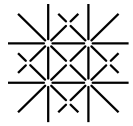


University of Basel

Center for Innovative Finance


# Smart Contracts and Decentralized Finance

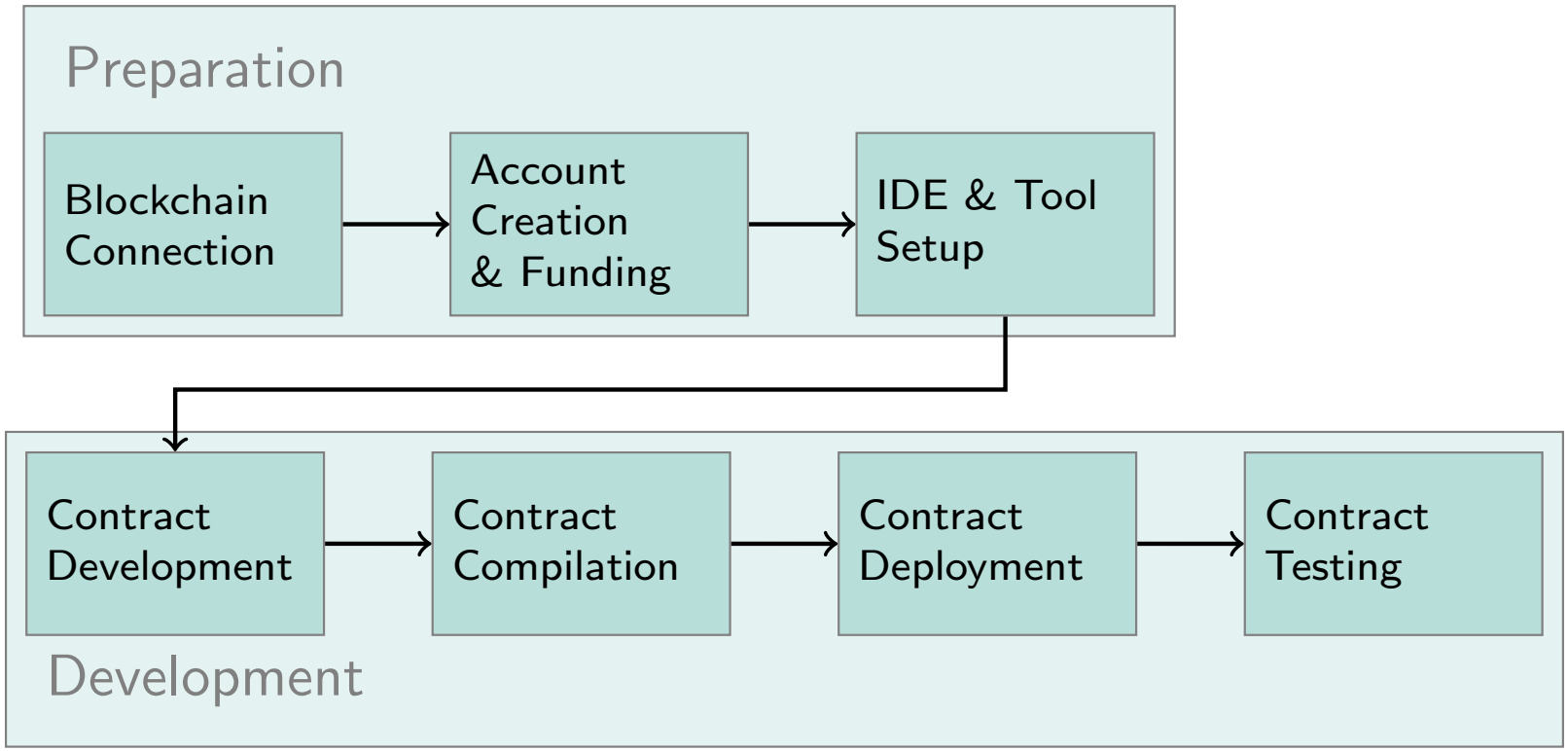
## Development Workflow

Prof. Dr. Fabian Schär
University of Basel

Release Ver.: (Local Release)
Version Hash: 008286996a77227dc66869f1889a7d765a5fb609
Version Date: 2022-09-18 10:42:50 +0200



# The Workflow



# Blockchain Connection

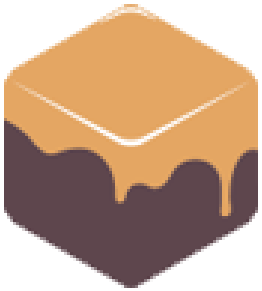
## What are you connecting to?

| Network | Security | Speed | Access | Value | Cost | Ideal for |
|---|---|---|---|---|---|---|
| Ethereum mainnet | High | Slow | Public | Yes | ETH | Production |
| Public testnet | Low | Slow | Public | No | Free (Faucet) | Open Alpha |
| Private testnet / JavaScript VM | Very low | Instant | Private | No | Free | Early development |
| Mainnet fork | Very low | Instant | Private | No | Free | Integrated development |

## How are you connecting?

- Local node
  - Best security
  - Local validation
  - Archive node: high hardware requirements
- Third party service (e.g., Infura)
  - Most convenient option
  - But: heavily centralized
  - Archive node: costly

# Create Your Private Test Chain with Ganache



Ganache quickly creates a private (Ethereum) Blockchain on your computer and prefunds 10 accounts with 100 custom Ether each. It is a great solution to quickly set up a private development chain, or to fork the Ethereum mainnet for integrated smart contract development.

**Exercise 1:**

1. Install Ganache on your personal computer.
2. Open Ganache and change hostname to local (0.0.0.0 – All Interfaces).
3. Download Metamask.
4. Connect Metamask to your private chain (RPC URL: https://localhost:8545 and Chain ID: 1337)
5. Import the Ganache private key of account 1 to Metamask.
6. Make a transaction from account 1 to account 2.

# Account Creation and Funding

Depending on the blockchain you are working on, you get ETH in different ways:

- On **mainnet** you have to **buy Ether** (usually from an exchange).
- On **public testnets** you get Ether through an **Ether faucet**.
- On **private chains** you can simply **create new custom Ether**.

**Additional Information on Private Chains**

↗ Ethereum Tutorials and Tips by Hudson

# IDE and Tool Setup

We recommend one of the following two IDEs:

**Remix** is a browser-based IDE for Solidity.

**Atom** is a general-purpose IDE. Make sure to download syntax highlighting packages for Solidity.

For versioning purposes we recommend Git:

**GitHub** is a hosting platform for Git repositories. Open Source projects are free and there is an educational account.

# Contract Development

- **Define your goal**
  - What are you trying to achieve with your smart contract?
- **Define your variables**
  - What needs to be stored in the smart contract?
  - Type and size needed.
- **Define your functions**
  - What is the function supposed to do?
  - What are the input parameters?
  - Are there any access restrictions?

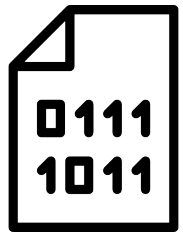
# Contract Compilation

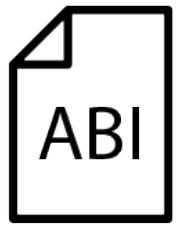
Solidity Code

Vyper

LLL(Lower-Level Lisp-like Language)

compiles to




**Bytecode**

- Deployed on blockchain
- Actual code that gets executed

**Application Binary Interface**

- List of contract functions and their arguments (JSON)
- En- & decoding of data (translation)

**The Need for an ABI**

Without the ABI, applications would not know the available functions and function arguments. The ABI can be seen as a translation or contract interface specification.

# Contract Deployment

**Contract Deployment**

Creating your contract on the blockchain, i.e., sending a deployment transaction.

**You can either . . .**

- . . . (compile and) deploy a contract directly from most development frameworks (incl. Remix).
- . . . use any wallet to deploy bytecode.

# Contract Testing

**Testing your contract is very important.**

- Manual testing
  - Remix
  - Metamask
- Automated testing
  - Hardhat (automated JavaScript test cases)
  - Brownie (automated Python test cases)
  - Truffle (automated JavaScript test cases)