C3.ai

Digital Acceleration

How the C3 AI Suite Accelerates AI Application Development by 40x or More

Summary

As organizations apply artificial intelligence (AI) to drive digital transformation, they are seeking ways to accelerate AI application development and deployment in order to achieve faster time-to-value. The current trend in building AI applications is to assemble native services of a general-purpose cloud service provider (CSP) like AWS, Microsoft Azure, Google Cloud, or IBM Cloud. An alternative approach is to use the C3 AI Suite—a software suite purpose-built for developing enterprise AI applications.

C3.ai commissioned an expert third-party developer firm to build a predictive maintenance AI application using both approaches. The third-party team found that using the C3 AI Suite reduced the total development effort by a factor of 40 compared to the assemble-on-CSP approach. They concluded that many enterprises would realize significantly greater reductions in time and effort—by a factor of 50 to 100—using the C3 AI Suite.

The C3 AI Suite was able to achieve 40x or greater acceleration over the assemble-on-CSP approach due to five key factors:

- Cohesive design – purpose-built for AI application development
- Model-driven architecture – provides an abstraction layer that removes complexity
- Time series engine – automatically time-aligns all data across all sources
- AI-centric processing – data flow and analytics engine optimized for AI applications
- Built-in bindings to tools, languages, and frameworks preferred by developers and data scientists

This paper describes the comparison project, providing a detailed breakdown of the development effort using both approaches, and explains how the C3 AI Suite accelerated each stage of the process.

The Digital Transformation Imperative

Today's organizations increasingly recognize the need to digitally transform in order to survive and thrive in the 21st century. Digital transformation requires the ability to create new value by harnessing four major technology vectors—elastic cloud computing, big data, artificial intelligence, and the internet of things. An essential requirement is the infusion of AI into every aspect of an organization's operations.

Rapid advances in AI in recent years have made it possible to achieve step-function improvements in business processes throughout the value chain. In industry after industry, organizations are now vying to establish an early lead in AI. Organizations that successfully apply AI can realize hundreds of millions to billions of dollars in annual economic value through increased operational efficiency, reduced costs, higher customer satisfaction, and new revenue streams.

Some of this value accrues from applying AI to relatively simple data sets and business processes like sales forecasting, customer churn, and facility energy management. But the vast majority of the value will come from applying AI to complex data sets and business processes, such as supply network and inventory optimization, process optimization, maintenance optimization, fraud detection, and many other areas.

These complex AI applications require an architecture that ingests data from multiple devices (e.g., sensors, controls, machines) and transactional systems in near-real time; blends together and enhances these data sets with master data from an array of enterprise, operational, and third-party data sources; uses the unified data to train predictive and optimization models to generate actionable insights; and embeds these insights into a business process. A typical enterprise or large organization will deploy hundreds, perhaps thousands of these AI applications across its operations.

How to Accelerate Digital Transformation?

The critical question for organizations is how to accelerate digital transformation in order to capture benefits early, establish first-mover advantage, and guard against being outpaced by competitors.

Approach 1: Assemble on CSP

The current trend for developing enterprise AI applications is to take a "building blocks" approach by assembling native microservices of cloud service providers like Amazon Web Services (AWS), Microsoft Azure, Google Cloud, or IBM Cloud. In this approach, the end-customer (and/or its consulting partner) first integrates the CSP's microservices as a foundation to design, develop, host, and operate an application, followed by building bespoke application logic.

Approach 2: C3 AI Suite

C3.ai offers a different approach. The C3 AI Suite is a purpose-built platform with pre-integrated services, and is designed with a model-driven architecture. It offers a cohesive, low-code/no-code development environment with a complete and comprehensive set of tools and services to design, build, deploy, and operate advanced, enterprise-scale AI applications.

These applications can use the infrastructure provided by any cloud service provider—AWS, Azure, Google Cloud, or IBM—as well as bare metal servers in an organization's data center or private cloud. The C3 AI Suite is designed and purpose-built specifically to enable rapid development and deployment of AI applications and efficient operation and maintenance of those applications over time.

To demonstrate the difference in these two approaches, C3.ai commissioned an expert third party—a Premier AWS Consulting Partner, with AWS competencies in Big Data and Machine Learning—to build a predictive maintenance AI application using both approaches. The firm has developed and deployed hundreds of applications on AWS for many Fortune 2000 customers. The developers on this project each had several years of AWS experience.

The Project Assignment and Application Specifications

A team of highly skilled developers was assigned to build a simple Predictive Maintenance Application for Light Bulbs using two approaches: (1) native AWS services (the "AWS Application") and (2) the C3 AI Suite (the "C3 Application"). The team recorded and compared the time and effort required to develop the identical application using AWS and the C3 AI Suite.

The objective of the application is to predict the likelihood of light bulb failure within the following 30 days from a given point in time. The industry best practice for making such predictions is to train a machine learning model using the provided data. With this model in place, predictions must be generated as new data are received for each bulb.

In both cases, the developer team sought to build an application that:

- ingests, unifies, and federates the raw data;
- processes the data;
- trains a machine learning model to predict which light bulb is likely to fail in the next 30 days;
- provides an application user interface.

The data sets provided for the application included:

- Bulb type, wattage, location, manufacturer, and date of manufacture
- Power grid status
- Bulb fixture location
- Bulb telemetry including watts, lumens, voltage, and temperature
- Bulb event history
- Bulb fixture data

Building a risk prediction model for each light bulb required the telemetry/measurement data to be analyzed over time. For example, the application uses the following time series:

- Average Lumens per Smart Bulb – Light generation over time for the smart bulb
- Average Power per Smart Bulb – Power usage over time for the smart bulb
- Duration On per Smart Bulb – The total amount of time (in hours) that a light bulb has been switched on up to the interval
- Switch Count per Smart Bulb – The number of times a bulb is switched on or off
- Power Grid Status per Building – An external factor indicating whether the local power grid was functional over time at a specific building

To make the predictions actionable, the assignment required the application to present information to end-users through an interface with two screens and a total of seven displays reporting the number, location, risk score, and status of light bulbs.

Breakdown of the Development Time and Effort

The charts below illustrate the developers' actual timeline with each approach. As clearly indicated, building the application using AWS services took 3 developers 15 weeks, and a total effort of 200 full-time-equivalent (FTE) days. By contrast, using the C3 AI Suite took just 1 developer a week for a total effort of only 5 FTE days.

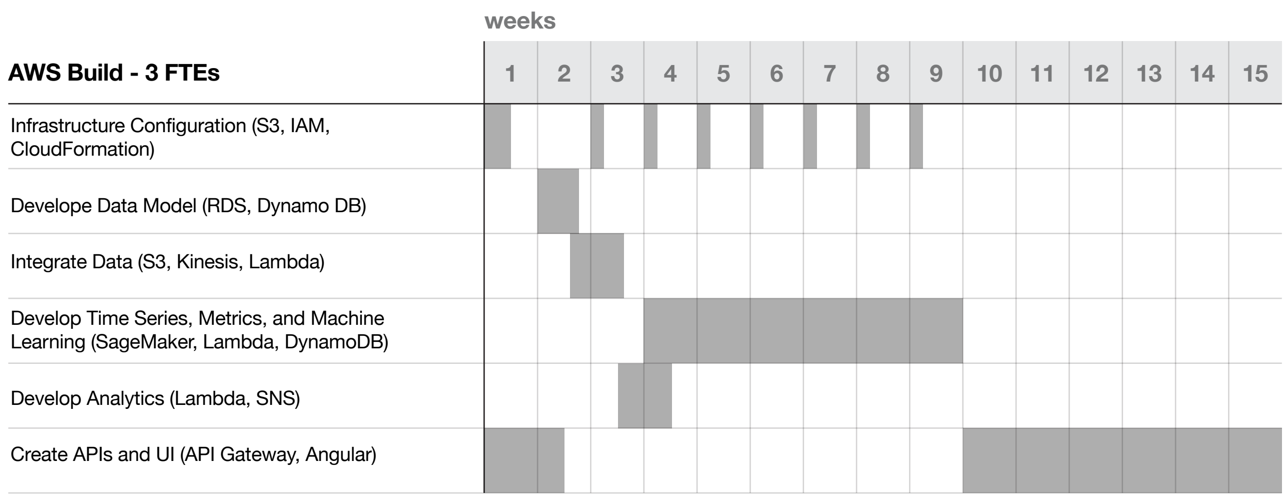

Figure 2. Timeline to implement the AWS Application.

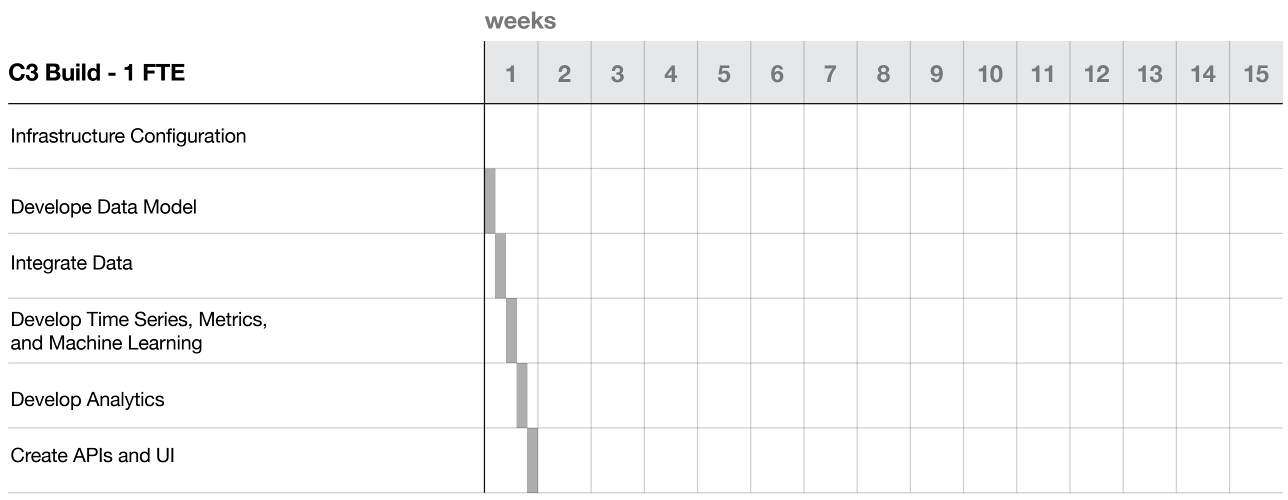

Figure 3: Timeline to implement the C3 Application.

The team concluded the C3 AI Suite reduced the effort and accelerated the development process by a factor of 40, while also reducing development risks. “For less experienced teams, this could easily increase to 50–100 times,” the developers concluded.

“Further, C3.ai solves the challenges of security, extensibility, and scalability—while streamlining the skills needed in an enterprise development team.”

This paper describes the comparison project and the third-party developers’ findings, which are summarized in the table below:

Task	**Assemble on CSP**			**C3 AI Suite**		
	Days	FTEs	Total	Days	FTEs	Total
Infrastructure Configuration	20	3	60	0	0	0
Data Model	10	2	20	0.75	1	0.75
Data Integration	5	3	15	0.75	1	0.75
Analytics and Machine Learning	20	2	40	2.5	1	2.5
User Interface and Testing	32.5	2	65	1	1	1
Total Effort (FTE Days)			**200**			**5**
Total Lines of Code Written			**83,000**			**1,450**

Figure 1. The C3 AI Suite provided a 40x acceleration in developer productivity compared to building the same application using only native AWS services, with 98% less code.

How the C3 AI Suite Accelerates AI Application Development

The C3 AI Suite is able to dramatically accelerate AI application development for five key reasons:

1. Cohesive, Purpose-Built Product Proven in the Most Demanding Conditions

The C3 AI Suite has been refined, tested, and proven over nearly a decade in the most demanding industries and production environments—electric utilities, manufacturing, oil and gas, and defense—comprising petabyte-scale data sets from thousands of vastly disparate source systems, massive volumes of high-frequency time series data from millions of devices, and hundreds of thousands of machine learning models. Unlike assembling applications on general-purpose CSP platforms such as AWS, the C3 AI Suite offers a development environment that is optimized specifically for AI and IoT application development. All of the tools and services provided by the C3 AI Suite work together seamlessly. They are part of a cohesive product, which is continuously maintained and updated by C3.ai as a fully unified suite. As a result, developers can learn to use the C3 AI Suite in a very short period of time, typically one to two weeks of training. There is no need to wade through numerous pages of documentation, or to master technical details of a dozen or more underlying components as is the case with the assemble-on-CSP approach.

2. Model-Driven Architecture

The C3 AI Suite is designed with a model-driven architecture. This provides an abstraction layer and semantics to represent the application, which removes the complexity that developers must otherwise handle when using the assemble-on-CSP approach. A model-driven architecture frees developers from worrying about underlying technical details such as data mapping, integrations, and computational processes like stream processing, machine learning pipeline management, security, and so on. The model-driven architecture enables developers to design and build applications by using C3 Models. These are representations of any concept or entity that is relevant to the application and the interrelationships among them. Anything can be represented as a Model—even, for example, applications including databases, natural language processing engines, and image recognition systems. All Models have RESTful interfaces. This level of abstraction vastly simplifies and speeds application development.

3. Time Series Engine

Many AI applications involve very large amounts of time series data. Working with time series data can be exceptionally complex for developers. Because the C3 AI Suite is designed with a powerful time series engine at its core, it removes that complexity by auto-managing everything required to prepare and process time series data. The C3 AI Suite treats all data as time series data, automatically time-aligning and normalizing data across all sources, significantly reducing development complexity and effort.

4. AI-Centric Processing

The C3 AI Suite includes a data flow and analytics engine specifically designed for supporting AI application processes. This relieves developers from having to write extensive and often complex code for a range of common AI requirements. For example, the C3 AI Suite understands data lineage to the features of a machine learning model and therefore will automatically update features, as necessary, only when new data are available. This not only simplifies the development effort but also results in better application performance, scalability, and resource and cost efficiency.

5. Bindings to Multiple Tools, Languages, and Frameworks

The C3 AI Suite is open and designed with prebuilt bindings to many popular tools, languages, and frameworks, including Python, Jupyter, R, Eclipse, JavaScript, Angular, and React, among others. This allows developers and data scientists to use their preferred tools, languages, and frameworks on a common platform, thereby speeding application development.

Multi-Cloud and Polyglot Cloud Support

While AWS was the CSP used in this comparison project, the results and conclusions of this report would apply to any other comparison, whether Azure, Google, or IBM. Similar to AWS, every CSP provides a large general-purpose collection of microservices. Using the assemble-on-CSP approach with any of those platforms, development teams would face the same complexities.

C3.ai asked the third-party consulting firm commissioned for this project to also provide a quote to build the same application on Google Cloud and on Microsoft Azure. The quotes were virtually identical to the AWS project effort in terms of time, resources, coding effort, and cost.

In contrast to the assemble-on-CSP approach, the C3 AI Suite provides multi-cloud support: an application built using the C3 AI Suite can use any CSP platform as infrastructure with little or no modification. Moreover, the C3 AI Suite also provides polyglot cloud support—i.e., the ability to use microservices from different cloud platform providers in an application. This capability not only allows application portability from one cloud vendor to another, but also affords the capability to run applications on multiple clouds simultaneously. As CSP vendors continue to innovate, an organization can choose best-of-breed microservices across multiple vendors to optimize the capability of its AI applications. When a new and more powerful microservice becomes available, it can easily be plugged in to replace the old microservice. The application keeps running, now with greater performance, precision, and economic benefit.

Conclusion: A Faster Choice

As demonstrated by this third-party side-by-side comparison, the C3 AI Suite, with its model-driven architecture, accelerates AI application development by 40x or greater over the assemble-on-CSP approach. With the C3 AI Suite, organizations can achieve faster time-to-value with significantly less cost and complexity.

Appendix A: Architecture Comparison

Assemble on AWS

The architecture for the application assembled on AWS, as depicted in Figure 4, made heavy use of AWS managed services, including AWS Lambda for serverless processing, Amazon Kinesis for data streaming, Amazon S3 for storing raw data, Amazon API Gateway for RESTful services, and Amazon SageMaker for machine learning training and inference. In addition, Amazon's Relational Database Service (RDS) and Amazon DynamoDB, a NoSQL distributed key-value store database, were utilized for persistence.

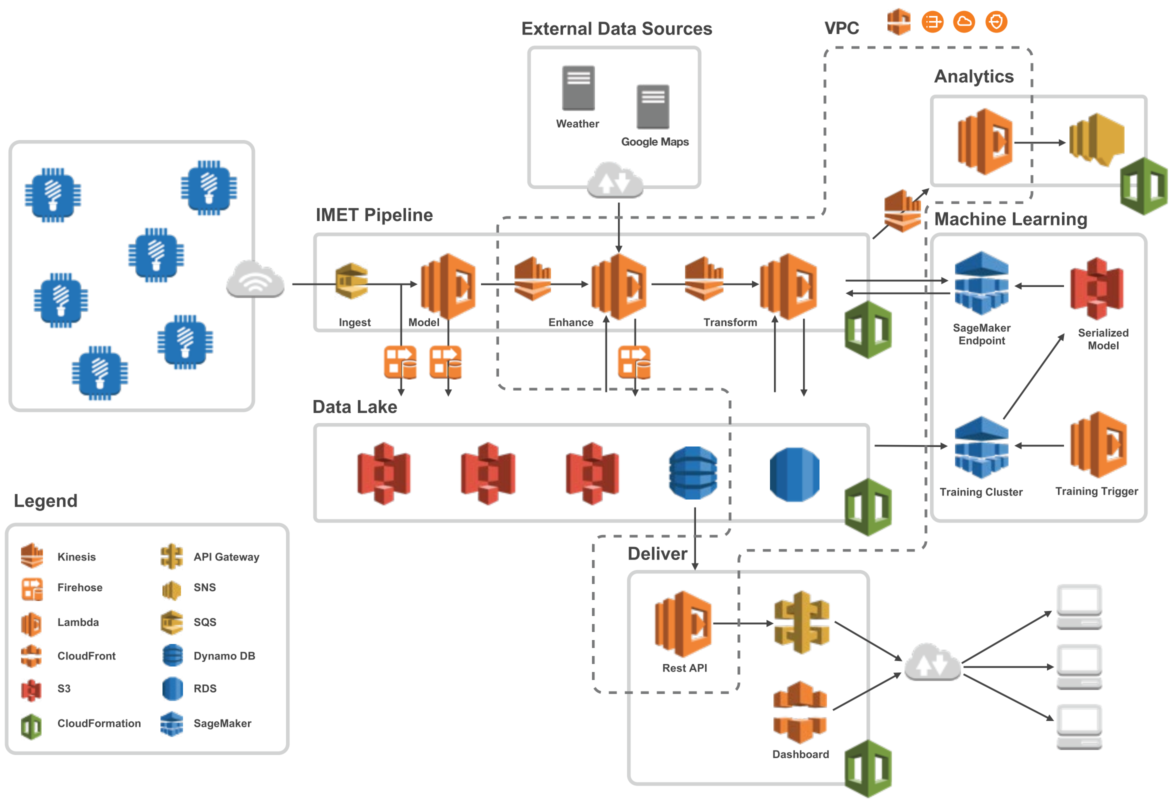

Figure 4: Architecture to build the Light Bulb Predictive Maintenance on AWS.

Appendix A: Architecture Comparison

C3 AI Suite Build

Developing the same application with the C3 AI Suite was much simpler. Learning the C3 AI Suite and the use of C3 Models required just eight days of training.

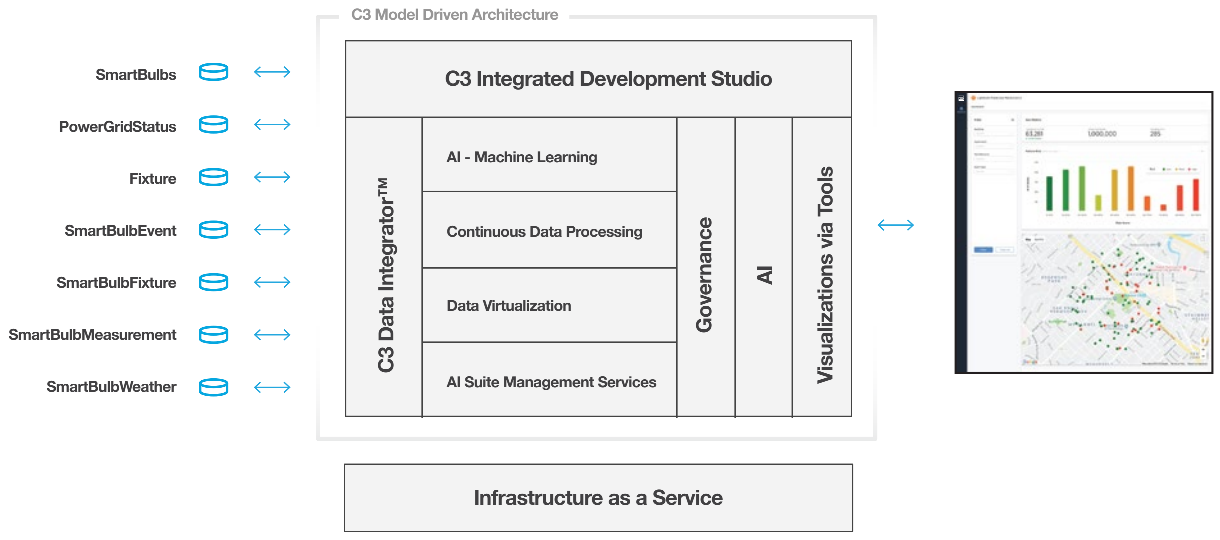

Figure 5. Architecture to build the Light Bulb Predictive Maintenance on the C3 AI Suite

Appendix B: Third-Party Developer Team's Key Findings

The third-party developer team summarized its key findings from the comparison project in its comments below.

Time and Cost of Development

The C3 AI Suite substantially accelerates development, allowing customers to derive economic value from AI applications faster than by building the necessary platform components themselves.

Risk Mitigation

Even organizations with well-funded IT departments and highly skilled developers face substantial risks building, deploying, and maintaining new applications. This risk increases many-fold when the scope grows beyond one simple application. The C3 AI Suite reduces exposure to the array of risks that face a company developing custom, enterprise-scale software.

Complexity

Stitching together core infrastructure, enterprise software, platform services, data science services, and UI components into a production-scale application is a task of enormous complexity. Companies that attempt to construct custom AI platforms using building blocks risk pouring time and resources into projects that fail to reach production or deliver on their value proposition. The C3 AI Suite, by abstracting away the underlying infrastructure and presenting all data and services as accessible, manipulatable "Models", minimizes this risk.

Extensibility

Companies that overcome the complexity of building a single AI application face the risk that their work will not be extensible—the data model cannot be extended to serve a second application, or their code is specific to infrastructure components that have become obsolete. Applications built using the C3 AI Suite are tied to metadata rather than to the underlying data and storage infrastructure. Because of this decoupling, the applications are therefore fully extensible to support new data sources, infrastructure, algorithms, and platform services.

Security

Getting a working application is complex enough. Encrypting all data in transit and at rest; limiting access to data at the row and user level; and guaranteeing backup, failover, and redundancy add new layers of complexity and represent major risks for applications that expose the enterprise's most sensitive data. The C3 AI Suite mitigates these risks by providing accredited, comprehensive security measures as part of its services.

Maintenance and Support

AI applications built from scratch are brittle, breaking when data sources, use cases, and support staff change. Companies are forced to divert resources to maintenance, which dilutes the value that AI applications deliver, limits new development, and increases the probability of abandoning applications altogether. C3.ai mitigates this risk in two ways: The C3 AI Suite requires significantly less maintenance than a bespoke solution, and C3.ai provides support resources and training as part of its solution.

Scalability

Infrastructure scalability is crucial to enable applications built on small data sets to scale to the enterprise. AWS enables scalability typically through a manual process to request, provision, and integrate additional compute and storage resources. The C3 AI Suite employs a modular scale-out architecture that automatically requests, provisions, and releases computing resources based on need and makes it simple to add more storage and other resources. Application scalability is also crucial to enable organizations to add more data, new sources of data, new transforms of data coming from different parts of the organization, or new application logic to extend previously built use cases. Native AWS development would require structural code changes and rewriting the entire application to incorporate these changes. C3.ai mitigates this risk via the C3 AI Suite's metadata-based abstraction.

Resource Capabilities

AWS development requires a very broad skill set in an organization's developers. At the most basic level, all developers need an understanding of AWS development principles, while specific team members might require a range of skills—from networking and Linux server configuration to the specific details of the various managed services—unique to each CSP, that come with their own methods of utilization. Additionally, organizations need to make a significant resource investment in DevOps to ensure that what they are building on AWS can mature in a safe and scalable way. For reference, AWS recommends a year of experience working in the AWS platform prior to securing a basic AWS certification. For C3.ai, a foundation in object-oriented concepts such as inheritance and static typing as well as exposure to the domain-specific syntax are the only critical skills to understanding the development approach. A typical C3 AI Suite developer receives eight days of training and is usually proficient in three months depending on skill level and prior experience.

Appendix C: Third-Party Developer Team's Commentary on the Development Process

To further detail the comparison between using the assemble-on-CSP approach and the C3 AI Suite, the developer team described their experience at each stage of the process with both methods. Below, the developer team's commentary is presented side-by-side for each of the five major task areas. The developers' commentary provides numerous insights explaining the significantly greater complexity in using the assemble-on-CSP approach compared to the relative ease of using the C3 AI Suite.

Task	AWS Application	C3 Application
Infrastructure Configuration	Our work began with creating a new AWS account, configuring basic networking, and establishing access control and security policy permissions. As we began development, the data lake and necessary infrastructure for a data processing pipeline was created. The data lake consisted of Amazon S3, DynamoDB, and RDS. The processing pipeline was a series of AWS Lambda functions that were linked together by Amazon Kinesis Streams, allowing data to flow through each stage of ingestion, modeling, enhancement, and transformation. These resources were managed using Amazon's infrastructure-as-code service, AWS CloudFormation. The easiest component of the architecture to implement and deploy was the S3 data lake and its associated data stores, RDS and DynamoDB. This was one of seven CloudFormation templates, each growing in complexity as we built up the architecture layers. Each CloudFormation template ranged from hundreds to thousands of lines of custom code.	The C3 AI Suite does not require any infrastructure to be configured or maintained. Deploying a new instance of the C3 AI Suite takes four hours. Deploying a new tenant within an existing instance takes approximately three minutes.

Task	AWS Application	C3 Application
Infrastructure Configuration (continued)	Deployment was a manual process that involved uploading the templates into CloudFormation and then running them to create or update the infrastructure. The AWS documentation reasonably described which resource parameters are necessary for a specific template resource, and the AWS console provides specific feedback if an attempted template upload has errors or requires refactoring. While we were able to understand and respond to these issues rapidly given our experience with AWS, our assessment is that a typical organization would need to build very robust DevOps pipelines and devote significant resourcing to ensure changes are promptly and definitively pushed into the account. **An enterprise architecture team with less experience on AWS would reasonably take at least twice as long as our experienced developer team.** It is notable that infrastructure configuration is a continuous process throughout the development life cycle and requires ongoing maintenance post-deployment. Each AWS service we utilized has unique networking and permissions configurations that must be tweaked and debugged.	
Data Model	To optimize for scalability, extensibility, and usability, we designed the AWS Application to leverage two databases. We used Amazon RDS, a fully managed relational database service, to store static structured data—bulb type, wattage, location, manufacturer, etc. To create our relational data model, custom SQL was hardcoded to define each table in our database. This manual process provided the necessary structure to easily store, organize, and query data. We used Amazon DynamoDB, a fully managed NoSQL distributed key-value store database, to store dynamic data—e.g., bulb telemetry. Unlike relational databases, NoSQL databases do not require a predefined data model.	We began building the C3 Application by creating C3 Models for use in our application. C3 Models are representations in code of any business-relevant objects—for instance, real-world entities that make up a business—in this case, light bulbs, buildings, facilities, manufacturers, etc. Each Model contains the metadata that define its relevant datastores (distributed file system, relational, NoSQL) and its relationships to other Models in the data model (e.g., one facility has 10 light bulbs from a single manufacturer).

Task	AWS Application	C3 Application
Data Model (continued)	Without two databases, we would be limited in our ability to filter on data, explore data, and evaluate the timed interval relationships between data objects. It would also couple the scaling of both data sources to a single configuration setup. By separating the two storage services, we can scale DynamoDB independently of RDS for both improved performance and cost savings compared to running a single, larger, and more expensive RDS instance. A two-database architecture was in sync with our microservice-based approach. **An enterprise architecture team with less experience on AWS and internal constraints on database structure would reasonably take twice as long as our experienced team. Further—and this is extremely crucial given our experience—changing or extending the data model would require the data model to be entirely refactored/rebuilt.**	The use of C3 Models allows individuals with different functions and specializations—e.g., developers, data scientists, and business analysts—to work on a shared abstraction layer without having to configure or maintain the underlying data federation and storage models, dependencies, or infrastructure.
Data Integration	Our initial data integration effort involved manually loading the raw CSV data via MySQL tooling. If the data were in a different format it would have likely involved writing either custom parsers, or manually converting the data into a more usable format combined with potentially writing the raw SQL commands to insert this data. Furthermore, this process was entirely manual at the outset of the project to "seed" the initial data with no process in place for the ongoing ingestion of new data. While this would not be a difficult process to engineer, it would require more development time along with a process to be put in place to allow for more fixtures, light bulbs, apartments, etc. to be introduced into the system for future use. **An enterprise architecture team with less experience on AWS would reasonably take 50 percent more time than our team.**	We used the C3 AI Suite's native data integration capabilities to integrate, index, and normalize the light bulb data. Prior to integrating data, we created six canonical Models for each of the data sources. The C3 AI Suite includes native functionality to import data from any source and map all fields to C3 Models for access by data scientists and developers. While we worked with CSV files, the C3 AI Suite includes prebuilt connectors to commonly used relational databases, NoSQL databases, and distributed file systems.

Task	AWS Application	C3 Application
Machine Learning and Analytics	Once we integrated the raw data, we began preparing it for our machine learning process. To create our time series metrics/machine learning features, we wrote custom logic in NodeJS and used Amazon Lambda's serverless computing service to execute at run-time. Lambda allowed us to easily deploy our custom logic and orchestrate it with AWS streaming service Kinesis, with less overhead and infrastructure configuration. However, our team encountered difficulty configuring the networking for the various AWS services we utilized. While a serverless approach typically removes the need for networking considerations, using Amazon RDS necessitated hands-on networking configuration. Our Lambda functions had to be placed into subnets within our VPC, which then required establishing VPC Endpoints to connect out to the AWS managed services such as AWS Key Management Service, DynamoDB, and SageMaker. In addition, we had to configure a new network address translation (NAT) gateway to facilitate our enhancement function to make calls over the internet to the weather endpoint provided by C3.ai. These changes required more manual configuration of the networking layer and could prove to be problematic for teams without a strong knowledge of AWS networking concepts. Additionally, this led to extra work that was not estimated and required unplanned developer time. Another key difficulty was transforming our time series metrics into the proper format for Amazon SageMaker. Rectifying this required a trial and error process and relying on colleagues with extensive experience working with SageMaker and its DeepAR algorithm. After revising and refactoring our approach for building the raw model input, we were able to successfully create a process for building machine learning models and for creating a request object that integrated with SageMaker.	We used our C3 Models to generate 13 metrics, which fetch Model data to produce a normalized time series. Metrics serve as features in machine learning algorithms and can be incorporated into application logic. We also wrote some methods for the SmartBulb Model, which allow for more complex calculations on the data using JavaScript or Python. We created risk-of-failure scores for the Application's light bulbs using Jupyter Notebook and Python, both supported natively by the C3 AI Suite. Having the full functionality of the C3 AI Suite and C3 Models natively integrated with Jupyter Notebook provides easy access for data scientists to leverage tools that are familiar and effective. We trained a classification model that regressed the metrics SwitchCountWeek and DurationOnInHours against the dependent variable WillFailNextMonth to calculate the probability of failure in the next 30 days. We stored this rolling risk score as its own time series metric RiskScore. Machine learning algorithms in the C3 AI Suite operate on all existing data, create new data that can be automatically attached to a C3 Model for future processing, and automatically update training and make predictions on the latest available data. The area under the receiver operating characteristic curve was .990. We used the C3 AI Suite's native asynchronous processing engine to create data flow events (DFEs). Using DFEs, we created three analytics that automatically generate operator alerts when certain operating thresholds were met/exceeded. These alerts could be routed via email or SMS messages.

Task	AWS Application	C3 Application
Machine Learning and Analytics (continued)	This effort involved complex custom code, and changing the algorithm required significant rework of the preparation process. The lack of usability is one reason an iterative approach needed to be taken until the process fully integrated with SageMaker DeepAR. Once the model and request object were successfully created and integrated, it was easy to get predictions from SageMaker. However, additional code was required to join the predictions into the data model and store in Amazon RDS. Precision relates to the proportion of light bulb failures that were correctly predicted. Recall is the proportion of actual high-risk bulbs that were identified correctly. The area under the receiver operating characteristic curve was .795. **An enterprise architecture team with less experience on AWS, especially with AWS networking concepts, would reasonably take twice as long as our team did.** A separate process was created to evaluate and save metrics based on specific rules and use cases. Once a light bulb measurement had been ingested and transformed, an analytics service written in Lambda was used to check if any rules were satisfied. If so, a new record was saved into RDS to mark the analytic as triggered, and a message was sent to Amazon Simple Notification Service. This allows for emails, text messages, or other notifications to be triggered so that further action can be taken as a result—e.g., “Inspect this light bulb.” Expanding the notifications simply involves writing further use cases and incorporating them into the analytics service. **A normal enterprise architecture team would reasonably take 50 percent more time than our team did.**	

Task	AWS Application	C3 Application
User Interface and Testing	Building the Application's UI required exposing RESTful APIs that served the results of our time series metrics. To accomplish this, we utilized Amazon API Gateway with Lambda functions written in NodeJS. The API Gateway configuration was completed in CloudFormation using an OpenAPI specification combined with specific configuration elements for the API Gateway. Configuring cross-origin resource sharing (CORS), which enables the client application to call the APIs directly from the browser, in the CloudFormation templates was challenging. We reverse engineered the methods and headers that the API Gateway console automatically adds when enabling CORS and figured out the corresponding CloudFormation syntax. We spent many iterations to successfully set up and test CORS. The Light Bulb API accessed RDS and DynamoDB, requiring different data access methods to be written for each database and different sets of permissions that needed to be setup and configured in the CloudFormation templates. It would be ideal to establish an architecture that abstracts the data access methods; however, it would require development time to create and maintain the methods and configuration to access those data stores. The API was secured with an API key, which also required moderately complex CloudFormation configuration. The UI components were built using Angular 6 and were hosted with S3 and Amazon CloudFront. TypeScript, SASS, RxJS, Angular Material, and the Angular FlexLayout were the primary front-end technologies utilized for the Angular components. The components can be easily added and removed without affecting the other components on the page. We spent one FTE day to ensure that duplicate API calls would not be made for a component if another component already retrieved the data. The components share a service that provides the API results as a RxJS observable. The observable provides the API data for the	We incorporated several of our C3 Models and metrics in a web interface built using custom C3 HTML and UI templates. We used these to create the dashboard of the Application. The dashboard UI template was one JSON styled file that contained the code for the components of the dashboard such as a status map, a filter, a histogram, and a table. Our UI also included continuously and automatically updated predictive risk scores about the likelihood of smart bulb failures (incorporated using the RiskScore metric). Finally, we created a few simple potential roles that would be used by future users of the Application. These roles consisted of restricting users to permissions for specific use cases pertinent to the user.

Task	AWS Application	C3 Application
User Interface and Testing (continued)	components and will refresh the components that are subscribed to it when new data is generated. This allows the components to efficiently retrieve data refreshed without a postback for actions such as filtering. We used client-side filtering, sorting, and paging for the detail tables, but these actions will need to be done on the server side if the quantity of data for the component becomes too large. Server-side filtering, sorting, and paging would add one week to the development effort, plus two to three days for unit testing. Hosting the front-end application on S3 alone, while possible, does not provide enough granular control over permissions and routing. We used CloudFront as an entry point to S3 and restricted access to only allow CloudFront access to the S3 bucket via an Origin Access Identity. Through CloudFront, we can enable client-side routing functionality and manage the access, custom error messages, and geographic distribution. All the S3, CloudFront, and Origin Access Identity setup was done through a CloudFormation template. **An enterprise architecture team with less experience would reasonably take 20 percent more time to complete these tasks.**	