14	`--cpu-rt-period`	API 1.25+ <https://docs.d ocker.com/engine/api/ v1.25/ >__Limit CPU real-time period in microseconds
15	`--cpu-rt-runtime`	API 1.25+ <https://docs.d ocker.com/engine/api/ v1.25/ >__Limit CPU real-time runtime in microseconds
16	`--cpu-shares , -c`	CPU shares (relative weight)

	Docker Run Argument	Description
17	`--cpus`	API 1.25+ <https://docs.d ocker.com/engine/api/ v1.25/ >__Number of CPUs
18	`--cpuset-cpus`	CPUs in which to allow execution (0-3, 0,1)
19	`--cpuset-mems`	MEMs in which to allow execution (0-3, 0,1)
20	`--detach , -d`	Run container in background and print container ID
21	`--detach-keys`	Override the key sequence for detaching a container
22	`--device`	Add a host device to the container
23	`--device-cgroup-rul e`	Add a rule to the cgroup allowed devices list
24	`--device-read-bps`	Limit read rate (bytes per second) from a device
25	`--device-read-iops `	Limit read rate (IO per second) from a device
26	`--device-write-bps `	Limit write rate (bytes per second) to a device
27	`--device-write-iops`	Limit write rate (IO per second) to a device
28	`--disable-content-t rust`	Skip image verification
29	`--dns`	Set custom DNS servers
30	`--dns-opt`	Set DNS options
31	`--dns-option`	Set DNS options
32	`--dns-search`	Set custom DNS search domains
33	`--domainname`	Container NIS domain name
34	`--gpus`	API 1.40+ <https://docs.d ocker.com/engine/api/ v1.40/ >__GPU devices to add to the container ('all' to pass all GPUs)
35	`--group-add`	Add additional groups to join
36	`--health-cmd`	Run to check health
37	`--health-interval`	Time between running the check (msm|h) (default 0s)

	Docker Run Argument	Description
38	`--health-retries`	Consecutive failures needed to report unhealthy
39	`--health-start-peri od`	API 1.29+ <https://docs.d ocker.com/engine/api/ v1.29/ >__Start period for the container to initialize before starting health-retries countdown (msm|h) (default 0s)
40	`--health-timeout`	Maximum time to allow one check to run (msm|h) (default 0s)
41	`--help`	Print usage
42	`--hostname , -h`	Container host name
43	`--init`	API 1.25+ <https://docs.d ocker.com/engine/api/ v1.25/ >__Run an init inside the container that forwards signals and reaps processes
44	`--interactive , -i `	Keep STDIN open even if not attached
45	`--io-maxbandwidth`	Maximum IO bandwidth limit for the system drive (Windows only)
46	`--io-maxiops`	Maximum IOps limit for the system drive (Windows only)
47	`--ip`	IPv4 address (e.g., 172.30.100.104)
48	`--ip6`	IPv6 address (e.g., 2001:db8::33)
49	`--ipc`	IPC mode to use
50	`--isolation`	Container isolation technology
51	`--kernel-memory`	Kernel memory limit
52	`--label , -l`	Set meta data on a container
53	`--label-file`	Read in a line delimited file of labels
54	`--link`	Add link to another container
55	`--link-local-ip`	Container IPv4/IPv6 link-local addresses
56	`--log-driver`	Logging driver for the container
57	`--log-opt`	Log driver options

	Docker Run Argument	Description
58	`--mac-address`	Container MAC address (e.g., 92:d0:c6:0a:29:33)
59	`--memory , -m`	Memory limit
60	`--memory-reservation`	Memory soft limit
61	`--memory-swap`	Swap limit equal to memory plus swap: '-1' to enable unlimited swap
62	`--memory-swappiness`	Tune container memory swappiness (0 to 100)
63	`--name`	Assign a name to the container
64	`--net`	Connect a container to a network
65	`--net-alias`	Add network-scoped alias for the container
66	`--network`	Connect a container to a network
67	`--network-alias`	Add network-scoped alias for the container
68	`--no-healthcheck`	Disable any container-specified HEALTHCHECK
69	`--oom-kill-disable `	Disable OOM Killer
70	`--oom-score-adj`	Tune host's OOM preferences (-1000 to 1000)
71	`--pid`	PID namespace to use
72	`--pids-limit`	Tune container pids limit (set -1 for unlimited)
73	`--platform`	API 1.32+ <https://docs.d ocker.com/engine/api/ v1.32/ >__Set platform if server is multi-platform capable
74	`--privileged`	Give extended privileges to this container
75	`--publish , -p`	Publish a container's port(s) to the host
76	`--publish-all , -P `	Publish all exposed ports to random ports
77	`--pull`	Pull image before running ("always" " never")

	Docker Run Argument	Description
78	`--read-only`	Mount the container's root filesystem as read only
79	`--restart`	Restart policy to apply when a container exits
80	`--rm`	Automatically remove the container when it exits
81	`--runtime`	Runtime to use for this container
82	`--security-opt`	Security Options
83	`--shm-size`	Size of /dev/shm
84	`--sig-proxy`	Proxy received signals to the process
85	`--stop-timeout`	API 1.25+ <https://docs.docker.com/engine/api/ v1.25/ >__Timeout (in seconds) to stop a container
86	`--storage-opt`	Storage driver options for the container
87	`--sysctl`	Sysctl options
88	`--tmpfs`	Mount a tmpfs directory
89	`--tty , -t`	Allocate a pseudo-TTY
90	`--ulimit`	Ulimit options
91	`--userns`	User namespace to use
92	`--uts`	UTS namespace to use
93	`--volume , -v`	Bind mount a volume
94	`--volume-driver`	Optional volume driver for the container
95	`--volumes-from`	Mount volumes from the specified container(s)

If you run a simulation job with the preceding runtime configurations, AWS RoboMaker **WILL** fail your simulation with a `4XX` error code.

Creating Images to Run GPU Applications

AWS RoboMaker GPU simulation jobs support CUDA, OpenGL, OpenCL and Vulkan API access. Therefore, the application using these APIs should have the corresponding drivers installed in their images.

To use DCV display with GPU rendering, you must install `nice-dcv-gl`. Note that X0 is the system's Xorg process that talks to the GPU. X1 and X2 are instead XDCV processes. When you start an OpenGL application on X1 or X2, `nice-dcv-gl` takes care of redirecting the calls and performing the rendering on X0, where the GPU is available.

To install `nice-dcv-gl`, download the archive, extract it, and install the `nice-dcv-gl` package following the DCV public documentation. See Install the NICE DCV Server on Linux.

The following example demonstrates Dockerfile installing nice-dcv-gl_2021.2 on a ubuntu18.04 base image.

```
FROM nvidia/opengl:1.0-glvnd-runtime-ubuntu20.04

ENV DEBIAN_FRONTEND="noninteractive"

RUN apt-get update && apt-get install -y --no-install-recommends \
        ca-certificates \
        gnupg2 \
        wget

RUN wget https://d1uj6qtbmh3dt5.cloudfront.net/NICE-GPG-KEY && gpg --import NICE-GPG-KEY &&
 \
        wget https://d1uj6qtbmh3dt5.cloudfront.net/2021.2/Servers/nice-dcv-2021.2-11048-
ubuntu1804-x86_64.tgz && \
        tar xvzf nice-dcv-2021.2-11048-ubuntu1804-x86_64.tgz && \
        cd nice-dcv-2021.2-11048-ubuntu1804-x86_64 && \
        apt install -y ./nice-dcv-gl_2021.2.944-1_amd64.ubuntu1804.deb
```

For detailed instructions on building a GPU application, see Running a GPU Sample Application with ROS2 Foxy and Gazebo 11 (p. 56).

Creating Images to Run the Hello World Sample Application

You can use the Hello World sample application that we provide to help you understand how to create and run your simulation and robot applications. In the following sections, we show you how to create and run images for the following development environments:

- ROS Melodic and Gazebo 9
- ROS 2 Foxy and Gazebo 11

ROS is the robot operating system used for your robot applications. Gazebo is the operating system for your simulation applications. AWS RoboMaker uses both software suites to use container images and provide validation checks. For more information on ROS and Gazebo, see Create a ROS Development Environment (p. 5).

The tutorials guide you through using AWS RoboMaker container images to set up the Hello World robot and simulation applications. The Hello World applications are example applications that help you understand how to work with AWS RoboMaker. For more information about the Hello World application, see Getting Started with Simulation WorldForge (p. 13).

For each tutorial, you create images for both your robot and simulation applications. You can run the images locally to test how they work. If your simulations work properly, you can push them to Amazon ECR and run simulation jobs in the cloud. For more information on simulation jobs, see Running Simulation Jobs (p. 151).

Running a Sample Application with ROS 2 Foxy and Gazebo 11

The following tutorial shows you how to use container images to develop with ROS 2 Foxy and Gazebo 11, by creating and running the Hello World robot application and simulation application. You can get the sample application to work by running the commands described in this document.

For this tutorial, we create and use three container images. The following shows the directory structure that we use for this example application.

```
### HelloWorldSampleAppROS2FoxyGazebo11 // Base Image
#   ### Dockerfile
### HelloWorldSampleAppROS2FoxyGazebo11RobotApp // Image for Robot App
#   ### Dockerfile
#   ### robot-entrypoint.sh
### HelloWorldSampleAppROS2FoxyGazebo11SimApp // Image for Simulation App
#   ### Dockerfile
#   ### simulation-entrypoint.sh
```

Each Dockerfile has the instructions needed to build each image;

- The Dockerfile for the base image has the commands to set up ROS and Gazebo.
- The Dockerfile for the robot application has the commands to set up the Hello World robot application.
- The Dockerfile for the simulation application has the commands to set up the Hello World simulation application.

Both the robot application and the simulation application have an entrypoint script. These scripts source the environments for their respective applications. They set up the path for you to run commands to start your robot and simulation applications.

Creating a Base Image

To create a base image, you save the commands to create your environment in a Dockerfile. You then build the Dockerfile.

- Save the following commands in a Dockerfile.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: MIT-0
FROM ros:foxy

ENV DEBIAN_FRONTEND noninteractive

RUN apt-get clean
RUN apt-get update && apt-get install -y \
    lsb  \
    unzip \
    wget \
    curl \
    sudo \
    python3-vcstool \
    python3-rosinstall \
    python3-colcon-common-extensions \
    ros-foxy-rviz2 \
    ros-foxy-rqt \
    ros-foxy-rqt-common-plugins \
    devilspie \
    xfce4-terminal

RUN wget https://packages.osrfoundation.org/gazebo.key -O - | sudo apt-key add -; \
    sh -c 'echo "deb http://packages.osrfoundation.org/gazebo/ubuntu-stable
 `lsb_release -cs` main" > /etc/apt/sources.list.d/gazebo-stable.list'
```

```
RUN apt-get update && apt-get install -y gazebo11

ENV QT_X11_NO_MITSHM=1

ARG USERNAME=robomaker
RUN groupadd $USERNAME
RUN useradd -ms /bin/bash -g $USERNAME $USERNAME
RUN sh -c 'echo "$USERNAME ALL=(root) NOPASSWD:ALL" >> /etc/sudoers'
USER $USERNAME

RUN sh -c 'cd /home/$USERNAME'

# Download and build our Robot and Simulation application
RUN sh -c 'mkdir -p /home/robomaker/workspace'
RUN sh -c 'cd /home/robomaker/workspace && wget https://github.com/aws-robotics/aws-
robomaker-sample-application-helloworld/archive/3527834.zip && unzip 3527834.zip && mv
 aws-robomaker-sample-application-helloworld-3527834771373beff0ed3630c13479567db4149e
 aws-robomaker-sample-application-helloworld-ros2'
RUN sh -c 'cd /home/robomaker/workspace/aws-robomaker-sample-application-helloworld-
ros2'

RUN sudo rosdep fix-permissions
RUN rosdep update
```

After you've created the Dockerfile, build it using the following commands on your terminal.

```
cd ../HelloWorldSampleAppROS2FoxyGazebo11
docker build -t helloworldsampleappros2foxygazebo11:latest .
```

Building the base image installs ROS 2 Foxy and Gazebo 11. You need both libraries installed to successfully run your applications.

Creating an Image for the Robot Application

After you've created the base image, you can create the image for your robot application. You save the following script in a Dockerfile and build it. This script downloads the Hello World robot application and sets it up.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: MIT-0
FROM helloworldsampleappros2foxygazebo11:latest

# Build the Robot application
RUN cd /home/robomaker/workspace/aws-robomaker-sample-application-helloworld-ros2/robot_ws
 && \
 /bin/bash -c "source /opt/ros/foxy/setup.bash && vcs import < .rosinstall && rosdep
 install --rosdistro foxy --from-paths src --ignore-src -r -y && colcon build"


COPY robot-entrypoint.sh /home/robomaker/robot-entrypoint.sh
RUN sh -c 'sudo chmod +x /home/robomaker/robot-entrypoint.sh'
RUN sh -c 'sudo chown robomaker:robomaker /home/robomaker/robot-entrypoint.sh'

CMD ros2 launch hello_world_robot rotate.launch.py
ENTRYPOINT [ "/home/robomaker/robot-entrypoint.sh" ]
```

The following command creates the image for the robot application from the Dockerfile.

```
cd HelloWorldSampleAppROS2FoxyGazebo11RobotApp/HelloWorldSampleAppROS2FoxyGazebo11RobotApp
docker build -t helloworldsampleappros2foxygazebo11robotapp:latest .
```

The following are the contents of the script that you can save as `robot-entrypoint.sh`. This script sources the environment for the robot application.

```
#!/bin/bash
cd /home/robomaker/workspace/aws-robomaker-sample-application-helloworld-ros2/robot_ws
source /opt/ros/foxy/setup.bash
source /usr/share/gazebo-11/setup.sh
source ./install/setup.sh
printenv

exec "${@:1}"
```

Creating an Image for the Simulation Application

After you've created the base image and the image for the robot application, you can create the image for your simulation application. You save the following script in a Dockerfile and build it. This script downloads the Hello World robot application and sets it up.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: MIT-0
FROM helloworldsampleappros2foxygazebo11:latest

# Build the Simulation application
RUN cd /home/robomaker/workspace/aws-robomaker-sample-application-helloworld-ros2/
simulation_ws && \
 /bin/bash -c "source /opt/ros/foxy/setup.bash && vcs import < .rosinstall && rosdep
 install --rosdistro foxy --from-paths src --ignore-src -r -y && colcon build"


COPY simulation-entrypoint.sh /home/robomaker/simulation-entrypoint.sh

RUN sh -c 'sudo chmod +x /home/robomaker/simulation-entrypoint.sh'
RUN sh -c 'sudo chown robomaker:robomaker /home/robomaker/simulation-entrypoint.sh'

CMD ros2 launch hello_world_simulation empty_world.launch.py
ENTRYPOINT [ "/home/robomaker/simulation-entrypoint.sh" ]
```

The following command creates the image.

```
cd HelloWorldSampleAppROS2FoxyGazebo11SimApp/HelloWorldSampleAppROS2FoxyGazebo11SimApp
docker build -t helloworldsampleappros2foxygazebo11simapp:latest .
```

The following are the contents of the script that you can save as `simulation-entrypoint.sh`. This script sources the environment for the simulation application.

```
#!/bin/bash
```

```
cd /home/robomaker/workspace/aws-robomaker-sample-application-helloworld-ros2/simulation_ws
source /opt/ros/foxy/setup.bash
source /usr/share/gazebo-11/setup.sh
source ./install/setup.sh
printenv

exec "${@:1}"
```

Running the Application and Pushing It to Amazon ECR

After you've created your images, make sure that they run properly in your local Linux environment. After you've checked that your image runs, you can push your Docker image to Amazon ECR and create a simulation job.

The following commands give you the ability to run the Hello World application in your local Linux environment.

```
docker run -it -e DISPLAY -v /tmp/.X11-unix/:/tmp/.X11-unix/ --name robot_app \
-u robomaker -e ROBOMAKER_GAZEBO_MASTER_URI=http://localhost:5555 \
-e ROBOMAKER_ROS_MASTER_URI=http://localhost:11311 \
helloworldsampleappros2foxygazebo11robotapp:latest
```

```
docker run -it -e DISPLAY -v /tmp/.X11-unix/:/tmp/.X11-unix/ --name sim_app \
-u robomaker -e ROBOMAKER_GAZEBO_MASTER_URI=http://localhost:5555 \
-e ROBOMAKER_ROS_MASTER_URI=http://localhost:11311 \
helloworldsampleappros2foxygazebo11simapp:latest
```

When you run the robot application and simulation application containers, you can visualize the simulation using the Gazebo GUI tool. Use the following commands to:

1. Connect to your container running the simulation application.
2. Visualize your application by running the Gazebo Graphical User Interface (GUI).

```
# Enable access to X server to launch Gazebo from docker container
$ xhost +

# Check that the robot_app and sim_app containers are running. The command should list both
 containers
$ docker container ls

# Connect to the sim app container
$ docker exec -it sim_app bash

# Launch Gazebo from within the container
$ /home/robomaker/simulation-entrypoint.sh ros2 launch gazebo_ros gzclient.launch.py
```

You can add tags to your images. The following commands give you the ability to tag your images.

```
docker tag helloworldsampleappros2foxygazebo11robotapp:latest accountID.dkr.ecr.us-
west-2.amazonaws.com/helloworldsampleappros2foxygazebo11robotapp:latest
```

```
docker tag helloworldsampleappros2foxygazebo11simapp:latest accountID.dkr.ecr.us-
west-2.amazonaws.com/helloworldsampleappros2foxygazebo11simapp:latest
```

After you've verified that the application is working properly, you can push to Amazon ECR using the following commands.

```
aws ecr get-login-password --region us-west-2 | docker login --username AWS —password-
stdin accountID.dkr.ecr.us-west-2.amazonaws.com
docker push accountID.dkr.ecr.us-west-2.amazonaws.com/
helloworldsampleappros2foxygazebo11robotapp:latest
docker push accountID.dkr.ecr.us-west-2.amazonaws.com/
helloworldsampleappros2foxygazebo11simapp:latest
```

You can then run a simulation job on the image. For more information on simulation jobs, see Running Simulation Jobs (p. 151).

Running a Sample Application with ROS Melodic and Gazebo 9

The following tutorial shows you how to use container images to develop with ROS and Gazebo 9 by creating and running the Hello World robot application and simulation application. You can get the sample application to work by running the commands described in this document.

For this tutorial, we create and use three container images. The following shows the directory structure that we use for this example application.

```
### HelloWorldSampleAppROSMelodicGazebo9 // Base Image
#   ### Dockerfile
### HelloWorldSampleAppROSMelodicGazebo9RobotApp // Image for Robot App
#   ### Dockerfile
#   ### robot-entrypoint.sh
### HelloWorldSampleAppROSMelodicGazebo9SimApp // Image for Simulation App
#   ### Dockerfile
#   ### simulation-entrypoint.sh
```

Each Dockerfile has the instructions needed to build each image.

- The Dockerfile for the base image has the commands to set up ROS and Gazebo.
- The Dockerfile for the robot application has the commands to set up the Hello World robot application.
- The Dockerfile for the simulation application has the commands to set up the Hello World simulation application.

Both the robot application and the simulation application have entrypoint scripts. These scripts source the environments for their respective applications. They set up the path for you to run commands that give you the ability to run your robot and simulation applications.

Creating a Base Image

To create a base image, you save the commands to create your environment in a Dockerfile. You then build the Dockerfile.

- Save the following commands in a Dockerfile.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: MIT-0
FROM ros:melodic

ENV DEBIAN_FRONTEND noninteractive

RUN apt-get clean
RUN apt-get update && apt-get install -y \
    lsb  \
    unzip \
    wget \
    curl \
    sudo \
    python-vcstool \
    python-rosinstall \
    python3-colcon-common-extensions \
    ros-melodic-rviz \
    ros-melodic-rqt \
    ros-melodic-rqt-common-plugins \
    devilspie \
    xfce4-terminal

ENV QT_X11_NO_MITSHM=1

RUN wget https://packages.osrfoundation.org/gazebo.key -O - | sudo apt-key add -; \
    sh -c 'echo "deb http://packages.osrfoundation.org/gazebo/ubuntu-stable
 `lsb_release -cs` main" > /etc/apt/sources.list.d/gazebo-stable.list'
RUN apt-get update && apt-get install -y gazebo9

ARG USERNAME=robomaker
RUN groupadd $USERNAME
RUN useradd -ms /bin/bash -g $USERNAME $USERNAME
RUN sh -c 'echo "$USERNAME ALL=(root) NOPASSWD:ALL" >> /etc/sudoers'
USER $USERNAME

RUN sh -c 'cd /home/$USERNAME'

# Download and build our Robot and Simulation application
RUN sh -c 'mkdir -p /home/robomaker/workspace'
RUN sh -c 'cd /home/robomaker/workspace && wget https://github.com/aws-robotics/aws-
robomaker-sample-application-helloworld/archive/ros1.zip && unzip ros1.zip'
RUN sh -c 'cd /home/robomaker/workspace/aws-robomaker-sample-application-helloworld-
ros1'

RUN sudo rosdep fix-permissions
RUN rosdep update
```

After you've created the Dockerfile, build it using the following commands on your terminal.

```
cd ../HelloWorldSampleAppROSMelodicGazebo9
docker build -t helloworldsampleapprosmelodicgazebo9:latest .
```

Building the base image installs ROS Melodic and Gazebo 9. You need both libraries installed to successfully run your applications.

Creating an Image for the Robot Application

After you've created the base image, you can create the image for your robot application. You save the following script in a Dockerfile and build it. This script downloads the Hello World robot application and sets it up.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: MIT-0
FROM helloworldsampleappros2foxygazebo11:latest

# Build the Robot application
RUN cd /home/robomaker/workspace/aws-robomaker-sample-application-helloworld-ros2/robot_ws
 && \
 /bin/bash -c "source /opt/ros/foxy/setup.bash && vcs import < .rosinstall && rosdep
 install --rosdistro foxy --from-paths src --ignore-src -r -y && colcon build"


COPY robot-entrypoint.sh /home/robomaker/robot-entrypoint.sh
RUN sh -c 'sudo chmod +x /home/robomaker/robot-entrypoint.sh'
RUN sh -c 'sudo chown robomaker:robomaker /home/robomaker/robot-entrypoint.sh'

CMD ros2 launch hello_world_robot rotate.launch.py
ENTRYPOINT [ "/home/robomaker/robot-entrypoint.sh" ]
```

The following command creates the image for the robot application from the Dockerfile.

```
cd HelloWorldSampleAppROSMelodicGazebo9RobotApp/
HelloWorldSampleAppROSMelodicGazebo9RobotApp
docker build -t helloworldsampleapprosmelodicgazebo9robotapp:latest image/.
```

The following are the contents of the script that you can save as `robot-entrypoint.sh`. This script sources the environment for the robot application.

```
#!/bin/bash
cd /home/robomaker/workspace/aws-robomaker-sample-application-helloworld-ros1/robot_ws
source /opt/ros/melodic/setup.bash
source /usr/share/gazebo-9/setup.sh
source ./install/setup.sh
printenv

exec "${@:1}"
```

Creating an Image for the Simulation Application

After you've created the base image and the image for the robot application, you can create the image for your simulation application. You save the following script in a Dockerfile and build it. This script downloads the Hello World robot application and sets it up.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: MIT-0
FROM helloworldsampleapprosmelodicgazebo9:latest

# Build the Simulation application
RUN cd /home/robomaker/workspace/aws-robomaker-sample-application-helloworld-ros1/
simulation_ws && \
    /bin/bash -c "source /opt/ros/melodic/setup.bash && vcs import < .rosinstall && rosdep
 install --rosdistro melodic --from-paths src --ignore-src -r -y && colcon build"

COPY simulation-entrypoint.sh /home/robomaker/simulation-entrypoint.sh

RUN sh -c 'sudo chmod +x /home/robomaker/simulation-entrypoint.sh'
RUN sh -c 'sudo chown robomaker:robomaker /home/robomaker/simulation-entrypoint.sh'

CMD roslaunch hello_world_simulation empty_world.launch
ENTRYPOINT [ "/home/robomaker/simulation-entrypoint.sh" ]
```

Running the Application and Pushing It to ECR

After you've created your images, make sure that they run properly in your local Linux environment. After you've checked that Docker image runs, you can push it to Amazon ECR and create a simulation job.

The following commands give you the ability to run the Hello World application in your local Linux environment.

```
docker run -it -e DISPLAY -v /tmp/.X11-unix/:/tmp/.X11-unix/ \
-u robomaker -e ROBOMAKER_GAZEBO_MASTER_URI=http://localhost:5555 \
-e ROBOMAKER_ROS_MASTER_URI=http://localhost:11311 \
helloworldsampleapprosmelodicgazebo9robotapp:latest
```

```
docker run -it -e DISPLAY -v /tmp/.X11-unix/:/tmp/.X11-unix/ \
-u robomaker -e ROBOMAKER_GAZEBO_MASTER_URI=http://localhost:5555 \
-e ROBOMAKER_ROS_MASTER_URI=http://localhost:11311 \
helloworldsampleapprosmelodicgazebo9simapp:latest
```

When you run the robot application and simulation application containers, you can visualize the simulation using the Gazebo GUI tool. Use the following commands to:

1. Connect to your container running the simulation application.
2. Visualize your application by running the Gazebo Graphical User Interface (GUI).

```
# Enable access to X server to launch Gazebo from docker container
$ xhost +

# Check that the robot_app and sim_app containers are running. The command should list both
 containers
$ docker container ls

# Connect to the sim app container
$ docker exec -it sim_app bash

# Launch Gazebo from within the container
$ /home/robomaker/simulation-entrypoint.sh ros2 launch gazebo_ros gzclient.launch.py
```

You can add tags to your images. The following commands give you the ability to tag your images.

```
docker tag helloworldsampleapprosmelodicgazebo9robotapp:latest accountID.dkr.ecr.us-
west-2.amazonaws.com/helloworldsampleapprosmelodicgazebo9robotapp:latest
```

```
docker tag helloworldsampleapprosmelodicgazebo9simapp:latest accountID.dkr.ecr.us-
west-2.amazonaws.com/helloworldsampleapprosmelodicgazebo9simapp:latest
```

After you've verified that the application is working properly, you can push to Amazon ECR using the following commands.

```
aws ecr get-login-password --region us-west-2 | docker login --username AWS —password-
stdin accountID.dkr.ecr.us-west-2.amazonaws.com
docker push accountID.dkr.ecr.us-west-2.amazonaws.com/
helloworldsampleapprosmelodicgazebo9robotapp:latest
docker push accountID.dkr.ecr.us-west-2.amazonaws.com/
helloworldsampleapprosmelodicgazebo9simapp:latest
```

You can then run a simulation job on the image. For more information on simulation jobs, see Running Simulation Jobs (p. 151).

Running a GPU Sample Application with ROS2 Foxy and Gazebo 11

This tutorial explains how to use GPU drivers within container images to develop with ROS 2 Foxy and Gazebo 11 by creating and running the Hello World robot application and simulation application using three container images outlined in the following example.

```
### SampleGPUBaseApp // Base Image
#    ### Dockerfile
### SampleGPURobotApp // Image for Robot App
#    ### Dockerfile
#    ### robot-entrypoint.sh
### SampleGPUSimulationApp // Image for Simulation App
#    ### Dockerfile
#    ### simulation-entrypoint.sh
```

Each Dockerfile contains the instructions needed to build each image.

- The Dockerfile for the base image includes commands to set up ROS, Gazebo and GPU drivers.
- The Dockerfile for the robot application includes the commands to set up the Hello World robot application.
- The Dockerfile for the simulation application includes the commands to set up the Hello World simulation application.

Both the robot application and the simulation application have an entrypoint script. These scripts source the environments for their respective applications and set up the path for you to run commands to start your robot and simulation applications.

Creating a Base GPU Image

The following Dockerfile contains the commands to create a base image from NVIDIA OpenGL and install DCV.

- Save the following commands in the Dockerfile in the `SampleGPUBaseApp` directory.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: MIT-0
FROM nvidia/opengl:1.0-glvnd-runtime-ubuntu20.04

ENV DEBIAN_FRONTEND="noninteractive"
ENV QT_X11_NO_MITSHM=1

RUN apt-get clean
RUN apt-get update && apt-get install -y --no-install-recommends \
        ca-certificates \
        devilspie \
        gnupg2 \
        mesa-utils \
        sudo \
        unzip \
        wget \
        xfce4-terminal

RUN wget https://d1uj6qtbmh3dt5.cloudfront.net/NICE-GPG-KEY && gpg --import NICE-GPG-KEY &&
 \
        wget https://d1uj6qtbmh3dt5.cloudfront.net/2021.2/Servers/nice-dcv-2021.2-11048-
ubuntu1804-x86_64.tgz && \
        tar xvzf nice-dcv-2021.2-11048-ubuntu1804-x86_64.tgz && \
        cd nice-dcv-2021.2-11048-ubuntu1804-x86_64 && \
        apt install -y ./nice-dcv-gl_2021.2.944-1_amd64.ubuntu1804.deb

RUN apt update && apt -y install locales && \
        locale-gen en_US en_US.UTF-8 && \
        update-locale LC_ALL=en_US.UTF-8 LANG=en_US.UTF-8

ENV LANG=en_US.UTF-8

RUN apt-get update && apt-get install -y --no-install-recommends curl lsb-release

RUN curl -sSL https://raw.githubusercontent.com/ros/rosdistro/master/ros.key  -o /usr/
share/keyrings/ros-archive-keyring.gpg && \
        curl -s https://raw.githubusercontent.com/ros/rosdistro/master/ros.asc | apt-key
 add - && \
        echo "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/ros-
archive-keyring.gpg] http://packages.ros.org/ros2/ubuntu $(lsb_release -cs) main" | tee /
etc/apt/sources.list.d/ros2.list > /dev/null && \
        apt update && \
        apt install -y ros-foxy-desktop && \
        /bin/bash -c "source /opt/ros/foxy/setup.bash"

RUN apt -y install ros-foxy-gazebo-ros-pkgs

RUN apt-key adv --fetch-keys 'http://packages.osrfoundation.org/gazebo.key' && \
        apt update && \
        apt install -y python3-rosdep git

RUN if [ ! -f "/etc/ros/rosdep/sources.list.d/20-default.list" ]; then \
        rosdep init; \
    fi

RUN rosdep update

RUN apt-get install -y python3-apt python3-pip python3-vcstool python3-testresources

RUN pip3 install -U pytest setuptools colcon-ros-bundle

RUN useradd --create-home robomaker && \
        sh -c 'echo "robomaker ALL=(root) NOPASSWD:ALL" >> /etc/sudoers'

RUN sh -c 'mkdir -p /home/robomaker/workspace' && \
        sh -c 'cd /home/robomaker/workspace && wget https://github.com/aws-robotics/aws-robomaker-sample-application-helloworld/archive/ros2.zip && unzip ros2.zip'
```

After you've created the Dockerfile, build it using the following commands on your terminal.

```
cd SampleGPUBaseApp
docker build -t samplegpubaseapp:latest .
```

Building the base image installs ROS 2 Foxy, Gazebo 11, NVIDIA OpenGL, and NICE-DCV.

Creating an Image for the Robot Application

After you've created the base image, you can create the image for your robot application. Save the following script in the Dockerfile in the `SampleGPURobotApp` directory and build it. This script downloads the Hello World robot application and sets it up.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: MIT-0
FROM samplegpubaseapp:latest

# Build the Robot application
RUN cd /home/robomaker/workspace/aws-robomaker-sample-application-helloworld-ros2/robot_ws
 && \
 /bin/bash -c "source /opt/ros/foxy/setup.bash && vcs import < .rosinstall && rosdep
 install --rosdistro foxy --from-paths src --ignore-src -r -y && colcon build"

COPY robot-entrypoint.sh /home/robomaker/robot-entrypoint.sh
RUN sh -c 'sudo chmod +x /home/robomaker/robot-entrypoint.sh'
RUN sh -c 'sudo chown robomaker:robomaker /home/robomaker/robot-entrypoint.sh'

CMD ros2 launch hello_world_robot rotate.launch.py
ENTRYPOINT [ "/home/robomaker/robot-entrypoint.sh" ]
```

The following are the contents of the script that you save as `robot-entrypoint.sh`. This script sources the environment for the robot application.

```
#!/bin/bash
cd /home/robomaker/workspace/aws-robomaker-sample-application-helloworld-ros2/robot_ws
source /opt/ros/foxy/setup.bash
source /usr/share/gazebo-11/setup.sh
source ./install/setup.sh
printenv

exec "${@:1}"
```

The following command creates the image for the robot application from the Dockerfile.

```
cd SampleGPURobotApp
docker build -t samplegpurobotapp:latest .
```

Creating an Image for the Simulation Application

Creating an Image for the Simulation Application

After you've created the base image and the image for the robot application, you can create the image for your simulation application. You save the following script in a Dockerfile in the `SampleGPUSimulationApp` directory and then build it. This script downloads the Hello World simulation application and sets it up.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: MIT-0
FROM samplegpubaseapp:latest

# Build the Simulation application
RUN cd /home/robomaker/workspace/aws-robomaker-sample-application-helloworld-ros2/
simulation_ws && \
 /bin/bash -c "source /opt/ros/foxy/setup.bash && vcs import < .rosinstall && rosdep
 install --rosdistro foxy --from-paths src --ignore-src -r -y && colcon build"

COPY simulation-entrypoint.sh /home/robomaker/simulation-entrypoint.sh

RUN sh -c 'sudo chmod +x /home/robomaker/simulation-entrypoint.sh'
RUN sh -c 'sudo chown robomaker:robomaker /home/robomaker/simulation-entrypoint.sh'

CMD ros2 launch hello_world_simulation empty_world.launch.py
ENTRYPOINT [ "/home/robomaker/simulation-entrypoint.sh" ]
```

The following are the contents of the script that you save as `simulation-entrypoint.sh`. This script sources the environment for the simulation application.

```
#!/bin/bash
if [ ! -z $GAZEBO_MASTER_URI ]; then
   tmp_GAZEBO_MASTER_URI=$GAZEBO_MASTER_URI
fi

cd /home/robomaker/workspace/aws-robomaker-sample-application-helloworld-ros2/simulation_ws
source /opt/ros/foxy/setup.bash
source /usr/share/gazebo-11/setup.sh

 if [ ! -z $tmp_GAZEBO_MASTER_URI ]; then
    export GAZEBO_MASTER_URI=$tmp_GAZEBO_MASTER_URI
    unset tmp_GAZEBO_MASTER_URI
fi

source ./install/setup.sh
printenv

exec "${@:1}"
```

The following command creates the image.

```
                cd SampleGPUSimulationApp
                docker build -t samplegpusimulationapp:latest .
```

Running the Application and Pushing It to Amazon ECR

After you've created your images, make sure they run properly in your local Linux environment. After you've checked that your image runs, you can push your Docker image to Amazon ECR and create a simulation job.

The following commands give you the ability to run the Hello World application in your local Linux environment.

```
docker run -it -e DISPLAY -v /tmp/.X11-unix/:/tmp/.X11-unix/ --name gpu_robot_app \
-u robomaker -e ROBOMAKER_GAZEBO_MASTER_URI=http://localhost:5555 \
-e ROBOMAKER_ROS_MASTER_URI=http://localhost:11311 \
samplegpurobotapp:latest

docker run -it -e DISPLAY -v /tmp/.X11-unix/:/tmp/.X11-unix/ --name gpu_sim_app \
-u robomaker -e ROBOMAKER_GAZEBO_MASTER_URI=http://localhost:5555 \
-e ROBOMAKER_ROS_MASTER_URI=http://localhost:11311 \
samplegpusimulationapp:latest
```

When you run the robot application and simulation application containers, you can visualize the simulation using the Gazebo GUI tool. Use the following commands to:

- Connect to your container running the simulation application.
- Visualize your application by running the Gazebo Graphical User Interface (GUI).

```
# Enable access to X server to launch Gazebo from docker container
$ xhost +

# Check that the robot_app and sim_app containers are running. The command should list both
 containers
$ docker container ls

# Connect to the sim app container
$ docker exec -it gpu_sim_app bash

# Launch Gazebo from within the container
$ /home/robomaker/simulation-entrypoint.sh ros2 launch gazebo_ros gzclient.launch.py
```

You can add tags to your images. The following commands give you the ability to tag your images.

```
docker tag samplegpurobotapp:latest accountID.dkr.ecr.us-west-2.amazonaws.com/
samplegpurobotapp:latest

docker tag samplegpusimulationapp:latest accountID.dkr.ecr.us-west-2.amazonaws.com/
samplegpusimulationapp:latest
```

After you've verified that the application is working properly, you can push it to Amazon ECR using the following commands.

```
aws ecr get-login-password --region us-west-2 | docker login --username AWS —password-stdin
 accountID.dkr.ecr.us-west-2.amazonaws.com
docker push accountID.dkr.ecr.us-west-2.amazonaws.com/samplegpurobotapp:latest
docker push accountID.dkr.ecr.us-west-2.amazonaws.com/samplegpusimulationapp:latest
```

You can now run a simulation job with GPU Compute using these images. For more information about simulation jobs, see Running Simulation Jobs (p. 151) and Configuring `SimulationJob` Compute (p. 152).

Creating a Simulation Job with a Container Image

After you've run your simulation successfully on your local machine, you can run a simulation job. A simulation job runs your simulation in the cloud.

Before you run an AWS RoboMaker simulation job, use the following AWS CLI commands to describe your applications. You can look at the output to make sure that you're using the correct version of your applications. You can also use it to get the image URI from the `environment` parameter.

```
aws describe-robot-application --application robot-application-arn
```

```
aws describe-simulation-application --application simulation-application-arn
```

After you get the image URI for both your robot and simulation applications, you can run a simulation job using the following AWS CLI commands.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: MIT-0
ROBOT_APP_ARN=your-robot-arn
SIM_APP_ARN=your-sim-arn
aws robomaker create-simulation-job --max-job-duration-in-seconds 300 --iam-
role arn:aws:iam::account-id:role/Admin --robot-applications application=
$ROBOT_APP_ARN,launchConfig="{packageName=hello_world_robot,launchFile=rotate.launch,environmentVariabl
terminal,command=\"/home/robomaker/entry.sh && xfce4-terminal
\",streamOutputToCloudWatch=true,exitBehavior=\"RESTART
\"}]",useDefaultTools="false" --simulation-applications application=
$SIM_APP_ARN,launchConfig="{packageName=hello_world_simulation,launchFile=empty_world.launch,environmen
\"/home/robomaker/entry.sh && gzclient\",streamOutputToCloudWatch=true,exitBehavior=
\"RESTART\"}]",useDefaultTools="false" --output-location s3Bucket=the-Amazon S3-bucket-
containing-the-output,s3Prefix=the-Amazon S3-prefix-that-you-use-to-specify-the-output-
location
{
    "arn": "arn:aws:robomaker-dev:us-west-2:028812250826:simulation-job/sim-33y1by7qprfv",
    "status": "Pending",
    "lastUpdatedAt": "2021-06-17T12:42:36-07:00",
    "failureBehavior": "Fail",
    "clientRequestToken": "21bdb695-c4d4-436b-b214-1bb4bc2df71c",
    "loggingConfig": {
        "recordAllRosTopics": false
    },
    "maxJobDurationInSeconds": 3600,
    "simulationTimeMillis": 0,
    "iamRole": "arn:aws:iam::028812250826:role/Admin",
    "robotApplications": [
        {
            "application": "arn:aws:robomaker-dev:us-west-2:028812250826:robot-application/
nobundle_bb_robot_app_1/1623894200715",
            "applicationVersion": "$LATEST",
            "launchConfig": {
                "packageName": "hello_world_robot",
                "launchFile": "rotate.launch",
                "streamUI": false
            },
            "tools": [
                {
                    "sessionId": "robot-rviz",
                    "streamUI": true,
                    "name": "rviz",
                    "command": "rviz",
                    "streamOutputToCloudWatch": false,
                    "exitBehavior": "RESTART"
                },
                {
                    "sessionId": "robot-rqt",
                    "streamUI": true,
                    "name": "rqt",
                    "command": "rqt",
                    "streamOutputToCloudWatch": false,
                    "exitBehavior": "RESTART"
                },
                {
                    "sessionId": "robot-Terminal",
                    "streamUI": true,
                    "name": "Terminal",
                    "command": "xfce4-terminal",
                    "streamOutputToCloudWatch": false,
                    "exitBehavior": "RESTART"
:...output truncated ...
```

You can check on the status of your simulation job by using the following AWS CLI command.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: MIT-0
aws robomaker describe-simulation-job --job job-arn
{
    "arn": "arn:aws:robomaker-dev:us-west-2:028812250826:simulation-job/sim-ml7z6w290hdz",
    "status": "Completed",
    "lastStartedAt": "2021-06-17T22:46:34-07:00",
    "lastUpdatedAt": "2021-06-17T22:52:54-07:00",
    "failureBehavior": "Fail",
    "clientRequestToken": "e5518bc5-bb13-424b-ae5b-ff737d50153d",
    "outputLocation": {
        "s3Bucket": "nobundle-bb-pdx",
        "s3Prefix": "simJob"
    },
    "loggingConfig": {
        "recordAllRosTopics": false
    },
```

Using Amazon S3 Prefix and Archive Data Sources for Your Assets

You can add data sources to either your container image or your simulation job. You can use these data sources to load in assets for a simulation job.

Your data source can be one of the following:

- Amazon S3 prefix
- Data archive
- File

If you don't specify a data source, AWS RoboMaker uses a file as the default data source.

Both Amazon S3 prefixes and data archives give you the ability to specify multiple files. For backwards compatibility with previous versions of AWS RoboMaker, this defaults to `/opt/robomaker/datasources`.

To specify a data source, you specify the `type` and `destination` fields in the data source object.

To use a data source in a simulation job, you specify the following fields under `dataSources` in the CreateSimulationJob (p. 287) operation:

- `destination`: The location in the file system where the data sources are located. For example, if you specify *`DOC-EXAMPLE-BUCKET`* as your bucket, `test.txt` as your `S3Key`, and `/home` as your destination, your robot and simulation applications have the file `/home/test.txt`.
- `type`: Whether your data source is an Amazon S3 prefix, archive, or file.

All of your data sources must be stored on Amazon S3 for AWS RoboMaker to access them.

If you're providing an archive, the suffix of the Amazon S3 object key that you provide must be `.tar.gz` or `.zip`. You can't provide more than one Amazon S3 key.

Archives are unarchived in your robot and simulation application. For example, set the following fields with the following values:

- `destination`: `/home`
- `type`: `archive`
- `s3Key`: `test.zip`

  For this example, `test.zip` contains `test1.txt` and `test2.txt`

If you set the preceding fields with the listed values, the following files are available to your application when you run it:

- `/home/test1.txt`
- `/home/test2.txt`

If you provide an Amazon S3 prefix, AWS RoboMaker uses all of the files that match the prefix. For more information, see Using Amazon S3 Prefixes to Load Your Data Into AWS RoboMaker (p. 63).

Topics

Using Amazon S3 Prefixes to Load Your Data Into AWS RoboMaker

To understand how to use prefixes as your data source in a simulation job, you need to be familiar with the following Amazon S3 concepts:

- *Bucket*: A container to store your objects.
- *Object*: The entity stored in the Amazon S3 bucket.
- *Key*: The unique identifier for an object within a bucket.
- *Prefix*: Any portion of the key up to the final delimiter. You use prefixes to organize your data and specify objects in the Amazon S3 bucket.

You store an object, your text file, in a bucket with a key that uniquely identifies the file.

For example, the key `s3://`*`DOC-EXAMPLE-BUCKET1`*`/myfiles/2020/may/file.txt` uniquely identifies a text file in an Amazon S3 bucket.

The prefix can be any part of the key that comes before its final delimiter.

For example, the key `myfiles/2020/may/file.txt` has the following prefixes:

- s3://*DOC-EXAMPLE-BUCKET1*/myfiles/2020/may/
- s3://*DOC-EXAMPLE-BUCKET1*/myfiles/2020/
- s3://*DOC-EXAMPLE-BUCKET1*/myfiles/

You can use any of the preceding prefixes to access `file.txt`. For more information on prefixes, see Organizing objects using prefixes.

Creating a Simulation Job

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

Topics

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. 22).

Optionally, you will need access to 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. 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=Melodic
```

   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.

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

3. Create a simulation application in AWS RoboMaker.

```
$ aws robomaker create-simulation-application --name MySimulationApplication
 --sources s3Bucket=myapplicationsource,s3Key=my-simulation-
application.tar.gz,architecture=X86_64 --robot-software-suite name=ROS,version=Melodic
 --simulation-software-suite name=Gazebo,version=9 --rendering-engine
 name=OGRE,version=1.x
```

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

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. 67).

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:

```
{
    "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"
        }
        {
            "Action": [
                "ecr:BatchGetImage",
                "ecr:GetAuthorizationToken",
                "ecr:BatchCheckLayerAvailability",
                "ecr:GetDownloadUrlForLayer"
            ],
            "Resource": "arn:partition:ecr:region:account#:repository/repository_name",
            "Effect": "Allow"
        }
    ]
}
```

Replace `my-input-bucket` and `my-output-bucket` with your input and output bucket names. Replace `account#` with your AWS account number.

Choose **Review policy**, type in a **Name**, and then choose **Create policy**.

3. Choose **Roles** and then choose **Create role**.
4. In the **Create role: Step 1** page, choose **RoboMaker**. Choose **RoboMaker - Simulation** as the use case, then choose **Next: Permissions**.
5. In the **Permissions** page, select the policy you created above, then choose **Next: Tags**.
6. In the **Add tags** page, add optional tags to the role, then choose **Next: Review**.
7. In the **Review** page, type in a **Role name** and then choose **Create role**.
8. Select **Roles** and then select the role you created. Note the Amazon Resource Name (ARN) for the role. You will use it when you create the simulation job.

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.

```
$ aws robomaker create-simulation-job --max-job-duration-in-seconds 3600 --
iam-role arn:aws:iam::111111111111:role/RoboMakerSimulationJobRole --robot-
applications application=arn:aws:robomaker:us-west-2:111111111111:robot-application/
MyRobotApplication/1551987693226,launchConfig='{packageName=robot_app,launchFile=rotate.launch}'
 --simulation-applications application=arn:aws:robomaker:us-
west-2:111111111111:simulation-application/
MyRobotApplication/1551988628094,launchConfig='{packageName=simulation_app,launchFile=example.launc
```

3. Check on the status of your simulation job using the following commands.

```
$ aws robomaker list-simulation-jobs
$ aws robomaker describe-simulation-job --job arn:aws:robomaker:us-
west-2:111111111111:simulation-job/sim-pql32v7pfjy6
```

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. 178).

Using generated worlds in your simulation

Simulation WorldForge is integrated with AWS RoboMaker. The worlds you create with Simulation WorldForge are available to simulations in AWS RoboMaker. You do not need to export, modify, bundle, and upload a world to use it in your simulation. However, you might need to export and modify a world if your world requires:

- Physics that are different from the default SDF physics
- Specialized lighting
- Custom models

This section provides more information about how you can use generated worlds in your simulation. You do not need to export a world to use it in your simulation.

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

Topics

Using an imported world in a simulation job

Simulation WorldForge is integrated into AWS RoboMaker. You can use a generated world in your simulation by modifying your simulation application launch file. You do not need to export or modify the world unless you have a unique scenario.

To modify your launch file

1. Update your simulation application launch file.

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

  <include file="$(find aws_robomaker_worldforge_worlds)/launch/launch_world.launch">
    <arg name="gui" value="$(arg gui)"/>
  </include>

  <!-- Your other launch commands go here. -->
</launch>
```

   You can spawn your robot at (0, 0, 0). The worlds Simulation WorldForge generates are guaranteed to have a 1 meter cylinder clear at (0, 0, 0).
2. On the console, when you create or clone a simulation job, in the **Import world from WorldForge** section, choose a world.

   In the AWS CLI, specify the world in `worldConfigs`:

```
worldConfigs={world=arn:aws:robomaker:eu-west-1:MyAccount:world/MyGeneration/MyWorld
```

Using an exported world locally in ROS and Gazebo

Simulation WorldForge makes it easy to export worlds you can use in your ROS environment. The world you choose to export is copied to a single .zip file. The .zip file includes all of the assets needed to modify and visualize the worlds using ROS and Gazebo. It includes the following important folders:

- The root folder, **workspace_src**, is the ROS workspace. It contains shared models, world data, and other information for the worlds. It is compatible with ROS 1 and ROS 2.
- **Shared models** is copied to `workspace_src/src/aws_robomaker_worldforge_shared_models/models`. For example, if the same chair is used in more than one worlds, it will be placed in the shared model folder.

- **World data** is copied to `workspace_src/src/aws_robomaker_worldforge_worlds/worlds/`.

To build and launch the world

1. Follow the procedure in Creating a World Export Job (p. 148) to export a world.
2. `Unzip` the world into an ROS workspace.

```
$ cd MyApplication/simulation_ws
$ unzip MyExportedWorld.zip
```
3. Build the world.

```
$ rosdep install --from-paths src --ignore-src -r -y
$ colcon build
```
4. Launch the world.

```
$ source install/setup.sh
$ roslaunch aws_robomaker_worldforge_worlds launch_world.launch gui:=true
```

Using an exported world with custom physics, lights, and models

If your simulation scenario requires customization, you can export and modify the world. For example, you can apply custom physics, different lighting effects, add custom models, or make other modifications.

After the world is exported, you need to modify the `.world` file to include the exported world model. The `.world` file uses SDF. For more information about SDF, see SDFormat.

To modify your `.world` file to include the exported world model

1. Follow the procedure in Creating a World Export Job (p. 148) to export a world.
2. Copy the following to your `.world` file. Make sure the world name matches the exported model name.

```
<sdf version="1.6">
  <world name="generation_82856b0yq33y_world_16">
    <model name="WorldForge World"
      <include>
        <uri>model://generation_82856b0yq33y_world_16</uri>
      </include>
    </model>
    <!-- Your other <world> elements go here -->
  </world>
</sdf>
```
3. Verify that your launch file includes the modified `.world` file. Use the updated launch file to launch your simulation.

AWS RoboMaker Cloud Extensions

Important
Support Ending January 2022

On January 31, 2022, the AWS RoboMaker cloud extensions GitHub repositories will be migrated to AWS Samples GitHub. You can continue to use the AWS RoboMaker cloud extensions within new and existing robot and simulation applications. But after January 31, 2022, if the cloud extensions break or require security updates, you will be responsible for implementing the updates.

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.
- **Amazon S3** — Easily record and store robotic application data.

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. 8).
- 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.
- ROS Melodic on Ubuntu 18.04. If you are using an AWS RoboMaker development environment, it is preconfigured for robotics development with the chosen distribution.

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
- Amazon S3 rosbags

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

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:

1. Sign in to the AWS RoboMaker console at https://console.aws.amazon.com/robomaker/.
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.
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**.

Virtual Desktop (NICE DCV)

When you launch a virtual desktop, AWS RoboMaker launches the NICE DCV web browser client. The NICE DCV web browser client provides access to your AWS Cloud9 desktop and streaming applications.

You can access your folders, launch applications like Gazebo and your own custom applications. This makes it easy to interact with your robot simulation and other ROS applications.

The virtual desktop provides a view into your active AWS Cloud9 session. If you launch more than one virtual desktop, each desktop will access the same AWS Cloud9 session. Changes made in one virtual desktop will be seen in all virtual desktop instances connected to the AC9; instance.

To launch the virtual desktop, open a development environment. On the menu choose **Virtual Desktop** and then choose **Launch Virtual Desktop**. It may take a couple seconds to load in a new browser tab.

To exit the virtual desktop, close the NICE DCV browser window.

For more information about NICE DCV web client, see Web Browser Client.

For a detailed example of using the virtual desktop, please see this blog post or the Getting Started section.

Build and Bundle Applications

Once you have a working simulation in a development environment, you can bundle it, upload to Amazon S3, and create a RoboMaker simulation job to run it. There are a couple of build tools that developers use with ROS. The one that is most commonly used with AWS RoboMaker is called colcon. It automates the building and bundling of ROS and ROS2 applications. It should be a drop-in replacement for `catkin_make`. Colcon bundle will generate an easy-to-deploy `.tar` file.

Install Colcon

To use colcon, install colcon-ros-bundle and Python 3.5 or above.The following commands will install`colcon, python3` and`colcon-bundle`:

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

*Note: When using the “sudo apt update” command, if you receive a GPG error follow these instructions Troubleshooting Development Environments (p. 231).

If you have already installed `colcon`, the following command will install bundling support:

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

Build and Bundle

After colcon is installed, use the following commands to build and then bundle your robotics application and simulation application:

```
cd your-robotic-application-workspace
colcon build
colcon bundle
cd your-simulation-application-workspace
colcon build
colcon bundle
```

You can test your application prior to bundling and uploading with the following:

```
source install/setup.bash
```

```
roslaunch package_name launch_file
```

The colcon build and bundle commands produces the artifacts `robot_ws/bundle/output.tar` and `simulation_ws/bundle/output.tar` respectively. You must upload these to an S3 bucket. Then you can use these files to do the following tasks:

1. Creating a Robot Application (p. 74)
2. Creating a Simulation Application (p. 79)
3. Creating a Simulation Job (p. 166)

Note
Before running a simulation job in AWS RoboMaker simulation, check the GUI argument in the launch file. The launch file starts the Gazebo server, and setting `gui:=true` will also launch the Gazebo client that is used for viewing the simulation on the Ubuntu desktop. However, when running an AWS RoboMaker simulation job, the Gazebo server and client are started separately and you should ensure that your launch files do not set the gui argument to true. You can start and configure the Gazebo client in the AWS RoboMaker simulation service using the simulation tool configuration.

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.

AWS RoboMaker Resources

Command	Description		
Download Samples	Download selected sample application.		
View Extensions	Download selected cloud extension.		

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:

1. Sign in to the AWS RoboMaker console at https://console.aws.amazon.com/robomaker/.
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

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

1. Sign in to the AWS RoboMaker console at `https://console.aws.amazon.com/robomaker/`.
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. - a. 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.
     b. Optionally, choose **Create new S3 folder** to go to the Amazon Simple Storage Service AWS Management Console to create and manage buckets.
     c. Using `$LATEST` doesn't protect you from changes in Amazon S3. When AWS RoboMaker access the file, it will set to read only. For more informatiom about versioning in AWS RoboMaker, see Application Versioning (p. 19).

- a. Instead of using a bundle application file, you can use container images to develop your applications. You can use images that you've pushed to Amazon ECR to develop your applications.

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

     For more about tagging, see Tagging Your AWS RoboMaker Resources (p. 224).

  c. 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.

```
$ 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
```

Example

Here's an example AWS CLI command that uses an image to create the robot application.

```
$ need the image command
```

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. 19).

To create a robot application version

Follow the steps under one of these tabs.

Using the console

1. Sign in to the AWS RoboMaker console at `https://console.aws.amazon.com/robomaker/`.
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.

```
$ 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

1. Sign in to the AWS RoboMaker console at `https://console.aws.amazon.com/robomaker/`.
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.

```
$ 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

1. Sign in to the AWS RoboMaker console at `https://console.aws.amazon.com/robomaker/`.
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.

```
$ 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

1. Sign in to the AWS RoboMaker console at `https://console.aws.amazon.com/robomaker/` .
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

1. Sign in to the AWS RoboMaker console at `https://console.aws.amazon.com/robomaker/` .
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

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

1. Sign in to the AWS RoboMaker console at https://console.aws.amazon.com/robomaker/.
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.

Using `$LATEST` doesn't protect you from changes in Amazon S3. When AWS RoboMaker access the file, it will set to read only. For more information about versioning in AWS RoboMaker, see Application Versioning (p. 19).

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. 224).
9. Choose **Create**.

Using the AWS CLI

Example

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

```
$ aws robomaker create-simulation-application \
--name HelloWorldSimulationApplication \
--rendering-engine name=OGRE,version=1.x \
--simulation-software-suite name=Gazebo,version=9 \
--robot-software-suite name=ROS,version=Melodic \
--sources architecture=X86_64,s3Bucket=my-bucket-name,s3Key=my-key-name/hello-world-
simulation.tar
```

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. 19).

To create a simulation application version

Follow the steps under one of the following tabs.

Using the console

1. Sign in to the AWS RoboMaker console at https://console.aws.amazon.com/robomaker/.
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.

```
$ 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

1. Sign in to the AWS RoboMaker console at https://console.aws.amazon.com/robomaker/.
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.

```
$ 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

1. Sign in to the AWS RoboMaker console at https://console.aws.amazon.com/robomaker/.

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.

Using the AWS CLI

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

- **API/SDK:** `UpdateSimulationApplication`
- **AWS CLI:** `update-simulation-application`

Example

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

```
$ aws robomaker update-simulation-application \
--application my-robot-application-arn \
--rendering-engine name=OGRE,version=1.x \
--simulation-software-suite name=Gazebo,version=9 \
--sources architecture=X86_64,s3Bucket=my-bucket-name,s3Key=my-key-name/hello-world-
simulation.tar
```

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

1. Sign in to the AWS RoboMaker console at https://console.aws.amazon.com/robomaker/.
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.

```
$ 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

1. Sign in to the AWS RoboMaker console at https://console.aws.amazon.com/robomaker/.
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
```

Creating Worlds with Simulation WorldForge

Simulation WorldForge makes it easier and faster to create simulation worlds. Worlds are generated from the simulation world templates you define. The simulation world template specifies the world layout, room dimensions, furnishings, how rooms are connected, and other details. Walls and floors and other room features can have material properties. Rooms can be furnished by room type automatically or you can select potential furnishings. Generated worlds can be used in your simulation jobs and exported to use on your developer machine.

Simulation WorldForge can help you manage simulation workloads that require a large number of simulation worlds with domain randomization. Common Simulation WorldForge scenarios include the following:

- **Regression testing** — Test your robotics applications in hundreds of worlds to verify correct behavior.
- **Synthetic imagery data generation** — You can capture images from the generated worlds to use in other robotic applications. For example, you can capture images of rooms with different furniture layout and material composition.
- **Reinforcement learning** — Create hundreds of unique worlds with an interior structure for your robotic application to explore. You control the composition of the world.
- **Developing algorithms** — A robotics navigation engineer could verify a navigation algorithm succeeds in a known layout with different furniture placement. A robotics localization engineer can ensure a layout algorithm detects different structural elements in different floor plans.

You do not need to know world generation algorithms or how to create and manage infrastructure. Simulation WorldForge and AWS RoboMaker are fully managed services.

Simulation WorldForge Concepts

Simulation WorldForge uses a collection of parameters (a *simulation world template*) to determine how to generate new worlds. One simulation world template can be used to generate hundreds of worlds. Each *world* contains a building. The building has a single floor. The floor has a *floorplan template* that describes the rooms sizes and shapes. It also suggests how the rooms might be connected. Floors also have an *interior template* that specifies how structural elements like walls and floors in the floor plan are finished. The interior template also has parameters describing how to populate each room with furnishing like tables and sofas and accessories like clothes and kitchenware.

You can create a simulation world template from a sample template, clone an existing template, or start from scratch using the console at https://console.aws.amazon.com/robomaker/. For example, if you want to generate worlds containing one bedroom, you can start with the one bedroom apartment sample template. It is a one bedroom, one bathroom open floor plan with a kitchen and a living room. It uses typical materials and furnishings and accessories appropriate for each room type. After it is saved, you can start a *world generator job* to generate worlds. You can generate up to 50 worlds in a world generation job.

You can also create a simulation world template using the SDK or the AWS Command Line Interface. For example, to create a template from the AWS CLI, first create a *world template JSON document* with the template body. It specifies the parameters for the building, floor plans, interiors, and other details. After you save it, you can create simulation world template by calling `create-world-template` and specifying the JSON file:

```
aws robomaker create-world-template --name "my-template" --templateBody file://
my_template_body.json
```

After you have configured and saved a simulation world template, you can create a world generation job and generate worlds. Hundreds of worlds can be generated from a single simulation world template. You can generate up to 100 worlds in a single world generation job. Worlds can be used with a simulation in AWS RoboMaker. You can also export worlds to modify and use in your own ROS environment.

Understanding Simulation World Templates

This section describes the components of a simulation world template. Components include the floor plan and preferences for interior materials and furniture. Simulation WorldForge provides defaults for many of the components including materials, furniture selection, and room connectivity. You can override the defaults with your own preferences. Simulation WorldForge will make a best effort to follow your preferences when generating worlds.

Floor Plan

The floor plan specifies an indoor floor plan for a single-story, residential building. It includes the world dimensions, the number and types of rooms, and parameters that influence how rooms are connected.

Every world is guaranteed to have a clear 1 meter cylinder centered at coordinate `(0,0,0)`, the default robot starting position. Simulation WorldForge determines the room.

World Dimensions

You can configure an aspect ratio and a ceiling height for the building. Valid aspect rations are from 1:4 to 4:1. Valid ceiling height is 2.4 to 4.0 meters. All measurements are in meters and square meters. The console supports conversion between the US and metric system.

Rooms

You can specify the number of rooms, room type, room name, desired area, desired aspect ratio, and interior features. The following room types are supported:

- Bedroom
- Bathroom
- Living
- Dining
- Kitchen
- Hallway
- Closet

Furniture, wall material, and floor material are selected from types appropriate for the room type. For example, a bathroom might be assigned a tile wall, a linoleum floor, and have a toilet and a shower.

Connections

Simulation WorldForge automatically connects all rooms by default. You can connect rooms by an opening or a doorway. When rooms are connected by an opening, the rooms will be in an open floor plan. There is no wall. Rooms connected by a doorway have a narrow, doorless opening. Doorway openings are randomly placed along the adjacent wall.

You can override default connections with desired connections. For example, if you have a kitchen, dining room, and a bedroom, you can request a door connection between the kitchen and bedroom. Simulation WorldForge will make a best effort to make the connection, but it is not guaranteed.

Interiors

You can select from a number of different interior material and furniture types. Simulation WorldForge randomly assigns flooring, walls, and furniture to rooms by room type. For example, a kitchen might be assigned an oven and dining room table and chairs.

You can select material types for flooring and walls as a custom set. When you create a custom set, you can apply the custom assignment by **room type** or **room name**. You can have multiple custom sets. If there is a conflict, a custom assignment for a room always has precedence over one for room type.

For example, assume you have custom set "Modern Flooring" assigned to all bedrooms and a custom set "Chic Flooring" assigned to the room "Master Bedroom". When Simulation WorldForge assigns flooring materials, "Master Bedroom" will be assigned flooring material from the "Chic Flooring" set. Other bedrooms will have flooring material selected from the "Modern Flooring" set.

This rule also applies to custom furniture sets.

Flooring Material Types

Supported flooring types include the following:

- Carpet
- Concrete
- Floorboards
- Linoleum
- Parquetry
- Tiles

Flooring material is randomly chosen from all of the flooring material types selected. For example, if you specify `Carpet`, `Concrete`, `linoleum` and `parquetry`, the floor of your room might be concrete.

Wall Material Types

Supported wall material types include the following:

- Brick
- Concrete
- Stone
- Tiles
- Wood panels
- Wall paint
- Wallpaper

Wall material is randomly chosen from all of the wall material types selected. For example, if you specify `Brick`, `Tiles` and `Wallpaper`, your room might have walls that use tile and wallpaper. Simulation WorldForge might not assign wall material from all chosen wall material types.

Furniture Types

Simulation WorldForge supports the following furniture types:

- Baths
- Bar cabinets
- Beds
- Bookcases
- Coffee tables
- Console tables
- Corner cabinets
- Desk chairs
- Desks
- Dining chairs
- Dining tables
- Dish washers
- Dressers
- End and side tables
- Floor lamps
- Fridges
- Living room chairs
- Kitchen islands and carts
- Media storage
- Nightstands
- Ottomans
- Ovens
- Serving carts
- Showers
- Sideboards and buffets
- Sofas
- Storage
- Storage benches
- Toilets
- Vanity counters
- Washing machines and dryers

Furniture is randomly chosen from all of the furniture types selected. For example, if you specify `Sideboards and buffets`, `Sofas` and `Console tables`, your room might have a sofa and two console tables, but no sideboard or buffet. Simulation WorldForge might not assign material types from all chosen furniture types.

Common Tasks

This section contains common tasks for creating simulation world templates. Many of the tasks specify desired connections or desired shapes. Simulation WorldForge makes a best effort to generate worlds according to simulation world template parameters. Generated worlds might not always include all desired properties.

Topics

Specifying a List of Rooms for a Floor

The room type influences the floor plan by contributing to what rooms are adjacent. The room type is also used to determine the types of material for its flooring and walls and the types of furniture to randomly place by default. You may override the default flooring and walls material types and furniture types by room type or room name.

You can select from the following room types: Bedroom, Bathroom, Living Room, Dining Room, Kitchen, Hallway, Closet.

The following examples specifies a three room house. The sizes and shapes of the rooms are determined by default.

Using the console

1. In the **Simulation world template edit** screen, under **Floor plan**, choose **Rooms**.
2. In the **Rooms** pane, choose **Add room**.
3. Add details for the room. You can specify a room **Name**, **Room type**, **Desired area** and **Desired aspect ratio**.
4. Choose **Save** to save the new room. Repeat until you have the rooms you desire. If you add too many, you can delete them from the **Rooms** pane.

Using the AWS CLI

Example

You can use the following JSON in the `templateBody` as part of a call to `create-world-template`.

```
"Rooms": [
  {
   "Type": "Bedroom",
   "Name": "My Master Bedroom",
  },
  {
   "Type": "Bathroom",
   "Name": "My Ensuite",
  },
  {
   "Type": "Kitchen",
   "Name": "My Kitchen",
  }
]
```

Requesting a Long Hallway

You can use the `DesiredShape` property to request the preferred shape of a room. `Type` has no affect on the shape. In the following example, the `Hallway` aspect ratio is low. When it is combined with a large enough `Area`, it indicates a desire for a long, narrow hallway. Simulation WorldForge will attempt to generate rooms similar to the desired shape.

Using the console

1. In the **Simulation world template edit** screen, under **Floor plan**, choose **Rooms**.
2. In the **Rooms** pane, choose **Add room**.
3. Specify a room **Name**, then choose **Hallway** for **Room type**.
4. Specify a **Desired area** of `20` and a **Desired aspect ratio** of `4:1`.
5. Choose **Save** to save the hallway.

Using the AWS CLI

Example

You can use the following JSON in the `templateBody` as part of a call to `create-world-template`.

```
"Rooms": [
  {
    "Type": "Hallway",
    "Name": "My Hallway",
    "DesiredShape": {
      "Area": 20.0,
      "AspectRatio": {
        "x": 4, "y": 1
      }
    }
  }
]
```

Valid room area range is 10 meters to 300 meters. Valid room aspect ratio range is 1:4 to 4:1.

Requesting a Doorway Between Rooms

If you have two rooms, and the rooms share at least one wall, you can request a `DesiredConnections` between the two rooms. Simulation WorldForge will try to place the rooms adjacent and, depending on the `ConnectionType`, either place a `Doorway` in a random location along an adjacent wall or create `Opening` by removing an adjacent wall entirely.

The following example requests an open connection for the living room and kitchen. It also requests a separate doorway connection for the bedroom and bathroom:

Using the console

1. In the **Simulation world template edit** screen, under **Floor plan**, choose **Connections**.
2. In the **Connections** pane, choose **Add connection**.
3. In the **Desired connections** pane, select **Opening** for **Connection type** and then select a room for **Location 1** and **Location 2**. For example, "My Living Room" and "My Kitchen".
4. Choose **Save** to save desired connections.
5. Repeat to add a **Door** as a desired connection between two other locations. For example, "My Bedroom" and "My Bathroom".

Using the AWS CLI

Example

You can use the following JSON in the `templateBody` as part of a call to `create-world-template`.

```
"DesiredConnections": [
  {
    "Location": [ "My Living Room", "My Kitchen" ],
    "ConnectionType": "Opening"
  },
  {
    "Location": [ "My Bedroom", "My Bathroom" ],
    "ConnectionType": "Doorway"
  }
]
```

Valid number of connections per room is 4 and a maximum of one opening connection for each pair of rooms.

Applying a Configuration to All Rooms

Note
You can only apply a configuration to all rooms by using templates that are Version 2 and above. For more information, see Applying a configuration to all rooms (p. 144).

You can use the `Target.All` keyword to apply a configuration to all rooms.

The following example changes the door state for all doors.

Using the console

The following procedure gives you the ability to apply a configuration for all the doors in your world. You can also apply a single configuration to all floors, material sets, walls, and furniture.

1. In the **Simulation world template edit** screen, under **Interiors**, choose **Doors**.
2. In the **Doors** pane, choose **Add custom doors**.
3. For **Set name**, specify a name for your set of custom doors.
4. For **Rooms affected**, specify **All rooms**.
5. For **Door state**, choose the open state of the door.
6. Choose **Save** to save the doors configuration.

Using the AWS CLI

Example

You can use the following JSON in the `templateBody` as part of a call to `create-world-template`. The following example targets all doors within a doorway set.

```
"Interior": {
  "Doorways": {
    "DoorwaySets": [
      {
        "Name": "your-doorway-set",
        "TargetSet": "Target.All",
        "Door": {
          "InitialState": {
            "OpenPosition": {
              "Percent": "percentage-that-you-specify"
            }
          }
        }
      }
    ]
  }
}
```

Requesting Doors in Doorways

Note
You can only configure doorways to have doors using world templates that are Version 2 and above.

You can use a template to specify doors in doorways in your AWS RoboMaker Simulation WorldForge world.

You can specify the following types of doors:

- Hinged door

You can configure the percentage in which these doors are open. For example, these are some open states that you can specify:

- 0% open – closed
- 50% open – halfway open
- 70% open – mostly open
- 100% open – entirely open

You can also choose to have AWS RoboMaker assign a randomized open percentage to each door.

You can use the following procedure to add doors to your doorways.

Using the console

1. In the **Simulation world template edit** screen, under **Interiors**, choose **Doors**.
2. In the **Doors** pane, choose **Add custom doors**.
3. For **Set name**, name your custom door set.
4. For **Rooms affected** under **Location**, choose the rooms that you want to have doors.

5. For **Door type** under **Customizations**, choose the type of door that you're adding.
6. Under **Door state**, choose whether the door is open, closed, partially open, or in a randomized state.
7. Choose **Save** to save the configuration.

Using the AWS CLI

Example

You can use the following JSON in the `templateBody` as part of a call to `create-world-template`.

```
"Interior": {
  "Doorways": {
    "DoorwaySets": [
      {
        "Name": "your-doorway-set",
        "TargetSet": "the-doorways-that-you-want-to-target",
        "Door": {
          "InitialState": {
          "OpenPosition": {
            "Percent": "the-open-percentage-that-you-specify-for-the-doors-that-you're-
targeting"
          }
        }
      }
    ]
  }
}
```

Requesting No Doors in Doorways

Note
You can only explicitly specify no doors in doorways using world templates that are Version 2 and above.

You can use a template to explicitly specify that the doorways in your AWS RoboMaker Simulation WorldForge world have no doors in the doorways.

The following example requests that there are no doors in the doorways between rooms.

Using the console

1. In the **Simulation world template edit** screen, under **Interiors**, choose **Doors**.
2. In the **Doors** pane, choose **Add custom doors**.
3. For the **Rooms affected** pane under **Location**, choose **All rooms**.
4. For **Door type** under **Customizations**, choose **No door in doorway**.
5. Choose **Save**.

Using the AWS CLI

Example

You can use the following JSON in the `templateBody` as part of a call to `create-world-template`.

```
"Interior": {
"Doorways": {
  "DoorwaySets": [
    {
      "Name": "doorway-set-name",
      "TargetSet": "Target.All",
      "Door": null
    }
  ]
}
```

Requesting a Wide Floor Plan Footprint

If you want a longer or wider floor plan layout that affects all of the rooms, you can request a `DesiredAspectRatio` for the `Footprint`. Simulation WorldForge uses this preference to influence the overall shape and positions of the rooms so the floor plan better fits the requested footprint aspect ratio. The desired aspect ratio is optional and defaults to a square.

The following examples overrides the default square ratio (1:1) to a prefer a wider layout where all of the rooms are more likely to be stretched and placed to create a non-square footprint:

Using the console

1. In the **Simulation world template edit** screen, under **Floor plan**, choose **World dimensions**.
2. In the **World dimensions** pane, under **Desired aspect ratio**, specify a **Width** of `1` and a **Length** of `4`.
3. Choose **Save** to save the new room.

Using the AWS CLI

Example

You can use the following JSON in the `templateBody` as part of a call to `create-world-template`.

```
"Footprint": {
  "DesiredAspectRatio": {
    "x": 1, "y": 4
  }
}
```

The valid range for `DesiredAspectRatio` is a range from 1:4 to 4:1.

Requesting a Custom Ceiling Height

The floor plan ceiling height determines the height of the walls for all the rooms. The default ceiling height is 2.4 meters. In this example, we override the default to 3.2 meters:

Using the console

1. In the **Simulation world template edit** screen, under **Floor plan**, choose **World dimensions**.
2. In the **World dimensions** pane, specify a **Ceiling height** of `3.2`.

3. Choose **Save** to save the new room.

Using the AWS CLI

Example

You can use the following JSON in the `templateBody` as part of a call to `create-world-template`.

```
"Ceiling": {
  "Height": 3.2
}
```

Specifying the Same Material Types to Floors in Different Rooms

Use either room types or room names and list multiple rooms for the interior flooring section. In the following example, all of the bedrooms, living rooms and dining rooms will have a random floorboard material assigned.

Using the console

1. In the **Simulation world template edit** screen, under **Interiors**, choose **Flooring**.
2. In the **Flooring** pane, choose **Add flooring**.
3. In the **Custom flooring** pane, specify a flooring **Set name**. For example, "Flooring Material Set 1".
4. Under **Filter type**, choose **By room type**.
5. Under **Room types**, select **Bedrooms**, **Living rooms**, and **Dining rooms**.
6. Under **Custom flooring**, choose **Add material** and then choose **Floorboard**.
7. Choose **Save** to save the flooring set.

Using the AWS CLI

Example

You can use the following JSON in the `templateBody` as part of a call to `create-world-template`.

```
"Flooring": {
  "MaterialSets": [
    {
      "Name": "Flooring Material Set 1",
      "TargetSet": {
        "RoomTypes": [ "Bedroom", "Living", "Dining" ]
      },
      "SampleSet": {
        "MaterialTypes": [ "Floorboards" ]
      }
    }
  ]
}
```

Specifying Different Material Types to Floors Between Rooms of the Same Type

In the following example, all of the bedrooms, living rooms and dining rooms will have a random floorboard material assigned except for “Bedroom 3”. It will be assigned a random carpet material.

Using the console

1. In the **Simulation world template edit** screen, under **Interiors**, choose **Flooring**.
2. In the **Flooring** pane, choose **Add flooring**.
3. In the **Custom flooring** pane, specify a flooring **Set name**. For example, "Flooring Material Set 1".
4. Under **Filter type**, choose **By room type**.
5. Under **Room types**, select **Bedrooms**, **Living rooms**, and **Dining rooms**.
6. Under **Custom flooring**, choose **Add material** and then choose **Floorboard**.
7. Choose **Save** to save the flooring set.
8. In the **Flooring** pane, choose **Add flooring**.
9. In the **Custom flooring** pane, specify a flooring **Set name**. For example, "Flooring Material Set for Bedroom 3".
10. Under **Filter type**, choose **By room name**.
11. Under **Room name**, select a room name. For example, "Bedroom 3".
12. Under **Custom flooring**, choose **Add material** and then choose **Carpet**.
13. Choose **Save** to save the flooring set.

Using the AWS CLI

Example

You can use the following JSON in the `templateBody` as part of a call to `create-world-template`.

```
"Flooring": {
  "MaterialSets": [
    {
      "Name": "Flooring Material Set 1",
      "TargetSet": {
        "RoomTypes": [ "Bedroom", "Living", "Dining" ]
      },
      "SampleSet": {
        "MaterialTypes": [ "Floorboards" ]
      }
    },
    {
      "Name": "Flooring Material Set for Bedroom 3",
      "TargetSet": {
        "RoomNames": [ "Bedroom 3" ]
      },
      "SampleSet": {
        "MaterialTypes": [ "Carpet" ]
      }
    }
  ]
}
```

Specifying More and Less Furniture in Rooms

You can specify how densely furniture is spaced by room names or room types. By default, rooms are randomly furnished with moderate spacing. In the following example, all bedrooms are randomly furnished with dense spacings. The living room and dining room are furnished sparsely. All other rooms are furnished by default.

Using the console

1. In the **Simulation world template edit** screen, under **Interiors**, choose **Furniture**.
2. In the **Furniture** pane, choose **Add custom furniture**.
3. In the **Custom furniture** pane, specify a custom furniture **Set name**. For example, "Dense Furniture Arrangement".
4. Under **Filter type**, choose **By room type**.
5. Under **Room types**, select **Bedrooms**.
6. Toggle **Override furniture** to use default furniture.
7. Under **Furniture density**, choose **Dense**.
8. Choose **Save** to save the furniture set.
9. In the **Furniture** pane, choose **Add custom furniture**.
10. In the **Custom furniture** pane, specify a custom furniture **Set name**. For example, "Sparse Furniture Arrangement".
11. Under **Filter type**, choose **By room name**.
12. Under **Room names**, select the rooms you want to have sparse furniture density. For example, "My Living Room" and "My Dining Room".
13. Toggle **Override furniture** to use default furniture.
14. Under **Furniture density**, choose **Sparse**.
15. Choose **Save** to save the furniture set.

Using the AWS CLI

Example

You can use the following JSON in the `templateBody` as part of a call to `create-world-template`.

```
"Furniture": {
  "FurnitureArrangements": [
    {
      "Name": "Dense Furniture Arrangement",
      "TargetSet": {
        "RoomTypes": [ "Bedroom" ]
      },
      "DesiredSpatialDensity": "Dense"
    },
    {
      "Name": "Sparse Furniture Arrangement",
      "TargetSet": {
        "RoomNames": [ "My Living Room", "My Dining Room" ]
      },
      "DesiredSpatialDensity": "Sparse"
    }
  ]
}
```

Adding Specific Furniture Types to all Bedrooms and a Single Shared Living/Dining Room

You can specify the types of furniture for a room by room names or room types. In the following example, all bedrooms are moderately furnished with random beds, desks, dressers, and floor lamps. The room “My living/dining room” is densely furnished with random dining tables, dining chairs, floor lamps, sofas, and coffee tables. All other rooms are furnished by default.

Using the console

1. In the **Simulation world template edit** screen, under **Interiors**, choose **Furniture**.
2. In the **Furniture** pane, choose **Add custom furniture**.
3. In the **Custom furniture** pane, specify a custom furniture **Set name**. For example, "Bedroom Furniture".
4. Under **Filter type**, choose **By room type**.
5. Under **Room types**, select **Bedrooms**.
6. Ensure **Override furniture** is selected. If it is not selected, Simulation WorldForge will use default furniture.
7. Under **Furniture types**, choose **Add furniture** and then select **Beds**, **Desks**, **Dressers**, and **Floorlamps**.
8. Choose **Save** to save the furniture set.
9. In the **Furniture** pane, choose **Add custom furniture**.
10. In the **Custom furniture** pane, specify a custom furniture **Set name**. For example, "Living and Dining Furniture".
11. Under **Filter type**, choose **By room name**.
12. Under **Room names**, select a room. For example, "My living and dining room".
13. Ensure **Override furniture** is selected. If it is not selected, Simulation WorldForge will use default furniture.
14. Under **Furniture types**, choose **DiningTables**, **DiningChairs**, **FloorLamps**, **Sofas**, and **CoffeeTables**.
15. Under **Furniture density**, choose **Dense**.
16. Choose **Save** to save the furniture set.

Using the AWS CLI

Example

You can use the following JSON in the `templateBody` as part of a call to `create-world-template`.

```
"Furniture": {
  "FurnitureArrangements": [
    {
      "Name": "Bedroom Furniture",
      "TargetSet": {
        "RoomTypes": [ "Bedroom" ]
      },
      "SampleSet": {
        "ModelTypes": [
          "Beds",
          "Desks",
          "Dressers",
          "FloorLamps"
        ]
      }
    }
    {
      "Name": "Living and Dining Furniture",
      "TargetSet": {
        "RoomNames": [ "My living and dining room" ]
      },
      "SampleSet": {
        "ModelTypes": [
          "DiningTables",
          "DiningChairs",
          "FloorLamps",
          "Sofas",
          "CoffeeTables"
        ],
        "DesiredSpatialDensity": "Dense"
      }
    }
  ]
}
```

Specifying a Room without Furniture

Specify an empty list for the model set for the furnishing arrangement. All other rooms are furnished by default:

Using the console

1. In the **Simulation world template edit** screen, under **Interiors**, choose **Furniture**.
2. In the **Furniture** pane, choose **Add custom furniture**.
3. In the **Custom furniture** pane, specify a custom furniture **Set name**. For example, "No furniture".
4. Under **Filter type**, choose **By room name**.
5. Under **Room names**, select the rooms that you want to have no furniture. For example, "My Spare Room".
6. Ensure **Override furniture** is selected. If it is not selected, Simulation WorldForge will use default furniture.
7. Under **Furniture types**, make sure no types are chosen.
8. Choose **Save** to save the furniture set.

Using the AWS CLI

Example

You can use the following JSON in the `templateBody` as part of a call to `create-world-template`.

```
"Furniture": {
  "FurnitureArrangements": [
    {
      "Name": "No Furniture",
      "TargetSet": {
        "RoomNames": [ "My Spare Room" ]
      },
      "SampleSet": {
        "ModelTypes": []
        }
      }
    ]
}
```

JSON Schema for Simulation World Template Body

The `templateBody` (simulation world template body) is an input parameter of the CreateWorldTemplate (p. 309) operation. This parameter is a JSON-formatted string. The JSON specifies a simulation world template and contains the parameters Simulation WorldForge will use to generate worlds.

The following shows the schema for the different versions of the world template.

Version 2

The following is the template for the Version 2 schema

```
{
  "title": "WorldTemplate",
  "description": "The top-level template for parameterizing a randomly generated world. By
 default, a single\nresidential building with one floor and one room is generated.",
  "type": "object",
  "properties": {
    "Version": {
      "title": "Version",
      "type": "string"
    },
    "Buildings": {
      "title": "Buildings",
      "default": [
        {
          "Floors": [
            {
              "Floorplan": {
                "Footprint": {
                  "DesiredAspectRatio": {
                    "x": 1.0,
                    "y": 1.0
                  }
                },
                "Ceiling": {
                  "Height": 3.0
                },
                "Rooms": [
                  {
                    "Type": "Living",
                    "Name": "My_Living_Room",
                    "OriginalName": "My Living Room",
                    "DesiredShape": {
                      "Area": 20.0,
                      "AspectRatio": {
                        "x": 1.0,
                        "y": 1.0
                      }
                    }
                  }
                ],
                "DesiredConnections": []
              },
              "Interior": {
                "Doorways": {
                  "DoorwaySets": []
                },
                "Flooring": {
                  "MaterialSets": []
                },
                "Walls": {
                  "MaterialSets": []
                },
                "Furniture": {
                  "FurnitureArrangements": []
                }
              }
            }
          ]
        }
      ],
      "type": "array",
      "items": {
        "$ref": "#/definitions/BuildingTemplate"
      },
      "minItems": 1,
      "maxItems": 1
    }
  },
  "required": [
    "Version"
  ],
  "additionalProperties": false,
  "definitions": {
    "AspectRatio": {
      "title": "AspectRatio",
      "type": "object",
      "properties": {
        "x": {
          "title": "X",
          "default": 1,
          "minimum": 1,
          "maximum": 4,
          "type": "number"
        },
        "y": {
          "title": "Y",
          "default": 1,
          "minimum": 1,
          "maximum": 4,
          "type": "number"
        }
      },
      "additionalProperties": false
    },
    "FloorplanFootprint": {
      "title": "FloorplanFootprint",
      "description": "The desired footprint of this floorplan.",
      "type": "object",
      "properties": {
        "DesiredAspectRatio": {
          "title": "Desiredaspectratio",
          "default": {
            "x": 1.0,
            "y": 1.0
          },
          "allOf": [
            {
              "$ref": "#/definitions/AspectRatio"
            }
          ]
        }
      },
      "additionalProperties": false
    },
    "FloorplanCeiling": {
      "title": "FloorplanCeiling",
      "description": "The height of the ceiling for this floorplan in metres.",
      "type": "object",
      "properties": {
        "Height": {
          "title": "Height",
          "default": 3.0,
          "type": "number",
          "minimum": 2.4,
          "maximum": 4.0
        }
      },
      "additionalProperties": false
    },
    "Rectangle": {
      "title": "Rectangle",
      "description": "A rectangle defined by area in square metres and aspect ratio.",
      "type": "object",
      "properties": {
        "Area": {
          "title": "Area",
          "type": "number"
        },
        "AspectRatio": {
          "$ref": "#/definitions/AspectRatio"
        }
      },
      "required": [
        "Area",
        "AspectRatio"
      ],
      "additionalProperties": false
    },
    "FloorplanRoom": {
      "title": "FloorplanRoom",
      "description": "A description for single room for this floorplan.",
      "type": "object",
      "properties": {
        "Type": {
          "title": "Type",
          "enum": [
            "Bedroom",
            "Bathroom",
            "Living",
            "Dining",
            "Kitchen",
            "Hallway",
            "Closet"
          ],
          "type": "string"
        },
        "Name": {
          "title": "Name",
          "maxLength": 255,
          "minLength": 1,
          "pattern": "^[a-zA-Z0-9_\\- ]*$",
          "type": "string"
        },
        "OriginalName": {
          "title": "Originalname",
          "type": "string"
        },
        "DesiredShape": {
          "title": "Desiredshape",
          "default": {
            "Area": 20.0,
            "AspectRatio": {
              "x": 1.0,
              "y": 1.0
            }
          },
          "allOf": [
            {
              "$ref": "#/definitions/Rectangle"
            }
          ]
        }
      },
      "required": [
        "Type",
        "Name"
      ],
      "additionalProperties": false
    },
    "FloorplanConnection": {
      "title": "FloorplanConnection",
      "description": "Descibes the desired layout of the rooms and their adjacent rooms. A
 connection can be either a doorway or\nan open space without any walls. Two rooms cannot
 both share an interior doorway and an opening.\nThe same two rooms can have multiple
 doorways, up to a limit.",
      "type": "object",
      "properties": {
        "Location": {
          "title": "Location",
          "type": "array",
          "items": {
            "type": "string"
          },
          "minItems": 2,
          "maxItems": 2
        },
        "ConnectionType": {
          "title": "Connectiontype",
          "enum": [
            "Doorway",
            "Opening"
          ],
          "type": "string"
        }
      },
      "required": [
        "Location",
        "ConnectionType"
      ],
      "additionalProperties": false
    },
    "FloorplanTemplate": {
      "title": "FloorplanTemplate",
      "description": "The top-level floorplan template that parameterizes the randomly
 generated\narchitectural layout. By default, a residential floorplan with bedroom and
\nliving room are generated with a random doorway or opening connection.\n\nThe footprint
 contributes to the overall shape of the floor layout along\nwith rooms. The footprint
 shape is desired as it is a preference and not\nguaranteed.\n\nThe ceiling determines the
 height of the walls. There are minimum and maximum ceiling heights. The ceiling height
 is guaranteed.\n\nRooms are required. Each room has a desired shape. Together, the room
\nshapes and footprint determine floor layout. The room types contribute to\nthe layout
 and are used when randomly selecting furniture and materials for\nthe walls and floors.
\n\nDesiredConnections are optional. Two rooms are connected if they share a\nwall and
 doorway or adjacent without any wall aka \"opening\". All rooms are\nguaranteed to be
 connected randomly if they are not specified in the\nconnections list. Connections that
 are specified are _not_ guaranteed but\nwill be attempted as best-effort.",
      "type": "object",
      "properties": {
        "Footprint": {
          "title": "Footprint",
          "default": {
            "DesiredAspectRatio": {
              "x": 1.0,
              "y": 1.0
            }
          },
          "allOf": [
            {
              "$ref": "#/definitions/FloorplanFootprint"
            }
          ]
        },
        "Ceiling": {
          "title": "Ceiling",
          "default": {
            "Height": 3.0
          },
          "allOf": [
            {
              "$ref": "#/definitions/FloorplanCeiling"
            }
          ]
        },
        "Rooms": {
          "title": "Rooms",
          "default": [
            {
              "Type": "Living",
              "Name": "My_Living_Room",
              "OriginalName": "My Living Room",
              "DesiredShape": {
                "Area": 20.0,
                "AspectRatio": {
                  "x": 1.0,
                  "y": 1.0
                }
              }
            }
          ],
          "type": "array",
          "items": {
            "$ref": "#/definitions/FloorplanRoom"
          },
          "minItems": 1,
          "maxItems": 6
        },
        "DesiredConnections": {
          "title": "Desiredconnections",
          "default": [],
          "type": "array",
          "items": {
            "$ref": "#/definitions/FloorplanConnection"
          },
          "minItems": 0,
          "maxItems": 12
        }
      },
      "additionalProperties": false
    },
    "RoomNameList": {
      "title": "RoomNameList",
      "description": "The set of all rooms matching any of the listed room names.",
      "type": "object",
      "properties": {
        "RoomNames": {
          "title": "Roomnames",
          "type": "array",
          "items": {
            "type": "string"
          },
          "minItems": 1,
          "maxItems": 6
        }
      },
      "required": [
        "RoomNames"
      ],
      "additionalProperties": false
    },
    "RoomTypeList": {
      "title": "RoomTypeList",
      "description": "The set of all rooms matching any of the listed room types.",
      "type": "object",
      "properties": {
        "RoomTypes": {
          "title": "Roomtypes",
          "type": "array",
          "items": {
            "enum": [
              "Bedroom",
              "Bathroom",
              "Living",
              "Dining",
              "Kitchen",
              "Hallway",
              "Closet"
            ],
            "type": "string"
          },
          "minItems": 1,
          "maxItems": 7
        }
      },
      "required": [
        "RoomTypes"
      ],
      "additionalProperties": false
    },
    "RoomPairTargetFilter": {
      "title": "RoomPairTargetFilter",
      "description": "Defines a target set as a pair of rooms. The pairs are defined as the
 cross product of two lists\nFrom and To.",
      "type": "object",
      "properties": {
        "From": {
          "title": "From",
          "anyOf": [
            {
              "$ref": "#/definitions/RoomNameList"
            },
            {
              "$ref": "#/definitions/RoomTypeList"
            }
          ]
        },
        "To": {
          "title": "To",
          "anyOf": [
            {
              "$ref": "#/definitions/RoomNameList"
            },
            {
              "$ref": "#/definitions/RoomTypeList"
            }
          ]
        }
      },
      "required": [
        "From",
        "To"
      ],
      "additionalProperties": false
    },
    "DoorOpenPosition": {
      "title": "DoorOpenPosition",
      "description": "Defines the amount of openness of an InteriorDoor.\n\nThe range for
Percent is [0., 100.]",
      "type": "object",
      "properties": {
        "Percent": {
          "title": "Percent",
          "default": 100.0,
          "anyOf": [
            {
              "type": "number",
              "minimum": 0.0,
              "maximum": 100.0
            },
            {
              "const": "Random",
              "type": "string"
            }
          ]
        }
      },
      "additionalProperties": false
    },
    "DoorInitialState": {
      "title": "DoorInitialState",
      "description": "Defines the initial state for an InteriorDoor object\n\nOpenPosition
specifies how much the door should be open.",
      "type": "object",
      "properties": {
        "OpenPosition": {
          "title": "Openposition",
          "default": {
            "Percent": 100.0
          },
          "allOf": [
            {
              "$ref": "#/definitions/DoorOpenPosition"
            }
          ]
        }
      },
      "additionalProperties": false
    },
    "InteriorDoor": {
      "title": "InteriorDoor",
      "description": "Custom configuration for each Doorway Set.\n\nInitial State of doors
includes the ability to configure how much the door should be open in\npercent [0.,
100.]",
      "type": "object",
      "properties": {
        "InitialState": {
          "title": "Initialstate",
          "default": {
            "OpenPosition": {
              "Percent": 100.0
            }
          },
          "allOf": [
            {
              "$ref": "#/definitions/DoorInitialState"
            }
          ]
        }
      },
      "additionalProperties": false
    },
    "InteriorDoorwaySet": {
      "title": "InteriorDoorwaySet",
      "description": "A set of doors to randomly assign to a set of interior target
elements.\n\nThe target set determines *what room pairs* are receive the doors as
specified in `Door`.\nRooms may be targeted by room type or room name.\n\nThe Door
customizes the configuration for doors added in the specified target set.",
      "type": "object",
      "properties": {
        "Name": {
          "title": "Name",
          "maxLength": 255,
          "minLength": 1,
          "pattern": "^[a-zA-Z0-9_\\- ]*$",
          "type": "string"
        },
        "TargetSet": {
          "title": "Targetset",
          "anyOf": [
            {
              "const": "Target.All",
              "type": "string"
            },
            {
              "$ref": "#/definitions/RoomPairTargetFilter"
            }
          ]
        },
        "Door": {
          "title": "Door",
          "anyOf": [
            {
              "$ref": "#/definitions/InteriorDoor"
            },
            {
              "const": null
            }
          ]
        }
      },
      "required": [
        "Name",
        "TargetSet"
      ],
      "additionalProperties": false
    },
    "InteriorDoorways": {
      "title": "InteriorDoorways",
      "description": "Describes the interior template parameters for all doorways for this
 floorplan.\nAll doorways not explicitly targeted will have a random door assigned fully
 opened.",
      "type": "object",
      "properties": {
        "DoorwaySets": {
          "title": "Doorwaysets",
          "default": [],
          "type": "array",
          "items": {
            "$ref": "#/definitions/InteriorDoorwaySet"
          },
          "minItems": 0,
          "maxItems": 13
        }
      },
      "additionalProperties": false
    },
    "MaterialSetByMaterialType": {
      "title": "MaterialSetByMaterialType",
      "description": "The set of materials that match any of the material types listed.  An
 empty\nset is invalid since all targets require materials.",
      "type": "object",
      "properties": {
        "MaterialTypes": {
          "title": "Materialtypes",
          "type": "array",
          "items": {
            "type": "string"
          },
          "minItems": 1
        }
      },
      "required": [
        "MaterialTypes"
      ],
      "additionalProperties": false
    },
    "InteriorMaterialSet": {
      "title": "InteriorMaterialSet",
      "description": "A set of sample materials to randomly assign to a set of interior
 target elements.\n\nThe target set determines *what rooms* receive the materials in the
 sample\nset. The targets in a room are the walls and flooring. Rooms may be targeted
\nby room type or room name.\n\nThe sample set determines *what materials* to randomly
 select for the\ntarget rooms' walls and floors.\n\nThe sample set is optional and when not
 specified (null) materials are\nrandomly selected according to the room type for each room
 in the target\nset.\n\nA sample set with an empty material set is invalid since all wall
\nand flooring targets require materials.",
      "type": "object",
      "properties": {
        "Name": {
          "title": "Name",
          "maxLength": 255,
          "minLength": 1,
          "pattern": "^[a-zA-Z0-9_\\- ]*$",
          "type": "string"
        },
        "TargetSet": {
          "title": "Targetset",
          "anyOf": [
            {
              "const": "Target.All",
              "type": "string"
            },
            {
              "anyOf": [
                {
                  "$ref": "#/definitions/RoomNameList"
                },
                {
                  "$ref": "#/definitions/RoomTypeList"
                }
              ]
            }
          ]
        },
        "SampleSet": {
          "$ref": "#/definitions/MaterialSetByMaterialType"
        }
      },
      "required": [
        "Name",
        "TargetSet"
      ],
      "additionalProperties": false
    },
    "InteriorFlooring": {
      "title": "InteriorFlooring",
      "description": "Describes the interior template parameters for all floors for this 
floorplan.\nAll floors not explicitly targeted will have a random floor material assigned 
by room type.",
      "type": "object",
      "properties": {
        "MaterialSets": {
          "title": "Materialsets",
          "default": [],
          "type": "array",
          "items": {
            "$ref": "#/definitions/InteriorMaterialSet"
          },
          "minItems": 0,
          "maxItems": 6
        }
      },
      "additionalProperties": false
    },
    "InteriorWalls": {
      "title": "InteriorWalls",
      "description": "Describes the interior template parameters for all walls for this 
floorplan.\nAll walls not explicitly targeted will have a random wall material assigned by 
room type.",
      "type": "object",
      "properties": {
        "MaterialSets": {
          "title": "Materialsets",
          "default": [],
          "type": "array",
          "items": {
            "$ref": "#/definitions/InteriorMaterialSet"
          },
          "minItems": 0,
          "maxItems": 6
        }
      },
      "additionalProperties": false
    },
    "ModelTypeList": {
      "title": "ModelTypeList",
      "description": "The set of all models matching any of the listed model types.\nAn
 empty set means zero models to sample/select.",
      "type": "object",
      "properties": {
        "ModelTypes": {
          "title": "Modeltypes",
          "type": "array",
          "items": {
            "enum": [
              "Baths",
              "BarCabinets",
              "Beds",
              "Bookcases",
              "CoffeeTables",
              "ConsoleTables",
              "CornerCabinets",
              "DeskChairs",
              "Desks",
              "DiningChairs",
              "DiningTables",
              "DishWashers",
              "Dressers",
              "EndAndSideTables",
              "FloorLamps",
              "Fridges",
              "LivingRoomChairs",
              "KitchenIslandsAndCarts",
              "MediaStorage",
              "Nightstands",
              "Ottomans",
              "Ovens",
              "ServingCarts",
              "Showers",
              "SideboardsAndBuffets",
              "Sofas",
              "Storage",
              "StorageBenches",
              "Toilets",
              "VanityCounters",
              "WashingMachinesAndDryers"
            ],
            "type": "string"
          },
          "minItems": 0
        }
      },
      "required": [
        "ModelTypes"
      ],
      "additionalProperties": false
    },
    "FurnitureArrangementSet": {
      "title": "FurnitureArrangementSet",
      "description": "Describes the interior template for placing furniture in one or
 more rooms.\n\n- TargetSet is the set of rooms to furnish, filter by room name or room\n
  type.\n- SampleSet is a set of all furnishing models to randomly choose and\n  place.\n-
 DesiredSpatialDensity is the desired level of free space after placing\n  furniture.",
      "type": "object",
      "properties": {
        "Name": {
          "title": "Name",
          "maxLength": 255,
          "minLength": 1,
          "pattern": "^[a-zA-Z0-9_\\- ]*$",
          "type": "string"
        },
        "TargetSet": {
          "title": "Targetset",
          "anyOf": [
            {
              "const": "Target.All",
              "type": "string"
            },
            {
              "anyOf": [
                {
                  "$ref": "#/definitions/RoomNameList"
                },
                {
                  "$ref": "#/definitions/RoomTypeList"
                }
              ]
            }
          ]
        },
        "SampleSet": {
          "$ref": "#/definitions/ModelTypeList"
        },
        "DesiredSpatialDensity": {
          "title": "Desiredspatialdensity",
          "default": "Moderate",
          "enum": [
            "Sparse",
            "Moderate",
            "Dense"
          ],
          "type": "string"
        }
      },
      "required": [
        "Name",
        "TargetSet"
      ],
      "additionalProperties": false
    },
    "InteriorFurnishings": {
      "title": "InteriorFurnishings",
      "description": "Describes the types of furniture models for randomly placing into
 each room\nin the world.  Rooms are targeted by room type or room name. Rooms that are
\nnot targeted are furnished at random by their room type with moderate density.\ndensity.
 For an empty room, specify an empty sample set.",
      "type": "object",
      "properties": {
        "FurnitureArrangements": {
          "title": "Furniturearrangements",
          "default": [],
          "type": "array",
          "items": {
            "$ref": "#/definitions/FurnitureArrangementSet"
          },
          "minItems": 0,
          "maxItems": 6
        }
      },
      "additionalProperties": false
    },
    "InteriorTemplate": {
      "title": "InteriorTemplate",
      "description": "Top-level template for parameterizing the interior finishes and
 furnishings for\nthis floorplan.",
      "type": "object",
      "properties": {
        "Doorways": {
          "title": "Doorways",
          "default": {
            "DoorwaySets": []
          },
          "allOf": [
            {
              "$ref": "#/definitions/InteriorDoorways"
            }
          ]
        },
        "Flooring": {
          "title": "Flooring",
          "default": {
            "MaterialSets": []
          },
          "allOf": [
            {
              "$ref": "#/definitions/InteriorFlooring"
            }
          ]
        },
        "Walls": {
          "title": "Walls",
          "default": {
            "MaterialSets": []
          },
          "allOf": [
            {
              "$ref": "#/definitions/InteriorWalls"
            }
          ]
        },
        "Furniture": {
          "title": "Furniture",
          "default": {
            "FurnitureArrangements": []
          },
          "allOf": [
            {
              "$ref": "#/definitions/InteriorFurnishings"
            }
          ]
        }
      },
      "additionalProperties": false
    },
    "FloorTemplate": {
      "title": "FloorTemplate",
      "description": "Describes a single foor within a building. Defaults to a single
 residential room\nof a randomy type and size, and the interior is randomly furnished.",
      "type": "object",
      "properties": {
        "Floorplan": {
          "title": "Floorplan",
          "default": {
            "Footprint": {
              "DesiredAspectRatio": {
                "x": 1.0,
                "y": 1.0
              }
            },
            "Ceiling": {
              "Height": 3.0
            },
            "Rooms": [
              {
                "Type": "Living",
                "Name": "My_Living_Room",
                "OriginalName": "My Living Room",
                "DesiredShape": {
                  "Area": 20.0,
                  "AspectRatio": {
                    "x": 1.0,
                    "y": 1.0
                  }
                }
              }
            ],
            "DesiredConnections": []
          },
          "allOf": [
            {
              "$ref": "#/definitions/FloorplanTemplate"
            }
          ]
        },
        "Interior": {
          "title": "Interior",
          "default": {
            "Doorways": {
              "DoorwaySets": []
            },
            "Flooring": {
              "MaterialSets": []
            },
            "Walls": {
              "MaterialSets": []
            },
            "Furniture": {
              "FurnitureArrangements": []
            }
          },
          "allOf": [
            {
              "$ref": "#/definitions/InteriorTemplate"
            }
          ]
        }
      },
      "additionalProperties": false
    },
    "BuildingTemplate": {
      "title": "BuildingTemplate",
      "description": "Describes a building to be randomly generated. Defaults to one
residential floor.",
      "type": "object",
      "properties": {
        "Floors": {
          "title": "Floors",
          "default": [
            {
              "Floorplan": {
                "Footprint": {
                  "DesiredAspectRatio": {
                    "x": 1.0,
                    "y": 1.0
                  }
                },
                "Ceiling": {
                  "Height": 3.0
                },
                "Rooms": [
                  {
                    "Type": "Living",
                    "Name": "My_Living_Room",
                    "OriginalName": "My Living Room",
                    "DesiredShape": {
                      "Area": 20.0,
                      "AspectRatio": {
                        "x": 1.0,
                        "y": 1.0
                      }
                    }
                  }
                ],
                "DesiredConnections": []
              },
              "Interior": {
                "Doorways": {
                  "DoorwaySets": []
                },
                "Flooring": {
                  "MaterialSets": []
                },
                "Walls": {
                  "MaterialSets": []
                },
                "Furniture": {
                  "FurnitureArrangements": []
                }
              }
            }
          ],
          "type": "array",
          "items": {
            "$ref": "#/definitions/FloorTemplate"
          },
          "minItems": 1,
          "maxItems": 1
        }
      },
      "additionalProperties": false
    }
  }
}
```

Version 1

The following is the template for the Version 1 schema.

```
{
  "title": "WorldTemplate",
  "description": "The top-level template for parameterizing a randomly generated world. By
 default, a single\nresidential building with one floor and one room is generated.",
  "type": "object",
  "properties": {
    "Version": {
      "title": "Version",
      "default": "1",
      "type": "string"
    },
    "Buildings": {
      "title": "Buildings",
      "default": [
        {
          "Floors": [
            {
              "Floorplan": {
                "Footprint": {
                  "DesiredAspectRatio": {
                    "x": 1.0,
                    "y": 1.0
                  }
                },
                "Ceiling": {
                  "Height": 3.0
                },
                "Rooms": [
                  {
                    "Type": "Living",
                    "Name": "My Living Room",
                    "DesiredShape": {
                      "Area": 20.0,
                      "AspectRatio": {
                        "x": 1.0,
                        "y": 1.0
                      }
                    }
                  },
                  {
                    "Type": "Bedroom",
                    "Name": "My Bedroom",
                    "DesiredShape": {
                      "Area": 20.0,
                      "AspectRatio": {
                        "x": 1.0,
                        "y": 1.0
                      }
                    }
                  }
                ],
                "DesiredConnections": []
              },
              "Interior": {
                "Flooring": {
                  "MaterialSets": []
                },
                "Walls": {
                  "MaterialSets": []
                },
                "Furniture": {
                  "FurnitureArrangements": []
                }
              }
            }
          ]
        }
      ],
      "type": "array",
      "items": {
        "$ref": "#/definitions/BuildingTemplate"
      },
      "minItems": 1,
      "maxItems": 1
    }
  },
  "additionalProperties": false,
  "definitions": {
    "AspectRatio": {
      "title": "AspectRatio",
      "type": "object",
      "properties": {
        "x": {
          "title": "X",
          "default": 1,
          "minimum": 1,
          "maximum": 4,
          "type": "number"
        },
        "y": {
          "title": "Y",
          "default": 1,
          "minimum": 1,
          "maximum": 4,
          "type": "number"
        }
      },
      "additionalProperties": false
    },
    "FloorplanFootprint": {
      "title": "FloorplanFootprint",
      "description": "The desired footprint of this floorplan.",
      "type": "object",
      "properties": {
        "DesiredAspectRatio": {
          "title": "Desiredaspectratio",
          "default": {
            "x": 1.0,
            "y": 1.0
          },
          "allOf": [
            {
              "$ref": "#/definitions/AspectRatio"
            }
          ]
        }
      },
      "additionalProperties": false
    },
    "FloorplanCeiling": {
      "title": "FloorplanCeiling",
      "description": "The height of the ceiling for this floorplan in metres.",
      "type": "object",
      "properties": {
        "Height": {
          "title": "Height",
          "default": 3.0,
          "type": "number",
          "minimum": 2.4,
          "maximum": 4.0
        }
      },
      "additionalProperties": false
    },
    "Rectangle": {
      "title": "Rectangle",
      "description": "A rectangle defined by area in square metres and aspect ratio.",
      "type": "object",
      "properties": {
        "Area": {
          "title": "Area",
          "type": "number"
        },
        "AspectRatio": {
          "$ref": "#/definitions/AspectRatio"
        }
      },
      "required": [
        "Area",
        "AspectRatio"
      ],
      "additionalProperties": false
    },
    "FloorplanRoom": {
      "title": "FloorplanRoom",
      "description": "A description for single room for this floorplan.",
      "type": "object",
      "properties": {
        "Type": {
          "title": "Type",
          "enum": [
            "Bedroom",
            "Bathroom",
            "Living",
            "Dining",
            "Kitchen",
            "Hallway",
            "Closet"
          ],
          "type": "string"
        },
        "Name": {
          "title": "Name",
          "type": "string"
        },
        "DesiredShape": {
          "title": "Desiredshape",
          "default": {
            "Area": 20.0,
            "AspectRatio": {
              "x": 1.0,
              "y": 1.0
            }
          },
          "allOf": [
            {
              "$ref": "#/definitions/Rectangle"
            }
          ]
        }
      },
      "required": [
        "Type",
        "Name"
      ],
      "additionalProperties": false
    },
    "FloorplanConnection": {
      "title": "FloorplanConnection",
      "description": "Descibes the desired layout of the rooms and their adjacent rooms. A 
connection can be either a doorway or \nan open space without any walls. Two rooms cannot 
both share an interior doorway and an opening. \nThe same two rooms can have multiple 
doorways, up to a limit.",
      "type": "object",
      "properties": {
        "Location": {
          "title": "Location",
          "type": "array",
          "items": {
            "type": "string"
          },
          "minItems": 2,
          "maxItems": 2
        },
        "ConnectionType": {
          "title": "Connectiontype",
          "enum": [
            "Doorway",
            "Opening"
          ],
          "type": "string"
        }
      },
      "required": [
        "Location",
        "ConnectionType"
      ],
      "additionalProperties": false
    },
    "FloorplanTemplate": {
      "title": "FloorplanTemplate",
      "description": "The top-level floorplan template that parameterizes the randomly
 generated \narchitectural layout. By default, a residential floorplan with bedroom and
 \nliving room are generated with a random doorway or opening connection. \n\nThe footprint
 contributes to the overall shape of the floor layout along\nwith rooms. The footprint
 shape is desired as it is a preference and not\nguaranteed.\n\nThe ceiling determines the
 height of the walls. There are minimum and\nmaximum ceiling heights. The ceiling height
 is guaranteed.\n\nRooms are required. Each room has a desired shape. Together, the room
\nshapes and footprint determine floor layout. The room types contribute to\nthe layout
 and are used when randomly selecting furniture and materials for\nthe walls and floors.
\n\nDesiredConnections are optional. Two rooms are connected if they share a\nwall and
 doorway or adjacent without any wall aka \"opening\". All rooms are\nguaranteed to be
 connected randomly if they are not specified in the\nconnections list. Connections that
 are specified are _not_ guaranteed but\nwill be attempted as best-effort.",
      "type": "object",
      "properties": {
        "Footprint": {
          "title": "Footprint",
          "default": {
            "DesiredAspectRatio": {
              "x": 1.0,
              "y": 1.0
            }
          },
          "allOf": [
            {
              "$ref": "#/definitions/FloorplanFootprint"
            }
          ]
        },
        "Ceiling": {
          "title": "Ceiling",
          "default": {
            "Height": 3.0
          },
          "allOf": [
            {
              "$ref": "#/definitions/FloorplanCeiling"
            }
          ]
        },
        "Rooms": {
          "title": "Rooms",
          "default": [
            {
              "Type": "Living",
              "Name": "My Living Room",
              "DesiredShape": {
                "Area": 20.0,
                "AspectRatio": {
                  "x": 1.0,
                  "y": 1.0
                }
              }
            },
            {
              "Type": "Bedroom",
              "Name": "My Bedroom",
              "DesiredShape": {
                "Area": 20.0,
                "AspectRatio": {
                  "x": 1.0,
                  "y": 1.0
                }
              }
            }
          ],
          "type": "array",
          "items": {
            "$ref": "#/definitions/FloorplanRoom"
          },
          "minItems": 1,
          "maxItems": 6
        },
        "DesiredConnections": {
          "title": "Desiredconnections",
          "default": [],
          "type": "array",
          "items": {
            "$ref": "#/definitions/FloorplanConnection"
          },
          "minItems": 0,
          "maxItems": 12
        }
      },
      "additionalProperties": false
    },
    "RoomNameList": {
      "title": "RoomNameList",
      "description": "The set of all rooms matching any of the listed room names.",
      "type": "object",
      "properties": {
        "RoomNames": {
          "title": "Roomnames",
          "type": "array",
          "items": {
            "type": "string"
          }
        }
      },
      "required": [
        "RoomNames"
      ],
      "additionalProperties": false
    },
    "RoomTypeList": {
      "title": "RoomTypeList",
      "description": "The set of all rooms matching any of the listed room types.",
      "type": "object",
      "properties": {
        "RoomTypes": {
          "title": "Roomtypes",
          "type": "array",
          "items": {
            "enum": [
              "Bedroom",
              "Bathroom",
              "Living",
              "Dining",
              "Kitchen",
              "Hallway",
              "Closet"
            ],
            "type": "string"
          }
        }
      },
      "required": [
        "RoomTypes"
      ],
      "additionalProperties": false
    },
    "MaterialSetByMaterialType": {
      "title": "MaterialSetByMaterialType",
      "description": "The set of materials that match any of the material types listed.  An
empty\nset is invalid since all targets require materials.",
      "type": "object",
      "properties": {
        "MaterialTypes": {
          "title": "Materialtypes",
          "type": "array",
          "items": {
            "type": "string"
          },
          "minItems": 1
        }
      },
      "required": [
        "MaterialTypes"
      ],
      "additionalProperties": false
    },
    "InteriorMaterialSet": {
      "title": "InteriorMaterialSet",
      "description": "A set of sample materials to randomly assign to a set of interior
target elements.\n\nThe target set determines *what rooms* receive the materials in the
sample\nset. The targets in a room are the walls and flooring. Rooms may be targeted \nby
room type or room name. \n\nThe sample set determines *what materials* to randomly select
for the\ntarget rooms' walls and floors. \n\nThe sample set is optional and when not
specified (null) materials are\nrandomly selected according to the room type for each room
in the target\nset.\n\nA sample set with an empty material set is invalid since all wall
\nand flooring targets require materials.",
      "type": "object",
      "properties": {
        "Name": {
          "title": "Name",
          "type": "string"
        },
        "TargetSet": {
          "title": "Targetset",
          "anyOf": [
            {
              "$ref": "#/definitions/RoomNameList"
            },
            {
              "$ref": "#/definitions/RoomTypeList"
            }
          ]
        },
        "SampleSet": {
          "$ref": "#/definitions/MaterialSetByMaterialType"
        }
      },
      "required": [
        "Name",
        "TargetSet"
      ],
      "additionalProperties": false
    },
    "InteriorFlooring": {
      "title": "InteriorFlooring",
      "description": "Describes the interior template parameters for all floors for this
 floorplan.\nAll floors not explicitly targeted will have a random floor material assigned
 by room type.",
      "type": "object",
      "properties": {
        "MaterialSets": {
          "title": "Materialsets",
          "default": [],
          "type": "array",
          "items": {
            "$ref": "#/definitions/InteriorMaterialSet"
          },
          "minItems": 0,
          "maxItems": 6
        }
      },
      "additionalProperties": false
    },
    "InteriorWalls": {
      "title": "InteriorWalls",
      "description": "Describes the interior template parameters for all walls for this
 floorplan.\nAll walls not explicitly targeted will have a random wall material assigned by
 room type.",
      "type": "object",
      "properties": {
        "MaterialSets": {
          "title": "Materialsets",
          "default": [],
          "type": "array",
          "items": {
            "$ref": "#/definitions/InteriorMaterialSet"
          },
          "minItems": 0,
          "maxItems": 6
        }
      },
      "additionalProperties": false
    },
    "ModelTypeList": {
      "title": "ModelTypeList",
      "description": "The set of all models matching any of the listed model types.\nAn
 empty set means zero models to sample/select.",
      "type": "object",
      "properties": {
        "ModelTypes": {
          "title": "Modeltypes",
          "type": "array",
          "items": {
            "type": "string"
          },
          "minItems": 0
        }
      },
      "required": [
        "ModelTypes"
      ],
      "additionalProperties": false
    },
    "FurnitureArrangementSet": {
      "title": "FurnitureArrangementSet",
      "description": "Describes the interior template for placing furniture in one or more
 rooms.\n\n- TargetSet is the set of rooms to furnish, filter by room name or room\n
 type.\n- SampleSet is a set of all furnishing models to randomly choose and\n  place. \n-
 DesiredSpatialDensity is the desired level of free space after placing\n  furniture.",
      "type": "object",
      "properties": {
        "Name": {
          "title": "Name",
          "type": "string"
        },
        "TargetSet": {
          "title": "Targetset",
          "anyOf": [
            {
              "$ref": "#/definitions/RoomNameList"
            },
            {
              "$ref": "#/definitions/RoomTypeList"
            }
          ]
        },
        "SampleSet": {
          "$ref": "#/definitions/ModelTypeList"
        },
        "DesiredSpatialDensity": {
          "title": "Desiredspatialdensity",
          "default": "Moderate",
          "enum": [
            "Sparse",
            "Moderate",
            "Dense"
          ],
          "type": "string"
        }
      },
      "required": [
        "Name",
        "TargetSet"
      ],
      "additionalProperties": false
    },
    "InteriorFurnishings": {
      "title": "InteriorFurnishings",
      "description": "Describes the types of furniture models for randomly placing into
 each room\nin the world.  Rooms are targeted by room type or room name. Rooms that are
\nnot targeted are furnished at random by their room type with moderate density.\ndensity.
 For an empty room, specify an empty sample set.",
      "type": "object",
      "properties": {
        "FurnitureArrangements": {
          "title": "Furniturearrangements",
          "default": [],
          "type": "array",
          "items": {
            "$ref": "#/definitions/FurnitureArrangementSet"
          },
          "minItems": 0,
          "maxItems": 6
        }
      },
      "additionalProperties": false
    },
    "InteriorTemplate": {
      "title": "InteriorTemplate",
      "description": "Top-level template for parameterizing the interior finishes and
furnishings for\nthis floorplan.",
      "type": "object",
      "properties": {
        "Flooring": {
          "title": "Flooring",
          "default": {
            "MaterialSets": []
          },
          "allOf": [
            {
              "$ref": "#/definitions/InteriorFlooring"
            }
          ]
        },
        "Walls": {
          "title": "Walls",
          "default": {
            "MaterialSets": []
          },
          "allOf": [
            {
              "$ref": "#/definitions/InteriorWalls"
            }
          ]
        },
        "Furniture": {
          "title": "Furniture",
          "default": {
            "FurnitureArrangements": []
          },
          "allOf": [
            {
              "$ref": "#/definitions/InteriorFurnishings"
            }
          ]
        }
      },
      "additionalProperties": false
    },
    "FloorTemplate": {
      "title": "FloorTemplate",
      "description": "Describes a single foor within a building. Defaults to a single
residential room\nof a randomy type and size, and the interior is randomly furnished.",
      "type": "object",
      "properties": {
        "Floorplan": {
          "title": "Floorplan",
          "default": {
            "Footprint": {
              "DesiredAspectRatio": {
                "x": 1.0,
                "y": 1.0
              }
            },
            "Ceiling": {
              "Height": 3.0
            },
            "Rooms": [
              {
                "Type": "Living",
                "Name": "My Living Room",
                "DesiredShape": {
                  "Area": 20.0,
                  "AspectRatio": {
                    "x": 1.0,
                    "y": 1.0
                  }
                }
              },
              {
                "Type": "Bedroom",
                "Name": "My Bedroom",
                "DesiredShape": {
                  "Area": 20.0,
                  "AspectRatio": {
                    "x": 1.0,
                    "y": 1.0
                  }
                }
              }
            ],
            "DesiredConnections": []
          },
          "allOf": [
            {
              "$ref": "#/definitions/FloorplanTemplate"
            }
          ]
        },
        "Interior": {
          "title": "Interior",
          "default": {
            "Flooring": {
              "MaterialSets": []
            },
            "Walls": {
              "MaterialSets": []
            },
            "Furniture": {
              "FurnitureArrangements": []
            }
          },
          "allOf": [
            {
              "$ref": "#/definitions/InteriorTemplate"
            }
          ]
        }
      },
      "additionalProperties": false
    },
    "BuildingTemplate": {
      "title": "BuildingTemplate",
      "description": "Describes a building to be randomly generated. Defaults to one
residential floor.",
      "type": "object",
      "properties": {
        "Floors": {
          "title": "Floors",
          "default": [
            {
              "Floorplan": {
                "Footprint": {
                  "DesiredAspectRatio": {
                    "x": 1.0,
                    "y": 1.0
                  }
                },
                "Ceiling": {
                  "Height": 3.0
                },
                "Rooms": [
                  {
                    "Type": "Living",
                    "Name": "My Living Room",
                    "DesiredShape": {
                      "Area": 20.0,
                      "AspectRatio": {
                        "x": 1.0,
                        "y": 1.0
                      }
                    }
                  },
                  {
                    "Type": "Bedroom",
                    "Name": "My Bedroom",
                    "DesiredShape": {
                      "Area": 20.0,
                      "AspectRatio": {
                        "x": 1.0,
                        "y": 1.0
                      }
                    }
                  }
                ],
                "DesiredConnections": []
              },
              "Interior": {
                "Flooring": {
                  "MaterialSets": []
                },
                "Walls": {
                  "MaterialSets": []
                },
                "Furniture": {
                  "FurnitureArrangements": []
                }
              }
            }
          ],
          "type": "array",
          "items": {
            "$ref": "#/definitions/FloorTemplate"
          },
          "minItems": 1,
          "maxItems": 1
        }
      },
      "additionalProperties": false
    }
  }
}
```

Sample World Templates in JSON

The `templateBody` (simulation world template body) is an input parameter of the https://docs.aws.amazon.com/robomaker/latest/dg/API_CreateWorldTemplate.html. This parameter is a

JSON-formatted string. The JSON specifies a simulation world template and contains the parameters Simulation WorldForge will use to generate worlds.

This section contains sample simulation world template bodies.

Topics

One Bedroom House

The following example specifies a one bedroom house. It specifies interior materials and furniture.

```
{
  "name": "OneBedroomHouse",
  "templateBody": {
    "Version": "2",
    "Buildings": [
      {
        "Floors": [
          {
            "Floorplan": {
              "Footprint": {
                "DesiredAspectRatio": {
                  "x": 1,
                  "y": 1
                }
              },
              "Ceiling": {
                "Height": 3
              },
              "Rooms": [
                {
                  "Type": "Bedroom",
                  "Name": "Bedroom",
                  "DesiredShape": {
                    "Area": 25,
                    "AspectRatio": {
                      "x": 1,
                      "y": 1.2
                    }
                  }
                },
                {
                  "Type": "Living",
                  "Name": "Living room",
                  "DesiredShape": {
                    "Area": 30,
                    "AspectRatio": {
                      "x": 1,
                      "y": 1.5
                    }
                  }
                },
                {
                  "Type": "Bathroom",
                  "Name": "Bathroom",
                  "DesiredShape": {
                    "Area": 10,
                    "AspectRatio": {
```

```
                "x": 1,
                "y": 1.5
              }
            }
          },
          {
            "Type": "Kitchen",
            "Name": "Kitchen",
            "DesiredShape": {
              "Area": 15,
              "AspectRatio": {
                "x": 1.5,
                "y": 1
              }
            }
          }
        ],
        "DesiredConnections": [
          {
            "Location": [
              "Bathroom",
              "Living room"
            ],
            "ConnectionType": "Doorway"
          },
          {
            "Location": [
              "Living room",
              "Kitchen"
            ],
            "ConnectionType": "Opening"
          },
          {
            "Location": [
              "Bedroom",
              "Living room"
            ],
            "ConnectionType": "Doorway"
          }
        ]
      },
      "Interior": {
        "Flooring": {
          "MaterialSets": [
            {
              "Name": "Floorboard room types",
              "TargetSet": {
                "RoomTypes": [
                  "Kitchen"
                ]
              },
              "SampleSet": {
                "MaterialTypes": [
                  "Floorboards"
                ]
              }
            },
            {
              "Name": "Carpet room types",
              "TargetSet": {
                "RoomTypes": [
                  "Living",
                  "Bedroom"
                ]
              },
              "SampleSet": {
```

```
        "MaterialTypes": [
          "Carpet"
        ]
      }
    },
    {
      "Name": "Bathroom",
      "TargetSet": {
        "RoomNames": [
          "Bathroom"
        ]
      },
      "SampleSet": {
        "MaterialTypes": [
          "Parquetry"
        ]
      }
    }
  ]
},
"Walls": {
  "MaterialSets": [
    {
      "Name": "Brick room types",
      "TargetSet": {
        "RoomTypes": [
          "Living"
        ]
      },
      "SampleSet": {
        "MaterialTypes": [
          "Brick"
        ]
      }
    },
    {
      "Name": "Tiles room types",
      "TargetSet": {
        "RoomTypes": [
          "Bathroom"
        ]
      },
      "SampleSet": {
        "MaterialTypes": [
          "Tiles"
        ]
      }
    }
  ]
},
"Furniture": {
  "FurnitureArrangements": [
    {
      "Name": "Dense furniture room types",
      "TargetSet": {
        "RoomTypes": [
          "Living",
          "Bedroom",
          "Kitchen",
          "Bathroom"
        ]
      },
      "DesiredSpatialDensity": "Dense"
    }
  ]
}
```

```
            }
          }
        ]
      }
    ]
  }
}
```

One room only

The following example specifies a one bedroom house. It specifies interior furniture.

```
{
  "Version": "2",
  "Buildings": [
    {
      "Floors": [
        {
          "Floorplan": {
            "Footprint": {
              "DesiredAspectRatio": {
                "x": 1,
                "y": 1
              }
            },
            "Ceiling": {
              "Height": 3
            },
            "Rooms": [
              {
                "Type": "Bedroom",
                "Name": "Bedroom",
                "DesiredShape": {
                  "Area": 40,
                  "AspectRatio": {
                    "x": 1,
                    "y": 1.61
                  }
                }
              }
            ],
            "DesiredConnections": []
          },
          "Interior": {
            "Furniture": {
              "FurnitureArrangements": [
                {
                  "Name": "Bedroom furniture",
                  "TargetSet": {
                    "RoomNames": [
                      "Bedroom"
                    ]
                  },
                  "DesiredSpatialDensity": "Dense"
                }
              ]
            }
          }
        }
      ]
    }
  ]
}
```

Two rooms

The following example specifies a one bedroom house. Simulation WorldForge will determine details including floor material, wall material, furniture placement, and connectivity.

```
{
  "name": "TwoRooms",
  "templateBody": {
    "Version": "2",
    "Buildings": [
      {
        "Floors": [
          {
            "Floorplan": {
              "Footprint": {
                "DesiredAspectRatio": {
                  "x": 1,
                  "y": 1
                }
              },
              "Ceiling": {
                "Height": 3
              },
              "Rooms": [
                {
                  "Type": "Living",
                  "Name": "Living room",
                  "DesiredShape": {
                    "Area": 30,
                    "AspectRatio": {
                      "x": 1,
                      "y": 1.5
                    }
                  }
                },
                {
                  "Type": "Dining",
                  "Name": "Dining room",
                  "DesiredShape": {
                    "Area": 30,
                    "AspectRatio": {
                      "x": 1,
                      "y": 1.5
                    }
                  }
                }
              ],
              "DesiredConnections": []
            },
            "Interior": {}
          }
        ]
      }
    ]
  }
}
```

Managing Simulation World Templates

This section provides information about how you can create and manage simulation world templates. You use a simulation world template to specify how Simulation WorldForge generates worlds. You can specify the number of rooms, how they are connected, furniture, and the material types used for interior elements.

To learn more about simulation world templates, begin with Understanding Simulation World Templates (p. 85). You can also review the JSON `templateBody` that describes a simulation world template. For more information, see JSON Schema for Simulation World Template Body (p. 99).

Topics

Creating a Simulation World Template

Create a simulation world template to specify how Simulation WorldForge generates worlds. When your simulation world template is complete, create a world generation job to generate worlds with different room and interior configurations.

You can create a simulation world template from a sample template, a saved template, or from scratch. After the template is created, you can modify the floor plan, the interiors, and other details. For more information on modifying the simulation world template, see Modifying a Simulation World Template (p. 142).

To create a simulation world template

Follow the steps under one of the following tabs:

Using the console

To create a simulation world template

1. Sign in to the AWS RoboMaker console at https://console.aws.amazon.com/robomaker/.
2. On the AWS RoboMaker console, expand **Simulation WorldForge** on the left and then choose **World templates**.
3. On the **World templates** page, choose **Create template**.
4. On the **Create a world template** page, choose one of the template options. You can choose one of the preconfigured **Sample template**, clone and modify a **Saved template**, or **Start from scratch** with a default world.
5. On the **Template detail** page, in the upper left, choose **Rename** and then specify a name for the template.
6. (Optional) Customize the floor plan and interior details. For more information, see Understanding Simulation World Templates (p. 85).
7. On the **Template detail** page, choose **Save and exit**.

Using the AWS CLI

Example

You can update the simulation world template using the AWS CLI. First, create a JSON document that specifies the worlds Simulation WorldForge will generate. Next, use `create-world-template` to create the simulation world template.

For example, the following JSON document specifies a one bedroom house.

```
{
  "title": "WorldTemplate",
  "description": "The top-level template for parameterizing a randomly generated world.
 By default, a single\nresidential building with one floor and one room is generated.",
  "type": "object",
  "properties": {
    "Version": {
      "title": "Version",
      "default": "1",
      "type": "string"
    },
    "Buildings": {
      "title": "Buildings",
      "default": [
        {
          "Floors": [
            {
              "Floorplan": {
                "Footprint": {
                  "DesiredAspectRatio": {
                    "x": 1.0,
                    "y": 1.0
                  }
                },
                "Ceiling": {
                  "Height": 3.0
                },
                "Rooms": [
                  {
                    "Type": "Living",
                    "Name": "My Living Room",
                    "DesiredShape": {
                      "Area": 20.0,
                      "AspectRatio": {
                        "x": 1.0,
                        "y": 1.0
                      }
                    }
                  }
                ],
                "DesiredConnections": []
              },
              "Interior": {
                "Flooring": {
                  "MaterialSets": []
                },
                "Walls": {
                  "MaterialSets": []
                },
                "Furniture": {
                  "FurnitureArrangements": []
                }
              }
            }
          ]
        }
      ],
      "type": "array",
      "items": {
        "$ref": "#/definitions/BuildingTemplate"
      },
      "minItems": 1,
      "maxItems": 1
    }
  },
  "additionalProperties": false,
  "definitions": {
    "AspectRatio": {
      "title": "AspectRatio",
      "type": "object",
      "properties": {
        "x": {
          "title": "X",
          "default": 1,
          "minimum": 1,
          "maximum": 4,
          "type": "number"
        },
        "y": {
          "title": "Y",
          "default": 1,
          "minimum": 1,
          "maximum": 4,
          "type": "number"
        }
      },
      "additionalProperties": false
    },
    "FloorplanFootprint": {
      "title": "FloorplanFootprint",
      "description": "The desired footprint of this floorplan.",
      "type": "object",
      "properties": {
        "DesiredAspectRatio": {
          "title": "Desiredaspectratio",
          "default": {
            "x": 1.0,
            "y": 1.0
          },
          "allOf": [
            {
              "$ref": "#/definitions/AspectRatio"
            }
          ]
        }
      },
      "additionalProperties": false
    },
    "FloorplanCeiling": {
      "title": "FloorplanCeiling",
      "description": "The height of the ceiling for this floorplan in metres.",
      "type": "object",
      "properties": {
        "Height": {
          "title": "Height",
          "default": 3.0,
          "type": "number",
          "minimum": 2.4,
          "maximum": 4.0
        }
      },
      "additionalProperties": false
    },
    "Rectangle": {
      "title": "Rectangle",
      "description": "A rectangle defined by area in square metres and aspect ratio.",
      "type": "object",
      "properties": {
        "Area": {
          "title": "Area",
          "type": "number"
        },
        "AspectRatio": {
          "$ref": "#/definitions/AspectRatio"
        }
      },
      "required": [
        "Area",
        "AspectRatio"
      ],
      "additionalProperties": false
    },
    "FloorplanRoom": {
      "title": "FloorplanRoom",
      "description": "A description for single room for this floorplan.",
      "type": "object",
      "properties": {
        "Type": {
          "title": "Type",
          "enum": [
            "Bedroom",
            "Bathroom",
            "Living",
            "Dining",
            "Kitchen",
            "Hallway",
            "Closet"
          ],
          "type": "string"
        },
        "Name": {
          "title": "Name",
          "maxLength": 255,
          "minLength": 1,
          "pattern": "^[a-zA-Z0-9_\\- ]*$",
          "type": "string"
        },
        "DesiredShape": {
          "title": "Desiredshape",
          "default": {
            "Area": 20.0,
            "AspectRatio": {
              "x": 1.0,
              "y": 1.0
            }
          },
          "allOf": [
            {
              "$ref": "#/definitions/Rectangle"
            }
          ]
        }
      },
      "required": [
        "Type",
        "Name"
      ],
      "additionalProperties": false
    },
    "FloorplanConnection": {
      "title": "FloorplanConnection",
      "description": "Descibes the desired layout of the rooms and their adjacent
rooms. A connection can be either a doorway or \nan open space without any walls. Two
rooms cannot both share an interior doorway and an opening. \nThe same two rooms can
have multiple doorways, up to a limit.",
      "type": "object",
      "properties": {
        "Location": {
          "title": "Location",
          "type": "array",
          "items": {
            "type": "string"
          },
          "minItems": 2,
          "maxItems": 2
        },
        "ConnectionType": {
          "title": "Connectiontype",
          "enum": [
            "Doorway",
            "Opening"
          ],
          "type": "string"
        }
      },
      "required": [
        "Location",
        "ConnectionType"
      ],
      "additionalProperties": false
    },
    "FloorplanTemplate": {
      "title": "FloorplanTemplate",
      "description": "The top-level floorplan template that parameterizes the randomly
 generated \narchitectural layout. By default, a residential floorplan with bedroom
 and \nliving room are generated with a random doorway or opening connection. \n\nThe
 footprint contributes to the overall shape of the floor layout along\nwith rooms. The
 footprint shape is desired as it is a preference and not\nguaranteed.\n\nThe ceiling
 determines the height of the walls. There are minimum and\nmaximum ceiling heights.
 The ceiling height is guaranteed.\n\nRooms are required. Each room has a desired
 shape. Together, the room\nshapes and footprint determine floor layout. The room
 types contribute to\nthe layout and are used when randomly selecting furniture and
 materials for\nthe walls and floors.\n\nDesiredConnections are optional. Two rooms are
 connected if they share a\nwall and doorway or adjacent without any wall aka \"opening
\". All rooms are\nguaranteed to be connected randomly if they are not specified in the
\nconnections list. Connections that are specified are _not_ guaranteed but\nwill be
 attempted as best-effort.",
      "type": "object",
      "properties": {
        "Footprint": {
          "title": "Footprint",
          "default": {
            "DesiredAspectRatio": {
              "x": 1.0,
              "y": 1.0
            }
          },
          "allOf": [
            {
              "$ref": "#/definitions/FloorplanFootprint"
            }
          ]
        },
        "Ceiling": {
          "title": "Ceiling",
          "default": {
            "Height": 3.0
          },
          "allOf": [
            {
              "$ref": "#/definitions/FloorplanCeiling"
            }
          ]
        },
        "Rooms": {
          "title": "Rooms",
          "default": [
            {
              "Type": "Living",
              "Name": "My Living Room",
              "DesiredShape": {
                "Area": 20.0,
                "AspectRatio": {
                  "x": 1.0,
                  "y": 1.0
                }
              }
            }
          ],
          "type": "array",
          "items": {
            "$ref": "#/definitions/FloorplanRoom"
          },
          "minItems": 1,
          "maxItems": 6
        },
        "DesiredConnections": {
          "title": "Desiredconnections",
          "default": [],
          "type": "array",
          "items": {
            "$ref": "#/definitions/FloorplanConnection"
          },
          "minItems": 0,
          "maxItems": 12
        }
      },
      "additionalProperties": false
    },
    "RoomNameList": {
      "title": "RoomNameList",
      "description": "The set of all rooms matching any of the listed room names.",
      "type": "object",
      "properties": {
        "RoomNames": {
          "title": "Roomnames",
          "type": "array",
          "items": {
            "type": "string"
          },
          "minItems": 1,
          "maxItems": 6
        }
      },
      "required": [
        "RoomNames"
      ],
      "additionalProperties": false
    },
    "RoomTypeList": {
      "title": "RoomTypeList",
      "description": "The set of all rooms matching any of the listed room types.",
      "type": "object",
      "properties": {
        "RoomTypes": {
          "title": "Roomtypes",
          "type": "array",
          "items": {
            "enum": [
              "Bedroom",
              "Bathroom",
              "Living",
              "Dining",
              "Kitchen",
              "Hallway",
              "Closet"
            ],
            "type": "string"
          },
          "minItems": 1,
          "maxItems": 7
        }
      },
      "required": [
        "RoomTypes"
      ],
      "additionalProperties": false
    },
    "MaterialSetByMaterialType": {
      "title": "MaterialSetByMaterialType",
      "description": "The set of materials that match any of the material types listed.
 An empty\nset is invalid since all targets require materials.",
      "type": "object",
      "properties": {
        "MaterialTypes": {
          "title": "Materialtypes",
          "type": "array",
          "items": {
            "type": "string"
          },
          "minItems": 1
        }
      },
      "required": [
        "MaterialTypes"
      ],
      "additionalProperties": false
    },
    "InteriorMaterialSet": {
      "title": "InteriorMaterialSet",
      "description": "A set of sample materials to randomly assign to a set of interior
target elements.\n\nThe target set determines *what rooms* receive the materials
in the sample\nset. The targets in a room are the walls and flooring. Rooms may be
targeted\nby room type or room name.\n\nThe sample set determines *what materials* to
randomly select for the\ntarget rooms' walls and floors.\n\nThe sample set is optional
and when not specified (null) materials are\nrandomly selected according to the room
type for each room in the target\nset.\n\nA sample set with an empty material set is
invalid since all wall\nand flooring targets require materials.",
      "type": "object",
      "properties": {
        "Name": {
          "title": "Name",
          "maxLength": 255,
          "minLength": 1,
          "pattern": "^[a-zA-Z0-9_\\- ]*$",
          "type": "string"
        },
        "TargetSet": {
          "title": "Targetset",
          "anyOf": [
            {
              "$ref": "#/definitions/RoomNameList"
            },
            {
              "$ref": "#/definitions/RoomTypeList"
            }
          ]
        },
        "SampleSet": {
          "$ref": "#/definitions/MaterialSetByMaterialType"
        }
      },
      "required": [
        "Name",
        "TargetSet"
      ],
      "additionalProperties": false
    },
    "InteriorFlooring": {
      "title": "InteriorFlooring",
      "description": "Describes the interior template parameters for all floors for
 this floorplan.\nAll floors not explicitly targeted will have a random floor material
 assigned by room type.",
      "type": "object",
      "properties": {
        "MaterialSets": {
          "title": "Materialsets",
          "default": [],
          "type": "array",
          "items": {
            "$ref": "#/definitions/InteriorMaterialSet"
          },
          "minItems": 0,
          "maxItems": 6
        }
      },
      "additionalProperties": false
    },
    "InteriorWalls": {
      "title": "InteriorWalls",
      "description": "Describes the interior template parameters for all walls for
 this floorplan.\nAll walls not explicitly targeted will have a random wall material
 assigned by room type.",
      "type": "object",
      "properties": {
        "MaterialSets": {
          "title": "Materialsets",
          "default": [],
          "type": "array",
          "items": {
            "$ref": "#/definitions/InteriorMaterialSet"
          },
          "minItems": 0,
          "maxItems": 6
        }
      },
      "additionalProperties": false
    },
    "ModelTypeList": {
      "title": "ModelTypeList",
      "description": "The set of all models matching any of the listed model types.\nAn
 empty set means zero models to sample/select.",
      "type": "object",
      "properties": {
        "ModelTypes": {
          "title": "Modeltypes",
          "type": "array",
          "items": {
            "enum": [
              "Baths",
              "BarCabinets",
              "Beds",
              "Bookcases",
              "CoffeeTables",
              "ConsoleTables",
              "CornerCabinets",
              "DeskChairs",
              "Desks",
              "DiningChairs",
              "DiningTables",
              "DishWashers",
              "Dressers",
              "EndAndSideTables",
              "FloorLamps",
              "Fridges",
              "LivingRoomChairs",
              "KitchenIslandsAndCarts",
              "MediaStorage",
              "Nightstands",
              "Ottomans",
              "Ovens",
              "ServingCarts",
              "Showers",
              "SideboardsAndBuffets",
              "Sofas",
              "Storage",
              "StorageBenches",
              "Toilets",
              "VanityCounters",
              "WashingMachinesAndDryers"
            ],
            "type": "string"
          },
          "minItems": 0
        }
      },
      "required": [
        "ModelTypes"
      ],
      "additionalProperties": false
    },
    "FurnitureArrangementSet": {
      "title": "FurnitureArrangementSet",
      "description": "Describes the interior template for placing furniture in one
or more rooms.\n\n- TargetSet is the set of rooms to furnish, filter by room name or
room\n  type.\n- SampleSet is a set of all furnishing models to randomly choose and\n
 place.\n- DesiredSpatialDensity is the desired level of free space after placing\n
furniture.",
      "type": "object",
      "properties": {
        "Name": {
          "title": "Name",
          "maxLength": 255,
          "minLength": 1,
          "pattern": "^[a-zA-Z0-9_\\- ]*$",
          "type": "string"
        },
        "TargetSet": {
          "title": "Targetset",
          "anyOf": [
            {
              "$ref": "#/definitions/RoomNameList"
            },
            {
              "$ref": "#/definitions/RoomTypeList"
            }
          ]
        },
        "SampleSet": {
          "$ref": "#/definitions/ModelTypeList"
        },
        "DesiredSpatialDensity": {
          "title": "Desiredspatialdensity",
          "default": "Moderate",
          "enum": [
            "Sparse",
            "Moderate",
            "Dense"
          ],
          "type": "string"
        }
      },
      "required": [
        "Name",
        "TargetSet"
      ],
      "additionalProperties": false
    },
    "InteriorFurnishings": {
      "title": "InteriorFurnishings",
      "description": "Describes the types of furniture models for randomly placing into
 each room\nin the world.  Rooms are targeted by room type or room name. Rooms that
 are\nnot targeted are furnished at random by their room type with moderate density.
\ndensity. For an empty room, specify an empty sample set.",
      "type": "object",
      "properties": {
        "FurnitureArrangements": {
          "title": "Furniturearrangements",
          "default": [],
          "type": "array",
          "items": {
            "$ref": "#/definitions/FurnitureArrangementSet"
          },
          "minItems": 0,
          "maxItems": 6
        }
      },
      "additionalProperties": false
    },
    "InteriorTemplate": {
      "title": "InteriorTemplate",
      "description": "Top-level template for parameterizing the interior finishes and
 furnishings for\nthis floorplan.",
      "type": "object",
      "properties": {
        "Flooring": {
          "title": "Flooring",
          "default": {
            "MaterialSets": []
          },
          "allOf": [
            {
              "$ref": "#/definitions/InteriorFlooring"
            }
          ]
        },
        "Walls": {
          "title": "Walls",
          "default": {
            "MaterialSets": []
          },
          "allOf": [
            {
              "$ref": "#/definitions/InteriorWalls"
            }
          ]
        },
        "Furniture": {
          "title": "Furniture",
          "default": {
            "FurnitureArrangements": []
          },
          "allOf": [
            {
              "$ref": "#/definitions/InteriorFurnishings"
            }
          ]
        }
      },
      "additionalProperties": false
    },
    "FloorTemplate": {
      "title": "FloorTemplate",
      "description": "Describes a single foor within a building. Defaults to a
single residential room\nof a randomy type and size, and the interior is randomly
furnished.",
      "type": "object",
      "properties": {
        "Floorplan": {
          "title": "Floorplan",
          "default": {
            "Footprint": {
              "DesiredAspectRatio": {
                "x": 1.0,
                "y": 1.0
              }
            },
            "Ceiling": {
              "Height": 3.0
            },
            "Rooms": [
              {
                "Type": "Living",
                "Name": "My Living Room",
                "DesiredShape": {
                  "Area": 20.0,
                  "AspectRatio": {
                    "x": 1.0,
                    "y": 1.0
                  }
                }
              }
            ],
            "DesiredConnections": []
          },
          "allOf": [
            {
              "$ref": "#/definitions/FloorplanTemplate"
            }
          ]
        },
        "Interior": {
          "title": "Interior",
          "default": {
            "Flooring": {
              "MaterialSets": []
            },
            "Walls": {
              "MaterialSets": []
            },
            "Furniture": {
              "FurnitureArrangements": []
            }
          },
          "allOf": [
            {
              "$ref": "#/definitions/InteriorTemplate"
            }
          ]
        }
      },
      "additionalProperties": false
    },
    "BuildingTemplate": {
      "title": "BuildingTemplate",
      "description": "Describes a building to be randomly generated. Defaults to one
residential floor.",
      "type": "object",
      "properties": {
        "Floors": {
          "title": "Floors",
          "default": [
            {
              "Floorplan": {
                "Footprint": {
                  "DesiredAspectRatio": {
                    "x": 1.0,
                    "y": 1.0
                  }
                },
                "Ceiling": {
                  "Height": 3.0
                },
                "Rooms": [
                  {
                    "Type": "Living",
                    "Name": "My Living Room",
                    "DesiredShape": {
                      "Area": 20.0,
                      "AspectRatio": {
                        "x": 1.0,
                        "y": 1.0
                      }
                    }
                  }
                ],
                "DesiredConnections": []
              },
              "Interior": {
                "Flooring": {
                  "MaterialSets": []
                },
                "Walls": {
                  "MaterialSets": []
                },
                "Furniture": {
                  "FurnitureArrangements": []
                }
              }
            }
          ],
          "type": "array",
          "items": {
            "$ref": "#/definitions/FloorTemplate"
          },
          "minItems": 1,
          "maxItems": 1
        }
      },
      "additionalProperties": false
    }
  }
}
```

If you save the JSON into a file named `one-bedroom-house.json`, you can use it with the AWS CLI to create a simulation world template:

```
$ aws robomaker create-world-template --template my-simulation-world-template-arn --
template-body file://one-bedroom-house.json
```

Viewing a Simulation World Template

View details about a simulation world template.

To see the details of a simulation world template

Follow the steps on one of the following tabs:

Using the console

1. Sign in to the AWS RoboMaker console at https://console.aws.amazon.com/robomaker/.
2. In the left navigation pane, choose **Simulation WorldForge**, and then choose **World templates**.
3. Choose the **Id** of a simulation world template to view its details including its floor plan and interiors. You can also generate worlds from the detail view.

Using the AWS CLI

Example

The following AWS CLI example uses `list-world-templates` to list existing templates, and then it uses `describe-world-template` and `get-world-template-body` to view the details of a simulation world template.

```
$ aws robomaker list-world-templates
$ aws robomaker describe-world-template --template my-simulation-world-template-arn
$ aws robomaker get-world-template-body --template my-simulation-world-template-arn
```

Modifying a Simulation World Template

Select the floor plan to customize the number and types of rooms and the connections between rooms in the floor plan. Choose interiors to customize flooring, walls, and furniture.

To modify a simulation world template

Follow the steps on one of the following tabs:

Using the console

To modify the simulation world template

1. Sign in to the AWS RoboMaker console at https://console.aws.amazon.com/robomaker/.
2. On the AWS RoboMaker console, expand **Simulation WorldForge** in the left navigation pane and then select **World templates**.
3. On the **World templates** page, choose the simulation world template you want to modify.
4. Choose **Edit** or **Override** next to each element you want to modify. For more information about simulation world template components, see Understanding Simulation World Templates (p. 85).

Using the AWS CLI

Example

The following AWS CLI example uses `list-world-templates` to list existing templates, and then it uses `describe-world-template` to view the details of a simulation world template and `get-world-template-body` to retrieve the template body JSON and write it to a file.

```
$ aws robomaker list-world-templates
$ aws robomaker describe-world-template --template my-simulation-world-template-arn
$ aws robomaker get-world-template-body --template my-simulation-world-template-arn --
output json > myTemplateBody.json
$ aws robomaker update-world-template-body --template my-simulation-world-template-arn
 --template-body file://myTemplateBody.json
```

Deleting a Simulation World Template

When you no longer need a simulation world template, you can delete it.

Using the console

1. Sign in to the AWS RoboMaker console at https://console.aws.amazon.com/robomaker/.
2. In the left navigation pane, choose **Simulation WorldForge**, and then choose **World templates**.
3. Choose the **Id** of a simulation world template, choose **Template actions**, choose **Delete**, and then confirm the deletion by selecting **Delete** in the dialog box.

Using the AWS CLI

Example

The following AWS CLI example uses `list-world-templates` to list existing templates, and then it uses `delete-world-template` to delete a simulation world template.

```
$ aws robomaker list-world-templates
$ aws robomaker delete-world-template --template my-simulation-world-template-arn
```

Simulation World Template Versions, Features, and Changes

AWS RoboMaker Simulation WorldForge releases new versions of the world templates. You can use the new features and improvements in these templates to create worlds that are better suited for your use case.

To use all of the features of a world template, upgrade your world template to the latest version. The latest version of a world template has all of the features that are present in previous versions.

You can update your world templates using either the AWS RoboMaker console or the AWS CLI. If you're using the AWS RoboMaker console, you'll see a prompt that you can use to upgrade your template.

To upgrade your world template to the latest version using the API, set the `Version` field of the JSON that defines the world template to the numeric value for the latest version. For example, if Version 2 is the latest version, you would specify `"Version": "2"` in the body of the world template. To view the latest schema, see JSON Schema for Simulation World Template Body (p. 99).

The following descriptions provide information about features and updates for the world templates. The updates for the latest version are shown first.

Simulation World Template Version 2 Release

The updates for Version 2 include:

- The ability to add hinged doors to your worlds.
- The ability to apply a configuration to all rooms.
- A new field that describes your world.
- Changes to the floor friction values.
- Version agnostic updates.

Doors

You can use Version 2 of the AWS RoboMaker Simulation WorldForge template to create a world that has hinged doors.

You can configure the percentage that these doors are open. For example, these are some open states that you can specify:

- 0% open – closed
- 50% open – halfway open
- 70% open – mostly open
- 100% open – entirely open

You can also specify that Simulation WorldForge randomize the openness of the doors by setting the open percentage to a random state.

You can configure the doors that you want to see in your world under the `Interior` section of your world template. To learn how to use a world template to create a room with doors, see Requesting Doors in Doorways (p. 91).

Applying a configuration to all rooms

You can use the `Target.All` keyword of the world template to apply a configuration change to all rooms. These are some of the things that you can change in all of the rooms:

- Flooring material
- Wall material
- Doorways
- Furniture arrangements

For example, if you want to specify that every door is closed in your world template, you can specify that the doors are zero percent open and use the `Target.All` keyword to apply that condition to all doors. For more information, see Applying a Configuration to All Rooms (p. 90).

A new field that describes your worlds

The worlds that are created with a Version 2 template have a `world_description.json` file. This file appears in the same directory as the Gazebo Worldforge `.world` file.

The `world_description.json` file lists all the doors in your Simulation WorldForge world. You can use the DescribeWorld (p. 362) operation to see a description of your world. The description is the value of the `worldDescriptionBody` field. If your world was created with a Version 1 template, the value of the field is empty.

Version 2 changes to the floor friction values

In Version 2, the floors have the same floor friction values as the Gazebo ground plane. The floor friction values in Version 1 are unchanged.

Version agnostic updates

For all world templates, the spaces in the room names are replaced with underscores in your Gazebo model names. This change gives you the ability to use ROS topics for all your Simulation WorldForge Gazebo models. You can use ROS topics to get information about your model, or make changes to your model.

Managing World Generation Jobs

Use a world generation job to generate worlds from a simulation world template. When you create a world generation job, you specify the number of different floor plans and interior configurations. You can generate up to 50 worlds per world generation job.

Topics

Creating a World Generation Job

Create a world generation job to generate worlds with different room and interior configurations. Each world generation job can generate up to 50 worlds.

To create a world generation job

Follow the steps on one of the following tabs:

Using the console

To create a simulation world template

1. Sign in to the AWS RoboMaker console at https://console.aws.amazon.com/robomaker/.
2. On the AWS RoboMaker console, expand **Simulation WorldForge** on the left and then choose **World templates**.
3. On the **World templates** page, choose the simulation world template you want to use to generate worlds, and then choose **Generate worlds**.
4. On the **Generate worlds** page, specify the **Number of floor plans**. The number of floor plans multiplied by the number of interior variations per floor plan must not exceed 50.
5. Specify the number of **Interior variations per floor plan**. The number of floor plans multiplied by the number of interior variations per floor plan must not exceed 50.
6. *Optional:* Add **World tags** that will be assigned to all of the worlds you generate.
7. *Optional:* Add **Generation job tags** that will be assigned to the generation job. These tags will not apply to worlds you generate.
8. Choose **Generate**.

   You can track the progress of your world generation job in the **World generation detail** page. The time required to generate your worlds depends on the complexity of the simulation world template and the number of worlds you are generating.

Using the AWS CLI

Example

You can generate worlds from a simulation world template using the AWS CLI. Use `create-world-generation-job` to create the world generation job.

The following AWS CLI example shows how to generate 4 worlds with 2 floor plans with 2 different interior floor plans.

```
$ aws robomaker list-world-templates
$ aws robomaker create-world-generation-job --template my-simulation-world-template-arn
 --worldCount floorplanCount=2,interiorCountPerFloorplan=2
$ aws robomaker list-world-generation-jobs
$ aws robomaker describe-world-generation-job --job my-world-generation-job-arn
```

Viewing a World Generation Job

You can view world generation progress, summary information, and other details about a world generation job.

To see the details of a world generation job

Follow the steps on one of the following tabs:

Using the console

1. Sign in to the AWS RoboMaker console at https://console.aws.amazon.com/robomaker/.
2. In the left navigation pane, choose **Simulation WorldForge**, then choose **World templates**.

3. Choose the **Id** of a world generation job to view its details. You can find generation jobs using the search bar.

Using the AWS CLI

Example

The following AWS CLI example uses the `list-world-generation-jobs` to list existing world generation jobs, and then it uses `describe-world-generation-job` to view the details of a specific world generation job.

```
$ aws robomaker list-world-generation-jobs
$ aws robomaker describe-world-generation-job --job my-world-generation-job-arn
```

Cancelling a World Generation Job

You can cancel a world generation job that is in progress.

To cancel a world generation job

Follow the steps under one of the following tabs:

Using the console

1. Sign in to the AWS RoboMaker console at https://console.aws.amazon.com/robomaker/.
2. In the left navigation pane, choose **Simulation WorldForge**, and then choose **Generation jobs**.
3. On the **Generation jobs** page, choose the world generation job you want to cancel.
4. Choose **Cancel**. On the **Cancel generation job** page, choose **Cancel job** to cancel the job.

Using the AWS CLI

Example

The following AWS CLI example uses the `list-world-generation-jobs` to list existing world generation jobs, and then it uses `cancel-world-generation-job` to cancel a specific world generation job.

```
$ aws robomaker list-world-generation-jobs
$ aws robomaker cancel-world-generation-job --job my-world-generation-job-arn
```

Managing World Export Jobs

You can export worlds generated by Simulation WorldForge to use in your own environment. Worlds are exported to your Amazon S3 bucket in a .zip file. The .zip file includes Gazebo assets and an ROS workspace for the worlds.

Topics

Creating a World Export Job

You can select worlds to export to your Amazon S3 bucket. All worlds selected for the export will be in a single .zip file.

To create a world export job

Follow the steps on one of the following tabs:

Using the console

You can export one world per export job.

To create a simulation world template

1. Sign in to the AWS RoboMaker console at https://console.aws.amazon.com/robomaker/.
2. On the AWS RoboMaker console, expand **Simulation WorldForge** in the left navigation pane, and then choose **Worlds**.
3. On the **Worlds** page, choose **Create export job**.
4. On the **Create export job** page, choose a **World** to export.
5. Choose an **IAM role** with `PutObject`, `GetObject` and `AbortMultipartUpload` permissions to your Amazon S3 bucket. Choose **Create** to have a role with appropriate permissions created for you.
6. Choose an **S3 destination for worlds output**. You can also create a new Amazon S3 bucket by choosing **Create new S3 bucket** near the bottom of the page.
7. *Optional:* On the **Create export job** page, add tags that will be assigned to the exported world.
8. Choose **Create** to create the world export job.

   You can track its progress of the export job in the world export job details page. You are taken there automatically after you create the job.

Using the AWS CLI

Example

You can export worlds using the AWS CLI. Use `create-world-export-job` to create the world export job. You can export one world per export job.

The following AWS CLI example shows how to export a world. First, you can list worlds using `list-worlds`, and then call `create-world-export-job` specifying a world arn. You can check status by calling `list-world-export-jobs` and `describe-world-export-job`.

```
$ aws robomaker list-worlds
$ aws robomaker create-world-export-job --worlds my-simulation-world-arn --iam-role my-
iam-role-arn --outputLocation s3Bucket=my-bucket,s3prefix=prefix
$ aws robomaker list-world-export-jobs
$ aws robomaker describe-world-export-job --job my-world-export-job-arn
```

Viewing a World Export Job

View the status and other details of a world export job.

To see the details of a world export job

Follow the steps on one of the following tabs:

Using the console

1. Sign in to the AWS RoboMaker console at https://console.aws.amazon.com/robomaker/.
2. In the left navigation pane, choose **Simulation WorldForge**, and then choose **Export jobs**.
3. Choose the **ID** of a world export job to view its details. You can also search and cancel world export jobs.

Using the AWS CLI

Example

The following AWS CLI example uses the `list-world-export-jobs` to list existing world export jobs, and then it uses `describe-world-export-job` to view the details of a specific world export job.

```
$ aws robomaker list-world-export-jobs
$ aws robomaker describe-world-export-job --job my-world-export-job-arn
```

Managing Generated Worlds

This section provides information about how you can manage generated worlds.

Exporting a Generated World

You can use a world export job to export worlds generated by Simulation WorldForge. For more information about creating a world export job to export worlds, see Creating a World Export Job (p. 148).

Viewing a Generated World

You can view the world status, the id of the simulation world template used to generate the world, and other details. You can view images of the world in the console.

To see the details of a world generation job

Follow the steps on one of the following tabs:

Using the console

To view images of worlds in the console, you should have permissions to call `DescribeWorldForgeImageRedirect` in the AWS RoboMaker API. For more information, see Permissions Required to View Worlds in the AWS RoboMaker in the Console (p. 197).

1. Sign in to the AWS RoboMaker console at https://console.aws.amazon.com/robomaker/.
2. In the left navigation pane, choose **Simulation WorldForge**, and then choose **Worlds**.
3. Choose the **Id** of a world to view its details. You can use the preview thumbnails to find worlds. You can also search by tags, IDs, or templates.

4. On the **World details** page, you can review details including a larger preview image. You can also export or delete the world.

Using the AWS CLI

Example

The following AWS CLI example uses `list-worlds` to list existing worlds, and then it uses `describe-world` to view the details of a specific world.

```
$ aws robomaker list-worlds
$ aws robomaker describe-world --world my-world-arn
```

Deleting a Generated World

When you no longer need a world, you can delete it.

Using the console

1. Sign in to the AWS RoboMaker console at https://console.aws.amazon.com/robomaker/.
2. In the left navigation pane, choose **Simulation WorldForge**, and then choose **Worlds**.
3. Choose the worlds you want to delete. You can use the preview thumbnails to find worlds. You can also search by tags, IDs, or templates.
4. Choose **Delete**, and then choose **Delete** on the **Delete world** page.

Using the AWS CLI

Example

The following AWS CLI example uses `list-worlds` to list existing worlds, and then it uses `batch-delete-worlds` to delete two worlds.

```
$ aws robomaker list-worlds
$ aws robomaker batch-delete-worlds --worlds my-world-arn1,my-world-arn2
```

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

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. 287)). 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 \
--iam-role my-role-arn \
--failure-behavior Continue \
--robot-applications application='my-robot-application-
arn,launchConfig={packageName="hello_world_robot",launchFile="rotate.launch"}' \
--simulation-applications application='my-simulation-application-
arn,launchConfig={packageName="hello_world_simulation",launchFile="empty_world.launch"}'
 \
--vpc-config assignPublicIp=true,subnets=comma-separated-vpc-subnet-
ids,securityGroups=comma-separated-security-group-ids
```

  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.

Configuring `SimulationJob` Compute

To use GPU in your `SimulationJobs`, you can configure the `ComputeType` of the `SimulationJob` to use GPU Compute.

You realize the following benefits when using Graphics Processing Unit (GPU)-based simulation jobs in AWS RoboMaker.

- GPU-based simulation jobs allow execution of applications that require GPU-enabled sensor plugins and high fidelity rendering and performance using OpenGL, CUDA, OpenCL, and Vulkan.
- GPU-based simulation jobs ensure that the AWS RoboMaker GUI tools have high-quality HD resolution so you can see objects in greater detail. The GUI tools experience is ideal because the GPU ensures a higher rate of frames per second.
- GPU-based simulation accelerates the simulation job completion time. With GPU, you can run complex simulation scenes without taking a performance hit on the real-time factor and the frames per second.
- GPU-based simulation jobs improve the training of reinforcement learning models.

To configure a simulation job with GPU

The `Compute` parameter in the `CreateSimulationJob` request can be used to configure which kind of Compute is needed for the `SimulationJob`.

ComputeType

`ComputeType` specifies the type of Compute required for the job. Valid values are `CPU` and `GPU_AND_CPU`. The default value is `CPU`. If `GPU_AND_CPU` is specified, the job created can use GPU along with CPU.

GpuUnitLimit

Using the `GpuUnitLimit` parameter, you can specify the number of GPU units that need to be allocated to your job. For `GPU_AND_CPU` ComputeType, it must be 1. For `CPU` ComputeType, it must be 0.

> **Note**
> `GPU_AND_CPU` ComputeType cannot be used for jobs using a `colcon` bundled application. It can only be used with jobs that use robot and simulation applications created with Amazon ECR images.

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

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. 166) 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

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:

```
<package>
   ...
   <exec_depend>rosbridge_suite</exec_depend>
</package>
```

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

```
<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:

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

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

```
<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`.

```
#!/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.

```
#!/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)
    rospy.loginfo('Webserver at port {}'.format(port))
    httpd.serve_forever()

def main():

    rospy.init_node("webserver")
    if len(sys.argv) > 1:
        server_port = sys.argv[1]

    start_server(server_port)
    rospy.spin()

if __name__ == '__main__':
    main()
```

5. 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 applicatoion 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.

Configuring Custom Uploads

When you want to capture output files and other artifacts from your simulation job, you can configure custom uploads. You can configure custom uploads for your robot application and your simulation application. When you configure a custom upload, files you specify are uploaded from the simulation job to the Amazon S3 simulation output location you provide. This can be useful when you want to review or analyze application output generated during a simulation run or reuse artifacts.

Before you can configure custom uploads, you must provide an Amazon S3 output destination for your simulation job. AWS RoboMaker will upload matching files to a folder using a name you specify.

Matching files can be uploaded when all of the simulation job tools terminate or uploaded as they are produced and then removed.

Default upload configurations are automatically added to your custom upload configurations unless you turn them off. The default upload configuration uploads ROS and Gazebo default logging output. This maintains compatibility with past simulation job output configurations. which uploaded ROS and Gazebo default logging output. You can turn off the default upload configuration when you configure a simulation job in the console. You can also turn it off by setting `useDefaultUploadConfigurations` to `false` in the API `CreateSimulationJob`.

Your simulation applications are extraded onto a single 128gb partition. You have write access to the partition.

Adding a Custom Upload Configuration

To create a custom upload configuration, you need to specify a prefix that specifies where the files will be uploaded to in Amazon S3, a Unix glob path specifying the files to upload, and an upload behavior specifying when the files are uploaded.

Name

A prefix that specifies how files will be uploaded in Amazon S3. It is appended to the simulation output location to determine the final path.

For example, if your simulation output location is `s3://my-bucket` and your upload configuration name is `robot-test`, your files will be uploaded to `s3://my-bucket/<simid>/<runid>/robot-test`.

Path

The path specifies which files are uploaded. Standard Unix glob matching rules are accepted subject to the following:

- The path must begin with `/home/robomaker/` or `/var/log`.
- The path must not contain a reverse path expression (`/..`).
- Symbolic links are not followed.
- You can use `**` as a *super asterisk* in your path. For example, specifying `/var/log/**.log` causes all `.log` files in the `/var/log` directory tree to be collected.

  You can also use the standard asterisk as a standard wildcard. For example, `/var/log/system.log*` matches files such as `system.log_1111`, `system.log_2222`, and so on in `/var/log`.

Upload Behavior

You can select one of the following upload behaviors:

- **Upload on terminate** (`UPLOAD_ON_TERMINATE`) uploads all files matching the path once the simulation job enters the terminating state. AWS RoboMaker will attempt to upload logs for a maximum of 60 minutes.

  AWS RoboMaker does not begin uploading files until all of your tools running in the simulation have stopped.
- **Upload rolling with auto remove** (`UPLOAD_ROLLING_AUTO_REMOVE`) uploads all files matching the path as they are generated. Paths are checked every 5 seconds. When the files are uploaded, the source files are deleted. Once a file is deleted, if a new file is generated with the same name, it will

replace the previously uploaded file. AWS RoboMaker performs a final check for files once all of your applications running in the simulation have stopped.

Upload rolling with auto remove is useful for uploading rolling logs. Write or stream output to an "active" file which is not covered by the path glob. Once you're done writing to the active file, roll the file into a location covered by the path glob to be uploaded and removed.

This setting can help you conserve space in your simulation job. It can also help you access files before your simulation job terminates.

The simulation job partition size is 128gb. If your simulation job ends for any reason, AWS RoboMaker will try to upload all files specified in your custom upload configuration.

For more information about information captured from a running simulation job, see Accessing Simulation Job Data (p. 156).

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
- sysfs
- _sysctl
- umount
- umount2
- unshare
- uselib
- userfaultfd
- ustat
- vm86
- vm86old

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. 224).

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. 200).

> **Note**
> This topic applies to ROS Melodic.

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. 195).

```
robomaker@9cc6d11dfa46:~$rosservice 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:

```
robomaker@9cc6d11dfa46:~$rosservice 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:

```
robomaker@9cc6d11dfa46:~$rosservice 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:

```
#!/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</deped>
```

```
#
# 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)
```

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.

By default, AWS RoboMaker does not record ROS messages. No ROS bags will be generated.

Topics

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>
```

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.

```
<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.

```
<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`.

```
<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.

```
<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.

```
<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`.

```
<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.

```
<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.

```
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.

```
<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 last for 100 seconds.

```
BAGS="/opt/robomaker/datasources/bags/myS3prefix/log_0.bag /opt/robomaker/datasources/
bags/myS3prefix/log_1.bag"
ARGS="--start 10 --duration 100"
```

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`.

```
<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. 164).

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. 159).

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. 195).

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.

```
robomaker@9cc6d11dfa46:~$rosservice call /robomaker/job/cancel
success: True
message: ''
```

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`.

   ```
   <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.

   ```
   #!/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")
        add_tags([Tag(key="status", value="timeout")])
        cancel_job()

if __name__ == "__main__":
    rospy.init_node('cancel_node')
    # wait for the required services to become available
    rospy.wait_for_service('/robomaker/job/cancel')
    rospy.wait_for_service('/robomaker/job/add_tags')
    # tag the simulation job with a descriptive name
    add_tags(key="name", value="my_test")
    # add a timer to tag status as timeout and cancel the job after 5 minutes
    rospy.Timer(rospy.Duration(secs=300), cancel_timeout, oneshot=True)
    # add a subscriber to tag status as pass and cancel the job
    clock = rospy.Subscriber('/clock', Clock, check_complete)
    # keep the node alive
    rospy.spin()
```

Managing Simulation Jobs

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

Topics

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. 161).

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

Topics

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

1. Sign in to the AWS RoboMaker console at https://console.aws.amazon.com/robomaker/.
2. In the left navigation pane, choose **Simulation run**, 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 **Compute**, select a simulation unit limit. Your simulation is allocated CPU and memory proportional to the supplied simulation unit limit. A simulation unit is 1 vcpu and 2GB of memory. The default is 15.
8. *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.
9. *Optional:* In **Networking**, if your robot application or simulation application accesses resources on an Amazon VPC, select the **VPC**, subnets and security groups. Select all available subnets to ensure all of your resource limits are available. For more information, see VPCs and Subnets.

   If you want to access the simulation job from outside of the VPC, select **Assign public IP**.
10. *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. 153).

11. 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*.
12. Choose **Next**.
13. 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.
14. Type a **name** for the robot application.
15. 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. 76).

    Using `$LATEST` doesn't protect you from changes in Amazon S3. When AWS RoboMaker access the file, it will set to read only. For more informatiom about versioning in AWS RoboMaker, see Application Versioning (p. 19).
16. In **Robot application configuration**, provide the roslaunch **Launch package name** for your robot application.
17. 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.
18. *Optional:* To configure robot application tools, expand **Robot application tools**. Select **Use default tools** to use preconfigured tools. Select **Customize tools** to add, remove or edit custom tools to use with application.

    To add a new custom tool:

    a. Select **Add tool**.
    b. On the **Add application tool**, specify a **Tool name**.
    c. Specify the command-line arguments for the tool. You must include the tool executable name.
    d. Choose an **Exit behavior**. If you select **Fail**, the simulation job will fail if the tool exits. Select **Restart** to restart the tool. The default is **Restart**.
    e. Choose to enable or disable UI streaming. UI streaming is disabled by default.
    f. Select **Send output to CloudWatch** to record logs for the tool. The logs will be available in CloudWatch. Output is not sent to CloudWatch by default.

    Custom tools will start only after the main ROS launch process has started. For more information about custom tools, see Configuring Custom Simulation Tools (p. 179).
19. *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.
20. *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.

21. *Optional:* Configure traffic forwarding from the simulation job port to the application port. Simulation job networking must be configured in order to specify port mapping for your robot and simulation applications.
22. *Optional:* Specify one or more **Robot application upload configurations**. A simulation job output destination must be configured in order to specify upload configurations. Each configuration specifies an upload behavior, a Unix glob file matching rule, and a location to place matching files.

    Default upload configurations maintain backwards compatibility with past simulation job output configurations. The default configurations will be added to additional upload configurations you create.

    For more information about custom uploads, see Configuring Custom Uploads (p. 156).
23. Choose **Next**.
24. 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.
25. Type a **name** for the simulation application.
26. Select the **Simulation software suite** and **Simulation rendering version** used by your simulation application.
27. 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.

    Using `$LATEST` doesn't protect you from changes in Amazon S3. When AWS RoboMaker access the file, it will set to read only. For more informatiom about versioning in AWS RoboMaker, see Application Versioning (p. 19).
28. In **Simulation application configuration**, provide the roslaunch **Launch package name** and the roslaunch **Launch file** for your simulation application.
29. *Optional:* To configure robot application tools, expand **Simulation application tools**. Select **Use default tools** to use preconfigured tools. Select **Customize tools** to add, remove or edit custom tools to use with application.

    To add a new custom tool:

    a. Select **Add tool**.
    b. On the **Add application tool**, specify a **Tool name**.
    c. Specify the command-line arguments for the tool. You must include the tool executable name.
    d. Choose an **Exit behavior**. If you select **Fail**, the simulation job will fail if the tool exits. Select **Restart** to restart the tool. The default is **Restart**.
    e. Choose to enable or disable UI streaming. UI streaming is disabled by default.
    f. Select **Send output to CloudWatch** to record logs for the tool. The logs will be available in CloudWatch. Output is not sent to CloudWatch by default.

    Custom tools will start only after the main ROS launch process has started. For more information about custom tools, see Configuring Custom Simulation Tools (p. 179).
30. *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.

31. *Optional:* If your simulation application uses environment variables, specify the **Name** and **Value** pairs. Select **Add environment variable** to add additional variables.
32. *Optional:* If you want to import a world generated with Simulation WorldForge, select **Browse worlds** and then select the world you want to import.

    You must configure your simulation application launch file to include the world. For more information about using imported worlds, see Using an imported world in a simulation job (p. 68).
33. *Optional:* Configure traffic forwarding from the simulation job port to the application port. Simulation job networking must be configured in order to specify port mapping for your robot and simulation applications.
34. *Optional:* Specify one or more **Simulation application upload configurations**. A simulation job output destination must be configured in order to specify upload configurations. Each configuration specifies an upload behavior, a Unix glob file matching rule, and a location to place matching files.

    Default upload configurations maintain backwards compatibility with past simulation job output configurations. The default configurations will be added to additional upload configurations you create.

    For more information about custom uploads, see Configuring Custom Uploads (p. 156).
35. Choose **Next**.
36. *Optional:* Configure traffic forwarding from the simulation job port to the application port. Simulation job networking must be configured in order to specify port mapping for your robot and simulation applications.
37. 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.

```
$ aws robomaker create-simulation-job --max-job-duration-in-seconds 3600
 --iam-role arn:aws:iam::111111111111:role/MyRole --robot-applications
 application=arn:aws:robomaker:us-west-2:111111111111:robot-application/
MyRobotApplication/1551203485821,launchConfig="{packageName=hello_world_robot,launchFile=rotate.lau
 --simulation-applications application=arn:aws:robomaker:us-
west-2:111111111111:simulation-application/
MySimulationApplication/1551203427605,launchConfig="{packageName=hello_world_simulation,launchFile=
 --tags Region=North
```

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. 161).

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.

1. Sign in to the AWS RoboMaker console at https://console.aws.amazon.com/robomaker/.
2. In the left navigation pane, choose **Simulation run**, 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 **Compute**, select a simulation unit limit. Your simulation is allocated CPU and memory proportional to the supplied simulation unit limit. A simulation unit is 1 vcpu and 2GB of memory. The default is 15.
8. *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.
9. *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**.
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 Tagging Your AWS RoboMaker Resources (p. 224).
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. 76).

    Using `$LATEST` doesn't protect you from changes in Amazon S3. When AWS RoboMaker access the file, it will set to read only. For more informatiom about versioning in AWS RoboMaker, see Application Versioning (p. 19).
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.

17. *Optional:* To configure robot application tools, expand **Robot application tools**. Select **Use default tools** to use preconfigured tools. Select **Customize tools** to add, remove or edit custom tools to use with application.

    To add a new custom tool:

    a. Select **Add tool**.
    b. On the **Add application tool**, specify a **Tool name**.
    c. Specify the command-line arguments for the tool. You must include the tool executable name.
    d. Choose an **Exit behavior**. If you select **Fail**, the simulation job will fail if the tool exits. Select **Restart** to restart the tool. The default is **Restart**.
    e. Choose to enable or disable UI streaming. UI streaming is disabled by default.
    f. Select **Send output to CloudWatch** to record logs for the tool. The logs will be available in CloudWatch. Output is not sent to CloudWatch by default.

    Custom tools will start only after the main ROS launch process has started. For more information about custom tools, see Configuring Custom Simulation Tools (p. 179).
18. *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.
19. *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.
20. *Optional:* Configure traffic forwarding from the simulation job port to the application port. Simulation job networking must be configured in order to specify port mapping for your robot and simulation applications.
21. *Optional:* Specify one or more **Robot application upload configurations**. A simulation job output destination must be configured in order to specify upload configurations. Each configuration specifies an upload behavior, a Unix glob file matching rule, and a location to place matching files.

    Default upload configurations maintain backwards compatibility with past simulation job output configurations. The default configurations will be added to additional upload configurations you create.

    For more information about custom uploads, see Configuring Custom Uploads (p. 156).
22. Choose **Next**.
23. 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.
24. Type a **name** for the simulation application.
25. For **Software suite name**, select **RosbagPlay**. `Melodic` will automatically be selected as the **Software suite version**.
26. Select the **Browse S3**, and then specify the path to your simulation application.

    Using `$LATEST` doesn't protect you from changes in Amazon S3. When AWS RoboMaker access the file, it will set to read only. For more informatiom about versioning in AWS RoboMaker, see Application Versioning (p. 19).
27. In **Simulation application configuration**, provide the roslaunch **Launch package name** and the roslaunch **Launch file** for your simulation application.

28. *Optional:* To configure simulation application tools, expand **Simulation application tools**. Select **Use default tools** to use preconfigured tools. Select **Customize tools** to add, remove or edit custom tools to use with application.

    To add a new custom tool:

    a. Select **Add tool**.
    b. On the **Add application tool**, specify a **Tool name**.
    c. Specify the command-line arguments for the tool. You must include the tool executable name.
    d. Choose an **Exit behavior**. If you select **Fail**, the simulation job will fail if the tool exits. Select **Restart** to restart the tool. The default is **Restart**.
    e. Choose to enable or disable UI streaming. UI streaming is disabled by default.
    f. Select **Send output to CloudWatch** to record logs for the tool. The logs will be available in CloudWatch. Output is not sent to CloudWatch by default.

    Custom tools will start only after the main ROS launch process has started. For more information about custom tools, see Configuring Custom Simulation Tools (p. 179).
29. *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.
30. *Optional:* If your simulation application uses environment variables, specify the **Name** and **Value** pairs. Select **Add environment variable** to add additional variables.
31. 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.
32. *Optional:* If you want to import a world generated with Simulation WorldForge, select **Browse worlds** and then select the world you want to import.

    You must configure your simulation application launch file to include the world. For more information about using imported worlds, see Using an imported world in a simulation job (p. 68).
33. *Optional:* Configure traffic forwarding from the simulation job port to the application port. Simulation job networking must be configured in order to specify port mapping for your robot and simulation applications.
34. *Optional:* Specify one or more **Simulation application upload configurations**. A simulation job output destination must be configured in order to specify upload configurations. Each configuration specifies an upload behavior, a Unix glob file matching rule, and a location to place matching files.

    Default upload configurations maintain backwards compatibility with past simulation job output configurations. The default configurations will be added to additional upload configurations you create.

    For more information about custom uploads, see Configuring Custom Uploads (p. 156).
35. Choose **Next**.
36. 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

1. Sign in to the AWS RoboMaker console at https://console.aws.amazon.com/robomaker/.
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

1. Sign in to the AWS RoboMaker console at https://console.aws.amazon.com/robomaker/.
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.

1. Sign in to the AWS RoboMaker console at https://console.aws.amazon.com/robomaker/.
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.
6. Select **Create** to create the simulation job.

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

1. Sign in to the AWS RoboMaker console at https://console.aws.amazon.com/robomaker/.
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 **Restart**.
5. In the **Restart simulation job** page, select **Yes, restart**.

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

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.

```
{
    "batchPolicy": {
        "timeoutInSeconds": 400,
        "maxConcurrency": 2
    },
    "createSimulationJobRequests": [
        {
            "maxJobDurationInSeconds": 300,
            "iamRole": "arn:aws:iam::111111111111:role/MyRole",
            "failureBehavior": "Fail",
            "robotApplications": [
                {
                    "application": "arn:aws:robomaker:us-east-1:111111111111:robot-
application/MyRobotApplicationArn",
                    "launchConfig": {
                        "packageName": "hello_world_robot",
                        "launchFile": "rotate.launch"
                    }
                }
            ],
            "simulationApplications": [
                {
                    "application": "arn:aws:robomaker:us-
east-1:111111111111:simulation-applicationMySimulationApplicationArn",
                    "launchConfig": {
                        "packageName": "hello_world_simulation",
                        "launchFile": "simulation.launch"
                    }
                }
            ],
            "tags": {
             "myRequestTagKey" : "myRequestTagValue"
            }
        },
        {
            "maxJobDurationInSeconds": 300,
            "iamRole": "arn:aws:iam::111111111111:role/MyRole",
            "failureBehavior": "Fail",
            “simulationApplications": [
                {
                    "application": "arn:aws:robomaker:us-
east-1:111111111111:simulation-applicationMySimulationApplicationArn",
                    "launchConfig": {
                        "packageName": "hello_world_simulation",
                        "launchFile": "simulation.launch"
                    }
                }
            ]
        }
    ],
    "tags": {
         "myBatchTagKey" : "myBatchTagValue"
     }
}
```

3. Open a command prompt, then run the following AWS CLI command:

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

To view the simulation job batch, see Viewing a Simulation Job Batch (p. 177).

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

1. Sign in to the AWS RoboMaker console at https://console.aws.amazon.com/robomaker/.
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

1. Sign in to the AWS RoboMaker console at https://console.aws.amazon.com/robomaker/.
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.

```
$ 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:

1. Sign in to the AWS RoboMaker console at https://console.aws.amazon.com/robomaker/.
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**.

   Ti 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. You can also customize tools for your simulation job applications.

The simulation tools are configured for each of the applications running in your simulation job. Each tool is run in the context of one of the applications. The associated simulation application bundle is sourced to setup the ROS workspace. For example if you are running a robot application and a simulation application, rqt, rviz and a terminal tool will be sourced with your robot application bundle and Gazebo client and an additional terminal will be sourced with you simulation application bundle.

Common tasks include:

- Pause a Running Simulation (p. 180)
- View Node Graph (p. 182)
- View Robot Sensor Data (p. 183)
- Inspect ROS Topics and Messages (p. 183)

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.

You can access simulation tools in the application section of the simulation details page.

Topics

Configuring Custom Simulation Tools

AWS RoboMaker makes it easy for you configure custom tools for the applications in a simulation job. Use custom tools to interact with the simulation, as diagnostic utilities, or for other purposes. You can also configure default tools like rqt or rviz provided by AWS RoboMaker. If your simulation job is part of an automated pipeline, you can disable default tools and use fewer resources.

You can configure up to 10 custom tools. Custom tools are started after the main ROS process has started.

A custom tool configuration includes the following elements:

- **Tool name** — The name of the tool.
- **Command** — The command-line arguments for the tool. You must include the tool executable name. You can use environment variables including custom variables in your arguments. For example, to use the current simulation job id, you can reference `AWS_ROBOMAKER_SIMULATION_JOB_ID`.

  For more information about environment variables, see Environment Variables Created by AWS RoboMaker (p. 159).
- **Exit behavior** — Determines what action is taken if the custom tool exits. If you specify fail, the simulation job will fail. If you specify restart, the tool will be restarted. The default is restart.
- **UI streaming** — Specifies whether a streaming session will be configured for the tool. If True, AWS RoboMaker will configure a connection so you can interact with the tool as it is running in the simulation. It must have a graphical user interface. The default is false.
- **Log behavior** — Specifies whether tool output is streamed to CloudWatch Logs. The default is false. For more information about the information captured from a running simulation job, see Accessing Simulation Job Data (p. 156).

GZClient

Gazebo lets you build 3D worlds with robots, terrain, and other objects. It also has has a physics engine for modeling illumination, gravity, and other forces. Robotics developers use Gazebo to evaluate and test robots in different scenarios, often times more quickly than using physical robots and scenarios. Gazebo also makes it easier to test other aspects of your robot like error handling, battery life, navigation, and machine learning algorithms.

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

Topics

Pause a Running Simulation

You can pause a running simulation in GZClient 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 GZClient, it presents a view of the simulated world. The initial perspective is configured by the simulation application developer.

1. In **GZClient**, 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

GZClient 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 **GZClient**, 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 distrurbances during operation. Objects might tumble, spin, and interact with neighborhing 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 Hello World robot sample from Getting Started with AWS RoboMaker (p. 7) sample. The principles apply to robots and objects that are not static. Entities marked as static only have collision geometry.

1. In **GZClient**, 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 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 updsidedown. 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

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. 180).

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 inforamtion.

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 changed, 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 GUI tool provides a convinent way to explore your simulation with command line based ROS commands. You can access a terminal for each of the applications running in your simulation. The application's bundle will be sourced to setup your ROS workspace. You can access the application terminal tools in the Robot or Simulation application section in the simulation job details page.

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

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

1. In **terminal**, type the following command to see a list of available topics:

```
rostopic list
```

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

```
rostopic echo /topic_name
```

Inspect ROS Nodes and Services

Use `rosnode` to display information about ROS nodes and services. For more information about `rosnode`, see http://wiki.ros.org/rosnode.

1. In **terminal**, type the following command to see a list of available topics:

   ```
   rosnode list
   ```
2. Use the following command to view messages associated with a listed topic:

   ```
   rostopic info /node_name
   ```

View Log Files in Real Time

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

```
tail -f /tmp/robot-logs/stdout_and_stderror
```

Application Deployment

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

Topics

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. 7).

> **Note**
> AWS RoboMaker does not support over-the-air deployments if you are running AWS IoT Greengrass in a Docker container on the robot.

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.

4. The Lambda downloads and uncompresses the robot application bundle from Amazon S3. If a pre-launch script was specified, it is run. Then the Lambda starts ROS using the launch file and package specified. If a post-launch script was specified, it is run after ROS is started. Finally, the status of the deployment is updated.

Managing Robots

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

Topics

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

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. 187).

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:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "robomaker:UpdateRobotDeployment"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:List*",
                "s3:Get*"
            ],
            "Resource": ["arn:aws:s3:::my-robot-application-source-bucket/*"]
        }
    ]
}
```

Choose **Review policy**, type in a **Name**, and then choose **Create policy**.

3. Choose **Roles** and then choose **Create role**.
4. In the **Create role: Step 1** page, choose **Greengrass** and then choose **Next: Permissions**.
5. In the **Permissions** page, select the policy you created above, then choose **Next: Tags**.
6. In the **Add tags** page, add optional tags to the role, then choose **Next: Review**.
7. In the **Review** page, type in a **Role name** and then choose **Create role**.
8. Next, update the trust policy to include AWS Lambda. Select the new role, then select the **Trust relationships** tab.
9. In the **Trust relationship** tab, select **Edit trust relationship**.
10. Update the trust relationship with the following policy document:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "lambda.amazonaws.com",
          "greengrass.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

11. Select **Update Trust Policy**.

Create a robot

To create a robot:

1. Sign in to the AWS RoboMaker console at https://console.aws.amazon.com/robomaker/.
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. 186).
7. Optionally, modify the **Greengrass prefix**. This string is prepended to AWS IoT Greengrass objects created on your behalf.

8. Select a **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.

   Your tags can be populated as environment variables on the robot when the robot application is deployed. Tags must only contain alphanumeric or underscore characters. The environment variable name will be formatted as `AWS_ROBOMAKER_ROBOT_TAG_KEY_TAGKEY`, where *`TAGKEY`* is your tag key. The tag value will be the environment variable value.

   For more about tagging, see Tagging Your AWS RoboMaker Resources (p. 224).
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.

    For more information about how you can verify your device supports AWS IoT Greengrass, see https://docs.aws.amazon.com/greengrass/latest/developerguide/device-tester-for-greengrass-ug.htmlAWS IoT Device Tester for AWS IoT Greengrass.

    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

1. Sign in to the AWS RoboMaker console at https://console.aws.amazon.com/robomaker/.
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/.

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

Application Deployment

Deploy, update, and manage your robotics applications throughout the production lifecycle of your robots. AWS RoboMaker application deployment is integrated with AWS IoT Greengrass for secure, over-the-air deployments. You can use AWS RoboMaker application deployment to group your robots and update them accordingly with bug fixes or new features.

Topics

Creating a Fleet

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

Using the console

1. Sign in to the AWS RoboMaker console at https://console.aws.amazon.com/robomaker/.
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. 224).
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:

Using the console

1. Sign in to the AWS RoboMaker console at https://console.aws.amazon.com/robomaker/.
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

1. Sign in to the AWS RoboMaker console at https://console.aws.amazon.com/robomaker/.
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

1. Sign in to the AWS RoboMaker console at https://console.aws.amazon.com/robomaker/.
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).

You can set fleet and robot deployment timeouts. The default fleet timeout is 30 days. The default robot deployment timeout is 5 hours. For example, if you have 100 robots in your fleet and you deploy to them sequentially with a robot timeout of 7 days, the deployment might take 700 days. AWS RoboMaker will timeout the deployment at 30 days, the default fleet timeout.

Topics

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
    exit 1
fi
```

The download condition file is downloaded to `/home/gcc_user/roboMakerDeploymentPackage/MyS3KeyName`. If the `/home/gcc_user/` directory does not exist, it is downloaded to `/tmp/roboMakerDeploymentPackage`. If there is an existing download condition file on the robot, it will be overwritten during the next deployment. It will also be ignored during robot reboot or restart.

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.

You can set fleet and robot deployment timeouts. The default fleet timeout is 30 days. The default robot deployment timeout is 5 hours. For example, if you have 100 robots in your fleet and you deploy to them sequentially with a robot timeout of 7 days, the deployment might take 700 days. AWS RoboMaker will timeout the deployment at 30 days, the default fleet timeout.

For more details about how AWS RoboMaker deploys a robot application, see How Robot Applications are Deployed (p. 185).

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:

1. Sign in to the AWS RoboMaker console at https://console.aws.amazon.com/robomaker/.
2. In the left navigation pane, choose **Application deployment**, 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. 75).
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 scrip 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 scrip 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.

    Your tags can be populated as environment variables on the robot when the robot application is deployed. Tags must only contain alphanumeric or underscore characters. The environment variable name will be formatted as `AWS_ROBOMAKER_ROBOT_TAG_KEY_`*`TAGKEY`*, where *`TAGKEY`* is your tag key. The tag value will be the environment variable value.

    For more about tagging, see Tagging Your AWS RoboMaker Resources (p. 224).
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.

```
$ aws robomaker create-deployment-job --fleet=my-fleet-
arn --deployment-application-configs application=my-
robotarn,applicationVersion="$LATEST",launchConfig={packageName="cloudwatch_robot",launchFile="clou
 --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

1. Sign in to the AWS RoboMaker console at https://console.aws.amazon.com/robomaker/.
2. In the left navigation pane, choose **Application deployment**, 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

The AWS shared responsibility model applies to data protection in AWS RoboMaker. As described in this model, AWS is responsible for protecting the global infrastructure that runs all of the AWS Cloud. You are responsible for maintaining control over your content that is hosted on this infrastructure. This content includes the security configuration and management tasks for the AWS services that you use. For more information about data privacy, see the Data Privacy FAQ. For information about data protection in Europe, see the AWS Shared Responsibility Model and GDPR blog post on the *AWS Security Blog*.

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). That way 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. We recommend TLS 1.2 or later.
- 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.
- If you require FIPS 140-2 validated cryptographic modules when accessing AWS through a command line interface or an API, use a FIPS endpoint. For more information about the available FIPS endpoints, see Federal Information Processing Standard (FIPS) 140-2.

We strongly recommend that you never put confidential or sensitive information, such as your customers' email addresses, into tags or 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 tags or free-form fields used for names may be used for billing or diagnostic logs. If you provide a URL to an external server, we strongly recommend that you do not include credentials information in the URL to validate your request to that server.

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. 212). You can optionally learn more about authentication and access control by viewing What is Authentication? (p. 202), What is Access Control? (p. 203), and What are Policies? (p. 206).

Topics

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. 202).

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. 203) and What are Policies? (p. 206).

> **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 **AWSRoboMaker_FullAccess** 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:

```
{
    "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 View Worlds in the AWS RoboMaker in the Console

You can grant permissions required to view AWS RoboMaker worlds in the AWS RoboMaker console by attaching the following policy to a user:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "robomaker: DescribeWorldForgeImageRedirect "
            ],
            "Resource": "*",
            "Effect": "Allow"
        }
    ]
}
```

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. 215).

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. 213).

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:

```
{
    "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"
        }
        {
            "Action": [
                "ecr:BatchGetImage",
                "ecr:GetAuthorizationToken",
                "ecr:BatchCheckLayerAvailability",
                "ecr:GetDownloadUrlForLayer"
            ],
            "Resource": "arn:partition:ecr:region:account#:repository/repository_name",
            "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. 159)

```
{
    "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:

```
{
  "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

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.

```
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. 212).

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? (p. 202)
- What is Access Control? (p. 203)
- What are Policies? (p. 206)

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. 195) and then see Getting Started with IAM (p. 212).

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. 205).
  - **AWS service access** – A service role is an IAM role that a service assumes to perform actions on your behalf. An IAM administrator can create, modify, and delete a service role from within IAM. 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. 195) and then see Getting Started with IAM (p. 212).

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. 206).

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. 204)** – Control what the person making the request (the *principal*) is allowed to do.
- **IAM Identities (p. 204)** – Control which IAM identities (groups, users, and roles) can be accessed and how.
- **IAM Policies (p. 204)** – Control who can create, edit, and delete customer managed policies, and who can attach and detach all managed policies.
- **AWS Resources (p. 205)** – Control who has access to resources using an identity-based policy or a resource-based policy.
- **AWS Accounts (p. 205)** – 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. 195) and then see Getting Started with IAM (p. 212).

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

You can attach policies to IAM identities. For example, you can do the following:

- **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.

AWS managed policies for AWS RoboMaker

To add permissions to users, groups, and roles, it is easier to use AWS managed policies than to write policies yourself. It takes time and expertise to create IAM customer managed policies that provide your team with only the permissions they need. To get started quickly, you can use our AWS managed policies. These policies cover common use cases and are available in your AWS account. For more information about AWS managed policies, see AWS managed policies in the *IAM User Guide*.

AWS services maintain and update AWS managed policies. You can't change the permissions in AWS managed policies. Services occasionally add additional permissions to an AWS managed policy to support new features. This type of update affects all identities (users, groups, and roles) where the policy is attached. Services are most likely to update an AWS managed policy when a new feature is launched or when new operations become available. Services do not remove permissions from an AWS managed policy, so policy updates won't break your existing permissions.

Additionally, AWS supports managed policies for job functions that span multiple services. For example, the **ReadOnlyAccess** AWS managed policy provides read-only access to all AWS services and resources. When a service launches a new feature, AWS adds read-only permissions for new operations and resources. For a list and descriptions of job function policies, see AWS managed policies for job functions in the *IAM User Guide*.

AWS managed policy: AWSRoboMaker_FullAccess

This managed policy gives you the ability to use all AWS RoboMaker operations. It gives AWS RoboMaker access to the AWS services that are required for AWS RoboMaker to work successfully.

This policy grants *contributor* permissions that allows AWS RoboMaker to read images or bundles that you can use to create applications. Additionally, this policy gives you access to all AWS RoboMaker resources and operations. It also creates an IAM role in your account that manages Amazon EC2 resources in your account.

Permissions details

This policy includes the following permissions.

- `s3:GetObject` – If you're using a bundle for either your robot or simulation application, it allows AWS RoboMaker to get the zip files from your Amazon S3 bucket.
- `ecr:BatchGetImage` – If you're using an image for either your robot or simulation application, it allows AWS RoboMaker to get the image from your Amazon ECR repository.
- `ecr-public:DescribeImages` – If you're using a publicly available image for either your robot or simulation application, it allows AWS RoboMaker to get information about that image from the Amazon ECR repository.
- `iam:CreateServiceLinkedRole` – Provides AWS RoboMaker with access with the Amazon EC2 resources it needs to operate successfully. For more information, see Using Service-Linked Roles for AWS RoboMaker (p. 210).

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "robomaker:*",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "s3:GetObject",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:CalledViaFirst": "robomaker.amazonaws.com"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": "ecr:BatchGetImage",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:CalledViaFirst": "robomaker.amazonaws.com"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": "ecr-public:DescribeImages",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:CalledViaFirst": "robomaker.amazonaws.com"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:AWSServiceName": "robomaker.amazonaws.com"
                }
            }
        }
    ]
}
```

AWS RoboMaker updates to AWS managed policies

View details about updates to AWS managed policies for AWS RoboMaker since this service began tracking these changes. For automatic alerts about changes to this page, subscribe to the RSS feed on the AWS RoboMaker Document history page.

Change	Description	Date
AWSRoboMaker_FullAccess (p. 208) – New policy	AWS RoboMaker added a new policy to allow access to resources it needs to successfully run. This policy gives AWS RoboMaker access to the Amazon ECR images or zip files that you've stored on Amazon S3 to create your robot and simulation applications. It also gives AWS RoboMaker the ability to access the Amazon EC2 it needs to run successfully.	July 27, 2021
AWS RoboMaker started tracking changes	AWS RoboMaker started tracking changes for its AWS managed policies.	July 27, 2021

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.

Region name	Region identity	Support in AWS RoboMaker
US East (N. Virginia)	us-east-1	Yes
US East (Ohio)	us-east-2	Yes
US West (N. California)	us-west-1	Yes
US West (Oregon)	us-west-2	Yes
Asia Pacific (Mumbai)	ap-south-1	Yes
Asia Pacific (Osaka)	ap-northeast-3	Yes
Asia Pacific (Seoul)	ap-northeast-2	Yes
Asia Pacific (Singapore)	ap-southeast-1	Yes
Asia Pacific (Sydney)	ap-southeast-2	Yes
Asia Pacific (Tokyo)	ap-northeast-1	Yes
Canada (Central)	ca-central-1	Yes
Europe (Frankfurt)	eu-central-1	Yes
Europe (Ireland)	eu-west-1	Yes
Europe (London)	eu-west-2	Yes
Europe (Paris)	eu-west-3	Yes
South America (São Paulo)	sa-east-1	Yes
AWS GovCloud (US)	us-gov-west-1	No

Getting Started with IAM

AWS Identity and Access Management (IAM) is an AWS service that allows you manage access to services and resources securely. IAM is a feature of your AWS account offered at no additional charge.

> **Note**
> Before you start with IAM, review the introductory information on Authentication and Access Control for AWS RoboMaker (p. 195).

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. Sign in to the IAM console as the account owner by choosing **Root user** and entering your AWS account email address. On the next page, enter your password.

    Note
    We strongly recommend that you adhere to the best practice of using the **`Administrator`** IAM user that follows 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`**.
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. 208).
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.

```
{
    "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/${aws:username}"
        },
        {
            "Sid": "AllowIndividualUserToViewAndManageTheirOwnMFA",
            "Effect": "Allow",
            "Action": [
                "iam:CreateVirtualMFADevice",
                "iam:DeleteVirtualMFADevice",
                "iam:EnableMFADevice",
                "iam:ListMFADevices",
                "iam:ResyncMFADevice"
            ],
            "Resource": [
                "arn:aws:iam::*:mfa/${aws:username}",
                "arn:aws:iam::*:user/${aws:username}"
            ]
        },
        {
            "Sid": "AllowIndividualUserToDeactivateOnlyTheirOwnMFAOnlyWhenUsingMFA",
            "Effect": "Allow",
            "Action": [
                "iam:DeactivateMFADevice"
            ],
            "Resource": [
                "arn:aws:iam::*:mfa/${aws:username}",
                "arn:aws:iam::*:user/${aws:username}"
            ],
            "Condition": {
                "Bool": {
                    "aws:MultiFactorAuthPresent": "true"
                }
            }
        },
        {
            "Sid": "BlockMostAccessUnlessSignedInWithMFA",
            "Effect": "Deny",
            "NotAction": [
                "iam:CreateVirtualMFADevice",
                "iam:DeleteVirtualMFADevice",
                "iam:ListVirtualMFADevices",
                "iam:EnableMFADevice",
                "iam:ResyncMFADevice",
                "iam:ListAccountAliases",
                "iam:ListUsers",
                "iam:ListSSHPublicKeys",
                "iam:ListAccessKeys",
                "iam:ListServiceSpecificCredentials",
                "iam:ListMFADevices",
                "iam:GetAccountSummary",
                "sts:GetSessionToken"
            ],
            "Resource": "*",
            "Condition": {
                "BoolIfExists": {
                    "aws:MultiFactorAuthPresent": "false"
                }
            }
        }
    ]
}
```

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 devices, then we recommend that you enable those devices for AWS. Otherwise, you must purchase a device for each of your users and wait for the hardware to arrive. For more information, see Enabling a U2F Security Key in the *IAM User Guide*.

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.
4. Choose the **Security credentials** tab. Next to **Assigned MFA device**, choose **Manage**.
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

AWS RoboMaker sends metrics to Amazon CloudWatch. You can use the AWS Management Console, the AWS CLI, or an API to list the metrics that AWS RoboMaker sends to CloudWatch.

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.

Topics

AWS RoboMaker Simulation Metrics

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.

The following metrics are available in the `SimulationJobId` dimension.

Metric	Description
`RealTimeFactor`	The ratio of the amount of time that was simulated versus wall clock time. For example, if it takes an hour to simulate 30 minutes, the factor is .5. More complex simulations have a lower real time factor.
`vCPU*`	Number of virtual CPU cores used by the simulation job Unit: Count
`Memory*`	Amount of memory, in GB, used by the simulation job Unit: GB
`SimulationUnit*`	`SimulationUnit` is calculated based on vCPU and memory consumption of the simulation job Unit: Count

Important
Metrics marked with * are for estimation purposes. AWS RoboMaker emits metrics while preparing to run a simulation job. Charges do not accrue until the simulation job is in the `Running` state.

AWS RoboMaker Usage Metrics

You can use CloudWatch usage metrics to provide visibility into your account's usage of resources. Use these metrics to visualize your current service usage on CloudWatch graphs and dashboards.

AWS RoboMaker usage metrics correspond to AWS service quotas. You can configure alarms that alert you when your usage approaches a service quota. For more information about CloudWatch integration with service quotas, see Service Quotas Integration and Usage Metrics.

The following metrics are available in the `AWS/Usage` dimension.

Metric	Description
`ResourceCount`	The number of the specified resources running in your account. The resources are defined by the dimensions associated with the metric.

Metric	Description
	The most useful statistic for this metric is `MAXIMUM`, which represents the maximum number of resources used during the 1-minute period.

The following dimensions are used to refine the usage metrics that are published by AWS RoboMaker.

Dimension	Description
`Service`	The name of the AWS service containing the resource. For AWS RoboMaker usage metrics, the value for this dimension is `RoboMaker`.
`Type`	The type of entity that is being reported. Currently, the only valid value for AWS RoboMaker usage metrics is `Resource`.
`Resource`	The type of resource that is running. Currently, the valid values for AWS RoboMaker usage metrics are `RobotApplication`, `SimulationApplication`, `ActiveSimulationJob` and `ActiveSimulationJobBatch`.
`Class`	The class of resource being tracked. For AWS RoboMaker usage metrics with `ActiveSimulationJob` as the value of the Resource dimension, the valid values are `CPU | GPU_AND_CPU`. The value for this dimension defines the kind of compute resources used by the simulation jobs reported by that metric. For others, the class value is `None`.

These metrics are emitted every minute. Use these metrics to monitor usage and then request a corresponding limit increase if needed. For more information about monitoring your usage, see Visualizing Your Service Quotas and Setting Alarms.

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.

```
{
    "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.metal1.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",
    "recipientAccountId": "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 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. 225).

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
- CreateWorldExportJob
- CreateWorldGenerationJob
- CreateWorldTemplate
- 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 create a robot with tag `"env=prod"` (in the example, see the line `"aws:RequestTag/env" : "prod"`).
- Cannot delete a robot that has an existing tag `"env=prod"` (in the example, see the line `"aws:ResourceTag/env" : "prod"`).

```
{
    "Version" : "2012-10-17",
    "Statement" : [
        {
          "Effect" : "Deny",
          "Action" : "robomaker:CreateRobot",
          "Resource" : "*",
          "Condition" : {
            "StringEquals" : {
              "aws:RequestTag/env" : "prod"
            }
          }
        },
        {
          "Effect" : "Deny",
          "Action" : "robomaker:DeleteRobot",
          "Resource" : "*",
          "Condition" : {
            "StringEquals" : {
              "aws:ResourceTag/env" : "prod"
            }
          }
        },
        {
          "Effect": "Allow",
          "Action": "robomaker:*",
          "Resource": "*"
        }
    ]
}
```

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

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`.

See Modifying a Role for more information on viewing role access and adding a trust policy to an IAM role.

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. 75) or Creating a Simulation Application Version (p. 75).

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.

```
[Wrn] [ModelDatabase.cc:340] Getting models from[http://models.gazebosim.org/]. This may
 take a few seconds.
[Wrn] [ModelDatabase.cc:212] Unable to connect to model database using [http://
models.gazebosim.org//database.config]. Only locally installed models will be available.
[Err] [ModelDatabase.cc:414] Unable to download model[model://model_name]
[Err] [SystemPaths.cc:429] File or path does not exist[""]
Error [parser.cc:581] Unable to find uri[model://model_name]
```

Troubleshooting Simulation WorldForge

This section can help you fix issues with Simulation WorldForge.

Topics

Why did my world generation job fail?

This section can help you fix issues with Simulation WorldForge simulation world generation.

Is your world count 0 or greater than 50?

If your world generation job did not complete, make sure your world count, `floorplanCount * interiorCountPerFloorplan`, is greater than 1 and less than 50.

Why did my world export job fail?

This section can help you fix issues with Simulation WorldForge world export jobs.

Do you 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-world-export-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 Amazon S3 bucket for a export job, your role must have permissions to the bucket. Update your trust policy to include the following permissions:

```
{
  "Effect": "Allow",
  "Action": [
    "s3:AbortMultipartUpload",
    "s3:GetObject",
    "s3:PutObject"
  ],
  "Resource": “my-bucket"
}
```

Did you modify or remove the bucket specified for the export job?

If you update your bucket during the export job, you may get a `ResourceNotFound` error from export job.

Why is there an issue with world image?

This section can help you fix issues with your world image.

Why is there no door in my doorway?

You can only add doors using a Version 2 template or later. You can update a Version 1 template to a later version. For more information, see Simulation World Template Versions, Features, and Changes (p. 144).

Because AWS RoboMaker Simulation WorldForge creates worlds that are unique and random, the door configuration that you've specified might not exist in the world when you generate it. For example, you might specify a door between a living room and a kitchen in your template, but there might be an open wall between those rooms. Because there's an open wall instead of a doorway, you wouldn't be able to add a door there.

Why does my door block the entrance to my room?

The door blocking the entrance to a room is a circumstance that you can use to challenge your robots. To create a world that doesn't present this challenge to your robots, you can do one of the following:

- Generate another world from your world template. The door generated in the new world might not block the entrance.
- Change the open percentage of the door in the world template.
-

Why are the walls in my world image shorter than the walls in my Simulation Job or exported world?

To give you the ability to see your Simulation WorldForge worlds without it being obscured by walls, AWS RoboMaker truncates the walls in the world image. The walls have the height that you specify in your world template in the worlds that you create.

For worlds generated by the Version 2 template or later, the door models are not truncated in the world images. The height of the doors in the world image will be the same as the height of the doors in the worlds you create.

Troubleshooting Development Environments

This section helps you fix issues with creating applications in the AWS RoboMaker environment.

ROS GPG Key Error

If you are receiving GPG errors when using commands such as `sudo apt update` or sudo `apt-get update`, your ROS GPG key may be expired. To read more details, visit this Open Robotics post.

Error

```
    W: An error occurred during the signature verification.
    The repository is not updated and the previous index files will be used.
    GPG error: http://packages.ros.org/ros/ubuntu bionic InRelease:
    The following signatures were invalid: EXPKEYSIG F42ED6FBAB17C654
    Open Robotics <info@osrfoundation.org>
```

Solution

To fix this issue you need to update the public key you use for ROS apt repositories.

For ROS 1 installations, run this command:

```
     curl -s https://raw.githubusercontent.com/ros/rosdistro/master/ros.asc | sudo apt-key
 add -
```

For ROS 2 installations, run this command:

```
     sudo curl -sSL https://raw.githubusercontent.com/ros/rosdistro/master/ros.key -o /
usr/share/keyrings/ros-archive-keyring.gpg
```

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 **Application deployment** 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.

For more information about how you can verify your device supports AWS IoT Greengrass, see https://docs.aws.amazon.com/greengrass/latest/developerguide/device-tester-for-greengrass-ug.htmlAWS IoT Device Tester for AWS IoT Greengrass.

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 `s 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.

```
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.

name:password:user_ID:group_ID: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/melodic/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. 19).

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/
# ### package.xml
| ### CMakeLists.txt # okay
### intermediate_directory/
    ### package_2
    | ### package.xml
    | ### CMakeLists.txt # okay
    ### package_3
    | ### package.xml
    | ### CMakeLists.txt  # okay
    ### CMakeLists.txt  # !!! remove !!!
```

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-melodic-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**.

API Reference

This section contains the API Reference documentation.

Actions

The following actions are supported:

- GetWorldTemplateBody (p. 377)
- ListDeploymentJobs (p. 379)
- ListFleets (p. 383)
- ListRobotApplications (p. 386)
- ListRobots (p. 389)
- ListSimulationApplications (p. 392)
- ListSimulationJobBatches (p. 395)
- ListSimulationJobs (p. 398)
- ListTagsForResource (p. 401)
- ListWorldExportJobs (p. 403)
- ListWorldGenerationJobs (p. 406)
- ListWorlds (p. 409)
- ListWorldTemplates (p. 412)
- RegisterRobot (p. 415)
- RestartSimulationJob (p. 418)
- StartSimulationJobBatch (p. 420)
- SyncDeploymentJob (p. 431)
- TagResource (p. 436)
- UntagResource (p. 438)
- UpdateRobotApplication (p. 440)
- UpdateSimulationApplication (p. 444)
- UpdateWorldTemplate (p. 449)

BatchDeleteWorlds

Deletes one or more worlds in a batch operation.

Request Syntax

```
POST /batchDeleteWorlds HTTP/1.1
Content-type: application/json

{
   "worlds": [ "string" ]
}
```

URI Request Parameters

The request does not use any URI parameters.

Request Body

The request accepts the following data in JSON format.

worlds (p. 239)

A list of Amazon Resource Names (arns) that correspond to worlds to delete.

Type: Array of strings

Array Members: Minimum number of 1 item. Maximum number of 100 items.

Length Constraints: Minimum length of 1. Maximum length of 1224.

Pattern: `arn:.*`

Required: Yes

Response Syntax

```
HTTP/1.1 200
Content-type: application/json

{
   "unprocessedWorlds": [ "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.

unprocessedWorlds (p. 239)

A list of unprocessed worlds associated with the call. These worlds were not deleted.

Type: Array of strings

Array Members: Minimum number of 1 item. Maximum number of 100 items.

Length Constraints: Minimum length of 1. Maximum length of 1224.

Pattern: `arn:.*`

Errors

For information about the errors that are common to all actions, see Common Errors (p. 532).

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 V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

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. 241)

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.

Length Constraints: Minimum length of 1. Maximum length of 1224.

Pattern: `arn:.*`

Required: Yes

Response Syntax

```
HTTP/1.1 200
Content-type: application/json

{
   "jobs": [
      {
         "arn": "string",
         "clientRequestToken": "string",
         "compute": {
            "computeType": "string",
            "gpuUnitLimit": number,
            "simulationUnitLimit": number
         },
         "dataSources": [
            {
               "destination": "string",
               "name": "string",
               "s3Bucket": "string",
               "s3Keys": [
                  {
                     "etag": "string",
                     "s3Key": "string"
                  }
               ],
               "type": "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": {
                  "command": [ "string" ],
                  "environmentVariables": {
                     "string" : "string"
                  },
                  "launchFile": "string",
                  "packageName": "string",
                  "portForwardingConfig": {
                     "portMappings": [
                        {
                           "applicationPort": number,
                           "enableOnPublicIp": boolean,
                           "jobPort": number
                        }
                     ]
                  },
                  "streamUI": boolean
               },
               "tools": [
                  {
                     "command": "string",
                     "exitBehavior": "string",
                     "name": "string",
                     "streamOutputToCloudWatch": boolean,
                     "streamUI": boolean
                  }
               ],
               "uploadConfigurations": [
                  {
                     "name": "string",
                     "path": "string",
                     "uploadBehavior": "string"
                  }
               ],
               "useDefaultTools": boolean,
               "useDefaultUploadConfigurations": boolean
            }
         ],
         "simulationApplications": [ 
            { 
               "application": "string",
               "applicationVersion": "string",
               "launchConfig": { 
                  "command": [ "string" ],
                  "environmentVariables": { 
                     "string" : "string" 
                  },
                  "launchFile": "string",
                  "packageName": "string",
                  "portForwardingConfig": { 
                     "portMappings": [ 
                        { 
                           "applicationPort": number,
                           "enableOnPublicIp": boolean,
                           "jobPort": number
                        }
                     ]
                  },
                  "streamUI": boolean
               },
               "tools": [ 
                  { 
                     "command": "string",
                     "exitBehavior": "string",
                     "name": "string",
                     "streamOutputToCloudWatch": boolean,
                     "streamUI": boolean
                  }
               ],
               "uploadConfigurations": [ 
                  { 
                     "name": "string",
                     "path": "string",
                     "uploadBehavior": "string"
                  }
               ],
               "useDefaultTools": boolean,
               "useDefaultUploadConfigurations": boolean,
               "worldConfigs": [ 
                  { 
                     "world": "string"
                  }
               ]
            }
         ],
         "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. 241)

A list of simulation jobs.

Type: Array of SimulationJob (p. 499) objects

unprocessedJobs (p. 241)

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.

Length Constraints: Minimum length of 1. Maximum length of 1224.

Pattern: `arn:.*`

Errors

For information about the errors that are common to all actions, see Common Errors (p. 532).

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 V2
- 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. 246)

The deployment job ARN to cancel.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1224.

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. 532).

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 V2
- 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. 248)

The simulation job ARN to cancel.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1224.

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. 532).

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 V2
- 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. 250)

The id of the batch to cancel.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1224.

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. 532).

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 V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

CancelWorldExportJob

Cancels the specified export job.

Request Syntax

```
POST /cancelWorldExportJob 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. 252)

The Amazon Resource Name (arn) of the world export job to cancel.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1224.

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. 532).

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 V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

CancelWorldGenerationJob

Cancels the specified world generator job.

Request Syntax

```
POST /cancelWorldGenerationJob 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. 254)

The Amazon Resource Name (arn) of the world generator job to cancel.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1224.

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. 532).

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 V2
- 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

```
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. 256)

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. 256)

The deployment application configuration.

Type: Array of DeploymentApplicationConfig (p. 460) objects

Array Members: Fixed number of 1 item.

Required: Yes

deploymentConfig (p. 256)

The requested deployment configuration.

Type: DeploymentConfig (p. 461) object

Required: No

fleet (p. 256)

The Amazon Resource Name (ARN) of the fleet to deploy.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1224.

Pattern: `arn:.*`

Required: Yes

tags (p. 256)

A map that contains tag keys and tag values that are attached to the deployment job.

Type: String to string map

Map Entries: Minimum number of 0 items. Maximum number of 50 items.

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. 257)

The Amazon Resource Name (ARN) of the deployment job.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1224.

Pattern: `arn:.*`

createdAt (p. 257)

The time, in milliseconds since the epoch, when the fleet was created.

Type: Timestamp

deploymentApplicationConfigs (p. 257)

The deployment application configuration.

Type: Array of DeploymentApplicationConfig (p. 460) objects

Array Members: Fixed number of 1 item.

deploymentConfig (p. 257)

The deployment configuration.

Type: DeploymentConfig (p. 461) object

failureCode (p. 257)

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.

InternalServerError

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

Valid Values: `ResourceNotFound | EnvironmentSetupError | EtagMismatch | FailureThresholdBreached | RobotDeploymentAborted | RobotDeploymentNoResponse | RobotAgentConnectionTimeout | GreengrassDeploymentFailed | InvalidGreengrassGroup | MissingRobotArchitecture | MissingRobotApplicationArchitecture | MissingRobotDeploymentResource | GreengrassGroupVersionDoesNotExist | LambdaDeleted | ExtractingBundleFailure | PreLaunchFileFailure | PostLaunchFileFailure | BadPermissionError | DownloadConditionFailed |`

`BadLambdaAssociated | InternalServerError | RobotApplicationDoesNotExist | DeploymentFleetDoesNotExist | FleetDeploymentTimeout`

failureReason (p. 257)

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. 257)

The target fleet for the deployment job.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1224.

Pattern: `arn:.*`

status (p. 257)

The status of the deployment job.

Type: String

Valid Values: `Pending | Preparing | InProgress | Failed | Succeeded | Canceled`

tags (p. 257)

The list of all tags added to the deployment job.

Type: String to string map

Map Entries: Minimum number of 0 items. Maximum number of 50 items.

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. 532).

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 V2
- 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. 262)

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. 262)

A map that contains tag keys and tag values that are attached to the fleet.

Type: String to string map

Map Entries: Minimum number of 0 items. Maximum number of 50 items.

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,
   "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.

arn (p. 262)

The Amazon Resource Name (ARN) of the fleet.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1224.

Pattern: `arn:.*`

createdAt (p. 262)

The time, in milliseconds since the epoch, when the fleet was created.

Type: Timestamp

name (p. 262)

The name of the fleet.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 255.

Pattern: `[a-zA-Z0-9_\-]*`

tags (p. 262)

The list of all tags added to the fleet.

Type: String to string map

Map Entries: Minimum number of 0 items. Maximum number of 50 items.

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. 532).

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 V2
- 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. 265)

The target architecture of the robot.

Type: String

Valid Values: `X86_64 | ARM64 | ARMHF`

Required: Yes

greengrassGroupId (p. 265)

The Greengrass group id.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1224.

Pattern: `.*`

Required: Yes

name (p. 265)

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. 265)

A map that contains tag keys and tag values that are attached to the robot.

Type: String to string map

Map Entries: Minimum number of 0 items. Maximum number of 50 items.

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. 266)

The target architecture of the robot.

Type: String

Valid Values: `X86_64 | ARM64 | ARMHF`

arn (p. 266)

The Amazon Resource Name (ARN) of the robot.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1224.

Pattern: `arn:.*`

createdAt (p. 266)

The time, in milliseconds since the epoch, when the robot was created.

Type: Timestamp

greengrassGroupId (p. 266)

The Amazon Resource Name (ARN) of the Greengrass group associated with the robot.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1224.

Pattern: `.*`

name (p. 266)

The name of the robot.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 255.

Pattern: `[a-zA-Z0-9_\-]*`

tags (p. 266)

The list of all tags added to the robot.

Type: String to string map

Map Entries: Minimum number of 0 items. Maximum number of 50 items.

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. 532).

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 V2
- 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

{
   "environment": {
      "uri": "string"
   },
   "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.

environment (p. 269)

The object that contains that URI of the Docker image that you use for your robot application.

Type: Environment (p. 466) object

Required: No

name (p. 269)

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. 269)

The robot software suite (ROS distribuition) used by the robot application.

Type: RobotSoftwareSuite (p. 492) object

Required: Yes

sources (p. 269)

The sources of the robot application.

Type: Array of SourceConfig (p. 512) objects

Required: No

tags (p. 269)

A map that contains tag keys and tag values that are attached to the robot application.

Type: String to string map

Map Entries: Minimum number of 0 items. Maximum number of 50 items.

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",
   "environment": {
      "uri": "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. 270)

The Amazon Resource Name (ARN) of the robot application.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1224.

Pattern: `arn:.*`

environment (p. 270)

An object that contains the Docker image URI used to a create your robot application.

Type: Environment (p. 466) object

lastUpdatedAt (p. 270)

The time, in milliseconds since the epoch, when the robot application was last updated.

Type: Timestamp

name (p. 270)

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. 270)

The revision id of the robot application.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 40.

Pattern: `[a-zA-Z0-9_.\-]*`

robotSoftwareSuite (p. 270)

The robot software suite (ROS distribution) used by the robot application.

Type: RobotSoftwareSuite (p. 492) object

sources (p. 270)

The sources of the robot application.

Type: Array of Source (p. 511) objects

tags (p. 270)

The list of all tags added to the robot application.

Type: String to string map

Map Entries: Minimum number of 0 items. Maximum number of 50 items.

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. 270)

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. 532).

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 V2
- 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",
   "imageDigest": "string",
   "s3Etags": [ "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. 274)

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. 274)

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

Length Constraints: Minimum length of 1. Maximum length of 40.

Pattern: `[a-zA-Z0-9_.\-]*`

Required: No

imageDigest (p. 274)

A SHA256 identifier for the Docker image that you use for your robot application.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 72.

Pattern: `[Ss][Hh][Aa]256:[0-9a-fA-F]{64}`

Required: No

s3Etags (p. 274)

The Amazon S3 identifier for the zip file bundle that you use for your robot application.

Type: Array of strings

Required: No

Response Syntax

```
HTTP/1.1 200
Content-type: application/json

{
   "arn": "string",
   "environment": {
      "uri": "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. 275)

The Amazon Resource Name (ARN) of the robot application.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1224.

Pattern: `arn:.*`

environment (p. 275)

The object that contains the Docker image URI used to create your robot application.

Type: Environment (p. 466) object

lastUpdatedAt (p. 275)

The time, in milliseconds since the epoch, when the robot application was last updated.

Type: Timestamp

name (p. 275)

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. 275)

The revision id of the robot application.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 40.

Pattern: `[a-zA-Z0-9_.\-]*`

robotSoftwareSuite (p. 275)

The robot software suite (ROS distribution) used by the robot application.

Type: RobotSoftwareSuite (p. 492) object

sources (p. 275)

The sources of the robot application.

Type: Array of Source (p. 511) objects

version (p. 275)

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. 532).

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 V2
- 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

{
   "environment": { 
      "uri": "string"
   },
   "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.

environment (p. 278)

The object that contains the Docker image URI used to create your simulation application.

Type: Environment (p. 466) object

Required: No

name (p. 278)

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. 278)

The rendering engine for the simulation application.

Type: RenderingEngine (p. 483) object

Required: No

robotSoftwareSuite (p. 278)

The robot software suite (ROS distribution) used by the simulation application.

Type: RobotSoftwareSuite (p. 492) object

Required: Yes

simulationSoftwareSuite (p. 278)

The simulation software suite used by the simulation application.

Type: SimulationSoftwareSuite (p. 510) object

Required: Yes

sources (p. 278)

The sources of the simulation application.

Type: Array of SourceConfig (p. 512) objects

Required: No

tags (p. 278)

A map that contains tag keys and tag values that are attached to the simulation application.

Type: String to string map

Map Entries: Minimum number of 0 items. Maximum number of 50 items.

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",
   "environment": {
      "uri": "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. 279)

The Amazon Resource Name (ARN) of the simulation application.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1224.

Pattern: `arn:.*`

environment (p. 279)

The object that contains the Docker image URI that you used to create your simulation application.

Type: Environment (p. 466) object

lastUpdatedAt (p. 279)

The time, in milliseconds since the epoch, when the simulation application was last updated.

Type: Timestamp

name (p. 279)

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. 279)

The rendering engine for the simulation application.

Type: RenderingEngine (p. 483) object

revisionId (p. 279)

The revision id of the simulation application.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 40.

Pattern: `[a-zA-Z0-9_.\-]*`

robotSoftwareSuite (p. 279)

Information about the robot software suite (ROS distribution).

Type: RobotSoftwareSuite (p. 492) object

simulationSoftwareSuite (p. 279)

The simulation software suite used by the simulation application.

Type: SimulationSoftwareSuite (p. 510) object

sources (p. 279)

The sources of the simulation application.

Type: Array of Source (p. 511) objects

tags (p. 279)

The list of all tags added to the simulation application.

Type: String to string map

Map Entries: Minimum number of 0 items. Maximum number of 50 items.

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. 279)

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. 532).

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 V2
- 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",
   "imageDigest": "string",
   "s3Etags": [ "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 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

Length Constraints: Minimum length of 1. Maximum length of 40.

Pattern: `[a-zA-Z0-9_.\-]*`

Required: No

imageDigest (p. 283)

The SHA256 digest used to identify the Docker image URI used to created the simulation application.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 72.

Pattern: `[Ss][Hh][Aa]256:[0-9a-fA-F]{64}`

Required: No

s3Etags (p. 283)

The Amazon S3 eTag identifier for the zip file bundle that you use to create the simulation application.

Type: Array of strings

Required: No

Response Syntax

```
HTTP/1.1 200
Content-type: application/json

{
   "arn": "string",
   "environment": {
      "uri": "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. 284)

The Amazon Resource Name (ARN) of the simulation application.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1224.

Pattern: `arn:.*`

environment (p. 284)

The object that contains the Docker image URI used to create the simulation application.

Type: Environment (p. 466) object

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. 483) object

revisionId (p. 284)

The revision ID of the simulation application.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 40.

Pattern: `[a-zA-Z0-9_.\-]*`

robotSoftwareSuite (p. 284)

Information about the robot software suite (ROS distribution).

Type: RobotSoftwareSuite (p. 492) object

simulationSoftwareSuite (p. 284)

The simulation software suite used by the simulation application.

Type: SimulationSoftwareSuite (p. 510) object

sources (p. 284)

The sources of the simulation application.

Type: Array of Source (p. 511) objects

version (p. 284)

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. 532).

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 V2
- 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",
   "compute": {
      "computeType": "string",
      "gpuUnitLimit": number,
      "simulationUnitLimit": number
   },
   "dataSources": [
      {
         "destination": "string",
         "name": "string",
         "s3Bucket": "string",
         "s3Keys": [ "string" ],
         "type": "string"
      }
   ],
   "failureBehavior": "string",
   "iamRole": "string",
   "loggingConfig": {
      "recordAllRosTopics": boolean
   },
   "maxJobDurationInSeconds": number,
   "outputLocation": {
      "s3Bucket": "string",
      "s3Prefix": "string"
   },
   "robotApplications": [
      {
         "application": "string",
         "applicationVersion": "string",
         "launchConfig": {
            "command": [ "string" ],
            "environmentVariables": {
               "string" : "string"
            },
            "launchFile": "string",
            "packageName": "string",
            "portForwardingConfig": {
               "portMappings": [
                  {
                     "applicationPort": number,
                     "enableOnPublicIp": boolean,
                     "jobPort": number
                  }
               ]
            },
            "streamUI": boolean
         },
         "tools": [
            {
               "command": "string",
               "exitBehavior": "string",
               "name": "string",
               "streamOutputToCloudWatch": boolean,
               "streamUI": boolean
            }
         ],
         "uploadConfigurations": [
            {
               "name": "string",
               "path": "string",
               "uploadBehavior": "string"
            }
         ],
         "useDefaultTools": boolean,
         "useDefaultUploadConfigurations": boolean
      }
   ],
   "simulationApplications": [
      {
         "application": "string",
         "applicationVersion": "string",
         "launchConfig": {
            "command": [ "string" ],
            "environmentVariables": {
               "string" : "string"
            },
            "launchFile": "string",
            "packageName": "string",
            "portForwardingConfig": {
               "portMappings": [
                  {
                     "applicationPort": number,
                     "enableOnPublicIp": boolean,
                     "jobPort": number
                  }
               ]
            },
            "streamUI": boolean
         },
         "tools": [
            {
               "command": "string",
               "exitBehavior": "string",
               "name": "string",
               "streamOutputToCloudWatch": boolean,
               "streamUI": boolean
            }
         ],
         "uploadConfigurations": [
            {
               "name": "string",
               "path": "string",
               "uploadBehavior": "string"
            }
         ],
         "useDefaultTools": boolean,
         "useDefaultUploadConfigurations": boolean,
         "worldConfigs": [
            {
               "world": "string"
            }
         ]
      }
   ],
   "tags": {
      "string" : "string"
   },
   "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. 287)

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

compute (p. 287)

Compute information for the simulation job.

Type: Compute (p. 454) object

Required: No

dataSources (p. 287)

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. 458) objects

Array Members: Minimum number of 1 item. Maximum number of 5 items.

Required: No

failureBehavior (p. 287)

The failure behavior the simulation job.

Continue

Leaves the instance running for its maximum timeout duration after a `4XX` error code.

Fail

Stop the simulation job and terminate the instance.

Type: String

Valid Values: `Fail | Continue`

Required: No

iamRole (p. 287)

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. 287)

The logging configuration.

Type: LoggingConfig (p. 476) object

Required: No

maxJobDurationInSeconds (p. 287)

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. 287)

Location for output files generated by the simulation job.

Type: OutputLocation (p. 478) object

Required: No

robotApplications (p. 287)

The robot application to use in the simulation job.

Type: Array of RobotApplicationConfig (p. 486) objects

Array Members: Fixed number of 1 item.

Required: No

simulationApplications (p. 287)

The simulation application to use in the simulation job.

Type: Array of SimulationApplicationConfig (p. 495) objects

Array Members: Fixed number of 1 item.

Required: No

tags (p. 287)

A map that contains tag keys and tag values that are attached to the simulation job.

Type: String to string map

Map Entries: Minimum number of 0 items. Maximum number of 50 items.

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. 287)

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. 520) object

Required: No

Response Syntax

```
HTTP/1.1 200
Content-type: application/json

{
   "arn": "string",
   "clientRequestToken": "string",
   "compute": {
      "computeType": "string",
      "gpuUnitLimit": number,
      "simulationUnitLimit": number
   },
   "dataSources": [
      {
         "destination": "string",
         "name": "string",
         "s3Bucket": "string",
         "s3Keys": [
            {
               "etag": "string",
               "s3Key": "string"
            }
         ],
         "type": "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": {
            "command": [ "string" ],
            "environmentVariables": {
               "string" : "string"
            },
            "launchFile": "string",
            "packageName": "string",
            "portForwardingConfig": {
               "portMappings": [
                  {
                     "applicationPort": number,
                     "enableOnPublicIp": boolean,
                     "jobPort": number
                  }
               ]
            },
            "streamUI": boolean
         },
         "tools": [
            {
               "command": "string",
               "exitBehavior": "string",
               "name": "string",
               "streamOutputToCloudWatch": boolean,
               "streamUI": boolean
            }
         ],
         "uploadConfigurations": [
            {
               "name": "string",
               "path": "string",
               "uploadBehavior": "string"
            }
         ],
         "useDefaultTools": boolean,
         "useDefaultUploadConfigurations": boolean
      }
   ],
   "simulationApplications": [
      {
         "application": "string",
         "applicationVersion": "string",
         "launchConfig": {
            "command": [ "string" ],
            "environmentVariables": {
               "string" : "string"
            },
            "launchFile": "string",
            "packageName": "string",
            "portForwardingConfig": {
               "portMappings": [
                  {
                     "applicationPort": number,
                     "enableOnPublicIp": boolean,
                     "jobPort": number
                  }
               ]
            },
            "streamUI": boolean
         },
         "tools": [
            {
               "command": "string",
               "exitBehavior": "string",
               "name": "string",
               "streamOutputToCloudWatch": boolean,
               "streamUI": boolean
            }
         ],
         "uploadConfigurations": [
            {
               "name": "string",
               "path": "string",
               "uploadBehavior": "string"
            }
         ],
         "useDefaultTools": boolean,
         "useDefaultUploadConfigurations": boolean,
         "worldConfigs": [
            {
               "world": "string"
            }
         ]
      }
   ],
   "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. 291)

The Amazon Resource Name (ARN) of the simulation job.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1224.

Pattern: `arn:.*`

clientRequestToken (p. 291)

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_\-=]*`

compute (p. 291)

Compute information for the simulation job.

Type: ComputeResponse (p. 455) object

dataSources (p. 291)

The data sources for the simulation job.

Type: Array of DataSource (p. 456) objects

failureBehavior (p. 291)

the failure behavior for the simulation job.

Type: String

Valid Values: `Fail | Continue`

failureCode (p. 291)

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 | RobotApplicationHealthCheckFailure | SimulationApplicationHealthCheckFailure | BadPermissionsRobotApplication | BadPermissionsSimulationApplication | BadPermissionsS3Object | BadPermissionsS3Output | BadPermissionsCloudwatchLogs | SubnetIpLimitExceeded | ENILimitExceeded | BadPermissionsUserCredentials | InvalidBundleRobotApplication | InvalidBundleSimulationApplication | InvalidS3Resource | ThrottlingError | LimitExceeded | MismatchedEtag | RobotApplicationVersionMismatchedEtag | SimulationApplicationVersionMismatchedEtag | ResourceNotFound | RequestThrottled | BatchTimedOut | BatchCanceled | InvalidInput | WrongRegionS3Bucket | WrongRegionS3Output | WrongRegionRobotApplication | WrongRegionSimulationApplication | UploadContentMismatchError`

iamRole (p. 291)

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. 291)

The time, in milliseconds since the epoch, when the simulation job was last started.

Type: Timestamp

lastUpdatedAt (p. 291)

The time, in milliseconds since the epoch, when the simulation job was last updated.

Type: Timestamp

loggingConfig (p. 291)

The logging configuration.

Type: LoggingConfig (p. 476) object

maxJobDurationInSeconds (p. 291)

The maximum simulation job duration in seconds.

Type: Long

outputLocation (p. 291)

Simulation job output files location.

Type: OutputLocation (p. 478) object

robotApplications (p. 291)

The robot application used by the simulation job.

Type: Array of RobotApplicationConfig (p. 486) objects

Array Members: Fixed number of 1 item.

simulationApplications (p. 291)

The simulation application used by the simulation job.

Type: Array of SimulationApplicationConfig (p. 495) objects

Array Members: Fixed number of 1 item.

simulationTimeMillis (p. 291)

The simulation job execution duration in milliseconds.

Type: Long

status (p. 291)

The status of the simulation job.

Type: String

Valid Values: `Pending | Preparing | Running | Restarting | Completed | Failed | RunningFailed | Terminating | Terminated | Canceled`

tags (p. 291)

The list of all tags added to the simulation job.

Type: String to string map

Map Entries: Minimum number of 0 items. Maximum number of 50 items.

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. 291)

Information about the vpc configuration.

Type: VPCConfigResponse (p. 521) object

Errors

For information about the errors that are common to all actions, see Common Errors (p. 532).

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 V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

CreateWorldExportJob

Creates a world export job.

Request Syntax

```
POST /createWorldExportJob HTTP/1.1
Content-type: application/json

{
   "clientRequestToken": "string",
   "iamRole": "string",
   "outputLocation": {
      "s3Bucket": "string",
      "s3Prefix": "string"
   },
   "tags": {
      "string" : "string"
   },
   "worlds": [ "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. 298)

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

iamRole (p. 298)

The IAM role that the world export process uses to access the Amazon S3 bucket and put the export.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 255.

Pattern: `arn:aws:iam::\w+:role/.*`

Required: Yes

outputLocation (p. 298)

The output location.

Type: OutputLocation (p. 478) object

Required: Yes

tags (p. 298)

A map that contains tag keys and tag values that are attached to the world export job.

Type: String to string map

Map Entries: Minimum number of 0 items. Maximum number of 50 items.

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

worlds (p. 298)

A list of Amazon Resource Names (arns) that correspond to worlds to export.

Type: Array of strings

Array Members: Minimum number of 1 item. Maximum number of 100 items.

Length Constraints: Minimum length of 1. Maximum length of 1224.

Pattern: `arn:.*`

Required: Yes

Response Syntax

```
HTTP/1.1 200
Content-type: application/json

{
   "arn": "string",
   "clientRequestToken": "string",
   "createdAt": number,
   "failureCode": "string",
   "iamRole": "string",
   "outputLocation": {
      "s3Bucket": "string",
      "s3Prefix": "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. 299)

The Amazon Resource Name (ARN) of the world export job.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1224.

Pattern: `arn:.*`

clientRequestToken (p. 299)

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. 299)

The time, in milliseconds since the epoch, when the world export job was created.

Type: Timestamp

failureCode (p. 299)

The failure code of the world export job if it failed:

InternalServiceError

Internal service error.

LimitExceeded

The requested resource exceeds the maximum number allowed, or the number of concurrent stream requests exceeds the maximum number allowed.

ResourceNotFound

The specified resource could not be found.

RequestThrottled

The request was throttled.

InvalidInput

An input parameter in the request is not valid.

AllWorldGenerationFailed

All of the worlds in the world generation job failed. This can happen if your `worldCount` is greater than 50 or less than 1.

For more information about troubleshooting WorldForge, see Troubleshooting Simulation WorldForge.

Type: String

Valid Values: `InternalServiceError | LimitExceeded | ResourceNotFound | RequestThrottled | InvalidInput | AccessDenied`

iamRole (p. 299)

The IAM role that the world export process uses to access the Amazon S3 bucket and put the export.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 255.

Pattern: `arn:aws:iam::\w+:role/.*`

outputLocation (p. 299)

The output location.

Type: OutputLocation (p. 478) object

status (p. 299)

The status of the world export job.

Pending

The world export job request is pending.

Running

The world export job is running.

Completed

The world export job completed.

Failed

The world export job failed. See `failureCode` for more information.

Canceled

The world export job was cancelled.

Canceling

The world export job is being cancelled.

Type: String

Valid Values: `Pending | Running | Completed | Failed | Canceling | Canceled`

tags (p. 299)

A map that contains tag keys and tag values that are attached to the world export job.

Type: String to string map

Map Entries: Minimum number of 0 items. Maximum number of 50 items.

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. 532).

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

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 V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

CreateWorldGenerationJob

Creates worlds using the specified template.

Request Syntax

```
POST /createWorldGenerationJob HTTP/1.1
Content-type: application/json

{
   "clientRequestToken": "string",
   "tags": {
      "string" : "string"
   },
   "template": "string",
   "worldCount": {
      "floorplanCount": number,
      "interiorCountPerFloorplan": number
   },
   "worldTags": {
      "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. 303)

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

tags (p. 303)

A map that contains tag keys and tag values that are attached to the world generator job.

Type: String to string map

Map Entries: Minimum number of 0 items. Maximum number of 50 items.

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

template (p. 303)

The Amazon Resource Name (arn) of the world template describing the worlds you want to create.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1224.

Pattern: `arn:.*`

Required: Yes

worldCount (p. 303)

Information about the world count.

Type: WorldCount (p. 524) object

Required: Yes

worldTags (p. 303)

A map that contains tag keys and tag values that are attached to the generated worlds.

Type: String to string map

Map Entries: Minimum number of 0 items. Maximum number of 50 items.

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",
   "clientRequestToken": "string",
   "createdAt": number,
   "failureCode": "string",
   "status": "string",
   "tags": {
      "string" : "string"
   },
   "template": "string",
   "worldCount": {
      "floorplanCount": number,
      "interiorCountPerFloorplan": number
   },
   "worldTags": {
      "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. 304)

The Amazon Resource Name (ARN) of the world generator job.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1224.

Pattern: `arn:.*`

clientRequestToken (p. 304)

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. 304)

The time, in milliseconds since the epoch, when the world generator job was created.

Type: Timestamp

failureCode (p. 304)

The failure code of the world generator job if it failed:

InternalServiceError

Internal service error.

LimitExceeded

The requested resource exceeds the maximum number allowed, or the number of concurrent stream requests exceeds the maximum number allowed.

ResourceNotFound

The specified resource could not be found.

RequestThrottled

The request was throttled.

InvalidInput

An input parameter in the request is not valid.

Type: String

Valid Values: `InternalServiceError | LimitExceeded | ResourceNotFound | RequestThrottled | InvalidInput | AllWorldGenerationFailed`

status (p. 304)

The status of the world generator job.

Pending

The world generator job request is pending.

Running

The world generator job is running.

Completed

The world generator job completed.

Failed

The world generator job failed. See `failureCode` for more information.

PartialFailed

Some worlds did not generate.

Canceled

The world generator job was cancelled.

Canceling

The world generator job is being cancelled.

Type: String

Valid Values: `Pending | Running | Completed | Failed | PartialFailed | Canceling | Canceled`

tags (p. 304)

A map that contains tag keys and tag values that are attached to the world generator job.

Type: String to string map

Map Entries: Minimum number of 0 items. Maximum number of 50 items.

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 _.\-\/+=:]*`

template (p. 304)

The Amazon Resource Name (arn) of the world template.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1224.

Pattern: `arn:.*`

worldCount (p. 304)

Information about the world count.

Type: WorldCount (p. 524) object

worldTags (p. 304)

A map that contains tag keys and tag values that are attached to the generated worlds.

Type: String to string map

Map Entries: Minimum number of 0 items. Maximum number of 50 items.

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. 532).

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 V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

CreateWorldTemplate

Creates a world template.

Request Syntax

```
POST /createWorldTemplate HTTP/1.1
Content-type: application/json

{
   "clientRequestToken": "string",
   "name": "string",
   "tags": {
      "string" : "string"
   },
   "templateBody": "string",
   "templateLocation": {
      "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.

clientRequestToken (p. 309)

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

name (p. 309)

The name of the world template.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 255.

Pattern: `.*`

Required: No

tags (p. 309)

A map that contains tag keys and tag values that are attached to the world template.

Type: String to string map

Map Entries: Minimum number of 0 items. Maximum number of 50 items.

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

templateBody (p. 309)

The world template body.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 262144.

Pattern: `[\S\s]+`

Required: No

templateLocation (p. 309)

The location of the world template.

Type: TemplateLocation (p. 513) object

Required: No

Response Syntax

```
HTTP/1.1 200
Content-type: application/json

{
   "arn": "string",
   "clientRequestToken": "string",
   "createdAt": number,
   "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.

arn (p. 310)

The Amazon Resource Name (ARN) of the world template.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1224.

Pattern: `arn:.*`

clientRequestToken (p. 310)

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. 310)

The time, in milliseconds since the epoch, when the world template was created.

Type: Timestamp

name (p. 310)

The name of the world template.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 255.

Pattern: `.*`

tags (p. 310)

A map that contains tag keys and tag values that are attached to the world template.

Type: String to string map

Map Entries: Minimum number of 0 items. Maximum number of 50 items.

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. 532).

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

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 V2
- 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. 313)

The Amazon Resource Name (ARN) of the fleet.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1224.

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. 532).

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 V2
- 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. 315)

The Amazon Resource Name (ARN) of the robot.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1224.

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. 532).

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 V2
- 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

```
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. 317)

The Amazon Resource Name (ARN) of the the robot application.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1224.

Pattern: `arn:.*`

Required: Yes

applicationVersion (p. 317)

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. 532).

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 V2
- 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. 319)

The application information for the simulation application to delete.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1224.

Pattern: `arn:.*`

Required: Yes

applicationVersion (p. 319)

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. 532).

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 V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

DeleteWorldTemplate

Deletes a world template.

Request Syntax

```
POST /deleteWorldTemplate HTTP/1.1
Content-type: application/json

{
   "template": "string"
}
```

URI Request Parameters

The request does not use any URI parameters.

Request Body

The request accepts the following data in JSON format.

template (p. 321)

The Amazon Resource Name (arn) of the world template you want to delete.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1224.

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. 532).

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 V2
- 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. 323)

The Amazon Resource Name (ARN) of the fleet.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1224.

Pattern: `arn:.*`

Required: Yes

robot (p. 323)

The Amazon Resource Name (ARN) of the robot.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1224.

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. 323)

The Amazon Resource Name (ARN) of the fleet.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1224.

Pattern: `arn:.*`

robot (p. 323)

The Amazon Resource Name (ARN) of the robot.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1224.

Pattern: `arn:.*`

Errors

For information about the errors that are common to all actions, see Common Errors (p. 532).

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 V2
- 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. 326)

The Amazon Resource Name (ARN) of the deployment job.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1224.

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. 326)

The Amazon Resource Name (ARN) of the deployment job.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1224.

Pattern: `arn:.*`

createdAt (p. 326)

The time, in milliseconds since the epoch, when the deployment job was created.

Type: Timestamp

deploymentApplicationConfigs (p. 326)

The deployment application configuration.

Type: Array of DeploymentApplicationConfig (p. 460) objects

Array Members: Fixed number of 1 item.

deploymentConfig (p. 326)

The deployment configuration.

Type: DeploymentConfig (p. 461) object

failureCode (p. 326)

The deployment job failure code.

Type: String

Valid Values: `ResourceNotFound | EnvironmentSetupError | EtagMismatch | FailureThresholdBreached | RobotDeploymentAborted | RobotDeploymentNoResponse | RobotAgentConnectionTimeout | GreengrassDeploymentFailed | InvalidGreengrassGroup | MissingRobotArchitecture | MissingRobotApplicationArchitecture | MissingRobotDeploymentResource | GreengrassGroupVersionDoesNotExist | LambdaDeleted | ExtractingBundleFailure | PreLaunchFileFailure | PostLaunchFileFailure | BadPermissionError | DownloadConditionFailed | BadLambdaAssociated | InternalServerError | RobotApplicationDoesNotExist | DeploymentFleetDoesNotExist | FleetDeploymentTimeout`

failureReason (p. 326)

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. 326)

The Amazon Resource Name (ARN) of the fleet.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1224.

Pattern: `arn:.*`

robotDeploymentSummary (p. 326)

A list of robot deployment summaries.

Type: Array of RobotDeployment (p. 490) objects

status (p. 326)

The status of the deployment job.

Type: String

Valid Values: `Pending | Preparing | InProgress | Failed | Succeeded | Canceled`

tags (p. 326)

The list of all tags added to the specified deployment job.

Type: String to string map

Map Entries: Minimum number of 0 items. Maximum number of 50 items.

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. 532).

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 V2
- 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. 330)

The Amazon Resource Name (ARN) of the fleet.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1224.

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"
      }
   ],
   "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. 330)

The Amazon Resource Name (ARN) of the fleet.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1224.

Pattern: `arn:.*`

createdAt (p. 330)

The time, in milliseconds since the epoch, when the fleet was created.

Type: Timestamp

lastDeploymentJob (p. 330)

The Amazon Resource Name (ARN) of the last deployment job.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1224.

Pattern: `arn:.*`

lastDeploymentStatus (p. 330)

The status of the last deployment.

Type: String

Valid Values: `Pending | Preparing | InProgress | Failed | Succeeded | Canceled`

lastDeploymentTime (p. 330)

The time of the last deployment.

Type: Timestamp

name (p. 330)

The name of the fleet.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 255.

Pattern: `[a-zA-Z0-9_\-]*`

robots (p. 330)

A list of robots.

Type: Array of Robot (p. 484) objects

Array Members: Minimum number of 0 items. Maximum number of 1000 items.

tags (p. 330)

The list of all tags added to the specified fleet.

Type: String to string map

Map Entries: Minimum number of 0 items. Maximum number of 50 items.

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. 532).

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 V2
- 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. 334)

The Amazon Resource Name (ARN) of the robot to be described.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1224.

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. 334)

The target architecture of the robot application.

Type: String

Valid Values: `X86_64 | ARM64 | ARMHF`

arn (p. 334)

The Amazon Resource Name (ARN) of the robot.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1224.

Pattern: `arn:.*`

createdAt (p. 334)

The time, in milliseconds since the epoch, when the robot was created.

Type: Timestamp

fleetArn (p. 334)

The Amazon Resource Name (ARN) of the fleet.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1224.

Pattern: `arn:.*`

greengrassGroupId (p. 334)

The Greengrass group id.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1224.

Pattern: `.*`

lastDeploymentJob (p. 334)

The Amazon Resource Name (ARN) of the last deployment job.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1224.

Pattern: `arn:.*`

lastDeploymentTime (p. 334)

The time of the last deployment job.

Type: Timestamp

name (p. 334)

The name of the robot.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 255.

Pattern: `[a-zA-Z0-9_\-]*`

status (p. 334)

The status of the fleet.

Type: String

Valid Values: `Available | Registered | PendingNewDeployment | Deploying | Failed | InSync | NoResponse`

tags (p. 334)

The list of all tags added to the specified robot.

Type: String to string map

Map Entries: Minimum number of 0 items. Maximum number of 50 items.

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. 532).

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 V2
- 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. 338)

The Amazon Resource Name (ARN) of the robot application.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1224.

Pattern: `arn:.*`

Required: Yes

applicationVersion (p. 338)

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",
   "environment": {
      "uri": "string"
   },
   "imageDigest": "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. 338)

The Amazon Resource Name (ARN) of the robot application.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1224.

Pattern: `arn:.*`

environment (p. 338)

The object that contains the Docker image URI used to create the robot application.

Type: Environment (p. 466) object

imageDigest (p. 338)

A SHA256 identifier for the Docker image that you use for your robot application.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 72.

Pattern: `[Ss][Hh][Aa]256:[0-9a-fA-F]{64}`

lastUpdatedAt (p. 338)

The time, in milliseconds since the epoch, when the robot application was last updated.

Type: Timestamp

name (p. 338)

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. 338)

The revision id of the robot application.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 40.

Pattern: `[a-zA-Z0-9_.\-]*`

robotSoftwareSuite (p. 338)

The robot software suite (ROS distribution) used by the robot application.

Type: RobotSoftwareSuite (p. 492) object

sources (p. 338)

The sources of the robot application.

Type: Array of Source (p. 511) objects

tags (p. 338)

The list of all tags added to the specified robot application.

Type: String to string map

Map Entries: Minimum number of 0 items. Maximum number of 50 items.

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. 338)

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. 532).

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 V2
- 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. 342)

The application information for the simulation application.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1224.

Pattern: `arn:.*`

Required: Yes

applicationVersion (p. 342)

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",
   "environment": {
      "uri": "string"
   },
   "imageDigest": "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. 342)

The Amazon Resource Name (ARN) of the robot simulation application.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1224.

Pattern: `arn:.*`

environment (p. 342)

The object that contains the Docker image URI used to create the simulation application.

Type: Environment (p. 466) object

imageDigest (p. 342)

A SHA256 identifier for the Docker image that you use for your simulation application.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 72.

Pattern: `[Ss][Hh][Aa]256:[0-9a-fA-F]{64}`

lastUpdatedAt (p. 342)

The time, in milliseconds since the epoch, when the simulation application was last updated.

Type: Timestamp

name (p. 342)

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. 342)

The rendering engine for the simulation application.

Type: RenderingEngine (p. 483) object

revisionId (p. 342)

The revision id of the simulation application.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 40.

Pattern: `[a-zA-Z0-9_.\-]*`

robotSoftwareSuite (p. 342)

Information about the robot software suite (ROS distribution).

Type: RobotSoftwareSuite (p. 492) object

simulationSoftwareSuite (p. 342)

The simulation software suite used by the simulation application.

Type: SimulationSoftwareSuite (p. 510) object

sources (p. 342)

The sources of the simulation application.

Type: Array of Source (p. 511) objects

tags (p. 342)

The list of all tags added to the specified simulation application.

Type: String to string map

Map Entries: Minimum number of 0 items. Maximum number of 50 items.

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. 342)

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. 532).

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 V2
- 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. 346)

The Amazon Resource Name (ARN) of the simulation job to be described.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1224.

Pattern: `arn:.*`

Required: Yes

Response Syntax

```
HTTP/1.1 200
Content-type: application/json

{
   "arn": "string",
   "clientRequestToken": "string",
   "compute": {
      "computeType": "string",
      "gpuUnitLimit": number,
      "simulationUnitLimit": number
   },
   "dataSources": [
      {
         "destination": "string",
         "name": "string",
         "s3Bucket": "string",
         "s3Keys": [
            {
               "etag": "string",
               "s3Key": "string"
            }
         ],
         "type": "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": {
            "command": [ "string" ],
            "environmentVariables": {
               "string" : "string"
            },
            "launchFile": "string",
            "packageName": "string",
            "portForwardingConfig": {
               "portMappings": [
                  {
                     "applicationPort": number,
                     "enableOnPublicIp": boolean,
                     "jobPort": number
                  }
               ]
            },
            "streamUI": boolean
         },
         "tools": [
            {
               "command": "string",
               "exitBehavior": "string",
               "name": "string",
               "streamOutputToCloudWatch": boolean,
               "streamUI": boolean
            }
         ],
         "uploadConfigurations": [
            {
               "name": "string",
               "path": "string",
               "uploadBehavior": "string"
            }
         ],
         "useDefaultTools": boolean,
         "useDefaultUploadConfigurations": boolean
      }
   ],
   "simulationApplications": [
      {
         "application": "string",
         "applicationVersion": "string",
         "launchConfig": {
            "command": [ "string" ],
            "environmentVariables": {
               "string" : "string"
            },
            "launchFile": "string",
            "packageName": "string",
            "portForwardingConfig": {
               "portMappings": [
                  {
                     "applicationPort": number,
                     "enableOnPublicIp": boolean,
                     "jobPort": number
                  }
               ]
            },
            "streamUI": boolean
         },
         "tools": [
            {
               "command": "string",
               "exitBehavior": "string",
               "name": "string",
               "streamOutputToCloudWatch": boolean,
               "streamUI": boolean
            }
         ],
         "uploadConfigurations": [
            {
               "name": "string",
               "path": "string",
               "uploadBehavior": "string"
            }
         ],
         "useDefaultTools": boolean,
         "useDefaultUploadConfigurations": boolean,
         "worldConfigs": [
            {
               "world": "string"
            }
         ]
      }
   ],
   "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. 346)

The Amazon Resource Name (ARN) of the simulation job.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1224.

Pattern: `arn:.*`

clientRequestToken (p. 346)

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_\-=]*`

compute (p. 346)

Compute information for the simulation job.

Type: ComputeResponse (p. 455) object

dataSources (p. 346)

The data sources for the simulation job.

Type: Array of DataSource (p. 456) objects

failureBehavior (p. 346)

The failure behavior for the simulation job.

Type: String

Valid Values: `Fail | Continue`

failureCode (p. 346)

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 | RobotApplicationHealthCheckFailure | SimulationApplicationHealthCheckFailure | BadPermissionsRobotApplication | BadPermissionsSimulationApplication | BadPermissionsS3Object | BadPermissionsS3Output | BadPermissionsCloudwatchLogs | SubnetIpLimitExceeded | ENILimitExceeded | BadPermissionsUserCredentials | InvalidBundleRobotApplication | InvalidBundleSimulationApplication | InvalidS3Resource | ThrottlingError | LimitExceeded | MismatchedEtag | RobotApplicationVersionMismatchedEtag | SimulationApplicationVersionMismatchedEtag | ResourceNotFound | RequestThrottled | BatchTimedOut | BatchCanceled | InvalidInput | WrongRegionS3Bucket | WrongRegionS3Output | WrongRegionRobotApplication | WrongRegionSimulationApplication | UploadContentMismatchError`

failureReason (p. 346)

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. 346)

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.

Pattern: `arn:aws:iam::\w+:role/.*`

lastStartedAt (p. 346)

The time, in milliseconds since the epoch, when the simulation job was last started.

Type: Timestamp

lastUpdatedAt (p. 346)

The time, in milliseconds since the epoch, when the simulation job was last updated.

Type: Timestamp

loggingConfig (p. 346)

The logging configuration.

Type: LoggingConfig (p. 476) object

maxJobDurationInSeconds (p. 346)

The maximum job duration in seconds. The value must be 8 days (691,200 seconds) or less.

Type: Long

name (p. 346)

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. 346)

The network interface information for the simulation job.

Type: NetworkInterface (p. 477) object

outputLocation (p. 346)

Location for output files generated by the simulation job.

Type: OutputLocation (p. 478) object

robotApplications (p. 346)

A list of robot applications.

Type: Array of RobotApplicationConfig (p. 486) objects

Array Members: Fixed number of 1 item.

simulationApplications (p. 346)

A list of simulation applications.

Type: Array of SimulationApplicationConfig (p. 495) objects

Array Members: Fixed number of 1 item.

simulationTimeMillis (p. 346)

The simulation job execution duration in milliseconds.

Type: Long

status (p. 346)

The status of the simulation job.

Type: String

Valid Values: `Pending | Preparing | Running | Restarting | Completed | Failed | RunningFailed | Terminating | Terminated | Canceled`

tags (p. 346)

The list of all tags added to the specified simulation job.

Type: String to string map

Map Entries: Minimum number of 0 items. Maximum number of 50 items.

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. 346)

The VPC configuration.

Type: VPCConfigResponse (p. 521) object

Errors

For information about the errors that are common to all actions, see Common Errors (p. 532).

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 V2
- 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. 354)

The id of the batch to describe.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1224.

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",
         "computeType": "string",
         "dataSourceNames": [ "string" ],
         "lastUpdatedAt": number,
         "name": "string",
         "robotApplicationNames": [ "string" ],
         "simulationApplicationNames": [ "string" ],
         "status": "string"
      }
   ],
   "failedRequests": [
      {
         "failedAt": number,
         "failureCode": "string",
         "failureReason": "string",
         "request": {
            "compute": {
               "computeType": "string",
               "gpuUnitLimit": number,
               "simulationUnitLimit": number
            },
            "dataSources": [
               {
                  "destination": "string",
                  "name": "string",
                  "s3Bucket": "string",
                  "s3Keys": [ "string" ],
                  "type": "string"
               }
            ],
            "failureBehavior": "string",
            "iamRole": "string",
            "loggingConfig": {
               "recordAllRosTopics": boolean
            },
            "maxJobDurationInSeconds": number,
            "outputLocation": {
               "s3Bucket": "string",
               "s3Prefix": "string"
            },
            "robotApplications": [
               {
                  "application": "string",
                  "applicationVersion": "string",
                  "launchConfig": {
                     "command": [ "string" ],
                     "environmentVariables": {
                        "string" : "string"
                     },
                     "launchFile": "string",
                     "packageName": "string",
                     "portForwardingConfig": {
                        "portMappings": [
                           {
                              "applicationPort": number,
                              "enableOnPublicIp": boolean,
                              "jobPort": number
                           }
                        ]
                     },
                     "streamUI": boolean
                  },
                  "tools": [
                     {
                        "command": "string",
                        "exitBehavior": "string",
                        "name": "string",
                        "streamOutputToCloudWatch": boolean,
                        "streamUI": boolean
                     }
                  ],
                  "uploadConfigurations": [
                     {
                        "name": "string",
                        "path": "string",
                        "uploadBehavior": "string"
                     }
                  ],
                  "useDefaultTools": boolean,
                  "useDefaultUploadConfigurations": boolean
               }
            ],
            "simulationApplications": [
               {
                  "application": "string",
                  "applicationVersion": "string",
                  "launchConfig": {
                     "command": [ "string" ],
                     "environmentVariables": {
                        "string" : "string"
                     },
                     "launchFile": "string",
                     "packageName": "string",
                     "portForwardingConfig": {
                        "portMappings": [
                           {
                              "applicationPort": number,
                              "enableOnPublicIp": boolean,
                              "jobPort": number
                           }
                        ]
                     },
                     "streamUI": boolean
                  },
                  "tools": [
                     {
                        "command": "string",
                        "exitBehavior": "string",
                        "name": "string",
                        "streamOutputToCloudWatch": boolean,
                        "streamUI": boolean
                     }
                  ],
                  "uploadConfigurations": [
                     {
                        "name": "string",
                        "path": "string",
                        "uploadBehavior": "string"
                     }
                  ],
                  "useDefaultTools": boolean,
                  "useDefaultUploadConfigurations": boolean,
                  "worldConfigs": [
                     {
                        "world": "string"
                     }
                  ]
               }
            ],
            "tags": {
               "string" : "string"
            },
            "useDefaultApplications": boolean,
            "vpcConfig": {
               "assignPublicIp": boolean,
               "securityGroups": [ "string" ],
               "subnets": [ "string" ]
            }
         }
      }
   ],
   "failureCode": "string",
   "failureReason": "string",
   "lastUpdatedAt": number,
   "pendingRequests": [
      {
         "compute": {
            "computeType": "string",
            "gpuUnitLimit": number,
            "simulationUnitLimit": number
         },
         "dataSources": [
            {
               "destination": "string",
               "name": "string",
               "s3Bucket": "string",
               "s3Keys": [ "string" ],
               "type": "string"
            }
         ],
         "failureBehavior": "string",
         "iamRole": "string",
         "loggingConfig": {
            "recordAllRosTopics": boolean
         },
         "maxJobDurationInSeconds": number,
         "outputLocation": {
            "s3Bucket": "string",
            "s3Prefix": "string"
         },
         "robotApplications": [
            {
               "application": "string",
               "applicationVersion": "string",
               "launchConfig": {
                  "command": [ "string" ],
                  "environmentVariables": {
                     "string" : "string"
                  },
                  "launchFile": "string",
                  "packageName": "string",
                  "portForwardingConfig": {
                     "portMappings": [
                        {
                           "applicationPort": number,
                           "enableOnPublicIp": boolean,
                           "jobPort": number
                        }
                     ]
                  },
                  "streamUI": boolean
               },
               "tools": [
                  {
                     "command": "string",
                     "exitBehavior": "string",
                     "name": "string",
                     "streamOutputToCloudWatch": boolean,
                     "streamUI": boolean
                  }
               ],
               "uploadConfigurations": [
                  {
                     "name": "string",
                     "path": "string",
                     "uploadBehavior": "string"
                  }
               ],
               "useDefaultTools": boolean,
               "useDefaultUploadConfigurations": boolean
            }
         ],
         "simulationApplications": [
            {
               "application": "string",
               "applicationVersion": "string",
               "launchConfig": {
                  "command": [ "string" ],
                  "environmentVariables": {
                     "string" : "string"
                  },
                  "launchFile": "string",
                  "packageName": "string",
                  "portForwardingConfig": {
                     "portMappings": [
                        {
                           "applicationPort": number,
                           "enableOnPublicIp": boolean,
                           "jobPort": number
                        }
                     ]
                  },
                  "streamUI": boolean
               },
               "tools": [
                  {
                     "command": "string",
                     "exitBehavior": "string",
                     "name": "string",
                     "streamOutputToCloudWatch": boolean,
                     "streamUI": boolean
                  }
               ],
               "uploadConfigurations": [
                  {
                     "name": "string",
                     "path": "string",
                     "uploadBehavior": "string"
                  }
               ],
               "useDefaultTools": boolean,
               "useDefaultUploadConfigurations": boolean,
               "worldConfigs": [
                  {
                     "world": "string"
                  }
               ]
            }
         ],
         "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. 354)

The Amazon Resource Name (ARN) of the batch.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1224.

Pattern: `arn:.*`

batchPolicy (p. 354)

The batch policy.

Type: BatchPolicy (p. 453) object

clientRequestToken (p. 354)

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. 354)

The time, in milliseconds since the epoch, when the simulation job batch was created.

Type: Timestamp

createdRequests (p. 354)

A list of created simulation job summaries.

Type: Array of SimulationJobSummary (p. 508) objects

Array Members: Minimum number of 0 items. Maximum number of 100 items.

failedRequests (p. 354)

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. 467) objects

failureCode (p. 354)

The failure code of the simulation job batch.

Type: String

Valid Values: `InternalServiceError`

failureReason (p. 354)

The reason the simulation job batch failed.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 1024.

Pattern: `.*`

lastUpdatedAt (p. 354)

The time, in milliseconds since the epoch, when the simulation job batch was last updated.

Type: Timestamp

pendingRequests (p. 354)

A list of pending simulation job requests. These requests have not yet been created into simulation jobs.

Type: Array of SimulationJobRequest (p. 505) objects

Array Members: Minimum number of 1 item. Maximum number of 1000 items.

status (p. 354)

The status of the 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 `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. 354)

A map that contains tag keys and tag values that are attached to the simulation job batch.

Type: String to string map

Map Entries: Minimum number of 0 items. Maximum number of 50 items.

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. 532).

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 V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

DescribeWorld

Describes a world.

Request Syntax

```
POST /describeWorld HTTP/1.1
Content-type: application/json

{
   "world": "string"
}
```

URI Request Parameters

The request does not use any URI parameters.

Request Body

The request accepts the following data in JSON format.

world (p. 362)

The Amazon Resource Name (arn) of the world you want to describe.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1224.

Pattern: `arn:.*`

Required: Yes

Response Syntax

```
HTTP/1.1 200
Content-type: application/json

{
   "arn": "string",
   "createdAt": number,
   "generationJob": "string",
   "tags": {
      "string" : "string"
   },
   "template": "string",
   "worldDescriptionBody": "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. 362)

The Amazon Resource Name (arn) of the world.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1224.

Pattern: `arn:.*`

createdAt (p. 362)

The time, in milliseconds since the epoch, when the world was created.

Type: Timestamp

generationJob (p. 362)

The Amazon Resource Name (arn) of the world generation job that generated the world.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1224.

Pattern: `arn:.*`

tags (p. 362)

A map that contains tag keys and tag values that are attached to the world.

Type: String to string map

Map Entries: Minimum number of 0 items. Maximum number of 50 items.

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 _.\-\/+=:]*`

template (p. 362)

The world template.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1224.

Pattern: `arn:.*`

worldDescriptionBody (p. 362)

Returns the JSON formatted string that describes the contents of your world.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 262144.

Pattern: `[\S\s]+`

Errors

For information about the errors that are common to all actions, see Common Errors (p. 532).

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 V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

DescribeWorldExportJob

Describes a world export job.

Request Syntax

```
POST /describeWorldExportJob 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. 365)

The Amazon Resource Name (arn) of the world export job to describe.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1224.

Pattern: `arn:.*`

Required: Yes

Response Syntax

```
HTTP/1.1 200
Content-type: application/json

{
   "arn": "string",
   "clientRequestToken": "string",
   "createdAt": number,
   "failureCode": "string",
   "failureReason": "string",
   "iamRole": "string",
   "outputLocation": {
      "s3Bucket": "string",
      "s3Prefix": "string"
   },
   "status": "string",
   "tags": {
      "string" : "string"
   },
   "worlds": [ "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. 365)

The Amazon Resource Name (ARN) of the world export job.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1224.

Pattern: `arn:.*`

clientRequestToken (p. 365)

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. 365)

The time, in milliseconds since the epoch, when the world export job was created.

Type: Timestamp

failureCode (p. 365)

The failure code of the world export job if it failed:

InternalServiceError

Internal service error.

LimitExceeded

The requested resource exceeds the maximum number allowed, or the number of concurrent stream requests exceeds the maximum number allowed.

ResourceNotFound

The specified resource could not be found.

RequestThrottled

The request was throttled.

InvalidInput

An input parameter in the request is not valid.

Type: String

Valid Values: `InternalServiceError | LimitExceeded | ResourceNotFound | RequestThrottled | InvalidInput | AccessDenied`

failureReason (p. 365)

The reason why the world export job failed.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 1024.

Pattern: `.*`

iamRole (p. 365)

The IAM role that the world export process uses to access the Amazon S3 bucket and put the export.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 255.

Pattern: `arn:aws:iam::\w+:role/.*`

outputLocation (p. 365)

The output location.

Type: OutputLocation (p. 478) object

status (p. 365)

The status of the world export job.

Pending

The world export job request is pending.

Running

The world export job is running.

Completed

The world export job completed.

Failed

The world export job failed. See `failureCode` and `failureReason` for more information.

Canceled

The world export job was cancelled.

Canceling

The world export job is being cancelled.

Type: String

Valid Values: `Pending | Running | Completed | Failed | Canceling | Canceled`

tags (p. 365)

A map that contains tag keys and tag values that are attached to the world export job.

Type: String to string map

Map Entries: Minimum number of 0 items. Maximum number of 50 items.

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 _.\-\/+=:]*`

worlds (p. 365)

A list of Amazon Resource Names (arns) that correspond to worlds to be exported.

Type: Array of strings

Array Members: Minimum number of 1 item. Maximum number of 100 items.

Length Constraints: Minimum length of 1. Maximum length of 1224.

Pattern: `arn:.*`

Errors

For information about the errors that are common to all actions, see Common Errors (p. 532).

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 V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

DescribeWorldGenerationJob

Describes a world generation job.

Request Syntax

```
POST /describeWorldGenerationJob 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. 369)

The Amazon Resource Name (arn) of the world generation job to describe.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1224.

Pattern: `arn:.*`

Required: Yes

Response Syntax

```
HTTP/1.1 200
Content-type: application/json

{
   "arn": "string",
   "clientRequestToken": "string",
   "createdAt": number,
   "failureCode": "string",
   "failureReason": "string",
   "finishedWorldsSummary": {
      "failureSummary": {
         "failures": [
            {
               "failureCode": "string",
               "failureCount": number,
               "sampleFailureReason": "string"
            }
         ],
         "totalFailureCount": number
      },
      "finishedCount": number,
      "succeededWorlds": [ "string" ]
   },
   "status": "string",
   "tags": {
      "string" : "string"
   },
   "template": "string",
   "worldCount": {
      "floorplanCount": number,
      "interiorCountPerFloorplan": number
   },
   "worldTags": {
      "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. 369)

The Amazon Resource Name (ARN) of the world generation job.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1224.

Pattern: `arn:.*`

clientRequestToken (p. 369)

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. 369)

The time, in milliseconds since the epoch, when the world generation job was created.

Type: Timestamp

failureCode (p. 369)

The failure code of the world generation job if it failed:

InternalServiceError

Internal service error.

LimitExceeded

The requested resource exceeds the maximum number allowed, or the number of concurrent stream requests exceeds the maximum number allowed.

ResourceNotFound

The specified resource could not be found.

RequestThrottled

The request was throttled.

InvalidInput

An input parameter in the request is not valid.

Type: String

Valid Values: `InternalServiceError | LimitExceeded | ResourceNotFound | RequestThrottled | InvalidInput | AllWorldGenerationFailed`

failureReason (p. 369)

The reason why the world generation job failed.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 1024.

Pattern: `.*`

finishedWorldsSummary (p. 369)

Summary information about finished worlds.

Type: FinishedWorldsSummary (p. 471) object

status (p. 369)

The status of the world generation job:

Pending

The world generation job request is pending.

Running

The world generation job is running.

Completed

The world generation job completed.

Failed

The world generation job failed. See `failureCode` for more information.

PartialFailed

Some worlds did not generate.

Canceled

The world generation job was cancelled.

Canceling

The world generation job is being cancelled.

Type: String

Valid Values: `Pending | Running | Completed | Failed | PartialFailed | Canceling | Canceled`

tags (p. 369)

A map that contains tag keys and tag values that are attached to the world generation job.

Type: String to string map

Map Entries: Minimum number of 0 items. Maximum number of 50 items.

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 _.\-\/+=:]*`

template (p. 369)

The Amazon Resource Name (arn) of the world template.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1224.

Pattern: `arn:.*`

worldCount (p. 369)

Information about the world count.

Type: WorldCount (p. 524) object

worldTags (p. 369)

A map that contains tag keys and tag values that are attached to the generated worlds.

Type: String to string map

Map Entries: Minimum number of 0 items. Maximum number of 50 items.

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. 532).

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 V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

DescribeWorldTemplate

Describes a world template.

Request Syntax

```
POST /describeWorldTemplate HTTP/1.1
Content-type: application/json

{
   "template": "string"
}
```

URI Request Parameters

The request does not use any URI parameters.

Request Body

The request accepts the following data in JSON format.

template (p. 374)

The Amazon Resource Name (arn) of the world template you want to describe.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1224.

Pattern: `arn:.*`

Required: Yes

Response Syntax

```
HTTP/1.1 200
Content-type: application/json

{
   "arn": "string",
   "clientRequestToken": "string",
   "createdAt": number,
   "lastUpdatedAt": number,
   "name": "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. 374)

The Amazon Resource Name (ARN) of the world template.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1224.

Pattern: `arn:.*`

clientRequestToken (p. 374)

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. 374)

The time, in milliseconds since the epoch, when the world template was created.

Type: Timestamp

lastUpdatedAt (p. 374)

The time, in milliseconds since the epoch, when the world template was last updated.

Type: Timestamp

name (p. 374)

The name of the world template.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 255.

Pattern: `.*`

tags (p. 374)

A map that contains tag keys and tag values that are attached to the world template.

Type: String to string map

Map Entries: Minimum number of 0 items. Maximum number of 50 items.

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. 374)

The version of the world template that you're using.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 1024.

Pattern: `.*`

Errors

For information about the errors that are common to all actions, see Common Errors (p. 532).

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 V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

GetWorldTemplateBody

Gets the world template body.

Request Syntax

```
POST /getWorldTemplateBody HTTP/1.1
Content-type: application/json

{
   "generationJob": "string",
   "template": "string"
}
```

URI Request Parameters

The request does not use any URI parameters.

Request Body

The request accepts the following data in JSON format.

generationJob (p. 377)

The Amazon Resource Name (arn) of the world generator job.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1224.

Pattern: `arn:.*`

Required: No

template (p. 377)

The Amazon Resource Name (arn) of the world template.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1224.

Pattern: `arn:.*`

Required: No

Response Syntax

```
HTTP/1.1 200
Content-type: application/json

{
   "templateBody": "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.

templateBody (p. 377)

The world template body.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 262144.

Pattern: `[\S\s]+`

Errors

For information about the errors that are common to all actions, see Common Errors (p. 532).

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 V2
- 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. 379)

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. 470) objects

Array Members: Fixed number of 1 item.

Required: No

maxResults (p. 379)

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. 379)

If the previous paginated request did not return all of the remaining results, the response object's `nextToken` parameter value is set to a token. To retrieve the next set of results, call `ListDeploymentJobs` again and assign that token to the request object's `nextToken` parameter. If there are no remaining results, the previous response object's NextToken parameter is set to null.

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. 380)

A list of deployment jobs that meet the criteria of the request.

Type: Array of DeploymentJob (p. 462) objects

Array Members: Minimum number of 0 items. Maximum number of 200 items.

nextToken (p. 380)

If the previous paginated request did not return all of the remaining results, the response object's `nextToken` parameter value is set to a token. To retrieve the next set of results, call `ListDeploymentJobs` again and assign that token to the request object's `nextToken` parameter. If there are no remaining results, the previous response object's NextToken parameter is set to null.

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. 532).

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 V2
- 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. 383)

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. 470) objects

Array Members: Fixed number of 1 item.

Required: No

maxResults (p. 383)

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. 383)

If the previous paginated request did not return all of the remaining results, the response object's `nextToken` parameter value is set to a token. To retrieve the next set of results, call `ListFleets` again and assign that token to the request object's `nextToken` parameter. If there are no remaining results, the previous response object's NextToken parameter is set to null.

> **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. 384)

A list of fleet details meeting the request criteria.

Type: Array of Fleet (p. 472) objects

Array Members: Minimum number of 0 items. Maximum number of 200 items.

nextToken (p. 384)

If the previous paginated request did not return all of the remaining results, the response object's `nextToken` parameter value is set to a token. To retrieve the next set of results, call `ListFleets` again and assign that token to the request object's `nextToken` parameter. If there are no remaining results, the previous response object's NextToken parameter is set to null.

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. 532).

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 V2
- 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

```
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. 386)

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. 470) objects

Array Members: Fixed number of 1 item.

Required: No

maxResults (p. 386)

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. 386)

If the previous paginated request did not return all of the remaining results, the response object's `nextToken` parameter value is set to a token. To retrieve the next set of results, call

`ListRobotApplications` again and assign that token to the request object's `nextToken` parameter. If there are no remaining results, the previous response object's NextToken parameter is set to null.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2048.

Pattern: `[a-zA-Z0-9_.\-\/+=]*`

Required: No

versionQualifier (p. 386)

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. 387)

If the previous paginated request did not return all of the remaining results, the response object's `nextToken` parameter value is set to a token. To retrieve the next set of results, call `ListRobotApplications` again and assign that token to the request object's `nextToken` parameter. If there are no remaining results, the previous response object's NextToken parameter is set to null.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2048.

Pattern: `[a-zA-Z0-9_.\-\/+=]*`

robotApplicationSummaries (p. 387)

A list of robot application summaries that meet the criteria of the request.

Type: Array of RobotApplicationSummary (p. 488) 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. 532).

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 V2
- 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. 389)

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. 470) objects

Array Members: Fixed number of 1 item.

Required: No

maxResults (p. 389)

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. 389)

If the previous paginated request did not return all of the remaining results, the response object's `nextToken` parameter value is set to a token. To retrieve the next set of results, call `ListRobots` again and assign that token to the request object's `nextToken` parameter. If there are no remaining results, the previous response object's NextToken parameter is set to null.

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. 390)

If the previous paginated request did not return all of the remaining results, the response object's `nextToken` parameter value is set to a token. To retrieve the next set of results, call `ListRobots` again and assign that token to the request object's `nextToken` parameter. If there are no remaining results, the previous response object's NextToken parameter is set to null.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2048.

Pattern: `[a-zA-Z0-9_.\-\/+=]*`

robots (p. 390)

A list of robots that meet the criteria of the request.

Type: Array of Robot (p. 484) 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. 532).

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 V2
- 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. 392)

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. 470) objects

Array Members: Fixed number of 1 item.

Required: No

maxResults (p. 392)

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. 392)

If the previous paginated request did not return all of the remaining results, the response object's `nextToken` parameter value is set to a token. To retrieve the next set of results, call `ListSimulationApplications` again and assign that token to the request object's `nextToken` parameter. If there are no remaining results, the previous response object's NextToken parameter is set to null.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2048.

Pattern: `[a-zA-Z0-9_.\-\/+=]*`

Required: No

versionQualifier (p. 392)

The version qualifier of the simulation 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",
   "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. 393)

If the previous paginated request did not return all of the remaining results, the response object's `nextToken` parameter value is set to a token. To retrieve the next set of results, call `ListSimulationApplications` again and assign that token to the request object's `nextToken` parameter. If there are no remaining results, the previous response object's NextToken parameter is set to null.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2048.

Pattern: `[a-zA-Z0-9_.\-\/+=]*`

simulationApplicationSummaries (p. 393)

A list of simulation application summaries that meet the criteria of the request.

Type: Array of SimulationApplicationSummary (p. 497) 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. 532).

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 V2
- 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

```
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. 395)

Optional filters to limit results.

Type: Array of Filter (p. 470) objects

Array Members: Fixed number of 1 item.

Required: No

maxResults (p. 395)

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. 395)

If the previous paginated request did not return all of the remaining results, the response object's `nextToken` parameter value is set to a token. To retrieve the next set of results, call `ListSimulationJobBatches` again and assign that token to the request object's `nextToken` parameter. If there are no remaining results, the previous response object's NextToken parameter is set to null.

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",
   "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. 396)

If the previous paginated request did not return all of the remaining results, the response object's `nextToken` parameter value is set to a token. To retrieve the next set of results, call `ListSimulationJobBatches` again and assign that token to the request object's `nextToken` parameter. If there are no remaining results, the previous response object's NextToken parameter is set to null.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2048.

Pattern: `[a-zA-Z0-9_.\-\/+=]*`

simulationJobBatchSummaries (p. 396)

A list of simulation job batch summaries.

Type: Array of SimulationJobBatchSummary (p. 503) objects

Errors

For information about the errors that are common to all actions, see Common Errors (p. 532).

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 V2
- 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. 398)

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. 470) objects

Array Members: Fixed number of 1 item.

Required: No

maxResults (p. 398)

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. 398)

If the previous paginated request did not return all of the remaining results, the response object's `nextToken` parameter value is set to a token. To retrieve the next set of results, call `ListSimulationJobs` again and assign that token to the request object's `nextToken` parameter. If there are no remaining results, the previous response object's NextToken parameter is set to null.

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",
   "simulationJobSummaries": [
      {
         "arn": "string",
         "computeType": "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. 399)

If the previous paginated request did not return all of the remaining results, the response object's `nextToken` parameter value is set to a token. To retrieve the next set of results, call `ListSimulationJobs` again and assign that token to the request object's `nextToken` parameter. If there are no remaining results, the previous response object's NextToken parameter is set to null.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2048.

Pattern: `[a-zA-Z0-9_.\-\/+=]*`

simulationJobSummaries (p. 399)

A list of simulation job summaries that meet the criteria of the request.

Type: Array of SimulationJobSummary (p. 508) 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. 532).

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 V2
- 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 uses the following URI parameters.

resourceArn (p. 401)

The AWS RoboMaker Amazon Resource Name (ARN) with tags to be listed.

Length Constraints: Minimum length of 1. Maximum length of 1224.

Pattern: `arn:.*`

Required: Yes

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. 401)

The list of all tags added to the specified resource.

Type: String to string map

Map Entries: Minimum number of 0 items. Maximum number of 50 items.

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. 532).

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 V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

ListWorldExportJobs

Lists world export jobs.

Request Syntax

```
POST /listWorldExportJobs 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. 403)

Optional filters to limit results. You can use `generationJobId` and `templateId`.

Type: Array of Filter (p. 470) objects

Array Members: Fixed number of 1 item.

Required: No

maxResults (p. 403)

When this parameter is used, `ListWorldExportJobs` 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 `ListWorldExportJobs` request with the returned `nextToken` value. This value can be between 1 and 100. If this parameter is not used, then `ListWorldExportJobs` returns up to 100 results and a `nextToken` value if applicable.

Type: Integer

Required: No

nextToken (p. 403)

If the previous paginated request did not return all of the remaining results, the response object's `nextToken` parameter value is set to a token. To retrieve the next set of results, call `ListWorldExportJobs` again and assign that token to the request object's `nextToken` parameter. If there are no remaining results, the previous response object's NextToken parameter is set to null.

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",
   "worldExportJobSummaries": [
      {
         "arn": "string",
         "createdAt": number,
         "status": "string",
         "worlds": [ "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. 404)

If the previous paginated request did not return all of the remaining results, the response object's `nextToken` parameter value is set to a token. To retrieve the next set of results, call `ListWorldExportJobsRequest` again and assign that token to the request object's `nextToken` parameter. If there are no remaining results, the previous response object's NextToken parameter is set to null.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2048.

Pattern: `[a-zA-Z0-9_.\-\/+=]*`

worldExportJobSummaries (p. 404)

Summary information for world export jobs.

Type: Array of WorldExportJobSummary (p. 525) 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. 532).

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 V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

ListWorldGenerationJobs

Lists world generator jobs.

Request Syntax

```
POST /listWorldGenerationJobs 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. 406)

Optional filters to limit results. You can use `status` and `templateId`.

Type: Array of Filter (p. 470) objects

Array Members: Fixed number of 1 item.

Required: No

maxResults (p. 406)

When this parameter is used, `ListWorldGeneratorJobs` 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 `ListWorldGeneratorJobs` request with the returned `nextToken` value. This value can be between 1 and 100. If this parameter is not used, then `ListWorldGeneratorJobs` returns up to 100 results and a `nextToken` value if applicable.

Type: Integer

Required: No

nextToken (p. 406)

If the previous paginated request did not return all of the remaining results, the response object's `nextToken` parameter value is set to a token. To retrieve the next set of results, call `ListWorldGenerationJobsRequest` again and assign that token to the request object's `nextToken` parameter. If there are no remaining results, the previous response object's NextToken parameter is set to null.

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",
   "worldGenerationJobSummaries": [
      {
         "arn": "string",
         "createdAt": number,
         "failedWorldCount": number,
         "status": "string",
         "succeededWorldCount": number,
         "template": "string",
         "worldCount": {
            "floorplanCount": number,
            "interiorCountPerFloorplan": number
         }
      }
   ]
}
```

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. 407)

If the previous paginated request did not return all of the remaining results, the response object's `nextToken` parameter value is set to a token. To retrieve the next set of results, call `ListWorldGeneratorJobsRequest` again and assign that token to the request object's `nextToken` parameter. If there are no remaining results, the previous response object's NextToken parameter is set to null.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2048.

Pattern: `[a-zA-Z0-9_.\-\/+=]*`

worldGenerationJobSummaries (p. 407)

Summary information for world generator jobs.

Type: Array of WorldGenerationJobSummary (p. 529) 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. 532).

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 V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

ListWorlds

Lists worlds.

Request Syntax

```
POST /listWorlds 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. 409)

Optional filters to limit results. You can use `status`.

Type: Array of Filter (p. 470) objects

Array Members: Fixed number of 1 item.

Required: No

maxResults (p. 409)

When this parameter is used, `ListWorlds` 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 `ListWorlds` request with the returned `nextToken` value. This value can be between 1 and 100. If this parameter is not used, then `ListWorlds` returns up to 100 results and a `nextToken` value if applicable.

Type: Integer

Required: No

nextToken (p. 409)

If the previous paginated request did not return all of the remaining results, the response object's `nextToken` parameter value is set to a token. To retrieve the next set of results, call `ListWorlds` again and assign that token to the request object's `nextToken` parameter. If there are no remaining results, the previous response object's NextToken parameter is set to null.

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",
   "worldSummaries": [
      {
         "arn": "string",
         "createdAt": number,
         "generationJob": "string",
         "template": "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. 410)

If the previous paginated request did not return all of the remaining results, the response object's `nextToken` parameter value is set to a token. To retrieve the next set of results, call `ListWorlds` again and assign that token to the request object's `nextToken` parameter. If there are no remaining results, the previous response object's NextToken parameter is set to null.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2048.

Pattern: `[a-zA-Z0-9_.\-\/+=]*`

worldSummaries (p. 410)

Summary information for worlds.

Type: Array of WorldSummary (p. 531) objects

Errors

For information about the errors that are common to all actions, see Common Errors (p. 532).

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 V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

ListWorldTemplates

Lists world templates.

Request Syntax

```
POST /listWorldTemplates HTTP/1.1
Content-type: application/json

{
   "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.

maxResults (p. 412)

When this parameter is used, `ListWorldTemplates` 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 `ListWorldTemplates` request with the returned `nextToken` value. This value can be between 1 and 100. If this parameter is not used, then `ListWorldTemplates` returns up to 100 results and a `nextToken` value if applicable.

Type: Integer

Required: No

nextToken (p. 412)

If the previous paginated request did not return all of the remaining results, the response object's `nextToken` parameter value is set to a token. To retrieve the next set of results, call `ListWorldTemplates` again and assign that token to the request object's `nextToken` parameter. If there are no remaining results, the previous response object's NextToken parameter is set to null.

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",
   "templateSummaries": [
      {
         "arn": "string",
         "createdAt": number,
         "lastUpdatedAt": number,
         "name": "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. 412)

If the previous paginated request did not return all of the remaining results, the response object's `nextToken` parameter value is set to a token. To retrieve the next set of results, call `ListWorldTemplates` again and assign that token to the request object's `nextToken` parameter. If there are no remaining results, the previous response object's NextToken parameter is set to null.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2048.

Pattern: `[a-zA-Z0-9_.\-\/+=]*`

templateSummaries (p. 412)

Summary information for templates.

Type: Array of TemplateSummary (p. 514) objects

Errors

For information about the errors that are common to all actions, see Common Errors (p. 532).

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 V2
- 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. 415)

The Amazon Resource Name (ARN) of the fleet.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1224.

Pattern: `arn:.*`

Required: Yes

robot (p. 415)

The Amazon Resource Name (ARN) of the robot.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1224.

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. 415)

The Amazon Resource Name (ARN) of the fleet that the robot will join.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1224.

Pattern: `arn:.*`

robot (p. 415)

Information about the robot registration.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1224.

Pattern: `arn:.*`

Errors

For information about the errors that are common to all actions, see Common Errors (p. 532).

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 V2
- 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. 418)

The Amazon Resource Name (ARN) of the simulation job.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1224.

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. 532).

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 V2
- 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

```
POST /startSimulationJobBatch HTTP/1.1
Content-type: application/json

{
   "batchPolicy": {
      "maxConcurrency": number,
      "timeoutInSeconds": number
   },
   "clientRequestToken": "string",
   "createSimulationJobRequests": [
      {
         "compute": {
            "computeType": "string",
            "gpuUnitLimit": number,
            "simulationUnitLimit": number
         },
         "dataSources": [
            {
               "destination": "string",
               "name": "string",
               "s3Bucket": "string",
               "s3Keys": [ "string" ],
               "type": "string"
            }
         ],
         "failureBehavior": "string",
         "iamRole": "string",
         "loggingConfig": {
            "recordAllRosTopics": boolean
         },
         "maxJobDurationInSeconds": number,
         "outputLocation": {
            "s3Bucket": "string",
            "s3Prefix": "string"
         },
         "robotApplications": [
            {
               "application": "string",
               "applicationVersion": "string",
               "launchConfig": {
                  "command": [ "string" ],
                  "environmentVariables": {
                     "string" : "string"
                  },
                  "launchFile": "string",
                  "packageName": "string",
                  "portForwardingConfig": {
                     "portMappings": [
                        {
                           "applicationPort": number,
                           "enableOnPublicIp": boolean,
                           "jobPort": number
                        }
                     ]
                  },
                  "streamUI": boolean
               },
               "tools": [
                  {
                     "command": "string",
                     "exitBehavior": "string",
                     "name": "string",
                     "streamOutputToCloudWatch": boolean,
                     "streamUI": boolean
                  }
               ],
               "uploadConfigurations": [
                  {
                     "name": "string",
                     "path": "string",
                     "uploadBehavior": "string"
                  }
               ],
               "useDefaultTools": boolean,
               "useDefaultUploadConfigurations": boolean
            }
         ],
         "simulationApplications": [
            {
               "application": "string",
               "applicationVersion": "string",
               "launchConfig": {
                  "command": [ "string" ],
                  "environmentVariables": {
                     "string" : "string"
                  },
                  "launchFile": "string",
                  "packageName": "string",
                  "portForwardingConfig": {
                     "portMappings": [
                        {
                           "applicationPort": number,
                           "enableOnPublicIp": boolean,
                           "jobPort": number
                        }
                     ]
                  },
                  "streamUI": boolean
               },
               "tools": [
                  {
                     "command": "string",
                     "exitBehavior": "string",
                     "name": "string",
                     "streamOutputToCloudWatch": boolean,
                     "streamUI": boolean
                  }
               ],
               "uploadConfigurations": [
                  {
                     "name": "string",
                     "path": "string",
                     "uploadBehavior": "string"
                  }
               ],
               "useDefaultTools": boolean,
               "useDefaultUploadConfigurations": boolean,
               "worldConfigs": [
                  {
                     "world": "string"
                  }
               ]
            }
         ],
         "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. 420)

The batch policy.

Type: BatchPolicy (p. 453) object

Required: No

clientRequestToken (p. 420)

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. 420)

A list of simulation job requests to create in the batch.

Type: Array of SimulationJobRequest (p. 505) objects

Array Members: Minimum number of 1 item. Maximum number of 1000 items.

Required: Yes

tags (p. 420)

A map that contains tag keys and tag values that are attached to the deployment job batch.

Type: String to string map

Map Entries: Minimum number of 0 items. Maximum number of 50 items.

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",
   "batchPolicy": {
      "maxConcurrency": number,
      "timeoutInSeconds": number
   },
   "clientRequestToken": "string",
   "createdAt": number,
   "createdRequests": [
      {
         "arn": "string",
         "computeType": "string",
         "dataSourceNames": [ "string" ],
         "lastUpdatedAt": number,
         "name": "string",
         "robotApplicationNames": [ "string" ],
         "simulationApplicationNames": [ "string" ],
         "status": "string"
      }
   ],
   "failedRequests": [
      {
         "failedAt": number,
         "failureCode": "string",
         "failureReason": "string",
         "request": {
            "compute": {
               "computeType": "string",
               "gpuUnitLimit": number,
               "simulationUnitLimit": number
            },
            "dataSources": [
               {
                  "destination": "string",
                  "name": "string",
                  "s3Bucket": "string",
                  "s3Keys": [ "string" ],
                  "type": "string"
               }
            ],
            "failureBehavior": "string",
            "iamRole": "string",
            "loggingConfig": {
               "recordAllRosTopics": boolean
            },
            "maxJobDurationInSeconds": number,
            "outputLocation": {
               "s3Bucket": "string",
               "s3Prefix": "string"
            },
            "robotApplications": [
               {
                  "application": "string",
                  "applicationVersion": "string",
                  "launchConfig": {
                     "command": [ "string" ],
                     "environmentVariables": {
                        "string" : "string"
                     },
                     "launchFile": "string",
                     "packageName": "string",
                     "portForwardingConfig": {
                        "portMappings": [
                           {
                              "applicationPort": number,
                              "enableOnPublicIp": boolean,
                              "jobPort": number
                           }
                        ]
                     },
                     "streamUI": boolean
                  },
                  "tools": [
                     {
                        "command": "string",
                        "exitBehavior": "string",
                        "name": "string",
                        "streamOutputToCloudWatch": boolean,
                        "streamUI": boolean
                     }
                  ],
                  "uploadConfigurations": [
                     {
                        "name": "string",
                        "path": "string",
                        "uploadBehavior": "string"
                     }
                  ],
                  "useDefaultTools": boolean,
                  "useDefaultUploadConfigurations": boolean
               }
            ],
            "simulationApplications": [
               {
                  "application": "string",
                  "applicationVersion": "string",
                  "launchConfig": {
                     "command": [ "string" ],
                     "environmentVariables": {
                        "string" : "string"
                     },
                     "launchFile": "string",
                     "packageName": "string",
                     "portForwardingConfig": {
                        "portMappings": [
                           {
                              "applicationPort": number,
                              "enableOnPublicIp": boolean,
                              "jobPort": number
                           }
                        ]
                     },
                     "streamUI": boolean
                  },
                  "tools": [
                     {
                        "command": "string",
                        "exitBehavior": "string",
                        "name": "string",
                        "streamOutputToCloudWatch": boolean,
                        "streamUI": boolean
                     }
                  ],
                  "uploadConfigurations": [
                     {
                        "name": "string",
                        "path": "string",
                        "uploadBehavior": "string"
                     }
                  ],
                  "useDefaultTools": boolean,
                  "useDefaultUploadConfigurations": boolean,
                  "worldConfigs": [
                     {
                        "world": "string"
                     }
                  ]
               }
            ],
            "tags": {
               "string" : "string"
            },
            "useDefaultApplications": boolean,
            "vpcConfig": {
               "assignPublicIp": boolean,
               "securityGroups": [ "string" ],
               "subnets": [ "string" ]
            }
         }
      }
   ],
   "failureCode": "string",
   "failureReason": "string",
   "pendingRequests": [
      {
         "compute": {
            "computeType": "string",
            "gpuUnitLimit": number,
            "simulationUnitLimit": number
         },
         "dataSources": [
            {
               "destination": "string",
               "name": "string",
               "s3Bucket": "string",
               "s3Keys": [ "string" ],
               "type": "string"
            }
         ],
         "failureBehavior": "string",
         "iamRole": "string",
         "loggingConfig": {
            "recordAllRosTopics": boolean
         },
         "maxJobDurationInSeconds": number,
         "outputLocation": {
            "s3Bucket": "string",
            "s3Prefix": "string"
         },
         "robotApplications": [
            {
               "application": "string",
               "applicationVersion": "string",
               "launchConfig": {
                  "command": [ "string" ],
                  "environmentVariables": {
                     "string" : "string"
                  },
                  "launchFile": "string",
                  "packageName": "string",
                  "portForwardingConfig": {
                     "portMappings": [
                        {
                           "applicationPort": number,
                           "enableOnPublicIp": boolean,
                           "jobPort": number
                        }
                     ]
                  },
                  "streamUI": boolean
               },
               "tools": [
                  {
                     "command": "string",
                     "exitBehavior": "string",
                     "name": "string",
                     "streamOutputToCloudWatch": boolean,
                     "streamUI": boolean
                  }
               ],
               "uploadConfigurations": [
                  {
                     "name": "string",
                     "path": "string",
                     "uploadBehavior": "string"
                  }
               ],
               "useDefaultTools": boolean,
               "useDefaultUploadConfigurations": boolean
            }
         ],
         "simulationApplications": [
            {
               "application": "string",
               "applicationVersion": "string",
               "launchConfig": {
                  "command": [ "string" ],
                  "environmentVariables": {
                     "string" : "string"
                  },
                  "launchFile": "string",
                  "packageName": "string",
                  "portForwardingConfig": {
                     "portMappings": [
                        {
                           "applicationPort": number,
                           "enableOnPublicIp": boolean,
                           "jobPort": number
                        }
                     ]
                  },
                  "streamUI": boolean
               },
               "tools": [
                  {
                     "command": "string",
                     "exitBehavior": "string",
                     "name": "string",
                     "streamOutputToCloudWatch": boolean,
                     "streamUI": boolean
                  }
               ],
               "uploadConfigurations": [
                  {
                     "name": "string",
                     "path": "string",
                     "uploadBehavior": "string"
                  }
               ],
               "useDefaultTools": boolean,
               "useDefaultUploadConfigurations": boolean,
               "worldConfigs": [
                  {
                     "world": "string"
                  }
               ]
            }
         ],
         "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. 423)

The Amazon Resource Name (arn) of the batch.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1224.

Pattern: `arn:.*`

batchPolicy (p. 423)

The batch policy.

Type: BatchPolicy (p. 453) object

clientRequestToken (p. 423)

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. 423)

The time, in milliseconds since the epoch, when the simulation job batch was created.

Type: Timestamp

createdRequests (p. 423)

A list of created simulation job request summaries.

Type: Array of SimulationJobSummary (p. 508) objects

Array Members: Minimum number of 0 items. Maximum number of 100 items.

failedRequests (p. 423)

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. 467) objects

failureCode (p. 423)

The failure code if the simulation job batch failed.

Type: String

Valid Values: `InternalServiceError`

failureReason (p. 423)

The reason the simulation job batch failed.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 1024.

Pattern: `.*`

pendingRequests (p. 423)

A list of pending simulation job requests. These requests have not yet been created into simulation jobs.

Type: Array of SimulationJobRequest (p. 505) objects

Array Members: Minimum number of 1 item. Maximum number of 1000 items.

status (p. 423)

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 `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. 423)

A map that contains tag keys and tag values that are attached to the deployment job batch.

Type: String to string map

Map Entries: Minimum number of 0 items. Maximum number of 50 items.

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. 532).

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 V2
- 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. 431)

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. 431)

The target fleet for the synchronization.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1224.

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. 431)

The Amazon Resource Name (ARN) of the synchronization request.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1224.

Pattern: `arn:.*`

createdAt (p. 431)

The time, in milliseconds since the epoch, when the fleet was created.

Type: Timestamp

deploymentApplicationConfigs (p. 431)

Information about the deployment application configurations.

Type: Array of DeploymentApplicationConfig (p. 460) objects

Array Members: Fixed number of 1 item.

deploymentConfig (p. 431)

Information about the deployment configuration.

Type: DeploymentConfig (p. 461) object

failureCode (p. 431)

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 |`

`BadLambdaAssociated | InternalServerError | RobotApplicationDoesNotExist | DeploymentFleetDoesNotExist | FleetDeploymentTimeout`

failureReason (p. 431)

The failure reason if the job fails.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 1024.

Pattern: `.*`

fleet (p. 431)

The Amazon Resource Name (ARN) of the fleet.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1224.

Pattern: `arn:.*`

status (p. 431)

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. 532).

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 V2
- 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 uses the following URI parameters.

resourceArn (p. 436)

The Amazon Resource Name (ARN) of the AWS RoboMaker resource you are tagging.

Length Constraints: Minimum length of 1. Maximum length of 1224.

Pattern: `arn:.*`

Required: Yes

Request Body

The request accepts the following data in JSON format.

tags (p. 436)

A map that contains tag keys and tag values that are attached to the resource.

Type: String to string map

Map Entries: Minimum number of 0 items. Maximum number of 50 items.

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: 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. 532).

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 V2
- 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 uses the following URI parameters.

resourceArn (p. 438)

The Amazon Resource Name (ARN) of the AWS RoboMaker resource you are removing tags.

Length Constraints: Minimum length of 1. Maximum length of 1224.

Pattern: `arn:.*`

Required: Yes

tagKeys (p. 438)

A map that contains tag keys and tag values that will be unattached from the resource.

Length Constraints: Minimum length of 1. Maximum length of 128.

Pattern: `[a-zA-Z0-9 _.\-\/+=:]*`

Required: Yes

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 (p. 532).

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 V2
- 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",
   "environment": {
      "uri": "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. 440)

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. 440)

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

environment (p. 440)

The object that contains the Docker image URI for your robot application.

Type: Environment (p. 466) object

Required: No

robotSoftwareSuite (p. 440)

The robot software suite (ROS distribution) used by the robot application.

Type: RobotSoftwareSuite (p. 492) object

Required: Yes

sources (p. 440)

The sources of the robot application.

Type: Array of SourceConfig (p. 512) objects

Required: No

Response Syntax

```
HTTP/1.1 200
Content-type: application/json

{
   "arn": "string",
   "environment": {
      "uri": "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. 441)

The Amazon Resource Name (ARN) of the updated robot application.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1224.

Pattern: `arn:.*`

environment (p. 441)

The object that contains the Docker image URI for your robot application.

Type: Environment (p. 466) object

lastUpdatedAt (p. 441)

The time, in milliseconds since the epoch, when the robot application was last updated.

Type: Timestamp

name (p. 441)

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. 441)

The revision id of the robot application.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 40.

Pattern: `[a-zA-Z0-9_.\-]*`

robotSoftwareSuite (p. 441)

The robot software suite (ROS distribution) used by the robot application.

Type: RobotSoftwareSuite (p. 492) object

sources (p. 441)

The sources of the robot application.

Type: Array of Source (p. 511) objects

version (p. 441)

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. 532).

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 V2
- 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",
   "environment": {
      "uri": "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. 444)

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. 444)

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

environment (p. 444)

The object that contains the Docker image URI for your simulation application.

Type: Environment (p. 466) object

Required: No

renderingEngine (p. 444)

The rendering engine for the simulation application.

Type: RenderingEngine (p. 483) object

Required: No

robotSoftwareSuite (p. 444)

Information about the robot software suite (ROS distribution).

Type: RobotSoftwareSuite (p. 492) object

Required: Yes

simulationSoftwareSuite (p. 444)

The simulation software suite used by the simulation application.

Type: SimulationSoftwareSuite (p. 510) object

Required: Yes

sources (p. 444)

The sources of the simulation application.

Type: Array of SourceConfig (p. 512) objects

Required: No

Response Syntax

```
HTTP/1.1 200
Content-type: application/json

{
   "arn": "string",
   "environment": {
      "uri": "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. 445)

The Amazon Resource Name (ARN) of the updated simulation application.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1224.

Pattern: `arn:.*`

environment (p. 445)

The object that contains the Docker image URI used for your simulation application.

Type: Environment (p. 466) object

lastUpdatedAt (p. 445)

The time, in milliseconds since the epoch, when the simulation application was last updated.

Type: Timestamp

name (p. 445)

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. 445)

The rendering engine for the simulation application.

Type: RenderingEngine (p. 483) object

revisionId (p. 445)

The revision id of the simulation application.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 40.

Pattern: `[a-zA-Z0-9_.\-]*`

robotSoftwareSuite (p. 445)

Information about the robot software suite (ROS distribution).

Type: RobotSoftwareSuite (p. 492) object

simulationSoftwareSuite (p. 445)

The simulation software suite used by the simulation application.

Type: SimulationSoftwareSuite (p. 510) object

sources (p. 445)

The sources of the simulation application.

Type: Array of Source (p. 511) objects

version (p. 445)

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. 532).

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 V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

UpdateWorldTemplate

Updates a world template.

Request Syntax

```
POST /updateWorldTemplate HTTP/1.1
Content-type: application/json

{
   "name": "string",
   "template": "string",
   "templateBody": "string",
   "templateLocation": {
      "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.

name (p. 449)

The name of the template.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 255.

Pattern: `.*`

Required: No

template (p. 449)

The Amazon Resource Name (arn) of the world template to update.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1224.

Pattern: `arn:.*`

Required: Yes

templateBody (p. 449)

The world template body.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 262144.

Pattern: `[\S\s]+`

Required: No

templateLocation (p. 449)

The location of the world template.

Type: TemplateLocation (p. 513) object

Required: No

Response Syntax

```
HTTP/1.1 200
Content-type: application/json

{
   "arn": "string",
   "createdAt": number,
   "lastUpdatedAt": number,
   "name": "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. 450)

The Amazon Resource Name (arn) of the world template.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1224.

Pattern: `arn:.*`

createdAt (p. 450)

The time, in milliseconds since the epoch, when the world template was created.

Type: Timestamp

lastUpdatedAt (p. 450)

The time, in milliseconds since the epoch, when the world template was last updated.

Type: Timestamp

name (p. 450)

The name of the world template.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 255.

Pattern: `.*`

Errors

For information about the errors that are common to all actions, see Common Errors (p. 532).

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 V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

Data Types

The following data types are supported:

BatchPolicy

Information about the batch policy.

Contents

maxConcurrency

The number of active simulation jobs create 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 V2
- AWS SDK for Ruby V3

Compute

Compute information for the simulation job.

Contents

computeType

Compute type information for the simulation job.

Type: String

Valid Values: `CPU | GPU_AND_CPU`

Required: No

gpuUnitLimit

Compute GPU unit limit for the simulation job. It is the same as the number of GPUs allocated to the SimulationJob.

Type: Integer

Valid Range: Minimum value of 0. Maximum value of 1.

Required: No

simulationUnitLimit

The simulation unit limit. Your simulation is allocated CPU and memory proportional to the supplied simulation unit limit. A simulation unit is 1 vcpu and 2GB of memory. You are only billed for the SU utilization you consume up to the maximum value provided. The default is 15.

Type: Integer

Valid Range: Minimum value of 1. Maximum value of 15.

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 V2
- AWS SDK for Ruby V3

ComputeResponse

Compute information for the simulation job

Contents

computeType

Compute type response information for the simulation job.

Type: String

Valid Values: `CPU | GPU_AND_CPU`

Required: No

gpuUnitLimit

Compute GPU unit limit for the simulation job. It is the same as the number of GPUs allocated to the SimulationJob.

Type: Integer

Valid Range: Minimum value of 0. Maximum value of 1.

Required: No

simulationUnitLimit

The simulation unit limit. Your simulation is allocated CPU and memory proportional to the supplied simulation unit limit. A simulation unit is 1 vcpu and 2GB of memory. You are only billed for the SU utilization you consume up to the maximum value provided. The default is 15.

Type: Integer

Valid Range: Minimum value of 1. Maximum value of 15.

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 V2
- AWS SDK for Ruby V3

DataSource

Information about a data source.

Contents

destination

The location where your files are mounted in the container image.

If you've specified the `type` of the data source as an `Archive`, you must provide an Amazon S3 object key to your archive. The object key must point to either a `.zip` or `.tar.gz` file.

If you've specified the `type` of the data source as a `Prefix`, you provide the Amazon S3 prefix that points to the files that you are using for your data source.

If you've specified the `type` of the data source as a `File`, you provide the Amazon S3 path to the file that you're using as your data source.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1024.

Pattern: `.*`

Required: No

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

Length Constraints: Minimum length of 3. Maximum length of 63.

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. 493) objects

Required: No

type

The data type for the data source that you're using for your container image or simulation job. You can use this field to specify whether your data source is an Archive, an Amazon S3 prefix, or a file.

If you don't specify a field, the default value is `File`.

Type: String

Valid Values: `Prefix | Archive | File`

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 V2
- AWS SDK for Ruby V3

DataSourceConfig

Information about a data source.

Contents

destination

The location where your files are mounted in the container image.

If you've specified the `type` of the data source as an `Archive`, you must provide an Amazon S3 object key to your archive. The object key must point to either a `.zip` or `.tar.gz` file.

If you've specified the `type` of the data source as a `Prefix`, you provide the Amazon S3 prefix that points to the files that you are using for your data source.

If you've specified the `type` of the data source as a `File`, you provide the Amazon S3 path to the file that you're using as your data source.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1024.

Pattern: `.*`

Required: No

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

Length Constraints: Minimum length of 3. Maximum length of 63.

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.

Length Constraints: Minimum length of 0. Maximum length of 1024.

Pattern: `.*`

Required: Yes

type

The data type for the data source that you're using for your container image or simulation job. You can use this field to specify whether your data source is an Archive, an Amazon S3 prefix, or a file.

If you don't specify a field, the default value is `File`.

Type: String

Valid Values: `Prefix | Archive | File`

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 V2
- 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

Length Constraints: Minimum length of 1. Maximum length of 1224.

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. 464) 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 V2
- 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. 494) 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 V2
- AWS SDK for Ruby V3

DeploymentJob

Information about a deployment job.

Contents

arn

The Amazon Resource Name (ARN) of the deployment job.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1224.

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. 460) objects

Array Members: Fixed number of 1 item.

Required: No

deploymentConfig

The deployment configuration.

Type: DeploymentConfig (p. 461) object

Required: No

failureCode

The deployment job failure code.

Type: String

Valid Values: `ResourceNotFound | EnvironmentSetupError | EtagMismatch | FailureThresholdBreached | RobotDeploymentAborted | RobotDeploymentNoResponse | RobotAgentConnectionTimeout | GreengrassDeploymentFailed | InvalidGreengrassGroup | MissingRobotArchitecture | MissingRobotApplicationArchitecture | MissingRobotDeploymentResource | GreengrassGroupVersionDoesNotExist | LambdaDeleted | ExtractingBundleFailure | PreLaunchFileFailure | PostLaunchFileFailure | BadPermissionError | DownloadConditionFailed | BadLambdaAssociated | InternalServerError | RobotApplicationDoesNotExist | DeploymentFleetDoesNotExist | FleetDeploymentTimeout`

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

Length Constraints: Minimum length of 1. Maximum length of 1224.

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 V2
- 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

Map Entries: Minimum number of 0 items. Maximum number of 16 items.

Key Length Constraints: Minimum length of 1. Maximum length of 1024.

Key Pattern: `[A-Z_][A-Z0-9_]*`

Value Length Constraints: Minimum length of 1. Maximum length of 1024.

Value Pattern: `.*`

Required: No

launchFile

The launch file name.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1024.

Pattern: `[a-zA-Z0-9_.\-]*`

Required: Yes

packageName

The package name.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1024.

Pattern: `[a-zA-Z0-9_.\-]*`

Required: Yes

postLaunchFile

The deployment post-launch file. This file will be executed after the launch file.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1024.

Pattern: `.*`

Required: No

preLaunchFile

The deployment pre-launch file. This file will be executed prior to the launch file.

Type: String

Length Constraints: Minimum length of 1. 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 V2
- AWS SDK for Ruby V3

Environment

The object that contains the Docker image URI for either your robot or simulation applications.

Contents

uri

The Docker image URI for either your robot or simulation applications.

Type: String

Length Constraints: Minimum length of 1. 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 V2
- 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 | RobotApplicationHealthCheckFailure | SimulationApplicationHealthCheckFailure | BadPermissionsRobotApplication | BadPermissionsSimulationApplication | BadPermissionsS3Object | BadPermissionsS3Output | BadPermissionsCloudwatchLogs | SubnetIpLimitExceeded | ENILimitExceeded | BadPermissionsUserCredentials | InvalidBundleRobotApplication | InvalidBundleSimulationApplication | InvalidS3Resource | ThrottlingError | LimitExceeded | MismatchedEtag | RobotApplicationVersionMismatchedEtag | SimulationApplicationVersionMismatchedEtag | ResourceNotFound | RequestThrottled | BatchTimedOut | BatchCanceled | InvalidInput | WrongRegionS3Bucket | WrongRegionS3Output | WrongRegionRobotApplication | WrongRegionSimulationApplication | UploadContentMismatchError`

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. 505) 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 V2
- AWS SDK for Ruby V3

FailureSummary

Information about worlds that failed.

Contents

failures

The worlds that failed.

Type: Array of WorldFailure (p. 527) objects

Array Members: Minimum number of 0 items. Maximum number of 100 items.

Required: No

totalFailureCount

The total number of failures.

Type: Integer

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 V2
- 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 V2
- AWS SDK for Ruby V3

FinishedWorldsSummary

Information about worlds that finished.

Contents

failureSummary

Information about worlds that failed.

Type: FailureSummary (p. 469) object

Required: No

finishedCount

The total number of finished worlds.

Type: Integer

Required: No

succeededWorlds

A list of worlds that succeeded.

Type: Array of strings

Array Members: Minimum number of 1 item. Maximum number of 100 items.

Length Constraints: Minimum length of 1. Maximum length of 1224.

Pattern: `arn:.*`

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 V2
- AWS SDK for Ruby V3

Fleet

Information about a fleet.

Contents

arn

The Amazon Resource Name (ARN) of the fleet.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1224.

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

Length Constraints: Minimum length of 1. Maximum length of 1224.

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 V2
- AWS SDK for Ruby V3

LaunchConfig

Information about a launch configuration.

Contents

command

If you've specified `General` as the value for your `RobotSoftwareSuite`, you can use this field to specify a list of commands for your container image.

If you've specified `SimulationRuntime` as the value for your `SimulationSoftwareSuite`, you can use this field to specify a list of commands for your container image.

Type: Array of strings

Length Constraints: Minimum length of 1. Maximum length of 255.

Pattern: `.+`

Required: No

environmentVariables

The environment variables for the application launch.

Type: String to string map

Map Entries: Minimum number of 0 items. Maximum number of 16 items.

Key Length Constraints: Minimum length of 1. Maximum length of 1024.

Key Pattern: `[A-Z_][A-Z0-9_]*`

Value Length Constraints: Minimum length of 1. Maximum length of 1024.

Value Pattern: `.*`

Required: No

launchFile

The launch file name.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1024.

Pattern: `[a-zA-Z0-9_.\-]*`

Required: No

packageName

The package name.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1024.

Pattern: `[a-zA-Z0-9_.\-]*`

Required: No

portForwardingConfig

The port forwarding configuration.

Type: PortForwardingConfig (p. 479) 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 V2
- 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 V2
- 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 V2
- AWS SDK for Ruby V3

OutputLocation

The output location.

Contents

s3Bucket

The S3 bucket for output.

Type: String

Length Constraints: Minimum length of 3. Maximum length of 63.

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

Length Constraints: Minimum length of 1. 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 V2
- AWS SDK for Ruby V3

PortForwardingConfig

Configuration information for port forwarding.

Contents

portMappings

The port mappings for the configuration.

Type: Array of PortMapping (p. 480) 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 V2
- AWS SDK for Ruby V3

PortMapping

An object representing a port mapping.

Contents

applicationPort

The port number on the application.

Type: Integer

Valid Range: Minimum value of 1024. Maximum value of 65535.

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

Valid Range: Minimum value of 1. Maximum value of 65535.

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 V2
- 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

Precentage 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 V2
- 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

Length Constraints: Minimum length of 1. Maximum length of 4.

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 V2
- 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

Length Constraints: Minimum length of 1. Maximum length of 1224.

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

Length Constraints: Minimum length of 1. Maximum length of 1224.

Pattern: `arn:.*`

Required: No

greenGrassGroupId

The Greengrass group associated with the robot.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1224.

Pattern: `.*`

Required: No

lastDeploymentJob

The Amazon Resource Name (ARN) of the last deployment job.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1224.

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 V2
- AWS SDK for Ruby V3

RobotApplicationConfig

Application configuration information for a robot.

Contents

application

The application information for the robot application.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1224.

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. 474) object

Required: Yes

tools

Information about tools configured for the robot application.

Type: Array of Tool (p. 516) objects

Array Members: Minimum number of 0 items. Maximum number of 10 items.

Required: No

uploadConfigurations

The upload configurations for the robot application.

Type: Array of UploadConfiguration (p. 518) objects

Array Members: Minimum number of 0 items. Maximum number of 10 items.

Required: No

useDefaultTools

A Boolean indicating whether to use default robot application tools. The default tools are rviz, rqt, terminal and rosbag record. The default is `False`.

Type: Boolean

Required: No

useDefaultUploadConfigurations

A Boolean indicating whether to use default upload configurations. By default, `.ros` and `.gazebo` files are uploaded when the application terminates and all ROS topics will be recorded.

If you set this value, you must specify an `outputLocation`.

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 V2
- AWS SDK for Ruby V3

RobotApplicationSummary

Summary information for a robot application.

Contents

arn

The Amazon Resource Name (ARN) of the robot.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1224.

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: `[a-zA-Z0-9_\-]*`

Required: No

robotSoftwareSuite

Information about a robot software suite (ROS distribution).

Type: RobotSoftwareSuite (p. 492) 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 V2
- AWS SDK for Ruby V3

RobotDeployment

Information about a robot deployment.

Contents

arn

The robot deployment Amazon Resource Name (ARN).

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1224.

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

Valid Values: `ResourceNotFound | EnvironmentSetupError | EtagMismatch | FailureThresholdBreached | RobotDeploymentAborted | RobotDeploymentNoResponse | RobotAgentConnectionTimeout | GreengrassDeploymentFailed | InvalidGreengrassGroup | MissingRobotArchitecture | MissingRobotApplicationArchitecture | MissingRobotDeploymentResource | GreengrassGroupVersionDoesNotExist | LambdaDeleted | ExtractingBundleFailure | PreLaunchFileFailure | PostLaunchFileFailure | BadPermissionError | DownloadConditionFailed | BadLambdaAssociated | InternalServerError | RobotApplicationDoesNotExist | DeploymentFleetDoesNotExist | FleetDeploymentTimeout`

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. 481) 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 V2
- 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 | General`

Required: No

version

The version of the robot software suite (ROS distribution).

Type: String

Valid Values: `Kinetic | Melodic | Dashing | Foxy`

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 V2
- 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

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 V2
- AWS SDK for Ruby V3

S3Object

Information about an S3 object.

Contents

bucket

The bucket containing the object.

Type: String

Length Constraints: Minimum length of 3. Maximum length of 63.

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

Length Constraints: Minimum length of 1. Maximum length of 1024.

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 V2
- AWS SDK for Ruby V3

SimulationApplicationConfig

Information about a simulation application configuration.

Contents

application

The application information for the simulation application.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1224.

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. 474) object

Required: Yes

tools

Information about tools configured for the simulation application.

Type: Array of Tool (p. 516) objects

Array Members: Minimum number of 0 items. Maximum number of 10 items.

Required: No

uploadConfigurations

Information about upload configurations for the simulation application.

Type: Array of UploadConfiguration (p. 518) objects

Array Members: Minimum number of 0 items. Maximum number of 10 items.

Required: No

useDefaultTools

A Boolean indicating whether to use default simulation application tools. The default tools are rviz, rqt, terminal and rosbag record. The default is `False`.

Type: Boolean

Required: No

useDefaultUploadConfigurations

A Boolean indicating whether to use default upload configurations. By default, `.ros` and `.gazebo` files are uploaded when the application terminates and all ROS topics will be recorded.

If you set this value, you must specify an `outputLocation`.

Type: Boolean

Required: No

worldConfigs

A list of world configurations.

Type: Array of WorldConfig (p. 523) objects

Array Members: Minimum number of 0 items. Maximum number of 1 item.

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 V2
- 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

Length Constraints: Minimum length of 1. Maximum length of 1224.

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. 492) object

Required: No

simulationSoftwareSuite

Information about a simulation software suite.

Type: SimulationSoftwareSuite (p. 510) 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 V2
- AWS SDK for Ruby V3

SimulationJob

Information about a simulation job.

Contents

arn

The Amazon Resource Name (ARN) of the simulation job.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1224.

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

compute

Compute information for the simulation job

Type: ComputeResponse (p. 455) object

Required: No

dataSources

The data sources for the simulation job.

Type: Array of DataSource (p. 456) objects

Required: No

failureBehavior

The failure behavior the simulation job.

Continue

Leaves the host running for its maximum timeout duration after a `4XX` error code.

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 | RobotApplicationHealthCheckFailure | SimulationApplicationHealthCheckFailure | BadPermissionsRobotApplication | BadPermissionsSimulationApplication | BadPermissionsS3Object | BadPermissionsS3Output | BadPermissionsCloudwatchLogs | SubnetIpLimitExceeded | ENILimitExceeded | BadPermissionsUserCredentials | InvalidBundleRobotApplication | InvalidBundleSimulationApplication | InvalidS3Resource | ThrottlingError | LimitExceeded | MismatchedEtag | RobotApplicationVersionMismatchedEtag | SimulationApplicationVersionMismatchedEtag | ResourceNotFound | RequestThrottled | BatchTimedOut | BatchCanceled | InvalidInput | WrongRegionS3Bucket | WrongRegionS3Output | WrongRegionRobotApplication | WrongRegionSimulationApplication | UploadContentMismatchError`

Required: No

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

Length Constraints: Minimum length of 1. Maximum length of 255.

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. 476) 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: 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. 477) object

Required: No

outputLocation

Location for output files generated by the simulation job.

Type: OutputLocation (p. 478) object

Required: No

robotApplications

A list of robot applications.

Type: Array of RobotApplicationConfig (p. 486) objects

Array Members: Fixed number of 1 item.

Required: No

simulationApplications

A list of simulation applications.

Type: Array of SimulationApplicationConfig (p. 495) 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

Map Entries: Minimum number of 0 items. Maximum number of 50 items.

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. 521) 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 V2
- AWS SDK for Ruby V3

SimulationJobBatchSummary

Information about a simulation job batch.

Contents

arn

The Amazon Resource Name (ARN) of the batch.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1224.

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.

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 `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`

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 V2
- AWS SDK for Ruby V3

SimulationJobRequest

Information about a simulation job request.

Contents

compute

Compute information for the simulation job

Type: Compute (p. 454) object

Required: No

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. 458) objects

Array Members: Minimum number of 1 item. Maximum number of 5 items.

Required: No

failureBehavior

The failure behavior the simulation job.

Continue

Leaves the host running for its maximum timeout duration after a `4XX` error code.

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. 476) 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. 478) object

Required: No

robotApplications

The robot applications to use in the simulation job.

Type: Array of RobotApplicationConfig (p. 486) 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. 495) 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

Map Entries: Minimum number of 0 items. Maximum number of 50 items.

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

A Boolean indicating whether to use default applications in the simulation job. Default applications include Gazebo, rqt, rviz and terminal access.

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. 520) 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 V2
- 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

Length Constraints: Minimum length of 1. Maximum length of 1224.

Pattern: `arn:.*`

Required: No

computeType

The compute type for the simulation job summary.

Type: String

Valid Values: `CPU | GPU_AND_CPU`

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 V2
- 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 | SimulationRuntime`

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|11|Kinetic|Melodic|Dashing|Foxy`

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 V2
- 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

Length Constraints: Minimum length of 3. Maximum length of 63.

Pattern: `[a-z0-9][a-z0-9.\-]*[a-z0-9]`

Required: No

s3Key

The s3 object key.

Type: String

Length Constraints: Minimum length of 1. 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 V2
- 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

Length Constraints: Minimum length of 3. Maximum length of 63.

Pattern: `[a-z0-9][a-z0-9.\-]*[a-z0-9]`

Required: No

s3Key

The s3 object key.

Type: String

Length Constraints: Minimum length of 1. 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 V2
- AWS SDK for Ruby V3

TemplateLocation

Information about a template location.

Contents

s3Bucket

The Amazon S3 bucket name.

Type: String

Length Constraints: Minimum length of 3. Maximum length of 63.

Pattern: `[a-z0-9][a-z0-9.\-]*[a-z0-9]`

Required: Yes

s3Key

The list of S3 keys identifying the data source files.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1024.

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 V2
- AWS SDK for Ruby V3

TemplateSummary

Summary information for a template.

Contents

arn

The Amazon Resource Name (ARN) of the template.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1224.

Pattern: `arn:.*`

Required: No

createdAt

The time, in milliseconds since the epoch, when the template was created.

Type: Timestamp

Required: No

lastUpdatedAt

The time, in milliseconds since the epoch, when the template was last updated.

Type: Timestamp

Required: No

name

The name of the template.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 255.

Pattern: `.*`

Required: No

version

The version of the template that you're using.

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 V2
- AWS SDK for Ruby V3

Tool

Information about a tool. Tools are used in a simulation job.

Contents

command

Command-line arguments for the tool. It must include the tool executable name.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1024.

Pattern: `.*`

Required: Yes

exitBehavior

Exit behavior determines what happens when your tool quits running. `RESTART` will cause your tool to be restarted. `FAIL` will cause your job to exit. The default is `RESTART`.

Type: String

Valid Values: `FAIL | RESTART`

Required: No

name

The name of the tool.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 255.

Pattern: `[a-zA-Z0-9_\-]*`

Required: Yes

streamOutputToCloudWatch

Boolean indicating whether logs will be recorded in CloudWatch for the tool. The default is `False`.

Type: Boolean

Required: No

streamUI

Boolean indicating whether a streaming session will be configured for the tool. If `True`, AWS RoboMaker will configure a connection so you can interact with the tool as it is running in the simulation. It must have a graphical user interface. The default is `False`.

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 V2
- AWS SDK for Ruby V3

UploadConfiguration

Provides upload configuration information. Files are uploaded from the simulation job to a location you specify.

Contents

name

A prefix that specifies where files will be uploaded in Amazon S3. It is appended to the simulation output location to determine the final path.

For example, if your simulation output location is `s3://my-bucket` and your upload configuration name is `robot-test`, your files will be uploaded to `s3://my-bucket/<simid>/<runid>/robot-test`.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 255.

Pattern: `[a-zA-Z0-9_\-]*`

Required: Yes

path

Specifies the path of the file(s) to upload. Standard Unix glob matching rules are accepted, with the addition of `**` as a *super asterisk*. For example, specifying `/var/log/**.log` causes all .log files in the `/var/log` directory tree to be collected. For more examples, see Glob Library.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1024.

Pattern: `.*`

Required: Yes

uploadBehavior

Specifies when to upload the files:

UPLOAD_ON_TERMINATE

Matching files are uploaded once the simulation enters the `TERMINATING` state. Matching files are not uploaded until all of your code (including tools) have stopped.

If there is a problem uploading a file, the upload is retried. If problems persist, no further upload attempts will be made.

UPLOAD_ROLLING_AUTO_REMOVE

Matching files are uploaded as they are created. They are deleted after they are uploaded. The specified path is checked every 5 seconds. A final check is made when all of your code (including tools) have stopped.

Type: String

Valid Values: `UPLOAD_ON_TERMINATE | UPLOAD_ROLLING_AUTO_REMOVE`

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 V2
- AWS SDK for Ruby V3

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.

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 V2
- 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 V2
- AWS SDK for Ruby V3

WorldConfig

Configuration information for a world.

Contents

world

The world generated by Simulation WorldForge.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1224.

Pattern: `arn:.*`

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 V2
- AWS SDK for Ruby V3

WorldCount

The number of worlds that will be created. You can configure the number of unique floorplans and the number of unique interiors for each floor plan. For example, if you want 1 world with 20 unique interiors, you set `floorplanCount = 1` and `interiorCountPerFloorplan = 20`. This will result in 20 worlds (`floorplanCount` * `interiorCountPerFloorplan`).

If you set `floorplanCount = 4` and `interiorCountPerFloorplan = 5`, there will be 20 worlds with 5 unique floor plans.

Contents

floorplanCount

The number of unique floorplans.

Type: Integer

Required: No

interiorCountPerFloorplan

The number of unique interiors per floorplan.

Type: Integer

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 V2
- AWS SDK for Ruby V3

WorldExportJobSummary

Information about a world export job.

Contents

arn

The Amazon Resource Name (ARN) of the world export job.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1224.

Pattern: `arn:.*`

Required: No

createdAt

The time, in milliseconds since the epoch, when the world export job was created.

Type: Timestamp

Required: No

status

The status of the world export job.

Pending

The world export job request is pending.

Running

The world export job is running.

Completed

The world export job completed.

Failed

The world export job failed. See `failureCode` for more information.

Canceled

The world export job was cancelled.

Canceling

The world export job is being cancelled.

Type: String

Valid Values: `Pending | Running | Completed | Failed | Canceling | Canceled`

Required: No

worlds

A list of worlds.

Type: Array of strings

Array Members: Minimum number of 1 item. Maximum number of 100 items.

Length Constraints: Minimum length of 1. Maximum length of 1224.

Pattern: `arn:.*`

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 V2
- AWS SDK for Ruby V3

WorldFailure

Information about a failed world.

Contents

failureCode

The failure code of the world export job if it failed:

InternalServiceError

Internal service error.

LimitExceeded

The requested resource exceeds the maximum number allowed, or the number of concurrent stream requests exceeds the maximum number allowed.

ResourceNotFound

The specified resource could not be found.

RequestThrottled

The request was throttled.

InvalidInput

An input parameter in the request is not valid.

Type: String

Valid Values: `InternalServiceError | LimitExceeded | ResourceNotFound | RequestThrottled | InvalidInput | AllWorldGenerationFailed`

Required: No

failureCount

The number of failed worlds.

Type: Integer

Required: No

sampleFailureReason

The sample reason why the world failed. World errors are aggregated. A sample is used as the `sampleFailureReason`.

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 V2
- AWS SDK for Ruby V3

WorldGenerationJobSummary

Information about a world generator job.

Contents

arn

The Amazon Resource Name (ARN) of the world generator job.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1224.

Pattern: `arn:.*`

Required: No

createdAt

The time, in milliseconds since the epoch, when the world generator job was created.

Type: Timestamp

Required: No

failedWorldCount

The number of worlds that failed.

Type: Integer

Required: No

status

The status of the world generator job:

Pending

The world generator job request is pending.

Running

The world generator job is running.

Completed

The world generator job completed.

Failed

The world generator job failed. See `failureCode` for more information.

PartialFailed

Some worlds did not generate.

Canceled

The world generator job was cancelled.

Canceling

The world generator job is being cancelled.

Type: String

Valid Values: `Pending | Running | Completed | Failed | PartialFailed | Canceling | Canceled`

Required: No

succeededWorldCount

The number of worlds that were generated.

Type: Integer

Required: No

template

The Amazon Resource Name (arn) of the world template.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1224.

Pattern: `arn:.*`

Required: No

worldCount

Information about the world count.

Type: WorldCount (p. 524) 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 V2
- AWS SDK for Ruby V3

WorldSummary

Information about a world.

Contents

arn

The Amazon Resource Name (ARN) of the world.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1224.

Pattern: `arn:.*`

Required: No

createdAt

The time, in milliseconds since the epoch, when the world was created.

Type: Timestamp

Required: No

generationJob

The Amazon Resource Name (arn) of the world generation job.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1224.

Pattern: `arn:.*`

Required: No

template

The Amazon Resource Name (arn) of the world template.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1224.

Pattern: `arn:.*`

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 V2
- AWS SDK for Ruby V3

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

NotAuthorized

You do not have permission to perform this action.

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.

Change	**Description**	**Date**
New service and guide	This is the initial release of AWS RoboMaker and the *AWS RoboMaker Developer Guide*.	11/07/2018
Support for tags	Added support for tags to many AWS RoboMaker resources.	1/24/2019
Samples deprecated	Deprecated self-driving reinforcement, navigation, person detection, and voice command samples.	5/15/2020

AWS glossary

For the latest AWS terminology, see the AWS glossary in the *AWS General Reference*.