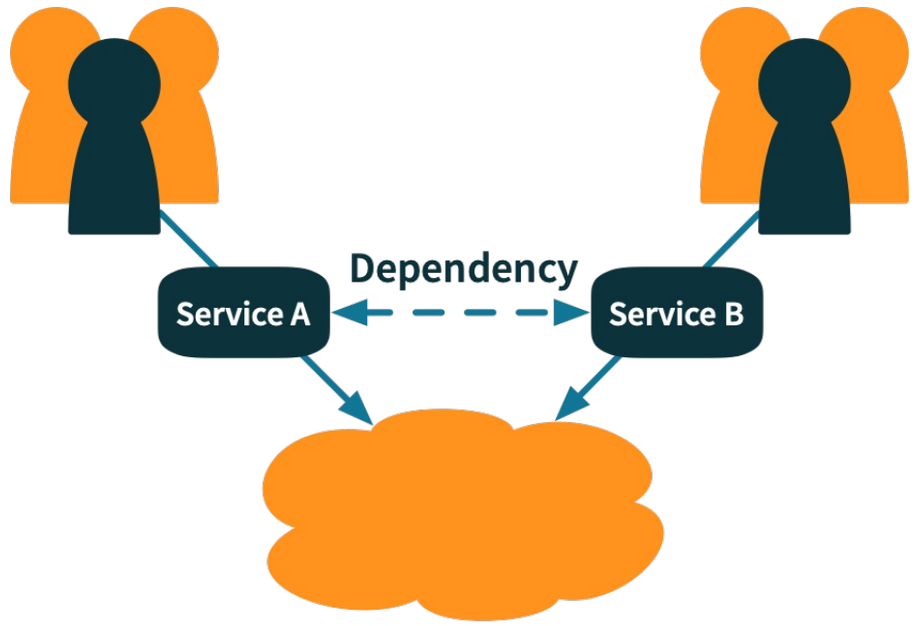
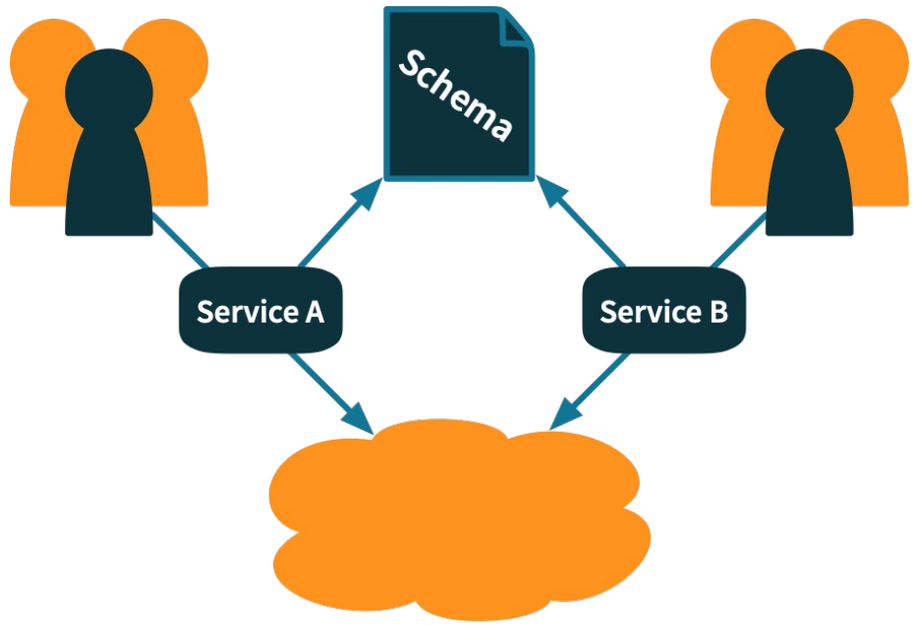
Schema First API Design

Autonomous Development Teams



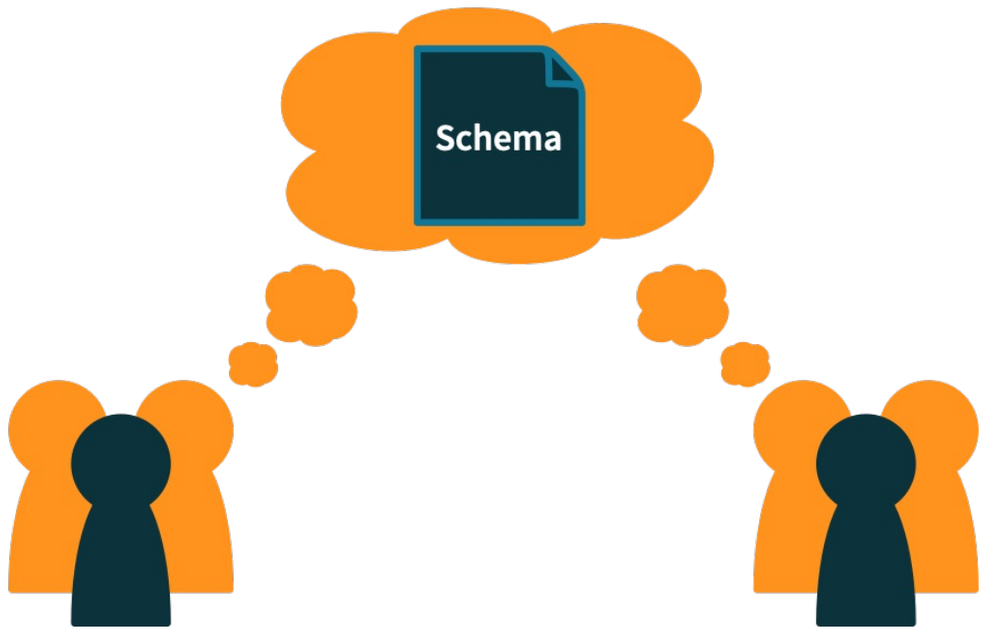
- Microservices aim for *Autonomy* between teams.
- Teams should be able to build/deploy features, independent of each other.
- However, dependencies exist. One team may depend on an API developed by another team.
- In order to maintain *Autonomy*, the teams need a shared *Schema*.

Schema First API Design



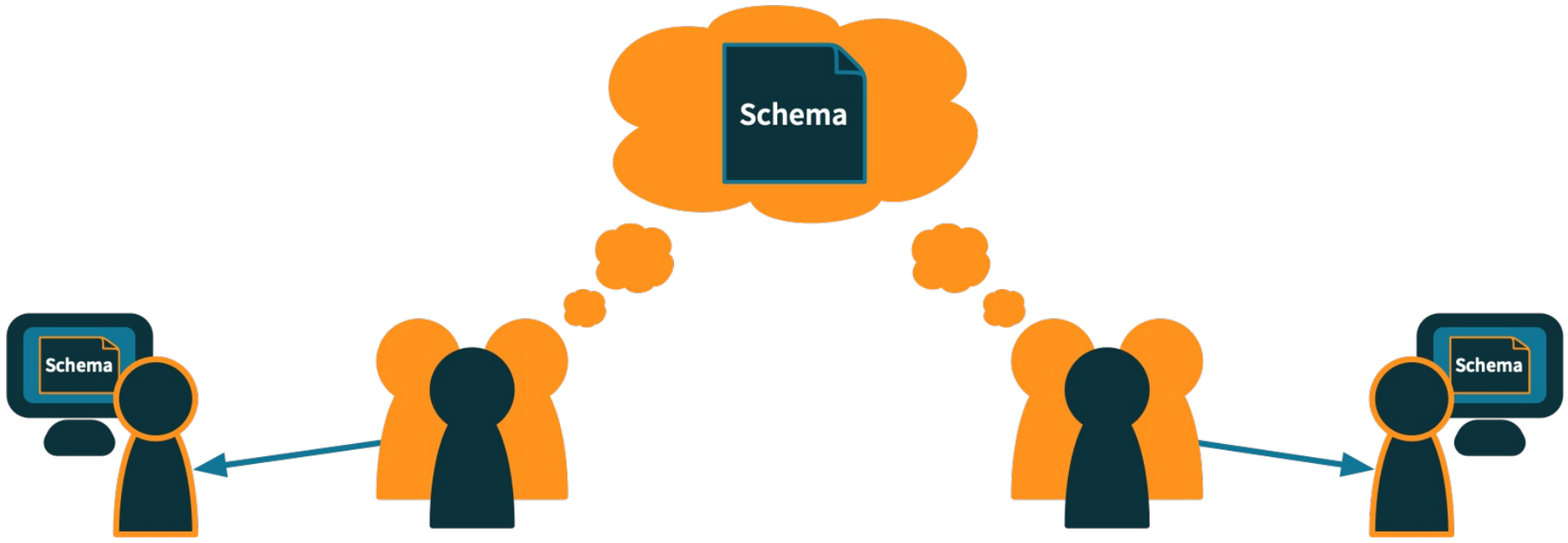
- Akka Serverless applications are designed *Schema First*.
- You define the *Schema* for your API, prior to writing any code.
- This *Schema* becomes a contract.
- Your *Service* will promise to fulfill that contract.

Designing Your Schema



- The *Schema* should be designed with the help of its clients.
- Clients of the API can provide feedback throughout the process.
- Once the design of the *Schema* is complete,
  - The API team can start writing code to support the API.
  - Client teams can start implementing consumers of the API.
- The presence of the *Schema* allows the teams to work independently toward a common goal.

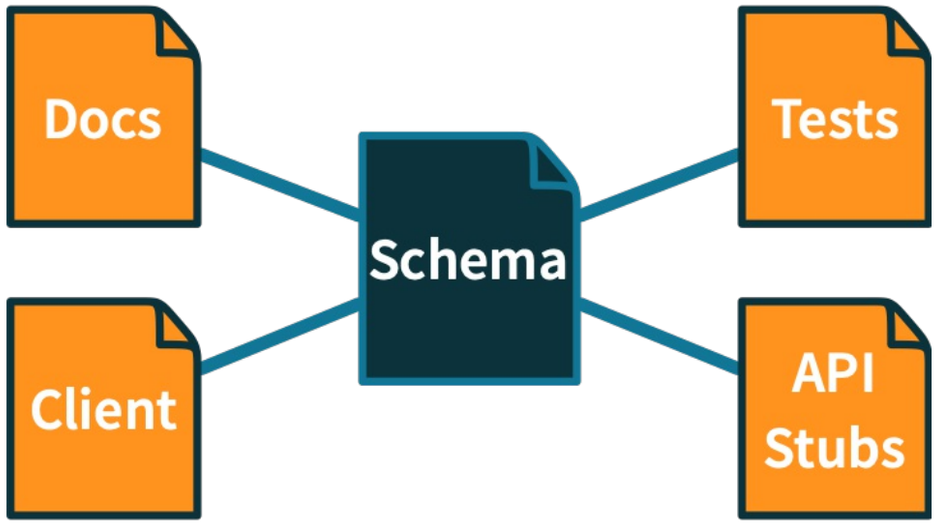
Upfront Communication



- During the design of your *Schema* communication may be required.
- Teams will share ideas, make revisions, provide reviews, etc.
- Once the *Schema* is complete, the need for communication will be reduced.
- If further communication is required, there is a common *Schema* to help frame the discussions.

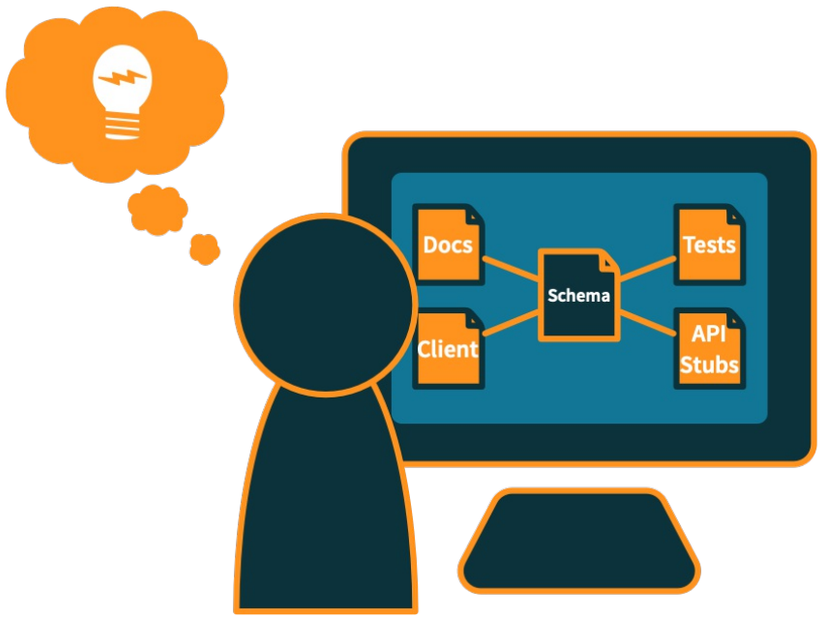
Benefits of Schema First

Artifact Generation



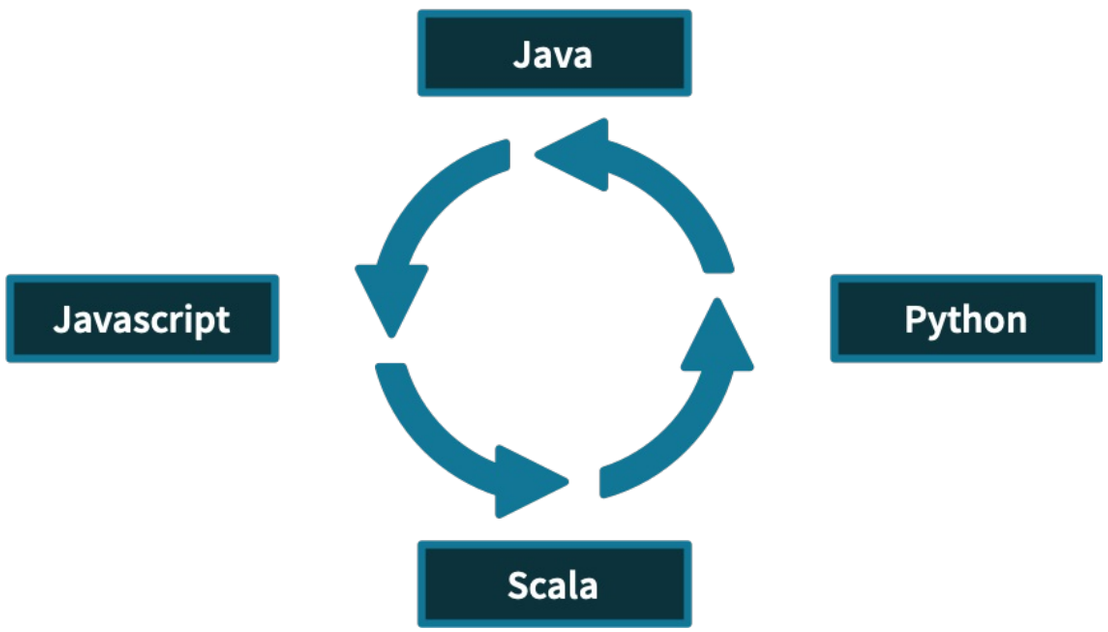
- The *Schema* can be used to generate a variety of artifacts, including:
  - Documentation.
  - Method stubs.
  - A *Client* library.
  - Black-box contract tests.
- These artifacts can be built by different teams, before the *Service* is complete.
- Tooling exists to generate some of them for you.

Improved Client Experience



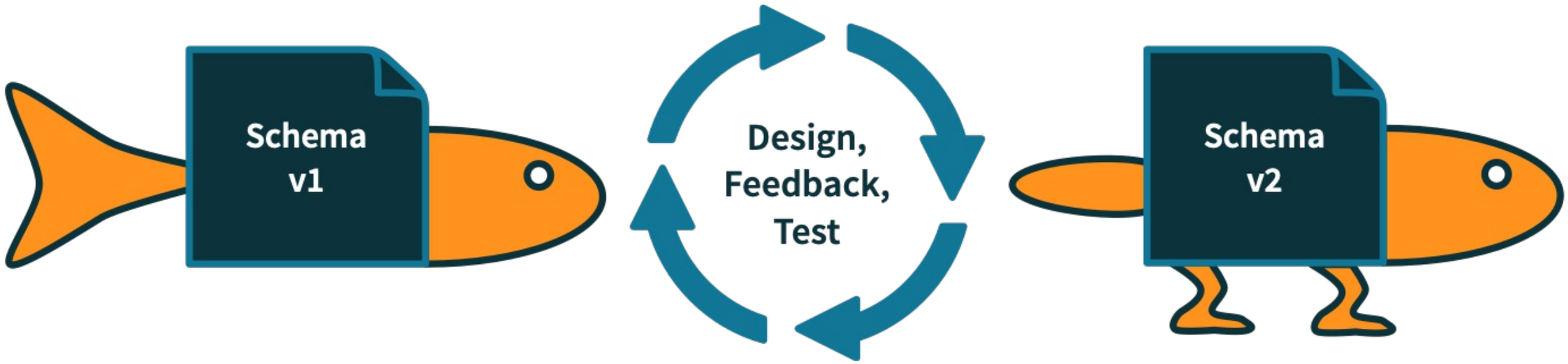
- These artifacts help to improve the client experience.
- Clients have access to a well-defined client library.
- When questions arise, they can fall back to the documentation.
- Contract tests mean that the API will behave the way it was promised.
- All of this means that developers can integrate with the API faster.

Improved Developer Experience



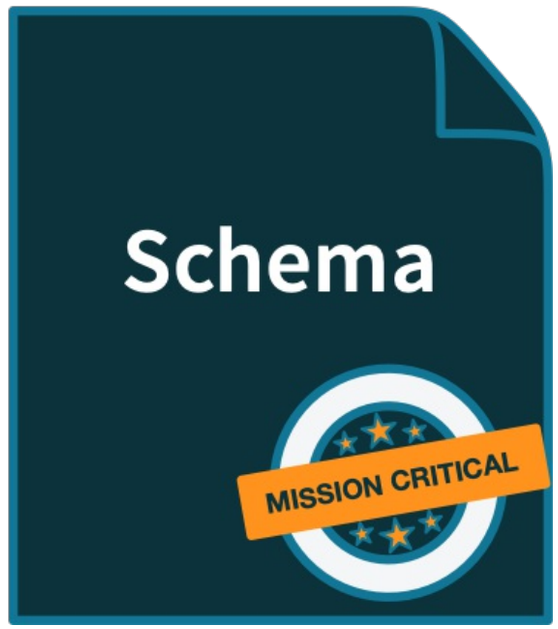
- Developers implementing the API also experience benefits.
- They have clear requirements, defined by the *Schema*.
- Contract tests verify when the requirements have been met.
- As long as it fulfills the *Schema*, and passes the tests, everything else is an implementation detail.
- The implementation (even the language) can change, without impacting clients.

Evolving the Schema



- *Schema First* design doesn't mean the API won't change. It will.
- It's about providing a framework to support change.
- Choose a *Schema* format that is flexible, and allows for evolution.
- Where more dramatic changes are required, use versioned *Schemas*.
- When changes are required, go back to the design process, and solicit feedback from clients.

Mission Critical APIs



- *Schema First* shines in mission critical APIs.
- Upfront design means you don't spend time working on an API that doesn't meet requirements.
- It provides the capabilities to scale out the work to multiple teams.
- When required, it provides a framework to evolve the design.
- The end result is a smoother development process, and more productive teams.