WELCOME TO THE GIVING CHAIN

2021

AGENDA

- Introduction and Housekeeping
- Recap Project
  - Current Projects and Checkpoints
- Working Group Updates
  - Technical Github
  - Business Timeline and Steps
- Getting Involved
  - Teams (Technical / Business)
  - Joining The Community Social Media

Antitrust Policy
Antitrust Policy Notice

- Antitrust Policy Antitrust Policy Notice: link: https://www.linuxfoundation.org/antitrust-policy/
- Linux Foundation meetings involve participation by industry competitors, and it is the intention of the Linux Foundation to conduct all of its activities in accordance with applicable antitrust and competition laws. It is therefore extremely important that attendees adhere to meeting agendas, and be aware of, and not participate in, any activities that are prohibited under applicable US state, federal or foreign antitrust and competition laws. Examples of types of actions that are prohibited at Linux Foundation meetings and in connection with Linux Foundation activities are described in the Linux Foundation Antitrust Policy available at http://www.linuxfoundation.org/antitrust-policy. If you have questions about these matters, please contact your company counsel, or if you are a member of the Linux Foundation, feel free to contact Andrew Updegrove of the firm of Gesmer Updegrove LLP, which provides legal counsel.

Team

Introduction

Project

Hyperledger, Ledger Academy
and
BC Princeton Presents

The Giving Chain
&
The D2R Application

Bobbi Muscara
Project Manager – Hyperledger Giving Chain
Director Education Ledger Academy
Au*thor of Linux Foundation Blockchain Courses for LFS171*
Hyperledger TSC Member
Chair OF THE Learning Materials Working Group

Hardik Gupta
Project Manager – Hyperledger Giving Chain
Member OF THE Learning Materials Working Group
The Giving Chain
Mentee / Project Management / Technical Lead

Madhu Bhatia
Project Manager – Hyperledger Giving Chain
DLT Talent program of Frankfurt School of Finance and Management Blockchain Center
2021 LiFT Scholarship Women in open-source
Research Scholar /Trainer

Project Recap

Giving Chain Princeton

Giving Chain India / Uttarakhand

Giving Chain India Women

Checkpoint 1: Impact Opportunity

Princeton

- **Select your challenge.**

- Develop an agile & transparent network for connecting surplus foods & supplies to communities in need.
- To create a decentralized system that collect excess food supplies from local resources and distributes these resources to local food insecure individuals.

- **Define the scope of problem**

- For our summer project, we are limiting the POC to a small number of local farmers with excess food supplies (two pick up dates TBD) to be picked up by transportation volunteers who will transport food supplies to agreed upon drop off point .

- **Why is blockchain needed to solve this problem**

- A supply chain model has been determined to be the best way to initiate, track and deliver excess food supplies. The above model has many unknown individuals who do not need to form a relationship with the parties at the other end of the supply chain.

Checkpoint 1: Impact Opportunity

India - Uttarakhand

Select your challenge.

Develop an agile & transparent network for connecting supplies and aid to flood damaged communities in need. To create a decentralized system that collect needed supplies from local resources and crowdfunding efforts and distributes these resources to flood victims.

Define the scope of problem

For our summer project, we are limiting the POC to Uttarakhand flood victims . Supplies collected and purchased to be picked up by transportation volunteers who will transport food supplies to agreed upon drop off point .

Why is blockchain needed to solve this problem

A supply chain model has been determined to be the best way to initiate, track and deliver supplies.

Checkpoint 1: Impact Opportunity

India - Women

Select your challenge.

Develop an agile & transparent network for connecting supplies for the women in crisis in India. Create a dignified distribution system for hygiene products to women by crowdfunding, purchasing and distributing these resources to women in rural areas in India.

Define the scope of problem

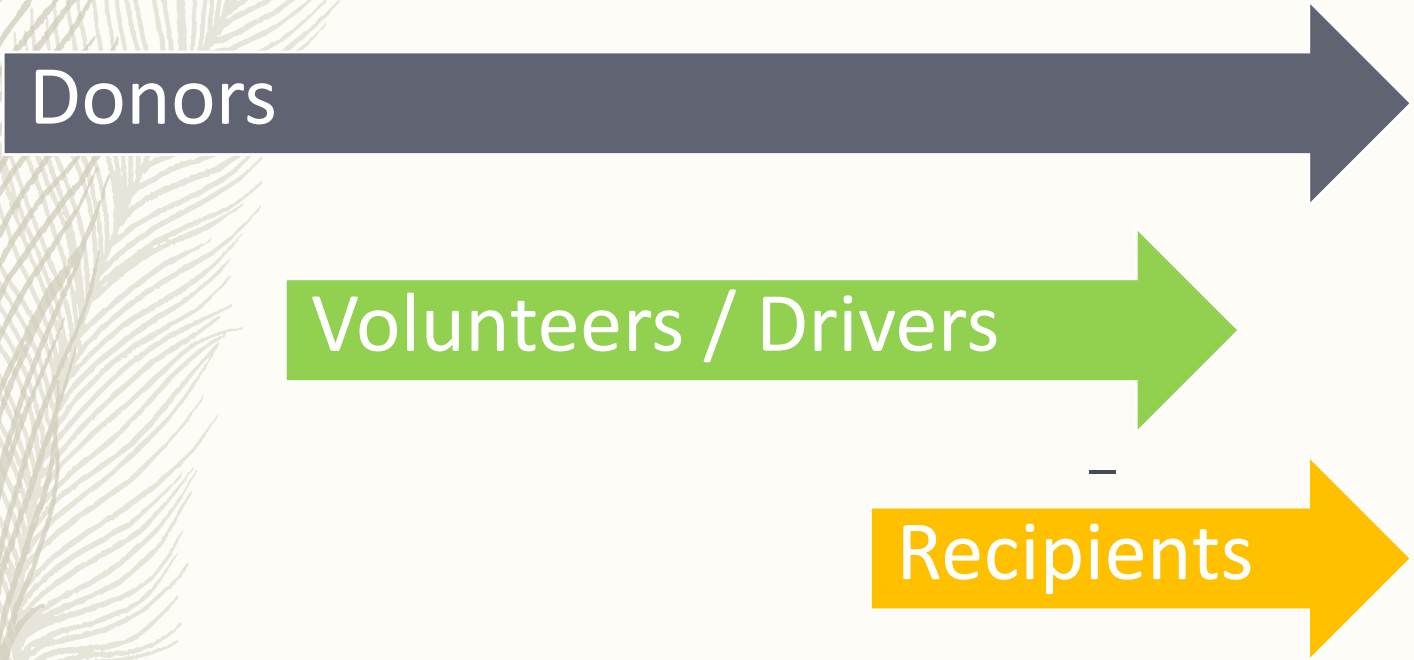
For our summer project, we are limiting the POC to Rural area in India. Supplies purchased will be e picked up by transportation volunteers who will transport hygiene supplies to drop off point in rural areas.

Why is blockchain needed to solve this problem

A supply chain model has been determined to be the best way to initiate, track and deliver supplies.

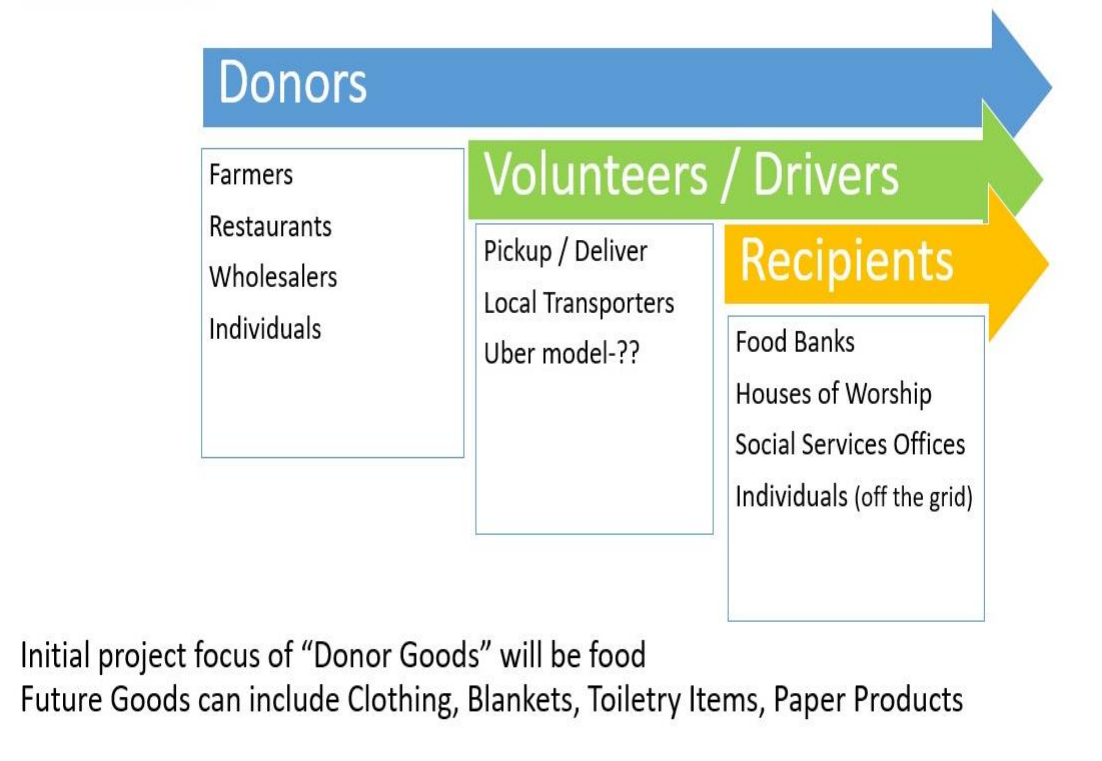
Checkpoint 2: Model Validation *(questions)*

- Who are your **clients/users?** *Sketch your user.*
- What are their **pain points**? How does your solution solve them?
  - Can it **be scaled**?



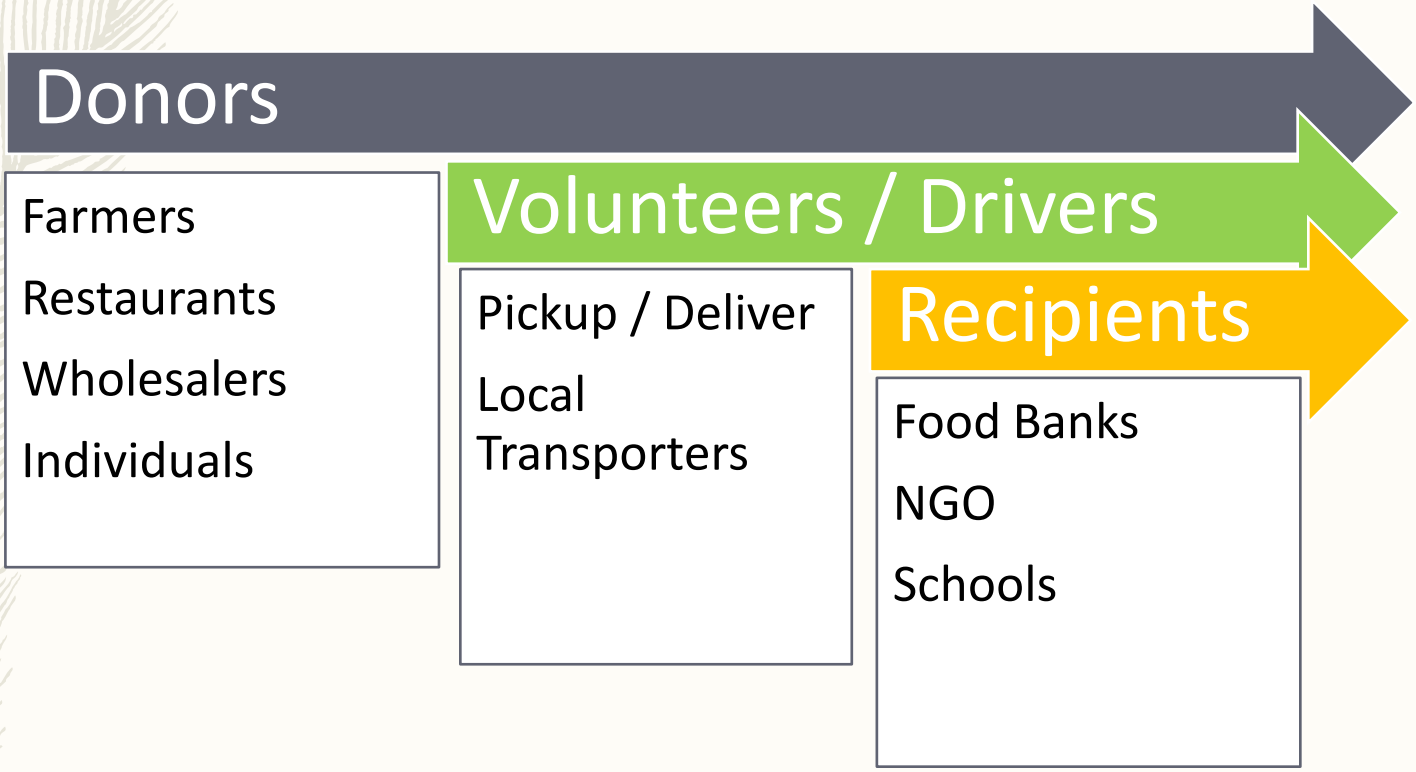
Checkpoints 2 Model Validation

Princeton



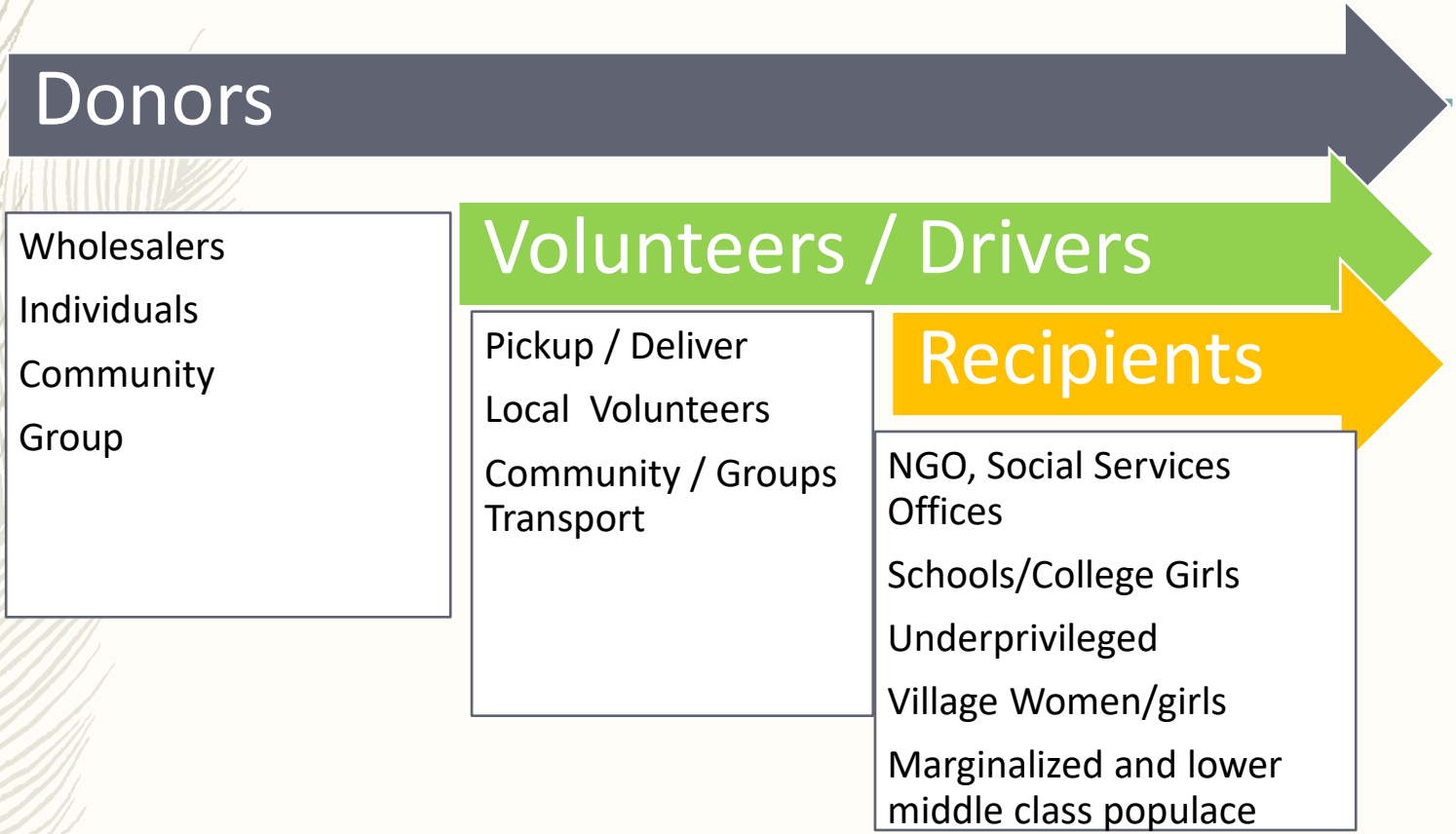
Checkpoints 2 Model Validation

India - Uttarakhand



Checkpoints 2 Model Validation

India - Women



Checkpoint 3 Business and Technical Modeling

Princeton: Business Model

Donors

Farmers

Individuals

Community / Church Groups

Volunteers / Drivers

Pickup / Deliver:

Local volunteers
Community / Groups
Transport

Recipients

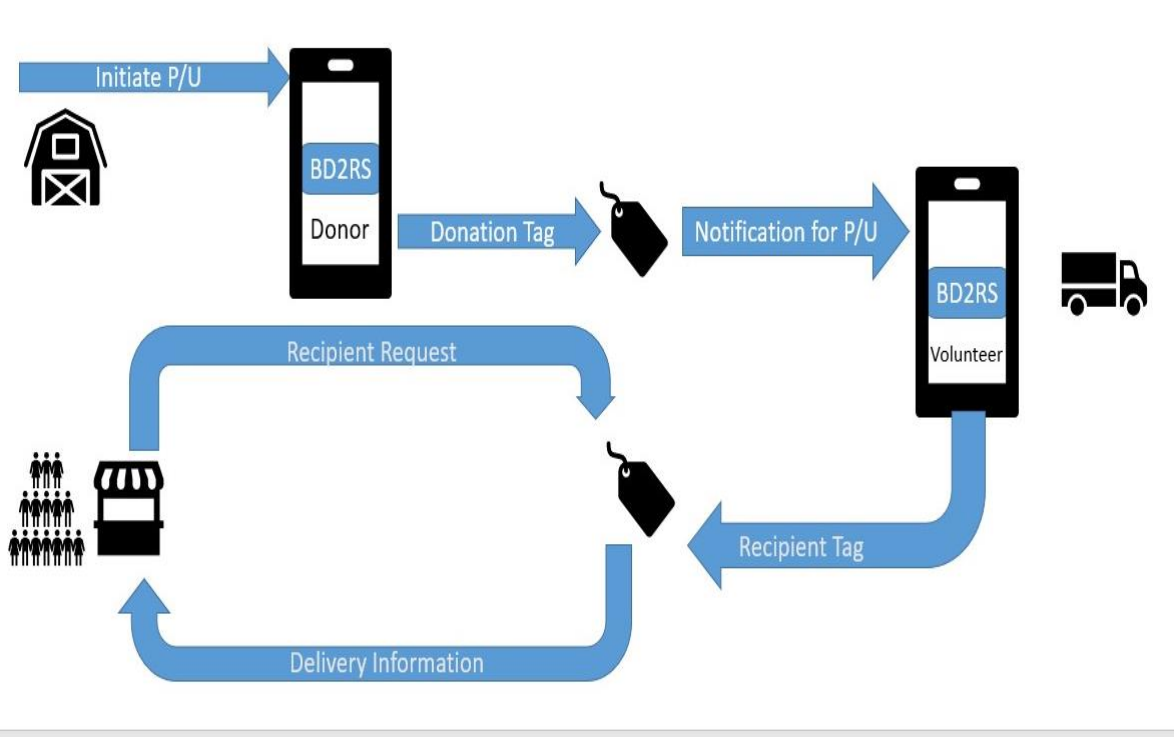
Food Banks

Social Services Offices

Individuals (off the grid)

Checkpoint 3 Business and Technical Modeling

Princeton: Technical Business Model

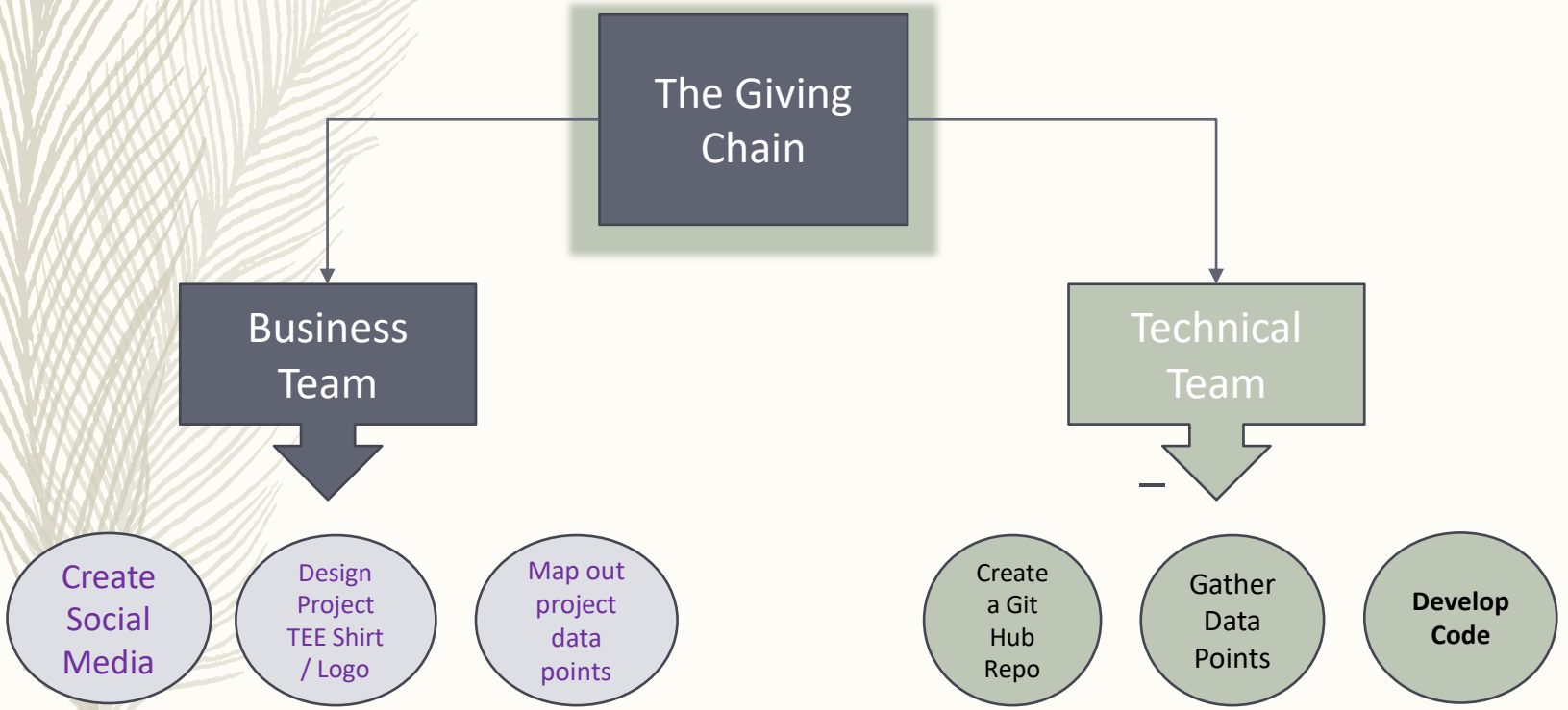


TRANSACTIONS IN OUR WORKFLOW:

help to identify the implicit transactions and features to be supported by this blockchain application solution via business network APIs.

- Identify an initial **list of Donors** who will donate. List will expand with additional Donors.
- Identify **the location** of the Donors. Locations will expand with additional Donors.
- Identify an **initial list of Transporters** - an initial list which will expand as additional Donors volunteer their services.
- Identify the **location o**f Transporters. Locations will expand with additional Donors.
- **Identify attributes** of Donors, Donors, Transporters that will help with identification and communication to ensure supply pipeline efficiency.
- Donor provides notification that they have a food supply ready for pickup by Transporters for delivery to Consumers.
- The Donor indicates the available quantity of food they are supplying.
- Transporter monitors the blockchain to determine when food is available for pickup.
- Transporter matches the Donor supply with specific Consumers (whose daily needs have not been filled) to whom they can deliver this food supply.
- Transporter volunteers to pickup the available supplies and deliver to specific Consumers (criteria may be location proximity).
- Transporter picks up the available food supply from the Donor.
- Transporter delivers the available food supply to the Consumers.
- Transporter records that they have provided this service.
- The Donor's quantity of food supply available for transport is decremented.
- The Consumer is checked-off as having had their daily needs filled.
- If additional quantities of food is still available from this Donor, the Donor notification remains active else it is closed.
- Application metrics are updated (e.g., quantify the amount of food delivered, the cost of delivered food, etc.) for analytics.
- At the end of day, all pending transactions are closed to reset for a new day of deliveries.

- Define Project and team members.



BCPrinceton's

SUMMER BLOCKCHAIN PROJECT

The Giving Chain

Get involved for Social Good This Summer!!	**Blockchain Princeton** and **Ledger Academy** are encouraging donations for local residence in need. Please drop off a donation and give back to your community!
Pickup Date	August 25, 2019
Donation Suggestions: Toothbrushes (Children's) Razors / Shaving Cream Creamy Peanut Butter Vegetable Soup Canned Chicken Canned Fruit Dog and Cat Food Feminine Hygiene Products Depends	THANK YOU SPONSORS www.bcprinceton.com AYLUS Hyperledger The Flemington Area FOOD PANTRY GYMNASTICS UNLIMITED PARAMOUNT Software Solutions

Collection Location

Sawtooth

Originally contributed by Intel, Sawtooth is a blockchain suite designed for versatility and scalability. Distributed Ledger Technology has potential in many fields with use cases from IoT to Financials. This architecture recognizes the diversity of requirements across that spectrum. Sawtooth supports both permissioned and permissionless deployments. It includes a novel consensus algorithm, Proof of Elapsed Time (PoET). PoET targets large distributed validator populations with minimal resource consumption. Transaction business logic is decoupled from the consensus layer into Transaction Families that allow for restricted or unfettered semantics.

Key Characteristics

Pluggable consensus algorithms (Change consensus on the fly by transaction)

Includes Proof of Elapsed Time (PoET) consensus

Write smart contracts in almost any language

Ethereum contract support via Hyperledger Burrow integration

Supply Chain example out of the box

Parallel transaction execution for added throughput

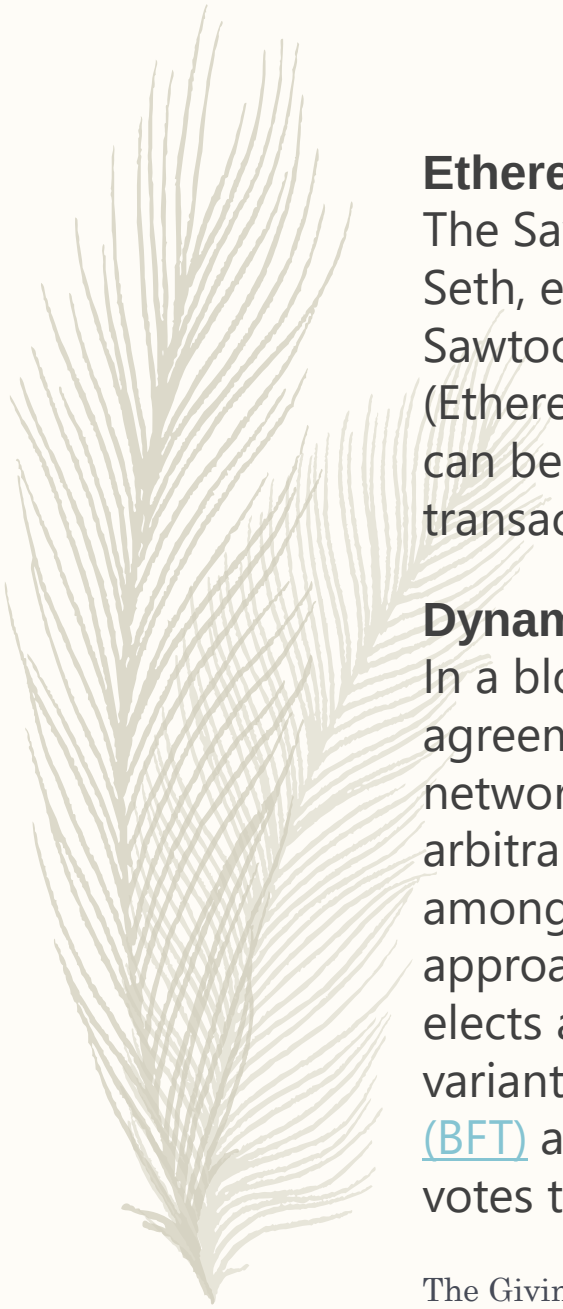
Separation Between the Application Level and the Core System

Sawtooth makes it easy to develop and deploy an application by providing a clear separation between the application level and the core system level. Sawtooth provides smart contract abstraction that allows application developers to write contract logic in a language of their choice.

core system level

application level

- native business logic or a smart contract virtual machine
- design decisions to be made in the transaction-processing layer, which allows multiple types of applications to exist in the same instance of the blockchain network.



Ethereum Contract Compatibility with Seth

The Sawtooth-Ethereum integration project, Seth, extends the interoperability of the Sawtooth platform to Ethereum. EVM (Ethereum Virtual Machine) smart contracts can be deployed to Sawtooth using the Seth transaction family.

Dynamic Consensus

In a blockchain, consensus is the process of building agreement among a group of participants in a network. Algorithms for achieving consensus with arbitrary faults generally require some form of voting among a known set of participants. General approaches include Nakamoto-style consensus, which elects a leader through some form of lottery, and variants of the traditional Byzantine Fault Tolerance (BFT) algorithms, which use multiple rounds of explicit votes to achieve consensus.

Private Networks with the Sawtooth Permissioning Features

Sawtooth is built to solve the challenges of permissioned (private) networks. Clusters of Sawtooth nodes can be easily deployed with separate permissioning. There is no centralized service that could potentially leak transaction patterns or other confidential information. The blockchain stores the settings that specify the permissions, such as roles and identities, so that all participants in the network can access this information.

Parallel Transaction Execution

Most blockchains require serial transaction execution in order to guarantee consistent ordering at each node on the network. Sawtooth includes an advanced parallel scheduler that splits transactions into parallel flows. Based on the locations in state which are accessed by a transaction, Sawtooth isolates the execution of transactions from one another while maintaining contextual changes.

When possible, transactions are executed in parallel, while preventing double-spending even with multiple modifications to the same state. Parallel scheduling provides a substantial potential increase in performance over serial execution.

Event System

Hyperledger Sawtooth supports creating and broadcasting events. This allows applications to:

•Subscribe to events that occur related to the blockchain, such as a new block being committed or switching to a new fork.

•Subscribe to application specific events defined by a transaction family.

•Relay information about the execution of a transaction back to clients without storing that data in state.

The Sawtooth consensus API supports a wide variety of consensus algorithms on a network. Sawtooth currently includes consensus engines for these algorithms:

•Sawtooth PBFT (Practical Byzantine Fault Tolerance) is a voting-based consensus algorithm that provides Byzantine fault tolerance with finality. Sawtooth PBFT extends the original PBFT algorithm with features such as dynamic network membership, regular view changes, and a block catch-up procedure. A Sawtooth network with PBFT consensus requires four or more nodes.

•PoET (Proof of Elapsed Time) is a Nakamoto-style consensus algorithm that is designed to be a production-grade protocol capable of supporting large network populations. PoET relies on secure instruction execution to achieve the scaling benefits of a Nakamoto-style consensus algorithm without the power consumption drawbacks of the Proof of Work algorithm. A Sawtooth network with PoET consensus requires at least three nodes.

Sawtooth includes two versions of PoET consensus:

- •PoET-SGX relies on a Trusted Execution Environment (TEE), such as Intel® Software Guard Extensions (SGX), to implement a leader-election lottery system. PoET-SGX is sometimes called “PoET/BFT” because it is Byzantine fault tolerant.
- •PoET simulator provides PoET-style consensus on any type of hardware, including a virtualized cloud environment. PoET simulator is also called “PoET/CFT” because it is crash fault tolerant, not Byzantine fault tolerant.

•Sawtooth Raft is a leader-based consensus algorithm that provides crash fault tolerance for a small network with restricted membership.

•Devmode (short for “developer mode”) is a simplified random-leader algorithm that is useful for developing and testing a transaction processor. Devmode is not recommended for multi-node networks and should not be used for production

•Sawtooth Supply Chain: Demonstrates how to trace the provenance and other contextual information of any asset. Supply Chain provides an example application with a transaction processor, custom REST API, and web app. This example application also demonstrates a decentralized solution for in-browser transaction signing and illustrates how to synchronize the blockchain state to a local database for complex queries. For more information, see the sawtooth-supply-chain repository on GitHub.

Transaction Family Overview

Sawtooth separates the application level from the core system level with transaction families, which allows application developers to write in the languages of their choice. Each application defines the custom transaction families for its unique requirements.

A transaction family includes these components:

- A transaction processor to define the business logic for your application
- A data model to record and store data
- A client to handle the client logic for your application

See Transaction Family Specifications for a list of example transaction families. Sawtooth provides these examples to serve as models for low-level functions (such as maintaining chain-wide settings and storing on-chain permissions) and for specific applications such as performance analysis and storing block information.

•Sawtooth Supply Chain: Demonstrates how to trace the provenance and other contextual information of any asset. Supply Chain provides an example application with a transaction processor, custom REST API, and web app. This example application also demonstrates a decentralized solution for in-browser transaction signing, and illustrates how to synchronize the blockchain state to a local database for complex queries. For more information, see the sawtooth-supply-chain repository on GitHub.

Try Hyperledger Sawtooth

The Sawtooth documentation explains how to set up a local validator for demonstrating Sawtooth functionality and testing an application.

Once running, you will be able to submit new transactions and fetch the resulting state and block data from the blockchain using HTTP and the Sawtooth REST API.

These methods apply to the included example transaction families, as well as to any transaction families you might write yourself.

Get the Sawtooth Software

The Sawtooth software is distributed as source code with an Apache license. You can get the code to start building your own distributed ledger.

- sawtooth-core: Contains fundamental classes used throughout the Sawtooth project, as well as the following items:
  - The implementation of the validator process which runs on each node
  - SDKs for writing transaction processing or validation logic in a variety of languages
  - Dockerfiles to support development or launching a network of validators
  - Source files for this documentation
- Sawtooth PBFT: Use PBFT consensus with Sawtooth
- Sawtooth Sabre: Run on-chain smart contracts executed in a WebAssembly virtual machine
- Sawtooth Seth: Deploy Ethereum Virtual Machine (EVM) smart contracts to Sawtooth
- Sawtooth Marketplace: Exchange customized "assets" with other users on the blockchain
- Sawtooth Supply Chain: Trace the provenance and other contextual information of any asset

Join the Sawtooth Community

Sawtooth can be run from a pre-built Docker container, from a Kubernetes cluster inside a virtual machine on your computer, or on a native Ubuntu installation.

Transaction Family Overview

Sawtooth separates the application level from the core system level with transaction families, which allows application developers to write in the languages of their choice. Each application defines the custom transaction families for its unique requirements.

A transaction family includes these **components:**

- A transaction processor to define the business logic for your application
- A data model to record and store data
- A client to handle the client logic for your application

See Transaction Family Specifications for a list of example transaction families. Sawtooth provides these examples to serve as models for low-level functions (such as maintaining chain-wide settings and storing on-chain permissions) and for specific applications such as performance analysis and storing block information.

Setting Up a Sawtooth Node for Testing

Before you can start developing for the *Hyperledger Sawtooth* platform, you'll need to set up a local Sawtooth node to test your application against.

Once the node is running, you will be able to submit new transactions and fetch the resulting state and block data from the blockchain using HTTP and the Sawtooth REST API.

The methods explained in this section apply to the example transaction processors, *IntegerKey* and *XO*, as well as any transaction processors you might write yourself.

Using Docker for a Single Sawtooth Node

This procedure explains how to set up Hyperledger Sawtooth for application development using a multi-container Docker environment.

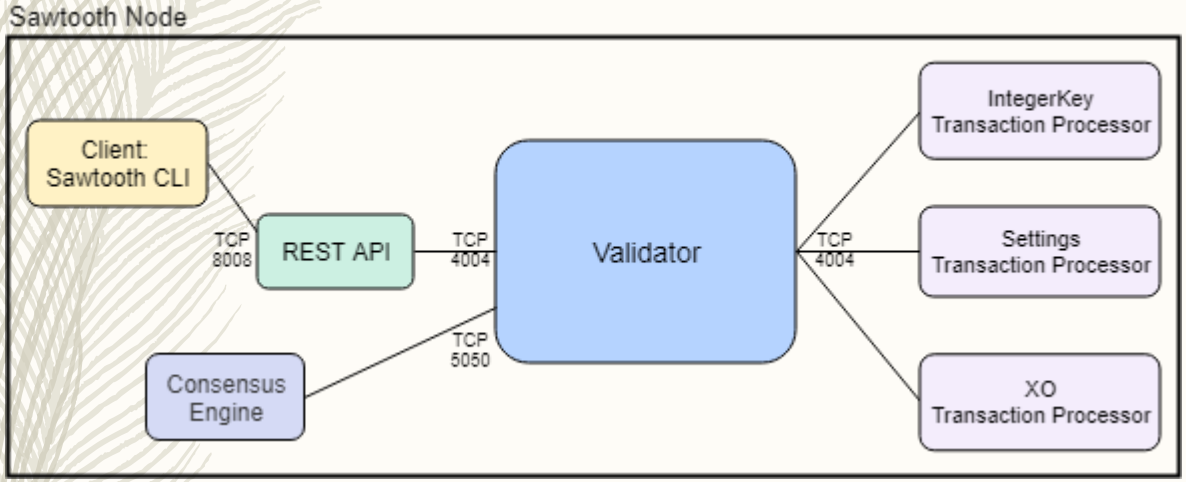
It shows you how to start Sawtooth and connect to the necessary Docker containers, then walks you through the following tasks:

- Checking the status of Sawtooth components
- Using Sawtooth commands to submit transactions, display block data, and view global state
- Examining Sawtooth logs
- Stopping Sawtooth and resetting the Docker environment

After completing this tutorial, you will have the application development environment that is required for the other tutorials in this guide. The next tutorial introduces the XO transaction family by using the `xo` client commands to play a game of tic-tac-toe. The final set of tutorials describe how to use an SDK to create a transaction family that implements your application's business logic.

About the Docker Test Node Environment

This Docker environment is a single Sawtooth node that is running a validator, a REST API, the Devmode consensus engine, and three transaction processors. The environment uses Devmode consensus and parallel transaction processing.



This environment introduces basic Sawtooth functionality with the IntegerKey and Settings transaction processors for the business logic and Sawtooth commands as a client.

It also includes the XO transaction processor, which is used in later tutorials. The IntegerKey and XO families are simple examples of a transaction family, but Settings is a reference implementation.

In a production environment, you should always run a transaction processor that supports the Settings transaction family.

Note

The Docker environment includes a Docker Compose file that handles environment setup steps such as generating keys and creating a genesis block. To learn how the typical startup process works, see Using Ubuntu for a Single Sawtooth Node.

Step 1: Download the Sawtooth Docker Compose File

Download the Docker Compose file for the Sawtooth environment, sawtooth-default.yaml.

This example Compose file defines the process for constructing a simple Sawtooth environment with following containers:

- A single validator using Devmode consensus
- A REST API connected to the validator
- The Settings transaction processor (sawtooth-settings)
- The IntegerKey transaction processor (intkey-tp-python)
- The XO transaction processor (xo-tp-python)

- A client (shell) container for running Sawtooth commands

The Compose file also specifies the container images to download from Docker Hub and the network settings needed for all the containers to communicate correctly.

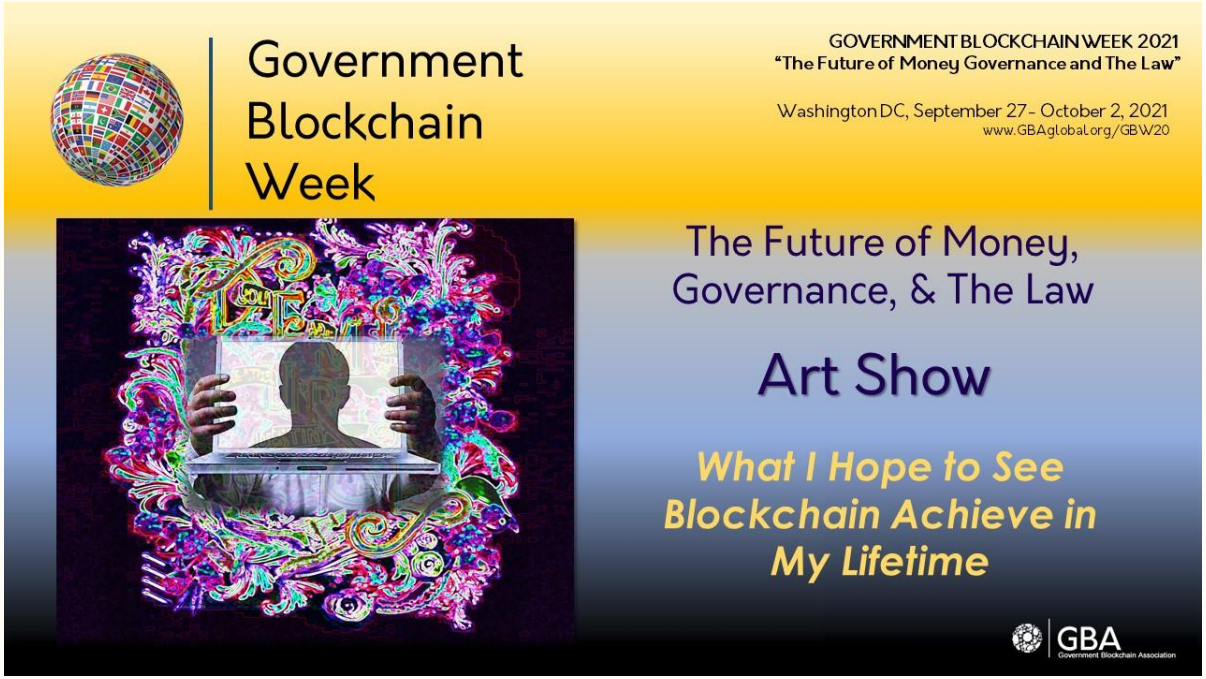
After completing the tutorials in this guide, you can use this Compose file as the basis for your own multi-container Sawtooth development environment or application.

Step 2: Configure Proxy Settings (Optional)
To configure Docker to work with an HTTP or HTTPS proxy server, follow the instructions for proxy configuration in the documentation for your operating system:

•Windows - See "Get Started with Docker for Windows".
•macOS - See "Get Started with Docker for Mac".
•Linux - See "Control and configure Docker with Systemd".

Government Blockchain Association

Government Blockchain Association



Artist Needed

The GBA will be hosting a digital (online) Art Show during Government Blockchain Week, Sept 27-Oct 2, 2021

'What I Hope to See Blockchain Achieve in My Lifetime'

The GBA is creating **NFT**s out of selected pieces that will be for sale at this event. Artists. All artwork in this show will be printed into a brochure with the artist's name, contact information, and a paragraph about their piece. Additionally, the entire show will be posted digitally for our members around the globe to see.

Submissions will be judged on 5 categories:
1) Creativity/Originality. How insightful and innovative is this idea?
2) Practicality. Could this idea work? Is it even remotely possible?
3) Helpful/ Humanity. Does this idea help anyone? Who? How?
4) Involvement. Is the artist involved in the deployment of this project? How? To what extent?
5) Visual appeal. How pleasing is the image to behold?

The call for entries is open now and we will be taking submissions until August 15, 2021.
This contest is open to anyone.

2021 GBA Art Show Submission

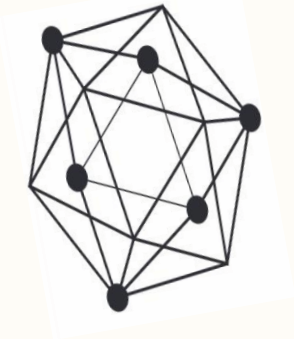
The Giving Chain

lay, August 2, 2021

BCEmploy

Session Three Princeton Meetup
8/12/2021

Session Four JOB FAIR 8/24/2021



bcemploy.com

Ledger Academy

INDIAN BLOCKCHAIN INSTITUTE