"Choreo: The Next-Generation Digital Platform As A Service"

https://wso2.com/choreo/

Srinath Perera,

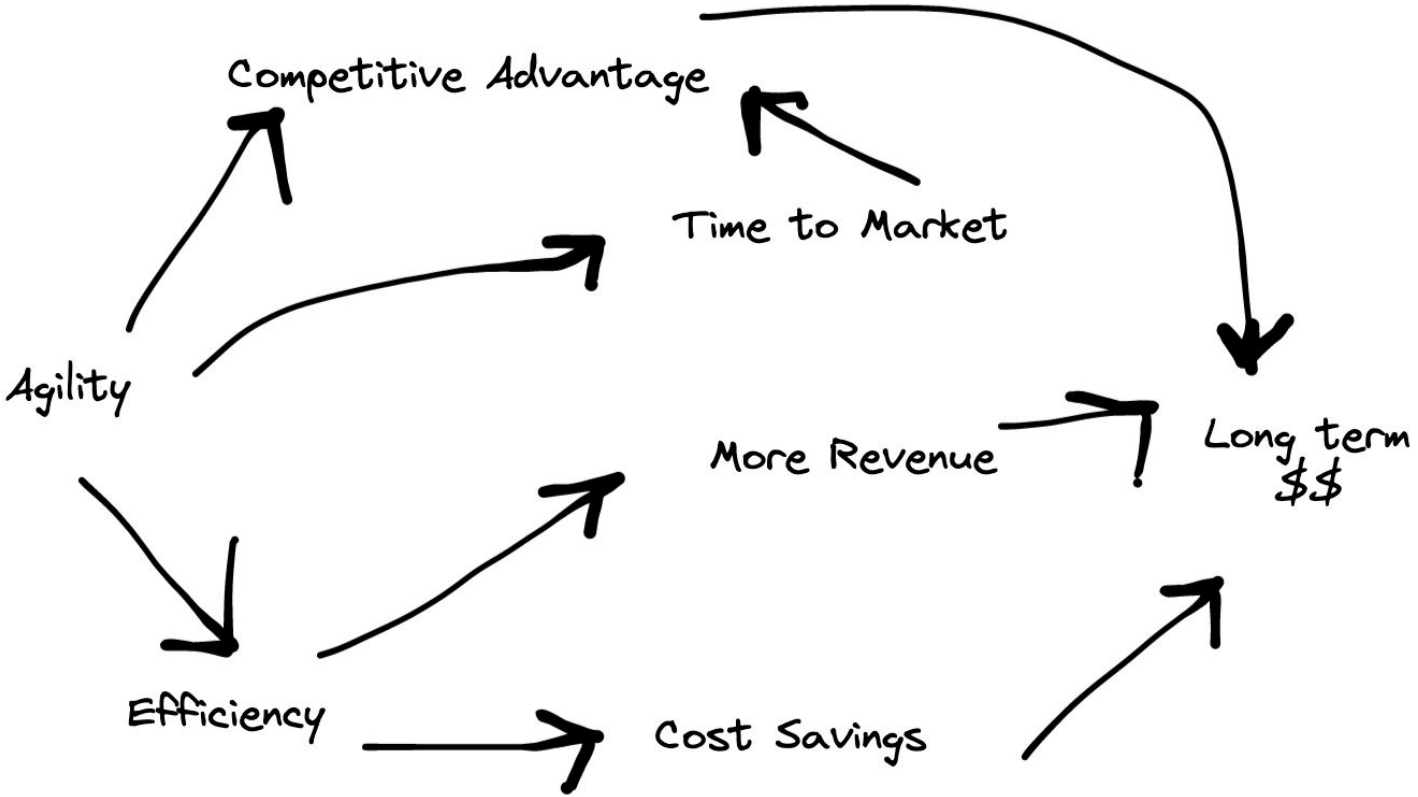
Chief Architect, WSO2, Mamber, Apache Software Foundation

Understanding the Enterprise

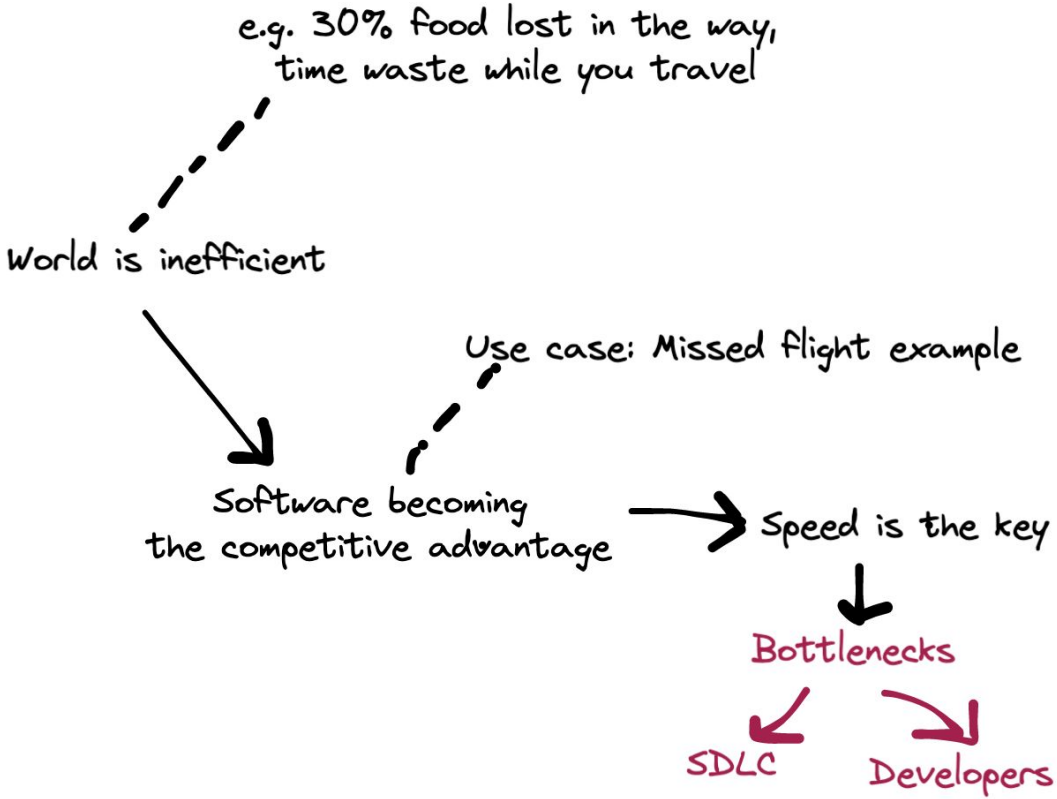
Software is just a means
to an end, not the end

Enterprises value long time $$

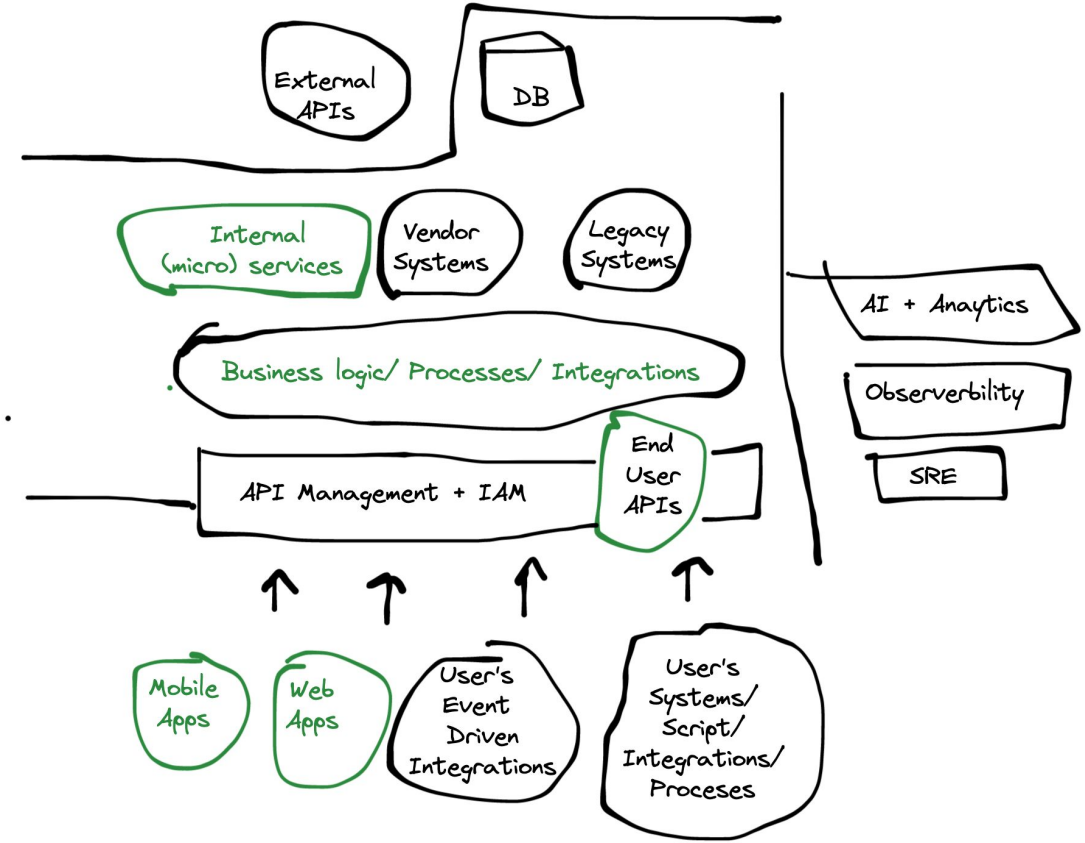
What does Enterprises Looking for?



Software is eating the world



How does Enterprise Architecture Look like?



Software Development is small part of end-to-end SDLC

Design

- Solutions Architecture
- System Architecture
- User Stories
- Interfaces
- Security Architecture
- Compliance

Development

- Source Control
- Debugging
- Code/Low-code
- Integration
- Code Reuse
- Marketplaces
- Code Templates
- Microservices
- Artificial Intelligence

Deployment

- Builds
- Containers
- CI/CD
- GitOps
- Pipelines
- Serverless
- Kubernetes
- Vaults
- Environments
- Releases

Testing

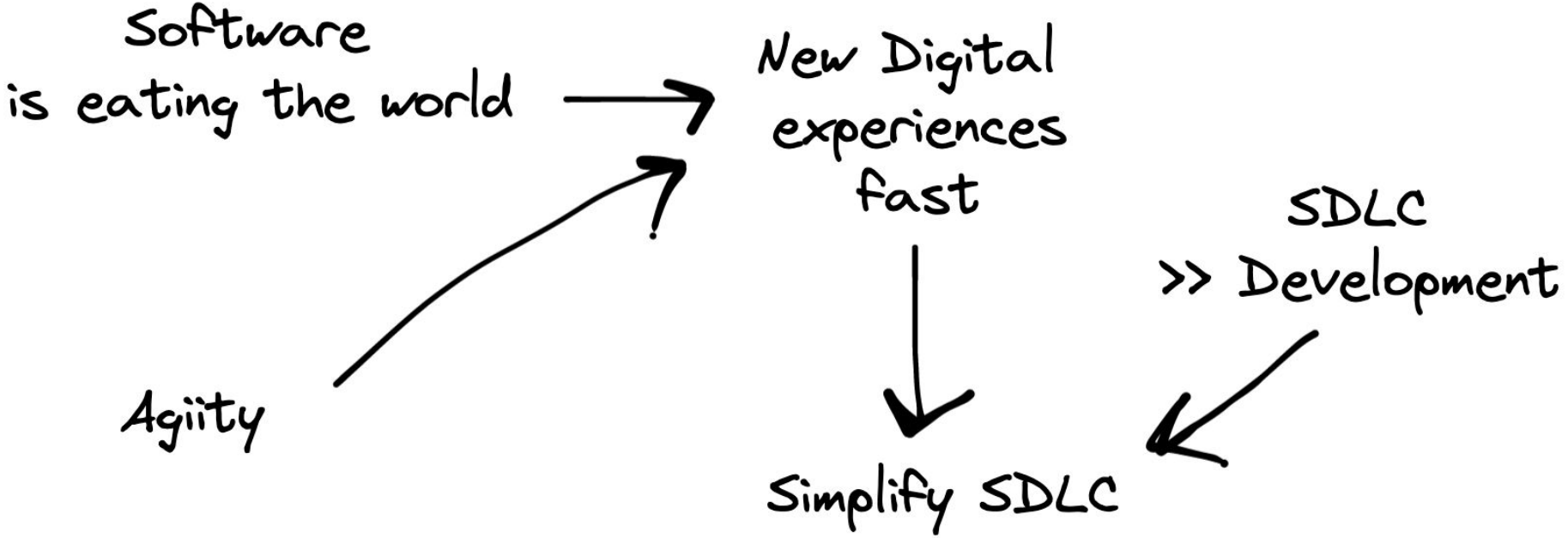
- Blue/Green
- Canary
- End to End Tests
- Unit Tests
- API Tests
- Load Tests
- Long Running Tests

API Management

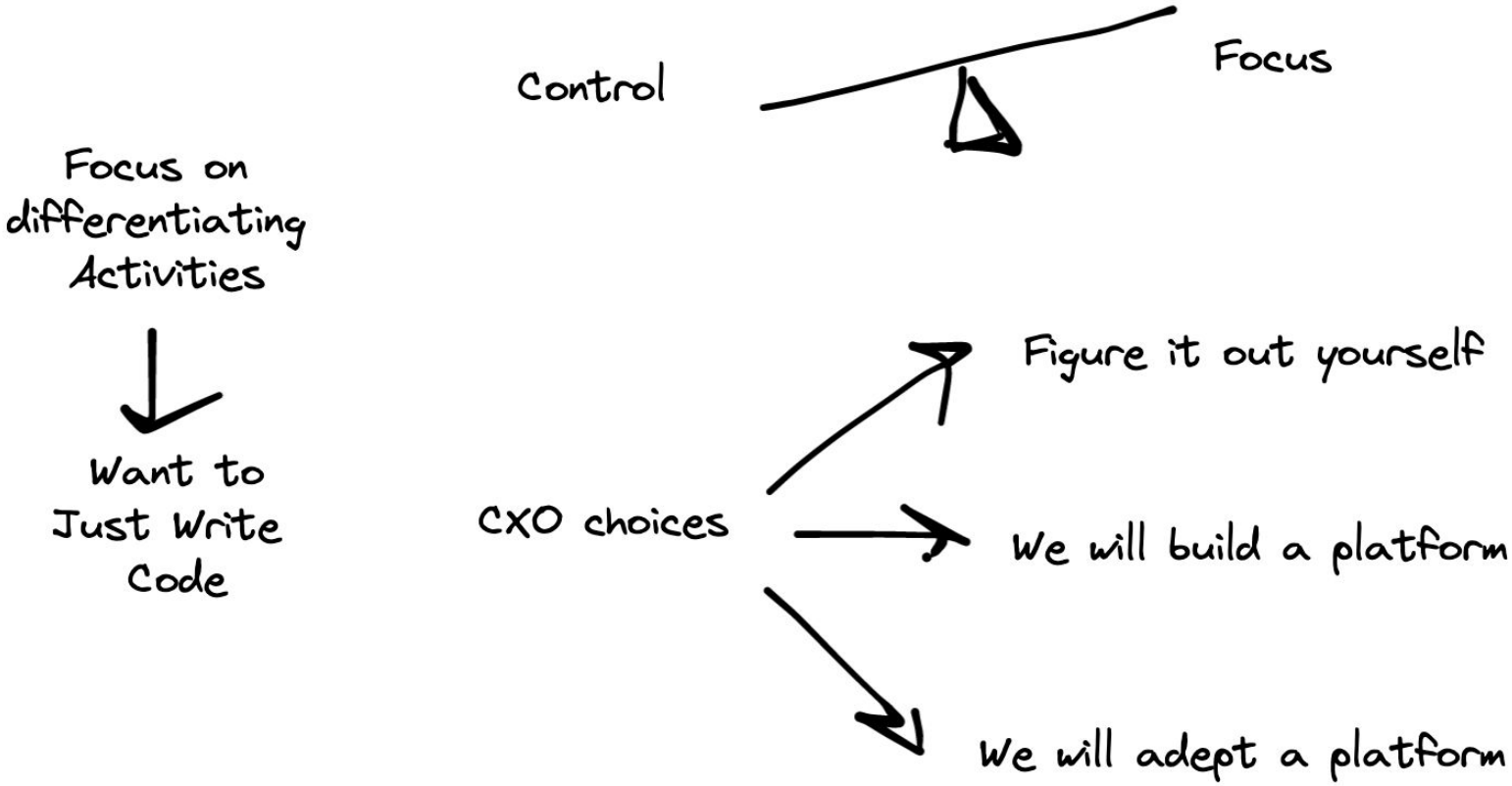
- Governance
- Policies
- Security
- Edge Management
- Usage Plans
- Developer Portal
- Monetisation
- Documentation

Observability and Insights

- Alerts
- Monitoring
- Performance Metrics
- Latencies
- Predictions
- Troubleshooting
- Root Cause Analysis
- Machine Learning
- Artificial Intelligence

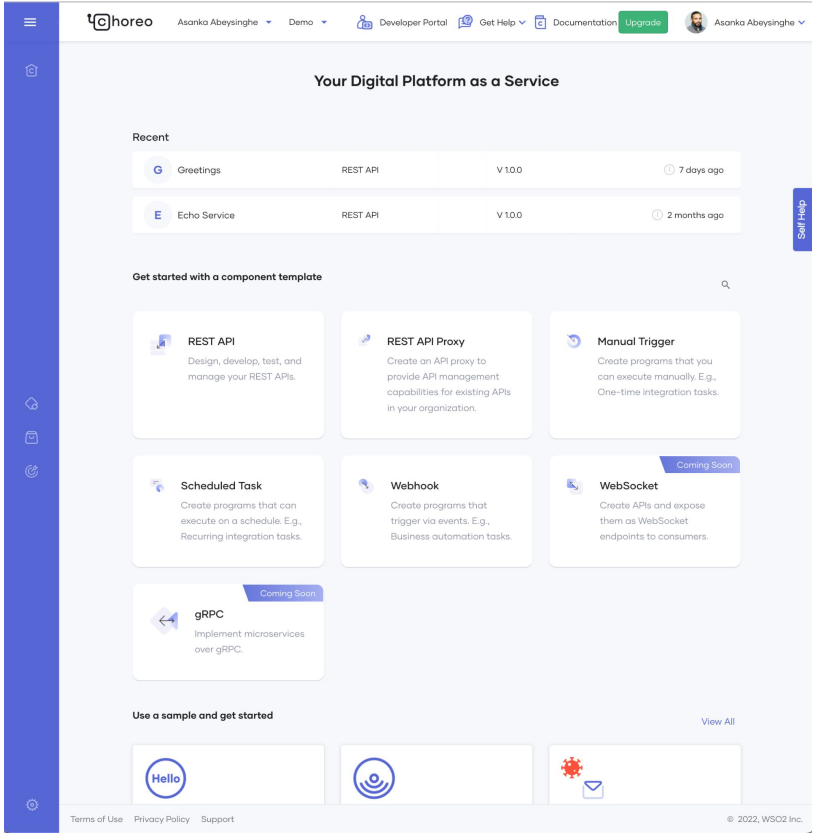

Software
is eating the world
New Digital
experiences
fast
SDLC
>> Development
Agility
Simplify SDLC

Goal of the Platform - Just add developers/ Just write the code



Choreo allows you to build and run…

- **APIs**
  - REST and Non-REST network endpoints exposed to internal and external consumers such as web apps or mobile apps.
- **API Proxies**
  - REST and Non-REST proxies to network endpoints that are hosted within or outside of Choreo.
- **Integrations**
  - Event or schedule based triggers used for integrating two or more systems. Ex: Creation of a billing account upon every customer registration.
- Microservices (TBD)
- API Products (TBD)

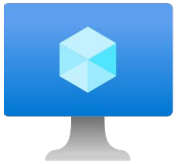


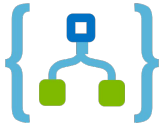
Infrastructure

Development

Deployment

Observability and Insights



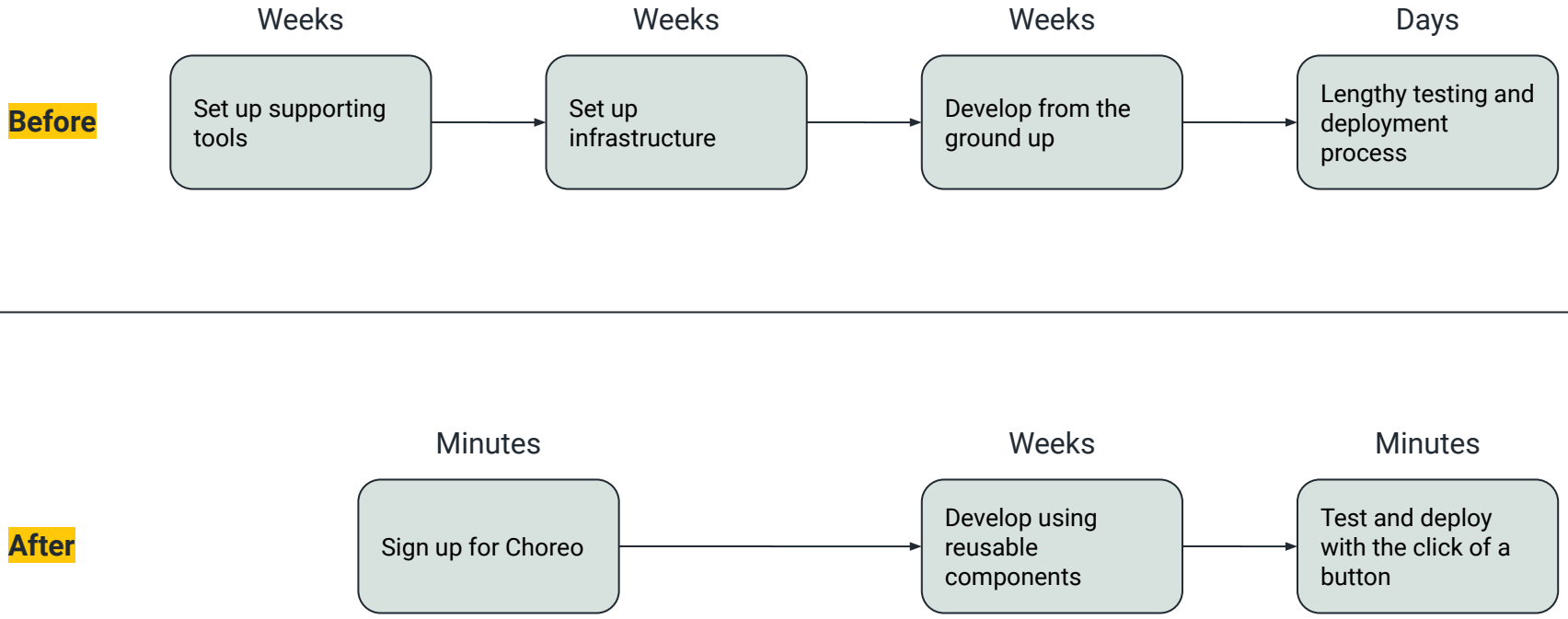


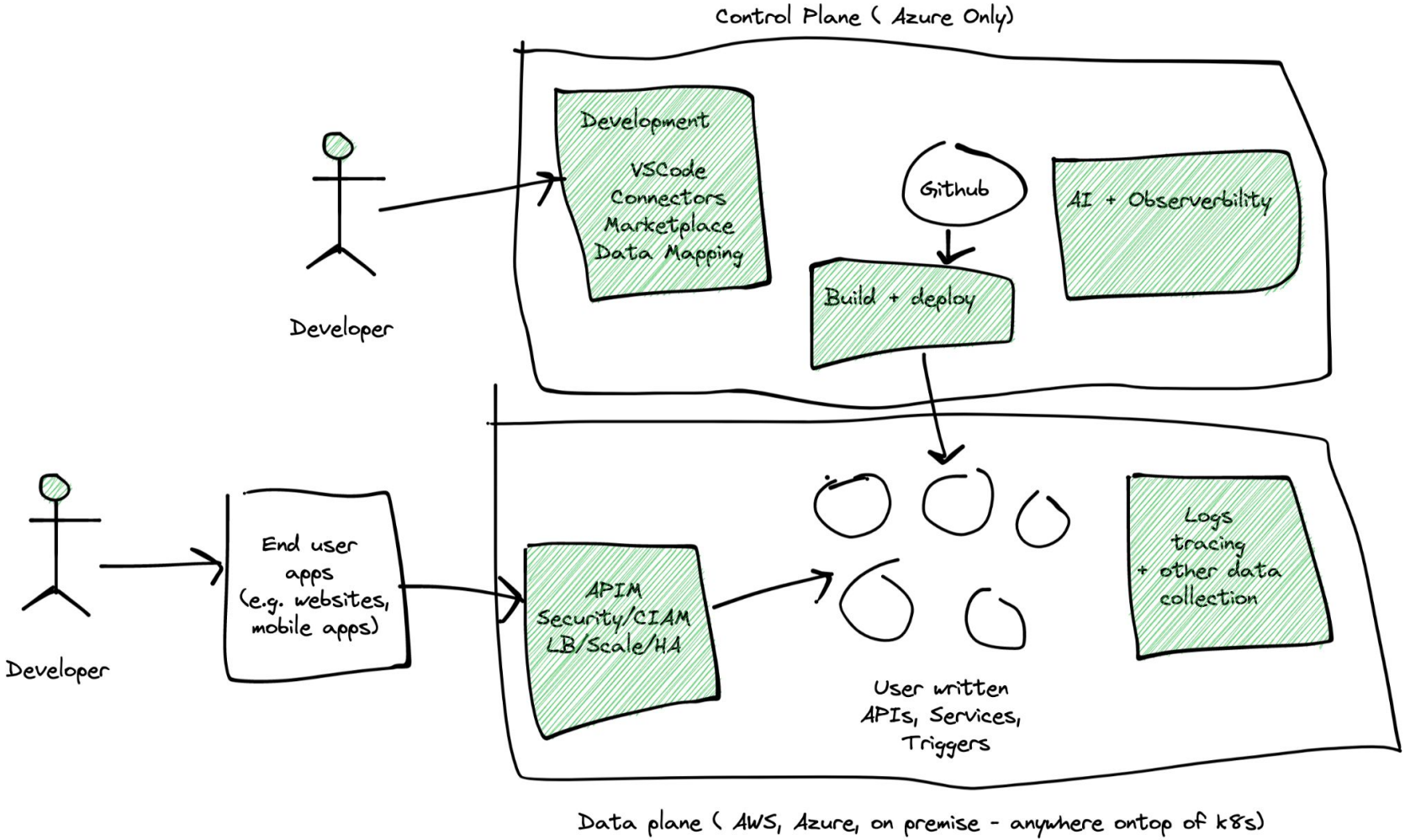
Choreo

Digital Platform as a Service

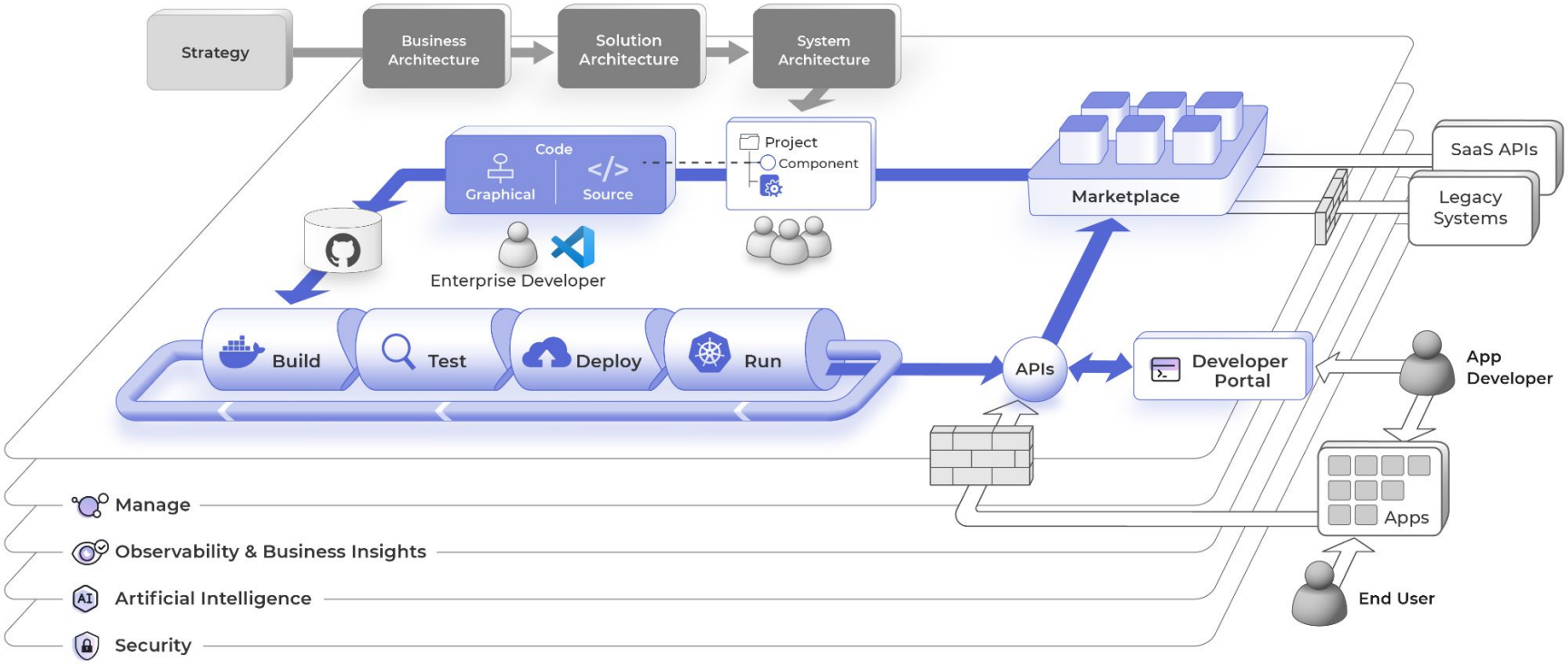
Demo

The developer experience: before and after Choreo




Control Plane (Azure Only)
Development
VSCode
Connectors
Marketplace
Data Mapping
Github
Build + deploy
AI + Observerbility
Developer
Developer
End user
apps
(e.g. websites,
mobile apps)
APIM
Security/CIAM
LB/Scale/HA
Logs
tracing
+ other data
collection
User written
APIs, Services,
Triggers
Data plane (AWS, Azure, on premise - anywhere ontop of k8s)

Choreo Architecture



Microsoft and Azure Services (e.g: AKS, Service Bus, Key Vault, MSSQL, GitHub, Log Analytics, etc.)

Choreo takes care of the end-to-end SDLC

Design

- Solutions Architecture
- System Architecture
- User Stories
- Interfaces
- Security Architecture
- Compliance

Development

- Source Control
- Debugging
- Code/Low-code
- Integration
- Code Reuse
- Marketplaces
- Code Templates
- Microservices
- Artificial Intelligence

Deployment

- Builds
- Containers
- CI/CD
- GitOps
- Pipelines
- Serverless
- Kubernetes
- Vaults
- Environments
- Releases

Testing

- Blue/Green
- Canary
- End to End Tests
- Unit Tests
- API Tests
- Load Tests
- Long Running Tests

API Management

- Governance
- Policies
- Security
- Edge Management
- Usage Plans
- Developer Portal
- Monetisation
- Documentation

Observability and Insights

- Alerts
- Monitoring
- Performance Metrics
- Latencies
- Predictions
- Troubleshooting
- Root Cause Analysis
- Machine Learning
- Artificial Intelligence

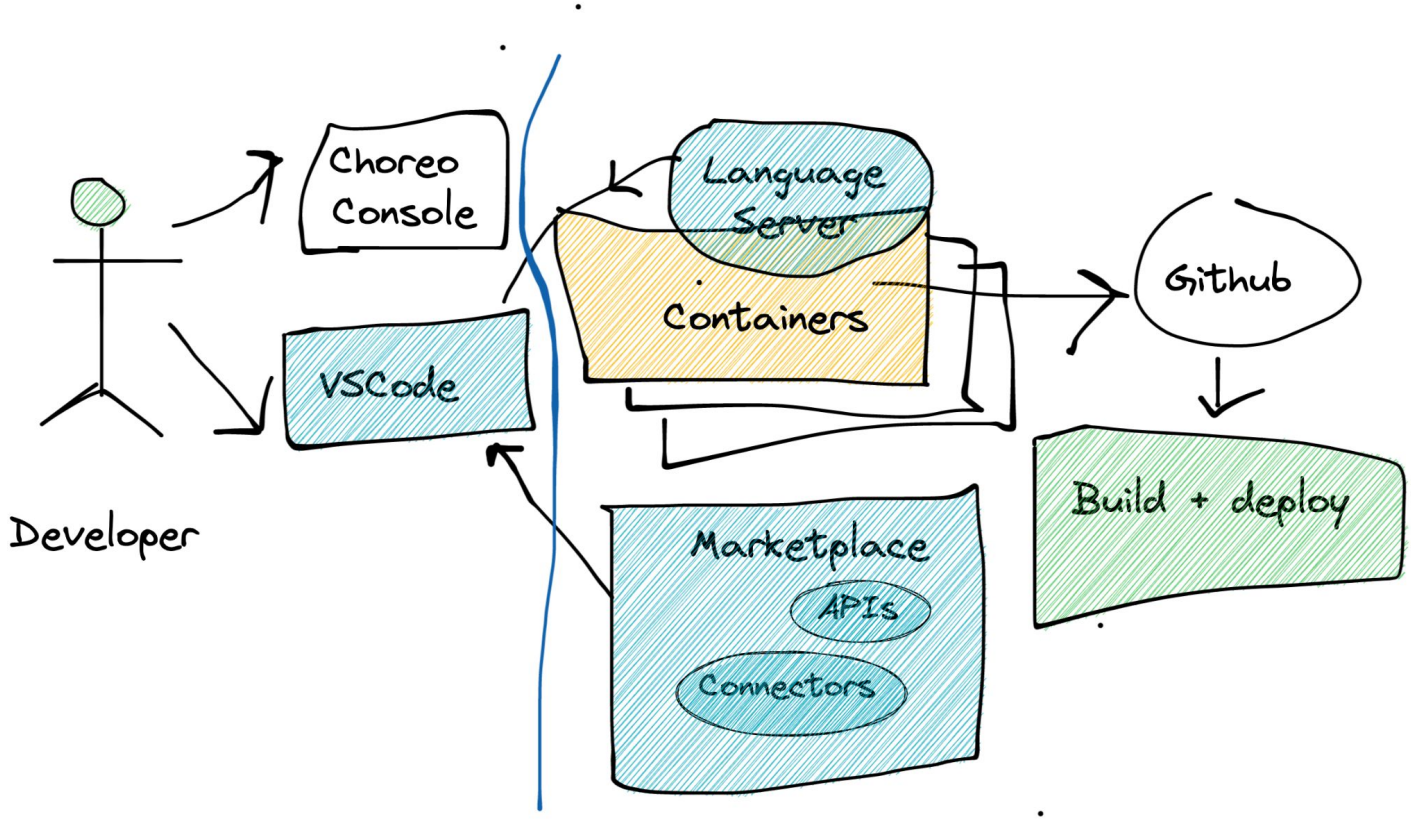
Platforms concentrate

data, needs, $$

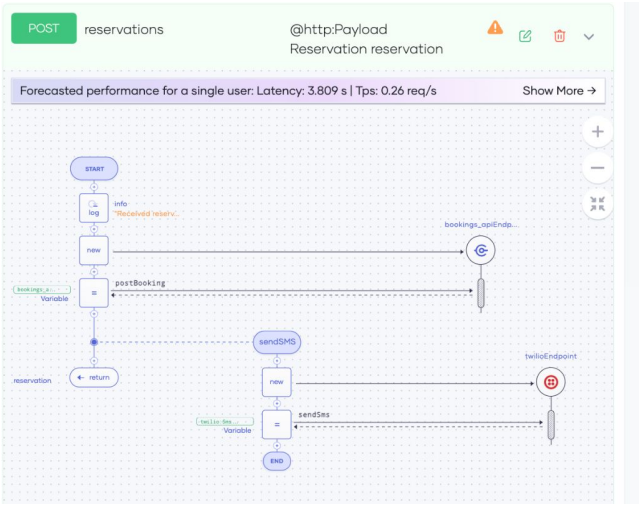
so that it can add

AI, testing, code help, root cause analysis, and other features that are unlikely to happen at individual developers or rarely at the organization level

Development Architecture



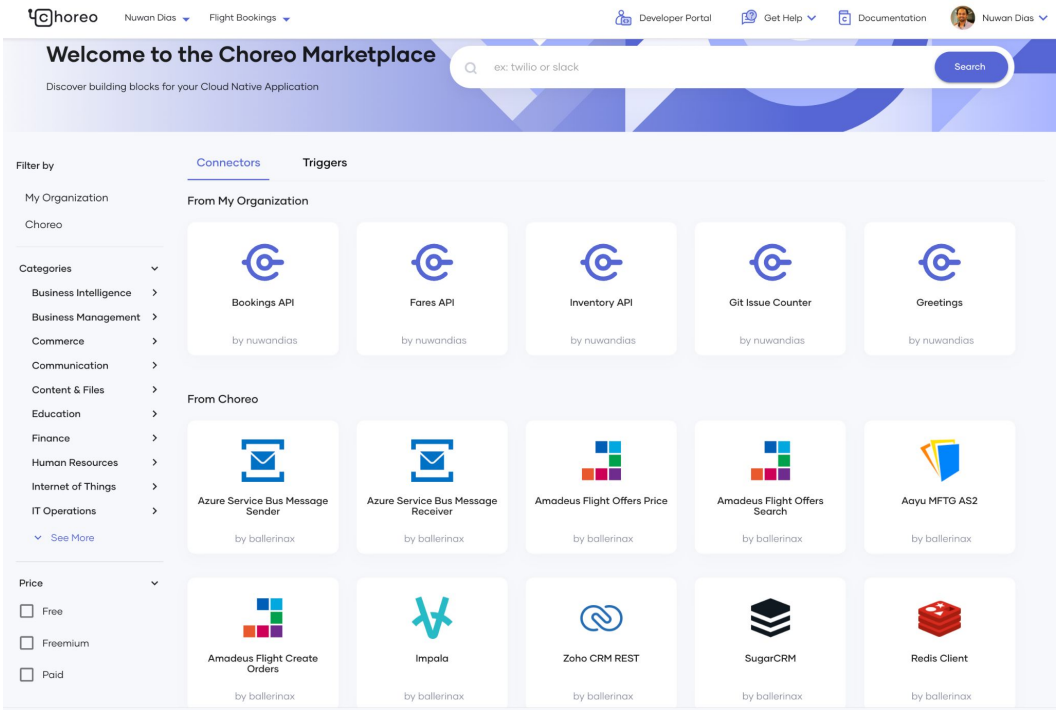
Development




```ballerina
import ballerinax/twilio;
import nuwandias/bookings_api;
import ballerina/log;
import ballerina/http;

configurable string twilioAccountSId = ?;
configurable string twilioAuthToken = ?;
configurable string clientId = ?;
configurable string bookingsClientSecret = ?;

Run | Debug | Try it
service /flights on new http:Listener(9090) {

    Forecasted minimum latency 3.809 s (with 0.26 tps for single user)
    resource function post reservations(@http:Payload Reservation reservation) r

        log:printInfo("Received reservation requeest for " + reservation.flightN
        bookings_api:Client bookings_apiEndpoint = check new ({auth: {clientId:
        bookings_api:BookingRecord _ = check bookings_apiEndpoint->postBooking({

        worker sendSMS returns error? {

            twilio:Client twilioEndpoint = check new ({auth: {accountSId: twilio
            twilio:SmsResponse _ = check twilioEndpoint->sendSms("+1984687256018

        }
        return reservation;
    }
}
```

- Program in both Code and Graphic.
- Out of the box integration with Github.
- AI for performance predictions and other development assistance.
- Easy reuse of APIs through the Marketplace.
- Automatic security handshakes.
- ~~Marshalling and unmarshalling data.~~
- Secure storage of sensitive data.
- Visual data mapping (TBD).
- Single source of truth for Code, Graphic and Spec.

API Marketplace



- For sharing and reuse of APIs both within and outside the organization.
- Contains APIs and event sources (data sources to be added).
- Developer can both consume and contribute to the API Marketplace.
- Searching and filtering.
- Streamlines API integration by taking off the need to developer boiler plate code for security and resilient connectivity.

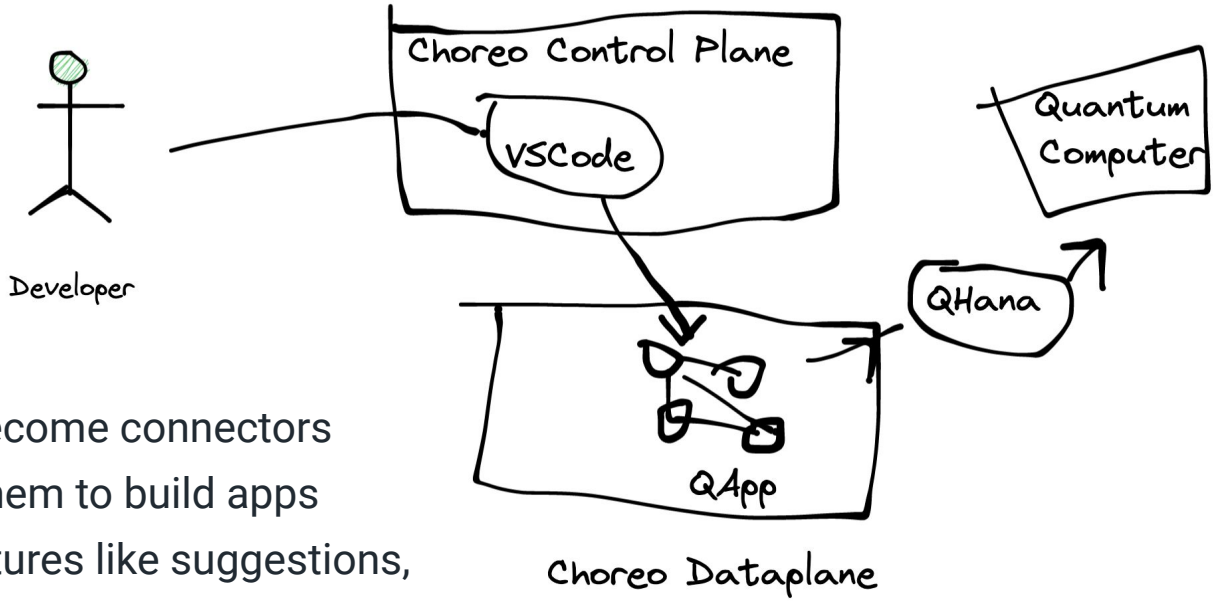
Ballerina vs. Other Programming Languages

- Support for programming languages other than Ballerina is coming
  - Any programming help in VSCode you can use
  - Once you develop, Choreo can do the rest
- Of course, Ballerina will give you more
  - Code and visual representation that is in sync is only in Ballerina and more (next slide)
  - but switching to a new Programming Language happens slowly

Why Ballerina?

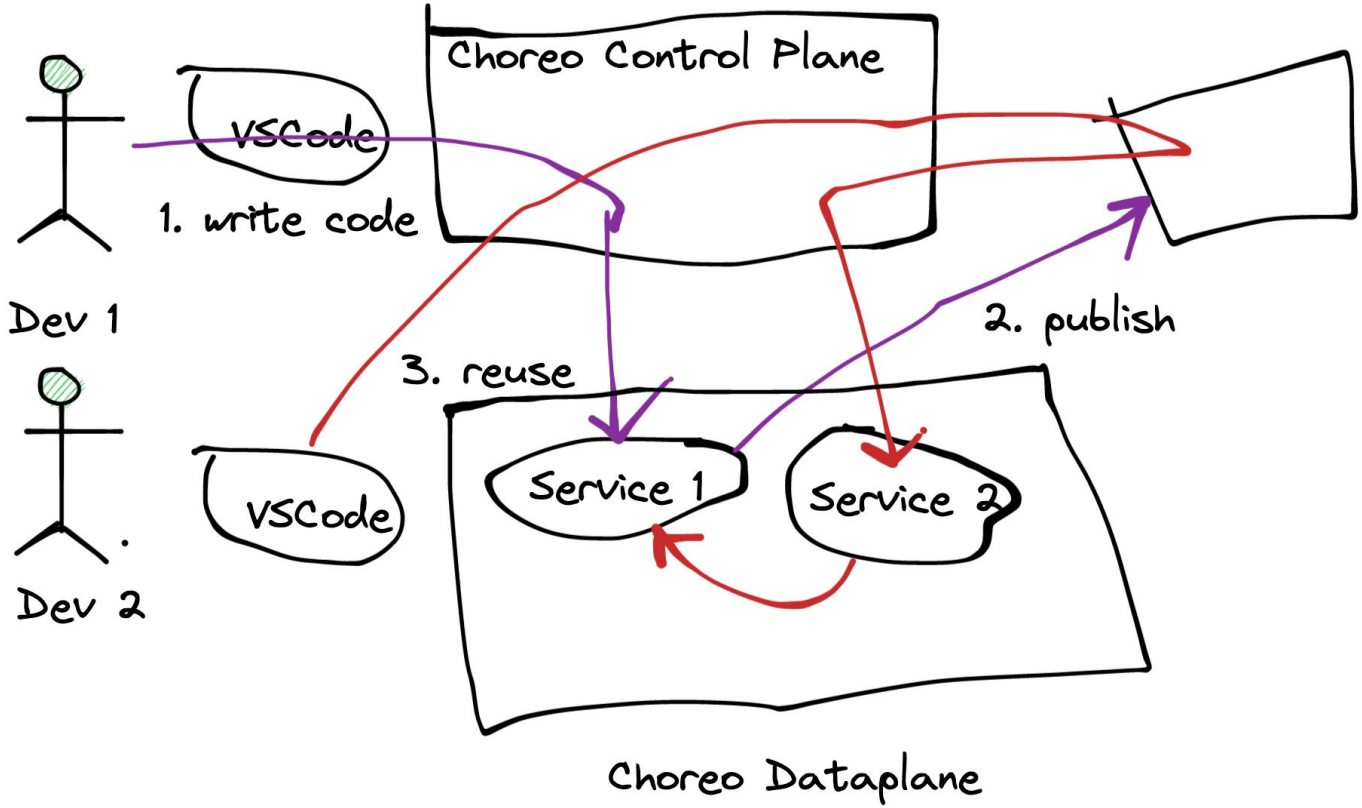
- Optimized for building new APIs, Services and communicating with other APIs and services
  - E.g., Generate clients from swagger
- Native support for JSON, and XML, avoiding the need for explicit data binding
- Rich support for data processing
- Graphical textual Parity
- Understand networked programming at the compiler level, unlocking optimizations
- More details
  - https://blog.jclark.com/2022/05/why-ballerina-is-language.html
  - https://ballerina.io/

Quantum Computing as a Case Study



- Quantum Operators Become connectors
- Developers combine them to build apps
- We will get a lot of features like suggestions, performance suggestions, and data mapping for free, etc

User Story - write a Service, get it out as API, reuse



Open Problems (Research + Applications)

- Provide latency guarantees while scaling/HA
- Performance predictions while programming
- Code suggestions
- Deep Observability and Root cause analysis
- Deployment Optimizations
  - E.g., co-locate pods that communicate a lot
- Test Generation
  - Test data generation
  - Test case generation (e.g., from few sample messages)

Lessons Learned

- Multi-cloud slows you down; embrace the cloud you are in
- Choice of VSCode saved us a lot of work
- Making sure the team has understood the user stories is hard
- Defining APIs between subsystems early lets us move fast
- Working hard to use standards (Swagger, containers, k8s) and available tools help a lot
- Making ML work in real life, and MLOps are hard

Thanks!

wso2.com