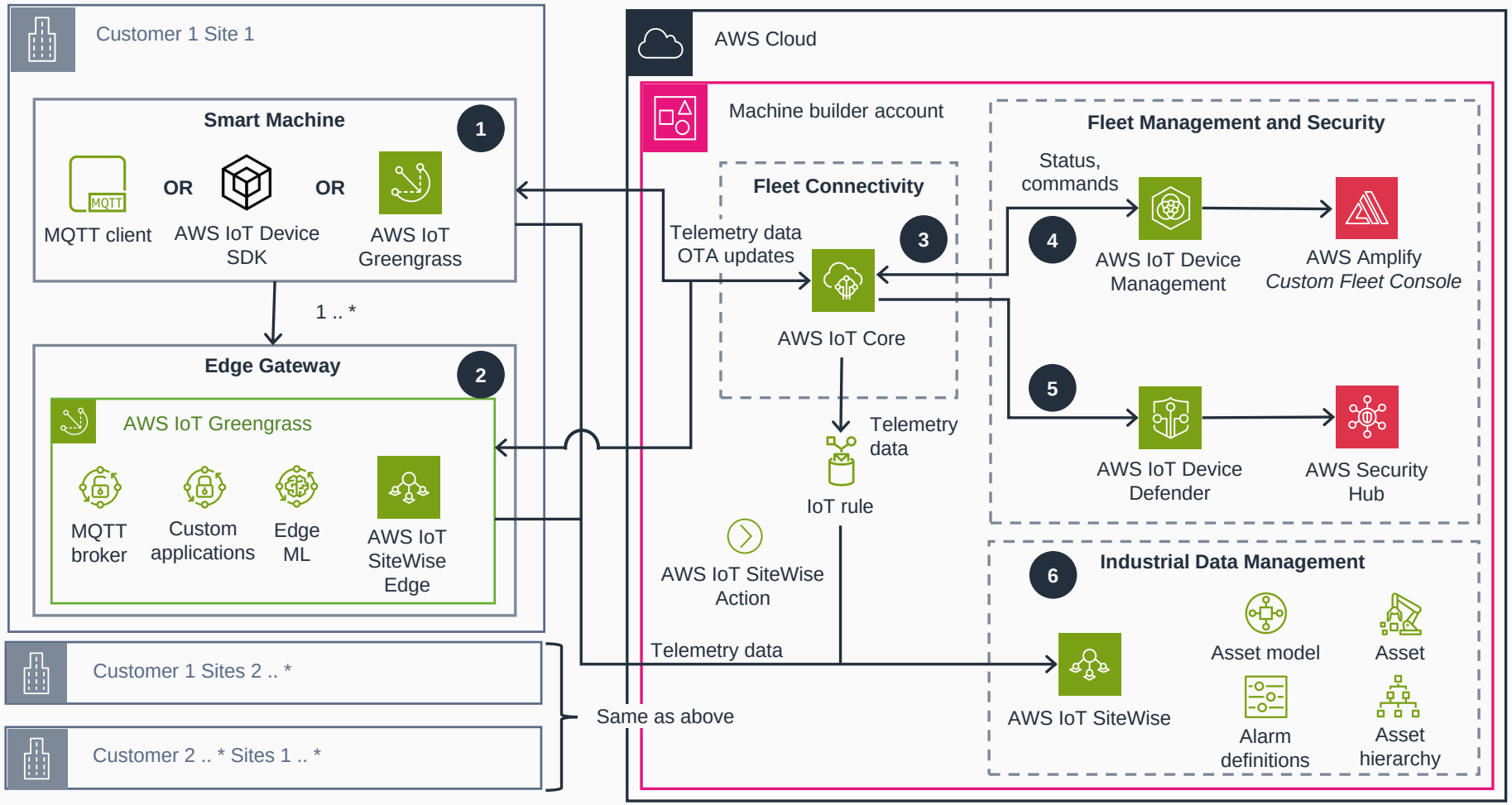
Guidance for Deploying Smart Machines on AWS

Connect and Manage Machines

This architecture diagram shows the process of connecting smart machines, remotely managing them, and constructing an industrial data management layer. The following slides show further details on building a data foundation and managing the device lifecycle.



1. A smart machine connects to **AWS IoT Core** using a Message Queuing Telemetry Transport (MQTT) client, AWS IoT Device SDK, or the edge runtime provided by **AWS IoT Greengrass**. The telemetry data is then ingested into **AWS IoT SiteWise** directly through **AWS IoT SiteWise Edge** or through **AWS IoT Core**.

2. If the machine lacks direct internet connectivity, use an edge gateway as a cloud connectivity layer. The edge gateway collects data from the machines, data historians, applications, then processes, stores, and forwards it to the AWS Cloud. Run custom applications and ML inferences at the edge.

3. Facilitate scalable two-way communication between machines or edge gateways and the AWS Cloud, without the need to manage infrastructure, using **AWS IoT Core**.

4. Remotely provision, monitor, update, and troubleshoot machines or edge gateways by leveraging **AWS IoT Device Management**. Build a custom fleet management console using **AWS Amplify** to visualize your fleet, and search across it to view machine state and health data.

5. Audit your fleet for compliance with security best practices and continuously monitor it using **AWS IoT Device Defender**. Any security findings are sent to **AWS Security Hub** for a centralized view of all security issues from various AWS services.

6. Ingest and contextualize operational data from your machines using **AWS IoT SiteWise** data streams and modeling capabilities. Additionally, compute performance metrics, store timeseries data, create alarm definitions, and provide flexible data access to external applications.

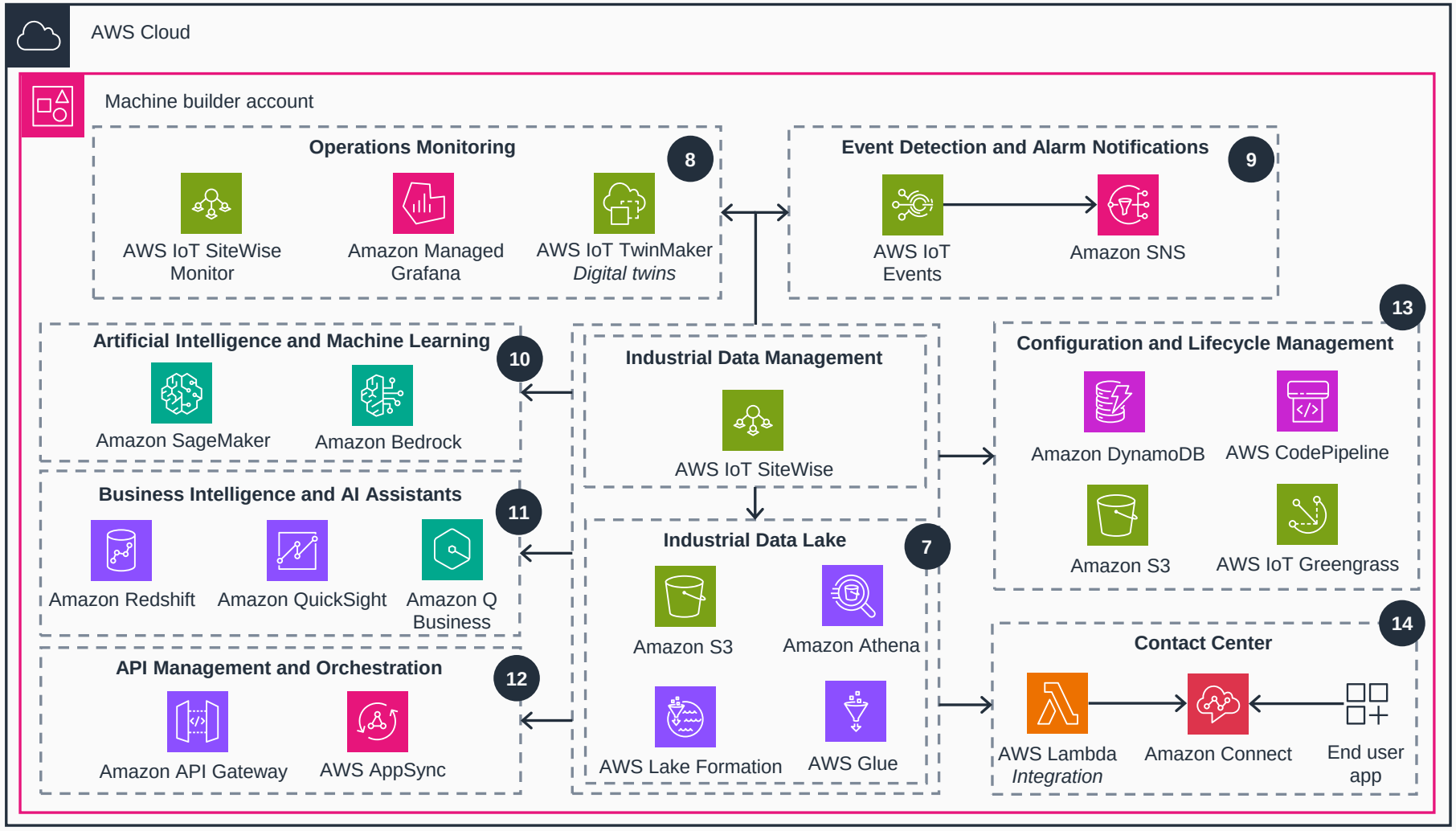
Reviewed for technical accuracy October 21, 2024

AWS Reference Architecture

Guidance for Deploying Smart Machines on AWS

Build an Industrial Data Foundation

This architecture diagram demonstrates how the industrial data foundation can enable operations monitoring, alarm notifications, AI/ML models, business intelligence dashboards and reports, AI assistants, APIs, lifecycle management—empowering contact center agents with contextual machine information.



AWS Reference Architecture

7. Build an industrial data lake using the contextual data from **AWS IoT SiteWise**. Govern, secure, and share data using **AWS Lake Formation** for advanced analytics. Catalog and analyze data with services like **AWS Glue** and **Amazon Athena**.

8. Remotely monitor machines using **AWS IoT SiteWise** Monitor or with **Amazon Managed Grafana** for rich, contextual dashboards. Build digital twins powered by **AWS IoT TwinMaker** to improve equipment performance.

9. Notify operational personnel about the health of machines using **AWS IoT Events** and **Amazon Simple Notification Service (Amazon SNS)**. Create state machines and event monitoring applications with **AWS IoT Events**.

10. Develop AI/ML solutions for predictive maintenance with **Amazon SageMaker** and build generative AI solutions using **Amazon Bedrock**.

11. **Amazon QuickSight** enables data-driven decisions. With the **Amazon Q** add-on, business users can ask natural language queries for quick insights. Empower employees with enterprise information using **Amazon Q Business**.

12. Provide historical and real-time machine data to customers by building serverless APIs using **Amazon API Gateway** and **AWS AppSync** that can scale to millions of users.

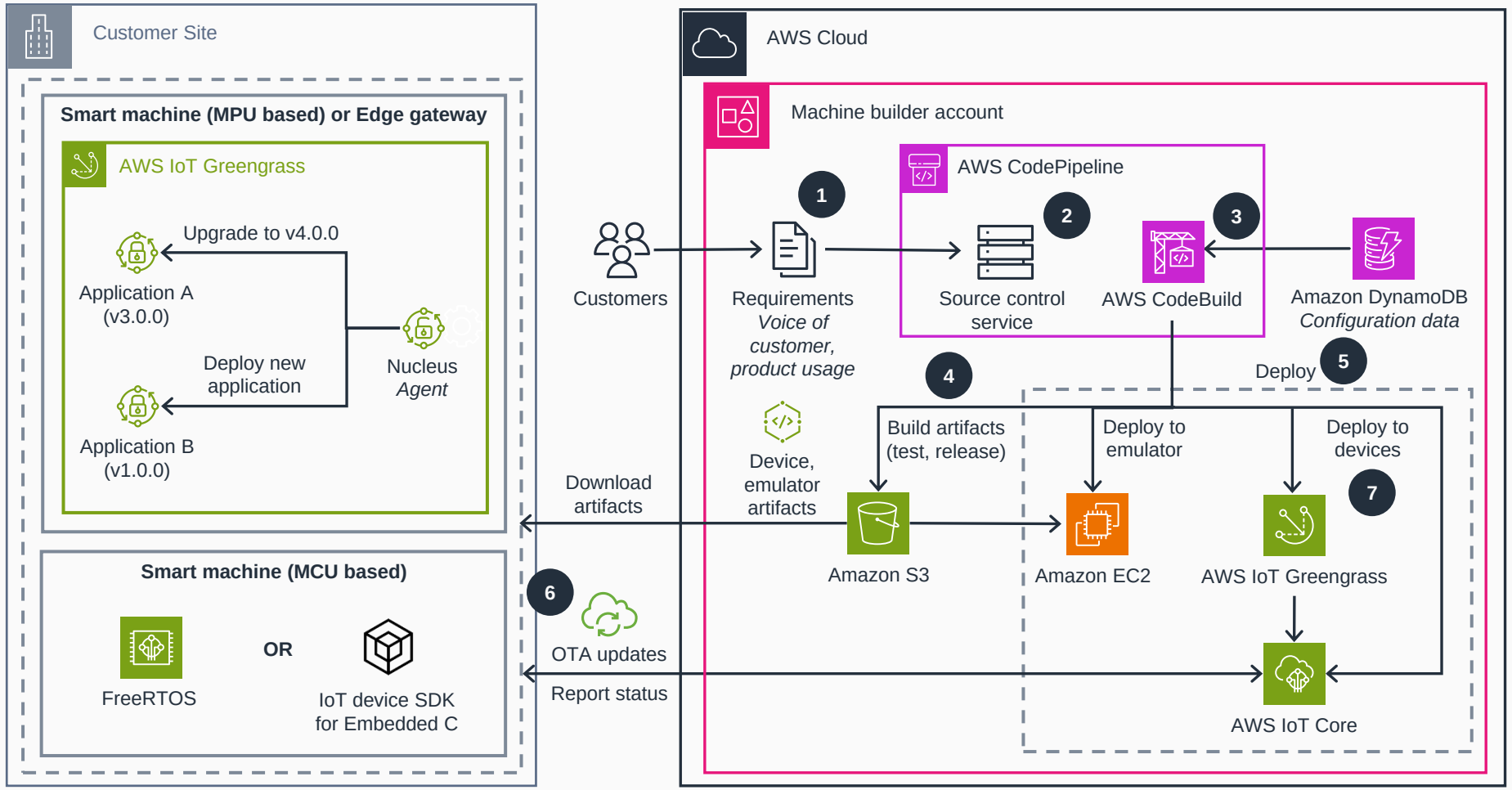
13. Use **Amazon DynamoDB** for storing machine configuration, **AWS CodePipeline** for automating continuous integration and continuous delivery (CI/CD), **Amazon Simple Storage Service (Amazon S3)** for storing artifacts, and **AWS IoT Greengrass** for managing edge devices.

14. Leverage **Amazon Connect** to meet customer service needs and empower agents with contextual machine information.

Guidance for Deploying Smart Machines on AWS

DevOps Lifecycle Management

This architecture diagram illustrates the process of enhancing machine capabilities and resolving issues through over-the-air (OTA) updates, leveraging an automated CI/CD pipeline that involves various stages of development, including build, test and deployment. This DevOps lifecycle helps close the loop to quickly respond to customer needs in the market.



1. The machine builder gathers requirements through Voice of Customer feedback and product usage analysis in an effort to enhance machine capabilities or resolve ongoing issues.
2. Software developers and embedded developers make changes to the source code hosted by source control services such as GitHub, GitLab, and Bitbucket.
3. Leverage **AWS CodeBuild** with cross-build tools to create artifacts for devices and emulators. **DynamoDB** provides the necessary machine-specific configuration. **CodePipeline** automates the CI/CD process by orchestrating various stages of development.
4. Store the artifacts meant for testing and production release securely in **Amazon S3**.
5. Test the artifacts by deploying them to emulated environment and a test group of physical devices. Emulated environments can be created using emulators such as Quick Emulator (QEMU) and Arm Virtual Hardware (AVH) on **Amazon Elastic Compute Cloud (Amazon EC2)**. Use thing groups from **AWS IoT Core** to organize the test devices for testing.
6. Devices receive over-the-air (OTA) updates from **AWS IoT Core** and securely download the necessary artifacts from **Amazon S3** using pre-signed URLs or MQTT file streams. They then update the firmware or software and report the status back to **AWS IoT Core**. The machine builder verifies the update for improved security, usability, reliability, and functionality and then approves it.
7. Deploy approved artifacts to all devices with configurable rollout rates and schedules, and monitor continuously during and after deployment.

Reviewed for technical accuracy October 21, 2024

AWS Reference Architecture