AWS Overview & Robomaker Introduction

6th Annual Entrepreneur's Forum and Technology Showcase
A Rising Tide - Organizing for Success
26 Aug 2020

David Bader – davbader@amazon.com
Business Development Manager, AWS – Robotics, Public Sector/DoD

Anabell St Vincent – anabell@amazon.com
Principal Solutions Architect, AWS Robotics & Autonomous Services

Camilo Buscaron – buscaron@amazon.com
Principal Open Source Tech, AWS Robotics & Autonomous Services
Most Functionality

Use the right tool for the job!

Fastest Pace of Innovation

Leverage the latest technologies to experiment and innovate more quickly.

Largest Community of Customers & Partners

The AWS Partner Network includes thousands of systems integrators who specialize in AWS services and tens of thousands of independent software vendors (ISVs) who adapt their technology to work on AWS.

Most Mature Cloud Platform

“There is no compression algorithm for experience”
Andy Jassy – CEO AWS

Most Secure Cloud Platform

AWS is architected to be the most flexible and secure cloud computing environment available today.
AWS has enabled on-demand delivery of computing power, database storage and other IT resources via the internet with pay-as-you-go pricing.
Enabling Lean Startups with AWS Cloud

Zero upfront cost
With AWS's infrastructure-on-demand, startups can pay only for the resources they use instead of investing in servers upfront.

Launch faster
Startups can have new IT resources available in just a few clicks, increasing agility.

Focus on core business value
Startups can focus on growing their business rather than on infrastructure.

Experiment often at lower risk
Being able to deprovision servers as needed enables startups to experiment often and fail fast if an idea doesn’t work.
Massive technology shifts such as cloud computing made it significantly cheaper to launch a startup:

AWS Startup BD: Working with venture capital and the startup ecosystem
We invest *indirectly* alongside venture funds and accelerators

**We don’t**
- Invest cash
- Take a capital position

**We do**
- Invest virtual currency (AWS credits)
- Invest time
- Share knowledge/experience/wisdom
- Help navigate AWS resources and support
- Open doors internally and externally
- Remove obstacles
- Leverage our global footprint
- Champion startups across all of Amazon
- Take a long-term view
We focus on helping our startup customers grow by wiring them into people, resources, opportunities across Amazon.

**Technical**
- Architecture design/optimization reviews
- Best practices
- Subject matter experts
- Betas/previews
- Security/compliance

**Go-to-market**
- Co-marketing
- PoC funding
- Sales referrals
- Distribution
- Capital intros
AWS RoboMaker Introduction
Robot landscape

International Space Station

Self Driving Vehicles

Tactical Ground Vehicles

UAV’s

UUV/USV

Robotic Arms

Mars Rover

Robotic Combat Vehicle (RCV)

Education
Robotics use is accelerating in key industries

Accelerated robot deployment in collaborative environments given enhanced capabilities such as autonomous mobility and artificial intelligence.

By 2023, it’s estimated that autonomous mobile robots will emerge as the standard for logistic and fulfillment processes.

By 2030, 70% of all mobile material handling equipment will be autonomous.

Source: IDTechEx

- Logistics
- Construction
- Retail
- Healthcare
- Consumer home
- Energy and utilities
- Oil and gas
- Public Sector
Challenges with robotics development and testing

- Multi-domain expertise required to build robots
- Iterative development to get it right
- Configuration management is hard
- Limited robot hardware available for testing
- Deployment and updates need to be managed
The role of simulations

**REPRODUCIBLE SCENARIOS**
Re-create edge cases and unsafe conditions to test for unexpected behaviors

**EXPANDED TEST COVERAGE**
Programmatically run many scenarios that cover the vast majority of situations your robot would encounter

**QUICKER RESULTS**
Run tests faster than real-time for certain scenarios

*Using simulation will increase development velocity, reduce the number of bugs and improve code quality.*
Simulations at scale in the cloud

**HIGHLY SCALABLE**
Concurrent simulations at cloud-scale via a single API call

**HIGHLY RELIABLE**
AWS cloud with enterprise-grade availability

**FULLY MANAGED**
Managed ROS and Gazebo software stack frees up engineering resources

**COST EFFECTIVE**
Pay-as-you-go pricing lowers cost of product testing

**AWS SERVICE INTEGRATIONS**
Suite of AWS services for building end-to-end solutions

*AWS RoboMaker removes barriers to use simulation and enables automated testing at scale.*
AWS RoboMaker: supporting the development lifecycle

- Design and develop
- Test and verify
- Deploy and update

Cloud-based development
Cloud-based simulation
Cloud-based fleet deployment
AWS contributions to ROS2

- Quality of Service (QoS) features for topics
- Cross-compilation tools
- ROS2 launch sandboxing extension
- Secure-ROS2 (SROS2) improvements
- Runtime analysis tools address & thread sanitizers (Asan/Tsan)
- Created and maintain rcpputils core package

https://github.com/aws-robotics
Managed ROS* environment

Support for ROS Kinetic, ROS Melodic, ROS 2 Dashing (beta)

Managed Gazebo (Gazebo 7, Gazebo 9)

Managed ROS/ Gazebo tools – rviz, rqt, GZ client

Application and tool launch orchestration

Batch simulation with API calls

*ROS: Robot Operating System
Generate a simulation world
Build a robot in simulation

Getting started
- ROS urdf examples
- ROS.org urdf packages
- ROS Industrial Intro to urdf

CAD conversion to URDF*
- SolidWorks to URDF
- Solidworks to Gazebo
- Blender plugin

Tutorials
- ROS.org tutorials
- Building a robot model
- Visualize a robot urdf with rviz

Proposed program benefit: Implementation guidance

*URDF: Unified Resource Description Format
Concurrent simulations for regression testing with CI/CD integration

NEED
Increase code release velocity while improving test coverage for different floor layouts and scenarios.

CHALLENGES
Testing was costly and time consuming, therefore limited in coverage. Discovery of software defects happened late in the release cycle.

SOLUTION
Automated simulation-based regression testing in a CI/CD pipeline using AWS RoboMaker, a fully managed cloud-based simulation service.

RESULTS
More than 40 automated tests on each code commit and more than 500 automated tests for each release candidate.
Regression testing with CI/CD integration

NEED
Increase code release velocity while improving test coverage for different floor layouts and scenarios.

CHALLENGES
Testing was costly and time consuming, therefore limited in coverage. Discovery of software defects happened late in the release cycle.

SOLUTION
Automated simulation-based regression testing in a CI/CD pipeline using AWS RoboMaker, a fully managed cloud simulation service.

RESULTS
More than 40 automated tests on each code commit and more than 500 automated tests for each release candidate.
Design and develop:
ROS extensions, AWS integrations, Tools support

- Integration with AWS services enable easy ingestion of data from robots
- Native ROS packages for Amazon S3 for secure, scalable storage: Rosbag logging and upload to S3
- Integration with Amazon CloudWatch for easy logging and metrics collection
- Bundle your own GUI-enabled tools as part of robot and simulation applications

*ROS: Robot Operating System*
ROS Application CI/CD Workflow

Application Architecture

1. **Git Repository (GitHub or AWS Code Commit)**
   - Branch: feature
   - Pull Request
   - Branch: integration
   - **Clone on Git Hook (Merge)**

2. **AWS CodePipeline**
   - **AWS CodeBuild**
     - Build and Bundle
     - Code in ROS Container
     - Bundles stored in S3

3. **AWS Step Functions: Testing**
   - **AWS Lambda**
     - Simulation Launcher
   - **AWS RoboMaker Simulation Service**
   - **AWS Lambda Status Checker**
   - **Tests Passed**

4. **AWS Lambda**
   - AWS RoboMaker Deploy Launcher
   - AWS RoboMaker Fleet Management
   - Test Fleet
   - Prod Fleet

5. **Clone on Git Hook (Merge)**
   - Branch: master (release)

6. **AWS Lambda**
   - AWS RoboMaker Deploy Launcher
   - AWS RoboMaker Fleet Management
   - Test Fleet
   - Prod Fleet

7. **AWS Lambda**
   - Production Deploy
   - Manual Action

ROS/ROS2 Development Environment

ROS/ROS2 Simulation Environment
Learn more!

Read about **CI/CD, fleet simulations, batch simulations, and more** in the **AWS Robotics Blog**:

https://aws.amazon.com/blogs/robotics/

Get hands-on with **AWS RoboMaker workshops**:
https://robomakerworkshops.com/

More details on **AWS RoboMaker**:
https://aws.amazon.com/robomaker

*Try AWS RoboMaker today!*
Questions?
Thank you!
Additional Information
**RoboMaker Simulation JumpStart program**

<table>
<thead>
<tr>
<th>Process Steps</th>
<th>Simulation JumpStart Program components</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Decide on test scenarios and simulation fidelity</td>
<td>Guidance on value of simulations and suitable test scenarios based on fidelity of the simulation environment</td>
</tr>
<tr>
<td>2. Build a robot URDF model</td>
<td>Documentation and guidance on how to create a URDF</td>
</tr>
<tr>
<td>3. Build a simulation world</td>
<td>Light-weight warehouse world under an open source license</td>
</tr>
<tr>
<td>4. Define test case</td>
<td>Definition of common test scenarios and test cases by category</td>
</tr>
<tr>
<td></td>
<td>Sample test code for common test cases</td>
</tr>
<tr>
<td></td>
<td>Whitepaper on testing best practices</td>
</tr>
<tr>
<td>5. Build testing definition</td>
<td>Sample test definition script</td>
</tr>
<tr>
<td>6. Build testing orchestration</td>
<td>Guidance on orchestration the test runs</td>
</tr>
<tr>
<td>7. Run simulations</td>
<td>AWS RoboMaker cloud service</td>
</tr>
<tr>
<td>8. Analyze test results</td>
<td>Guidance and best practices</td>
</tr>
<tr>
<td>Deployment support and Credits</td>
<td>Potential service offerings for building simulation assets and/ or simulation testing code</td>
</tr>
<tr>
<td></td>
<td>Credits for AWS service usage</td>
</tr>
</tbody>
</table>

*Note: Program is currently under Beta, available for select customers only.*
Developers create feature branches for the user stories they are implementing. They work in a ROS development environment (IDE) of their choice.

When the developer completes their code, they submit a pull request to an integration branch. Once the code is reviewed, it is merged into the integration branch which invokes an AWS Code Pipeline job through a git hook.

The code is then copied into a AWS CodeBuild server, which is configured to use a custom Docker image with ROS build tools installed (based on ros:<version>). The build server runs colcon build and bundle commands, then uploads the bundles to an Amazon S3 bucket.

After the bundles are successfully in Amazon S3, an AWS Step Functions workflow is initiated to create AWS RoboMaker simulation jobs, monitor their progress and analyze the test simulation results. If the tests passed, the code can be safely merged into the master branch for deployment to physical robots.

The merge into the master branch invokes a new build process to prepare the robot application for deployment.

An AWS Step Functions workflow is used to manage the progress of the application code delivery. First, the production ROS application bundle is deployed to a fleet of test robots using AWS RoboMaker. Engineers will then review the physical test results, and if passed, advance the AWS Step Functions workflow with a manual action.

Conditions for production deployment (such as current battery life and location) as well as the number of concurrent deployments and failure thresholds are preconfigured in AWS RoboMaker. The tested, production ready code is now ready and is safely and securely deployed to the production fleet of robots.
Launch (n) number of AWS RoboMaker Simulation Jobs, with various environment variables.
An **AWS Lambda** function is used to parameterize and launch multiple simulations, each spawning a robot in an individual simulation job, but using the same Gazebo simulation world assets.

Once multiple simulation jobs are invoked, each individual robot is registered in **AWS IoT Device Management** and has an associated device shadow where robot state information (type and ID of the robot) is stored. As the robot moves, the telemetry data is published to an **AWS IoT Core topic over MQTT**.

The **AWS Lambda** function will spawn as many robots as defined, all publishing their state and location data to **AWS IoT**.

The MQTT message broker in **AWS IoT** is leveraged as a **many-to-one bridge** between the multiple simulations and a single, centralized and consolidated simulation.

The centralized ROS application running in **AWS RoboMaker Simulation** loads each device shadow and renders the robots in a consolidated view. This application subscribes to the telemetry topics and mirrors the robot shadow in the central application.
Robotics with Machine Learning

- ROS Application
- ROS Node with ML Libraries
- Fleet management
- Pre-launch Scripts
- AWS Greengrass Core Device
- AWS RoboMaker ROS Extensions
- Amazon CloudWatch
- Amazon Polly
- AWS IoT
- Status Updates AWS IoT
- AWS IoT Topic Rules
- Device Shadow
- More...

- Download New Models
- Sagemaker Trained ML Model
- Training
- Amazon S3 Dinosaur Images Training Data
- Tagged New Dinosaurs
- Amazon SageMaker Ground Truth
- AWS SageMaker
- AWS RoboMaker Simulation Environment
- Iterative, Test-driven Development
- AWS RoboMaker
- AWS Greengrass Core Device
- AWS Amplify React Website
- Upload and Tag Images of New Dinosaurs Found
- Amazon SageMaker
- Ground Truth
- Tagged New Dinosaurs
- More...
The Robot Operating System (ROS) application running on the NVIDIA® JetBot uses a local model, trained in AWS SageMaker, to identify dinosaurs in video frames as it explores a LEGO dinosaur world. The JetBot uses a small, powerful computer that runs in as little as 5 watts - the NVIDIA® Jetson Nano™ Developer Kit.

When a dinosaur is found, the application plays a short audio clip recorded using the AWS RoboMaker ROS extension for Amazon Polly, a text-to-speech service. A message with the dinosaur found is also sent to an AWS IoT MQTT topic.

An Amazon CloudFront and Amazon S3 client-side front-end react web application, built using the AWS Amplify CLI tools, then receives the AWS IoT message and updates the dashboard.

New robot features are developed using a local ROS/ROS2 development environment or the AWS RoboMaker cloud-based IDE. During the development the new ROS code can be easily simulated in AWS RoboMaker.

Once the feature it is ready, the code is integrated into git automation processes that use AWS RoboMaker simulation service to test various scenarios. Then, the code is reliably deployed over-the-air (OTA) to production robot fleets using AWS RoboMaker fleet management.

As new dinosaurs are added to the LEGO world, they are tagged and uploaded to Amazon S3 where a new model is trained in AWS SageMaker and then deployed to identify the new dinosaur.