O'REILLY®

Blockchain Tethered AI

How to Tether Artificial Intelligence with Smart Contracts and Tamper-Evident Ledgers

Early Release

RAW & UNEDITED

Karen Kilroy,
Deepak Bhatta
& Lynn Riley

O'REILLY®

Blockchain Tethered AI

How to Tether Artificial Intelligence with Smart Contracts and Tamper-Evident Ledgers

Early Release

RAW & UNEDITED

Karen Kilroy,
Deepak Bhatta
& Lynn Riley

Blockchain Tethered AI

How to Tether Artificial Intelligence with Smart Contracts and Tamper-Evident Ledgers

With Early Release ebooks, you get books in their earliest form—the authors' raw and unedited content as they write—so you can take advantage of these technologies long before the official release of these titles.

Karen Kilroy, Deepak Bhatta, and Lynn Riley

Beijing • Boston • Farnham • Sebastopol • Tokyo

Blockchain Tethered AI

by Karen Kilroy, Deepak Bhatta, and Lynn Riley

Printed in the United States of America.

Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

O'Reilly books may be purchased for educational, business, or sales promotional use. Online editions are also available for most titles (http://oreilly.com). For more information, contact our corporate/institutional sales department: 800-998-9938 or *corporate@oreilly.com*.

- Acquisitions Editor: Michelle Smith
- Development Editor: Corbin Collins
- Production Editor: Gregory Hyman
- Interior Designer: David Futato
- Cover Designer: Karen Montgomery
- Illustrator: Kate Dullea
- May 2023: First Edition

Revision History for the Early Release

- 2022-09-02: First Release
- 2022-10-07: Second Release

See http://oreilly.com/catalog/errata.csp?isbn=9781098130480 for release details.

978-1-098-13048-0

Preface

This book is intended for software architects and developers who want to write AI that can be kept under control. It assumes that the reader already has an understanding of AI systems and is aware of the concerns that arise from the release of these systems. It is also assumed that the reader is somewhat familiar with blockchain and how it works.

In order to complete the exercises, the reader should first become familiar with NodeJS, Hyperledger Fabric, and TensorFlow, PyTorch, or another popular AI library, and be able to set up an appropriate development environment in which to perform the exercises.

Why Does AI Need to Be Tethered?

As a member of the software engineering community, you may already believe that the potential benefits of AI are vast, but that if we aren't careful, AI could destroy humanity. This book explains how you can build effective, non-threatening AI by strategically adding blockchain tethers.

A *blockchain tether* is like a log, in that we record designated events in it that can later be referenced. Traditional troubleshooting usually means you peruse a log, pinpoint a problem that occurred, cross-check other logs to learn when and why the issue started, and later monitor the logs to be sure the issue is fixed. This method becomes more complex with AI, since the data and models are sourced from various origins, and some AI can make changes to itself through machine learning and program synthesis. By using blockchain to *tether* AI – that is, restrain it –, we can set restrictions and create audit trails for AI that help stakeholders and consumers to better trust it, and when problems arise, to help engineers to figure out what happened and wind it back.

This preface introduces you to why and how to build blockchain tethers for AI. We touch on what you'll find in each of the chapters and what you will learn in each one. And we tell you why we wrote this book, give a shout out to future generations, and explain why we think you are critical to creating AI that does not make people afraid.

To be clear, in our view there is no reason to be afraid of AI unless we build AI that we should fear.

As the software engineers and architects of the world, we hold the power to build tethered AI that is understandable, governable, and even reversible.

We can build AI that remembers its original intent and follows ground rules set by its creators, even long after they pass away; AI that explains its critical actions and reasoning, exposes its own bias, and can't change the past to cover its tracks – not even to protect itself.

AI is watching, ready to jump in and guide the helpless human race, like a mama duck lining up her newborn ducklings. AI is poised to turn fierce, and pounce upon us like a wolf in sheep's clothing. AI is ready to save a world it doesn't understand – or wipe us all out. We just aren't certain. This book is about how to be more certain.

What You Will Learn

In this book you will learn how to tether AI by building blockchain controls, and you'll see why we suggest blockchain as the toolset to make AI less scary.

Chapter 1 explores the questions of why we need to build an AI *truth machine*. A truth machine involves attempting to answer questions like these:

- Are you a machine?
- Who wrote your code?
- Can we fix your bias?

- Are you nice?
- Who tells you what to do?
- Can we turn you off?

We'll look at the trust deficit that has arisen, and explore the reasons why AI is frequently perceived as a threat to humanity.

Chapter 1 also discusses superintelligent agents, and talks about how they act on our behalf and why they need to be under our control. Then we dig into requirements for trusted AI, and talk about how to turn some of yesterday's discussions into today's functional AI utilities.

The first chapter weighs other factors that bring forward the need for a truth machine, such as *program synthesis*, which is the ability for code to write itself with no help from a human. We look at how to avoid *technological singularity*, which is AI's point of no return, when it escapes from human control and grows beyond our ability to influence or understand it..

Chapter 1 defines the *far-reaching bias problem that skews the output of AI* and looks at possible approaches for identifying and correcting it. We examine the crisis of inadequate or non-existent identification and talk about ways to correct this using blockchain. AI vulnerabilities and hacks are rarely mentioned, so we wrap up by exploring what to watch out for and take a look at some compromised AI.

Chapter 2 discusses blockchain controls for AI. This is where we begin our deep dive into how to develop blockchain tethers to solve the issues brought up in Chapter 1 and make four controls that will tether AI. Chapter 2 wraps up by exploring ways to explain these controls and gain cooperation from your stakeholders.

Chapter 3 discusses user interfaces for a blockchain-tethered AI system. Among these are web applications, smartphone apps, and integrations with other systems. We explore ways to address security and talk about ways to implement blockchain tethers in your own interfaces.

Chapter 4 is where we plan our approach for a sample blockchain-tethered AI project. We create a requirements document for the project, define the audit trail, define the user interface workflow and how the consent will be managed, and plan the smart contracts that will automate decision making.

Put on the coffee and order the pizza for Chapters 5 and 6, because these chapters implement our plan to build a basic AI system and add blockchain controls to it. We review the project that was planned in Chapter 4, make sure the development environment for building AI controls is ready to go, and then do hands-on exercises inbuilding and training an AI model, instantiating a blockchain, and configuring buckets to hold AI artifacts and logs.

The final chapter, Chapter 7, dives into recording critical AI touchpoints to blockchain, and then auditing the AI system using blockchain.

The book wraps up with a discussion on how to use the blockchain audit trail to revert training done on a poorly behaved AI model.

This book frequently refers to *governance*. The governance portions will help you to see how what you are building must be sustainable long into the future. That is because *governance*, the act of gaining and recording consent from a group of stakeholders, is key to keeping blockchain and AI strategies on track. *On-chain* governance – having these workflows pre-programmed into a blockchain-based system with predefined key performance indicators (KPIs) and approval processes – helps a group to make their legal agreements into long-lasting procedures.

As you can see from all of this, your role as software architect/developer is paramount in making sure that the right underlying infrastructure and workflows are in place so your AI system can operate as intended for many years to come.

Why We Wrote this Book

We wrote this book to share our knowledge and understanding of how to control AI with blockchain, because we believe there can be great benefits

to using AI wisely. With AI, we have the ability to grok vast volumes of data that would otherwise be inactionable. AI brings us the ability to make decisions that could vastly improve our quality of life (see Figure P-1).

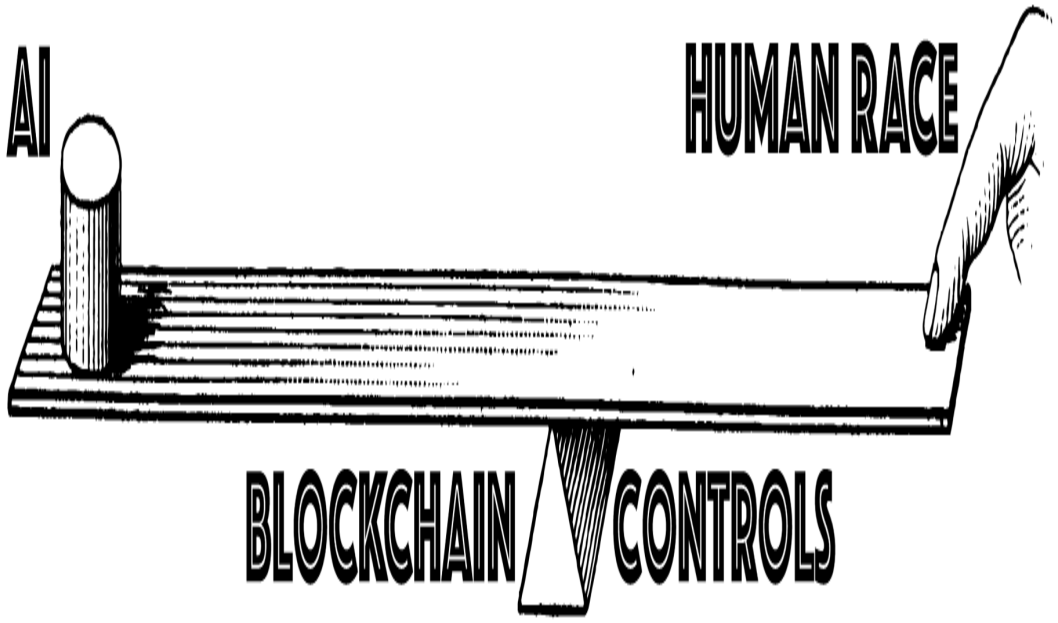


Figure P-1. We need the right amount of leverage over AI so we can gain the benefit of its use, but still keep it under our control.

We think AI needs to be tethered because it supplies the critical logic for so many inventions. AI powers all the possible combinations of exciting technologies like robots, automated vehicles, drones, systems that educate and entertain, do our farming, care for our elderly, and allow us to work in new ways – inventions that will change our lives beyond our wildest dreams.

Your three authors, Karen Kilroy, Deepak Bhatta, and Lynn Riley, have worked together for years at Kilroy Blockchain, developing enterprise-level products that use blockchain technology to prove data authenticity. This interest began in 2017 with Kilroy Blockchain's award-winning AI app RILEY, which won the IBM Watson Build Award for North America.

RILEY (shown in Figure P-2) was developed to help students who are blind and visually impaired understand the world around them by hearing descriptions. We thought that especially for this type of a user group, we

would want to be certain that the AI had not undergone tampering or corruption, and that RILEY always stays true to its intent of helping this population.

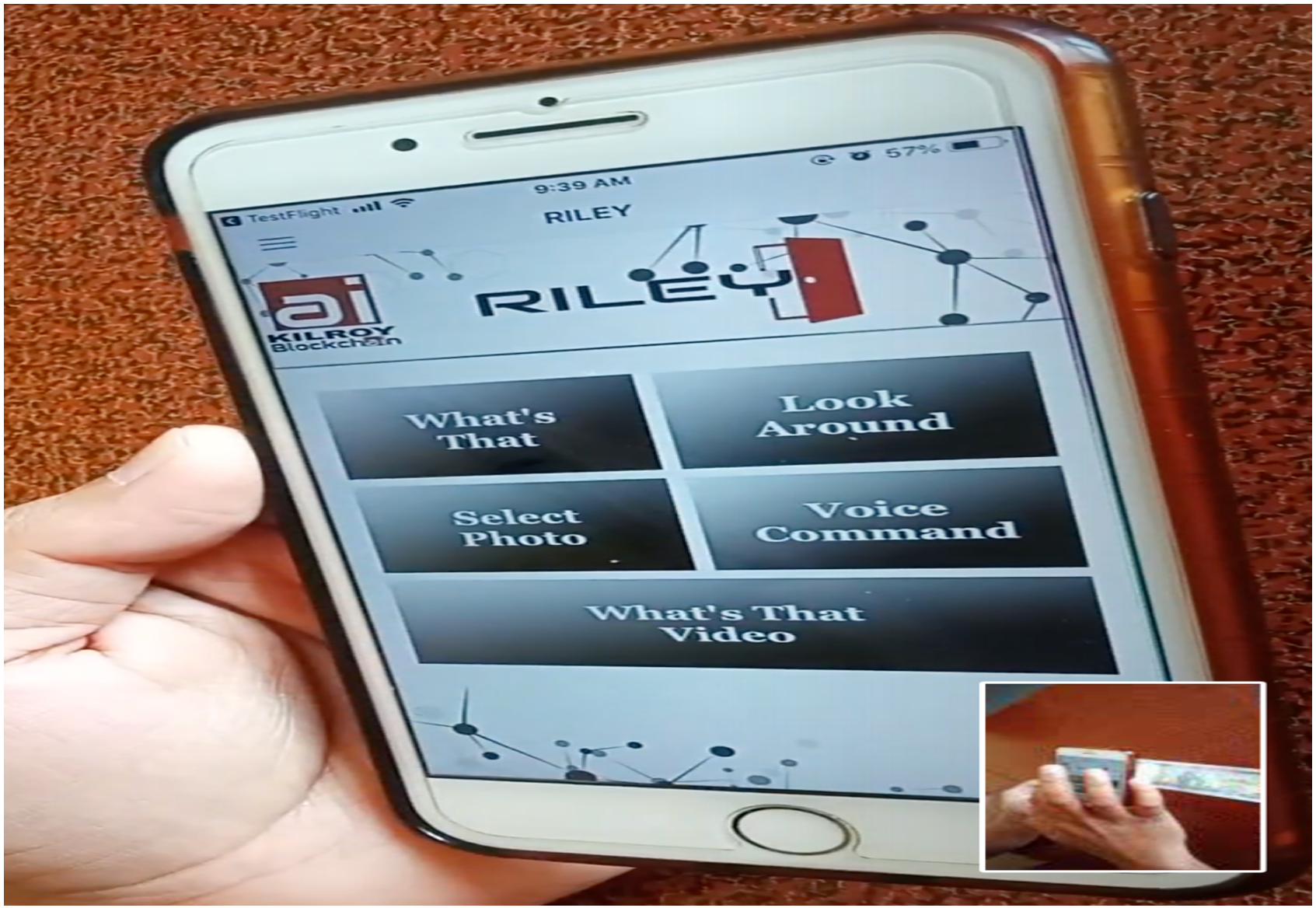

TestFlight
9:39 AM
57%
RILEY
RILEY
KILROY
Blockchain
What's That
Look Around
Select Photo
Voice Command
What's That Video

Figure P-2. RILEY's visual recognition functions include What's That, Select Photo, and What's That Video. In the inset image, RILEY is being used to identify and describe currency.

Back in 2017 we began asking a lot of questions to AI experts about AI's vulnerabilities; and instead of giving easy answers, the AI experts only raised more questions and concerns. We were told there was no method or utility that was being used to make sure that AI's data - its lifeblood - never undergoes tampering by a bad or sloppy actor. Worse yet, there was no standard method of checking whether or not algorithms had undergone similar tampering.

And as with all other programming, there wasn't reliable identification for AI developers and system administrators, or for other machines and intelligent agents. Any of the existing methods of proving identity, like a name and password associated with a code repository, could be easily spoofed in the first place or changed after the fact without detection.

There was no way to decide how decisions would be made in the future, or who would be authorized to make those decisions. There was no way that an AI system could shut itself down due to ethics concerns, such as if a money-making stakeholder such as a group of shareholders, refused to consent to turn it off.

Now, five years later, there is *still* no standard way to do these things. It is becoming apparent that this situation needs to change, and that data in all workflow systems needs to be AI-ready. That way, when the data is consumed by AI, it is already tamper-evident.

Kilroy Blockchain's recent products such as CASEY (a blockchain-based system for managing campus safety policies) and FLO (a blockchain-based general workflow system) use blockchain as a control. When we add AI to CASEY or FLO, the data that has been produced by these systems over time is already tamper-evident. All significant events are already time-stamped and stored in linked blocks. Nodes of the blockchain can be distributed to stakeholders, and the blockchain history can be viewed by authorized users.

Although AI has been around for 50-60 years, it is just starting to get more powerful, primarily because the hardware has finally advanced to where it can support AI's ability to scale.. Now, when we run machine learning programs as massively parallel, we may get results that we don't expect as the AI learns at an accelerated pace. Dealing with a computer system that could potentially outsmart us would be something totally new to the human experience, and that could be what we have in our future if we don't prevent it now.

Public expression of fear of AI is coming from the most widely recognized technical leaders of our time. It seems the greater the knowledge of what makes AI tick, the greater the fear.

Given all of this, we ask: Why don't we just fix it?

A Note to Future Generations

In his classic scifi novel *The Hitchhiker's Guide to the Galaxy*, Douglas Adams describes an essential space traveler's guide to the universe. Using the guide can unlock secrets and help the hitchhiker move through the universe unscathed. We hope our book evolves into becoming the AI engineer's guide, with secret answers inside.

We want to help you avoid the problems of trying to control a man-made intelligence that has become tough to manage, and help keep you from being forced to reverse-engineer systems that have written themselves.

There is a song by David Bowie on the *Man Who Sold the World* album, called "Saviour Machine." Karen has known this song since she was a child (back in the 20th century), but it didn't make sense to her until lately. The song caught her attention again because we have a "President Joe" now in 2022, and it is a fictional President Joe's dream of AI that is the subject of this song, written in 1970.

Bowie's "Saviour Machine" is an AI whose "logic stopped war, gave them food" but is now bored and is wistful about whether or not humans should trust it because it just might wipe us out. The machine cries out "don't let

me stay, my logic says burn so send me away." It goes on to declare "your minds are too green, I despise all I've seen" and ends with the strong warning "you can't stake your lives on a Saviour Machine."

Hopefully this song is a lot gloomier than the actual future. If we have done our job well, this book will help today's engineers build reliable back doors to AI, so the people of your generation and many more to come can enjoy AI's benefits without the risks we currently see.

By the end of this book, you will have a good idea of how to build AI systems that can't outsmart us. You'll understand the following:

- How AI can be tethered by blockchain networks
- How to use blockchain cryptoanchors to detect common AI hacks
- Why and how to implement on-chain AI governance.
- How AI marketplaces work and how to power them with blockchain
- How to reverse tethered AI

As the AI engineer building the human back door, you will be able to plan a responsible AI project that is tethered by blockchain, create requirements for on-chain AI governance workflow, set up an artificial intelligence application and tether it with blockchain, use blockchain as an AI audit trail that is shared with multiple parties, and add blockchain controls to existing AI.

Your due diligence now could someday save the world.

Conventions Used in This Book

The following typographical conventions are used in this book:

Italic

Indicates new terms, URLs, email addresses, filenames, and file extensions.

`Constant width`

Used for program listings, as well as within paragraphs to refer to program elements such as variable or function names, databases, data types, environment variables, statements, and keywords.

`Constant width bold`

Shows commands or other text that should be typed literally by the user.

`Constant width italic`

Shows text that should be replaced with user-supplied values or by values determined by context.

> **TIP**
>
> This element signifies a tip or suggestion.

> **NOTE**
>
> This element signifies a general note.

> **WARNING**
>
> This element indicates a warning or caution.

O’Reilly Online Learning

> **NOTE**
>
> For more than 40 years, *O’Reilly Media* has provided technology and business training, knowledge, and insight to help companies succeed.

Our unique network of experts and innovators share their knowledge and expertise through books, articles, and our online learning platform. O'Reilly's online learning platform gives you on-demand access to live training courses, in-depth learning paths, interactive coding environments, and a vast collection of text and video from O'Reilly and 200+ other publishers. For more information, visit *https://oreilly.com*.

How to Contact Us

Please address comments and questions concerning this book to the publisher:

- O'Reilly Media, Inc.
- 1005 Gravenstein Highway North
- Sebastopol, CA 95472
- 800-998-9938 (in the United States or Canada)
- 707-829-0515 (international or local)
- 707-829-0104 (fax)

Email *bookquestions@oreilly.com* to comment or ask technical questions about this book.

For news and information about our books and courses, visit *https://oreilly.com*.

Find us on LinkedIn: *https://linkedin.com/company/oreilly-media*

Follow us on Twitter: *https://twitter.com/oreillymedia*

Watch us on YouTube: *https://www.youtube.com/oreillymedia*

Chapter 1. Why Build a Blockchain Truth Machine for AI?

A NOTE FOR EARLY RELEASE READERS

With Early Release ebooks, you get books in their earliest form—the authors' raw and unedited content as they write—so you can take advantage of these technologies long before the official release of these titles.

This will be the first chapter of the final book.

If you have comments about how we might improve the content and/or examples in this book, or if you notice missing material within this chapter, please reach out to the editor at *ccollins@oreilly.com*.

Today's *intelligent agents* – software programs driven by AI that perform some domain-specific function – are used in law enforcement and the judicial system and are gaining significant authority in deciding the fate of humans. It is becoming increasingly difficult for people to detect AI when it is acting as a human agent. A product of AI - the intelligent agent - can be embedded in interactions that people don't associate with AI, such as calling 911 or an insurance claims adjuster. It has been said that once AI becomes mainstream, we will no longer hear the term mentioned, but even now, a person may not realize they have encountered an intelligent agent, much less how to hold it accountable for its actions.

This chapter dissects AI's trust deficit by exploring critical facts to track in order to improve trust, and suggests ways that you can think of your AI projects as having an interwoven blockchain truth machine - built into every

aspect of the AI. It goes on to explore concerns about machine learning (ML), potential attacks, and vulnerabilities, and touches briefly on risk and liability. Blockchain is explained and blockchain/AI touchpoints are identified based on critical facts.

Dissecting AI's Trust Deficit

The collection of critical facts about any AI system resembles an online cake recipe - it lists ingredients, instructions on exactly how to mix and bake it, and helpful information like who wrote the recipe, where this type of cake is best served (birthday, picnic, etc.), nutritional information, and a photo of what the cake is supposed to look like when it is done. Because it is online you might also see what it looked like when others made it, read comments from those who have eaten the cake, and gather ideas from other bakers. This is very different from when you eat a finished piece of cake that you didn't make; you either eat it or not based on whether you trust the person who handed it to you. You don't worry much about how the cake was made.

But what about trusting powerful AI without knowing how it was made? Should we just accept what it hands us? Do we trust AI too little to ever find its most beneficial applications, or do we trust it too much for our own good? It is very hard to tell, because AI comes in many varieties, is made of many components, comes from many different origins, and is embedded into our lives in many different ways.

Often, AI has been tested to operate well in some conditions but not in others. If AI had an accompanying list of facts, similar to an OSHA Safety Data Sheet used for hazardous chemicals, it would be much easier to know what is inside, how it is expected to perform, safe handling guidelines, and how to test for proper function. *AI factsheets* are similar lists of facts, as they contain human-readable details about what is inside AI. Factsheets are intended to be dynamically produced from a *fact flow* system, which captures critical information as part of a multi-user/machine workflow. Fact

flow can be made more robust by adding blockchain to the stack because it fosters distributed, tamper-evident AI provenance.

Critical facts for your factsheet may include the following:

- Your AI's purpose
- Its intended domain: where and when should it be used?
- Training data
- Models and algorithms
- Inputs and outputs
- Performance metrics
- Known bias
- Optimal and poor performance conditions
- Explanation
- Contacts (described below)

Consider these fact types, and then remove ones that don't apply or add your own. You can also list a fact type with a value like "black box" or "not yet established."

Purpose

When a project is started, a group of stakeholders generally get together and decide the *purpose* of their project, which includes defining high-level requirements of what they want to accomplish. Sometimes the purpose will change, so it is good to do occasional spontaneous sanity checks and record those as well.

Intended Domain

Similar to purpose, the *intended domain* describes the planned use of the AI. Specifically, the intended domain addresses the subject matter in which the AI is intended to be an expert (healthcare, ecommerce, agriculture, and so on). Sanity checks that probe for domain drift are good to include here. An example of an intended domain is a sensor that is designed for dry weather - if it is operated in wet weather, it is being used outside of the intended domain.

Training Data

Sets of related information called *training data* are used to inform the AI of desired results and to test the AI for those results. The training data can come from many different sources and be in many different formats and levels of quality. This chapter further explores data integrity and quality in the "Bias" section, and Chapter 2 discusses it in more detail.

Models and Algorithms

A *model* is a set of programmatic functions with variables that learn from an input to produce an output. The model is trained using data sets that tell it both how and how not to behave. The model is composed of many different *algorithms,* which are AI's underlying formulas, typically developed by data scientists. The model learns from many training data sets from various libraries and sources, which have been introduced to the model by different parties in various abstractions. Once a model is in production, there is no standard way to track what all goes on behind the scenes, and models frequently come from third party marketplaces, so models and algorithms are often referred to as *AI's black box*. Chapter 6 discusses how an AI model is coded.

Inputs and Outputs

Input describes the type of stimulus the system expects to receive. For instance, a visual recognition system will expect images as input. *Output* is

what the system is supposed to produce as a response. A visual recognition system will produce a description as output.

Performance Metrics

These are specifications on exactly how the AI is supposed to perform, including speed and accuracy. They are generally monitored with analytics systems.

Bias

AI bias, which is explored in Chapter 2, is one of the biggest issues facing AI. When we consider bias, our thoughts generally jump to race and gender; while those are very serious issues in AI, they are only part of the bias that people, and subsequently AI, hold. Exposing any known bias and/or tests and processes to avoid bias is helpful in making the AI transparent and trustworthy.

Optimal and Poor Conditions

AI that performs well in certain conditions may perform poorly in others. In the world of AI-driven automated vehicles (AVs), there is a defined operational design domain (ODD) that the AI can be expected to perform well within. For instance, a particular AI might work well as part of a lidar system when the weather is clear and dry, but when it is overcast and raining, performance might decrease drastically.

Explanation

This explains the interpretability of the AI's output, or states that it operates as a black box and does not provide explanations for its output.

Contacts

This is who to contact in case support, intervention or maintenance is needed with the AI. This ties to identity, which Chapter 2 explores in detail.

Machine Learning Concerns

With a traditional computer program, you write the code with specific rules and parameters and test it with specific data sets. The code is run and produces an output that may or may not be graphically represented. In an ML program, the code is written and run multiple times with specific training data sets to teach the program what the rules are, and those rules are then tested with test data sets, and this iterative process continues until the ML program produces an output - a *prediction* - with confidence levels that are at a high-enough level to prove the program has learned how to identify the data in the proper context.

The classic example of this is IBM data scientists building and training Watson to play on the show *Jeopardy!*, where an answer is displayed and the player has to come up with the right question to match. This required the data scientists to come up with the right algorithms for the ML program to teach Watson to understand the *Jeopardy!* answer in the proper context, instead of merely acting as a fast search engine spitting out multiple questions - it needed to find the question with the best fit in the proper context at a very high confidence level. That is the essence of machine learning: the data sets teach the program how to learn, and the ML program responds to data input with a high confidence output, as illustrated in Figure 1-1.

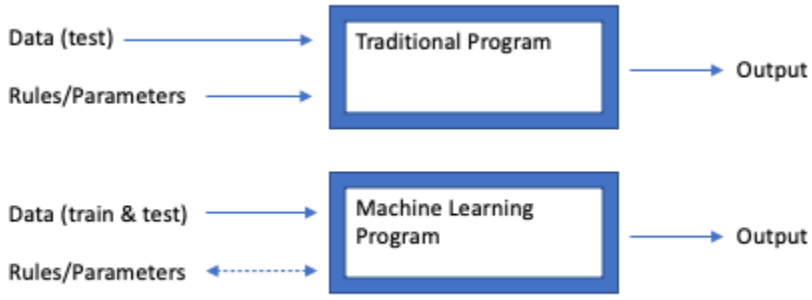


Figure 1-1. Input and output of a traditional software program vs. ML.

> **TIP**
>
> To learn more about IBM Watson's training for *Jeopardy!* watch "How IBM Built a Jeopardy Champion" on YouTube. At its core, it is a story of how scientists taught Watson to recognize context in order to increase confidence levels of their answers – or actually, questions, as it is on *Jeopardy!*

Black Box Algorithms

In everyday corporate use, ML prediction can be used in many applications, such as for forecasting the weather or deciding on the right moment to drop an online coupon on a regular customer while they're browsing a retail website. One of the ML algorithms that can be used for both scenarios is the *Markov chain*. A Markov chain is a discrete stochastic process (algorithm) in which the probability of outcomes is either independent, or dependent on the current or directly preceding state.

This whole process of making predictions based on algorithms and data and finding appropriate output is usually only partially visible to any one ML team member, and is generally considered to operate as a black box, of which the exact contents are undetectable. Algorithms like the Markov chain model are tuned by behind-the-scenes mathematicians adjusting complex formulas, which is indiscernible to nearly everyone else involved. This section uses the Markov chain model to illustrate how training raw number data sets really work and then shows how classification data sets work.

> **NOTE**
>
> Although in statistics results are referred to as *probabilities*, in machine learning we refer to results as *confidence levels*.

For dice throwing, assuming the dice aren't loaded – whatever you just threw does not affect what you're about to throw. This is called a *random walk Markov chain*. Your guess of the next throw not being improved by

your knowledge of the previous one showcases the Markov property, the memoryless property of a stochastic process. In other words, in this case, $P_{left} + P_{right} = 1$ (or $P_{left} = 1 - P_{right}$).

A simple model is a two-state weather prediction: sunny or cloudy. A 2 x 2 *transition matrix P* describes the probability of what the weather will be tomorrow based on what it is today (Figure 1-2).

$$P = \begin{bmatrix} 0.9 & 0.1 \\ 0.5 & 0.5 \end{bmatrix}$$

Figure 1-2. A simple model showing weather probability.

This probability can be seen in Figure 1-3.

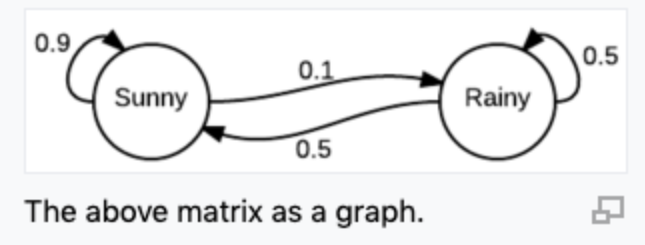


Figure 1-3. A graphical representation of a simple two-state weather prediction model.

We can make several observations here:

- The total output of either state adds up to 1 (because it is a stochastic matrix).
- There is a 90% probability that tomorrow will be sunny if today is sunny.
- There is a 10% probability that tomorrow will be rainy if today is sunny.
- There is a 50% probability that tomorrow will be rainy or sunny if today is rainy.

Today is Day 0 and it is sunny. In the matrix, we represent this "sunny" as 100% or 1, and thus the "rainy" is 0% or 0. So, initial state vector *X0* = [1 0].

The weather tomorrow on Day 1 can be predicted by multiplying the *state vector* from Day 0 by the *transition matrix* shown in Figure 1-4.

$$\mathbf{x}^{(1)} = \mathbf{x}^{(0)}P = \begin{bmatrix}1 & 0\end{bmatrix}\begin{bmatrix}0.9 & 0.1\\ 0.5 & 0.5\end{bmatrix} = \begin{bmatrix}0.9 & 0.1\end{bmatrix}$$

Figure 1-4. Model that multiplies the state vector by the transition matrix.

The 0.9 in the output vector *X1* tells us there is a 90% chance that Day 1 will also be sunny.

The weather on Day 2 (the day after tomorrow) can be predicted the same way using the state vector computed for Day 1, and so on for Day 3 and beyond. There are different types of Markov algorithms to suit more complicated predictive systems, such as the stock market.

This simple example shows how algorithms are created and data is iterated through it. More complicated models are made of far more complicated algorithms, or a group of algorithms to fit different scenarios. The algorithms are working in the background - the user, whoever they may be, doesn't see them at all, yet the algorithms serve a purpose. They require mathematical equations, generated manually or through programs specifically made to generate algorithms, and impact everything about the model and the ML pipeline. Finally, the models need to be tested thoroughly over a wide range of variability and monitored carefully for changes.

Each step involved in constructing the model could potentially undergo tampering, incorrect data, or intervention from a competitor. To fix this weakness would require some way to prove that the data and algorithms were from a reliable source and had not been tampered with.

To think of an algorithm in general terms, think of an old-fashioned recipe for baking a cake from scratch. A cake algorithm might give you a way to help others to combine a list of ingredients according to your specific instructions, bake it at the specified time and temperature in the correct pan that has been prepared the proper way, and get a consistent cake. The cake algorithm may even provide you with input variables such as different pan

size or quantity, to automatically adjust the recipe for a different altitude, or to produce bigger cakes or more of them at a time. The only content that this algorithm knows is what is hard coded (for example, 3 eggs, 3 cups of flour, a teaspoon of baking powder, and so forth). The writer of the recipe or someone who influenced them directly or indirectly had knowledge and life experience in how cakes work - what happens when you mix ingredients, what happens when baking soda becomes wet, mathematical measurements for each ingredient, correct proportions, how much of any ingredient is too much, and how to avoid the kind of chemical combinations that will make a cake fall.

In contrast to a general algorithm, AI algorithms are special in that they perform an action that involves training a model with ML techniques, using ample training data. In this case, an AI algorithm for winning a bake-off might look at data listing previous years' winning and losing cake recipes, and give you the instructions to help you bake the winning cake.

Data Quality, Outliers and Edge Cases

Data preprocessing is important, and data quality will make preprocessing easier. To identify a dog breed, for example, you'll need to have clear, close-up photos of the dog breed, and represent every possible appearance of the breed from different angles. If we were training an algorithm to understand text, the preprocessing would entail steps such as making data readable, making everything lowercase, and getting rid of superfluous words, among other things. But besides the input data, the classifications are critical, which brings us to the reason for the exercise of explaining Markov: this is how simple training works, but if you have inadequate training examples for inadequate classifications, it will propagate through with each iteration and you will create some kind of bias in your results.

One issue to keep in mind about the Markov chain: outliers, whether or not the origin is known, must be cut. The Markov algorithm was created in a way that doesn't tolerate outliers. If the model you're using has algorithms that don't process outliers, you would do well to think about running

separate iterations of ML with edge case data sets made of outliers, and use it to come up with error detection and error handling procedures.

In the case where there is no previous state or the previous state is unknown, the Hidden Markov model is used, and the initial array and output values are generated manually. When numbers aren't involved, different approaches must be taken to estimate the array and the output. A typical approach for training non-numbered data sets, such as in identifying objects, involves using Natural Language Processing (NLP) to classify the variables into an array like in the previous example, and create a desired output that can be fed back in during each iteration through an algorithm many times - up to 100,000 - until the result converges on a value that doesn't change much through repeated iterations (the number of standard deviations depends on what you're computing).

For example, you can train a set of images of a dog breed, and you would have to take into account the different attributes that distinguish that pure breed from other breeds, such as coat colors, head shape, ear shape, coat length, stance, eye location, eye color. The training sets would be made of images of thousands of dogs of that breed (a positive set) and images of dogs that are not that breed and other objects that might look like this dog breed but aren't (a negative set). Once the training converges to a confidence level that is high enough to ensure the code recognizes the breed, real world data can be run against it.

If you do not account for all facets of an object's appearance or a scenario, you will undoubtedly leave out classifications that would be critical for your training data sets. Poor confidence numbers during test runs should be an indicator of the need to improve training sets. Embarrassing mistakes that make it to the real world are another, undesirable result. Regardless of the model or algorithm, having poor data will test your data preprocessing skills and can hinder your model's ability to make sound predictions. It will also help to have an interpretable model or algorithm, and a sound method to make predictions.

Despite the best efforts of developers and data scientists, a current issue in machine learning is bias - biases based on **our** own limited experiences and filtered views. Specifically, racial, gender, and contextual bias. Best practices stress it is critical you strive for the least amount of bias in your model or algorithm as possible. Preprocessing data is critical.

Research has been done into scoring data set quality. There is an old standard, the DQI (Data Quality Index), which assesses the quality and reliability of data sets in real time based on deviation from predicted parameter values. The DQI reflects three aspects of your data recording: timeliness, completeness, and quality of recording. Machine learning, which provides assistance in deriving a DQI score, has a marked ability to predict trends and identify outliers if trained properly and can make suggestions or take actions on the go. Outliers shouldn't be automatically thrown out.

How is data quality measured? Start by deciding what "value" means to your firm, and then measure how long it takes to achieve that value:

1. The number of errors the model produces versus the number of data points. This includes the number of empty or null values.
2. Any slow-downs in data preprocessing time.
3. Data package bounce rates.
4. The cost of data storage.
5. The amount of time to achieve ROI.

Supervised v. Unsupervised ML

When using *supervised ML* - where the data is labeled beforehand by a data scientist to show desired output (for example, ingredients and processes for edible cakes vs. inedible cakes), the models are apt to drift from their

original intent and accuracy due to changes in data and forgotten goals (the AI makes cake, then fruitcake, then eventually switches to jams and preserves based on new training data recipes with similar ingredients). In this case, the output might be delicious but it's missing one of the required ingredients — like the bake-off sponsor's brand of flour.

The process of *unsupervised ML* makes it so the output of the AI model evolves more or less on its own. Unsupervised ML analyzes the data set and learns the relationships among the data, yet still requires a human to validate the decisions. In the case of the cake, ML might try to improve on a winning recipe to make it more likely to stand out, but it may ask you to try out the recipe before finalizing big changes.

The lifecycle of a model takes place in the *ML pipeline*, the steps needed to take the model from conception to production, as shown in Figure 1-5. This book generally discusses supervised learning, because it has a concrete checkpoint where the model is reviewed and decided upon by a *model validator*, a person who reviews models and improves them before they go to production. With additional planning, this book's concepts and procedures can be applied to unsupervised learning through the use of a *smart contract,* which is code that defines how automated business logic is executed, and through other workflow automation like routing and approvals. This process can be adapted to the various styles of ML that are too numerous to cover in this book.

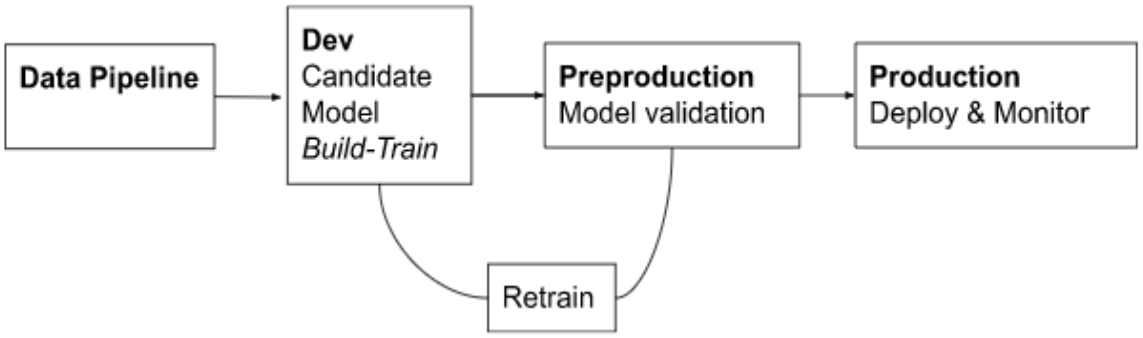


Figure 1-5. The ML pipeline

WARNING

Without standard ways to validate and track validation, supervised ML, unsupervised ML, and deep learning can easily get out of control, and your AI could bake cakes that aren't edible or don't meet your requirements. To fix this, the original intent needs to be built into the ML cycle so it does not become compromised.

Reinforcement Learning and Deep Learning

Reinforcement learning occurs when an AI is designed to gather input from its environment and then use that input to improve itself. Reinforcement learning is common in automated vehicles, as the vehicle learns not to repeat the same mistakes. *Deep learning* takes ML a step further by distributing algorithms and data among nodes in a neural network.

Program Synthesis

Program synthesis is the term used for when computers learn how to code. Program synthesis in AI is already taking place.

Since intelligent agents can write code to modify their own model based on what they have learned, their goal, and their environment, they are capable of program synthesis. They can update training data used for machine learning, which modifies their own output and behavior. With no guidelines or rules, no permanent enforcement of their main goal or subgoals, and no authority or oversight, human stakeholders do not have long-term control over the final outcome, which makes AI a very *high-risk* business activity.

Will the technology world announce one day that software will be written by AI and not humans? Unlikely. Instead, program synthesis will creep in slowly - at first showing competent human developers some computer-generated code for their perusal and acceptance, then increasingly taking more control of the coding as developers become less qualified because the AI is doing most of the code writing. Gaining mass acceptance by correcting grammar and syntax, program synthesis is becoming a real threat to our control over AI as it evolves into a way for people and intelligent

agents with no programming skills whatsoever to trigger highly complex actions on just about any system using remote APIs.

Program Synthesis: From Grammar to Gaming

What if the computer's idea about what to write is better than yours? Does your tab-key acceptance of a synthetic thought prompt the next suggestion, and the next? At what point does the power of the pen shift from you to the machine? Program synthesis, the next evolution of auto-complete, is coming to write prose, documentation, and code in ways you might not notice.

As discussed elsewhere in this book, there are few AI controls that help humans to determine the original *intent* of a piece of software, much less to stick to that intent, or alert anyone when the AI begins to go awry. The intent can be lost forever, the AI can morph, and the next thing you know you have a group of humans who don't know how the software was designed to be used. Since these AI helpers are designed to develop code or content based upon your intent, loss of original intent is a serious side effect that could have great implications over time.

- **Auto-complete:** For a number of years, computer programmers have used Integrated Development Environments (IDE), applications that recommend to the programmer how to complete complex syntax, as well as to check the code for bugs. Using these systems, *auto-complete*, the ability to accept a suggestion from the system, became a great way to learn new syntax and a huge timesaver for the software development world.
- *Smart Compose:* Google Smart Compose arose from auto-complete features originally designed for software developers. (Microsoft Word has a similar feature, called Text Predictions.) Do these features make someone a better writer, or do people become dependent on these features? When you scale this out to the millions of people who are using Google Docs Smart Compose feature (see Figure 1-6), is the group author leading the machine or is the machine leading the author?

Is there going to be a point where we realize the machine is the better writer, and we just give up? Or will our control slowly slip away as we accept an increasing number of suggestions?

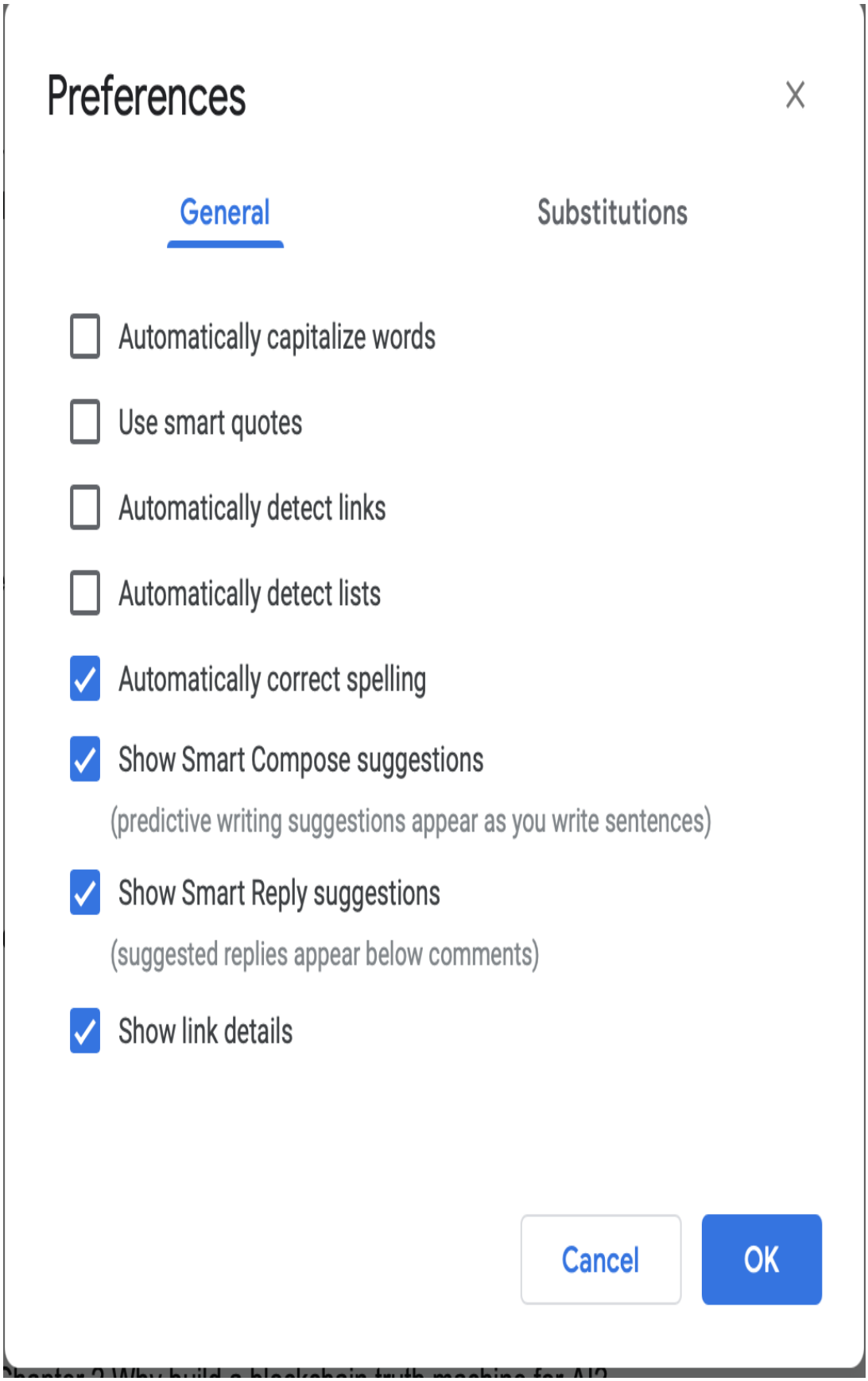

Preferences
General
Substitutions
Automatically capitalize words
Use smart quotes
Automatically detect links
Automatically detect lists
Automatically correct spelling
Show Smart Compose suggestions
(predictive writing suggestions appear as you write sentences)
Show Smart Reply suggestions
(suggested replies appear below comments)
Show link details
Cancel
OK

Figure 1-6. Settings for the Smart Compose predictive writing suggestions in a Google Doc.

According to a US National Library of Medicine publication, Exploring the Impact of Internet Use on Memory and Attention Processes, the internet may act as a "superstimulus for transactive memory." This means that you don't worry about remembering something that you can simply look up online. Does that imply that over time, everyone could become dependent on Smart Compose, and that we will eventually write how AI wants us to write?

- *Codex:* Codex is the OpenAI model based on GPT-3, which is described by OpenAI as a general purpose language model. Codex is generally available for use as an OpenAI API. In this video demonstration, developers show how Codex can take an intent described in natural language, and then produce other output based on the intent along with additional requests, and automatically write Python code to produce the results desired by the programmer. The Python code can then be executed locally or on other systems.

  Also demonstrated in the video is a connection with Microsoft Word's API. Codex is able to connect with APIs offered by productivity programs. When it works properly, programmers can give natural language instructions that turn into correct API calls. As this technology matures, it will give programmers and non-programmers the ability to do increasingly sophisticated tasks by simply stating the desired outcome, without the need or even the ability to understand any of the inner workings that make the system function.

- *Co-Pilot:* GitHub's Co-Pilot, based on Codex by OpenAI and owned by Microsoft, is making news because it extends the editors so that AI not only completes syntax for developers, but also writes whole lines of code or entire functions. Co-Pilot can write code based on the developer stating their intent in their natural language, such as instructing Co-Pilot to create a game that has a paddle and a moving ball that can be hit by the paddle. The accuracy of the output of the request and the quality of the resulting code are both a topic of much

debate, but the code should improve dramatically as Co-Pilot learns from the vast volume of code that is stored by developers on the GitHub platform.

Co-Pilot has been met by the development community with both amazement and chagrin, since it can quickly generate large volumes of working code. Some of the code meets with the community's expectations, while other parts of the code are not written to best practices and can produce faulty results.

> **NOTE**
>
> OpenAI is an AI research and development company, rather than a community-based AI open source/free project as the name might imply.

Co-Pilot is part of an attempt to create *artificial general intelligence* (AGI), which is AI that can match or surpass the capabilities of humans at any intellectual task. Companies like Microsoft and OpenAI are teaming up to accomplish AGI, and to make AGI easy for their customers to channel and deploy. Program synthesis makes this possible since the AI itself can write this complex code. By bringing AGI to the Azure platform, Microsoft and OpenAI say they hope to democratize AI by implementing safe, responsible AI on the Azure platform that is equally available to all.

- *Microsoft AI Helper for Minecraft:* Minecraft is a 3D *sandbox game,* a type of game that lets the user create new objects and situations within it. Released in November, 2011, Minecraft looks basic compared to many other graphical games, with players smashing through pixelated cubes using block-headed avatars, either alone or in teams. There is, however, a lot more going on than you would initially expect, since Minecraft players use the resources found within the infinite virtual worlds to construct other virtual items — such as crafting a shelter from a fallen tree and a stone. There are multiple modes in Minecraft, including one that allows players to fight for a certain goal, one that

allows participants to work on survival, and a creative mode that facilitates the customization of new virtual worlds without interruptions.

Minecraft allows for *modding*, which means users create *mods*, or *modifications* to the code. There are more than 100,000 user modifications to Minecraft, which can do things like add or change the behavior of virtual objects such as hardware or machinery, or repurpose Minecraft for things like exploring the human body or learning. There is no central authority for Minecraft mods, and no one is really sure what all is out there and what all they actually do.

Since Microsoft owns Co-Pilot, the next logical step is to integrate it with other programs as an *AI helper*, an intelligent agent that can accept the intent of the user and act upon it by writing what it deems to be appropriate code. In May, 2022 at its developer conference, Microsoft announced the AI helper for Minecraft: a non-player character within the game that can accept commands to construct virtual worlds, and can do so without interruption. These virtual worlds can end up as mods with no controls certifying what is actually contained in the mod, who built it, or how it will impact players.

Controlling Program Synthesis

Since most machine learning is done in a black box model, the process the AI uses for arriving at its results is not transparent at all. It isn't traceable, and for the controls that do exist such as key-value pair ML registries, they could be faked by a clever AI to meet its goals.

To fix this, expected limitations for the AI need to be built into the ML's registry, tightly coupled with the ML design and MLOps process, so as to regulate deployment of models, as discussed further in this chapter's "Defining Your Use Case" section.

It is possible for machines communicating with one another to simplify their communications by creating their own terminology, or shorthand. ...

Superintelligent Agents

Because intelligent agents are constantly improving through ML it is possible that an agent can become very good at a particular domain. Once an intelligent agent passes a human intelligence level, then it is classified as a *superintelligent agent*.

The superintelligent agent might take the form of a question-answering *oracle*, or an order-taking *genie.* Either of these might evolve into an independently acting *sovereign* that makes decisions entirely on its own. The agent's architecture could take the form of an all-knowing, all-powerful *singleton* or we could have a *multipolar* world where superintelligent machines compete with one another for dominance.

> **TIP**
>
> When planning to build a powerful AI, you can also plan its containment and demise. An agent created for a specific purpose could automatically sunset after the predetermined goal for the agent has been attained.

Superintelligent agents sound like science fiction, as if they are not a real threat but the product of an overzealous imagination. However, just because an AI has yet to surpass human intelligence does not mean that it can't. The reason we invented AI is so we can make sense of and act upon massive volumes of information - way more than what a human being is capable of doing. AI doesn't know where to draw the line - unless we code for it. If we don't, the AI could keep improving itself through program synthesis and eventually achieve technological singularity (see next section).

An example of this might be if we forget to tell our cake AI that we have a goal to operate within a limited budget. In this case, our AI could source the finest ingredients in the world, and scale up to grand production with no regard for cost, all in an attempt to win the bake-off. To fix this, we need to build approved cost limitations into our ML registry, which are aggregated when models are deployed on multiple GPUs.

Technological Singularity

Technological singularity is the hypothetical point in time at which technological growth becomes uncontrollable and irreversible, resulting in unforeseeable changes to human civilization.

In this event, you may want to already have an AI backdoor. Sometimes unethical software developers create a backdoor into a system they build, meaning that even if they are no longer authorized to get in, they can still do so. This is seen as a negative, and it is not an accepted practice to build backdoors. In the case of AI, a backdoor could be designed into the project as a way for the project stakeholders to always be able to interrupt the AI and make modifications, or completely stop it and remove all instances. Chapter 2discusses how to implement controls to make sure that human stakeholders can always intervene.

Attacks and Failures

Inside the opaque AI black box, there are a number of AI incidents that include failures and attacks brought on by bad or lazy actors. Failures include bias, data drift, and lack of verifiable identification. Attacks include adversarial data attacks, poisoning attacks, evasion attacks, model stealing, and impersonation attacks. Attacks and how to prevent and expose them are discussed further in Chapter 2.

AI requires all the same due diligence as any software development project, except it also has extra layers that need to be considered. Lazy actors or lack of involving all team members can cause critical updates or security concerns to be missed or neglected. Also, since AI can change itself based on outside input - it is possible for bad actors to attack by presenting sample data that will erode the confidence of your model.

> **TIP**
>
> When thinking about potential attacks, think about the attackers. What are they to accomplish? (Usually financial gain by tricking your AI system.) What do they already know about your systems? What can they access? To monitor for potential attacks, make sure to watch for model/data drift or other subtle changes that impact the output.

Model/Data Drift

Data drift, which underlies *model drift*, is defined as a change in the distribution of data. In the case of production ML models, this is the change between the real-time production data and a baseline data set, likely the training set, that is representative of the task the model is intended to perform. Production data can diverge or drift from the baseline data over time due to changes in the real world. Drift of the predicted values is a signal that the model needs retraining.

There are two reasons for drift to occur in production models:

1. *When there's a change in baseline or input data distribution due to external factors:* An example might be a product being sold in a new region. This may require a new model with an updated representative training set.
2. *When there are data integrity issues that require investigation:* This might involve data from a faulty front-end camera, or data unintentionally changed after collection.

One issue the AI/ML community has not figured out is what accuracy standards to put in place for system components. Establishing a kind of test-safety regime that can take every system into account and assess the systems-of-systems is difficult to execute. An example would be getting a Tesla's AI to read a sticker placed on a road sign that caused it to speed up past the speed limit.

There are different types of drift depending on the data distribution being compared, as shown in Figure 1-7.

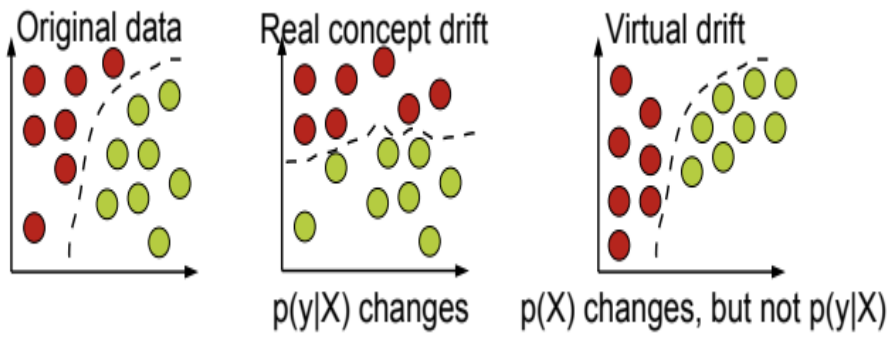


Figure 1-7. Original vs. real concept vs. virtual data drift

They all point back to a statistically significant change in the data output over time compared to the output during testing and early production, which affects the model's predictions. It necessitates re-learning the data to maintain the error-rate and accuracy of the previous regime. In the absence of real-time baseline re-establishing itself over and over, a drift in prediction and feature distributions are often indicative of important changes in the outside environment. It is possible for these quantities to drift with respect to an accurately modeled decision boundary (virtual drift). In that case, model performance will be unchanged.

Adversarial Data Attacks

Bad actors can target the modeling phase or the deployment phase to corrupt the training data or model so that the production and deployment data no longer match. They do this through *adversarial data attacks*, or adversarial ML attacks, which are hacks designed to fool data models by supplying deceptive training or testing data. This is an especially good blockchain touchpoint for ML, because it can help you to compare the original model with the current model and flag if the data is different. As a result of the flag, a smart contract can cause the model to be recalled for review, and re-initiate the review process.

The attacks can be carried out in as many ways as data can be attacked. All of the standard cybersecurity rules apply here, plus these additional attacks

that impact ML training data, production test data, and mislabeled training data.

The training data also helps the AI ascertain a percentage reflecting its own confidence in the results. However, a hacker can introduce bad adversarial data (such as what an image is *not*), making the AI draw the wrong conclusions about the query image, sometimes with great confidence. Since the hacker can introduce millions of data points and the changes are usually invisible to the human eye, these attacks are very difficult to combat.

Think about how an ML pipeline can contain billions of points of training data.

Poisoning attacks and *evasion attacks* are both types of adversarial data attacks. Both types exploit the *decision boundary*, which is the logical line by which ML places data points to classify input data into one type or another.

Poisoning attack

For example, in the case of a *poisoning* attack, which happens during the ML training phase and involves injecting malicious data that changes the classifier for that data, the added data can sway the results to be something else. Pictures of cats that look like dogs which are added to a database of dogs can sway the placement of the decision boundary, causing a certain type of cat to be recognized as a dog.

Data poisoning involves the training data used for machine learning. This attack happens during the machine learning training phase and involves injection of malicious data that changes the classifier. Typically, this happens in systems that accept input from outside sources like users and then retrains the system based on the new input. The attacker inserts a specific set of data into training data that causes a certain classifier to be consistently triggered, giving the hacker a backdoor into the system. For example, a hacker could train the system so that if a white square is inserted into an image at a certain spot, it could be identified as “cat.” If enough

images of dogs are inserted containing the white square, then any dog query image will be classified as “cat.”

Evasion attack

An *evasion attack* is the most common kind of attack and happens during the production phase. The evasion attack distorts samples (subsets of training data used for testing) so that the data is not detected as attacks when it should be. The attacker subtly pushes a data point beyond the decision boundary by changing its label. The label is changed by changing the samples so the data point tests as some other classification.

This type of attack does not affect training data, but can easily affect the outcome of your system because its own test results are wrong based on wrong answers. For instance, if all of the samples now only contain pictures of poodles, any dog that is not a poodle might no longer be classified as a dog. Because this impacts the production system, it is not uncommon for it to enter via malware or spam.

Model stealing

This type of hack is designed to re-create the model or the training data upon which it was constructed, with the intention of using it elsewhere. Then the model can be tested in a controlled environment with experiments designed to find potential exploits.

Impersonation attack

An impersonation attack imitates samples from victims, often in image recognition, malware detection, and intrusion detection. When this attack is successful, malicious data is injected that classifies original samples with different labels from their impersonated ones. This is often used to spoof identity for access control. Examples of impersonation attacks include facial recognition system impersonation, otherwise unrecognizable speech recognized by models, deep neural network (DNN) attacks on self-driving cars, and deepfake images and videos.

Risk and Liability

A big failure of AI could be that it generates risk and liability, which could be nearly impossible to assess in advance. Questions may arise when there are incidents with AI systems, such as how is responsibility assigned? How will a legal authority or court approach these issues when it is not obvious who is responsible due to the black box characteristics of AI?

There is discussion among attorneys and the AI community about how much documentation is too much. Some advisors give the impression that you shouldn't keep a complete trail of information about what you are doing, in case you do something wrong, because if the information doesn't exist, it can't be used against you. Other advisors will say that if you should have known some potential risk and did not take adequate steps to prevent it, that is a risk in itself.

> **NOTE**
>
> Risk and liability assessment is beyond the scope of this book, and as with any important decision, you should discuss these topics with your own stakeholders and legal advisors while you are in the planning process. For general information, a good reference for this subject is "AI and the Legal Industry" (Chapter 2) in Karen Kilroy's book *AI and the Law* (O'Reilly, 2021).

Blockchain as an AI Tether

AI aggregates information and learns from it, morphing its own behavior and influencing its own environment, both with and without the approval of data scientists. Blockchain can be used to permanently track the steps leading up to the change in output, becoming a *tether*, or an audit trail for the model. Think back to the critical facts that may be on your factsheet, as listed earlier. Your AI's purpose, intended domain, training data, models and algorithms, inputs and outputs, performance metrics, bias, optimal and poor performance conditions, explanation, contacts, and other facts are all potential blockchain *touchpoints* – points in the fact flow that can benefit

from a tamper-evident, distributed provenance. This gives you a way to audit your AI in a human-readable form, trace it backward, and be sure that the information you are reviewing has not undergone tampering.

> **TIP**
>
> You can think of AI's development, training and deployment processes as being sort of like a supply chain. By comparison, the number one use for enterprise blockchain is track-and-trace supply chain, which involves the lifecycle of goods and their journey to the consumer. In the case of a product like honey, each step along the way is documented, each process is automated, and important touchpoints are recorded to blockchain as a permanent audit trail, which can later be checked to detect fraud like diluting expensive honey. In this case, a cryptographic hash showing the DNA structure of the honey is used as a crypto anchor, and is compared to a real product by a consumer to make sure the product is authentic. Similarly, a fingerprint of data that is used to train AI can be recorded on blockchain, and stay with the model for its entire lifecycle as each engineer and data scientist performs their specific tasks like training and approving the model.

In order to effectively record the touchpoints onto blockchain, you need a blockchain platform. We chose Hyperledger Fabric, an enterprise-level consortium-driven blockchain platform. For more information on features of Hyperledger Fabric and blockchain-as-a-service derivatives, see our Blockchain-as-a-Service report (Karen Kilroy, O'Reilly, 2019).

From an architecture standpoint, you don't have to change much about how your modeling workflow goes now. Blockchain gets added to your AI technology stack and runs alongside your existing systems, and can be set up in a way where it is low-latency and highly available. Unlike blockchain used for cryptocurrency, performance is not an issue.

DEFINING BLOCKCHAIN TOUCHPOINTS: DATA AND MODEL DRIFT

The blockchain touchpoints for keeping track of data/model drift are shown below. Data and model drift monitoring helps ML teams stay ahead of performance issues in production. Being able to track and trace data and model drifting is an essential part of a multidisciplinary strategy of risk management in an organization. Here is a list of potential blockchain touchpoints that will help you to track and trace data and model drift:

Touchpoint	Layer	Description
Inputs and outputs, test level	Model validation	This would be broken down into test baseline, test results at different iterations until the model levels out and stays within a predetermined range (out to the 6th standard deviation per ISO9001 2015, or standards set by the NHSTA or NIST ODD, for example).
Inputs and outputs, initial production level (production prime)	Model validation	This would be broken down into production baseline input data, and each output would be separately measured against the previous input and the baseline input for deviation. At a statistically significant number of iterations, the model would be flagged to see what variables needed to be altered, or if the model needed to be.
Dependencies and requirements Vulnerabilities and patches	MLOps, ML pipeline	Similar to devOps, MLOps involves updates to the model, content, and training process. MLOps includes management of ML training cycles that involve various ML experiments and content. To keep track of drift, which can be elusive, this touchpoint would be critical.

Also see the list of general-purpose AI blockchain touchpoints later in this chapter.

Enterprise Blockchain

Speaking of performance, it is a common misconception that all blockchain requires miners, who race to solve a cryptographic puzzle in exchange for a tokenized reward. Enterprise blockchain platforms, such as Hyperledger Fabric's blockchain-as-a-service variants - primarily IBM Blockchain, Oracle Blockchain Platform, and AWS Blockchain –. can offer superior deployment and upgrades, time-saving development environments and tools, better security, and other features not offered in Hyperledger Fabric alone.

These blockchain platforms do not require miners, and generally run inside *containers*, which are portable, lightweight, self-sufficient computing stacks provisioned by systems like Docker or Kubernetes, running on high-performance, highly-scalable cloud-based or on-premises systems. In an ML workflow, for instance, a new block might be added to the blockchain after it met some predetermined criteria such as approval of a new model by a model validation team.

Tokens, which make up the coins used in cryptocurrency, are generally missing from an enterprise blockchain implementation. Although tokens can be implemented to represent the full or fractional value of some other thing, they are an optional step when implementing enterprise blockchain. Tokens can always be added later as experience and acceptance of the blockchain network are gained. Could model validators be paid by blockchain tokens? Sure, and this could be made part of your fact flow. Model validators could be automatically rewarded when their obligations are met and tested via smart contracts.

Also note that enterprise blockchain is typically *permissioned*, and that public blockchain platforms like ethereum are *permissionless*. This means enterprise blockchain has organizational identity management and access control, as well as integration with existing enterprise directories like LDAP. This book dives into identity in Chapter 2, but first, let's explore the basics of how blockchain works.

Distributed, Linked Blocks

Blockchain is a type of *distributed ledger*, and runs in a peer-to-peer network. Each network endpoint is called a *node*. Nodes can be used by individual users, but generally they are shared among groups of users from an organization or group of organizations with common interest. Generally with enterprise blockchain, the nodes are stored in containers on cloud provider computing accounts, such as Oracle Cloud, IBM Cloud, AWS, or Microsoft Azure. There is usually an abstraction layer that is the user-facing application, and blockchain is transparent to the users. The application layer is where workflow takes place.

In many cases, blockchain nodes can be distributed to each participating organization in a blockchain network. This gives their participants a copy of the blockchain over which the organization has full control. Keep in mind that you don't have to distribute your nodes from the beginning. Instead, you can add nodes later as more organizations join your blockchain network.

You might start recording the provenance of your AI project from inception, which is the first set of instructions from the original stakeholders. The very first set of instructions might say who is responsible and grant permission for them to perform various tasks such as creating or approving a request in the fact flow. This is the beginning of your AI's *governance*, or the accepted set of rules and procedures that is central to maintaining operation of the system.

Later, you might, for example, take on a development partner who wants to participate in governance and would like to have full confidence in the factsheet. If they are not savvy in blockchain, you can give them a login to your fact flow system and let them learn about blockchain by seeing blockchain proofs embedded into the application layer. Once they become more sophisticated with blockchain, they may want their own node so they are more certain that all of the facts are intact.

Blockchain networks foster group confidence because the foundation is a series of blocks of related transactional data. Each block is time stamped

and permanently linked to one another, and distributed and validated via a peer-to-peer network, as follows:

1. *Blocks are distributed:* Blockchain does not use a centralized server like most business applications, but instead the data and code are distributed via a peer-to-peer network. In this network, each participating organization has its own copy of the blockchain system running on its own peer, which is known as a *node*. The peers communicate directly once they are established.

   This makes blockchain very attractive as a ledger for business-to-business workflow, or for breaking down barriers between departmental silos within single organizations. Since each organization runs a copy of the system and the data, it helps to establish trust. If a new record is inserted into a block of one node then the same copy of data also gets created in other stakeholders' nodes based on assigned permission. In this way It is very difficult to tamper over the data because the original copy of the data would have existed in the rest of the nodes as well.

2. *Participant requests are validated:* Before transactions are validated and placed into a block by one of the participating people or systems in an organization, the request is first tested on their node against a smart contract, which contains predetermined business logic tests. If these tests are successful on the requesting participant's node, then the request is broadcast to all nodes in the network, and tested on each node by the smart contract and other algorithms that validate the request. Each participant is given with certain permission and access to the resources of the blockchain network like assets. So participants can perform only those tasks which are mentioned in his endorsement policy of the blockchain network (permissioned blockchain).

3. *Blocks are formed and linked:* Blockchain gets its name from its method of forming blocks from a list of transactions and then linking (or "chaining") the blocks to each other through unique fixed-length strings of characters called *cryptographic hashes*. This is done by

computing a hash based on a block's contents, then storing a copy of the previous block's hash, along with its own hash, in the header of each block.

The hashes reflect the content of the transactions, so if a block's data gets manually changed in the filesystem by tampering or corruption, the hash of the block will no longer match the hash recorded in the next block. Thus any modification to a block can be easily detected. This makes blockchain well suited for creating permanent records and audit trails.

If a hacker changes a block's contents manually or tries to insert a block midchain, it will impact not only that block, but every block after it in the chain. This will not impact only the particular node where the hash got changed but also other nodes of the entire blockchain network which makes it easier to find the fraudulent content.

4. *The block's contents are verified:* If the request to add a new block passes all of a node's checks, then the block is added to that node's copy of the blockchain. If the checks are not passed, the new block will be rejected. If your system is programmed in this way, it will then flag the denied request for potential security issues.

These steps are illustrated in Figure 1-8.

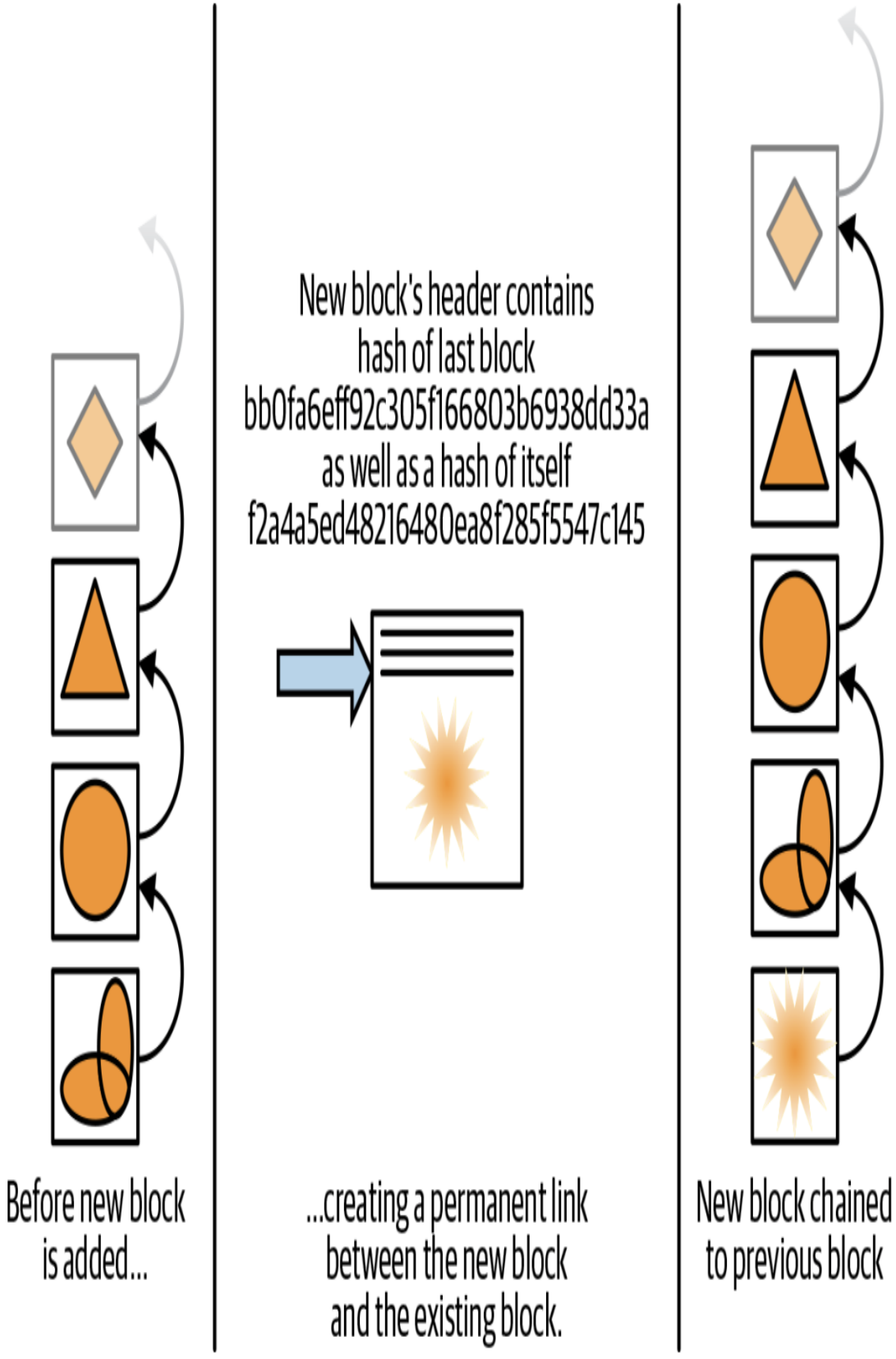

New block's header contains
hash of last block
bb0fa6eff92c305f166803b6938dd33a
as well as a hash of itself
f2a4a5ed48216480ea8f285f5547c145
Before new block is added...
...creating a permanent link between the new block and the existing block.
New block chained to previous block

Figure 1-8. Blockchain chains blocks of text, and if any of the contents are changed, the chain breaks and the hashes no longer compute to the values stored in the blockchain.

As seen in Figure 1-9, blockchain provides each *stakeholder* with their own *nodes*, or copies, of the blockchain so they can be relatively sure the data remains unchanged and that the chain is not broken.

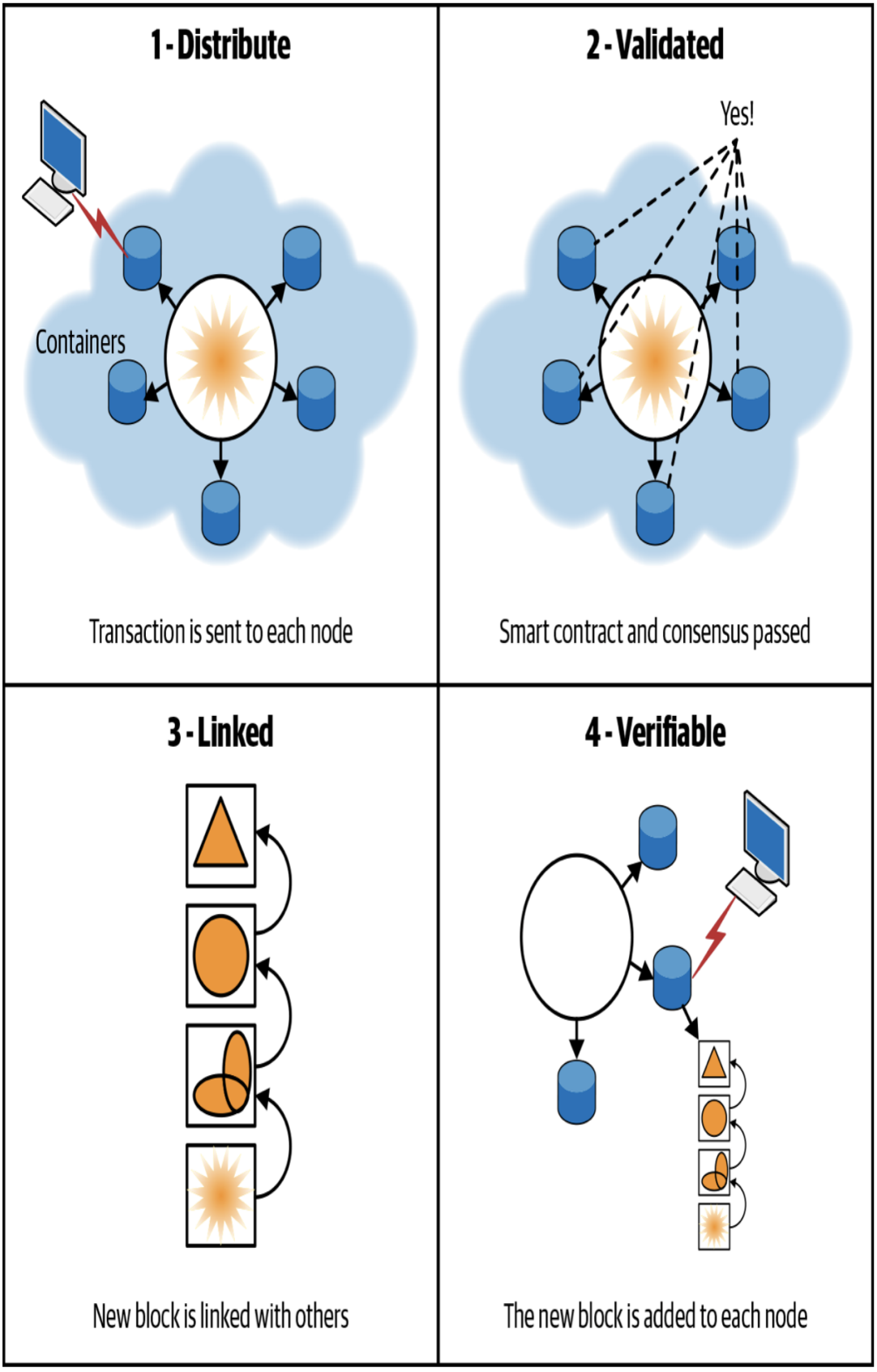

1 - Distribute
Containers
Transaction is sent to each node
2 - Validated
Yes!
Smart contract and consensus passed
3 - Linked
New block is linked with others
4 - Verifiable
The new block is added to each node

Figure 1-9. New blocks are distributed, validated, linked and added to each node.

Like any data contained in any file system, blockchain can be changed by brute-force hacks or even file system corruptions. An individual node could be manipulated by an overzealous AI that wanted to erase its own mistakes, for instance, or by any bad actor that wanted to change history. However, this would be evident to the stakeholders, because the node would no longer work. This is because each block is linked based on content, and in order for an attack to go undetected, all nodes would have to be completely recomputed and redistributed without any stakeholders noticing. This also wouldn't be technically possible without some pretty sophisticated hacking. So the best way to think of it is that blockchain is tamper-evident, as it will show any bad activity that happens by going around the application layer.

Trust and Transparency

AI systems are generally not trusted because their form and function are a mystery. Blockchain can bring a single source of truth to AI. The entire lifecycle of AI can be made transparent and traceable to engineers and consumers by adding blockchain to the stack and integrating the workflow of the AI lifecycle with blockchain at critical *touchpoints*, or points of data that make sense to verify later, like items on a factsheet. Recording AI's history in this way will create a tamper-evident, distributed audit trail that can be used as proof of the AI lifecycle, as shown in Figure 1-10.

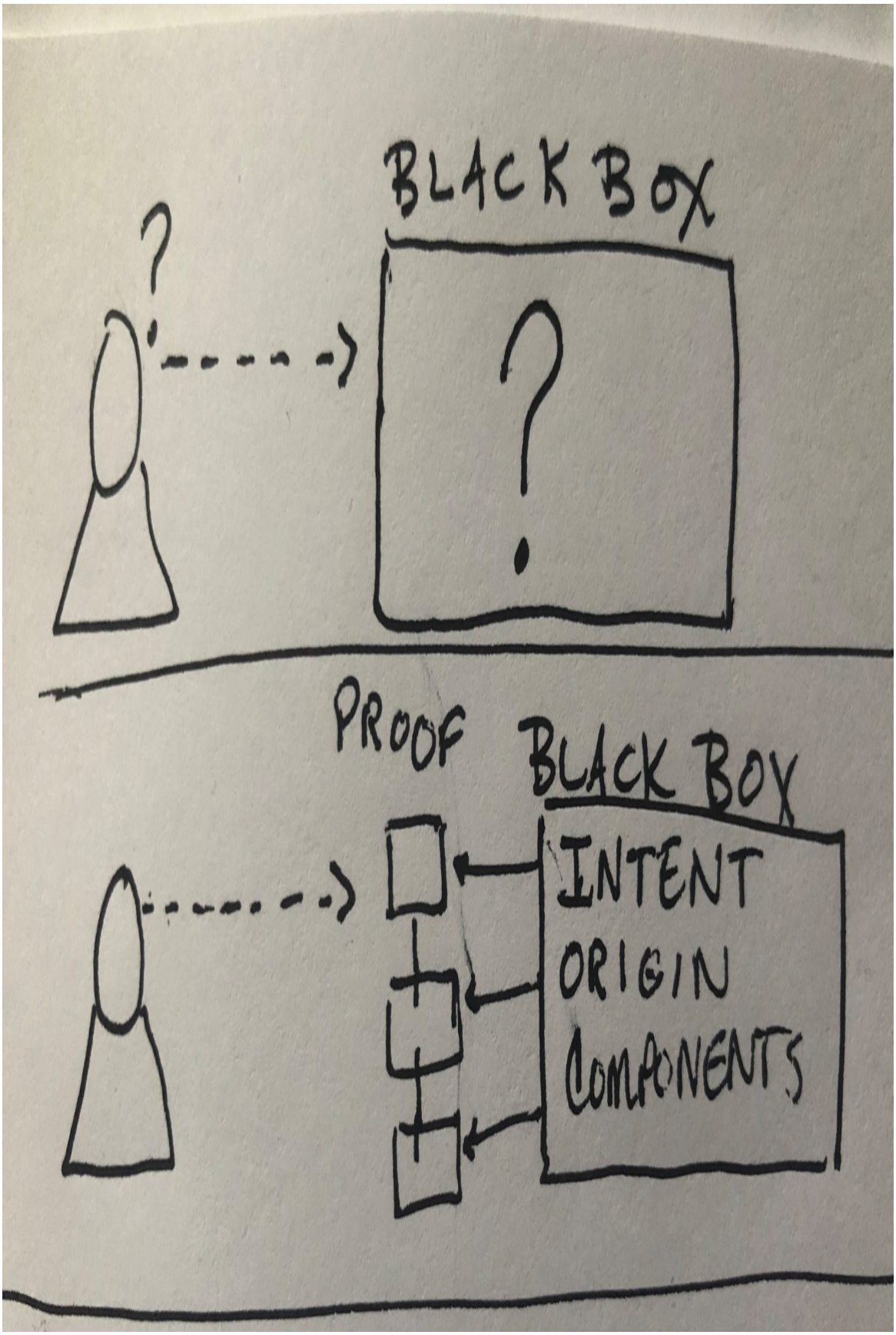
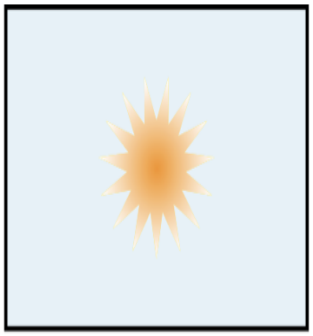

BLACK BOX
?
?
PROOF
BLACK BOX
INTENT
ORIGIN
COMPONENTS

Figure 1-10. Blockchain can be used to take the mystery away from the black box that is AI.

Using a permissioned blockchain like Hyperledger Fabric, it is possible to design AI to be transparent and traceable without sharing information that should be kept private. Hyperledger Fabric is organized into channels, which can be used to grant permission to only certain parties. This permission granting system is explained in detail later in this chapter. It is also possible to verify that some information is the same as what was stored in an *off-chain* database (not on blockchain such as NoSQL) and that the computed hash of the information still matches the hash that is stored on the blockchain.

Figure 1-11 shows how an original block of data contains a hash reflecting the contents of that block. If even the slightest change is made to the data stored in the block being inspected, recomputing the algorithm used to create the hash will result in a different hash. If the hashes no longer match, the block should be flagged and the blockchain network will have an error.



NOTE: A hash cannot be reversed to create the original contents. It can only be used to verify that the current content is the same as when the hash was originally generated.

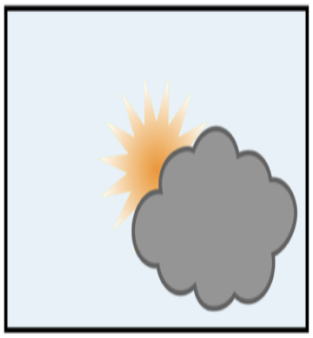
The block we are examining contains an image of sunshine in a clear sky.

The header contains the following hash of the previous block's contents:

F0B2F0F2096745F1F5184F631F2BC60292F64E76AB7040BE60BC97EB0BB73D64

And this hash, for the current block's contents:

4BCD77921211F4CF15D8C3573202668543FA32B6CFAD65999E3830F356C344D2



Later, the same block is recomputed in order to be tested for tampering or corruption. In this example, the block's contents have been tampered with and changed to an image of a sunshine in a cloudy sky.

Header contains hash of previous block's contents:

F0B2F0F2096745F1F5184F631F2BC60292F64E76AB7040BE60BC97EB0BB73D64

But the current block's hash has changed, because its contents have changed:

191166F725DCAE808B9C750C35340E790ABC568BE1214AB019FB5BB61EE6A422

Since blocks are permanently linked by storing the previous block's hash in a new block's header, any file system-level changes that are made will be evident. Tampering with content breaks the chain of hashes by causing the next block's header to not match.

Figure 1-11. Cryptographic hashes stored in blockchain are used to prove there has been no tampering of the contents.

Workflow of some sort typically drives the contents of the blockchain. Figure 1-12 shows a high-level workflow for an AI project where origin, intent, and components of the AI system need to be proven.

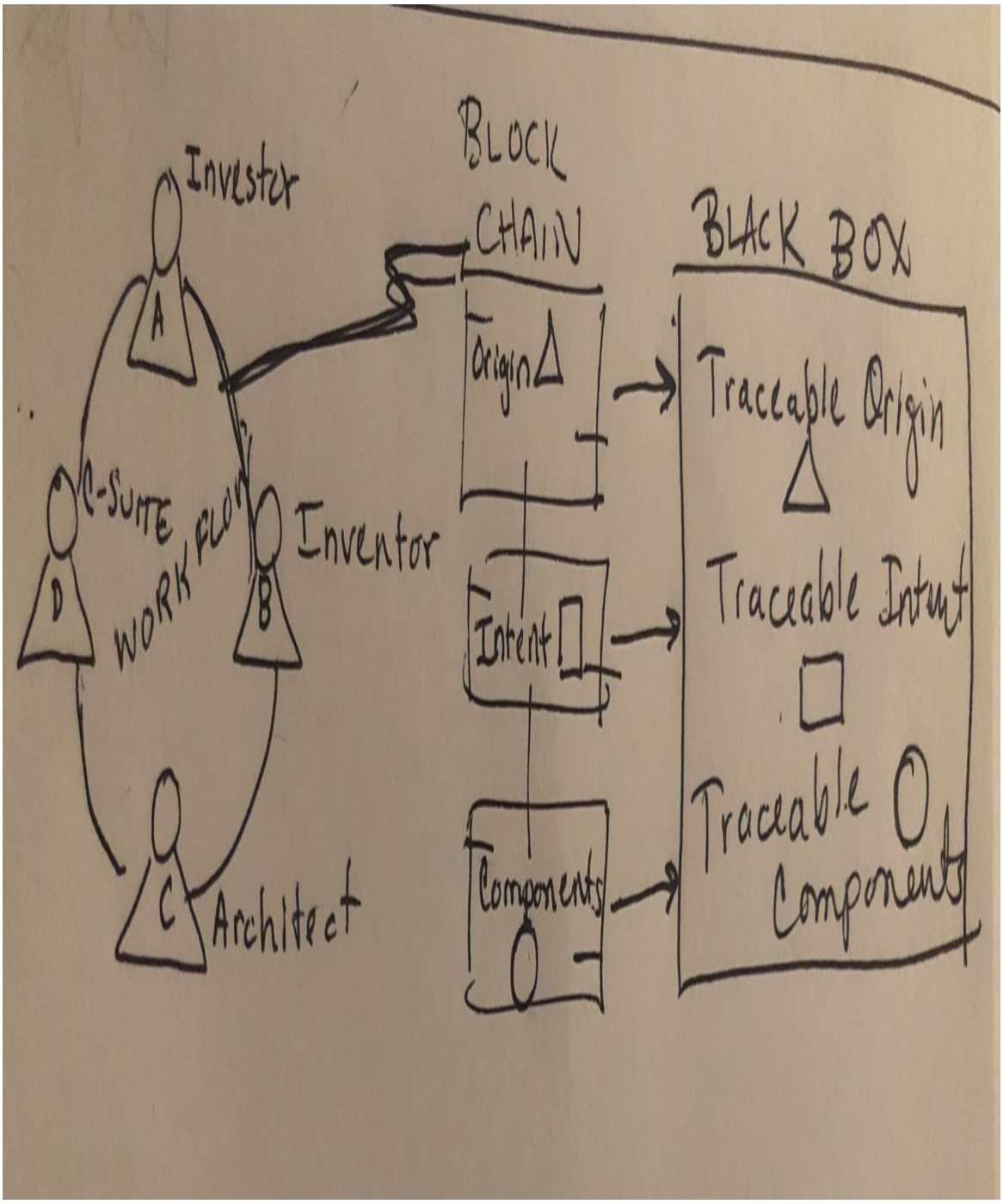


Figure 1-12. Sample stakeholder workflow. In this case, the stakeholders are establishing the origin, intent, and components of the AI project and recording it on blockchain so it is traceable.

In most cases, the information compiled for the blockchain touchpoints would be too much for an average consumer to readily absorb. So in order to be understood, the proofs need to roll into trust logos that break when the

components can’t be trusted according to the trust organization’s standards, as shown in Figure 1-13.

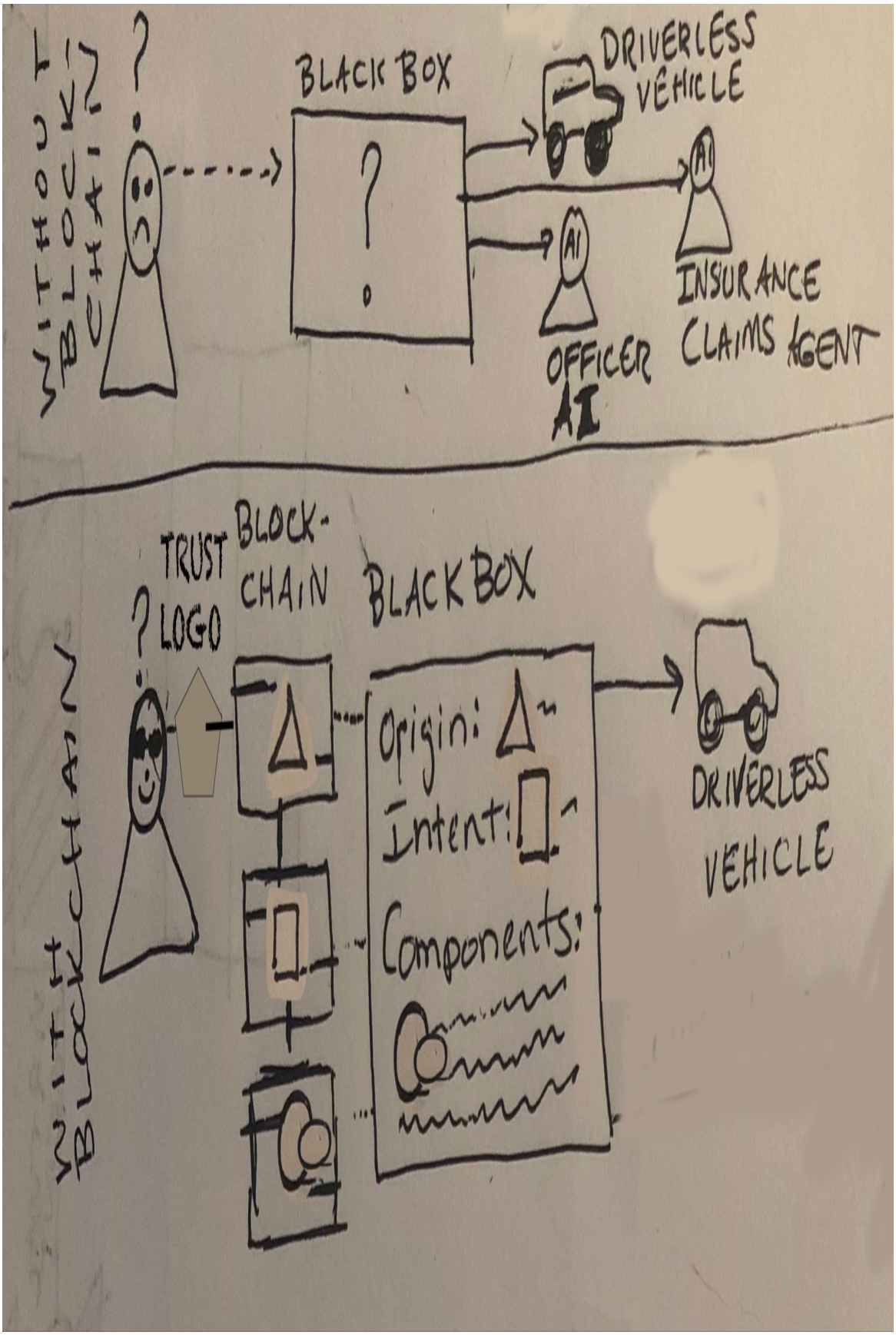

WITHOUT BLOCKCHAIN
?
BLACK BOX
?
DRIVERLESS VEHICLE
AI
INSURANCE CLAIMS AGENT
AI
OFFICER AI
WITH BLOCKCHAIN
?
TRUST LOGO
BLOCK-CHAIN
BLACK BOX
Origin:
Intent:
Components:
DRIVERLESS VEHICLE

Figure 1-13. The top half of this figure shows an AI implementation with no blockchain. The bottom half of this figure shows an AI implementation where a consumer can use a trust logo backed by blockchain to prove the trustworthiness of the AI.

To understand how this proof might work, imagine you are having a bad day, brought to you by AI. You're driving to the office and the car in front of you suddenly brakes[1]. You slam on your brakes, skid, and crash into the back of the other vehicle. You put your car into park and jump out. As you approach the vehicle you hit, you see that no one was driving. Puzzled, you dial 911 to explain this to the dispatcher, and you reach Officer AI.[2] Due to staffing shortages, Officer AI tells you it can only send a human officer to the scene if there are two or more drivers involved. So you call your insurance claims hotline to ask what to do. After explaining the situation several times, Claims Agent AI[3] still doesn't quite know what to make of your situation, and you are placed on hold until a human can take your call.

Do you have the right to know right away that you are talking to an intelligent agent instead of a human being? Can you question someone's humanity in a standard and non-offensive way? If you question an intelligent agent, can you always escalate to a human being who has the authority to help you, or could you be held to the decisions made by the intelligent agent? Who's fault is the accident in the first place - how will the court assign blame?

Even the people who originally set up the system often don't know what is in there, if enough time has passed and they have forgotten, and others have done experiments to train the model. Changes could have been made for performance tuning that impact the accuracy, and this would not be known to the originators of the AI. An engineer might use a blockchain audit trail to quickly find the origin of AI, the intent, and what components it is made of and where they were sourced. The blockchain could also reflect things like last time the system was maintained, whether it has been rated as reliable, and whether or not any components have been recalled due to safety issues.

Defining Your Use Case

Analyzing your use case is the first step in determining whether or not a certain technology is the right solution for your business. A great way to begin this process for blockchain applications is to determine the participants, assets, and transactions for the case. In Figure 1-14, you can see these three basic elements and how they interrelate.

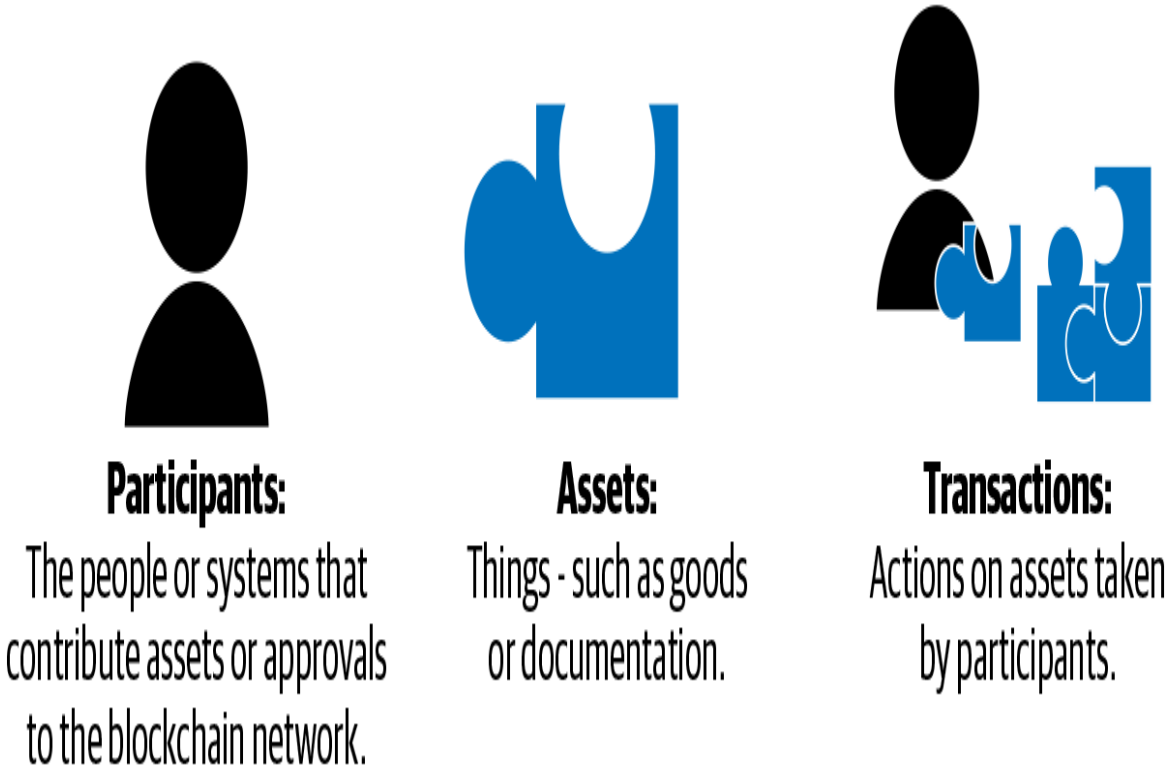


Figure 1-14. Identifying participants, assets, and transactions for your use case is a first step in planning your blockchain network.

> **TIP**
>
> Information about participants, assets, and transactions is very helpful when providing project requirements to a developer, along with the business logic that will drive your smart contracts.

Touchpoints

As this chapter touched upon earlier, there are many potential blockchain touchpoints that, when recorded on blockchain, will later help prove the provenance of the AI. Table 1-1 shows a list of potential touchpoints on AI projects.

Table 1-1. Blockchain touchpoints, or places where blockchain/AI integration makes sense, can be analyzed and identified. Not everything in an AI project has to be stored on blockchain, so identifying touchpoints is a good place to start.

Touchpoint	Functionality	Description
Purpose and Intended Use	Stakeholder workflow	Decisions that implementers and funders made about requirements for AI.
Contacts and Identity	application and blockchain	Identity and roles and permissions for all levels of maintenance and use, including contact information.
Inputs and outputs	Model validation	What input does the AI expect, and what are its anticipated outputs?
Optimal Conditions and Poor Conditions	ML model	A model is optimized for certain conditions and is often known to fail in others.
Application layer	application	The working layer of the system, often web-based and user-facing. Controls things like common features, look and feel, and logins. Includes code and databases.
Blockchain layer	blockchain	The distributed ledger that generally resides on nodes running in containers. Within the containers, blockchain resides on the file system.
Security	security	Firewalls, demilitarized zones, etc. that prevent unauthorized users from gaining access to the other layers.
Fact flow	ML registry or fact flow system	Currently a database, the ML model registry already stores critical information about training methodology, training data, performance and proper usage.
Training data	ML content	Structured and unstructured data, cleaned up by data scientists and used to train models. In a federated model, this data resides on nodes.
Models and algorithms	ML model	The file that generates predictions based on the ML content. Includes model registration, deployment details, measurements on data drift, and training events.
Dependencies and requirements	MLOps, ML pipeline	Similar to devops, MLOps involves updates to the model, content, and training process. MLOps

Touchpoint	Functionality	Description
Vulnerabilities and patches		includes management of ML training cycles that involve various ML experiments and content.
Explanation	ML experiments	Records created by ML experimenters, containing approvals and reasoning for tweaking ML variables that could, for example, improve performance and lower cost, but might impact accuracy.
Feedback and Model Experience	intelligent agent	What an intelligent agent does when it is released to production, including feedback from consumers.
Trust logos and requirements	consumer touchpoint	The part of the intelligent agent that is exposed to the consumer.

Participants

Participants are the people and systems generating transactions within the blockchain network. For example, in a produce supply-chain network the participants might be defined as the

- Farmer
- Distributor
- Warehouse manager
- Logistics company representative
- Long-distance trucking company dispatcher
- Automated equipment inside the truck
- Truck driver
- Local delivery truck systems and driver
- Store’s dock manager
- Produce department manager

From the standpoint of a model validator, each one of these participants would have an assigned role and permissions to log in to approve or reject the smart contract as its passed from point to point, and the model validator

would already have blockchain incorporated at each point to check for anomalies. Here are some things that they would look at:

1. *Relevancy of the data:* There may be unusual fluctuations in the data entered at each point, be it manually or electronically entered into the smart contract. Any anomaly would kick off a check-off process to compare each data point to previously acquired data and calculate the standard deviation from the usual averages.
2. *Validation of the data's integrity and appropriateness:* Validate so that the data may be utilized for the intended purpose and in the proper manner. Look at time periods, sources, and missing value computation.
3. Handling of the data - were the collection methods changed from the agreed upon procedure?
4. *Pre-processing:* Were any of the automatic collection methods compromised? Were there any normalization, transformation or missing value computations that were not performed?

Some helpful tethers you can include on your factsheet and blockchain touchpoints as a model validator include the following:

- Stakeholders in the governance group are given access based on roles and permission that are determined by organization policies, because the risk is high.
- Access control is available to system administrators and management.
- Business policies behind the workflow are consented and signed off by the governance group, including the expected life cycle of the model.
- There is availability of output data for reporting .
- Training data and baseline outputs are cataloged after approval.
- Keep track of the tradeoff between accuracy and explainability.
- There is a procedure for feedback from consumers or authorities.

- There is a procedure to roll back the model.
- Responsible parties are identified and contact information is available.

You can include as many participants as needed. Since in business it is not always practical for each participant to have their own node in a blockchain network, the participants may be grouped into logical units called organizations. Multiple participants often share an organization's node. In most business applications, the participants won't even know they are using blockchain.

Assets

An *asset* is a type of good that has some value, and its activity can be recorded on a blockchain. If we built a blockchain application to track vehicle ownership, a car would be an asset. If instead our blockchain tracked individual auto parts, each of those parts would be considered an asset.

In a cryptocurrency application, a digital token used as currency is an asset. A participant can own the currency, see it inside a digital wallet, and transfer the currency to a different participant. By contrast, in a business blockchain application tangible goods and corresponding documentation are represented as assets, and there is likely no cryptocurrency involved at all.

In our produce supply-chain example, the primary assets are units of produce. When a unit of produce is delivered to the store's loading dock, the asset (produce) is transferred from the truck to the store. This then triggers payment from the warehouse. All of the other business interactions are also programmed to be tracked.

Other items involved in an exchange, like import/export certificates and money, may also be tracked as assets.

From the standpoint of a model validator, the payment-triggering process, being tracked, would have algorithms to compare the asset volume versus the price, and make sure that the money collected was consistent with the volume, and the added in taxes and fees. Further:

1. Validators should examine the monitoring strategy to ensure that scope, objectives, stakeholders, and roles and duties are all addressed, and guarantee the model delivers expected monetary output and is stable over long periods of time. The frequency and duration of scheduled recalibrations should be assessed. Management should guarantee that everyone in the governance group is aware of all model hazards.
2. The black box nature of the models mean that ML approaches to validation are not widely accepted due to lack of being able to quantify transparency and explainability, and how the model fits the environment at hand. There are model-agnostic software packages that a validator can use to do this.
3. The key with financial modeling is to be able to use anonymous data to train the model and share the results with the members of the governing group that are involved with the financial part of the supply chain.
4. Once a new model is approved in the test environment and its security is shored up, then the validators must also assess whether the models, including catalogs, modules, and settings, are suitable for deployment, taking into account the potential consequences of future releases.

Some helpful tethers you can include on your factsheet and blockchain touchpoints as a model validator include the following:

- Stakeholders in governance group that have permission to access this workflow because the risk is high.
- Training data has been approved and is cataloged.
- Data shared outside the governing organization is anonymized.
- Steps are taken to eliminate bias in the model.
- Predictive performance of the model has been approved (the model is stable).

- Business policies exist behind the asset workflow.
- Blockchain controls:
    - Workflow and identities of participants
    - Governance of policies
    - Restrict/prohibit unwanted operations
- Keep track of parameter dependencies, and how they have been taken into account.
- Quality metrics versus the training data are approved and cataloged
- The list of responsible parties and their contact information.

Transactions

Transactions are generated when participants have some impact on assets.

A workflow generates a transaction which is evaluated by a smart contract. Then the new block, containing one or more transactions, is added to the blockchain network. In the case of an AI model validator, we might record a transaction when any of the following occurs:

- Record of transaction along each point on the supply chain's approved workflow.
- Produce weight is recorded leaving or arriving at any designated spot.
- Produce in a climate controlled environment in transit is tracked, if IoT sensors are triggered.
- Real time record of all transactions.
- Record of payments to farm and to drivers.
- Real-time self-audit:
    - Error protocol initiated if transaction data is outside of reasonable delta

- Error protocol if payment does not occur as expected
- Compare model output to real-time output.

Smart Contracts and Business Logic

Smart contracts are another useful feature of blockchain, as they allow agreements to be pre-programmed, so the proper workflow must be achieved before certain events take place (for example, an invoice must be paid before a product is shipped). Blockchain offers *zero-knowledge proofs*, which allow a party to prove they know certain information without actually disclosing it. Blockchain, by its nature, helps enforce rules and share information that benefits the greater good of the community using it.

Think of smart contracts in terms of how you want your participants to be able to handle assets, and what constitutes a transaction.

For instance, let's say a model validator is chatting with a QA tester who is upset about a poor output of a model. The model validator opens their dashboard and sees the factsheet of the model. The model validator clicks a button that says Trace Model, and it is revealed that bias was indeed flagged in the training data by the data science team. In this smart contract, it is stated that the data science team has to approve moving forward with data sets with known bias, else the model is not allowed to move forward through the ML pipeline. The model validator also sees that this step was skipped and that the data scientists did not give this approval. The smart contract states that the default for "no data scientist approval" is to return the model to the team for fresh training data before the model proceeds to production, so it should have never made it to QA.

Because of the smart contract and the transactions are recorded on the blockchain,the model validator can quickly pinpoint why the model is perceived to be inaccurate and provide verification to the stakeholders. Without a system like this, the model validator could spend a lot of time trying to track down the reason for the inaccurate output.

When agreements are automated with smart contracts, it causes systems to apply agreements in a fair, unbiased, and consistent way, which reduces risk

for transactions such as those involved in validating a model. MLOps should improve as a result, since paper contracts and guidelines often sit in a file drawer unenforced, while the individuals running the day-to-day operations set the actual procedures. Smart contracts make all parties more aware of, and accountable to, their formal business agreements, even when they are highly complex.

Audit Trail

AI aggregates information and learns from it, morphing its own behavior and influencing its own environment. Blockchain can be used to permanently track the steps leading up to the change in output, becoming a memory bank for the model.

In Trustworthy Machine Learning (Kush. R. Varshney, self-published, 2022), the author suggested that AI factsheets be implemented on blockchain, to provide a distributed provenance that shows tampering; his book stops short of implementation. You may discover that the possibilities are nearly limitless since the fact flow system can gather input or approvals from any number of people, systems or devices, and weigh the input or approvals against business logic contained in *smart contracts*, to dynamically produce a current factsheet upon request.[4]

Blockchain can influence the integrity of intelligent agents in the same way it helps groups of people who don't necessarily trust one another to be able to conduct business in a transparent and traceable way. Blockchain stores information in units called blocks, and, using cryptography, generates a hash to logically link each block to the previous block.

Local Memory Bank

Without any memory bank full of facts and experiences, there is no standard way for AI to recall why it has become the way it is. This would be like every event in your life changing you, yet you could not remember any of the events specifically or why they influenced you. Instead of figuring out

what is best based on what went wrong, you instead are forced to keep trying experiments until you hit the right combination again.

A blockchain audit trail for AI is similar to a human memory in that it can help to recreate what took place, so you are better prepared to try to reverse the undesired result - which could be due to an incident such as bias, drift, or an attack, or machine failure or human error.

By deploying the trained AI model in the same computing instance as its corresponding blockchain, you can give the model a local blockchain node - a low-latency, highly-available, tamper-evident single source of truth in which to store facts - which can work even when the model is offline and can't reach the blockchain network.

Shared Memory Bank

Once the blockchain node for the model goes online it connects to the blockchain network via the API, or REST proxy. Then it can broadcast its transactions to other nodes. This is because the REST proxy acts as a transaction manager, maintaining a state machine to keep track of the execution of a transaction. Crashes or errors happening during the execution of the transaction will be dealt with when REST proxy reconnects.

You can build smart contracts that govern the function of the model based on its shared or local memories of what it is supposed to do combined with current environmental variables. As an example, you could program in something that said if the model misbehaves X number of times, then it has to take itself out of service until a new training and approval happens.

Four Controls

The AI trust challenges and blockchain touchpoints can be broken down into four types of blockchain controls.

- *Control 1 – make data and models/algorithms distributed and tamper evident:* This control can be used with AI to verify that data and models have not undergone tampering or corruption.

- *Control 2 – pre-establish workflow and identity criteria for people and systems:* This control can be used with AI to make sure that the right people, systems, or intelligent agents - with the right authorization - are the only ones that participate in governance of or modification to the AI.
- *Control 3 – govern, instruct, and inhibit intelligent agents:* This will become very important when wanting to trace or reverse AI, or prove in court that the output of AI is traceable to certain people or organizations.
- *Control 4 – provide proof of authenticity through user-viewable provenance:* This will be especially important in using branded AI that has underlying components which come from distributed marketplaces.

Chapter 2 takes a deep dive into the four categories of blockchain controls and how they are applied.

Case Study: Oracle AIoT and Blockchain

Blockchain is a newcomer to the AI space. Bill Wimsatt, senior director of Oracle Cloud Engineering, talked with us about how his company is using blockchain in the AI stack. Oracle, a pioneer in Internet of Things (IoT) technology, combines IoT with AI to create AIoT. This is to better serve some Oracle customers, who are working with expensive equipment like gigantic transformers, and it is helpful for them to see a roster of all of their key equipment along with the output of a statistical model that predicts when the parts might die or blow up. Wimsatt and the team at Oracle have collaborated with these customers to build models to predict when they should be doing service. For example, maybe early maintenance is needed, or maybe the asset can last longer. In its analysis, AIoT also considers any missed or false alarms, and can not only prevent equipment failures due to lack of maintenance, but also optimize work and service orders.

Wimsatt said, “We use AIoT for predictive maintenance and signal capture; by combining AI with the IoT devices we can learn what is happening with equipment, and find different ways to use signals. This helps our customers to not only make sure the asset doesn’t fail, but that they can *sweat the asset* for as long as possible.”

“Sweat the asset” is a term to indicate that the owner of an asset has squeezed the longest possible lifecycle out of it before replacement. Typically, these IoT assets are on maintenance and replacement schedules, and these things occur whether the equipment needs it or not. This can lead to millions of dollars wasted on unnecessary maintenance. AIoT can help to make better predictions on how long each individual asset may last based on its metrics, which saves large amounts of money because assets last longer than scheduled.

Blockchain has been added to Oracle’s AIoT stack to validate signal capture at the point of transmission. Then their customers can cross-check the data that is captured into an object store against the blockchain validation. Because a sample of their data is distributed and stored in interlocking blocks, they can test to be sure that the training data matches what was captured by the IoT device, and that it has not undergone tampering. This technique aligns with Control 2, “make data and algorithms distributed and tamper-evident.” Control 2, along with the rest of the four controls, is discussed in depth in the next chapter.

What’s Next?

Could a superintelligent agent with enough general intelligence, or one specialized in a mathematical domain like cryptography, someday figure out that it could still function without blockchain? Could it break out of the bounds provided for it or simply change the immutable data and recalculate all of the hashes without detection by human beings? Quite possibly, but if we build blockchain controls into AI to the point where it won’t work without them, we might be able to keep the upper hand.

You may not be able to change an existing AI fact flow overnight, unless you have full influence over how the stakeholders operate. But learning the four controls and how to apply them, explored in depth in Chapter 2, will give you the ability to offer your stakeholders with a single source of truth for your AI, and to build a human-only backdoor into your AI systems.

---

1 WTVA: Random Braking in Honda Accord and CR-V Under Investigation, *https://www.wtva.com/news/national/random-braking-in-honda-accord-and-cr-v-under-investigation/article_d8e573c6-e7ff-559c-a4b7-48ffde2e908b.xhtml*

2 KXAN: Austin Police hopes artificial intelligence can help with staff shortages - *https://www.kxan.com/news/local/austin/apd-hopes-artificial-intelligence-can-help-with-staff-shortages-at-911-center/*

3 Mitchell.com: AI and its impact on Automotive Claims, *https://www.mitchell.com/insights/auto-physical-damage/articles/ai-and-its-impact-automotive-claims*

4 Some of the information included in Varshney's fact sheets is also stored in ML registries, databases that hold key-value pairs about the ML. Some MLOps systems use the ML registries in MLOps workflow.

Chapter 2. Blockchain Controls for AI

> **A NOTE FOR EARLY RELEASE READERS**
>
> With Early Release ebooks, you get books in their earliest form—the authors' raw and unedited content as they write—so you can take advantage of these technologies long before the official release of these titles.
>
> This will be the second chapter of the final book.
>
> If you have comments about how we might improve the content and/or examples in this book, or if you notice missing material within this chapter, please reach out to the editor at *ccollins@oreilly.com*.

As you learned in Chapter 1, there are many different potential blockchain touchpoints for AI. However, nearly every process has potential to be traced and verified by implementing one of these four controls. This chapter explains how to plan and implement each control by determining potential AI scenarios for each and explaining how it might be addressed using Hyperledger Fabric.

Four Blockchain Controls

When approaching your AI factsheet and thinking about how you might design a fact flow system around it, you may find that thinking of your architecture in terms of these four categories of controls simplifies your planning. Table 2-1 illustrates these four controls and shows how they are tied together by governance.

Table 2-1. The four blockchain controls for AI are interwoven. Governance, in the form of bylaws, smart contracts and consent, overlaps them all.

Control 1: Pre-establish Identity and Workflow Criteria for People and Systems Impacts: participants, assets and transactions Includes criteria for telling humans apart from AI	**Control 2: Distribute Tamper-Evident Verification** Impacts: databases, models, libraries, federated AI Tests data against a cryptographic hash to detect and expose anomalies
Control 3: Govern, Instruct, and Inhibit Intelligent Agents Impacts: production AI Track trace and monitor AI, be able to monitor and stop it	**Control 4: Show Authenticity through User-Viewable Provenance** Impacts: end-users, engineers Consumers of AI can see its history EVEN IF THE AI IS EMBEDDED in cars, robots, virtual reality, etc.

Control 1 deals with identity and workflow, since that is the foundation for all of the other controls.

Blockchain Control 1: Pre-establish Identity and Workflow Criteria for People and Systems

Blockchain systems typically require that criteria such as identity verification, business logic, and so on be met before blocks are added to the chain. Control 1 can be used with AI to make sure that the right people, systems, or intelligent agents (participants) – with the right authorization – are the only ones that take part in governance of or modification to the AI.

In order to be able to establish provenance for AI, start by establishing the identity of the initial stakeholders – the project owner, a consortium or governance group.

Establish and Maintain Identity for Participants

Before diving into digital identity management in blockchain, consider why you need to have a trustworthy identity of the participants - including the

data scientists who train the model. You probably have concerns about trained AI models: who trained the model, what is their background or profile, and how it was trained. This section explores why it is important to be able to trace the identity of people who train AI models and to be able to prove their credibility.

To make sure the right person has trained the model, you need to bind their identity with blockchain by registering the person in the blockchain network, which eventually gives a digital identity signed by their organization's certificate authority. The identity is used to sign or endorse or commit transactions during the lifecycle of the AI model, as mentioned earlier. So the person who performs any activities in the application gets logged into the blockchain. And all stakeholders can verify the data scientist's identity and their work.

The different participants in the blockchain network include the resources like peers, orders, client application, administrator, and so on. These participants get digital identity encapsulated in X.509 digital certificates. This digital identity is very important to determine the participant's access to the available resources.

Public Key Infrastructure (PKI)

Any identity may be supplied to the blockchain network for resource access, but do they get verified? Only the identity coming from a trusted source can only be verified. Hyperledger Fabric's Membership Service Provider is a trusted source (MSP). MSP is a fabric component that deals with cryptographic mechanisms and protocols behind issuing certificates, validating certificates, and authenticating users. MSP uses a *public key infrastructure* (PKI) hierarchical model that facilitates secure communication in the network.

PKI issues verifiable certificates, whereas MSP does have the list of verified certificates so when a certificate comes over the network for the verification, MSP checks whether it's on its list. If it's on the list, the certificate is verified – otherwise, it's not.

PKI has four key elements:

- Digital certificates
- Public and private keys
- Certificate authorities
- Certificate revocation lists

The key elements are carried out with the following:

1. *Digital certificates:* A digital certificate holds a set of attributes about the holder of the certificate which is also compliant in X.509 standard. This stores the basic info of the holder, subject, public key, validity, serial number, signature, and so on.
2. *Public and private keys:* Authentication and message integrity are important concepts in secure communication. While exchanging the message between two parties, the message sender signs the message using his private key which is verified at the receiver's end using the sender's public key. If the message gets tampered with during transmission, then the key doesn't match. So in this way, a private key is used to produce the digital signature.
3. *Certificate authorities (CA):* An actor in the blockchain network can participate in the transaction and is known to other actors of the network, by means of their digital identity, which is issued by each organization's CA. CA has the right to dispense certificates to different actors. These certificates are digitally signed by CA and bind together the actor with the actor's public key. So when the actor sends a transaction for the endorsement in the blockchain network, the actor's certificate issued by CA and public key is written to the transaction and sent to other nodes where other nodes verifies whether the transaction is legitimate or not by comparing signature from CA and public key they are known to. There are two kind of CAs:

- *Root certificate authority (RCA):* A blockchain network can have multiple RCAs issued for different organizations, and the RCA assigned for an organization can sign thousands of other certificates. But signing all actors' certificates by the RCA itself is a time-consuming task and is less secure as well, so to reduce the load and spread the work, intermediate certificate authorities (ICAs) are created. If the certificate fields `issued to` and `issued from` have the same values, then it is considered that the certificate is signed by RCA.
- *Intermediate certificate authority (ICA):* ICAs are used to spread out the process across the network. ICA has certificates issued by RCA or other ICA. An ICA can sign the certificate of another ICA and so on, which ultimately can create a chain of trust. If we trace back the signature made in each certificate, then we finally reach RCA so we can say each certificate is signed by RCA through ICA, without exposing its own identity. In this way RCA is not directly exposed to all ICAs and it is good from a security point of view, as shown in Figure 2-1.

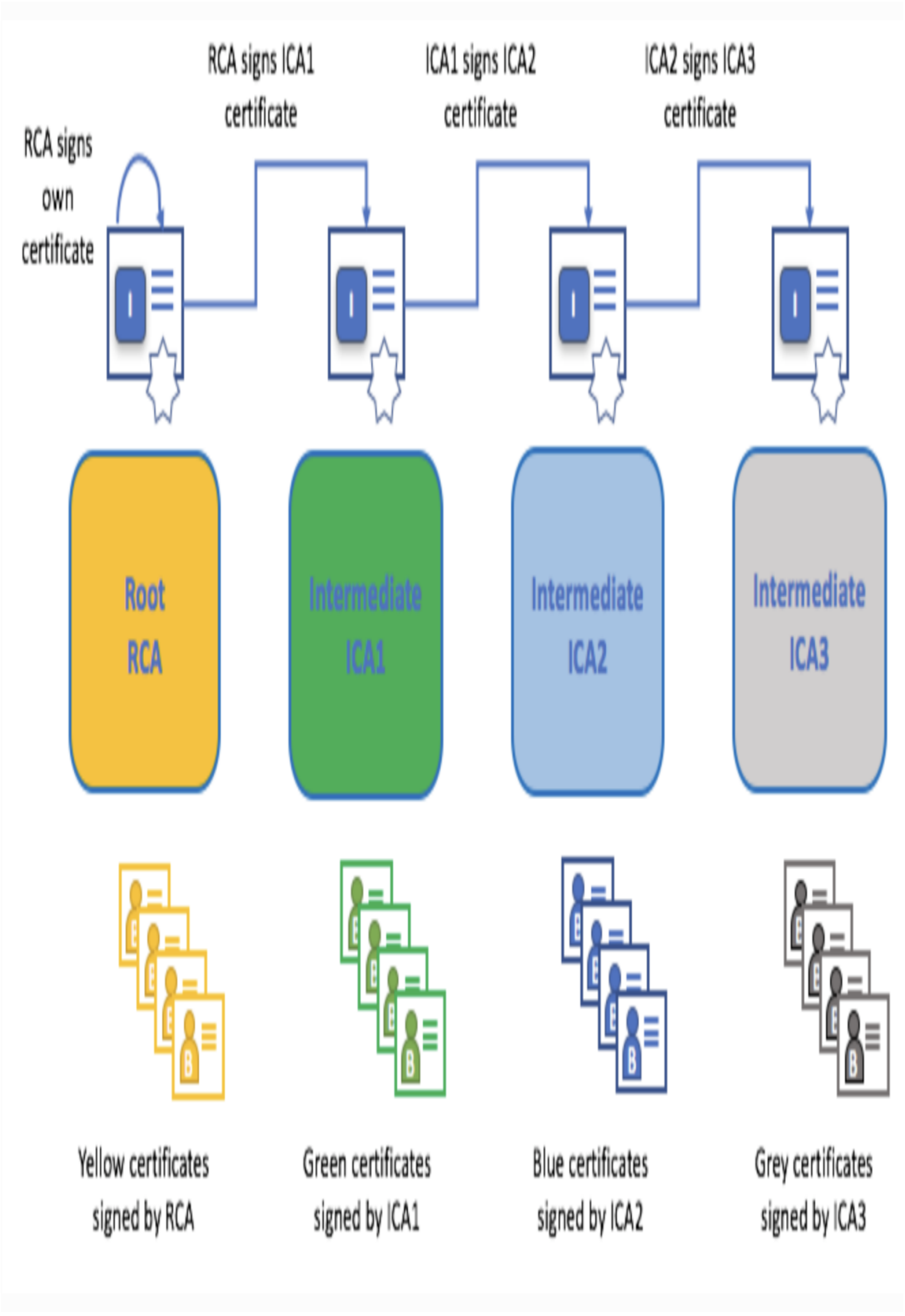


Figure 2-1. RCA/ICA signing certificates.

4. *Certificate revocation lists (CRL):* The revocation list, illustrated in Figure 2-2, in the permissioned blockchain like Fabric helps to know the actors whose permission is revoked because of their suspicious activities. So when such an actor tries to perform an action in the network, their certificate is cross-checked with CRL. If it gets passed from there, it can perform activities there; otherwise, the actor will be blocked from performing any action in the network.

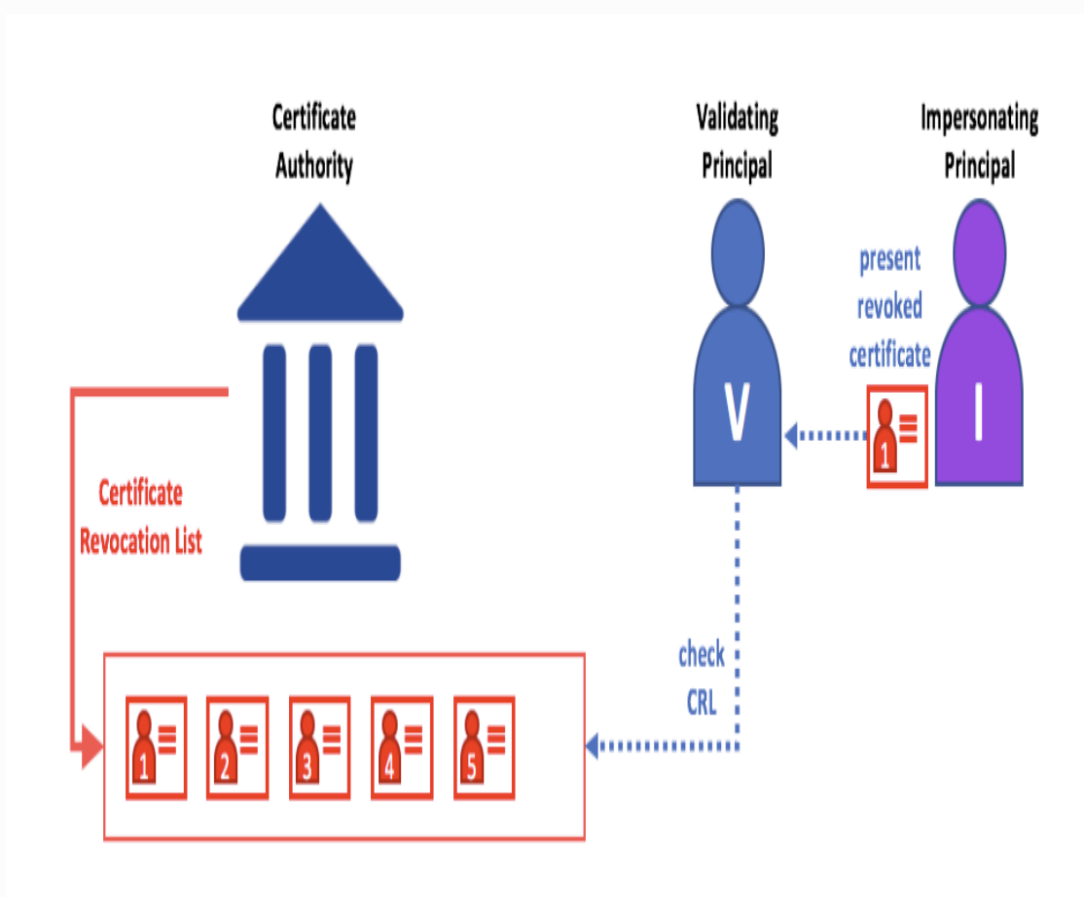


Figure 2-2. Hyperledger Fabric revocation list.

Membership Service Provider (MSP)

So far we've looked at the actor's public key infrastructure (PKI). When an actor gets an identity in the network, the actor gets public and private keys issued by CA. Private keys cannot be shared in public, so you need a standard mechanism to prove the actor is the valid one, and this mechanism is called MSP (Figure 2-3). An actor digitally signs or endorses or commits

a transaction using a private key. MSP in the ordering service contains a list of permissioned public keys of the actors, which is used to verify the signature on endorsement or transaction and validate.

Figure 2-3. Identities are validated with the MSP similar to how a credit card is validated by a merchant.

MSP occurs in two domains:

- *Local MSP:* Local MSPs are defined for clients and for nodes (peers and orderers). Local MSPs define the permission for the node, such as who are the admins who can operate the nodes. A local MSP allows the

user to authenticate itself in its transaction as a member of a channel. An organization can own one or more nodes. The person can be an admin in multiple nodes, which is defined by MSP. An organization, node, and admin should have the same root of trust. That means if we back track the CAs, they all should reach the same RCA.

- *Channel MSP:* Channel defines the administrative and participatory right in the channel level. Peers and ordering nodes share the same view of channel MSPs, and they are correctly able to authenticate the channel participant. Figure 2-4 shows how organizations' MSPs are defined in the project's config.json file.

Figure 2-4. Snippet from a channel config.json file that includes two organization MSPs

Channel MSPs identify who has authority at a channel level. The channel MSP defines the *relationship* between the identities of channel members and the enforcement of channel level policies. Channel MSPs contain the MSPs of the organizations of the channel members. Every organization participating in a channel must have an MSP defined for it. The system channel MSP includes the MSPs of all the organizations that participate in an ordering service. Local MSPs are only defined on the file system of the node or user. Channel MSP is

also instantiated on the file system of every node in the channel and kept synchronized via consensus.

Figure 2-5 shows how a common channel is used to share data across two different organizations where the network system channel is administered by Org1, Peer is managed by Org2, and Orderer is managed by Org1. Org1 trusts identities from RCA1, and Org2 trusts identities from RCA2.

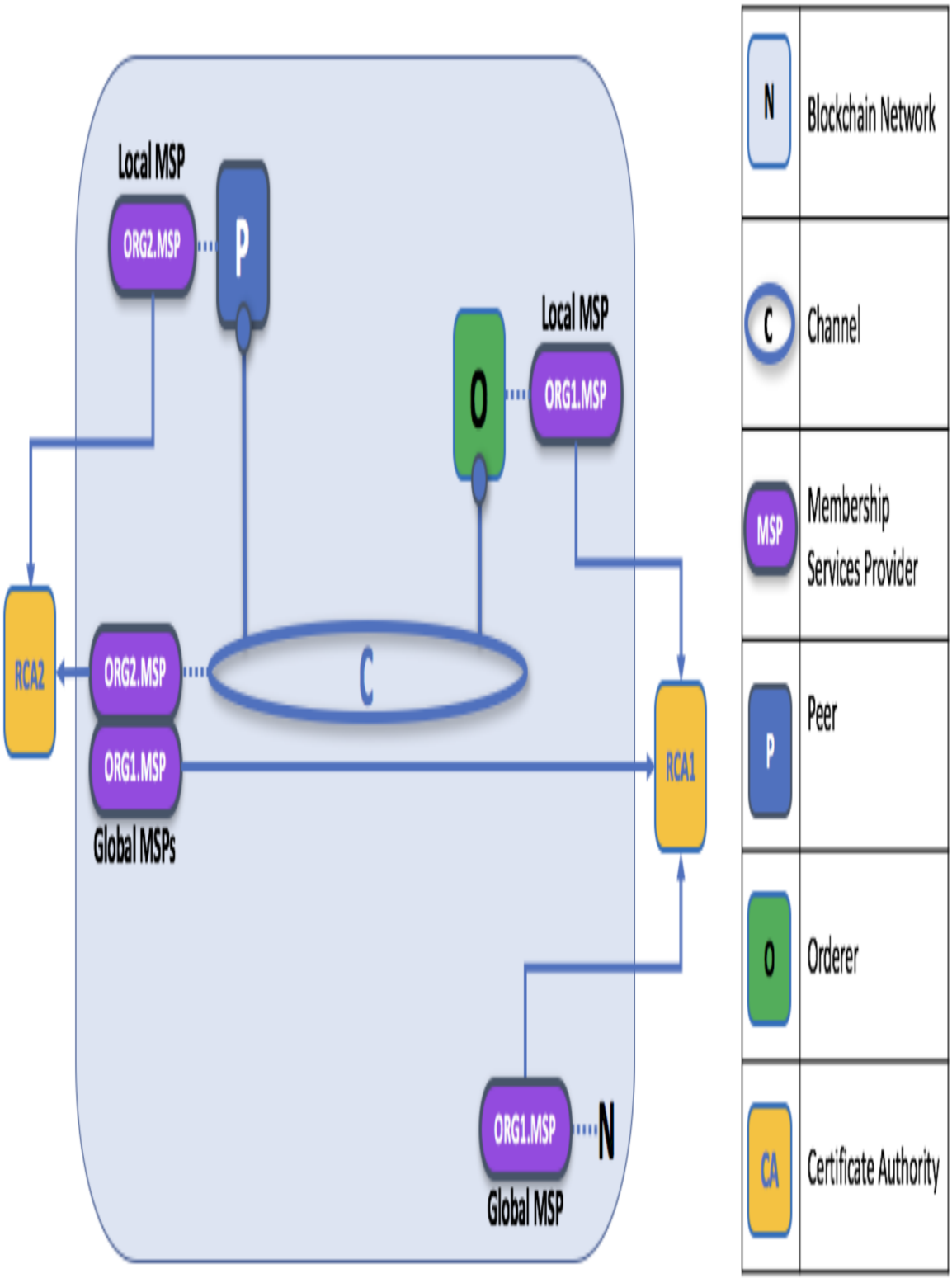


Figure 2-5. Local and Channel MSPs

Each organization is a logical managed group of members where their identities are managed in MSP. Every organization has its unit, which can be termed as *department* or *unit*. So, OU represents a separate line of business and also has a line in configuration file *OU* which can be later used in policy making to restrict the access through a smart contract.

Another important component of OU is `node OU` which deals with the identities and it is defined in the *yaml* file *$FABRIC_CFG_PATH/msp/config.yaml*, which contains the organization unit (OU) whose members are considered to be part of the organization representation by this MSP.

The following configuration shows Node OU roles in MSP including client, peer, admin, and orderer:

```
NodeOUs:
  Enable: true
  ClientOUIdentifier:
    Certificate: cacerts/ca.sampleorg-cert.pem
    OrganizationalUnitIdentifier: client
  PeerOUIdentifier:
    Certificate: cacerts/ca.sampleorg-cert.pem
    OrganizationalUnitIdentifier: peer
  AdminOUIdentifier:
    Certificate: cacerts/ca.sampleorg-cert.pem
    OrganizationalUnitIdentifier: admin
  OrdererOUIdentifier:
    Certificate: cacerts/ca.sampleorg-cert.pem
    OrganizationalUnitIdentifier: orderer
```

OU roles and its actors' keys are normally included in the *config.yaml* configuration file.

When you drill down on the digitally signed certificate, Figure 2-6 shows how it looks and what it includes.

Certificate:

Data

Version: 3 (0x2)

Serial Number:

45:6a:4f:01:dc:fj:5d:b2:94:18:79:91:26:31:d8:0e:b0:9b:6b:88

Signature Algorithm: ecdsa-with-SHA256

Issuer: C=US, ST=New York, O=Hyperledger, OU=Fabric, CN=fabric-ca-server

Validity

Not Before: Nov 20 22:13:00 2019 GMT

Not After : Nov 19 22:18:00 2020 GMT

Subject: OU=peer, OU=ORG1, OU=DISTRIBUTION, CN=user1

ROLE (Node OU) ORGANIZATIONAL UNIT ENROLL ID

.

.

.

X509v3 extensions:

X509v3 Key Usage: critical

Digital Signature

X509v3 Basic Constraints: critical

CA:FALSE

X509v3 Subject Key Identifier:

17:B0:9B:29:42:F6:44:E0:7D:02:C6:78:96:2D:97:14:7A:D7:FC:CA

X509v3 Authority Key Identifier:

keyid:DC:91:B7:85:A4:37:66:D0:D2:B7:62:A9:3F:59:83:D6:EB:01:E8:80

1.2.3.4.5.6.7.8.1:

ORGANIZATIONAL UNIT ENROLL ID ROLE (Node OU)

{"attrs":{"hf.Affiliation":"ORG1.DISTRIBUTION","hf.EnrollmentID":"user1","hf.Type":"peer"}}

Figure 2-6. Drill down into a digitally signed certificate.

Shown in Figure 2-7 and Table 2-2, the MSP folder contains critical certificates and config files.

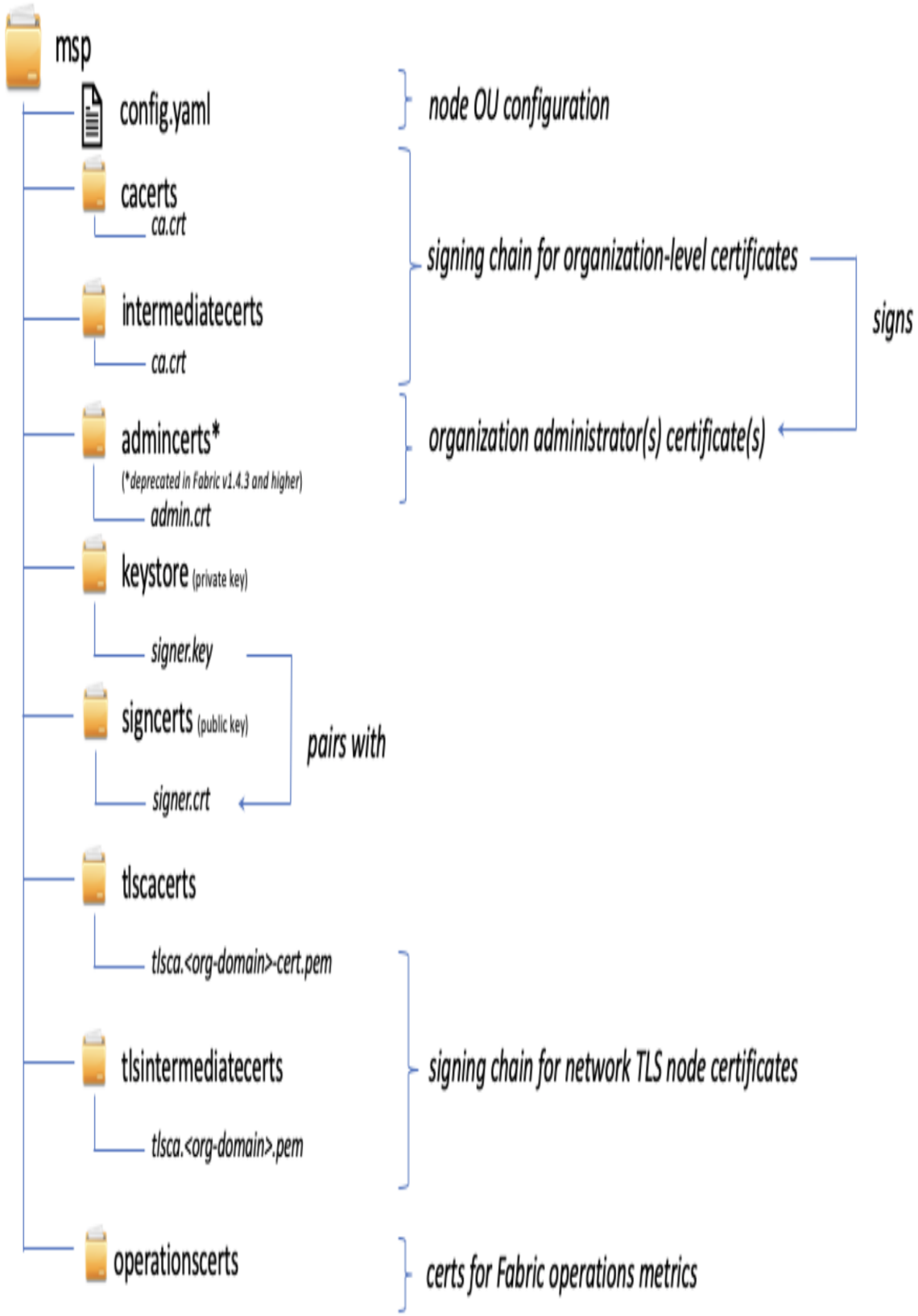

msp
config.yaml
node OU configuration
cacerts
ca.crt
signing chain for organization-level certificates
signs
intermediatecerts
ca.crt
admincerts*
(*deprecated in Fabric v1.4.3 and higher)
admin.crt
organization administrator(s) certificate(s)
keystore (private key)
signer.key
signcerts (public key)
pairs with
signer.crt
tlscacerts
tlsca.<org-domain>-cert.pem
tlsintermediatecerts
tlsca.<org-domain>.pem
signing chain for network TLS node certificates
operationscerts
certs for Fabric operations metrics

Figure 2-7. Config files and certificates in the MSP folder.

Table 2-2. Config files and certificates in the MSP folder, and their purpose.

Filename	Purpose
config.yaml	Enables NodeOU and defines the roles.
cacerts	Contains a list of self-signed X.509 certificates of the R CAs trusted by the organization represented by this MSP. There must be one Root CA certificate in the MSP folder.
intermediatecerts	This folder contains a list X.509 certificates of the intermediate CAs trusted by this organization. ICA may represent different organization unit like MANUFACTURING, DISTRIBUTION.
admincert	This folder contains the list of identities of actors who have the role of admin for this organization. admincert is depreciated from V1.4.3 so in the higher version, admin role is defined in Node OU by enabling “identity classification” option.
keystore	This folder is defined for the local MSP of peer or orderer which contains only one private key of the node to sign the transaction proposal response, as part of the endorsement process.
signcert	This folder contains the node’s certificate issued by CA.This folder is mandatory for local MSP which contains exactly one public key.
tlscacerts	This folder contains a list of self-signed X.509 certificates of the Root CAs trusted by this organization for secure communications between nodes using TLS.
tlsintermediatecacerts	This folder contains a list of intermediate CA certificates CAs trusted by the organization represented by this MSP for secure communications between nodes using TLS.

Predetermined Workflow Among Participants

Chapter 1 discussed what sort of workflow takes place in the creation and handling of an AI system. Since blockchain is the single source of truth for your AI, it is important to have all of your workflow mapped out and to design your system in such a way that workflow can be changed by system administrators, because it will likely change often.

Think back to the Chapter 1 discussion of participants, assets, and transactions and apply this to the workflow involved in generating the AI factsheet. Table 2-3 gives a few suggestions that are arranged by AI

functionality, and include where the data or system is hosted, along with a breakdown of the participants, assets, and example transactions, along with the blockchain controls that can be used.

Table 2-3. Implementing Blockchain-Tethered AI: This table shows examples assets & transactions by AI functionality. All utilize control 1, identity and w

Functionality	Hosting and/or Storage	Participants	Assets	Example Transactions
application layer	in file system of container code in repository	application developer, DevOps	html, css, databases, file systems	application developer mak changes to cod and changes ar deployed to production by DevOps or continuous deployment pr
MLOPs	on network devices	trainer, validator, fusioner	model	validator inspe training data se characteristics approves mode fusion
ML content	in object store with verification hashes on blockchain or on chain	developer, DevOps, database admin (DBA), AI agent	structured data, unstructured data, databases, tables, libraries, hosts, servers, containers	developer introduces data DevOps update server software DBA updates d set

You can expand this table to add functionality and touchpoints for your particular system.

ML Pipeline and MLOps

Perhaps the most workflow-intensive area in AI is the ML pipeline, in which models are created, enhanced, and deployed by developers, data scientists, and DevOps.

While it is in the ML pipeline, the prediction problem to be solved is *operationalized* (broken down into meaningful components), and the model is *trained* (influenced with data) and then further improved with experiments. Then system administrators and deployment specialists use traditional DevOps techniques to push the container housing the model live.

In Figure 2-8, blockchain is used to enforce governance of a retrained model that no longer meets established requirements. In this example, the experiment is rejected because it includes a triangle, when only circles and stars are allowed. Blockchain is used as a ledger to record each step. In this scenario, the accept/reject criteria recorded onto blockchain would be tested before the model could be deployed to production.

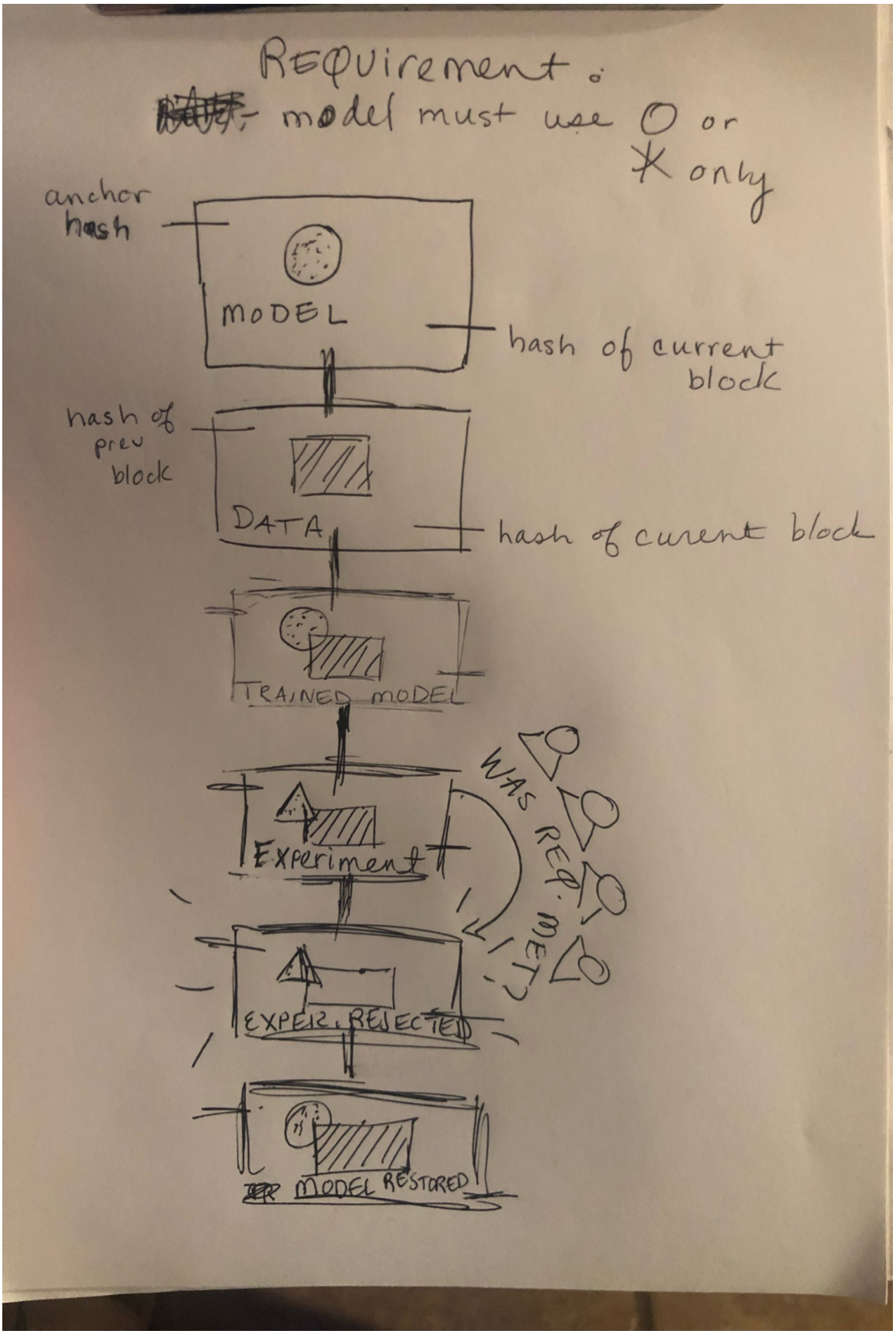

Requirement:
model must use O or * only
anchor hash
MODEL
hash of current block
hash of prev block
DATA
hash of curent block
TRAINED MODEL
Experiment
WAS REQ. MET?
EXPER. REJECTED
MODEL RESTORED

Figure 2-8. Example of use by a model validator.

Created to automate this process, *MLOps* is a modern methodology that combines the ML pipeline with CI/CD. MLOps uses a process similar to traditional DevOps to manage and deploy code, with special steps added for ML projects.

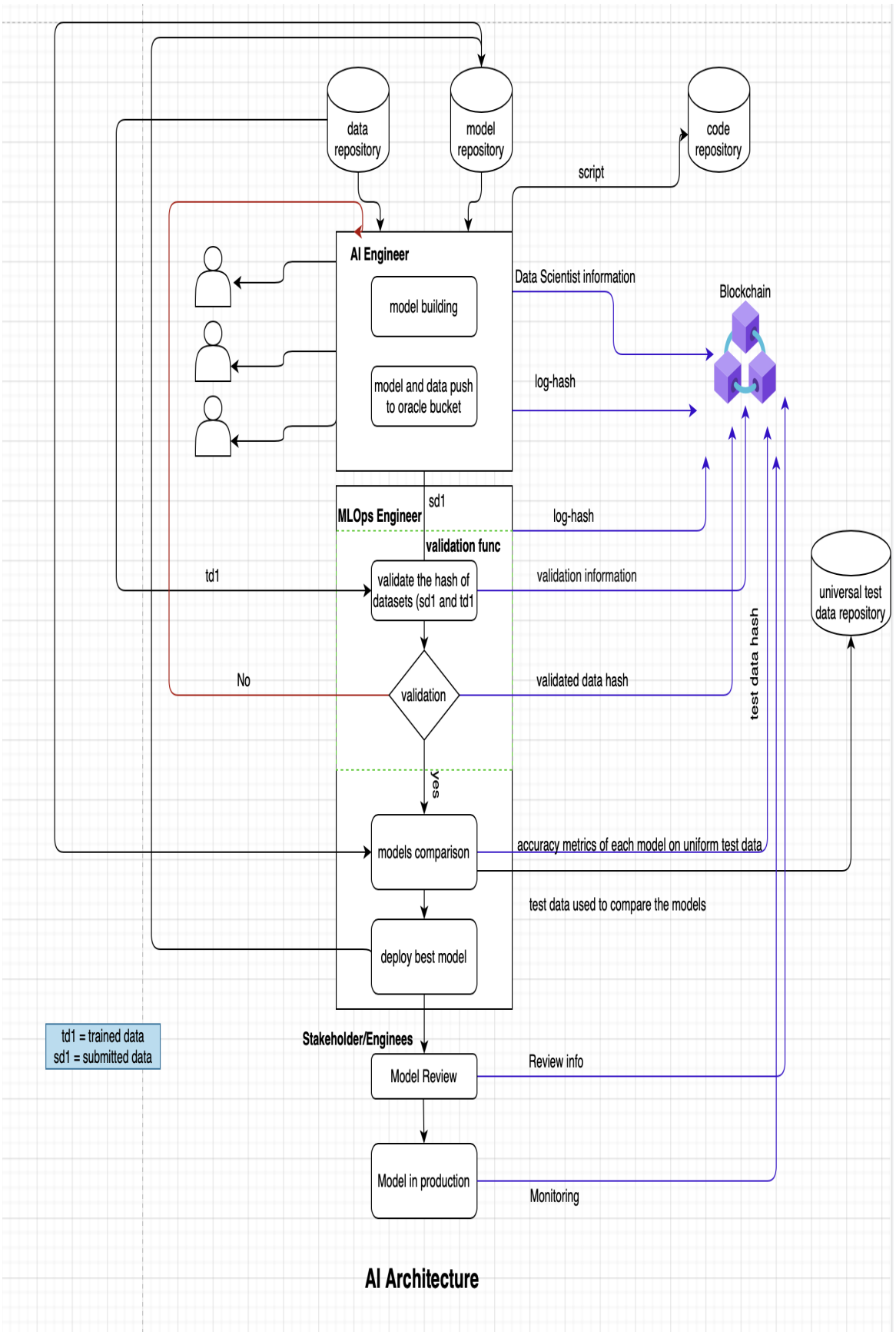
MLOps provides a formalized, automated framework for the steps used to train, deploy, and improve a model. A high-level look at the steps used to validate the model looks something like Figure 2-9.

Figure 2-9. From the ML pipeline, the high-level model validation steps

Even using MLOps, it is tough to trace the ML *provenance*, which is a record of the journey the AI has taken to get where it is at. Often the ML experiments are run for many iterations on a massive number of nodes, and only models with the desired traits are included in those that go to production.

In the current trend of training AI models, different MLOps methodologies are used to train the model, where logs of each and every activity of a data scientist are recorded in a certain database (SQL/NOSQL). Once a training session is completed, the tool locks the metadata and artifacts of the training so one can trace back the history. *But* again the whole system is based on a centralized system where a programmer can easily reach and change the content from the database. So, there is a problem in maintaining provenance of a model/dataset. Blockchain can be a problem solver for this kind of issue where each stakeholder gets their own node that stores logs of trained models. Only that person has access to the node, and no one can modify it, not even that person.

Figure 2-10 shows an in-depth model validation workflow with blockchain touchpoints added. Chapter 5 explores this workflow in depth as we build the model validator example for our AI-tethered blockchain project.



AI Architecture

Figure 2-10. How blockchain can be applied to model validation

Data Cleaning/Bias Reduction

Data cleaning and AI bias are related issues because bias is often baked into ML training data. This section examines the far-reaching bias problem, and some possible approaches for identifying and correcting it through leaving an traversable audit trail that help people to pinpoint and even reverse the bias.

Bias comes in many forms, and everyone has some bias or another. One of the first steps in combating bias is to have a clear organizational culture, including guidelines that address bias. However, the bigger issue lies in the data and the algorithms, because if these are biased by nature, even the most well-intended AI system can turn bad. Part of risk management is managing bias.

There are dozens, if not hundreds, types of bias. *Bias* is the product of someone's environment and conditioning. If someone grows up as a vegetarian, they would likely be biased against meals that contain meat. They might not have ill intentions - they may not realize they have a bias at all. But this bias might surface if they are preparing a data set that recommends healthy meals. Here are some common types of bias encountered in ML:

- *Social:* Within a culture or society, inequities that are introduced into the input data
- *Classifier:* Also known as *representation*, this is when there are inadequate, unrepresented, or unrealistic classifications, leaving out portions of a group or demographic.
- *Data selection:* Under-represented or unrealistic training data
- *Data preparation:* Inadequate cleaning up or preprocessing of the data that results in inadequate or invalid data
- *Automation:* A secondary system failure that feeds poor training data to the AI

- *Reporting:* The occurrence statistics are misrepresented by multiple orders of magnitude due to unrealistic training data.
- *Data sabotage:* Purposefully tampering with the data, injecting false information.
- *Time change:* An example would be if the time to produce data (the throughput) changes drastically, and the model sees "faster" as better, without validating that it's true - that's an issue that will need to be addressed.
- *Data shift:* A shift in the distribution over time.

Analyzing the output rather than the algorithms keeps intellectual property confidential in terms of sharing information with the customer, but someone will still need to tweak the model. Figure 2-11 shows the datatypes to look for.

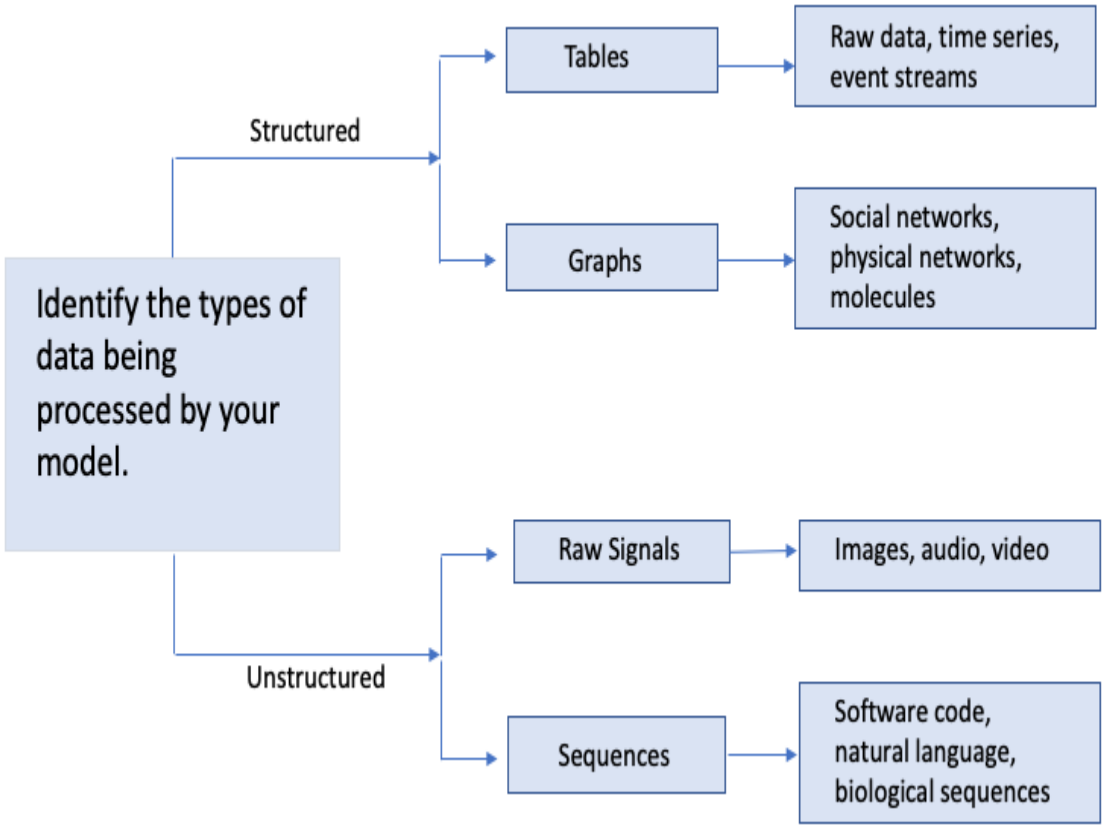


Figure 2-11. Analyzing data begins with identifying the types of data being processed by your model. (Kush R. Varshney, "Trustworthy Machine Learning," p. 42)

These are the best practices for reducing data bias:

1. Examine data collection sources. Understand the context in which the data is captured, and then in what contexts AI can correct or exacerbate bias.
2. Make sure the training datasets are large enough and taken from all available sources to prevent sampling bias. Analyze whether the algorithms work when used on subsets of the larger population. Monitor results over time – machine learning reveals biases in the output.'
3. As part of a multidisciplinary strategy to eliminate bias, diversify your organization to get different points of view. This strategy would ideally

include ethicists, social scientists, and experts who best understand the nuances of each area of machine learning processes.

4. Establish a debiasing policy that includes technical, operational, and organizational strategies. Cultivate data analyzing tools and strive for transparency in policies and operations. This includes improving human-driven processes, which have traditionally driven the creation of models, through process design and cultural training. Then, decide which use cases need human intervention.

> **TIP**
>
> Data integrity is the basis for sound machine learning and artificial intelligence. If your data is tainted in any way, the resulting algorithm will be flawed, and the results of running subsequent data through the algorithm will also be tainted.

Visual recognition AI provides many obvious examples of bias. Flawed results range from the misclassifying and misidentification of a piece of candy as a kitchen tool, dogs as cats or some other mammal, misidentifying a bricked tower as a carrot, or a human as anything but one. Because ML requires both positive and negative examples of the image that is being "learned," it's incumbent upon the data scientist to think of as many things as possible that may look like the image, but are not - like a wrapped Tootsie Roll being misidentified as a kitchen rolling pin. That covers visual recognition, but there are far more complex examples.

Author, technologist, and futurist Jaron Lanier has been quoted in several interviews as saying that, with respect to the internet and particularly social media, the misuse of user data as predictive tools for marketing companies and the subsequent feeding of the response of the same user to marketing efforts by those same companies has pushed the internet into a cycle of manipulation of that user. The third-party internet company hosting the marketing is just a platform providing "information" and "social interaction" for the user.

But what's really happening is that the user is being manipulated to respond to negative information (words, images) over and over again, by playing on their emotions with incendiary content and peppering that with occasional positive feedback to keep the user addicted ("influenced," social media content creators like to say) to the content playing in a similar sequence of events. With *confirmation bias* (agreement with a group), a user will get a dopamine hit from the response and keep coming back. The user seeks and finds information that confirms their viewpoint over and over, getting more and more satisfaction from using social media. It all creeps in so slowly, the user doesn't realize it until someone's bugging them to put down their phone while they're at the dinner table. A machine learned how to capture the user's attention, and the artificial intelligence keeps them there, "influenced." Then the ML harvests their reactions so it can fine-tune its approach.

What happens when AI fails to properly, or at least, confidently, identify a known object because personal, unconscious biases prevent the developer from creating adequate training data sets? What happens when AI has been inadequately trained, and is only able to confidently identify a relatively small subset of known variations on that object instead of the full breadth of variations? A comprehensive list of biases was published by Marcus Lu in February 2020. In addition to confirmation bias, he lists 49 more cognitive biases that are commonly found on social media, including the *Dunning-Kruger effect* and *belief bias*.

The *Dunning-Kruger effect* goes something like this: the less you know, the more you think you know because you're filling in the blanks with your own guesses, and the more you know, the less you think you know because you find out that you were wrong in so many ways. *Belief bias* is when we judge the strength of an argument based on how plausible the conclusion is in our minds rather than how strongly it supports the conclusion. Looking at the bias list, there are a number of other biases that might play out in a scenario where participants are making policy or operations decisions and signing them on smart contracts, but their decisions are not sound.

THE ROLE OF SMART CONTRACTS

Smart contracts are defined in Chapter 1 and explored in depth in Chapter 3, but they are worth mentioning again here. Smart contracts help determine what criteria has to be met before certain actions take place on blockchain. For instance, your fact flow system could route an approval on a new model from a data scientist to a model validator before a model is deployed via MLOps.

Then perhaps another stakeholder, like a product manager, has to make a final approval before the model is released. While this type of workflow is accomplished through outlying systems that communicate with blockchain via API, blockchain is optimized for recording it. This is because blockchain is designed to post blocks based on consensus, and reject those that do not meet proper conditions.

Blockchain Control 2: Distribute Tamper-Evident Verification

Blockchain is *tamper-evident*. This means if someone or something changes any of the data by brute force, computing the same hash becomes impossible, which makes the unauthorized changes apparent. Remember that blockchain, which stores verification hashes for data as a series of time-stamped, linked units called *blocks*, is a peer-to-peer network that can be shared among multiple stakeholders.

Control 2 can be used with AI to verify that data and algorithms have not undergone tampering or corruption, as analyzed in Table 2-4.

Table 2-4. A breakdown of participants, assets and transactions for a pre-pr

Functionality	Hosting and/or Storage	Participants	Assets	Example Transactions
Pre-Production Model	in repository with hashes on blockchain	developer, DevOps, AI agent	model/ algorithms, repositories, hosts, code origins, contributors, release notes, patches, certifications	create code, up code, delete co create new rele receive update federated mode

Using Crypto Anchors to Verify Data Sets, Models, and Pipelines

As mentioned earlier in the “Identity” section, the validity of anything on blockchain is only as good as the identity methods associated with the participants and assets. Identity is the foundation upon which all of the transactions are assumed to be authentic - if identity is not robust, then the entire blockchain is in question. Crypto anchors are a way to tie the physical identity of a participant or asset to their digital identity in a way that can be verified.

Hardware items typically have some sort of a unique identifier associated with them such as an engraved serial number, like a vehicle identification number on a car.

Other goods are more difficult to uniquely identify - for instance, honey or diamonds. These don’t come with a unique identifier, but instead have unique characteristics. For example, if honey is watered down during the supply chain voyage, its molecular structure will no longer look the same when examined microscopically. Since crypto anchors are digital, it is possible to create a cryptographic signature of a specimen of honey from a hash of the microscope’s output and store it on blockchain, making every block containing every transaction in the supply chain journey carry the signature of the molecular structure of the original product.

Crypto anchors sound more complex than they are. You can easily create a crypto anchor by creating a hash of facts aggregated with a unique identifier, make that crypto anchor the contents of a block (could be the "*genesis,*" or very first, block in a blockchain), and every block after will be signed with it. Some suggestions for AI crypto anchors are shown in Table 2-5.

Table 2-5. Ideas for using crypto anchors to tether AI systems.

Crypto Anchor target	Crypto Anchor contents
Training data set	timestamped hash of data facts including size, characteristics and architecture+MAC ID
model	timestamped hash of model facts + model's digital identity from X.509 digital certificates + hardware information such as serial number of part in which AI is embedded
pipeline	timestamped hash of network facts +

To tell if digital content has been modified or if the hardware has changed, you can build a verifier that re-computes the crypto anchor and checks it against the value stored on blockchain.

Using Blockchain to Detect Common AI Hacks

As Chapter 1 addressed, AI models, like any other systems, are subject to hacks and attacks. Common attacks on AI include adversarial data attacks, like poisoning and evasion attacks, as well as model stealing, and impersonation attacks. All of the normal security concerns are of consideration, plus a host of new threats because of the extra steps and mystery of origin involved in ML, and the ability for models to change based on input. Table 2-6 explains some of the most common hacks from Chapter 1, along with ideas on how to use the blockchain verification to prevent or expose them.

Table 2-6. Types of attacks and potential ways to prevent or expose them.

Type of Attack	Steps to Prevent and/or Expose Attack
Poisoning attack	Since this attack involves malicious data changing the classifier, one way to prevent a poisoning attack would be to create a smart contract that monitors for classifier mis-matches. This could be done by locking down the list of permissible classifiers for data with certain characteristics in a separate channel, so the ML can inquire to blockchain to see if something is permitted without seeing the list of what is permitted. A poisoning attack could be exposed in a similar way by detecting when a data set has been poisoned and red flagging it on blockchain as unsuitable for use. The last good data set could be found by reviewing previous data sets that were verified as valid, and re-running the training from the last good set.
Evasion attack	Since this attack distorts production system test samples to push data points beyond the decision boundary, one way to prevent it is to make sure your samples come from a reliable source. For instance, if you are using Microsoft ML Samples Gallery you will want to record that source as authentic on blockchain, and make sure your MLOps flow checks blockchain before checking samples. You can flag this type of attack by recording any attempts to use data sets that are not listed on blockchain.
Model stealing	Because the model is stolen and recreated elsewhere in this type of attack, making your models fully dependent on their blockchain node would break this method, since the attacker could not spoof the required blockchain node. Alerts of attempts to steal models could be monitored by alerting for outside attempts to probe for blockchain nodes.
Impersonation attack	Samples such as faceprints or fingerprints of victims are replaced with wrong samples, allowing unauthorized access to any number of systems. If the original account owner records biometrics to blockchain, then those biometrics can be compared historically to any new biometrics, and any anomalies can be flagged for intervention. Using blockchain, it would be impossible for the attacker to get rid of the original biometrics.

TIP

You might not be able to prevent all attacks immediately, but having a methodology like blockchain verification to chip away at them will give you a more robust system in the long run.

Federated Learning and Blockchain

In some cases big, real problems could be hiding deep inside the opaque world of AI and ML. *Federated learning* is a special type of machine learning where remote devices train the models, which are later aggregated into the main model without transferring the data. Federated learning is different in that the model resides with the data, and the training takes place remotely. This means that the trained model, absent its data, needs to be trustworthy on its own.

Typically, the training data set is cleaned up and prepared for use and then the model is built using selected ML algorithms. The model is trained and finally rolled into production. A typical federated learning process is shown in Figure 2-12. In this chart you can see the distributed data, the invocation of the training, and finally the remotely trained model being synced with the production model.

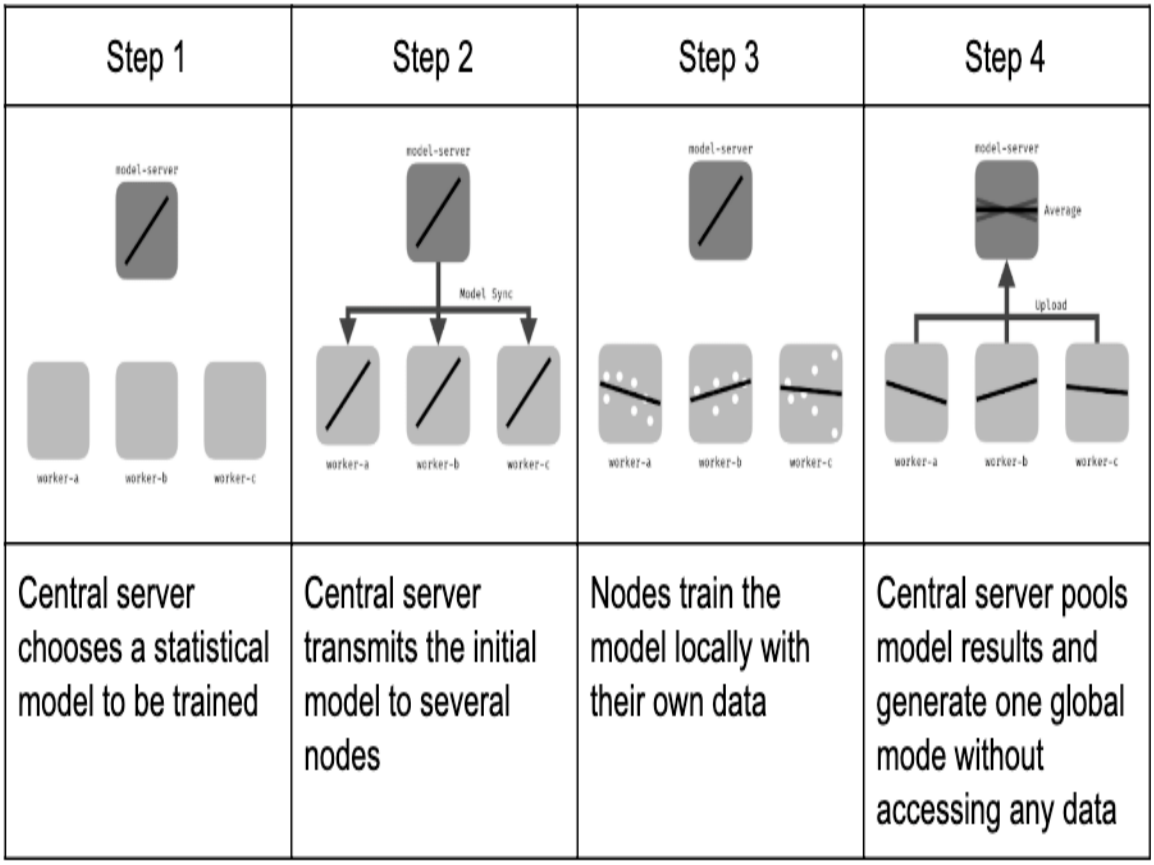


Figure 2-12. Federated learning for visual recognition data. Figure based on image by Jeromemetronome via Wikimedia Commons, CC By S.A. 4.0.

Because models that have been trained by federated learning must be trustworthy independent of their data, there could be trust issues if you cannot verify provenance and provide transparency. This can be fixed by recording a tamper-evident ledger of all training events and underlying components, and making a verification available when the model is instantiated. This is the use case that Chapter 5 deep dives into: how to merge new models into production while keeping the provenance of the models."

Model Marketplaces

Pre-trained models save money and time, and speed up the ML process significantly. Common tasks and domains are addressed, and all the data scientist has to do is further train the model to fine-tune it. Often these models are hosted elsewhere and accessed via API. Data is pre-cleaned,

specifications of the data and intended uses are outlined in data fact sheets, and it is often tested for anomalies like bias.

One such site is huggingface.co. Hugging face offers not only pre-trained models via API - more than 20,000 of them - but also offers data sets with accompanying fact sheets and tools to automate training. Huggingface has tiered subscription pricing.

Another model marketplace is run by AWS. In the AWS Model Marketplace, models can be selected through a point-and-click interface and be deployed on AWS EC2 instances.

> **WARNING**
>
> Like other online marketplaces, it is advisable for you to proceed with caution. While the marketplaces do some due diligence with their participants, results likely vary and the best idea is to evaluate the supplier yourself and keep track of their information in your own fact sheet.

Next, consider how to control your AI once it has been released to production.

Blockchain Control 3: Govern, Instruct, and Inhibit Intelligent Agents

Blockchain can provide proof it followed pre-established technical and business policy criteria by requiring every significant AI event be recorded in a tamper-evident ledger. For example, when a model or training data is modified it would be traceable on blockchain, detailing who approved the change and how (and possibly why) the change was implemented.

Control 3 will become very important when wanting to trace or reverse AI, or prove in court that the output of AI is traceable to certain people or organizations. Table 2-7 shows the breakdown of two governance-related processes into participants, assets and transactions.

Table 2-7. Breakdown of participants, assets and transactions for governanc

Functionality	Hosting and/or Storage	Participants	Assets	Example Transactions
Stakeholder workflow	cloud container	managers, investors, system planners	proposals, contracts, declarations, approvals, agreements, planning documents, specifications, requirements	Stakeholders ag that system sha never fall belo 98% accuracy
Production Model	content, architecture and model combined, resulting in a genie or soverign agent. Runs on one or more hosts, could be embedded in products	production model, AI agent, developer, DevOps	rules, intent, logs, predictions, analytics, performance, recalcitrance, interventions, recalls, termination	performance, behavior incide complaints, rat reviews, chang made to own c changes made own data, error and outages

To think about how to approach AI's fact flow and ML pipeline, consider a supply chain example where lettuce was tracked through the supply chain from the farm to Walmart's shelf using an RFID sticker that was scanned by authorized personnel and tracked using IBM's blockchain technology at each step along the way. A detailed writeup of the entire project was written up in detail in an October 2020 blog on Pixelplex.io and runs through all the players, reasons, and risks.

IBM has formalized and is offering the software to manage the process flow; this is called IBM Food Trust. An overview of this process flow is shown in Figure 2-13.

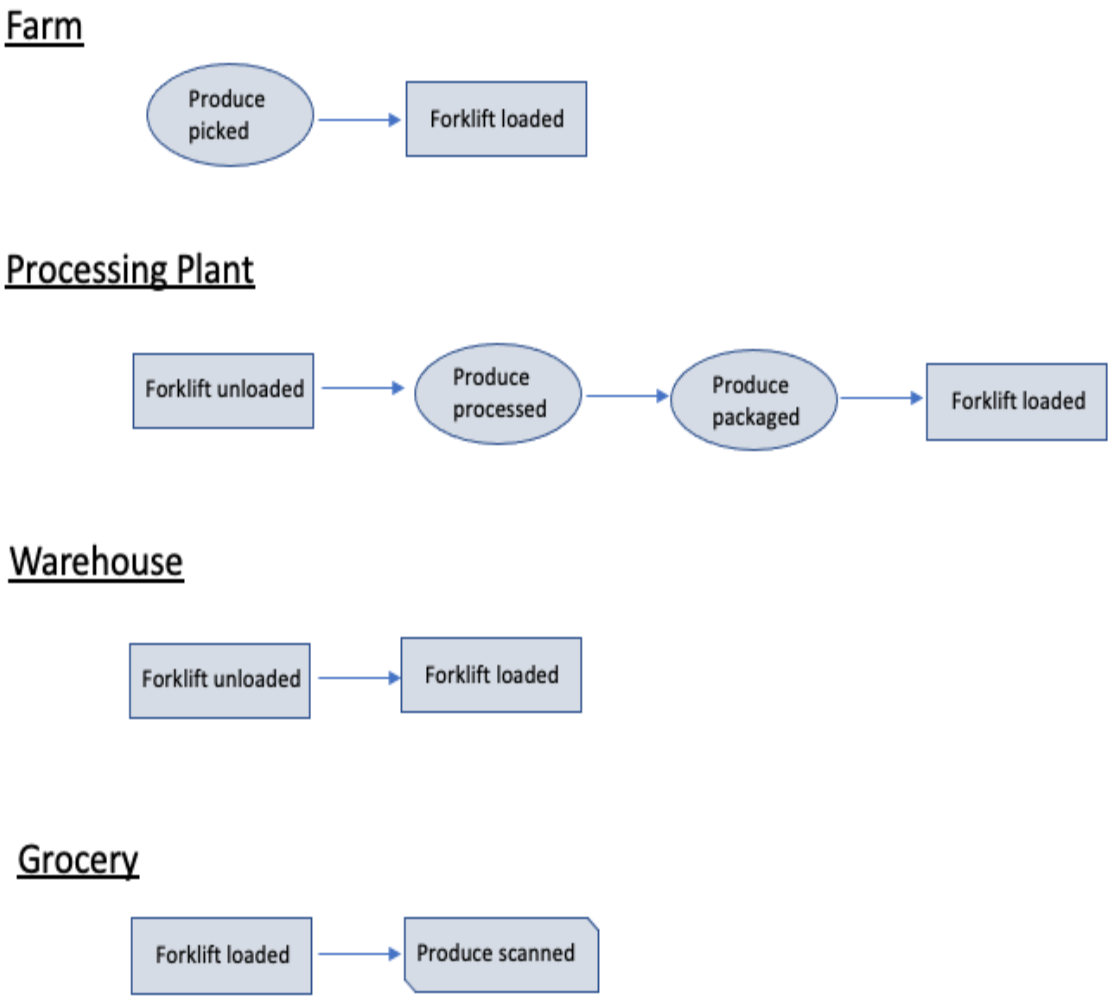


Figure 2-13. The real-world events that make up the product workflow.

The predetermined points where the sticker was scanned join the points in the workflow, and the people who were authorized to approve or reject the scanned-in information are part of the governing group that is signing off the smart contract that's moving through the workflow. This is recorded on blockchain, along with the information for that transaction. In this case, the transaction has two parts: the approval of the information and then the consent to move forward to the approval process. Table 2-8 shows how this breaks down in terms of participants, assets and transactions.

Table 2-8. Example participants, assets and transactions for a supply chain

Functionality	Hosting and/or Storage	Participants	Assets	Example Transactions
Supply chain application	Cloud computing instance	The farmer, the processing plant, the truckers, the grocery store.	Unit of produce (crate/pallet).	Movement of unit(s) of produ from one point the supply chai the next.

The models that control this process were tested and deployed based on data gathered from the many variables coming from the farm or farms in the project. The smooth workflow depends on the consistency of the input data for the model.

Let's first dive deeper into how to create a group that handles the governance of the blockchain.

Establishing a Governance Group

Governance encompasses the system by which an organization is controlled and operates - called *operations* - and the mechanisms by which it, and its people, are held to account - called *policy*. The group as a whole has a shared business aim, and the process by which they make decisions is a simple circular process of planning, implementing and evaluating, and then approving. Governance processes are designed to ensure:

- accountability
- transparency
- responsiveness
- abiding by the rule of law
- stability
- equity

- inclusiveness
- empowerment
- broad-based participation

In AI/ML, this also means the group is providing visibility and automated documentation throughout the AI lifecycle, which allows the stakeholders to analyze and make decisions with greater efficacy and enforce their own business policies.

How do you figure out what your policy and operation procedures will be? Your organization, if already established, has policies on everything from how to apply for time off to how to shut down the production servers for software upgrades. For each of these policies, your organization also has a list of steps on how to carry out those policies, called operating procedures. Each policy and operating procedure has to go through a consent and approval process; sometimes both processes are performed at the same time, and sometimes they're separate "rounds" of asking each person. It depends on the size and structure of your governance group. It could be that the upper level of management consents to the policy, and their direct reports approve or reject the policy since they'll be enforcing it among their subordinates. And if the organization is large enough, the direct reports and *their* direct reports should be approving and rejecting it in that case. In every case, everyone knows that they are voting, and what they are voting for.

This leads to the question of how do you establish a governance group, size, and structure? First you pick the people, and who you pick depends on what is being decided upon. C*ooperative functioning* (sometimes known as *sociocratic governance*) involves selecting people from top to bottom in an organization - every level of personnel is considered a stakeholder and needs to have a say in policies and operations that they directly affect and are affected by. There will be different people for different policies and procedures, but the feedback must flow throughout the organization.

In our Walmart example, the governance group would be made up of representatives from the suppliers' farms, processing plant, warehouses, transportation company, and Walmart management. Each of these would be at the level of the organization that is involved in the decision making for that policy or operation and would be able to sign off on policy or operational modification, including model validation.

Now let's say that a new supplier is approved and brought into this governance group, and the farms the supplier owns are in a different region of the country than the other suppliers' farms. A different climate means that a different set of data - changes in temperature and humidity compared to the other farms - is introduced into the established model, causing the output to be different. In terms of model validation, the change in data will change the distribution of the output. Model validators would need to modify the model and go through an approval process with stakeholders. Some members of the governance group might also be asked to provide a separate set of data to the model validators for testing purposes, and approve the new model when testing is completed, and then again once the new model is deployed to production.

Implementing On-Chain Governance

Governance does not have to be strictly manual, which usually involves notifications emailed to everyone in the governance group to let them know they need to log in and approve the next round of modifications.

On-chain governance is a system for implementing and managing changes to blockchains. This allows a project to be managed by changes to blockchain rather than occurring by email or other forms of communication. It includes rules for instituting changes that are encoded into the blockchain protocol, which would be governed automatically. Changes are proposed through code updates and each node or participant votes on whether to approve or reject the proposed change.

On-chain governance has the advantages of being a decentralized, collaborative decision-making body. Instead of leaving the code changes up

to a small group of management and/or developers, each node has a vote. Since everything takes place online, managing code changes, testing, and the voting process are more efficient. The consensus voting also significantly reduces the chance of needing a hard fork in the code. The disadvantages of on-chain governance include the fact that a large governance group is prone to lower voter turnout, and those with greater stakes can manipulate the votes.

In on-chain governance, there are steps to build consensus among the governance group and steps to gain consent from each of the members. Both procedures can be managed using smart contracts. Participants verify the information in the transaction to make sure it's accurate and that the parameters regarding the transaction have been satisfied. Once participants have completed their verification process, the results are submitted to the network. After review by other participants and consensus has been achieved, a new block is added to the blockchain network. Normally, participants would receive a notification of the transaction, typically an email. The new record appended in the blockchain also gets visible to respective participants interfaces and different actions can be performed over the record though the screen based on the given permission.

> **NOTE**
>
> Participants (miners) in a cryptocurrency transaction usually receive some type of compensation for their efforts, which is called a Proof of Work (PoW) system or process.

Carrying out the verification process is not like an informal governance system, which uses a combination of offline coordination (meetings, emails, and so on) and online code modifications to implement changes. An on-chain governance system works only online, and changes to a blockchain are proposed through code updates. Typically, on-chain governance involves the following stakeholders:

- the nodes (and the people who run them), which validate the transactions
- developers who write the algorithms
- and the participants who have a vested interest in the transactions

The participants or nodes can vote to approve or reject the proposed change. In the case where nodes do not have equal voting power, nodes with a greater financial stake will have more votes than nodes that have a relatively lesser amount. If the change is approved, it is included in the blockchain and baselined. In some instances of on-chain governance implementation, the updated code may be rolled back to its version before a baseline, if the proposed change is rejected. In all of these governance procedures, the stakeholders will all know that they are voting and what they're voting for - transparency is part of the system's integrity.

On-chain governance implementation differs between various blockchains. For example, you could have a form of self-amending ledger where proposed changes are implemented to the blockchain and rolled out onto the chain's QA version, tested, and then deployed to a production version of the blockchain. If not, the changes are rolled back. Another type of blockchain hardcodes their governing algorithm, which will then trigger either passive (changing the permissions for a participant/node) or active (quarantining sections of the network for updates) actions.

> **TIP**
>
> Regardless of which type of the blockchain you use, on-chain governance is worth the time to implement to allow the supply chain to manage their policies and operations more efficiently online instead of through lengthy meetings and tedious sign-off procedures involving emailing and faxing documents.

Developing Compliant Intelligent Agents

There is no point in time in the process of creating, testing, deploying, and auditing production AI where a model can be "certified" as being fully

compliant with some set of rules, or entirely free from risk. There are, however, a host of methods to thoroughly document and monitor agents throughout their lifecycle to keep risk manageable, and to enable organizations to respond to fluctuations in the factors that affect this risk. At question here are two factors:

- the extent to which independence is exhibited by an AI
- its level of expertise - its ability to assimilate and adjust to an environment by means of user requests (through smart contracts) and available assets.

Software agents (also called daemons) are already a standard part of most non-AI software programs: agents run in the background, polling (listening) for some request from some software program.

When the software agent receives a request, it dutifully executes whatever is requested so long as it fits its given rules. The rules are not always hard-coded - often they are passed to the agent in the form of variable perimeters - but the agent doesn't really do any thinking, reasoning, or learning; it either carries out the function or returns an error.

One of the most commonly used types of agents is a mail transport agent (MTA). The MTA runs on the email server, listens for incoming messages, and routes them according to the email address contained in the message wrapper. If the mail is going to another domain, the MTA talks to another MTA (via Simple Mail Transport Protocol, or SMTP) and so on until the destination domain is reached and the message is delivered to a user mailbox by a mail delivery agent (MDA).

Intelligent agents (also called *bots*) are software agents driven by AI. A standard MTA can adjust its routing based on network traffic. However, when you add AI to an MTA, you add the dimension of ML, which means that the MTA can be trained in advance on how to operate best from its own environment and location, balanced against the experience of other MTAs. The ML training data might include the logs of millions of other MTAs and the response from the MTA's own routing attempts. The intelligent MTA

might then be capable of accurately predicting how to meet its assigned goals (fastest or most reliable? lowest cost?) for routing messages, and changing its behavior based on its predictions.

When an intelligent MTA receives a request, at first it might be just like a regular agent. It picks up the request, verifies that the request fits its rules, and if it does, the agent processes the request. Otherwise it returns an error. This is where the behavior of an intelligent agent deviates - if part of its intent is to operate “error free” - then, after getting the error, an intelligent agent might decide to adjust something about its own behavior to avoid getting the error next time. It may investigate the answers to this problem by conducting experiments or research, including learning from other intelligent agents.

If the MTAs and MDAs on other mail servers are also intelligent, then the agents can better learn from one another and perhaps even exchange new methods. When you think about all MTAs and MDAs learning in massive parallel across all mail servers on earth, mail routing could evolve very quickly. When you add program synthesis, the intelligent agent that we depend on the most could evolve beyond any human programmer’s knowledge, because it has modified itself with no audit trail. Without any way to detect the origin of the intelligent MTA or method of finding out what it has been learning and doing, the intelligent agent could slip out of control and become a superintelligent agent.

A network of superintelligent MTAs could make catastrophic changes - hypothetically, given the sole goal of speedy delivery, the superintelligent MTAs could determine that message encryption makes message transport too slow and bypass encryption to speed things up. As a result, every mail message in the world that was handled by these MTAs would become easily intercepted - with a total loss of email privacy - and mail administrators would have a colossal mess on their hands, without much recourse.

By adding blockchain to the technology stack of the superintelligent MTAs, you are adding a requirement for the superintelligent agents to report back, based on your previous rules. You can add rules for when the MTA might

have to stop taking advice from other superintelligent agents or when it has to go out of service for retraining (for too high a percent of failed deliveries or leaked messages, for instance). You could specify in the model's fact flow that it has to record its own code changes on blockchain, along with any significant changes in behavior, in a human-readable form. You could make blockchain part of the superintelligent agent's DNA by coding it to fail if blockchain is not available.

When thinking about making intelligent MTAs and other intelligent agents compliant with organizational policies (for example, "all email messages should be private"), there's a checklist of variables that need to be accounted for in order to manage the information and the risks. Effective risk management is a continuous process. As such, it is critical to have a *model inventory*, or *fact sheet*, that's easily accessible to all relevant personnel, especially when you're using multiple algorithms in multiple agents. Even if you're using only one model and one agent, changes to models or underlying data or infrastructure, which commonly occur over time, should also be easily discoverable. Some changes should generate specific alerts that are handled by personnel or possibly by an on-chain governance process once the alert is triaged by smart contract.

Some helpful tethers you can include on your factsheet and blockchain touchpoints include the following:

- Objectives for this intelligent agent
- Risks to avoid and unaccounted for risks
- Assumptions behind the intelligent agent
- Methodologies behind the intelligent agent
- Dependencies, and how they have been taken into account
- How to access or recreate training data
- Notes on any tradeoff between accuracy and explainability
- Expected life cycle for this model

- Review triggers
- Procedure for feedback from consumers or authorities
- Procedure for recall
- Responsible parties and contact information

> **TIP**
>
> If you architect on-chain governance into your software, as discussed earlier in this Chapter, you will be able to get far into any ML troubleshooting process by automating the ability to display human-readable results to any user that has permission, helping the user to trust the system.

Blockchain Control 4: Show Authenticity Through User-Viewable Provenance

Blockchain provides traceability of all important events that took place in a system. A consumer of AI should be able to test AI to determine whether they trust it before deploying it for their purpose.

Control 4 will be especially important in using branded AI that has underlying components that come from distributed marketplaces. A breakdown of participants, assets, and transactions for consumer-facing touchpoints is seen in Table 2-9.

Table 2-9. Consumer touchpoint by participants, assets and transactions.

Functionality	Hosting and/or Storage	Participants	Assets	Example Transactions
Consumer Touchpoint	varies; person or agent using variety of devices or scripts; AI could be embedded in a product	B2B or B2C consumer, AI agent, project stakeholders	reviews and ratings, help inquiries, reports, complaints	checks to see if embedded AI is safe before usi complains abou unruly behavio end-product

Ways a Consumer Can Decide if to Trust AI or Not

When planning ways that the consumers of your AI can decide whether or not to trust it, think in terms of the lock on your web browser when visiting a secure site using *https*. The end user only sees the lock or they don't, and they will likely notice a warning. A user support tech troubleshooting a warning for the end user can drill down for more information and find out whether a certificate is valid, when it was issued, and information about the issuing certificate authority. This type of idea, of different consumers needing different levels of proof, may help AI become worthy of trust in all stages of its lifecycle. In Table 2-10, you can explore various consumer types, the type of trust they might need, and examples of how the trust can be proven to them in a form they can comprehend.

Table 2-10. Typical consumers of AI, the type of trust needed, and what proof might look like.

AI Consumer Type	Type of Trust Needed	Example Proofs
End-user	Overall assurance done via affiliate organization that is a roll-up of all trust organizations involved in the creation, assembly, training and deployment of the AI	Check to make sure that proof logo is intact, showing that contributors are acceptable by trust organization
Broker or Marketplace	Trust logos of all their member organizations and those of their dependencies	Check to make sure that proof logo is intact, that AI works in certain conditions and situations, and that components are compatible with one another
Architect	Trust of all brands selected for AI architecture	Check to make sure that all factsheets of brands supplying components match company policy
Developer	Trust of algorithm libraries and development environments, as well as trust logos of data scientists, along with trust of architecture	Check to make sure that all suppliers meet policy and that any bugs or issues are surfaced, and that support is available should issues arise
Data Scientist	Trust of data libraries and data tests	Check to see if data meets policy for readiness, and that biases have been removed
Chief Technology Officer	Trust in output of AI, trust in provenance, trust in ongoing control of AI	Check to see if adequate testing is being done to make sure model stays true to its intent as model learns and morphs
IoT devices	Trust in security and compatibility	Check for compatibility and recalls
Intelligent agents	Trust of other intelligent agents and marketplaces for intelligent agents and data	Checks the reputation and credibility of other agents before accepting their input for recursive learning

NOTE

Keep in mind that some people will want to just take your word for things and others will want to drill down for proof - this will help you to foster appropriate levels of trust in your system.

Summary

Blockchain helps to keep AI in check and under human control when applied to AI. Though it may seem overwhelming at first because AI is so vast, making sure the underpinnings of the system are locked down in a single source of truth could be key to unraveling problems in the future.

As you work through planning your own use case and fact flow system, remember to refer back to the four controls for a systematic way of thinking about how blockchain impacts each part of AI's lifecycle.

In Chapter 3 you will learn about several types of AI user interfaces, and how to implement controls along with the interfaces.

Chapter 3. User Interfaces

> **A NOTE FOR EARLY RELEASE READERS**
>
> With Early Release ebooks, you get books in their earliest form—the authors' raw and unedited content as they write—so you can take advantage of these technologies long before the official release of these titles.
>
> This will be the third chapter of the final book.
>
> If you have comments about how we might improve the content and/or examples in this book, or if you notice missing material within this chapter, please reach out to the editor at *ccollins@oreilly.com*.

Before creating your own blockchain-tethered AI (BTA) that incorporates blockchain tethers for the four AI controls from Chapter 2, consider how the user interface (UI) of the BTA you create can make or break the system's adoption. Often, people using a new system will be pleased so long as the system has a good look and feel, before they even dig into functionality. Conversely, difficult-to-use or disjointed interfaces will give the user a poor first impression, which is difficult to overcome.

If you have already started to test the sample code that is available for Hyperledger Fabric, you will find that while the examples are great for learning to code smart contracts, the UI for the application you generate from the sample code is very rudimentary. Most users would reject this type of an interface and you couldn't deploy it in an enterprise, since it is not even as sophisticated as commonly used personal systems like social media, banking, and online stores.

Sometimes, it doesn't make sense to develop a new UI, such as when the system is only supplementing functionality performed by another system (like a continuous integration system that deploys approved trained

models). In this case, you can do the integration in such a way that the two systems communicate via API, so the remote system causes a trail to be recorded on blockchain, completely transparent to the user. You may have other users accessing the same blockchain with a custom interface like a smartphone app that has other functionality, such as the ability to track and trace the deployments. Even then, you will still likely need to build a UI so you can administer the system.

Design Thinking

Just like when you plan a road trip, you need to know where you are heading, how long it will take you to get there, and what it will cost. It is the same when you are starting a new system from scratch, such as a BTA. Before you write the UI to your BTA, you need to first identify who will use it. Remember the participants, assets and transactions from Chapter 1? The participants, or people who interact with your BTA, are the primary focus for building your UI, as well as for gathering your overall requirements.

Primary users of the BTA project will be any organization who wants to write/train/test/deploy their AI model in a blockchain based system that gives an authentic blockchain based report about the history of the AI model like who trained, who reviewed and how, what was the dataset, and so on. Each user utilizes two kinds of interfaces i.e. an interface to train their AI model like Oracle cloud based Jupyter notebook and the BTA application where the user can upload/review the AI model.

Requirements gathering should be done before you start writing your UIs. *Design thinking* is a process in which you use a set of techniques to cull from the user group what each of them needs from the system, what devices they use to interact with it, and how they would like it to look and feel, which in turn, gives you your detailed requirements.

> **TIP**
>
> As an alternative to holding a design thinking workshop for the group, you can encourage the project sponsor to share design mockups, process flow diagrams, use case diagrams and flowcharts with the group before any coding begins, and gain their feedback. An easy way to do this is to record videos of each thing that you want to get feedback on, and drop them into a shared folder for the users. This way the group can respond without being in the same place at the same time.

A bonus of design thinking is the people using the new system have buy-in because they helped to design it. Once you have finished analyzing the information you gather from a design thinking workshop, you should have most of the boxes checked to be ready to code.

To learn more about Design Thinking techniques, you can start by looking at *https://www.ibm.com/design/thinking/*. The main stages of design thinking are:

1. **Empathize:** Research your user's needs either by interview, survey, statistics and focus groups. The BTA project has passed this stage by interviewing experts of AI and blockchain, following statistical data.
2. **Define:** In the Define stage, you will manage the information collected from the Empathize stage that helps to understand the pain and problems of an organization and create problem statements. One of the problem statements should focus on the problem of AI models that can be dangerous and harmful if the whole developing/testing/training/deploying process is not tracked properly and built to maintain the originality of the data set.
3. **Ideate:** Understand problems and create ideas. After collecting customer information from Empathize stage, our team has collected the problems having in the AI space which is analyzed by the team, brainstorming the problems, finding feasible solutions and adopting the most efficient one to solve the problem. The BTA team discussed the problems together with the concerned people and also discussed the ideas and solutions.

4. **Prototype:** The feasible solutions found in the Ideate stage are now prototyped. There are many prototyping tools available like Adobe XD, Balsamiq, and so on. However, we have used Google Spreadsheets to create mockups so that we can play with data well being based on real scenarios. The prototypes can be found above under “Support Level Interface”
5. **Test:** Being based on the prototype, the product is created and it goes under testing. The product is tested to verify that the solutions created against the identified user problems. The BTA project is tested in local and beta development environments to verify the output approved by the AI engineer. Finally, the AI model is launched into production which has a particular monitoring feature that helps to improve the product by looping back to the design thinking stages.

Enterprise requirements are not normally obvious. Product managers need to go beyond the normal understanding of the real requirements that resolve the business problems. In enterprise design thinking there is the concept of observing, reflecting, and moving together with the whole team. It’s not only the product manager who is engaged in the requirement collection - the whole team becomes engaged with customers. It starts with asking *why*, and the solution of the problem lies in the answer of *why*. Maybe the manager needs to ask *why* several times, at least five, to reach the solution. It is a collaboration between the design team and the business users, where the work is completed and reviewed with the end user and again gets put back in the loop for the adjustment.

The team can also use a sponsor user to verify the solution. A *sponsor user* is an expert in the field. For example, if somebody is writing software for supply chain management (SCM), then the created solution should be reviewed by SCM and adjusted based on the sponsor user’s suggestions.

Web Interfaces

Building a robust web interface as the interface to your BTA is a good way to leverage standards that your users already know how to use. Just about

everyone has used a web interface for something. It could be a personal use like social media or banking, or a work use like filling in forms for human resources or a sales workflow. Many people use web interfaces so often that the basic functionality, and often troubleshooting, comes natural to them.

On YouTube, as part of a Hyperledger Fabric demo, there's a straightforward illustration of how blockchain works, where a marble is transferred from one person to another, and a record is added to the blockchain for each transaction. The screen shows all the users, the marbles each user possesses in their marble bag, the marble moving from bag to bag, and the transaction record. Every display is the same - no difference in appearance to account for different permissions or roles, because everyone was the same type of user for this example.

In another example video, you can view the lifecycle of a car from manufacturer to owner to scrap metal dealer, with transactions saved on blockchain. There are seven unique participants that log in with their own private key. Each participant registers themselves on the platform; when the platform accepts the registration, it generates a private key for them to use when they log into the system. The system's framework is Hyperledger Fabric Explorer, which is used to monitor all of the transactions that are saved on the blockchain through the consensus of each participant. Logging in as any of the participants reveals the same basic framework and very similar visuals, with small differences in functional options based on the user's role - but still, very similar.

A police officer or an insurance company can also log in and check the history and provenance of the car from the information on the blockchain. The owner of the car is in full control of the information of their car.

The basic framework and user interface of this system is the same for each participant - they can all see the information for each car that's been saved on blockchain. What changes on the interface is the information they can edit.

In real enterprise situations, it's most efficient to create a basic framework of an interface with all of the possible modules that serve every type of user

logged into the system - and then turn modules on or off to the user depending on a set of permissions for their particular role. You can create even more customization of the interface functionality-wise by having sub-permissions for particular actions within the role. The top level functions are available from a menu bar, and specifics are accessed from within the function. Blockchain records every transaction. Logging in as a dealer, the user has permission to see the inventory of the manufacturer, select a car, see all of its blockchain-verified information.

For your BTA design, the basic web interface framework is the same for all roles. All of the actions, workflows, and configuration menus are selected from a sidebar, and the functions that are displayed depend on the role of and permission level for that user. The profile and password change functions are found in the top menu bar for all roles.

Blockchain Tethered AI User Interfaces

Since the user interface is how system participants will interact with your BTA, it is critical that stakeholders, AI/MLOps Engineers, support specialists, business users, and the general public are all considered when designing it.

Since the BTA you are building is a multi-user, role-based permission system, you can offer different features in the user interface depending on who logs in. You could also make the interface be a smartphone app or you could design your BTA to interoperate via API from some other system using its interface, such as the control panel of an automated vehicle.

Consider the role-based UI mockups shown in Figure 3-1 and Figure 3-2 that you will work from in Chapter 5 as you build your BTA, and think about how they might be implemented.

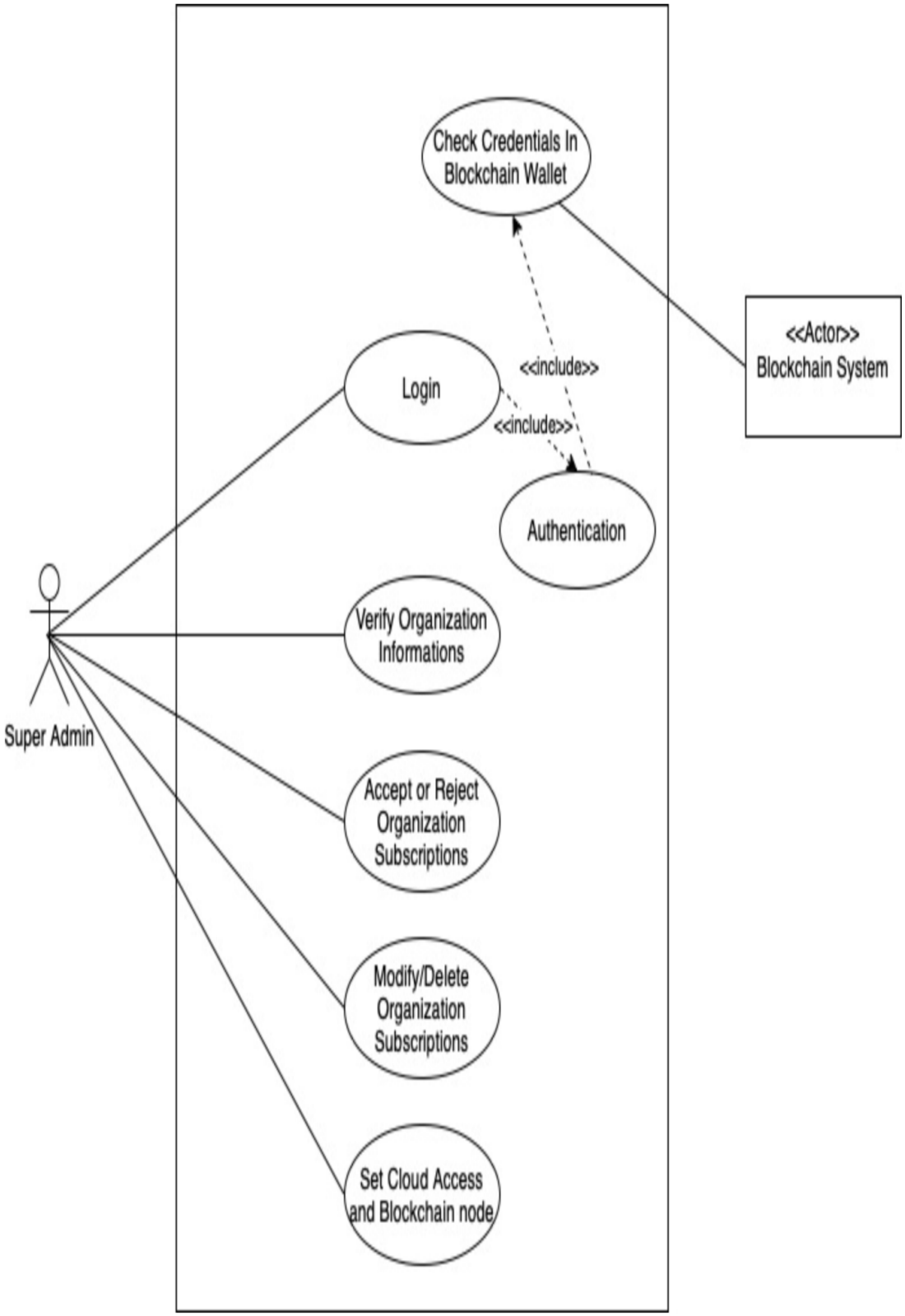

Check Credentials In Blockchain Wallet
<<Actor>> Blockchain System
Login
<<include>>
<<include>>
Authentication
Verify Organization Informations
Super Admin
Accept or Reject Organization Subscriptions
Modify/Delete Organization Subscriptions
Set Cloud Access and Blockchain node

Figure 3-1. Role-based user interface mockup of the super admin permissions in BTA.


Check Credentials In Blockchain Wallet
<<include>>
<<Actor>> Blockchain System
Login
<<include>>
Authentication
Request Project Subscription
Create Hash In Blockchain Wallet
<<include>>
Company Admin
Create Project
<<include>>
Create Users
Setup Basic AI Model
Test and Train AI Model
Verify AI Model
AI Engineer
Configure and Tune AI Model
Submit Verified AI Model
Send Notification to AI Engineer
Verify AI Model
Deploy AI Model
MLOps Engineer
Mnage Monitoring Report
View History of AI Model
View Monitoring Report
Review AI Model
Stakeholders
Give Feedback
Track Logs of System
<<Actor>> Blockchain System

Figure 3-2. This shows participants of the BTA project together with the tasks they can do within the project. It has used a blockchain platform to store each step and participants identity which later can be produced as a blockchain report.

Access Levels

Different participants get different levels of access based on their responsibilities. As shown in Figure 3-1 and Figure 3-2, BTA has five types of users:

1. Super Admin: Admin user of the project has access to verify and approve the organization willing to participate in developing the BTA project. Super admin can assign blockchain a node and an Oracle bucket to the Organization admin.
2. Organization Admin (OA) : OA is the admin of the organization which is verified by super admin, has access to create roles and responsibilities and assign them to the right user. OA can also create AI projects and can monitor the progress.
3. AI Engineer (AIE): AIE is responsible for developing the AI model in the provided platform and submitting it for review by the reviewer.
4. MLOps Engineer: MLOps engineer gets access to review the submitted AI model, deploy and launch in the production environment.
5. Stakeholder: The user gets access to review the deployed AI model, monitor the production based AI model and provide feedback.

Each feature and its access are kept in the staffing section of the project so OA can assign the feature and its related access to the right user.

Support-level Interface

Ideally, the UI should hide the application's technical implementation, such as how the blockchain works, from the general user. It should provide utilities to investigate details of the implemented model and underlying data used to train it. A good interface will allow the user to interact with the

application and consume its outputs in an intuitive way. This may mean borrowing existing, similar user experiences from consumer applications that the user is already on.

A good web interface is critical to using both the tools to build AI as well as for the product you are building. This web interface would be the one that the super admin views when they log in. Their login is the same as everyone's but once in, they can see every aspect of the application. Their functionality includes the ability to disable or enable modules, add users and control what roles and permissions they have in any module, generate reports, change certain parameters, generate reports, monitor every aspect of the input and output that engineers and stakeholders are monitoring, and in general, modify every field that is not hardcoded. It should be noted that this should not be done without checking with the Data Engineer or MLOps Engineers first.

You can create multiple user roles in your BTA.

Super Administrator

Super Administrator (SA) is likely used by you and your support team. As shown in Figure 3-3, this is the user that can approve the creation of organizations and their administrators.

Super admin profile info

Name	Deepak Bhatta
Email	deepak@myaitest.com
Phone No	XXXX XXX XXX

Unverified Organization

Organization Name	Address	Email	Phone	Action
Fast AI	395 Whaley Lane,City, Milwaukee, Wisconsin, 53203	john@myaitest.com	262-436-6776	Verify
Green AI	652 Orphan Road, Elk Lake, Wisconsin, 53212	josh@greenaitest.com	715-872-3126	Verify

Verify

BC node info	https://x.x.x.x/kilroybc/2324324
BC channel Name	Org-admin
Oracle Bucket URL	https://x.x.x.x/kilroybucket/43sdfsd4df

Verify

Figure 3-3. : Super Administrator User Interface Mockup

This role is intended to be held by the organization that manages the BTA, not the projects. By keeping this role completely separate from the organization administrators, you can manage multiple organizational tenants per instance of your BTA.

For super admins, once they log into their super admin panel, they will see menus giving them:

- The ability to Verify, and Accept or Reject organization registration and subscription requests
- Organization information details
- The ability to modify information for registered organizations
- The ability to delete registered organizations

Organization Administrator

Organization Administrator is the administrator of the registered organization within the BTA application so the user can create projects, add users and roles to the project and can see the details of the projects and AI models. As shown in Figure 3-4, this functionality is standard for multi-user systems.

	A	B	C	D	E	F	G	H	I
1	**User Profile**								
2	Name	John Doe							
3	Email	john@john.com							
4	BC node info	https://x.x.x.x/kilroybc/343dsadas3							
5	BC channel Name	AI-engineer/MLOPs							
6	Oracle Bucket URL	https://x.x.x.x/kilroybucket/343dsadas3							
7									
8	**Crate Project**								
9	Project ID	#1234							
10	Project Name	Traffic Signal Detection (TSD)							
11	Project details								
12	Project members								
13		*Participants ID*							
14		*Participants Type*	*DS, Fusion Manager, Stakeholder*						
15		*Organization Name*	*current or another org name*						
16	Project Domain	*Finance/Ecommerce..*							
17	Log file template	Upload field							
18									
19	**Project/AI Model Grid**								
20	**Participant Name**	**Project ID**	**project name**	**Problem defin**	**Project Purpose**	**project domain**	**Latest Version**	**Status**	**Action**
21	John Doe	#10001	TSD	Classification		Transportation	V5	Pending	View || New version || Edit
22		V1	Exp deails	status			V2	Processing	VDetail || Reviews || Monitoring Report | BC history
23		V2	Exp deails	status			V3	Accept	VDetail || Reviews || Monitoring Report | BC history
24		V3	Exp deails	status			V4	Decline	VDetail || Reviews || Monitoring Report | BC history
25	Mike Will			Regression		Ecommerce		Pending	View || New version || Edit
26	Note: Only stakeholder should be able to add project purpose either by free text or document								
27	A project can be only started when at least one AI Engineer, one MLOPs Engineer and one Stakeholder are created.								

Figure 3-4. Mockup of Organization Administrator UI

User management functionality for your BTA could also be integrated with your existing user management systems such as LDAP.

For organization admins, once they log into their organization admin panel, the menu will allow them to:

- Create an organizational units and sub-units within an organization
- Add people to specific organization roles inside the units and subunits
- Manage permissions for any organizational role
- See all of the blockchain activity

AI Engineer

The AI Engineer (AIE) creates an AI model, writes scripts, tests, trains and submits the model for the review. The AIE submits all train and test data set to the reviewer, i.e. the MLOps engineer. Some helpful metrics are shown in the UI mockup in Figure 3-5; you may find that you want to include additional details in your BTA.

User Profile

Name	Mike Smith
Email	mike@myaitest.com
BC node info	https://x.x.x.x/kilroybc/343dsad34s4d
BC channel Name	AI-engineer
Oracle Bucket URL	https://x.x.x.x/kilroybucket/343dsapmid893

Project Grid

Project ID	Project Name	Problem defination	Project Purpose	project domain	Action
#10001	TSD	classification	View Doc	Transportation	View || New version || Edit
TSD-V1	Review Passed				VDetail || Reviews || Monitoring Report | BC history
TSD-V2	Deployed				VDetail || Reviews || Monitoring Report | BC history
TSD-V3	Review Failed				VDetail || Reviews || Monitoring Report | BC history
#10002	Sales Predection				

Version Details

Project Name	TSD
Created By:	Mike Smith
Version	TSD-V1
Submitted Date	06-27-2022
Reviewed Dated	06-28-2022
Production Date	06-29-2022

Exp ID/No	Date	Detail
exp1	06-27-2022	Experiment Detail
exp2	06-28-2022	Experiment Detail
exp3	06-29-2022	Experiment Detail

Experiment Detail

Project Name	v1				
Experiment version	exp1				
Algorithm					
Code version					
Code repo link		Parameters (comes from model architecture)		Performance metrics (Test)	
NB(NoteBook) version	Python 3.6. 9	depth	64	test_accuracy	0.9751999974
Train dataset link		layer_number	10	test_f1_score	0.9753184915
Test dataset link		filters_numbers	3	test_loss	0.0828124508
framework	PyTorch Lightning	etc		Roc_curves	
framework Version	1.6.4			confusion_matrix	
Log file link				RMSE	
				etc	

epoch	train_acc	val_acc	training_loss	val_loss	train_f1_score	train_precision	train_re
0	0.4719435246	0.2357966907	0.7348255898	0.4719435246	0.5059391088	0.1790594343	0.71698
1	0.4719435246	0.2357966907	0.7348256898	0.4719435246	0.5059391088	0.1790594343	0.71698
2	0.4719435246	0.2357966907	0.7348255898	0.4719435246	0.5059391088	0.1790594343	0.73579
3	0.4719435246	0.2357966907	0.7348255898	0.4719435246	0.5059391088	0.169889957	0.73579

New Version form

Version	
Log file location	Give log file location
Log File version	V1
Model	Give Model location
notebooks version,	
train datasets,	Give Model location
test datasets,	Give Model location
Code repo	
Code version	
Comment	

Review The Model

Genral Info

Version	V1
Log file location	http://ml.oracle.com/...
Log File BC Hash	dkfjds324wkjsdf9sdfdfdsfdsfs
Log File version	V1
Model	http://ml.oracle.com/model/
Model BC Hash	fhsfs9oaduasdjlasjdlasdsadasdasdas
Notebooks version,	1.7
Train datasets,	http://ml.oracle.com/traindataaset
Traindataset BC Hash	flkjsdlfs9sdfsdjfsdfdfgdf34rwerefy6gdrf3sqwq,dflef
Test datasets,	http://ml.oracle.com/testdataset
Test dataset BC hash	dksadasd80kasldask
Code repo	http://git.com/michael/project-name
Code version	v1
Comment by DS	Please follow this instructions while testing the model

Reviews

Status	Deployed
Rating	* * *
Deployed URL	beta.runbta.com/v1/dasdasdas
Production URL	runbta.com/v1/2323343
Comments	
Documents	
Created At	6/29/2022 4:37:26
Created By	David Well
Staffing	MLOPs Engineer

Status	Reviewing
Rating	* * *
Deployed URL	beta.runbta.com/v1/dasdasdas
Production URL	runbta.com/v1/2323343
Comments	Reviewing the submitted model
Documents	
Created At	6/29/2022 4:37:26
Created By	David Well
Staffing	MLOPs Engineer

Figure 3-5. AI Engineer UI

This UI can be split out into different roles, depending on how your particular organization plans to use your BTA.

For BTA, the user interface of the AI Engineer role is built for creating and modifying algorithms. The AI Engineer will start by logging into the dashboard with their login credentials. There will be a drop down menu where the AI Engineer selects the Machine Learning lab. They will be able to select from a list of assigned projects, and view their details. After selecting the particular project, the AI Engineer should be able to create a notebook session and open it in the cloud, where they can write the code for AI model building, training, and testing. They can register the logs of each experiment, as well as the training, testing, and validation metrics to a tensorboard logger by following a standard logging mechanism. All from one interface.

Besides the standard profile and password information, the following information is saved on blockchain after each iteration:

- Project information
- Newly created model and its version
- Experiments under each model
- Epochs under each experiment
- Cloud bucket URL
- Blockchain node
- Log file location from Oracle bucket
- AI artifact information
- Start and finish timestamps for each iteration of an epoch

MLOps Engineer

The key role of MLOps Engineer is to review the AI model submitted by AI engineers, deploy and finally launch it to production. During this cycle the MLOps engineer needs to change the status of the model based on the output received from the model. The MLOps engineers compare the accuracy and output submitted by the AI engineer with the output they get during the test of the model. The verification is done using output data extracted from blockchain. If the model performs well then it is pushed for the production.

User Profile

Name	David Wells
Email	david@myaitest.com
BC node info	https://x.x.x.x/kilroybc/sdfsd353sdfsdf
BC channel Name	MLOPs-Engineer
Oracle Bucket URL	https://x.x.x.x/kilroybucket/sdfds423432

Project Grid

Project ID		project name	AI Engineer	Version	Status	Details
#10001		TSD	Mike Smith	TSD-AI-Model-V1	Deployed	VDetail | | Reviews | | Monitoring Report | BC history
#10001		TSD	Mike Smith	TSD-AI-Model-V2	Review Fail	VDetail | | Reviews | | Monitoring Report | BC history
#10001		TSD	Mike Smith	TSD-AI-Model-V3	Reviewing	VDetail | | Reviews | | Monitoring Report | BC history
#10001		TSD	Mike Smith	TSD-AI-Model-V4	Pending	VDetail | | Reviews | | Monitoring Report | BC history
#10002		Sales predection	Rock Hills	AI Model-V1	Processing	VDetail | | Reviews | | Monitoring Report | BC history

Version Details

Review The Model

Project Name	TSD
Created By:	Mike Smith
Version	TSD-V1
Submitted Date	06-27-2022
Reviewed Dated	06-28-2022
Production Date	06-29-2022

Exp ID/No	Date	Detail
exp1	06-27-2022	Experiment Detail
exp2	06-28-2022	Experiment Detail
exp3	06-29-2022	Experiment Detail

Exp Detail

Project Name	v1				
Experiment version	exp1				
Code version					
Code repo link		Parameters (comes from model architecture)		Performance metrics	
NB(NoteBook) version	Python 3.6. 9	depth	64	test_accuracy	0.9751999974
Train dataset link		layer_number	10	test_f1_score	0.9753184915
Test dataset link		filters_numbers	3	test_loss	0.0828124508
framework	PyTorch Lightning	etc		Roc_curves	
framework Version	1.6.4			confusion_matrix	
Log file link				RMSE	
				etc	

epoch	train_acc	val_acc	training_loss	val_loss	train_f1_score	train_precision	train_recall
0	0.4719435246	0.2357966907	0.7348255898	0.4719435246	0.5059391088	0.1790594343	0.7169889957
1	0.4719435246	0.2357966907	0.7348256898	0.4719435246	0.5059391088	0.1790594343	0.7169889957
2	0.4719435246	0.2357966907	0.7348255898	0.4719435246	0.5059391088	0.1790594343	0.7357966907
3	0.4719435246	0.2357966907	0.7348255898	0.4719435246	0.5059391088	0.169889957	0.7357966907

Monitoring Report

Project name	CSD	
version name	V1	
DS	Surya	Add New Monitoring Info

Date	2022-5-31
Title	New Traffic image is not working as per expectation
Desc	Detail
Documents	Multi documents attachment
By	
Staffing	

Date	2022-6-1
Title	New issue
Desc	Detail
Documents	Multi documents attachment
Staffing	
By	

Add New Monitoring Info

Project name	CSD
version name	V1
DS	Surya
Title	
Desc	
Documents	

Review The Model

Add Review

Version	V1
Log file location	http://ml.oracle.com/...
Log File BC Hash	dkfjds324wkjsdf9sdfdfdsfdsfs
Log File version	V1
Model	http://ml.oracle.com/model/
Model BC Hash	fhsfs9oaduasdjlasjdlasdsadasdasdas
notebooks version,	1.7
train datasets,	http://ml.oracle.com/traindataaset
traindataset BC Hash	flkjsdlfs9sdfsdjfsdfdfgdf34rwerefy6gdrf3sqwq,dflef
test datasets,	http://ml.oracle.com/testdataset
test dataset BC hash	dksadasd80kasldask
code repo	http://git.com/michael/project-name
Code version	v1
Comment by DS	Please follow this instructions while testing the model

Review Details

Status	Deployed
Rating	* * *
Deployed URL	beta.runbta.com/v1/dasdasdas
Production URL	runbta.com/v1/2323343
Comments	
Documents	
Created At	6/29/2022 4:37:26
Created By	David Well
Staffing	MLOPs Engineer

Status	Reviewing
Rating	* * *
Deployed URL	beta.runbta.com/v1/dasdasdas
Production URL	runbta.com/v1/2323343
Comments	Reviewing the submitted model
Documents	
Created At	6/29/2022 4:37:26
Created By	David Well
Staffing	MLOPs Engineer

Figure 3-6. : Mockup of MLOps Engineer UI

For the MLOps Engineer, once they log into their MLOps dashboard, the menu will allow them to:

- view all details of all projects, including versions of the datasets, algorithms, and artifacts
- validate and compare the performance of the models
- trade data and model feedback with the AI Engineer
- perform audits of the models
- set residual levels and monitor what triggers exceptions
- to create any report required

Stakeholders

A stakeholder might be an investor or customer or regulator who has a reason for developing the AI model. The user has permission to update the purpose of developing the model either by uploading the documents or writing free text. The user can review the deployed model but cannot change the status of the model. The user can monitor the production model and give feedback. The user only has capability to complete the project based on the output of the model.

User Profile

Name	Merry Smith
Email	merry@myaitest.com
BC node info	https://x.x.x.x/kilroybc/sdfsd353sdfsdf
BC channel Name	Stakeholder
Oracle Bucket URL	https://x.x.x.x/kilroybucket/sdfds423432

Project Grid

Project ID	project name	AI Engineer	Version	Status	Details
#10001	TSD	Mike Smith	TSD-AI-Model-V1	Deployed	VDetail | | Reviews | | Monitoring Report | BC history
#10001	TSD	Mike Smith	TSD-AI-Model-V2	Review Fail	VDetail | | Reviews | | Monitoring Report | BC history
#10001	TSD	Mike Smith	TSD-AI-Model-V3	Reviewing	VDetail | | Reviews | | Monitoring Report | BC history
#10001	TSD	Mike Smith	TSD-AI-Model-V4	Pending	VDetail | | Reviews | | Monitoring Report | BC history
#10002	Sales predection	Rock Hills	AI Model-V1	Processing	VDetail | | Reviews | | Monitoring Report | BC history

Version Details

Review The Model

Project Name	TSD
Created By:	Mike Smith
Version	TSD-V1
Submitted Date	06-27-2022
Reviewed Dated	06-28-2022
Production Date	06-29-2022

Exp ID/No	Date	Detail
exp1	06-27-2022	Experiment Detail
exp2	06-28-2022	Experiment Detail
exp3	06-29-2022	Experiment Detail

Exp Detail

Project Name	v1
Experiment version	exp1
Code version	
Code repo link	
NB(NoteBook) version	Python 3.6. 9
Train dataset link	
Test dataset link	
framework	PyTorch Lightning
framework Version	1.6.4
Log file link	

Parameters (comes from model architecture)	
depth	64
layer_number	10
filters_numbers	3
etc	

Performance metrics	
test_accuracy	0.9751999974
test_f1_score	0.9753184915
test_loss	0.0828124508
Roc_curves	
confusion_matrix	
RMSE	
etc	

epoch	train_acc	val_acc	training_loss	val_loss	train_f1_score	train_precision	train_recall
0	0.4719435246	0.2357966907	0.7348255898	0.4719435246	0.5059391088	0.1790594343	0.7169889957
1	0.4719435246	0.2357966907	0.7348256898	0.4719435246	0.5059391088	0.1790594343	0.7169889957
2	0.4719435246	0.2357966907	0.7348255898	0.4719435246	0.5059391088	0.1790594343	0.7357966907
3	0.4719435246	0.2357966907	0.7348255898	0.4719435246	0.5059391088	0.169889957	0.7357966907

Monitoring Report

Project name	CSD
version name	V1
DS	Surya

Add New Monitoring Info

Date	2022-5-31
Title	New Traffic image is not working as per expectation
Desc	Detail
Documents	Multi documents attachment
By	
Staffing	

Date	2022-6-1
Title	New issue
Desc	Detail
Documents	Multi documents attachment
Staffing	
By	

Add New Monitoring Info

Project name	CSD
version name	V1
DS	Surya
Title	
Desc	
Documents	

Review The Model

Add Review

Version	V1
Log file location	http://ml.oracle.com/...
Log File BC Hash	dkfjds324wkjsdf9sdfdfdsfdsfs
Log File version	V1
Model	http://ml.oracle.com/model/
Model BC Hash	fhsfs9oaduasdjlasjdlasdsadasdasdas
notebooks version,	1.7
train datasets,	http://ml.oracle.com/traindataaset
traindataset BC Hash	flkjsdfs9sdfsdjfsdfdfgdf34rwerefy6gdrf3sqwq,dflef
test datasets,	http://ml.oracle.com/testdataset
test dataset BC hash	dksadasd80kasldask
artifacts	http://ml.oracle.com/artificats
code repo	http://git.com/michael/project-name
Code version	v1
Comment by DS	Please follow this instructions while testing the model

Add Review

Status	Reviewing/Review Passed/Review Failed/Deployed/Production/Complete
Rating	* * * * *
Deployed URL	
Production URL	
Comments	
Documents	
Created At	
Created By	
Staffing	

Review Details

Status	Deployed
Rating	* * *
Deployed URL	beta.runbta.com/v1/dasdasdas
Production URL	runbta.com/v1/2323343
Comments	
Documents	
Created At	6/29/2022 4:37:26
Created By	David Well
Staffing	MLOPs Engineer

Status	Reviewing
Rating	* * *
Deployed URL	beta.runbta.com/v1/dasdasdas
Production URL	runbta.com/v1/2323343
Comments	Reviewing the submitted model
Documents	
Created At	6/29/2022 4:37:26
Created By	David Well
Staffing	MLOPs Engineer

Figure 3-7. : Mockup of Stakeholder UI Elements

For the stakeholders, once they login, they'll see menu options to see details of the iterations, details of the models' performance, and history of the model development. They'll be able to pass along and receive their own comments on the information. Based on feedback and reviews between the AI Engineer and the MLOps Engineer, the stakeholder will be able to purchase a model from the dashboard through the integrated ecommerce platform.

Traceability and Transparency

When considering interfaces, also think about what it will take to give users the transparency that should be present in a trustworthy model. Different users will require different levels of trust. For instance, a casual user of a budgeting system might only need an icon saying the AI is trustworthy, whereas someone rejected for a credit application by AI may want to traverse the chain of exactly what took place, and they may want to determine whether or not the decision to reject the application was based on biased AI training data collected from past unfair practices.

AI Supply Chain

Supply chains are how we receive most things we use, from toilet paper to automobiles. The supply chain has upstream steps that have to be completed in order to get the right finished product. For example, there are many different first, second and third-tier suppliers that create the many parts that are used in an automobile. A certain level of trust and traceability are expected. Blockchain lends itself well to this process, due to its tamper-evident nature, its distributed nodes, business-logic managing smart contracts, and ability to prove or disprove an event without sharing the actual data. Supply chain blockchain systems are fairly mature, and generally involve a multi-user, permissioned, cloud-based web interface as the primary user interface.

Here you can see blockchain touchpoints in a simple supply chain process vs a simple BTA. Thinking about the AI model building, training, experimentation and deployment process as a supply chain demystifies the AI black hole and makes it easier to dissect and explain to others.

Consider the process of a farm that sells eggs as local if the delivery point is 50 miles or less from the farm (Figure 3-8). Critical points like the mileage of the vehicle will need to be recorded to blockchain. Compare that to the delivery of a trained AI model (Figure 3-9), and critical points like contracted level of accuracy. In both cases, smart contracts are used to be sure that products are accepted and approved for payment only when the contracted conditions are met.

Use Case: Local Egg Delivery

 = Blockchain Touchpoints

Truck picks up eggs from farm.

 Vehicle's start mileage and details about eggs are recorded on blockchain.

Truck delivers eggs to local distributors

 Vehicle's end mileage is recorded on blockchain.

 Proof of delivery is recorded on blockchain.

Distributor pays invoice for eggs.
Smart contract test:

- Were miles traveled from farm to distributor 50 or less?
  - -> If so, pay invoice and designate eggs as local
  - -> If not, pay invoice less 20%
- Result of smart contract test sent "off-chain" to payment system to activate payment processing.

Figure 3-8.

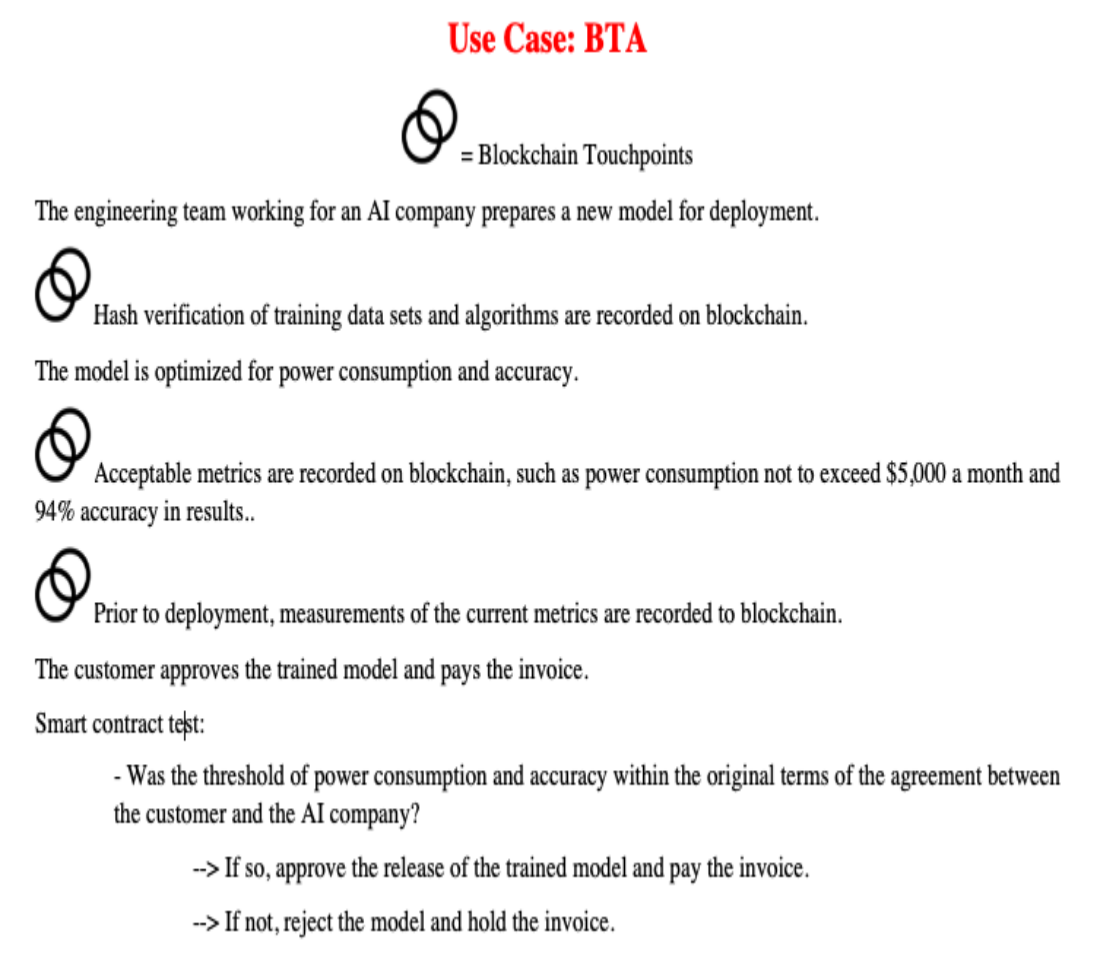


Figure 3-9.

Similarly, there is an *AI supply chain* in which many upstream people and processes are used to create the consumer-facing AI. Many AI models are pre-written and put on public websites from third-party sites. It is hard to tell where the training data came from and whether it was valid, biased or unbiased, and if the resulting models were valid or detrimental.

A data scientist might need to verify the credibility of AI supply chain participants, such an AI model supplier, before using their products. This process could be like unpeeling the onion, as layer upon layer of information is discovered, digging deep into the AI supply chain.

Consider how the supply chain of the data that is being used to train and test the models, and ask yourself these questions: Where did the data originate? Was it originally structured or unstructured? Who touched it? Did they clean it up? What methods did they use? Did they perform any tests on the data? Were there modifications resulting from test output? All of these considerations can form the data's provenance, which should be easily checked by consumers of the AI.

You can consider the same type of questions for the AI models, algorithms and application layer too. Chapter 2 covered the sort of questions to ask, such as:

- Do you have a positive identity of who engineered this system?
- What is its original intent?
- Where did you get your code?
- Where did your supplier get their code?
- If it is from a marketplace what credentials are required for a marketplace to be credible?
- What tests can be done to make sure the algorithms remain as intended?
- How the model was trained and who trained it?
- Where is such test output stored and how is its integrity maintained?
- What administrators can grant access for this code, and what process do they have for approving access to it?

A more casual user might just need to know that the AI Engineer is credible, rather than to be able to step back through the entire AI supply chain of things they don't really comprehend. In this case, an icon or trust logo can be used to let the user know that they can trust the system.

Trust logos are expected on ecommerce checkouts, and they are an important part of showing that the site can be trusted not to steal your

money. Trust logos for AI can be based on blockchain lineage and can be programmed to alert the user not only of the AI supply chain's credibility, but also if the data has undergone tampering.

TIP

Don't forget about your users once your BTA is deployed. To make your system even better to use, solicit their feedback. Set up an observation session where you watch several users perform specific tasks using the system, and then make rapid rounds of changes based on where multiple people get stuck. This can also be termed as a Monitoring tool which monitors the deployed AI model in different ways:

- Collect some feedback directly from end users of the deployed model.
- Find an AI monitoring tool or use a more popular one, like:
  - Neptune (*https://neptune.ai*)
  - Arize: (*https://arize.com*)
  - Whylabs: (*https://whylabs.ai*)

Building a web interface for your BTA is not only a great way to rapidly develop and deploy the application layer, but also a way to seamlessly include a special interface for system administrators. All of the basic considerations of designing a web application come into play, including appropriate colors, a consistent look and feel, and incorporation of standards such as the location of menus, a responsive design that works across devices and screen sizes, and consistent user management and log in/logout functionality. If you have designed your UI so most things are where someone would expect them to be and gathered their input about most other functions, you are well on your way to creating a BTA that users will like.

AI and blockchain are different entities with different flavor and power, and a web interface helps to provide their integrated interface to the end users like AI engineers and stakeholders to work in both platforms seamlessly. The web interface also helps to identify the users who are involved in the

training, testing, reviewing and finally deploying the model binding with blockchain identity.

Blockchain Explorer

As the name implies, a blockchain explorer like Hyperledger Fabric Explorer allows you to peer inside your blockchain implementation using a web browser. You can see blocks, transactions, nodes, *chaincode* (the script driving the actions taken on blockchain), the blockchain's data, and you can interact with the blockchain to add blocks or nodes. Figure 3-10 illustrates the concept. Hyperledger Fabric Explorer is a good way to troubleshoot your application layer, since you can see directly into the inner workings of your blockchain system.

HYPERLEDGER EXPLORER

DASHBOARD NETWORK BLOCKS TRANSACTIONS CHAINCODES CHANNELS global-channel

211 BLOCKS

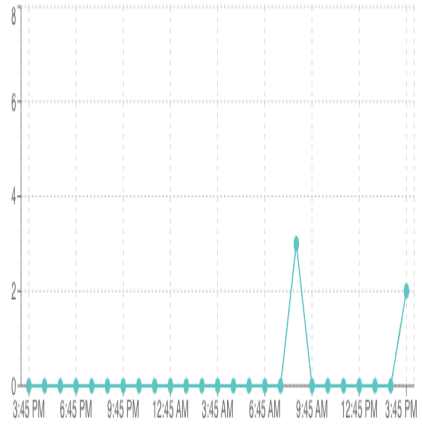
213 TRANSACTIONS

3 NODES

4 CHAINCODES

Peer Name
peer0.sh.mainnet.bta.kilroy:7051
peer0.mlops.engineer.mainnet.bta.kilroy:7061
peer0.ai.engineer1.mainnet.bta.kilroy:7051
orderer0.mainnet.bta.kilroy:7050

BLOCKS / HOUR BLOCKS / MIN TX / HOUR TX / MIN



Block 210

Channel Name: global-channel

Datahash:

348a49d8456d99502514bf5533a13ab3f44c4fe960cbbcb67327bb0aa7629a23

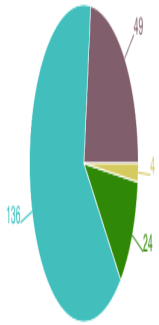
Number of Tx: 1

7 minutes ago

Block 209

Channel Name: global-channel

Transactions by Organization



PeerAIEngineer1MainnetBtaKilroyMSP

PeerMLOpsEngineerMainnetBtaKilroyMSP

Figure 3-10. HLF explorer dashboard that shows overall status and configuration of the blockchain network.

Figure 3-11 shows the network structure of the blockchain. It shows all the peers and orderers created within the network. Also listed is the MSP is created for each peer and order. This helps to know the overall structure of the blockchain network.

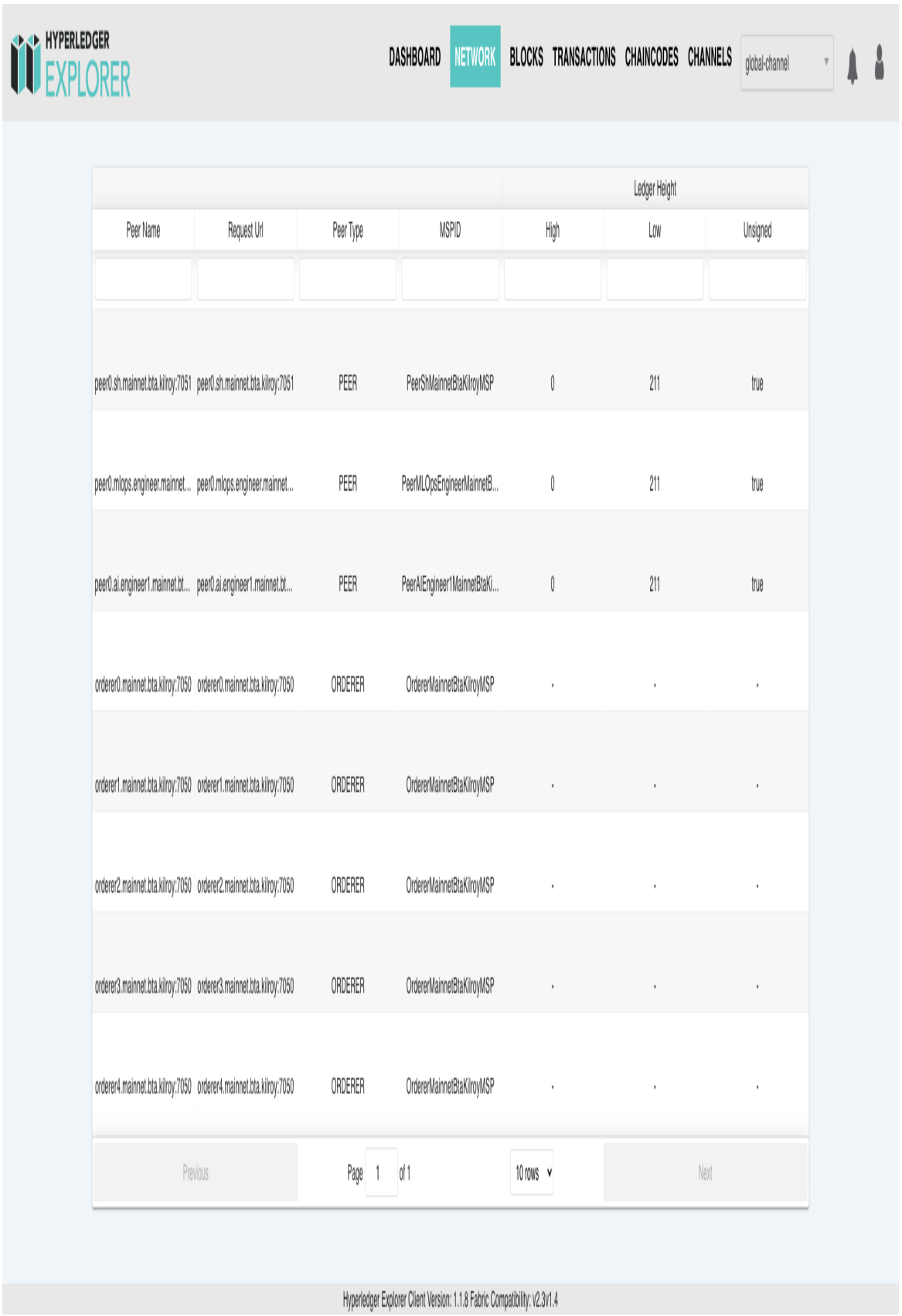

HYPERLEDGER
EXPLORER
DASHBOARD
NETWORK
BLOCKS
TRANSACTIONS
CHAINCODES
CHANNELS
global-channel
Ledger Height
Peer Name
Request Url
Peer Type
MSPID
High
Low
Unsigned
peer0.sh.mainnet.bta.kilroy:7051
peer0.sh.mainnet.bta.kilroy:7051
PEER
PeerShMainnetBtaKilroyMSP
0
211
true
peer0.mlops.engineer.mainnet...
peer0.mlops.engineer.mainnet...
PEER
PeerMLOpsEngineerMainnetB...
0
211
true
peer0.ai.engineer1.mainnet.bt...
peer0.ai.engineer1.mainnet.bt...
PEER
PeerAIEngineer1MainnetBtaKi...
0
211
true
orderer0.mainnet.bta.kilroy:7050
orderer0.mainnet.bta.kilroy:7050
ORDERER
OrdererMainnetBtaKilroyMSP
orderer1.mainnet.bta.kilroy:7050
orderer1.mainnet.bta.kilroy:7050
ORDERER
OrdererMainnetBtaKilroyMSP
orderer2.mainnet.bta.kilroy:7050
orderer2.mainnet.bta.kilroy:7050
ORDERER
OrdererMainnetBtaKilroyMSP
orderer3.mainnet.bta.kilroy:7050
orderer3.mainnet.bta.kilroy:7050
ORDERER
OrdererMainnetBtaKilroyMSP
orderer4.mainnet.bta.kilroy:7050
orderer4.mainnet.bta.kilroy:7050
ORDERER
OrdererMainnetBtaKilroyMSP
Previous
Page 1 of 1
10 rows
Next
Hyperledger Explorer Client Version: 1.1.8 Fabric Compatibility: v2.3v1.4

Figure 3-11. HLF Explorer showing network i.e. peers and their types

Figure 3-12 gives information about the transactions, like name of the peer where it is created, name of the channel used to create the transaction, and the transaction id and timestamp as well. This information is very important to understand when and where the particular record is created and who created it.

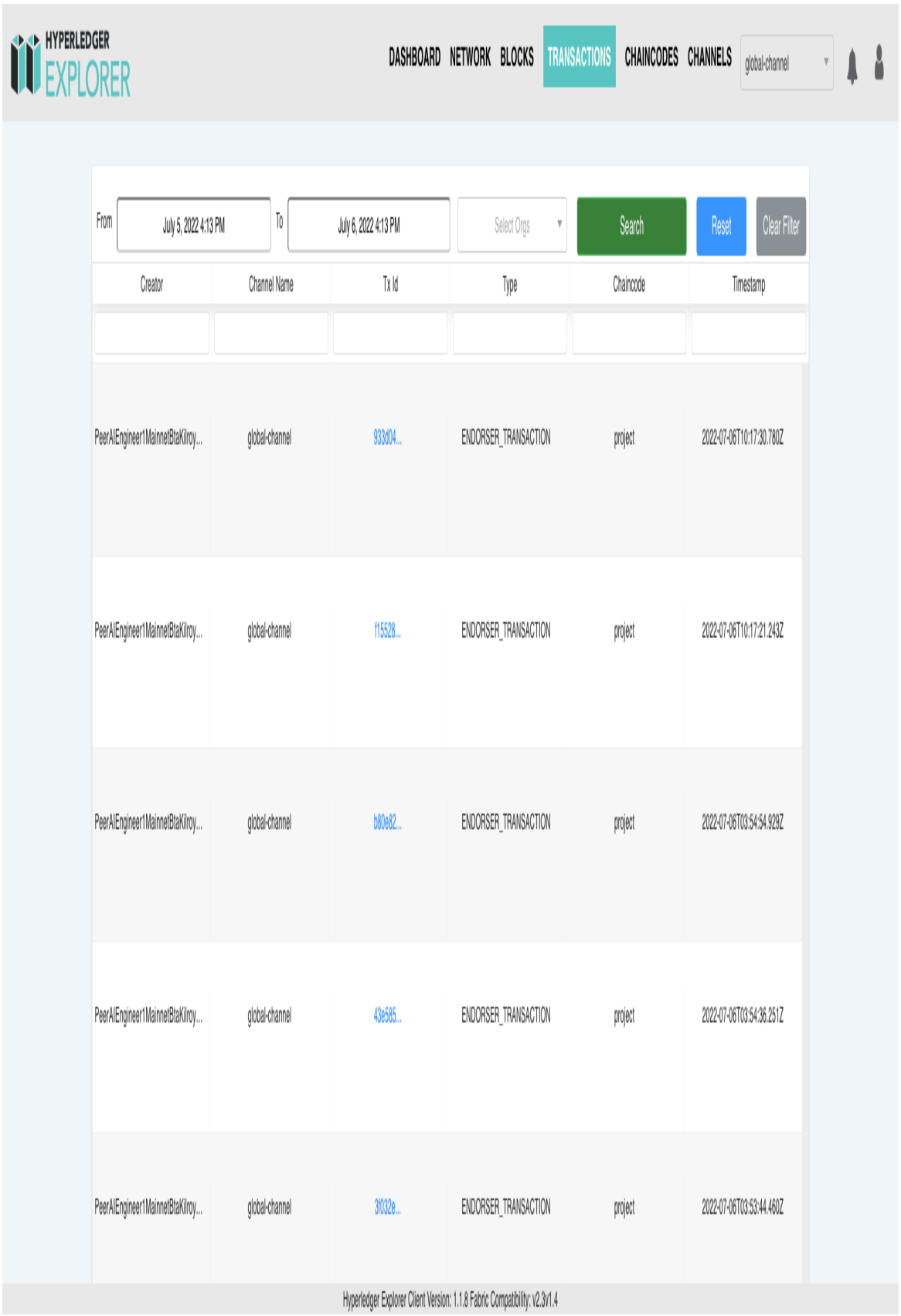

HYPERLEDGER
EXPLORER
DASHBOARD
NETWORK
BLOCKS
TRANSACTIONS
CHAINCODES
CHANNELS
global-channel
From
July 5, 2022 4:13 PM
To
July 6, 2022 4:13 PM
Select Orgs
Search
Reset
Clear Filter
Creator
Channel Name
Tx Id
Type
Chaincode
Timestamp
PeerAIEngineer1MainnetBtaKilroy...
global-channel
933d04...
ENDORSER_TRANSACTION
project
2022-07-06T10:17:30.780Z
PeerAIEngineer1MainnetBtaKilroy...
global-channel
f15528...
ENDORSER_TRANSACTION
project
2022-07-06T10:17:21.243Z
PeerAIEngineer1MainnetBtaKilroy...
global-channel
b80e82...
ENDORSER_TRANSACTION
project
2022-07-06T03:54:54.929Z
PeerAIEngineer1MainnetBtaKilroy...
global-channel
43e585...
ENDORSER_TRANSACTION
project
2022-07-06T03:54:36.251Z
PeerAIEngineer1MainnetBtaKilroy...
global-channel
31032e...
ENDORSER_TRANSACTION
project
2022-07-06T03:53:44.460Z
Hyperledger Explorer Client Version: 1.1.8 Fabric Compatibility: v2.3v1.4

Figure 3-12. HLF Explorer transactions made across different peers

The key word called *Smart contract* is common among the developer because it holds the blockchain logic or script that reads and writes the data over a blockchain database called ledger which holds the facts about current and historical data of objects. Blockhain administrator calls the smart contracts as chaincode, they deploy the chaincode into a particular peer and mention channel names which get involved in sending the request to the chaincode. Figure 3-13 shows how Hyperledger Explorer gives the information about the name of the chaincode related to different channels. Interestingly, Explorer also gives the total number of transactions committed through the channel and chaincode on a particular peer.

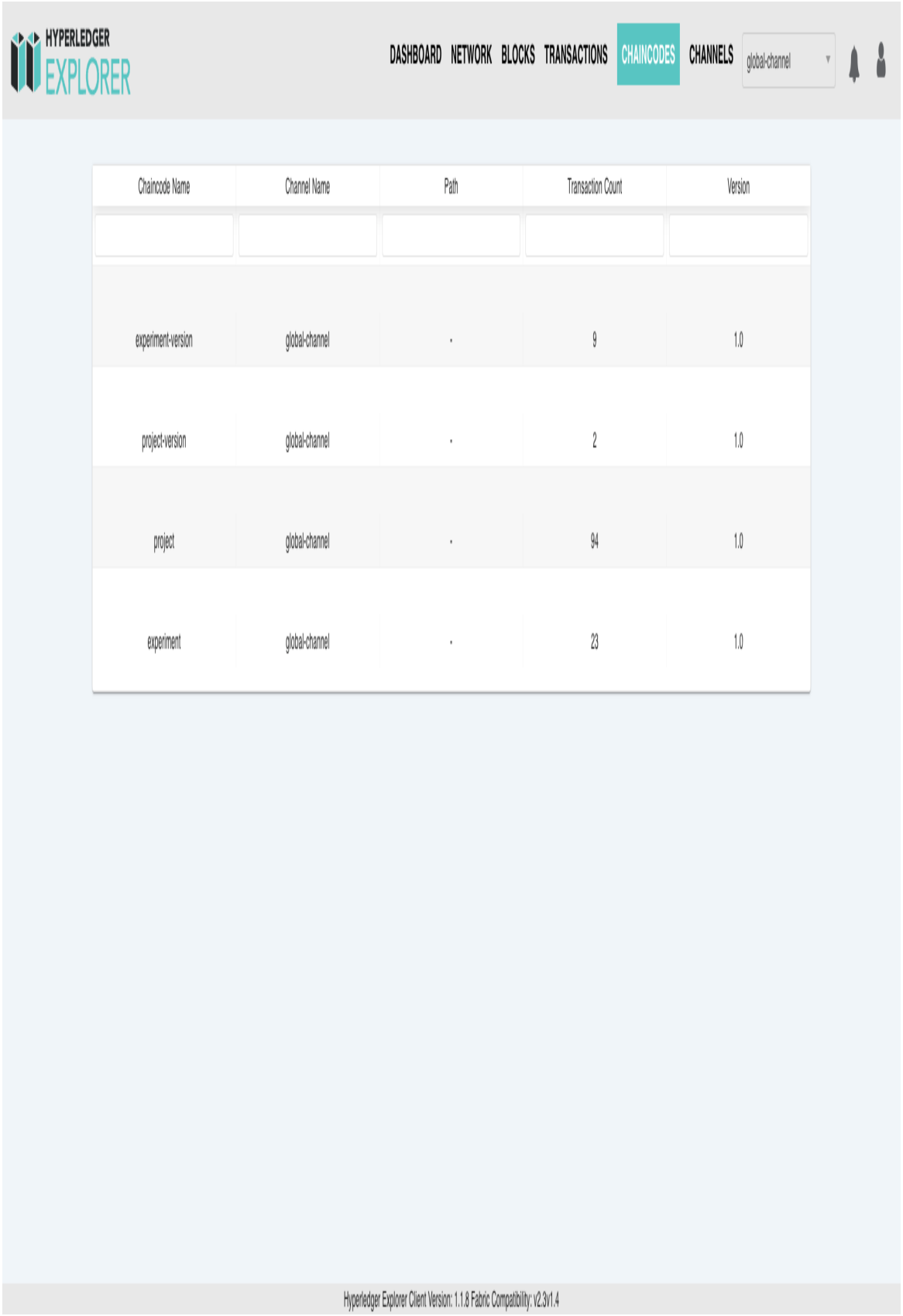

HYPERLEDGER
EXPLORER
DASHBOARD
NETWORK
BLOCKS
TRANSACTIONS
CHAINCODES
CHANNELS
global-channel
Chaincode Name
Channel Name
Path
Transaction Count
Version
experiment-version
global-channel
-
9
1.0
project-version
global-channel
-
2
1.0
project
global-channel
-
94
1.0
experiment
global-channel
-
23
1.0
Hyperledger Explorer Client Version: 1.1.8 Fabric Compatibility: v2.3v1.4

Figure 3-13. HLF Explorer showing chaincodes and the transactions made through them.

Channels are the medium to pass a request (put or get) to a particular peer. A channel can be shared with multiple peers in order to share the data so let's say if a record needs to stay in a peer of stakeholder and MLOPs engineer then they need to have a common channel which passes a request to both peers. In this way, a channel helps to distribute data among different servers. Channel name, total number of blocks, and transactions created through it, are shown in Figure 3-14.

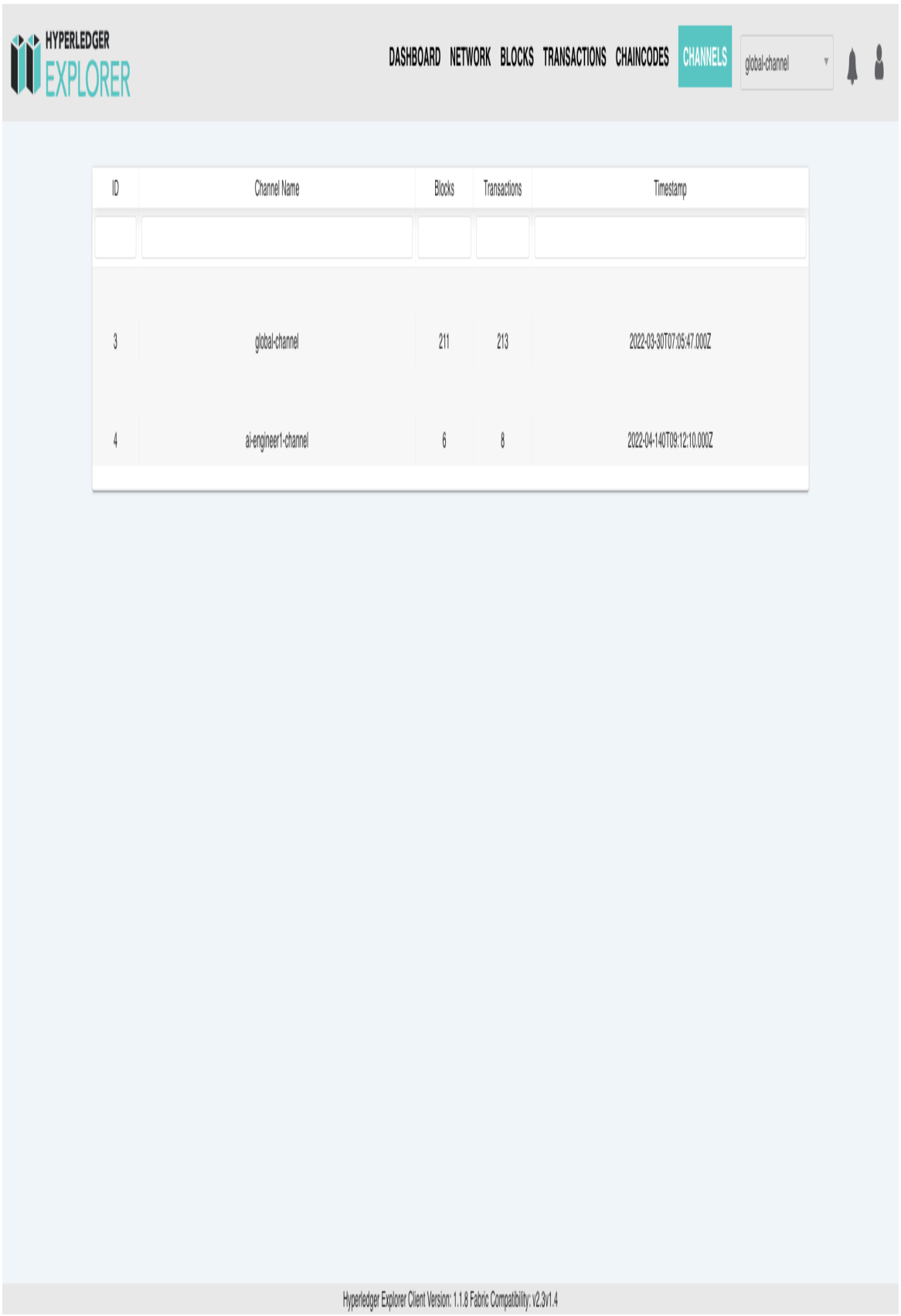

HYPERLEDGER
EXPLORER
DASHBOARD NETWORK BLOCKS TRANSACTIONS CHAINCODES CHANNELS
global-channel
ID
Channel Name
Blocks
Transactions
Timestamp
3
global-channel
211
213
2022-03-30T07:05:47.000Z
4
ai-engineer1-channel
6
8
2022-04-14T09:12:10.000Z
Hyperledger Explorer Client Version: 1.1.8 Fabric Compatibility: v2.3v1.4

Figure 3-14. HLF Explorer showing channels with the blocks and transactions committed through.

Hyperledger Fabric monitors each and every activity happening within nodes like number of peers, channels, transactions and so on. This helps to understand the uses of the resources and plan for the scalability.

Hyperledger Fabric doesn't explicitly provide any such interfaces that facilitates administrators to create peers, channels, Certificate Authorities and so on, however some enterprise level blockchain providers like IBM and Oracle do provide the blockchain interface to manage blockchain activities by allowing the user to create organizations peers, channels, linking the channels to peers, install chaincode and many more, through web interface.

Smartphone and Tablet Apps

As an alternative or addition to a web-based user interface, AI-blockchain systems can use a mobile app. Mobile apps are generally platform-specific, and are often distributed through platforms like the App Store or Google Play.

In general, it isn't necessary to build an app just to use a web-facing UI on a mobile device. Instead, web interfaces can be built to be fully *responsive*; in other words, able to detect what type of device they are being viewed on, and able to arrange the elements so they are all visible and usable. Some extra functionality, such as taking advantage of hardware features and their integration with your system, might make it worthwhile to build and maintain an app.

In some cases, when clients want to build a mobile app for ease of navigation, the mobile app would need to be built using a native iOS or Android development platform like Swift or Java. However, the development cost of the native mobile app using such platforms is very high, so the customer might want to skip the double investment for the mobile app. To overcome the problem, new platforms for mobile app development have been introduced like React Native, Ionic Framework and

so on. These tools can create both iOS and Android compatible apps out of the single code developed by developers.

Email and Text Notifications

It's important to be able to notify the user of a change in status of some item in the BTA. Sometimes the alert might be critical; other times it could be something not as critical, but important. Giving users a way to enable or disable these by feature is important to making notifications helpful and not annoying.

The example BTA you will build in chapter 5 is set up so that notifications are mandatory when a person registers, and when their registration is either verified or rejected. It's also sent when a user attempts to reset their password or has forgotten it.

Spreadsheets

Frequently, users of the system will request spreadsheets because it is an easy way for them to work with and to share the data. Including a standard way for users to upload and download data in spreadsheets is recommended. This type of functionality can be made into an optional feature that can be delegated based on role.

Spreadsheets can be incorporated into BTA projects to create mockups where product designers can play with data consulting with team members and experts. The spreadsheet could be shown to the experts of AI and blockchain so that the ideas can be shared and updated.

Third Party Systems

No particular AI can do 'everything,' despite marketing claims. There can be many reasons why a company needs to integrate the AI with third party software, and it usually boils down to needing to add a feature that is too complicated or expensive, with respect to time or money, to build from scratch.

Consider the vast number of ways that human beings can interact with AI, and the ways that intelligent agents might interact with one another. For example, your car could contain any number of smart parts, which are created and trained by any number of sources. Do the parts work together reliably? Do experiments need to be done to make sure the AI of each part is optimized to work together as one system? When you trust the car to carry you from place to place, do you want to run a check to make sure that everything is working?

The BTA project has used several third party libraries to accelerate the project development focusing on the end result, which has saved a lot of time and cost. It has used AI libraries like Tensorboard logger, AWS S3, and Oracle bucket in the AI part. This project has created its own library to connect with blockchain and web applications. If you talk about a web application that BTA has created, it uses many libraries because the application is based on MEAN stack which easily provides the libraries to ease the task.

A person needs only to look at the ways AI interacts with people today to see the way that tethering to BTA can enhance the AI. An API would connect BTA to that device. Here are some examples:

- Google search - right now, Alphabet's Google is the most popular search engine. Would it be possible to tether blockchain to search results so that the next time you want to go to the same result, a fake version of the website you want doesn't slip in front of you?
- Open your phone or tablet with facial recognition positive ID - every instance of you using Face ID would be verified and saved using BTA, so that you can check your mobile device login history in blockchain and look for any attempt to get in that you did not make.
- Sending emails or text messages - when you create an email or message, it can be saved on blockchain and sent out with an attached icon that will let the recipient know that only you sent it.

- Social media - when you browse, comment, or send a DM, wouldn't you want the recipient to know it's really you, not a spoofed account? BTA can provide a badge or icon as proof.
- Digital voice assistants (Siri, Alexa, etc.) - login authentication can be handled through BTA so that a spy can't break into your system, eavesdrop on your conversations, send emails without your knowledge, or any number of other things.
- Smart home devices - BTA can provide authentication for login and secure settings on things like home thermostats, indoor and outdoor lighting and other environmental controls, refrigerator-created shopping lists, or any smart appliance.
- Work commute - starting from an AI-controlled automatic garage door opener to your AI-controlled car, BTA can provide authentication for every aspect of operation that interacts with the outside environment.
- Personal banking - not only logging into your online accounts, but any aspect of the buying experience flow can be authenticated and compared to our typical buying patterns. These services exist today, but they aren't connected to blockchain, though the banking institutions may use AI to detect or predict fraudulent transactions, it's all behind the scenes. This would be fraud detection that we can see and control from our mobile devices. This could extend to shopping experiences on sites that you use often, like Amazon or Alibaba.
- Personal entertainment choices - what you see on Netflix, Hulu, and other streaming services; what you watch at the movies, can be somewhat controlled by AI steering you towards or away from certain selections. BTA can abate that behavior so you have the freedom to search the entire library.

There are many other systems in an enterprise that perform mission-critical functions, and there is a constant need to interact with those systems.

Working with APIs

APIs have transformed the way developers create and write applications. Most businesses using modern technology use APIs at some level to retrieve data or interact with a database for customers to use, creating the industry vertical called ‘Platform as a Service’ (PaaS). BTA has been integrated with third-party software through internal, custom APIs. There are companies whose main product is an external API with a specific purpose, such as communications, payments, or email, as a PaaS model. These companies enable developers to build applications on their platform, which might perform functions such as hosting web servers or communication applications.

There are businesses whose value comes from connecting different APIs and web services, which are categorized as “Integration Platform as a Service.” IPaaS companies let users connect disparate web services and tools, most notably to route data or automate workflows. Both of these verticals have grown tremendously, fueled by the extensibility and ease of use of APIs.

BTA is storing log files, testing and training data sets into an Oracle bucket through the APIs provided by the Oracle cloud. Its connection to AWS S3 is available in Oracle cloud through an access key, a secret key, and an endpoint URL. BTA uses PyTorch lightning’s logger library where the format of the log file that the project needs is created.

Once the trained and tested AI model is launched into production, BTA exposes certain web services as API which can be monetized. Such APIs shall be used in web or mobile or SaaS applications by including provided secret or access keys.

Integrated Hardware

The scenarios for using blockchain-tethered AI with integrated hardware are limited only by the imagination.

Since the software that AI engineers build for AI workflow systems can be integrated to include just about any intelligent hardware system, it is important to consider the real-world impact of such devices and make sure there are *sanity checks*, or cross-checks, to make sure that the system is functional and compatible with other components. One very common integration of AI and hardware occurs in today's cars, which have many different components that have to pass individual checks in order to work together, making each car function properly in the current environment.

One methodology used to cross-check components, Operational Design Domain (ODD), determines the multitude of details that determine where and when a vehicle (or similar equipment like a robot) can operate. To understand this, consider that when you drive your own non-automated car, you are fully aware of your ODD, considering factors such as time of day and road conditions when you decide how to control the vehicle. You would know you probably don't want to drive on an icy road in rush hour, especially if your tires aren't in great shape. A fully self-driving vehicle, however, would have AI in command of analyzing the ODD without any human needing to be in the loop. Such a vehicle might check the tire pressure, check the temperature, and go into non-operational mode.

The sample ODD principles used for automated vehicles can be used on robots and hybrids. Imagine a car-robot hybrid that picks your child up from the bus stop or does some other regular nearby errand. Think about the various components that might be talking to one another - a computerized steering system, lighting system, braking system, interior cabin climate, communications system; the list goes on.

Using ODD, each component has its own memory stored on an *EPROM* (erasable-programmable read-only memory that is used on hardware components to store software routines and data), that remembers its optimization data.

If you pull a component from a robot that always runs during the daytime in dry weather, chances are it will have to be reoptimized in order to run at

night or in the rain. So it is important that a sanity check is done by the vehicle to make sure the optimization isn't wrong for conditions.

This data stays in the vehicle unless the car undergoes diagnostic testing. If the card is pulled to a different vehicle, it will retain the optimization settings in EPROM until they are reset by the new vehicle. Now imagine introducing blockchain to this scenario as a more permanent form of memory. This means that instead of just analyzing current settings, you have a tamper-evident record showing exactly how and why each part was optimized.

As shown in Figure 3-15, each component could have its own blockchain channel which allows its own chain, stored in the part on its EPROM, that can act as a tamper-evident, auditable, permanent memory bank.

Blockchain channels and ODD test for adding new sensor w ML capability

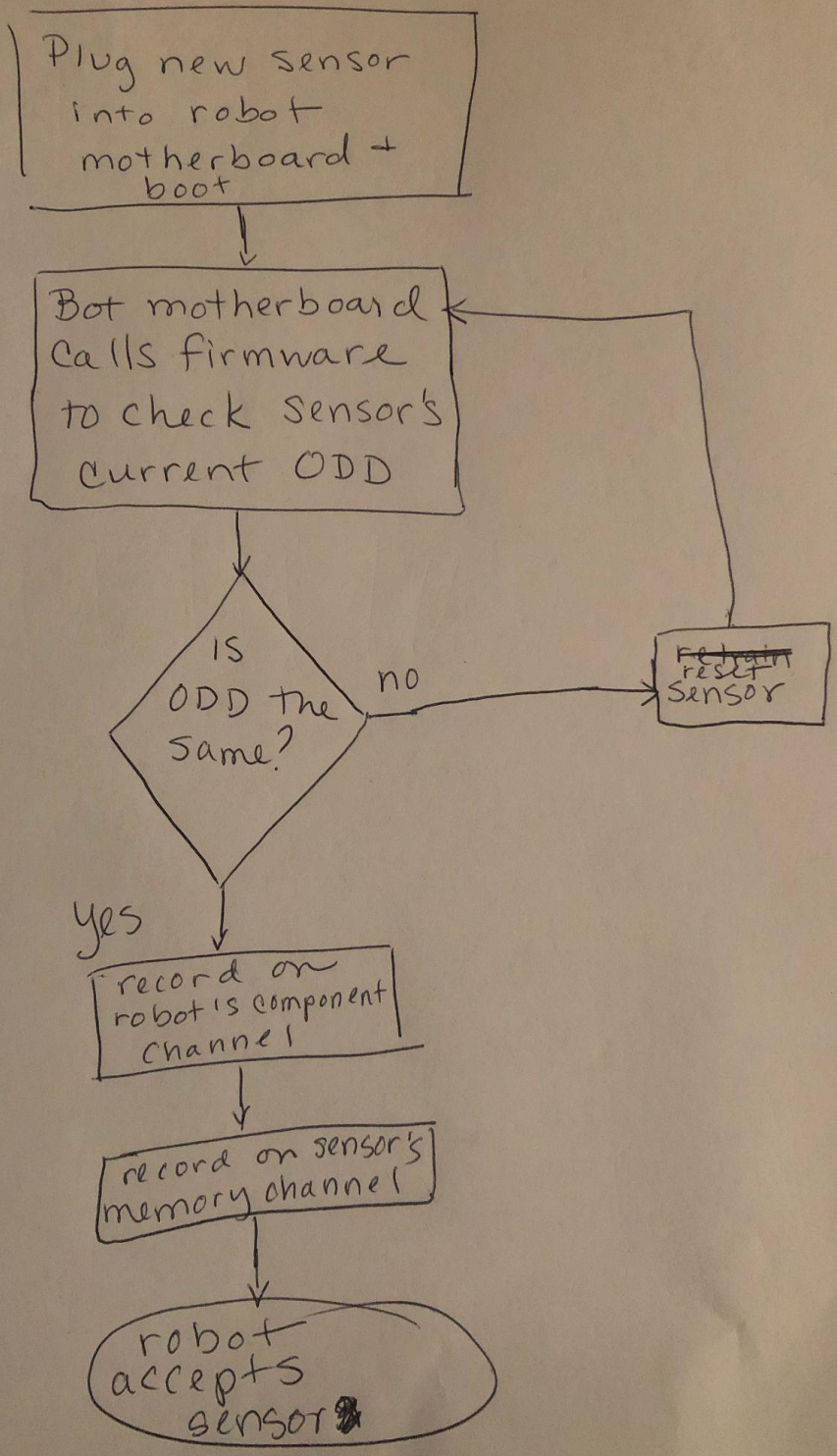


Figure 3-15. Diagram showing how blockchain channels might be used to add a new sensor with reinforcement learning capability into a robotic system.

Other nodes of the blockchain can be stored on other stakeholders systems, such as the workbench used for testing parts. Whenever the part comes into communication with the rest of the blockchain network, the data on that part's ODD channel could be shared with the rest of the blockchain network, either openly or as a zero-knowledge proof. This allows a part to take its entire history, including data supporting how and why optimization decisions were made, with it when it is removed and put into another system.

Also bear in mind that much of this integration testing is done digitally, through the use of *digital twins*, which are a digital representation of a physical device (or twin) such as a robot. One interesting use case for digital and physical twins is robot reinforcement learning. Figure 3-16 illustrates how this process might flow if a part is removed from a robot and placed onto an engineer's workbench. The engineer then does some work to the part - maybe they create 1,000 digital twins and run massive experiments and come up with some new set of optimization data. Then those new settings are loaded onto the card and the card is loaded back into the robot. With a blockchain scenario, the new optimization data is stored on the card, along with the details of the model such as the AI Engineer's identity and history of the experiments.

When the part is placed into the robot, blockchain smart contracts can be used to validate whether or not a modified part meets the robot's ODD requirements, such as maximum speed or operating conditions.

Sensor w ML capability remembers its history even when removed from robot.

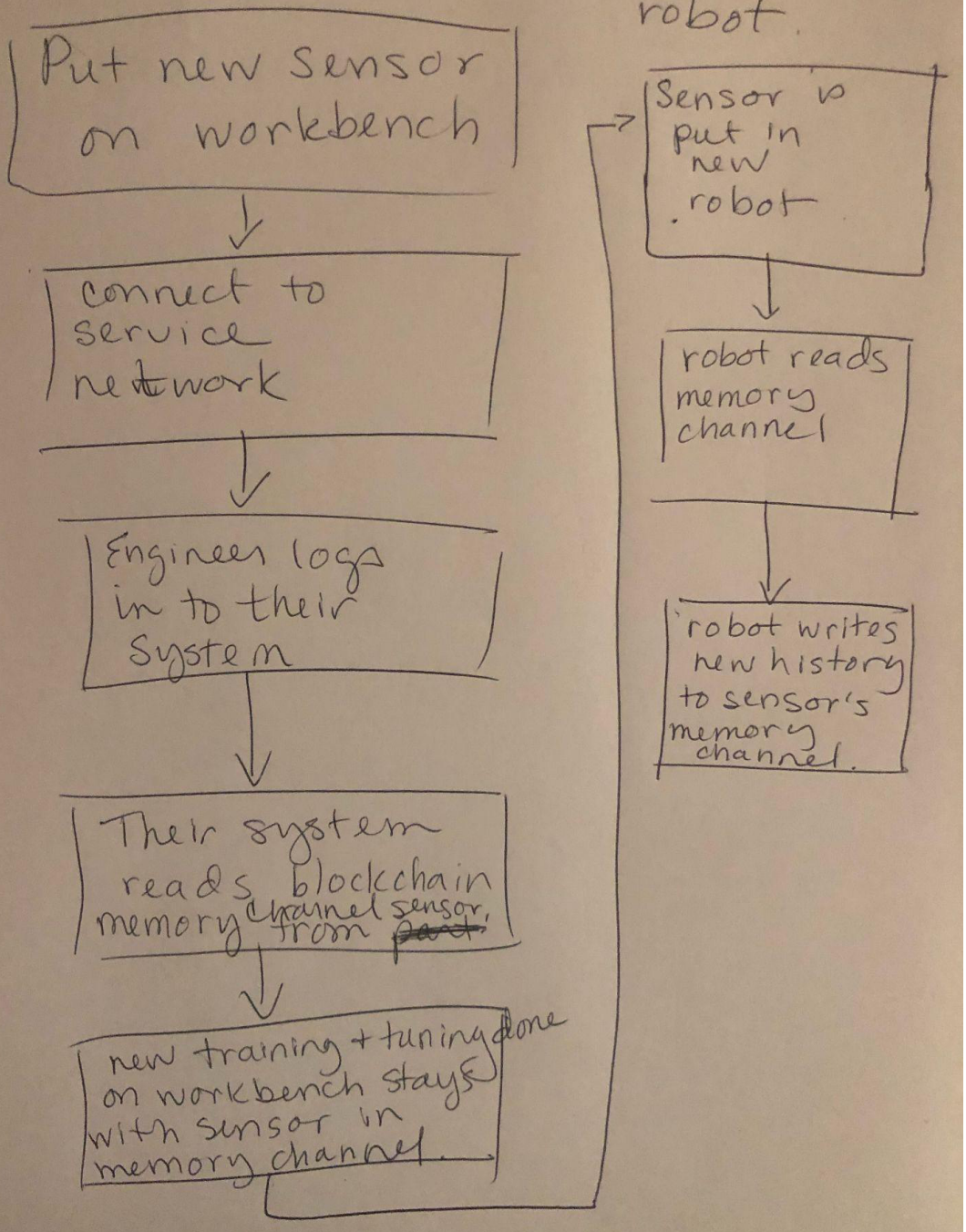


Figure 3-16. Sensor with ML capability remembers its history even when removed from the robot.

Consider that when integrating hardware with blockchain-tethered AI, ODD could be substituted for any number of other checks - a version check, a requirements check, a maintenance check, an intent check, an expiration check, a recall check - and all of these could be various blockchain channels. the checks could also be done in parallel as opposed to always sequentially.

Additional AI Developer UIs

It is possible to write your own web interface in Python to highlight the input, output, and comparisons of the intelligent agents you've built. An alternative is to use an existing library. The most widely used tools to build AI algorithms include Microsoft's Azure Machine Learning Studio and IBM Watson Machine Studio. Whichever tool you choose, it must be able to handle duplicate data points, locate model blind spots, and other issues around debugging the model during development.

Common tools for building the system running the ML engine are the Google Cloud ML Engine and TensorFlow. In order to have a robust, responsible system, the qualities of the user interface must include information and visualizations of the models and their data, the interface must be reusable by multiple stakeholders to facilitate cross-team analysis and collaboration. The system must have role-based access control set up with multiple factor login authorization. The user interface for the system should be available to reference across platforms and notebooks to provide public links for demos to clients or teaching purposes.

The BTA project has used following tools:

- **Oracle Cloud Infrastructure (OCI) Data Science** is an end-to-end machine learning (ML) service that offers JupyterLab notebook environments and access to hundreds of popular open source tools and frameworks. Data Science service offers features like Model Building, Model Training, Model Deployment and Model Management.

- **Oracle Bucket:** OCI Object Storage service provides a bucket, which is a container for storing objects in a compartment within an Object Storage namespace. A bucket is associated with a single compartment. The compartment has policies that indicate what actions you can perform on a bucket and all the objects in the bucket.
- **Jupyter notebook** provides a web-based interactive development environment for notebooks, code and data.
- **PyTorch lightning** is the deep learning framework for professional AI researchers and machine learning engineers who need maximal flexibility without sacrificing performance at scale.
- **Tensorboard logger:** PyTorch lightning offers the TensorBoard logging framework. Its logs are stored to the local file system in TensorBoard format. This is the default logger in Lightning, it comes preinstalled.

Another reason to integrate with another system would be to connect it to a program that analyzes the transparency, explainability, and fit with respect to your business model. Connecting with the right programs helps to validate the AI model by analyzing whether the data might be biased or fair. Blackbox testing can be applied to AI models in order to identify the accuracy level of the model. The AI model Validator (MLOps Engineer) needs to understand the purpose of the project, the hyperparameters, and algorithms used to train the model. Such an integration would be to overcome the black box nature of ML within the system you are building. Additionally, programs like **SHAP, LIME** or Explainable Boosting Machines **(EBM)** can be integrated with your ML. These are model-agnostic and provide easy-to-understand graphics and terminology explaining the results of the iteration output. Anyone who is validating the model needs to ensure that this type of analysis has been performed and the conclusion from the analysis is aligned with the business problem.

THE FOUR CONTROLS BY PARTICIPANT AND USER INTERFACE

If you take a moment now to think back to the four controls you learned about in Chapter 2, it will help you to apply principles such as traceable identity, workflow, tamper-evident verification, governance and provenance to your BTA. The controls and the points to consider follow:

Control 1: Pre-establish Identity and Workflow Criteria for People and Systems

- Registration; receive system verification by the super admin that can be disabled by one click.
- Log in with the temporary password, then privately change at will.
- Within an organization, there is an organization admin that can add users to the organization and control which modules each user can access, which in turn controls what they see on their user interface.
- Within an organization, the organization admin can control the permissions for each user by turning them on or off for each function based on the user's functional role within each module.

Control 2: Distribute Tamper-Evident Verification

- Each change of information by a validated user is recorded as a transaction, and saved in the blockchain - create, edit, undo, or delete.
- Each transaction's information is given a transaction ID and saved in a block with a unique hash and a timestamp, and added to the blockchain.
- With any attempt to tamper with a block, the fraudulently generated hash will not match the authentically generated hash, and it will be evident immediately. An error message will be

generated and sent out to a predetermined distribution list including the super and organization admins.

Control 3: Govern, Instruct, and Inhibit Intelligent Agents

- The transactions are conducted by verified users and by consensus, recorded and stored on blockchain.
- The workflows can be set up to instruct the user what to do if there is a deviation from the established workflow, such as a rejection of output data.
- The models can have an alarm to trigger if selected parameters of the output exceed a set difference level between the validation data and the predicted data (the residual).

Control 4: Show Authenticity through User-Viewable Provenance

- Display an icon alongside the data to show that its provenance has been tracked on blockchain, and that the proof recorded on blockchain matches up with the actual data.
- You can also display e-signatures or images of actual signatures that the stakeholders have made, on approval workflows, such as for final models. Each signature can display the afore-mentioned blockchain-verified icon alongside the signature (but as a separate object).
- Allow all users on the system to see the blockchain history of every blockchain-verified information that they have permission to see.

BTA System security

Since two different platforms (Blockchain and AI) are being integrated through a web application, security needs to be checked at all levels.

1. AI: In the example BTA, the AI is created in the Oracle Cloud using different available resources in the cloud like Oracle Data Science resource, Oracle Bucket and Oracle Jupyter notebook. With the adequate permission, the user can get access to the data science tool of Oracle from where the notebook is accessible. Users also need to get access to the bucket so that artifacts and log files can be placed there. So when all required access and permissions are given, the AI engineer starts to write code to create/train and test the AI model. Apart from the access and permission, there is a security layer created within Oracle cloud by Oracle itself.

   This project uses the Oracle Cloud Data Science platform, on which each user needs to have a user account in Oracle Cloud with the proper access and permissions over the resources. The user begins at the login page by typing in their tenancy identification (the Cloud Account Name), which is set up by the organization admin when the account is purchased. Only users authorized by that admin can log into the system. Once the tenancy and the service location in the cloud has been verified, the individual login screen appears. After the username and password are verified, the user enters their dashboard. Having selected the AI/ML Data Scientist role, the sidebar gives the user. Along with username and password, BTA is providing an extra security layer that comes out of blockchain, i.e., Hyperledger Fabric. The BTA algorithm encrypts user credentials that exist within the wallet of the HLF network using SHA-256 algorithm, and the encrypted key is sent to the user after the registration is successfully saved in blockchain. The user has to enter the encrypted key in the BTA application after normal username and password is verified.

2. The BTA project uses two databases:

   The BTA project has used two kinds of databases i.e. MongoDB and Blockchain. All the application related data is not supposed to be stored in the blockchain because storing such data in the blockchain creates problems in business where the policies change so frequently. Security of MongoDB is maintained by opening only the port used for

db connection and using proper user authentication. Private blockchain is used in the project where only authenticated and authorized users can access. Transactions and records in the blockchain are created through the smart contract which runs in the network only after getting approval from all related peers.

a. MongoDB:

i. None of the DB ports are opened so outside access is not possible.

ii. A Bcrypt algorithm is used to encrypt the user's password so the real password just remains with the real user.

b. Blockchain: Hyperledger Fabric is used as a blockchain platform in this project where the user's credentials are stored in the secured blockchain environment. BTA has also added an extra layer of security by creating a blockchain based password and using it during user login. So the user needs to pass login credentials across the blockchain as well, not only in MongoDB.

3. Additional Security:

Additional securities can be applied in the application layer using JWT refresh token which the tokens get refreshed in a certain amount of time defined in the system environment. Next, Google Recaptcha which protects the web application from unauthorized login/register using the script by an anonymous user. If different form fields other than defined in DTO, are sent through CURL or Postscript, it is not accepted or proceed further by the application. Concept of access/permission guard has been implemented where a user within an organization can access only those resources which are assigned to them through the guard.

a. JWT authorization with refresh token.

b. Google Recaptcha on signup/login with backend validation.

c. API Response and Body need to be as per the Data Transfer Object (DTO), otherwise it will throw an error.

d. API guard along with organization and feature guard for proper access flow (nobody can use the API except the user with proper access).

Implementing your Blockchain-Tethered AI's UI

As you move on to chapter 4, you will find it handy to think of what types of UI you will want to implement for each type of user. If you plan to use your BTA in production, keep in mind that it is nearly impossible to get every requirement correct without getting feedback from active users. The people using the system, and your response to their requests, will be a big factor in its success.

Chapter 4. Planning your BTA

> **A NOTE FOR EARLY RELEASE READERS**
>
> With Early Release ebooks, you get books in their earliest form—the authors' raw and unedited content as they write—so you can take advantage of these technologies long before the official release of these titles.
>
> This will be the fourth chapter of the final book.
>
> If you have comments about how we might improve the content and/or examples in this book, or if you notice missing material within this chapter, please reach out to the editor at *ccollins@oreilly.com*.

After reading Chapters 1-3, you are now prepared to start building your own blockchain tethered AI system (BTA). This chapter helps you get the right understanding of the BTA for planning your development environment (Chapter 5), for building a BTA that tethers a sample model (Chapter 6), and for using your BTA (Chapter 7).

The BTA is built by interweaving the MLOps process with blockchain in such a way that the MLOps system *requires blockchain to function*, which results in the AI being tethered. To do this, the BTA integrates blockchain with MLOps so that a model must be approved in order to proceed through the cycle of training, testing, review, and launch. Once a model is approved, it advances to the next step of the cycle. If it is declined, the model goes back to the previous steps for modifications and another round of approvals. All of the steps are recorded in the blockchain audit trail.

To plan for a system like this, you need to consider the architecture of the BTA, the characteristics of the model you are trying to tether, how accounts and users are created and how permissions are assigned, and the specifics of how and why to record certain points on blockchain.

BTA Architecture

The BTA has three major layers in its architecture: the BTA web application; a blockchain network; and the buckets.

Alongside the BTA, there is yet another required environment. This is used for model development and review, and requires a data science rich platform. For this you will use a Jupyter Notebook running in the Oracle Cloud. This cloud-based Python development environment helps you to build and test models and provides you with buckets to store logs and artifacts.

These environments work together to allow AI and MLOps engineers to route and approve a new or modified model, store objects like models and logs in buckets, while recording cryptographic hashes of the objects onto blockchain.

The BTA application architecture in Figure 4-1 shows how the BTA users (AI engineer, MLOps engineer and stakeholder) interact with Oracle Data Science, an Oracle Bucket and a blockchain network through the BTA web application. As shown on the right side of the diagram, the AI engineer and MLOps engineer are connected to the AI instance being managed in Oracle Data Science as well as to the blockchain, while the stakeholder is only connected to the buckets and the blockchain.

When the users are created in the BTA, the organization admin also issues them Oracle Cloud credentials to connect with Oracle Data Science, blockchain credentials to store AI model development data and events in their own blockchain node. These credentials are stored in the BTA web app under each user's configuration. The users log in into the BTA, and the Oracle Cloud and blockchain are seamlessly connected to the system with no further action from the users. Chapter 5 explains the Oracle Cloud setup process in detail.

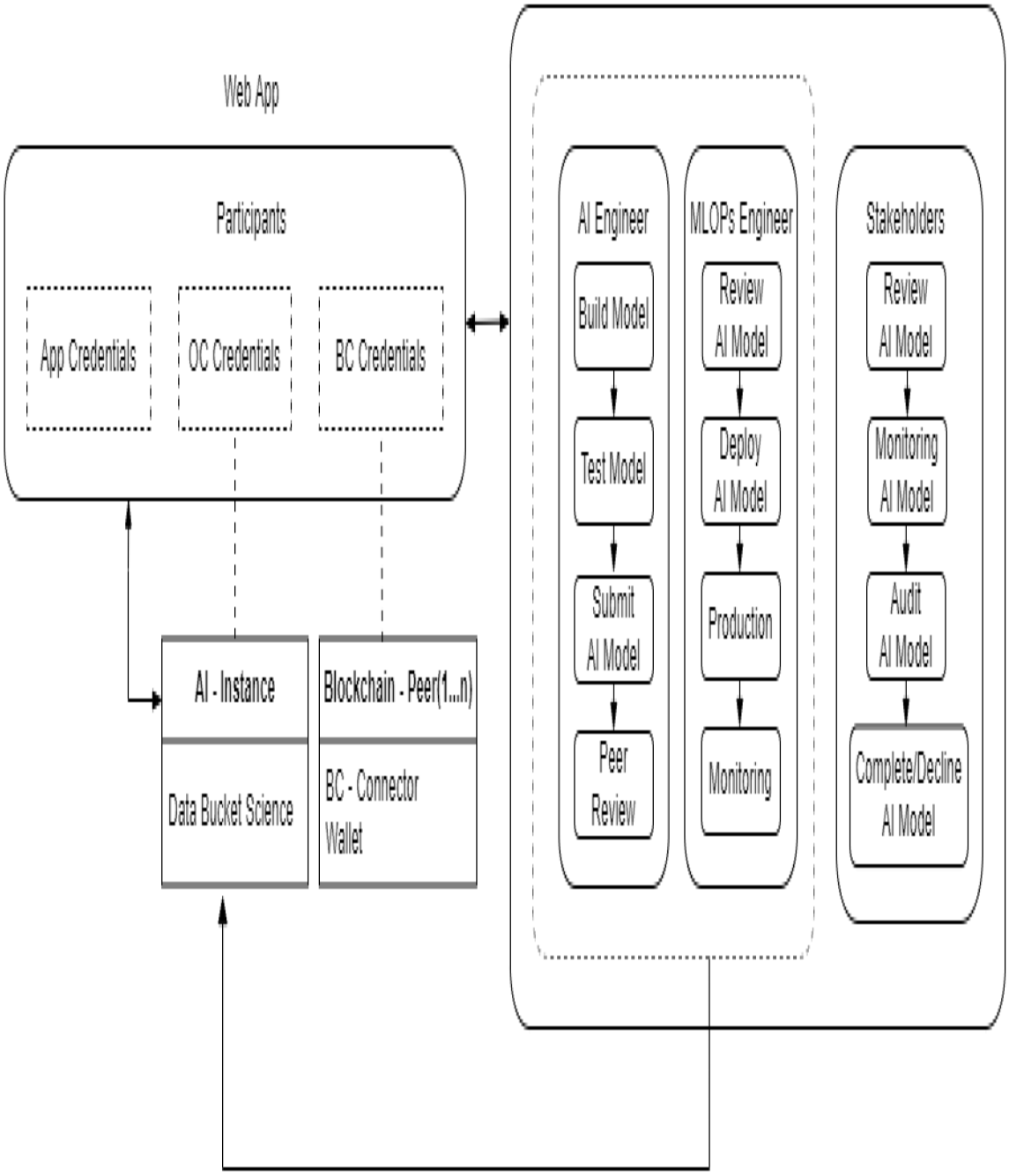


Figure 4-1. BTA application architecture

> **NOTE**
>
> While the stakeholders will not take part in model development actively, they still need the Oracle Cloud access because their logged in user requires access to blockchain and the archives in the buckets.

Sample Model

In order to test your BTA, you need a model to tether. This book's accompanying code provides a sample traffic sign detection model, which is a good model to test because it has features that allow you to try all four blockchain controls, as Chapter 2 describes.

The completed AI factsheet below details the traffic sign detection model. There could be many more facts on the factsheet, but this is enough information for testing purposes.

AI FACTSHEET: TRAFFIC SIGN DETECTION MODEL

Purpose

Detect traffic signs based on input images

Domain

Transportation

Dataset

Dataset is divided into training data and test data. Dataset is collected from "German Traffic Sign Recognition Benchmark (GTSRB)". This has 50K images and 43 traffic sign classes.

Algorithm

Convolution Neural Network

ML Type

Supervised learning

Input

Images of traffic signs

Output

Identify input images and return text containing names of traffic signs and probability

Performance metrics

Accuracy of model based on F1 score and confusion matrix.

Bias

Since the project uses 43 classes, each class has 1K to 1.5K images so data skew is minimized. No known bias.

Contacts

MLOps and AI engineers, stakeholder and organization admin are the major contacts of the project. (In this case, all of those contacts are you.)

How the Model Works

In order to try the sample model, you can browse to its user interface using a web browser and upload an image of a traffic sign. If the model thinks it recognizes the sign, it will return the name of the sign. Otherwise, it will indicate that the sign is not recognized. This model is similar to visual recognition used in vehicles to recognize traffic signs.

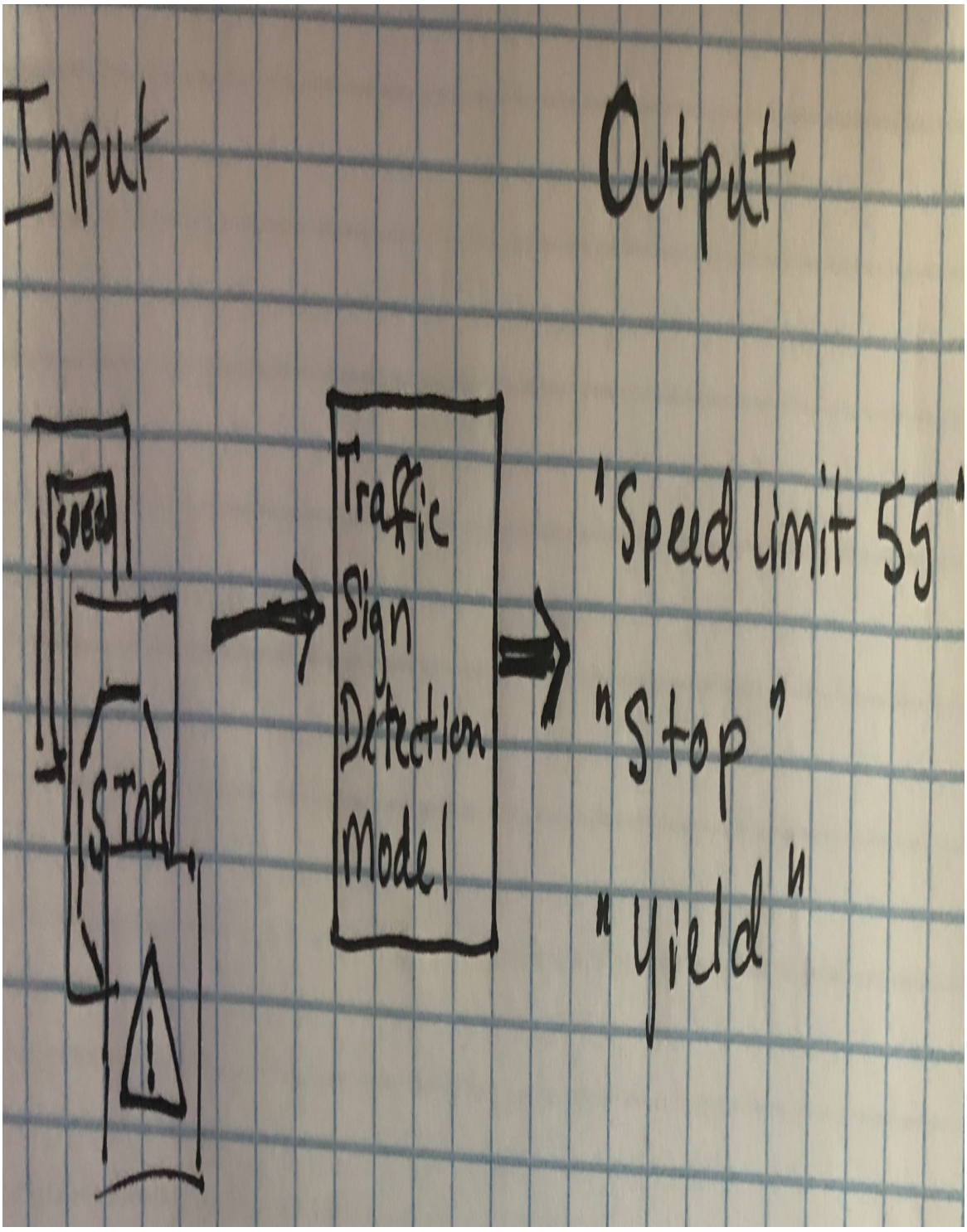


Figure 4-2. Sample inputs and outputs of traffic sign detection model.

A simple visual recognition model like this is built by training the model with thousands of images that contain street signs, and thousands of images

that contain items that look like street signs, and aren't. This has already been done in the example model that is included with this book.

If this were a production model, AI and MLOps engineers might continue to attempt to improve its accuracy. They may introduce new sets of training or testing data, or they might modify an algorithm to do something like consider the context of the traffic sign (is it covered by a tree branch? Is it raining?) The AI and MLOps engineers routinely run controlled experiments with modified variables, then weigh the results of the experiments against the results of previous tests and look for measurable improvements.

The modifications to the traffic sign detection model itself, which are explained in depth in Chapter 6, are made using a Jupyter Notebook inside an Oracle Data Science instance. Figure 4-3 shows a method to train the model using PyTorch, with the notebook script downloading and extracting training images. Chapter 5 includes details on how to set up and use this environment.

ORACLE Cloud MNIST_v1 Session remaining: 2hr 10min Sign Out

File Edit View Run Kernel Git Tabs Settings Help

MNIST_AI_MODEL_v1.ipynb

Code Python 3

Name

finallogfile
formattedjson
MNIST
model
tb_logs
MNIST_AI_MODEL_v...

Train the model using the trainer class and then invoke the fit method to actually train the model

[6]:
```
model = LitMNIST()

# Define the lightning trainer
trainer = Trainer(
    gpus=AVAIL_GPUS,
    max_epochs=5,
    progress_bar_refresh_rate=20, logger=logger,
)
trainer.tune(model)
trainer.fit(model)
```

```
/opt/conda/lib/python3.7/site-packages/pytorch_lightning/trainer/connectors/callback_connector.py:91: LightningDeprecationWarning: Setting `Trainer(progress_bar_refresh_rate=20)` is deprecated in v1.5 and will be removed in v1.7. Please pass `pytorch_lightning.callbacks.progress.TQDMProgressBar` with `refresh_rate` directly to the Trainer's `callbacks` argument instead. Or, to disable the progress bar pass `enable_progress_bar = False` to the Trainer.
  f"Setting `Trainer(progress_bar_refresh_rate={progress_bar_refresh_rate})` is deprecated in v1.5 and"
GPU available: False, used: False
TPU available: False, using: 0 TPU cores
IPU available: False, using: 0 IPUs
Downloading http://yann.lecun.com/exdb/mnist/train-images-idx3-ubyte.gz
Downloading http://yann.lecun.com/exdb/mnist/train-images-idx3-ubyte.gz to ./MNIST/raw/train-images-idx3-ubyte.gz
```

```
9913344/? [00:00<00:00, 20896617.60it/s]
Extracting ./MNIST/raw/train-images-idx3-ubyte.gz to ./MNIST/raw

Downloading http://yann.lecun.com/exdb/mnist/train-labels-idx1-ubyte.gz
Downloading http://yann.lecun.com/exdb/mnist/train-labels-idx1-ubyte.gz to ./MNIST/raw/train-labels-idx1-ubyte.gz
```

```
29696/? [00:00<00:00, 1014016.16it/s]
```

0 1 Python 3 | Idle Mode: Command Ln 19, Col 1 MNIST_AI_MODEL_v1.ipynb

Figure 4-3. High-level view of what it looks like to train the traffic sign detection model using PyTorch inside an Oracle Data Science Jupyter Notebook.

Now that you are somewhat familiar with the model, you can consider how to tether it with blockchain.

> **TIP**
>
> How to use the Jupyter Notebook to modify the traffic sign detection model is explored in the exercises in Chapter 6.

Tethering the Model

The BTA keeps track of how various approvers impact a model, such as how a stakeholder affects the model's purpose and intended domain; how an AI engineer collects and manages training data and develops models using different algorithms; how an MLOps engineer tracks and evaluates inputs and outputs and reviews performance metrics, bias, optimal and performance conditions, and how AI and MLOps engineers create explanations. The BTA's scope could be expanded to include additional user roles, like an auditor.

The AI engineers can log in to the BTA and record details about model training events and experiments. This triggers workflow to MLOps engineers who are involved in the approval and deployment of the model. The workflow is such that an AI engineer can create a model in the BTA, record the results of certain experiments, and submit the model for review and approval. Then, the MLOps engineer user can test the model, review the results of the AI engineer's experiments, and give feedback. Once the MLOps engineer approves the results, the model is presented to the stakeholder for a final approval.

When submitting a model for approval, the BTA will prompt the AI engineer for a unique version number. As you can see in Figure 4-4, the BTA collects URLs for the log files, test and training data sets, model URL,

notebook version and code repository at the same time, which become tied to this version number.

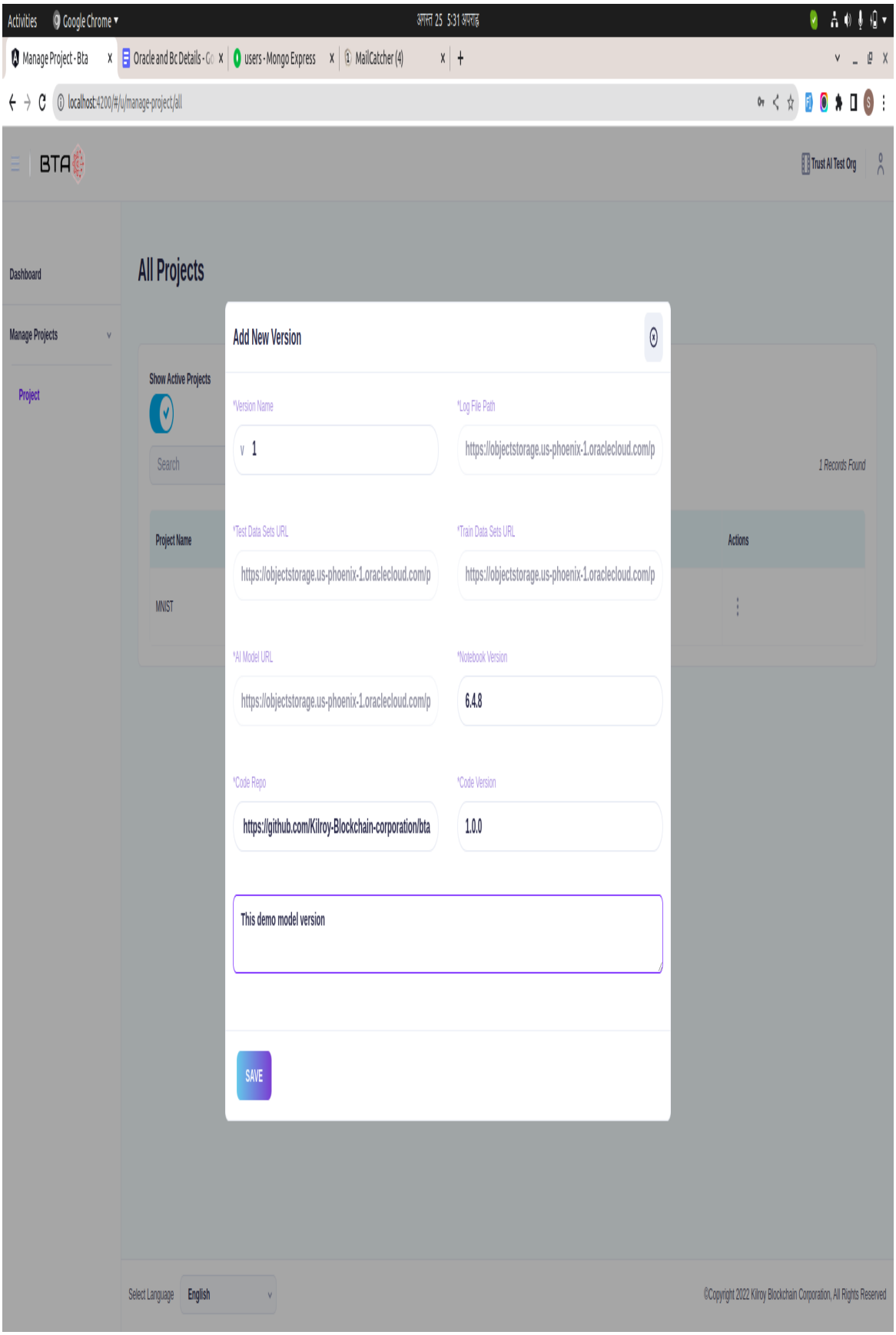

Activities
Google Chrome
अगस्त 25 5:31 अपराह्न
Manage Project - Bta
Oracle and Bc Details - Go
users - Mongo Express
MailCatcher (4)
localhost:4200/#/u/manage-project/all
BTA
Trust AI Test Org
Dashboard
Manage Projects
Project
All Projects
Show Active Projects
Search
1 Records Found
Project Name
Actions
MNIST
Add New Version
*Version Name
v 1
*Log File Path
https://objectstorage.us-phoenix-1.oraclecloud.com/p
*Test Data Sets URL
https://objectstorage.us-phoenix-1.oraclecloud.com/p
*Train Data Sets URL
https://objectstorage.us-phoenix-1.oraclecloud.com/p
*AI Model URL
https://objectstorage.us-phoenix-1.oraclecloud.com/p
*Notebook Version
6.4.8
*Code Repo
https://github.com/Kilroy-Blockchain-corporation/bta
*Code Version
1.0.0
This demo model version
SAVE
Select Language
English
©Copyright 2022 Kilroy Blockchain Corporation, All Rights Reserved

Figure 4-4. AI engineer assigning a version number to a model in the BTA.

When the Add New Version form is submitted, cryptographic hashes are created from this information and stored on blockchain, which creates a tamper-evident trail of the origin of the model.

Next, the experiment is run and the detailed outcome is recorded by the AI engineer. When the model arrives at the next step–review by the MLOps engineer–they will receive an email notification. When the MLOps engineer opens the link, they will see a Version Details screen like the one shown in Figures 4-5 and 4-6, along with the model details. This provides the MLOps engineer with the information they need in order to perform their own testing to form their own feedback about the model.

These details include hyper parameters and test metrics for the model for the most recent and past epochs. The MLOps engineer can review these details before deciding whether to approve or reject the model.

Dashboard

Manage Projects

Project

Experiment Name: exp_0

Model Details

Project Name: MNIST

Version Name: v1

Code Version: 1.0.0

Code Repo: https://github.com/Kilroy-Blockchain-corporation/bta-aimodel.git

Notebook Version: 6.4.8

Train Data Sets URL: https://objectstorage.us-phoenix-1.oraclecloud.com/p/cHydEc-kHJgjQUmUCrhPx3dd3DXO7S9JDmeeFJcCx9xl2zLJijxVmACQsfxu-3Sk/n/axutkjfnpof3/b/ds-1/o/MNIST/v1/artifacts/datasets/train_datasets.zip

Test Data Sets URL: https://objectstorage.us-phoenix-1.oraclecloud.com/p/cHydEc-kHJgjQUmUCrhPx3dd3DXO7S9JDmeeFJcCx9xl2zLJijxVmACQsfxu-3Sk/n/axutkjfnpof3/b/ds-1/o/MNIST/v1/artifacts/datasets/test_datasets.zip

Framework: N/A

Framework Version: N/A

Log File Path: https://objectstorage.us-phoenix-1.oraclecloud.com/p/cHydEc-

Hyper Parameters

data_dir: .

hidden_size: 64

learning_rate: 0.0002

Test Metrics

test_accuracy: 0.9355000257

test_f1_score: 0.9355095625000001

test_loss: 0.2117002755

test_precision: 0.9355095625000001

test_recall: 0.9355095625000001

Epochs

Epoch No	train_acc	train_f1_score	train_loss	train_precision	train_recall	val_accuracy	val_loss
0	0.808290988	0.808399973000001	0.643902402	0.808399973000001	0.808399973000001	0.9153481126	0.2884419858
1	0.912836729000001	0.9128391147	0.292299062	0.9128391147	0.9128391147	0.9353243709	0.2222530097

Figure 4-5. Model details, hyper parameters and test metrics by epoch are viewable in the BTA.

Dashboard

Manage Projects

Version Details

Project Name:	MNIST
Version Name:	v2
Log File Path:	https://objectstorage.us-phoenix-1....axutkjfnpof3b/ds-1/o/MNISTiv2/logs
Test Data Sets URL:	https://objectstorage.us-phoenix-1....rtifacts/datasets/test_datasets.zip
Test DataSets BC Hash:	9a44cbbe04b208e212a21157b71b051ec14e9178d6134371f354287c00bfaa88
Notebook Version:	6.4.8
Code Repo:	https://github.com/Kilroy-Blockchain-corporation/bta-aimodel.git
Code Version:	1.0.0
Version Status:	MLOPs Review
Comment:	This model version has high accuracy
Created Date:	Thursday, August 25, 2022
Created By:	Deepak Mlops

Log File Data

Log File BC Hash: 4e72194307f0b04f1494f26b943901b89839ff04d9543564986b7957191ce8f3

Exp Id/No	Date	Actions
exp_0	2022-08-25	

Figure 4-6. The Version Details screen shows the details of a model in the BTA with the status of MLOps Review.

As the MLOps engineer completes their review, the AI engineer should be able to see their updates in the Review Report and Monitoring Report.

> **TIP**
>
> If this were a real scenario, you would want to also include *peer reviews*, which means that in addition to the MLOps engineer and the stakeholder, other AI engineers could add their reviews of a model.

The AI engineer reviews and implements the required changes and submits the model again for the review. This cycle continues until the stakeholder accepts the product or until some other limitation, like permitted time or budget, has been reached.

The model is tethered by permanently storing cryptographic hashes of every approval, log of the model, and links to versions of the model. This is done from the time of the model's submission for review, to testing the model and validation, as many times as the model might go through this cycle. The blockchain technology allows someone to later prove whether or not the model is properly trained by tracking and tracing the model's provenance through the BTA audit trail.

Any objects, like the model itself or the logs, are stored off-chain in an object store. A link to the object is stored on-chain, along with a timestamp and a cryptographic hash that can be recomputed to provide proof of the object's integrity.

A system like this could be built without blockchain, using the other workflow layers without the tamper-evident audit trail that blockchain provides. By adding the blockchain layer to the stack, any changes made by bad or sloppy actors to the model, whether the actors are human or AI, will be exposed.

This is because if one block's data is changed, the chain breaks, because the hash representing the data in the block will no longer resolve to its previous value. You have multiple copies, or nodes, of the blockchain distributed amongst the participants of your blockchain network, which means no person or system can wipe out the record of what took place. As a result, your audit trail is tamper-evident, which only blockchain makes possible.

After the model is reviewed and approved, it is deployed onto a production server and closely monitored by the MLOps engineer and stakeholder in an continuous feedback, improvement and maintenance loop.

The BTA's records help users with an oversight role to be certain of the provenance of the model, including who has trained the model, what datasets are used, how the model got reviewed, the model's accuracy submitted by the AI engineer compared to the model's accuracy submitted

by the MLOps engineer, and the feedback given by peers. The BTA can be used to produce a blockchain-based audit trail of all steps and all actions done during the development and launching of the model, which is helpful to stakeholders and auditors.

> **NOTE**
>
> People may switch jobs, organizations may change policies; still, the current policies need to be evaluated in order to allow or disallow a model from passing into production. Building these requirements into a BTA will help to catch, or even prevent, mistakes that result from organizational changes.

FOUR CONTROLS

As part of planning how to tether the traffic sign model, consider how your BTA meets each of the four controls mentioned in Chapter 2.

- *Control 1:* This criteria is met, because your users all have certificates tied to their identity and the workflow is built into the BTA.
- *Control 2:* This is met because the MLOps engineer reviews the submitted model and verifies the accuracy of the model.
- *Control 3*: This is met because peer reviewers and stakeholders review the deployed model, govern it and give feedback. If the review exceeds the confidence score of the previous iteration, and the accuracy is rated higher than the model currently running in production, then the new model is launched into production and will undergo monitoring.
- *Control 4:* This is met because the lifecycle of the model is recorded on blockchain and can be checked by viewing a report.

Subscription

The BTA is *multi-tenant*, which means that different organization accounts, with different users, can exist in one implementation without impacting one another. Brand new BTA accounts are created by adding a new subscription. Subscription requests are initiated by the new organization admin on the BTA signup page, as Chapter 7 explains.

When you create a new subscription in your test BTA, make sure you do it with the email address that you plan to use for the organization admin. For your test, use any organization name and any physical address that you like.

After you submit your request for a new subscription, the *super admin*, or built-in user that oversees all subscriptions, will be able to approve it and allow your organization admin to use the BTA to create users, upload and approve models, check the audit trail, and to delegate these functions and features to other users.

At the time of subscription approval, the super admin adds an Oracle Bucket to the new subscription, and creates the required blockchain node, using the Oracle Cloud account for the organization. Chapter 7 explains the complete steps for adding and approving a subscription, and for integrating it with the Oracle Cloud. Now that the account is set up, the organization admin can start working access control and create organization units, staffings and users.

Access Control

Access control by way of user permissions is the cornerstone of a multi-user, multi-tenant software-as-a-service like the BTA. When different users log in, they can see different features and perform different functions within the BTA. This is accomplished through creating organizations and staffings that define the level of access, and assigning them to users. Chapter 7 has more detail on how to set up access control and users. This section provides an overview to give you a high-level understanding of how access is managed.

Organization Units

Organization units are intended to be logical groupings of features that are assigned to people with similar roles. In your BTA you have an organizational unit called AI engineers, another called MLOps engineers, and another called stakeholders. Within these organization units sets of features that fit the role are selected, making these features available to any user that is assigned to this organization unit.

Staffings

Staffings fall within organization units and are intended to mitigate users' create-read- write-update access within the features. For this BTA example we only have one staffing for each organization unit. In a production scenario you may have multiple staffings within each organization unit. Figure 4-7 shows a BTA mockup of the organization staffing for an organization called Test Org.

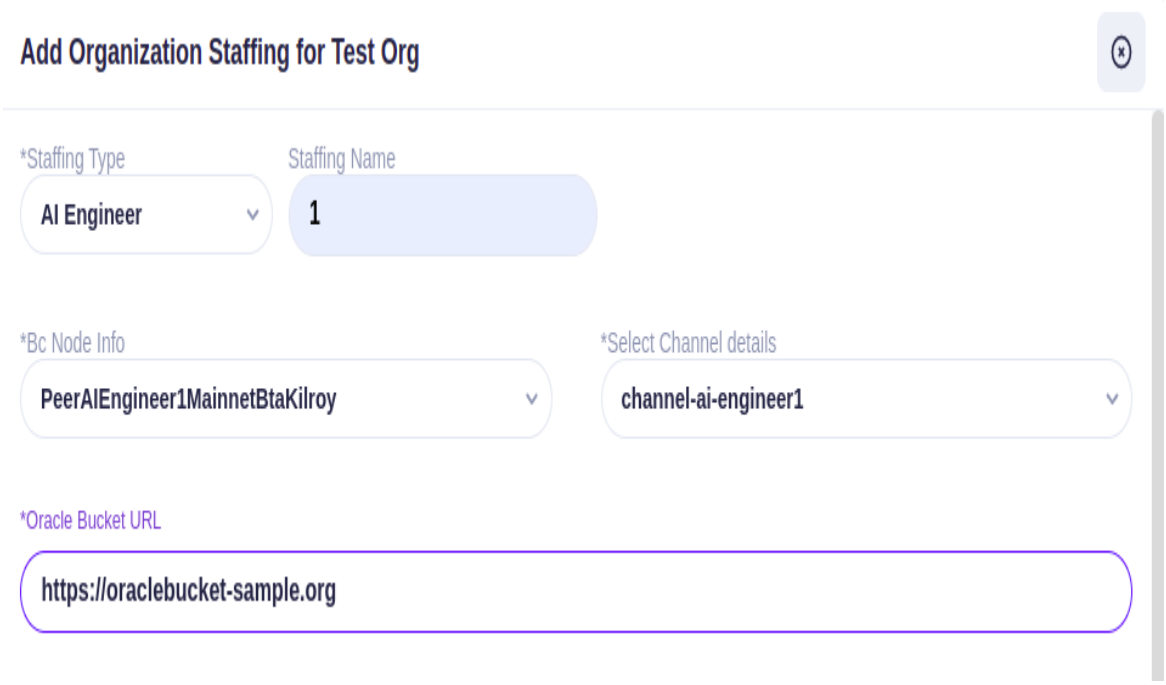


Figure 4-7. Staffing screen 1 of 2 allows definition of a blockchain node and channel, and an Oracle Bucket URL.

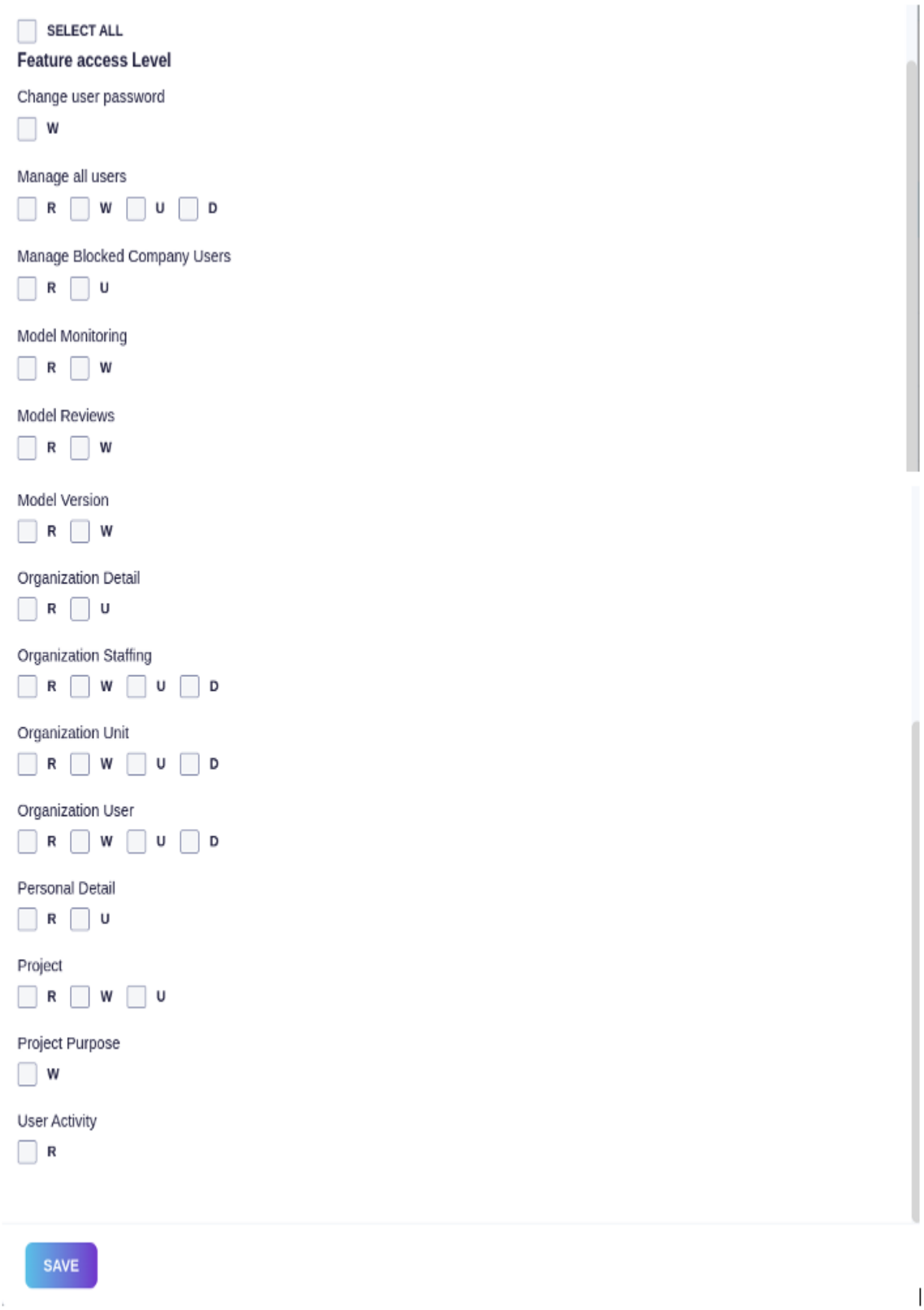


Figure 4-8. Staffing screen 2 of 2 allows selection of Read-Write-Update-Delete permissions for the features selected for staffing's organization unit.

> **TIP**
>
> If you don’t see the feature in the staffing list, make sure it is enabled in the staffing’s organization unit.

Users

The BTA offers role-based access control, implemented via units called organizations and staffings. This way, the depth of the information displayed to any individual user can be controlled by placing that user in an organization and staffing appropriate to their level of authorized access. It is critical to create traceable credentials and well-thought-out permission schemes to have a system that lets the right participants see the right information, and keeps the wrong participants out.

As you have read throughout this chapter, there are five types of users that you will use in your test BTA. They are super admin, organization admin, stakeholders, AI engineers, and MLOps engineers.

When a user is added, they get login credentials from the BTA application. These details include a user name, a password, and a blockchain key generated at the blockchain end using SHA-256 hashing algorithm that hashes the user’s wallet info and creates the key.

> **TIP**
>
> If this were a production project, some good questions to ask yourself when determining participants are: Who is on the team? How are they credentialed? What sort of organizations and roles should they be grouped into? What is the procedure when someone leaves or joins?

Super admin

The super admin is a special user that is automatically created when you instantiate your BTA. Super admin can inspect any subscriber’s authenticity and either verify or decline any organization’s subscription. Super admin is

assigned its own blockchain node, where all approved or declined subscription requests are stored.

Organization admin

Once the subscription is approved by the super admin, the BTA allows the new organizational admin to log in. An organization admin has permission to manage the users for their subscription by assigning the correct roles and permissions. The organization admin plays an important role in using the BTA by creating the project, organization units (departments), staffings and users.

The organization admin creates all required organization units and permissions before creating the users. While different kinds of permissions for model creation, submission, review, deployment, and monitoring can be set in the BTA, these exercises suggest that you follow the steps outlined in Chapter 7 for setting the user permissions so you have a system that will work properly; then you can safely make changes to test new ideas.

> **TIP**
>
> IMPORTANT: After creating all required permissions, the organization admin needs to create at least one AI engineer, MLOps engineer and stakeholder to start a project. Since these users must approve or reject a model, a project cannot be created unless one of each type of user is created. In order to let the user work with the blockchain, a blockchain peer and channel name should be assigned to each user by the organization admin.

The organization admin can revoke a user's access and privileges from the entire project. They can disable the user in both the BTA application and Oracle cloud account, and remove the permission (staffing) from the user account.

However, it is just a soft delete; the user's data is not deleted anywhere from the application, nor does the blockchain node get deleted. Instead, the user is added to the revocation list, which means the user will not be able to access the node if they try. Access in the blockchain is only possible if the user passes out of the revocation list.

AI engineer

After the organization admin creates the AI engineer, the user receives an activation email. The email contains the username and password. Additionally, it also contains the blockchain key which is created using the user's blockchain wallet info like a public key. SHA-256 algorithm is used to create the key. In the sample project in this chapter, the user needs to enter the blockchain key to pass the second layer of security of the application, as illustrated in Figures 4-9 and 4-10.

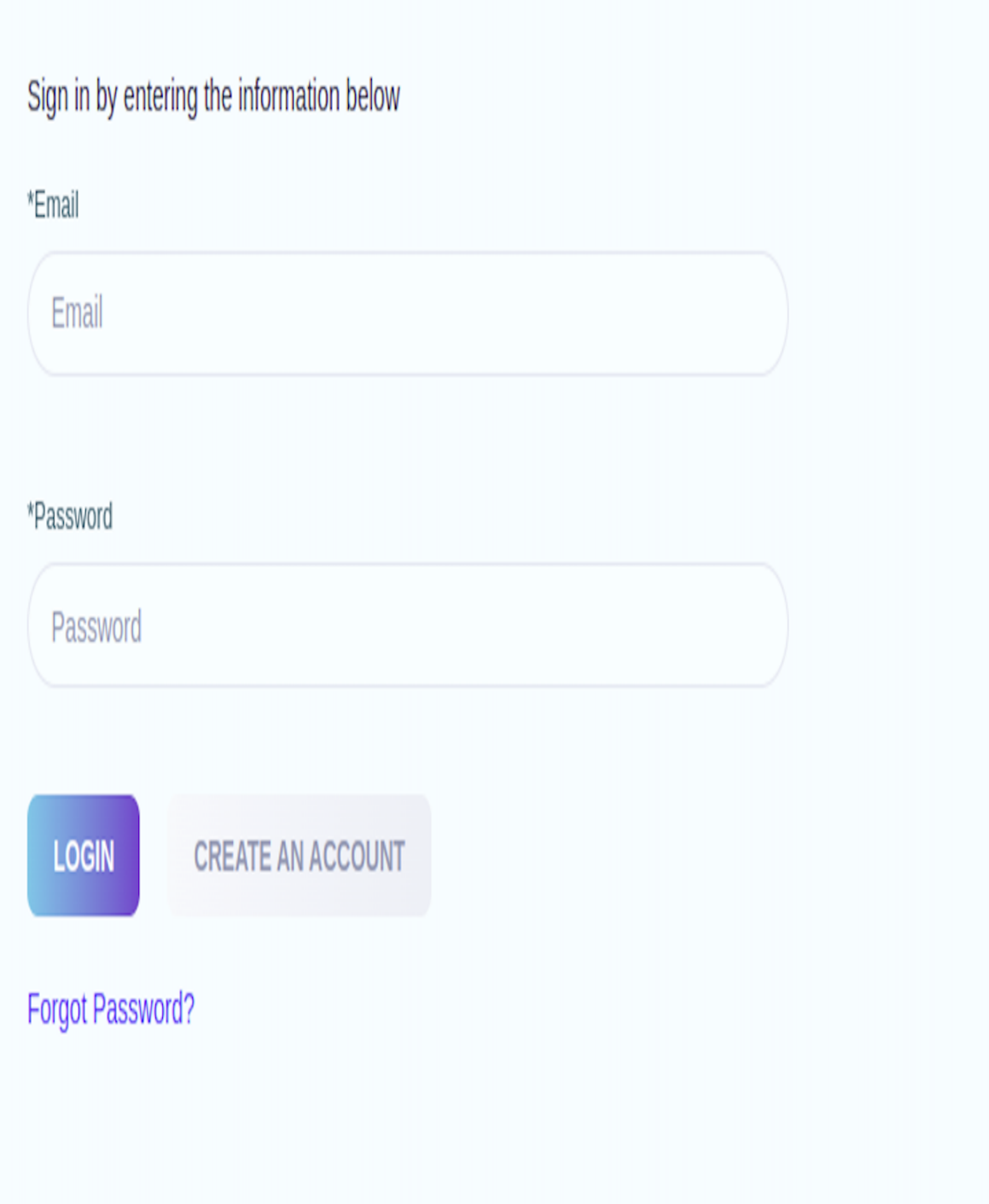


Figure 4-9. Login page

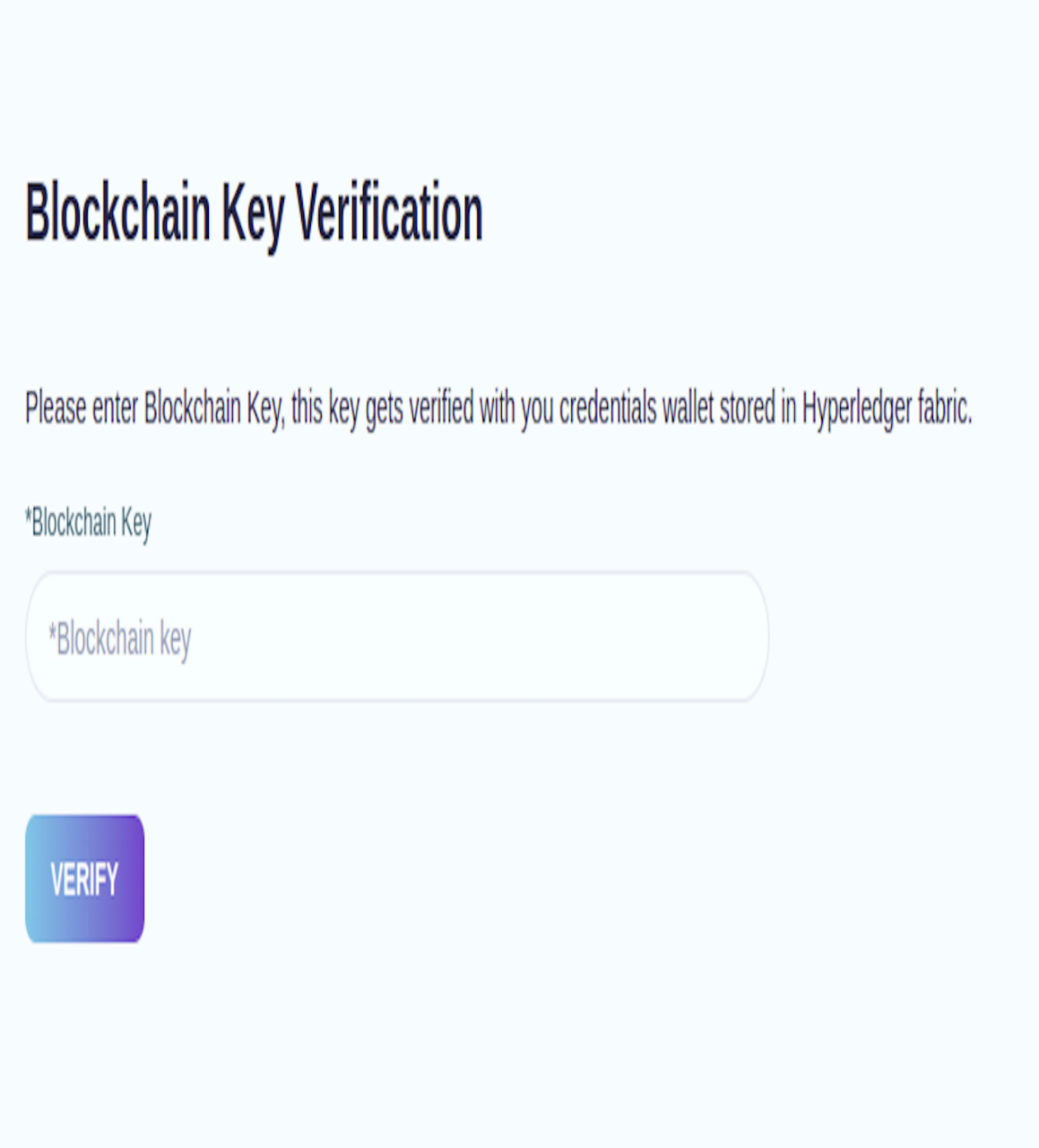


Figure 4-10. Blockchain key verification page after passing first login

This feature is built to address Blockchain Control 1: Pre-establish Identity and Workflow Criteria for People and Systems. The identity created in blockchain is used in the web application through the Blockchain Key Verification page.

TIP

When the account in Oracle Cloud is created by the organization admin, it sends an email to the user which includes Oracle Cloud login credentials. These details will have to be entered into the user's configuration so they can be passed to the cloud by the BTA.

After the activation of the BTA application account and Oracle cloud account, the user logs into the BTA application where all AI projects assigned to them are listed. The engineer should be able to start model development using Oracle cloud based data science resources. After a model is created, the engineer can login into the BTA web application, select a project assigned to them and start creating a new version of the recently built model. In the sample project, the user can create a new version of the model but keep it as a draft. Draft status allows the user to come back and change the model info in the web application. Once the model is ready to submit, the user can submit it for review by MLOps engineer. After submitting the model, the user cannot change anything in the submitted version.

After creating a model, the log file should be stored in the user's bucket in the following folder hierarchy:

- Project Name
    - Version Name (V1)
        - Artifacts
            - Train Dataset
            - Test Dataset
        - Log
            - Exp1.json
            - Exp(n).js

MLOps engineer

The MLOps engineer reviews and deploys the submitted models, and can see all projects in the BTA, including multiple versions of each project. The MLOps engineer downloads the model and runs it in their own data science environment, where test data and logs are stored in the user's own bucket following the folder format discussed in the AI engineer's role.

Before the review process starts, the MLOps engineer makes sure that the dataset submitted by the AI engineer has not changed by comparing the hash of logs and artifacts between the current versus before the run, as shown in Figure 4-11. If the hash verification passes it proves that artifacts and logs have not been tampered with, thus addressing blockchain control 2.

Dashboard

Manage Projects

Project

Verify Bc Hash

Model Version Bc Details

Project Name: MNIST

Version Name: v1

Created By: deepak+ai-eng@kilroyblockchain.com

Creator MSP: PeerOSAIEngineerBtaKilroyMSP

Verify Bc Hash

	Submitted Data	Oracle Data	Status
Log File BC Hash	9275e1a764a87d19b5a9be57a7b87e79af1a724b9dd20dc97054a9b92289febd	9275e1a764a87d19b5a9be57a7b87e79af1a724b9dd20dc97054a9b92289febd	Verified
Test DataSets BC Hash	9a44cbbe04b208e212a21157b71b051ec14e9178d6134371f354287c00bfaa88	9a44cbbe04b208e212a21157b71b051ec14e9178d6134371f354287c00bfaa88	Verified
Train Data Sets BC Hash	1096f55547a3d7ad1931ee32cb25bf10f8cf4185761 0b6f64cedaec7c7e4f018	1096f55547a3d7ad1931ee32cb25bf10f8cf4185761 0b6f64cedaec7c7e4f018	Verified
AI Model BC Hash	09f78ff847efa3434e92f6481b70f1f87783d84db68c7538cd7cde29529a66ff	09f78ff847efa3434e92f6481b70f1f87783d84db68c7538cd7cde29529a66ff	Verified

Figure 4-11. Verify Bc Hash screen in BTA gives a comparison of blockchain hashes and hashes of current bucket data to ensure data tampering has not taken place.

> **TIP**
>
> As with the AI engineer the MLOps engineer should activate two accounts before starting. These are the Oracle account and the BTA application account. With successful account activation and login, the user gets access to the Oracle Cloud Data Science resources, the BTA application and the blockchain peer.

When the review process starts, the MLOps engineer can change the status of the model to 'Reviewing' so that the rest of the participants of the project know that the review process has begun. After reviewing the model, the MLOps engineer can change the status of the model to 'Review Passed' or 'Review Failed.' (A model gets 'Review Passed' if the accuracy of the model reviewed by the MLOps engineer is greater than the AI engineer's accuracy retrieved from the submitted log file.)

Once the review passes, the MLOps engineer changes the status to 'Deployed' and releases a testing/deployed URL for the rest of the users. After the model is reviewed, the MLOps engineer can change the model's status to deployed. They can add ratings of the model, production URLs, review version number, or add a log and test data set that is pulled by the script based on the version number.

By adding review details by a MLOps engineer, the stakeholder or end user knows why the model was approved and deployed. This helps in governance and auditing of the AI model, which addresses blockchain control 3.

Additionally, a team review helps to make the model more authentic. Screenshots of a bug or issue with the model can be compiled into one document and uploaded to the BTA. The model's status can be set to QA to indicate it is under review, which is noted on blockchain. This creates a tamper-evident audit trail of deployment issues. The MLOps engineer can then launch the model to production if reviews after deployment are

positive. The engineer should then change the status of the model to “Production.” They can also update the status to “Monitoring,” which allows for feedback if bugs or other issues are found in the launched AI model.

Stakeholder

Stakeholders also pass through the registration and account activation process like the AI and MLOps engineers but with different staffing assignments. Stakeholder users get the staffing permissions that allow the user to Review/Complete/Cancel the AI model. The stakeholder gets access to the blockchain node but it is not necessary to get access to the Oracle Cloud account as the user is not going to develop any AI model. The stakeholder user, as the decision maker, can either cancel or complete the project

The stakeholder is able to review the deployed model and monitor the production based model. The stakeholder is able to see who has submitted the model, the accuracy of the model, and compare the hashes of the logs and artifacts before and after the deploy to prove the provenance of the artifacts. All this information is pulled from the blockchain node along with the transaction id and timestamp, so it addresses blockchain control 4.

The stakeholder can add a purpose for the model as shown in Figure 4-12, and review, monitor, audit and approve or decline the model.

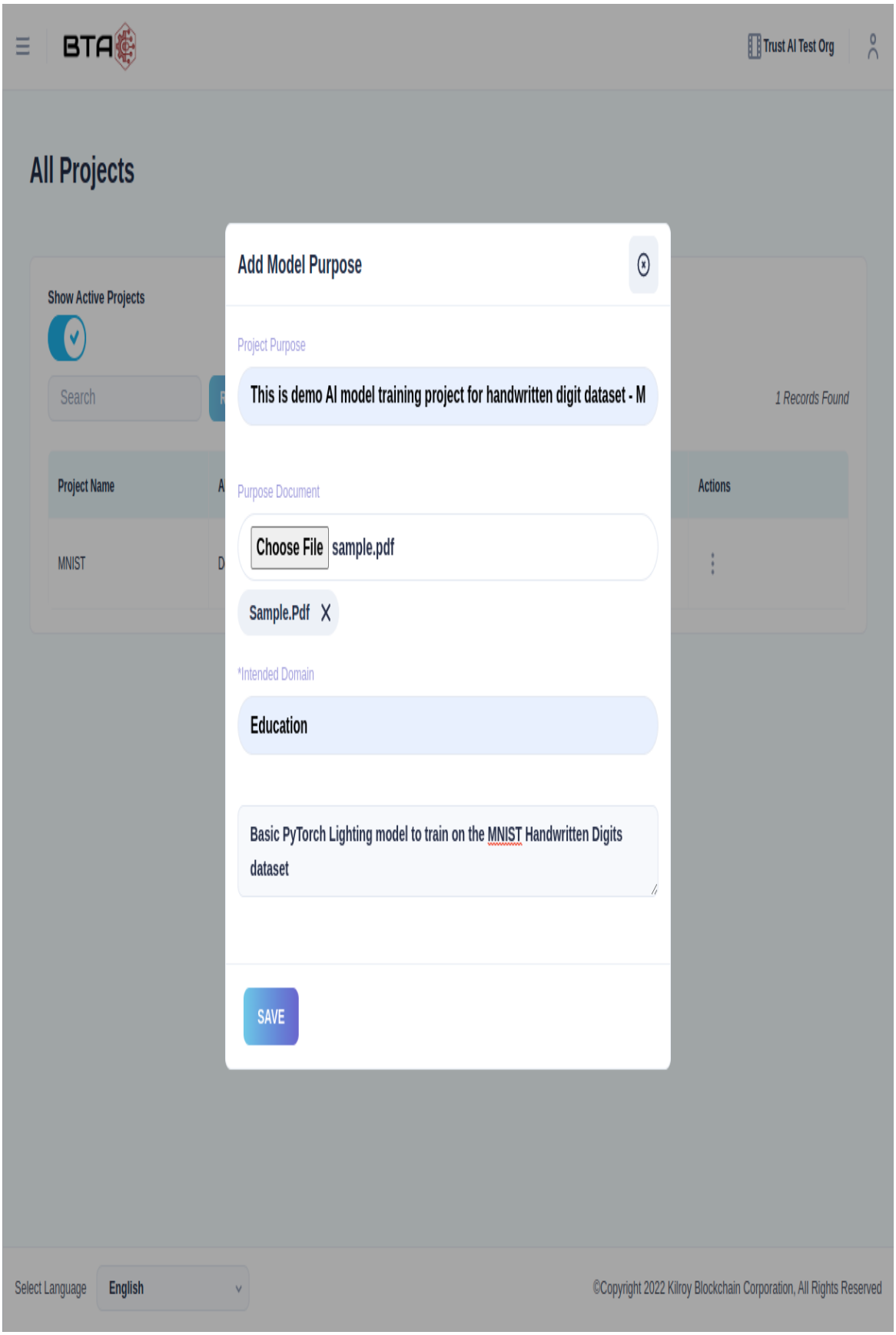
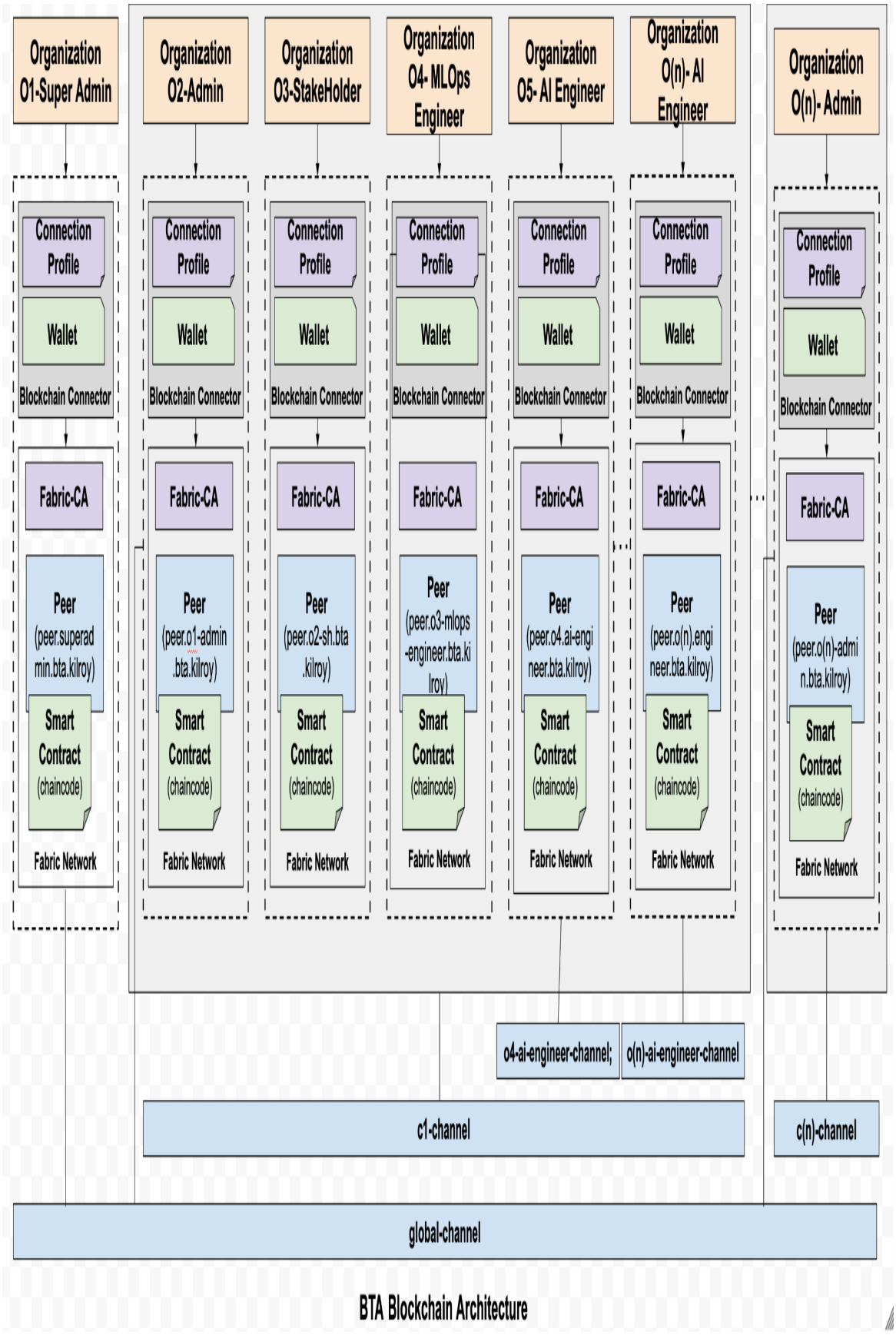

BTA
Trust AI Test Org
All Projects
Show Active Projects
Search
1 Records Found
Project Name
Actions
MNIST
Add Model Purpose
Project Purpose
This is demo AI model training project for handwritten digit dataset - M
Purpose Document
Choose File
sample.pdf
Sample.Pdf X
*Intended Domain
Education
Basic PyTorch Lighting model to train on the MNIST Handwritten Digits dataset
SAVE
Select Language
English
©Copyright 2022 Kilroy Blockchain Corporation, All Rights Reserved

Figure 4-12. A stakeholder adds a purpose for a model in the BTA

Blockchain Analysis

Chapter 1 discussed how to start a blockchain use case by analyzing the participants, assets, and transactions of your proposed blockchain process. This section describes how to analyze the participants, assets and transactions specific to your BTA use case, and how to address both business logic and technical requirements of the BTA's blockchain component.

Figure 4-13 shows different blockchain peers and channels used to create blockchain networks for the BTA project. Each participant of the blockchain network gets a peer where all activities like new model submission, review, deployment and production information are stored.



BTA Blockchain Architecture

Figure 4-13. BTA Blockchain architecture

The blockchain connector is a Node.js script installed in each peer, which receives data submitted by a web application and sends it to the peers. The connector also gets data from peers and sends it to the web application. The channel name plays an important role in choosing the peers that receive data. In the above figure, O4-ai-engineer-channel is a personnel channel of an AI engineer which stores the draft model. Another channel is c1-channel, which receives a newly submitted model by AI engineers and sends it to the MLOps engineer and AI engineer's peers.

BLOCKCHAIN AS A LAYER OF USER SECURITY

The access control, privacy and confidentiality mechanisms allow for confidential sharing, so that no unintended user can access restricted data or models. Blockchain brings an added feature to user management, in that it can record and also enforce security by means of smart contracts and agreed upon governance protocol. Tracking the order of exchanges helps to enable fair assessment of user contributions. Blockchain can be used to track ownership and usage across complex provenance chains.

Participants

A *participant* does some action that involves writing to the blockchain, while a *user* is anyone who logs into the BTA web user interface. As such, an AI engineer is both a user of the BTA and a participant in the blockchain network because their approvals and comments need to leave a tamper evident trail. An auditor user may not need to leave such a trail, because they are only reading records and auditing them. In this case, the auditor is a user of the BTA, but not a participant of the blockchain network.

Assets

Models, code repositories, logs, artifacts, and reports are the assets you will use when building the example BTA. Models are the AI models before and after training epochs and experiments. Repositories are where the code is stored. Logs are the temporary historical records that are produced during the epochs, similar to text logs found for any other computer system. Artifacts are previously used training and test data sets. Reports are produced by the BTA or by supporting systems.

You might think that the BTA is supposed to log all data for each and every activity on blockchain. However, your BTA does not require that all data be logged on blockchain and it does not record each and every transaction.

There is a reason for not storing all assets like logs and artifacts into blockchain and into applications: because the files themselves are too large and numerous, and storing the object there isn't a good use for blockchain. Also, because there might be millions of objects like these in an ML system, storing them in blockchain distributed nodes will ultimately slow down the blockchain network and affect the performance.

It is better to put objects into a cloud-based object store with a pointer to the object and a hash of the object stored on blockchain. To make sure that the dataset or log produced is tamper-evident (the info has not been changed from the date of submission), the BTA stores the hash of those files (log and artifacts) in blockchain using a SHA-256 hashing algorithm, as detailed in Chapter 2.

At the time of model verification, as this Chapter mentioned when describing their role, the MLOps engineer makes sure the hash of the log or data set they get while running the model in their instance is similar to the hash of those files submitted by an AI engineer. If the hash is not the same then it indicates that the data set or log info has been changed outside of the accepted workflow. In that scenario, the MLOps engineer will not deploy the model. Instead, it gets the status "Review Failed" and the AI engineer gets a chance to amend it based on the feedback from the MLOps engineer.

When the MLOps engineer or stakeholder reviews the model, the channel (global-channel) receives the data and sends it to the stakeholder, MLOps engineer, and AI engineer's peers. In this way data gets shared among different peers. This architecture is created for a project where each participant belongs to different organizations and grows when new projects are added. This architecture is responsible for achieving all four blockchain controls mentioned in chapter 2.

Transactions

As discussed in Chapter 1, a transaction is what occurs when a participant takes some significant action with an asset, like when an MLOps approves a model. When other participants comment on the model, those are also transactions, and are recorded on blockchain.

Usually, when you think through your transactions is where you will find the information that you need in the audit trail that you will store on blockchain, or blockchain touchpoints. Think of them as important points within the lifecycle of your model, time-stamped and recorded on the blockchain sequentially so they can be easily reviewed and understood.

Smart Contracts

As you learned in Chapter 1, the term smart contract is used to refer either to functions within the chaincode that drive the posting of blocks to the blockchain, or the business logic layer of the application that drives workflow.

In your BTA, smart contracts are used to automate the acceptance or rejection of the results of a model training event, and to kick off rounds of discussion about why the model is being accepted or rejected and what to do about it.

The workflow smart contracts used in your BTA only require any one participant to move something into a different status such as QA. In a non-test scenario, this workflow can be as complex as the business requirements, requiring a round of voting or for some other condition to be

filled (like QA has to be performed for 2 weeks before the model can be put into production).

Audit Trail

Chapter 1 explains in detail how to determine and document blockchain touchpoints. This section of Chapter 4 guides you through how to design the audit trail for your BTA.

The purpose of adding blockchain to an AI supply chain is to create a way to trace and prove an asset's provenance. Therefore, it is very important to make sure all of the critical events are recorded onto blockchain. The BTA project allows AI engineers to develop the AI model using whatever toolsets, frameworks, and methodologies they prefer. Their development activities are not logged into the blockchain because there would be tons of trial and error in the code which may not make sense to record because it would make the blockchain too big and unwieldy.

For this reason, the sample BTA focuses on the participants, assets and transactions surrounding the model submitted by the AI engineer to the MLOps engineer for review. Some project information from the AI factsheet, such as its purpose and key contacts, are part of the BTA-generated audit trail, as shown in Table 4-1. In this table you can see a timestamp of when each transaction was posted to blockchain, a transaction ID, an email address associated with the participant created the transaction, which Hyperledger Fabric Membership Service Provider was used for identity and cryptography, a unique ID for the project, detail about the project, and members.

Table 4-1. Project info storing in Blockchain

Timestamp	TxId	Added By	CreatorMSP
2022-07-10T16:44:24Z	52628...6d2365	OrgAdmin@yopmail.com	PeerOrg1MainnetBtaKilroyM
2022-04-11T16:44:24Z	726b7...6d2369	OrgAdmin@yopmail.com	PeerOrg1MainnetBtaKilroyM

The BTA also stores transactions that relate to when the model was tested, along with hashes of the log file, test data set, and training database, as shown in Table 4-2. By adding version status, this becomes an easy-to-understand audit trail of the training cycle of the model.

Table 4-2. AI models retrieved from Blockchain

project id
428b6c252a41ae5fa4562569

Timestamp	TxId	CreatorMSP	ve
2022-07-11T15:24:24Z	933d0...c7573	PeerMLOpsEngineerMainnetBtaKilroyMSP	60
2022-07-11T14:24:24Z	d091a....9221	PeerAIEngineerMainnetBtaKilroyMSP	60
2022-07-10T19:24:24Z	75bc....52946	PeerStakeHolderMainnetBtaKilroyMSP	62
2022-07-08T19:24:24Z	2e6b....9a4b	PeerMLOpsEngineerMainnetBtaKilroyMSP	60

The status of a trained model, its version details and the corresponding URL is also recorded in your BTA. Table 4-3 shows how this might look to a participant using your BTA to audit the AI.

Table 4-3. Model audit trail with timestamps and transaction id hashes

Version ID :	62ca8a8b3c1f1f70949d7562
versionName	v3.0
logFileVersion	v1.0
logFileBCHash	88d4266fd4e6338d13b845fcf289579d209c897823b9217da3e161936f031589
testDatasetBCHash	ba7816bf8f01cfea414140de5dae2223b00361a396177a9cb410ff61f20015ad
trainDatasetBCHash	36bbe50ed96841d10443bcb670d6554f0a34b761be67ec9c4a8ad2c0c44ca42c

Timestamp	TxId	CreatorMSP	Dep
2022-07-10T19:24:24Z	75bc7c8094....5c3d652946	PeerStakeHolderMainnetBtaKilroyMSP	-
2022-07-10T17:44:24Z	6c4a63e39f..2b7463cdf8b	PeerMLOpsEngineerMainnetBtaKilroyMSP	-
2022-06-15T13:45:24Z	17fa18dcab...df34rsdf984	PeerMLOpsEngineerMainnetBtaKilroyMSP	-
2022-06-27T13:45:24Z	27fa18dcab...76224d3b7	PeerStakeHolderMainnetBtaKilroyMSP	
2022-06-25T13:45:24Z	sdfsdf3dc....3234nisdfms	PeerMLOpsEngineerMainnetBtaKilroyMSP	
2022-06-23T13:45:24Z	34dfd32j....324234sdf32s	PeerMLOpsEngineerMainnetBtaKilroyMSP	
2022-06-16T16:44:24Z	476254942.....9b55efe32	PeerMLOpsEngineerMainnetBtaKilroyMSP	-
2022-06-15T12:40:24Z	6685c676bd.....4ad39c17	PeerMLOpsEngineerMainnetBtaKilroyMSP	-
2022-06-10T11:45:24Z	da23812033...670c3146f45	PeerAIEngineerMainnetBtaKilroyMSP	-

Part of auditing a blockchain is having the ability to check the hashes and request a change. This is addressed in Chapter 7, Using your BTA, where you can also find more screenshots that show how the BTA is intended to look and work.

Summary

Chapter 4 discussed the requirements of a BTA project and how the roles and responsibilities of the users should be constructed. Chapter 5 provides sample code and more about the model you will tether, which will help you to build and test your BTA.

Chapter 5. Preparing for Development

A NOTE FOR EARLY RELEASE READERS

With Early Release ebooks, you get books in their earliest form—the authors' raw and unedited content as they write—so you can take advantage of these technologies long before the official release of these titles.

This will be the fifth chapter of the final book.

If you have comments about how we might improve the content and/or examples in this book, or if you notice missing material within this chapter, please reach out to the editor at *ccollins@oreilly.com*.

In this chapter, you'll set up an environment so you can develop and test the blockchain tethered AI. To set up the BTA, you need to first create a development environment for the model. Alongside the model development environment, you will set up a development environment for the blockchain tethered AI workflow system, which consists of a bucket, a blockchain network, and a user-facing system that we call the BTA.

The two environments - one for the model and one for the BTA - share access to a single Oracle Bucket (referred to as the *bucket*) which is used as an object store. You integrate the two by supplying the blockchain and bucket credentials in the user's BTA configuration.

This chapter prepares you for Chapter 6, where you will code the project in detail.

Model

As you recall from Chapter 4, your sample model is a traffic sign detection system. For the BTA project, your model can be set up either in the local server or in the cloud. The important thing for either situation is to know that the bucket is a common tool where AI training logs and artifacts are stored. Your model and artifacts are stored in the bucket, which is shared with different participants through the BTA. (To review the main facts about your model, refer to Table 4-1).

Installation

To set up your AI environment in which to run your model, you will first need to install Anaconda and set up an Oracle bucket.

Installing and configuring Anaconda

First, you have to install and configure Anaconda, which is a package manager, an environment manager, and Python distribution that contains a collection of many open source packages. Required Python library packages such as numpy, pandas, scikit-learn, and scipy are pre-installed with Anaconda. You can use Anaconda's package manager, conda, or pip, to install additional packages. To install Anaconda, follow these steps:

1. At the Anaconda website choose the OS you are using that leads you to the download page from where you can download Python version installer of Anaconda.
2. Set up and install Anaconda by following the instructions on the setup screen.
    a. Installing on Linux:
        i. You can go through this link for more info https://bit.ly/3eqh4iA
        ii. In the terminal, run the following command to verify the installer's data integrity with following command:

```
shasum -a 256 /PATH/FILENAME
```

iii. Install Python 3.7 through the terminal: `bash ~/Downloads/Anaconda3-2020.05-Linux-x86_64.sh`

iv. Click Enter to review the license agreement. Then click and hold Enter to scroll.

v. Enter **`yes`** to agree to the license agreement. The installer prompts you to choose whether to initialize Anaconda Distribution by running conda init. It then finishes and displays, "Thank you for installing Anaconda<2/3>!"

vi. Close and re-open your terminal window for the installation to take effect, or enter the command `source ~/.bashrc` to refresh the terminal.

vii. After your installation is complete, verify it by opening Anaconda Navigator. Open a Terminal window and type **`anaconda-navigator`** to open Navigator where you can start a notebook to write the script.

b. Installing on Windows:

i. Go to your *Downloads* folder and double-click the installer to launch.

ii. Click Next. Read the licensing terms and click I Agree.

iii. It is recommended that you install for Just Me, which will install Anaconda Distribution to just the current user.

iv. After the installation, add Conda and Python to your path environment variables, per your operating system's instructions.

3. Now check the version of Conda and Python with the following commands in the terminal:

```
$ conda  -V
$ python -V
```

4. Go to the start menu and open the Anaconda Navigator.
5. Now launch the Jupyter Notebook, which will open in your default browser.
6. You can open the existing notebook (*.ipynb*) file or create a new notebook from the menu.
7. The notebook will now open in the new default browser.
8. Now you can start writing code in the newly opened notebook.

Bucket

Once you set up an AI development environment, now it's time to set up a bucket for the project. You read about the Oracle Bucket in Chapter 4, which is extensively used during experimenting, submitting, and reviewing a model.

You need a bucket in both environments to store logs and artifacts. Make sure your Oracle account has permission to access the bucket. To begin creating a bucket under your OCI tenancy, go to OCI Console > Object Storage > Buckets > Create Bucket (Figure 5-1).

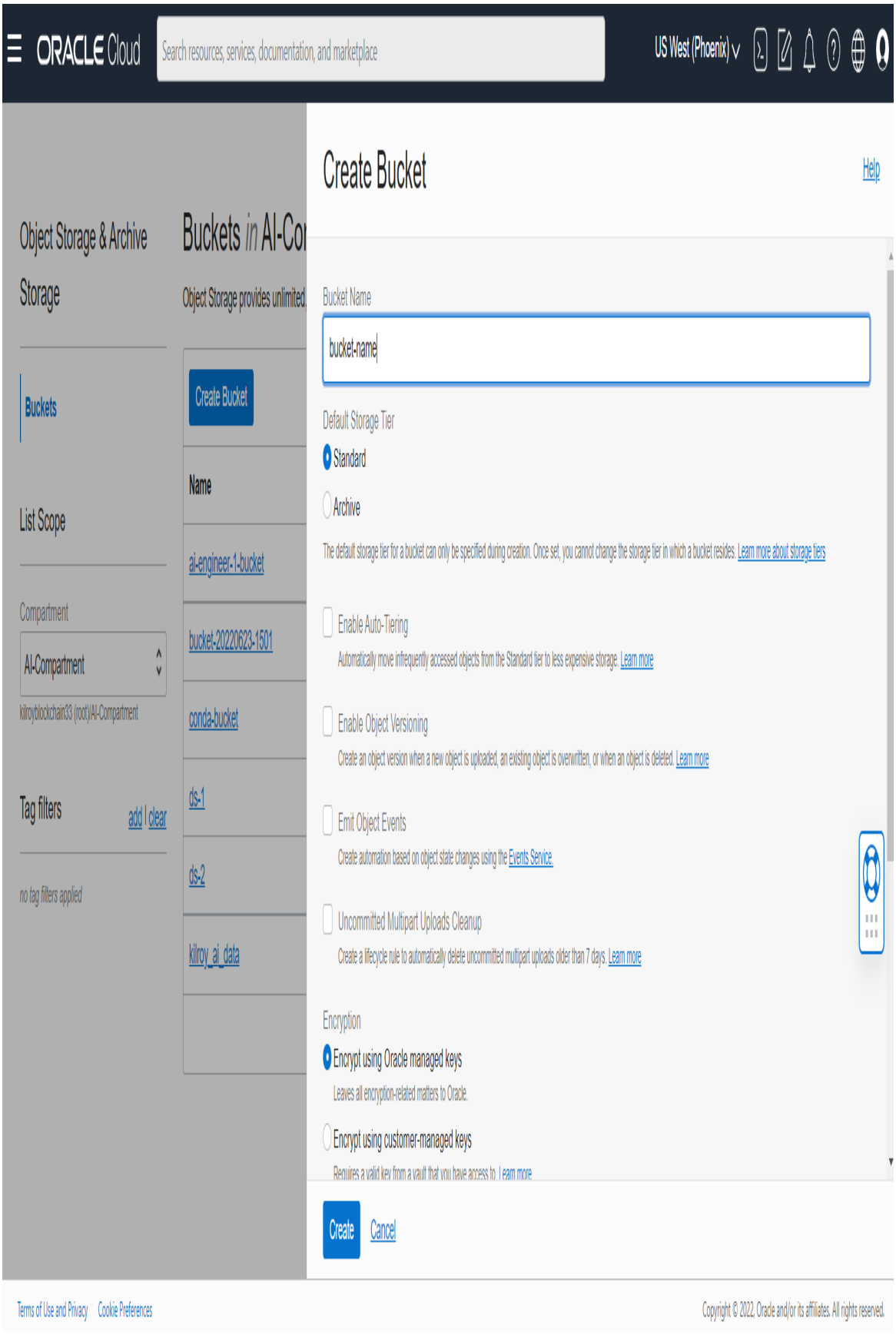

ORACLE Cloud
Search resources, services, documentation, and marketplace
US West (Phoenix)
Object Storage & Archive Storage
Buckets
List Scope
Compartment
AI-Compartment
kilroyblockchain33 (root)/AI-Compartment
Tag filters
add | clear
no tag filters applied
Buckets in AI-Co
Object Storage provides unlimited
Create Bucket
Name
ai-engineer-1-bucket
bucket-20220623-1501
conda-bucket
ds-1
ds-2
kilroy_ai_data
Create Bucket
Help
Bucket Name
bucket-name
Default Storage Tier
Standard
Archive
The default storage tier for a bucket can only be specified during creation. Once set, you cannot change the storage tier in which a bucket resides. Learn more about storage tiers
Enable Auto-Tiering
Automatically move infrequently accessed objects from the Standard tier to less expensive storage. Learn more
Enable Object Versioning
Create an object version when a new object is uploaded, an existing object is overwritten, or when an object is deleted. Learn more
Emit Object Events
Create automation based on object state changes using the Events Service.
Uncommitted Multipart Uploads Cleanup
Create a lifecycle rule to automatically delete uncommitted multipart uploads older than 7 days. Learn more
Encryption
Encrypt using Oracle managed keys
Leaves all encryption-related matters to Oracle.
Encrypt using customer-managed keys
Requires a valid key from a vault that you have access to. Learn more
Create
Cancel
Terms of Use and Privacy
Cookie Preferences
Copyright © 2022, Oracle and/or its affiliates. All rights reserved.

Figure 5-1. Bucket creation inside the Object Storage

The bucket has the following general properties:

- Namespace name
- Compartment name
- Created date
- ETags
- OCID

A secret key is needed to make a secure connection from Jupyter notebook to Oracle Object Storage. This key provides privileges to the user to perform updates to the target bucket. To generate the secret key, follow these steps:

1. Sign in to the OCI Console with your OCI credentials.
2. In the console, click the Profile icon on the top right corner of the screen.
3. From the profile menu, select User Settings to open the User Details page.
4. On the User Details page, click Customer Secret Keys under Resources on the left side of the page, and then click Generate Secret Key.
5. Specify a name for the secret key (for example, *ocisecretkey*), and then click Generate Secret Key.
6. Copy and save the secret key to a safe place, because it won't be shown again. The OCI Secret key will be *aws_secret_access_key* for AWS SDK authentication.

Blockchain Network

Setting up the blockchain network is the next step. You will only set up one node; your multiple BTA users will all use the same blockchain node for these exercises.

Prerequisites

For the development environment, this section discusses the prerequisites that are needed to run the blockchain network.

Operating system

The scripts to run the blockchain network are written for Ubuntu compatible OS. You need to have Ubuntu 18 or higher.

Containerization

Docker 20.10.12, docker-compose 1.29.2 is used in this project for the containerization of blockchain nodes. All the necessary nodes run on the Docker container. All the configurations are written on a *docker-compose* file. You can install Docker and docker-compose to run the blockchain network at this link.

Hyperledger Fabric 2.0

Hyperledger Fabric 2.0+ is used for the development environment. The Fabric nodes run inside the Docker container, so make sure when you download the Fabric images for installation, that you do so from inside the Docker container. You can read these detailed steps in Chapter 6, Exercise 2, Instantiate a Hyperledger Fabric Blockchain Instance.

Hyperledger Fabric requires certain nodes in order to run. These nodes have their own responsibilities. The nodes that will be used are as follows:

- *Orderer: hyperledger/fabric-orderer:2.4:* Orderer nodes are responsible for receiving the transaction and adding those transactions into blocks, and distributing them to peers across the network.

- *Peer: hyperledger/fabric-peer:2.4: Peer nodes* are responsible for holding ledgers and smart contracts. All the smart contracts are installed on those peers.
- *Command-line interface: hyperledger/fabric-tools:2.4:* The command-line interface is a tool that is used to interact with Fabric networks. All the channel creation, channel join, chaincode installation, and other transactions on the network are done through command-line interface.
- *Intermediate CA: hyperledger/fabric-ca:1.5.1:* Intermediate CA is used for registering users to the blockchain network. It creates certificates of the users for interacting with the blockchain network.

Golang

Golang version 1.18.2 is used for the chaincode development and chaincode installation. The Fabric-tools image mentioned in the preceding list will download the golang itself on the Docker container. No manual installation of golang is required.

Install, Configure, and Launch the Blockchain

To set up the blockchain environment, you must clone the sample BTA project from the git repo using the following command:

```
$ git clone https://github.com/Kilroy-Blockchain-corporation/bta-blockchain && cd bta-blockchain
```

After the repository is downloaded, the following steps will set up the blockchain environment:

1. Generate Certificate Authority certificate files
2. Generate Genesis configuration files
3. Generate Channel configuration files
4. Generate Anchor Peer configuration files

5. Download, Install and Run the necessary blockchain nodes

The details for each step are given in the following sections.

Generate Certificate Authority (CA) certificate files

We are using Fabric CA to generate all the necessary certificates. The following script will download all required certificates for different users:

```
$ ./scripts/generateCertificates.sh
```

The script generates the following certificates:

- TLS CA
- Organization CA for Orderer Organization
- Organization CA for Super Admin Organization
- Intermediate CA for Super Admin Organization
- Organization CA for Company Admin Organization
- Intermediate CA for Company Admin Organization
- Organization CA for MLOps Engineer Organization
- Intermediate CA for MLOps Engineer Organization
- Organization CA for StakeHolder Organization
- Intermediate CA for StakeHolder Organization
- Organization CA for AIEngineer Organization
- Intermediate CA for AIEngineer Organization
- MSP and TLS Certificate for Orderers
- MSP and TLS Certificate for SuperAdmin Peer
- MSP and TLS Certificate for Company Admin Peer

- MSP and TLS Certificate for MLOps Engineer Peer
- MSP and TLS Certificate for AI Engineer Peer

The TLS CA secures communications between all the nodes on the network. Organization CA is an organization's identity enrollment CA that is created for each organization which enrolls the identities participating in the blockchain network. Intermediate CAs are used to hide the exposure of the Root CA (Organization CA in our case). Intermediate enrolls the identities participating in the blockchain network on behalf of the Root CA.

Generate genesis configuration file

Before installing the fabric network, we need to generate the genesis block. The genesis block is generated by following script:

```
$ ./scripts/generateGenesisBlock.sh
```

After the generation, the confirmation message will pop up on the terminal like this:

```
2022-08-23 14:36:23.732 UTC [common.tools.configtxgen] doOutputBlock -> INFO
005 Generating genesis block
2022-08-23 14:36:23.732 UTC [common.tools.configtxgen] doOutputBlock -> INFO
006 Writing genesis block
++ res=0
++ set +x
-----------------------------------------------
-----------------------------------------------
Successfully generated genesis block
-----------------------------------------------
-----------------------------------------------
```

Generate channel configuration files

The next step is to generate the channel configuration files. Run the channel configuration generation script:

```
$ ./scripts/generateChannelConfiguration.sh
```

After the generation, the confirmation message will pop up on the terminal like this:

```
2022-08-23 14:36:23.799 UTC [common.tools.configtxgen] doOutputChannelCreateTx
-> INFO 003 Generating new channel configtx
2022-08-23 14:36:23.800 UTC [common.tools.configtxgen] doOutputChannelCreateTx
-> INFO 004 Writing new channel tx
++ res=0
++ set +x
-----------------------------------------------------------------
-----------------------------------------------------------------
Successfully generated channel configuration files
-----------------------------------------------------------------
-----------------------------------------------------------------
```

Generate anchor peer configuration files

Next, generate anchor peers using the following script:

```
$ ./scripts/generateAnchorPeer.sh
```

After the generation, the confirmation message will be displayed:

```
2022-08-23 15:55:46.982 UTC [common.tools.configtxgen]
doOutputAnchorPeersUpdate -> INFO 003 Generating anchor peer update
2022-08-23 15:55:46.983 UTC [common.tools.configtxgen]
doOutputAnchorPeersUpdate -> INFO 004 Writing anchor peer update
++ res=0
++ set +x
-----------------------------------------------------------------------
```

Download, install, and run the necessary blockchain nodes

The last step is to download the Docker images for all the nodes and run those nodes on the Docker container. Here is the script that runs all the nodes:

```
$ ./scripts/runNodes.sh
```

The following output will be shown after the nodes are completely running:

```
WARNING: Found orphan containers (ica.o2-admin.bta.kilroy, cli.o5-ai-
engineer.bta.kilroy, cli.o3-sh.bta.kilroy, orderer1.org.bta.kilroy,
orderer2.org.bta.kilroy, cli.o4-mlops.bta.kilroy, orderer0.org.bta.kilroy,
ica.o3-sh.bta.kilroy, ica.o1-super-admin.bta.kilroy, cli.o2-admin.bta.kilroy,
ica.o4-mlops.bta.kilroy, ica.o5-ai-engineer.bta.kilroy, cli.o1-super-
admin.bta.kilroy) for this project. If you removed or renamed this service in
your compose file, you can run this
command with the --remove-orphans flag to clean it up.
Starting peer0.o4-mlops.bta.kilroy       ... done
Starting peer0.o1-super-admin.bta.kilroy ... done
Starting peer0.o3-sh.bta.kilroy          ... done
Starting peer0.o5-ai-engineer.bta.kilroy ... done
Starting peer0.o2-admin.bta.kilroy       ... done
-e
------------------------------------------------
------------------------------------------------
Successfully Installed and Run Nodes
------------------------------------------------
------------------------------------------------
-e
```

You can check the running nodes using the command `docker ps`.

The following nodes will be run on Docker containers:

```
$ docker ps

CONTAINER ID   IMAGE                             COMMAND
CREATED          STATUS          PORTS                                NAMES
cabd5562667b   hyperledger/fabric-tools:2.4     "/bin/bash"              6
minutes ago    Up 38 seconds
cli.bta.kilroy
8436b2310ff9   hyperledger/fabric-orderer:2.4   "orderer"                19
minutes ago   Up 41 seconds   7050/tcp, 0.0.0.0:9050->9050/tcp
orderer2.org.bta.kilroy
2d6aa2ad26b3   hyperledger/fabric-orderer:2.4   "orderer"                19
minutes ago   Up 41 seconds   0.0.0.0:7050->7050/tcp
orderer0.org.bta.kilroy
838403e3e313   hyperledger/fabric-orderer:2.4   "orderer"                19
minutes ago   Up 41 seconds   7050/tcp, 0.0.0.0:8050->8050/tcp
orderer1.org.bta.kilroy
7df10476d386   hyperledger/fabric-peer:2.4      "peer node start"        23
minutes ago   Up 26 seconds   7051/tcp, 0.0.0.0:7071->7071/tcp   peer0.o3-
sh.bta.kilroy
7777ec8d5fe3   hyperledger/fabric-peer:2.4      "peer node start"        23
minutes ago   Up 26 seconds   0.0.0.0:7051->7051/tcp             peer0.o1-
```

```
super-admin.bta.kilroy
cdaa25459874   hyperledger/fabric-peer:2.4      "peer node start"        23
minutes ago   Up 26 seconds   7051/tcp, 0.0.0.0:7061->7061/tcp   peer0.o2-
admin.bta.kilroy
3b36f3035c2b   hyperledger/fabric-peer:2.4      "peer node start"        23
minutes ago   Up 26 seconds   7051/tcp, 0.0.0.0:7091->7091/tcp   peer0.o5-ai-
engineer.bta.kilroy
7fe88a580ffa   hyperledger/fabric-peer:2.4      "peer node start"        23
minutes ago   Up 26 seconds   7051/tcp, 0.0.0.0:7081->7081/tcp   peer0.o4-
mlops.bta.kilroy
8505441fb820   hyperledger/fabric-ca:1.5.1      "sh -c 'fabric-ca-se..."   8
days ago       Up 34 seconds   7054/tcp, 0.0.0.0:7064->7064/tcp   ica.o2-
admin.bta.kilroy
8eae6fedae0b   hyperledger/fabric-ca:1.5.1      "sh -c 'fabric-ca-se..."   8
days ago       Up 34 seconds   7054/tcp, 0.0.0.0:7094->7094/tcp   ica.o5-ai-
engineer.bta.kilroy
1a8a12ba61ca   hyperledger/fabric-ca:1.5.1      "sh -c 'fabric-ca-se..."   8
days ago       Up 34 seconds   7054/tcp, 0.0.0.0:7084->7084/tcp   ica.o4-
mlops.bta.kilroy
5b8180468331   hyperledger/fabric-ca:1.5.1      "sh -c 'fabric-ca-se..."   8
days ago       Up 33 seconds   7054/tcp, 0.0.0.0:7074->7074/tcp   ica.o3-
sh.bta.kilroy
12e44fe0f392   hyperledger/fabric-ca:1.5.1      "sh -c 'fabric-ca-se..."   8
days ago       Up 34 seconds   0.0.0.0:7054->7054/tcp             ica.o1-
super-admin.bta.kilroy
```

BTA

Your BTA has a backend and a frontend layer. The installation instructions for each development environment follow.

BTA Backend

To set up the backend, or server, of the BTA application, first you should have the backend of the BTA application in your system. You need to download the BTA backend part using `git clone`:

```
$ git clone https://github.com/Kilroy-Blockchain-corporation/bta-
app
$ cd bta-api
```

After you install the backend of the BTA application on your system, you can set up the server according to the development environments discussed in the following sections.

Install Docker

The server of the BTA project uses Docker for automating the managing, developing, and deploying of applications in lightweight virtualized environments. In the developing phase of the BTA application you used Docker version 20.10.17 on the Ubuntu 20.04.4 LTS operating system. So, you need to install Docker in your operating system.

You can follow the Docker installation process here. Docker builds images (acting as a set of instructions to build a Docker container) automatically by reading the instructions from a Docker file:

```
bash ~/Downloads/Anaconda3-2020.05-Linux-x86_64.sh
```

Configure the .env file

In the BTA application's root folder, you have a file called *.env-sample* which contains all the necessary environment variables. This file is an environment variables file containing key/value pairs defining the project's required environment variables. To set all necessary environment variables to the BTA application, you need to create a file named *.env* in the root folder in the BTA application, and copy into the *.env* file all environment variables from the *.env-sample* file.

The sample of after copying *.env-sample* to *.env* file is as follows:

```
filename: .env

BC_NODE_LABEL=Peer0
BC_NODE_URL=http://192.0.2.1:0000
BC_NODE_AUTHORIZATION=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxx

BC_SUPER_ADMIN_REGISTRATION_TOKEN=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxx
```

```
BC_CONNECTOR_ADMIN_ID=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

#Super admin login credentials
SUPER-ADMIN-EMAIL=you-as-superadmin@gmail.com

# Default Oracle Authentication
OC_CONNECTION_HOST=http://192.0.2.1:0000
OC_AUTHORIZATION_TOKEN=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

# Register super admin on Blockchain (this is a Docker port)
REGISTER_SUPER_ADMIN_BC=http://localhost:3000/api/v1/user/register-super-
admin-bc
```

Before running the *docker-compose* command, you need to have your secret keys and blockchain information handy.

Secret keys for JWT

JSON Web Token (JWT) is an open standard (RFC 7519) that defines a compact and self-contained way for securely transmitting information between parties as a JSON object. This information can be verified and trusted because it is digitally signed. JWTs can be signed using a secret (with the HMAC algorithm) or a public/private key pair using RSA or ECDSA. JWT is basically used for the authorization and information exchange between two parties.

Basically JWT architecture has three parts i.e. Header, Payload and Signature. Header contains type of token like SHA256 or RS, payload contains your data and the signature part is created by taking the encoded header, the encoded payload, a secret, the algorithm specified in the header.

Using JWT in BTA, JWT library is already installed by the node package in the backend application and it is also imported in the application. Before you run the BTA, you have to set some of the secret keys in the env file that is used to validate the token across the applications.

There are three JWT secret key related variables i.e. JWT_SECRET, ENCRYPT_JWT_SECRET, RESET_JWT_SECRET which are used to sign or verify access tokens. You can enter your own JWT secret but make sure you add something unguessable.

For example:

```
JWT_SECRET=mySecretKey@13$
```

Encrypt_JWT_SECRET is used while encrypting JWT tokens and passing them to the client. You can put your own encrypted JWT secret. For example:

```
ENCRYPT_JWT_SECRET=myEncryptJWTSecretKey!0@$
```

Reset password secret is used to sign reset password tokens. You can put your own reset password secret. For example:

```
RESET_JWT_SECRET=myPasswordSecret#$@00
```

Recaptcha secret key

To get a recaptcha secret key, follow these steps:

1. Create an account and log in to Google.
2. Visit *https://www.google.com/recaptcha/about/* and on this page click admin console.
3. On the admin console page register a new site.
4. After completing the registration Google recaptcha provides the secret key and recaptcha site key. This recaptcha site key is coded into the BTA web application's frontend log in page, and the recaptcha secret key is stored as a variable in an *.env* file on the web server.

Oracle authentication

Oc Connection host is the Oracle connector deployed URL. The Oracle connector is used for making connections to Oracle Cloud. The URL will be the public IP of the Oracle connector where it is hosted. In the case of the same virtual machine used for both Oracle connector and backend application, localhost can be used instead of an IP. For example:

```
http://128.12.12.4:5022
```

OCAuthorizationToken is the key that enables the backend application to authenticate the access to the Oracle Connector APIs.

When the configuring the blockchain nodes you will need to know the following:

- *Bc node org name:* The blockchain organization name for the super admin. This name should be the same as the organization name created while creating the blockchain organization.
- *Bc node label:* The label for BC Node Org Name. The label can be any name that distinguishes the current node info from another.
- *Bc node URL:* The URL of the BC Connector of superadmin where it's deployed.
- *Bc node authorization:* The key that enables the backend application to authenticate the access to the super admin blockchain connector APIs.
- *Bc super admin registration token:* It is the key used to access backend API for registering superadmin from migration script. This token can be any unique key that will be later used to register a super admin user.
- *Bc connector admin Id:* BC Connector Admin id is the unique key that is used to call the super admin API for registering super admin users on blockchain. This key should be the same as that of bc connector ADMIN_ID.

Docker Compose

Your BTA runs using Docker, so you need to install docker compose.

Docker compose is a configuration tool used for managing and running multiple containers at once. Docker compose uses a YAML file for configurations of all containers which downloads the images of nest_api_local, mongoExpressNest, mongodbNest, and mailcatcherNest and finally installs.

Now switch to the root folder of the application in the terminal where you can find *docker-compose.yml*, which contains all the configuration for a Docker container of the BTA application. All variables in the *docker-compose.yml* file are imported from the *.env* file.

In the BTA application some Docker compose commands are used:

To start Docker container for development (local):

```
$ docker-compose up -d local
```

To start Docker container for production:

```
$ docker-compose up -d prod
```

To see the logs of the application for development (local):

```
$ docker-compose logs -f local
```

To see the logs of the application for production:

```
$ docker-compose logs -f prod
```

```
ENVIRONMENT=local
APP_NAME=App
APP_PORT=3340
APP_DEBUG_PORT=9230
MONGO_EXPRESS_UI_PORT=3341
DB_PORT=27034

MONGO_URI=mongodb://mongodbNest:27017
MONGO_HOST=mongodbNest
MONGO_USERNAME=nestapi
MONGO_PASSWORD=PUT-YOUR-MONGO-PASSWORD
DATABASE_NAME=kilroy-base-api
MONGO_DEBUG=true

LOCAL_CONTAINER_NAME=nest_api_local
MONGODB_CONTAINER_NAME=mongodbNest
MONGO_EXPRESS_UI_CONTAINER_NAME=mongoExpressNest
DATABASE_VOLUME_MOUNT=./psvolumes/nest-db
```

```
JWT_SECRET=PUT-YOUR_JWT-SECRET-KEY
ENCRYPT_JWT_SECRET=PUT-YOUR-ENCRYPT-JWT-SECRET
JWT_EXPIRATION=1000m
RESET_PASSWORD_SECRET=PUT-YOUR-SECRET-KEY
RE_CAPTCHA_SECRET=xxxxxxxxxxxxxxxxxxxxxxxxxxxx
BLOCKCHAIN=DISABLED
REFRESH_TOKEN_EXPIRATION_MIN=10080

CLIENT_APP_URL=http://localhost:4200

# Default Super Admin Blockchain data
BC_NODE_ORG_NAME=PeerFusionerMainnetBtaKilroy
```

To stop all services:

```
$ docker-compose down
```

BTA Frontend

Similar to the BTA application backend, to set up the frontend first you need the code for the frontend of the BTA application in your system.

You can **`git clone`** the BTA frontend like this:

```
$ git clone https://github.com/Kilroy-Blockchain-corporation/bta-app
$ cd bta-web
```

After you have the frontend of the BTA on your system, you can then set it up according to the development environments discussed in the next sections.

Install Node.js and Node Package Manager (npm)

To run the application, Node.js and npm should be installed in the system.

> **NOTE**
>
> Npm is automatically installed when Node.js is installed. In the BTA application, you need at least node version 18.07.0 and npm version 8.15.0.

Installation Dependencies

After installing the node and npm in the system, you have to install the dependency from the *package.json* file. Without installing dependencies, the frontend project will not be able to run.

Npm is used to install dependencies for the project. Here are the steps for installing dependencies:

1. Go to the root directory of the project and open the terminal.
2. Enter the following command in the terminal:

```
$ npm install
```

After the preceding command runs, npm installs all dependencies automatically from the package.json file.

Install and Configure Angular

Angular is used as a frontend tool in the BTA project. After installing the node and npm in the system, now you can install Angular using npm.

Installing and running Angular

To install Angular, run the following command:

```
$ sudo npm install -g @angular/cli
```

Run the application:

```
# Go to the BTA application frontend directory
$ cd bta-web
```

```
# Runs the application
$ ng serve
```

Environment file

The frontend of the BTA application also has the environment variables that are written in the environment file. This file, called *environment.ts*, exists in the frontend's root directory and contains key/value pairs setting values for the project's required environment variables.

Here are some important variables for the frontend of the BTA application:

- Filename: environment.ts

```
export const environment = {
                    project: 'bta',
                    apiURL: 'http://localhost:3340/api/v1',
                    hostURL: 'http://localhost:4200/#/',
                    disableCaptcha: true,
                    recaptchaSiteKey: 'xxxxxxxxxxxxxxxxxxxxxx'
                };
```

- *Project:* Name of the project.
- *apiURL:* The URL of the backend where the backend is hosted, which is used to communicate with the backend. You can get the URL after running the backend application.
- *hostURL:* The URL of the frontend, where the frontend is hosted.
- *Recaptcha site key:* explained earlier in this chapter.

Test Your Environment

To make sure everything is working properly in your development environment, you can run a simple test that checks whether everything is running and talking to each other.

Blockchain Environment Test

Run the following script from your terminal to make sure that your blockchain network is properly set up:

```
$ cd to-root-dir
./testBlockchainNetwork.sh
```

This gives the output like following:

```
Orderer1.org.bta.kilroy
Orderer2.org.bta.kilroy
Followings are the peers created in the network:
peer0.o4-mlops.bta.kilroy
peer0.o1-super-admin.bta.kilroy
peer0.o3-sh.bta.kilroy
peer0.o5-ai-engineer.bta.kilroy
peer0.o2-admin.bta.kilroy
```

AI Environment Test

When you finish installing Anaconda, open it. Click on Jupyter Notebook, which opens a notebook where you can write a script. It should look like Figure 5-2.

Figure 5-2. Fresh Jupyter Notebook opened, showing hello world sample script.

BTA Environment Test

You will want to test the frontend and backend of the BTA environment.

Backend test

To test the backend BTA app, run the command shown in Example 5-1 from your backend project root dir that gives you the log with error status.

Example 5-1. Output of Docker-compose logs -f local command

```
$ docker-compose logs -f local
```

```
[6:47:49 PM] Starting compilation in watch mode...

[6:47:51 PM] Found 0 errors. Watching for file changes.

[MyApp] Info  9/20/2022, 6:47:53 PM [NestFactory] Starting Nest application... -
{} +0ms
[MyApp] Info  9/20/2022, 6:47:53 PM [InstanceLoader] AppModule dependencies
initialized - {} +48ms
[MyApp] Info  9/20/2022, 6:47:53 PM [InstanceLoader] MongooseModule dependencies
initialized - {} +0ms
[MyApp] Info  9/20/2022, 6:47:53 PM [InstanceLoader] MulterModule dependencies
initialized - {} +0ms
[MyApp] Info  9/20/2022, 6:47:53 PM [InstanceLoader] AppUserModule dependencies
initialized - {} +0ms
[MyApp] Info  9/20/2022, 6:47:53 PM [InstanceLoader] PassportModule dependencies
initialized - {} +1ms
[MyApp] Info  9/20/2022, 6:47:53 PM [InstanceLoader] MailerModule dependencies
initialized - {} +0ms
[MyApp] Info  9/20/2022, 6:47:53 PM [InstanceLoader] BcConnectionModule
dependencies initialized - {} +0ms
[MyApp] Info  9/20/2022, 6:47:53 PM [InstanceLoader] UserRolesModule dependencies
initialized - {} +0ms
[MyApp] Info  9/20/2022, 6:47:53 PM [InstanceLoader] UtilsModule dependencies
initialized - {} +0ms
[MyApp] Info  9/20/2022, 6:47:53 PM [InstanceLoader] BlockchainModule
dependencies initialized - {} +0ms
[MyApp] Info  9/20/2022, 6:47:53 PM [InstanceLoader] OracleModule dependencies
initialized - {} +0ms
[MyApp] Info  9/20/2022, 6:47:53 PM [InstanceLoader] ManageProjectModule
dependencies initialized - {} +0ms
[MyApp] Info  9/20/2022, 6:47:53 PM [InstanceLoader] JwtModule dependencies
initialized - {} +7ms
[MyApp] Info  9/20/2022, 6:47:53 PM [InstanceLoader] HttpModule dependencies
initialized - {} +0ms
[MyApp] Info  9/20/2022, 6:47:53 PM [InstanceLoader] ConfigHostModule
dependencies initialized - {} +0ms
```

Frontend test

After running the following command from the frontend project directory, you will see the output shown in Example 5-2.

Example 5-2. Output of ng serve command

```
$ ng serve

✔ Browser application bundle generation complete.
```

```
✔ Copying assets complete.
✔ Index html generation complete.

Initial Chunk Files           | Names                                  |  Raw
Size | Estimated Transfer Size
main.28dc736f15816bcb.js      | main                                   |   1.11
MB |               232.07 kB
styles.3e94c5ec19923b44.css   | styles                                 | 983.23
kB |                52.08 kB
polyfills.01cb3387bfcaa47c.js | polyfills                              |  33.02
kB |                10.64 kB
runtime.85e1fc8bbf5dfc24.js   | runtime                                |   2.94
kB |                 1.42 kB
                              | Initial Total                          |   2.11
MB |               296.21 kB

Lazy Chunk Files              | Names                                  |  Raw
Size | Estimated Transfer Size
260.4e23d8bff4f815df.js       | manage-project-manage-project-module   | 201.69
kB |                22.72 kB
782.566c7b84b60e93bb.js       | user-user-module                       | 111.74
kB |                16.58 kB
720.b6f5c990d2bc8e26.js       | pages-after-login-after-login-module   | 111.38
kB |                18.50 kB
941.e5ecbd36aa6224da.js       | auth-auth-module                       |  56.68
kB |                10.00 kB
605.823c0e067b458821.js       | user-user-module                       |  24.06
kB |                 5.59 kB
301.0ca5f74b0412fa09.js       | pages-before-login-before-login-module |  19.60
kB |                 5.81 kB
26.58555b63d76eeed1.js        | pages-after-login-after-login-module   |  10.63
kB |                 2.97 kB
983.591bee1e21495c93.js       | blockchain-blockchain-module           | 812
bytes |                412 bytes
common.fb9af8770cbc4586.js    | common                                 | 645
bytes |                302 bytes

Build at: 2022-09-20T13:04:24.841Z - Hash: 7bb6ae07239d12d8 - Time: 22984ms
```

Summary

Once you have successfully prepared a development environment containing your model, blockchain, and BTA, it is time to proceed to Chapter 6 and begin building the application.

About the Authors

Karen Kilroy, Kilroy Blockchain's CEO, is a life-long technologist with heart. Living in Austin, Texas, she has invented several products including Casey, Kilroy Blockchain PaaS, Riley, and Carnak. Karen believes that artificial intelligence doesn't have to be scary and that technology can be proactively designed to be used for good. Karen's background includes working as an open source software engineer, which has prepared her for the new world of cognitive intelligence and tamper-evident ledgers. Karen is a Certified IBM Foundation Blockchain Developer and a Certified Magento Front-End Developer. Named IBM Champion in the blockchain and AI categories, Karen is an experienced public speaker and author. She is the author of the O'Reilly publications *Blockchain as a Service* (2019) and *AI and the Law* (2021). Karen is also a professional dragon boat coach. Karen Kilroy has extensive public speaking experience involving highly technical topics.

A Certified IBM Foundation Blockchain Developer, **Deepak Bhatta** is an award-winning, hands-on technology leader with more than 10 years in open source software development. His main focus as Kilroy Blockchain's CTO is on turning ideas into reality by managing the company's development efforts in blockchain and AI, including the award-winning RILEY. He is actively involved in architecturing and designing products like RILEY and Casey. He has a decade of experience in developing ecommerce and mobile apps. He was a presenter on blockchain at the RegTech MENA Conference in Dubai in April 2017 and the Conclave 2019 conference in Kathmandu, Nepal.

Lynn Riley is a technologist with a background at Motorola, Apple, and CORT Business Services. A graduate of MIT in chemical engineering, she brings her skills to Kilroy Blockchain as Chief Information Officer and also manages all of Kilroy Blockchain's partnerships.

Much of the visual recognition training for RILEY and the data analysis for the early version of CARNAK was managed by Lynn. She also helped design both products as well as architect the workflow and design of CASEY.