category - meeting-notes tags - twitter - yc - robotics - startup date: 2024-08-08 createdAt: 2024-08-08 14:47

Vedant Nair @Miru Robotics meeting-notes

Agenda

Background: RR started as a drone company ~2016? and pivoted to autonomous robots for warehousing/3pl ~2018 Core DNA of the founders was robotics. ex ETH and mostly mech + controls people.

Was brought in to operationalize their robot deployment. they had already felt the need for a cloud solution given the need to capture operational data atleast. circa 2018

1. Consistent builds and deploys.
2. Staging environment for robotics at parity with production.
3. Centralize logs,rosbags,traces later.
4. Standardize communication. Onsite to cloud.
   Every client site had different networking requirements. Firewall setups etc. To appeal to enterprises which were primary buyers for our Warehouse robots. we needed to cut this to and fro time to < 1 week.

KPI

- Time to go live. Solutioning, vs Deployment.

We got this from 3 months down to 1 week.

What were the robotics DevOps tools that you built for Rapyuta Io?

rapyuta.io platform did three things.

- Infrastructure as Code for robotics. we found that you cant take either k8s manifests or docker-compose directly to a fleet of robots deployment. Vocabulary wasn’t a match. k8s/docker were built for the REST/HTTP world. where deployment nodes were ephemeral. and a docker container could go up and down without much care. till the time it came online, everything was good.
    - Robotics has state specially in terms of navigation state and localizaiton state. Deployments need to be careful in when they restart. They should give a pre-kill signal so that container code can save state and exit gracefully.
- Robot/Edge/Cloud networking.
    - We had a few communication components which made robot to robot ROS communication reliable in a local area network, or even across edge-internet. There were tradeoffs between models. These components called cloud bridges were pivotal in making communication transparent. This was tightly integrated into the rapyuta.io platform. Robotics developer never had to worry about how comms happened. You deploy a routed-ros-network. add a bunch of deployments, either on device or cloud, and they would communicate.
- Operations tools
    - RBAC, vpn connectivity for support, remote file uploads, log rotation, streaming logs, rosbag recording.

Why did people consider buying Rapuyta Io? What was the problem you were solving? How did you quantify your value?

We had a track record of operating our own fleet of 400+ robots back in 2020. We had done solutions for some big Jap companies. They saw our tools and wanted something similar for their other projects.

Why did you not sell this to other companies? What was their willingness to pay and outsource this core tech

We didn’t sell because as a company stuck in the pandemic, we had to decide our core product offering.
Our PMF in japan was in Logistics Automation. So we stuck to that. and stopped selling. This was also a business our investors undersood better.

Our sales org was also a hardware solutions and inside-sales enterprise selling team. To sell cloud, we would need a new sales structure

There was also a language problem. Our sales team was Japanese speaking whie our initial market survey for robotics showed interest in US/Europe.

There was also the case that should we focus our cloud engineering team to solve for problems of the internal customer [we were starting 2 more robotics product lines] vs solving for external would be customer.

What were your biggest challenges of building this out, we you able to abstract this to different robots/configs/compute

devices

> I see you used Kubes/Docker

Well this was the most rewarding bit. We started off when k8s didn’t have much of edge focus.

We were able to abstract pretty good.

Devices

- Robot or Edge
- Could host both storage or ROS networks
- vpn, logging, remote management, ssh built in.
- concept of a runtime. each device supported a dockerized runtime, or an ondevice-runtime [runs directly in a shell. useful for device drivers and maintenance.]

Packages. Superset of what a kubernetes Deployment Manifest or chart would be. but about 2 years before charts.

- Set of configuration and code often packaged or even obfuscated if needed.
- easy to distribute and check-in to GIT.
- having the manifest of a package was not a security risk. It still needed right Secrets to pull the images
- defined not just the runtime/commands, but also had provisions to describe, web endppints, ros networking topics, and loging,rosbag recording rules with log rotation.

Deployments

- Running instance of a package on a runtime. `cloud, device-docker, device-shell`
- Could also be templatized. `Run amr-robot package on robot{idx} for idx in range(1: 10)`

- Could take a setup of package defined runtime arguments and ensure things were valid before starting deployments.
- versioned. rollback was possible too.

Configuration

- json/yaml configurations.
- was possible to have merge trees.

Storage

- on device or cloud storage.

ROS Network

- Sort of routers for all robot communication.
- Offered
  - routed mode. where one edge server would mediate comms between N robots.
  - p2p mode. where NxM robots could communicate with each other. One edge server would maintain discovery information to support error free subscribe broadcast.
  - cloud-routed. Communication across edge-cloud. Used for live dashboards and remote operations.

Secrets

- git
- docker.

Managed Services

- run elastic search, postgres, s3 storage, redis using managed providers.

- This would provision say elastic search on Azure, or on Elastic Cloud, and inject the access credentials into a rapyuta deployment.
- That ways, we had a small team managing our cloud infra and its billing etc. And never had to share ACL/Access to the whole company.
- We could add a new managed service pretty easily.

All this was earlier managed using a ui/python scripts.

Biggest contribution was `rio-cli` which made all this seamless. https://cli.rapyuta.io/

Do you have any takes on building software for robotics? How has the market changed, anything changed about the problems you are solving?

1. I think we were early to robotics+cloud and it was the right product built in the wrong setting. born out of necessity so it did what an operational product needed to do.

2. Building for robotics users is a PITA. A lot of our internal robotics teams would love to just SSH and fix things in prod robots. Whichw as a big no from my teams and hence from our Standard Operating Procedure side. This is getting better now. But this kind of culture would be common for your customers. They would push you to give low level tools.

3. Attributing errors. Its always hard to offer SLAs on where the platform screwed up, vs where the end user screwed up.

4. Maintaining upgrade windows is hard. if you are to support 24x7 operations, your sales contract upfront will have to provision some safe times.

5. Cluster upgrades for security is a royal PITA. cluster migrations are a nightmare. We went from GCP to Azure after about 2 years or running on GCP. That was a 3month effort, just to ensure our live robots had a <5minute downtime for the oldest robots which was <5percent of the fleet. We had to do an openshift 3 to openshift 4 upgrade. another 2 months.
   We moved our k8s distribution from openshift to managed k8s - another 2 months. Migrating storage is the biggest tight-rope walk we had to do. Overall, just stick to single cloud provider, managed service.

6. You need to decide how you would make money. By selling a tool/per user subscription? by earning an arbitrage on cloud running cost?

Discussion : Open Ended

ROS and ROS2 is a split brain problem for the community. I am not sure if you guys are focussing on ros

Robot networking is different from plain networking although it builds on top of it. ROS netwroking is somewhat broken. ROS2

Have a remote operations angle to your product. Have a security and data lake/analytics offering as an addon to your product. Every robot deployment gets to this eventually. Integrate don’t build.