BUILD & REPEAT

CI/CD FOR TWILIO PROJECTS

Test and deployment automation

Jessica Cregg & Ana Andrés del Valle
jcregg@twilio.com & aandresdelvalle@twilio.com

TWILIO SIGNAL

CREATE DIFFERENT ENVIRONMENTS AND ASSIGN DEDICATED TWILIO ACCOUNTS

Twilio dev account

DEV ENVIRONMENT

Multiple individual environments
Unit and Component tests

Happy path / unhappy path
interface tests

Twilio stage account

STAGING ENVIRONMENT

E2E integration tests
full interface and user
acceptance tests

Twilio prod account

PRODUCTION ENVIRONMENT

Production checks and
monitoring

Release management

Pipeline testing and deployment automation

EXAMPLE: Customer Satisfaction SMS survey

Twilio Projects : Think automation first

Test Driven / Behaviour Driven Development (TDD/BDD)

Define all user flows and their corresponding tests
Given-When-Then framework (e.g Cypress)

1. Set-up Twilio Dev Environment (Studio Flow + Function)
2. Automate Dev Env Tests (e.g. Cypress)
3. Automate Deployment in Staging (CLI & API + CI/CD pipeline scripts)
4. Automate E2E integrated tests (Cypress)
5. Automate Deployment in production & repeat as needed

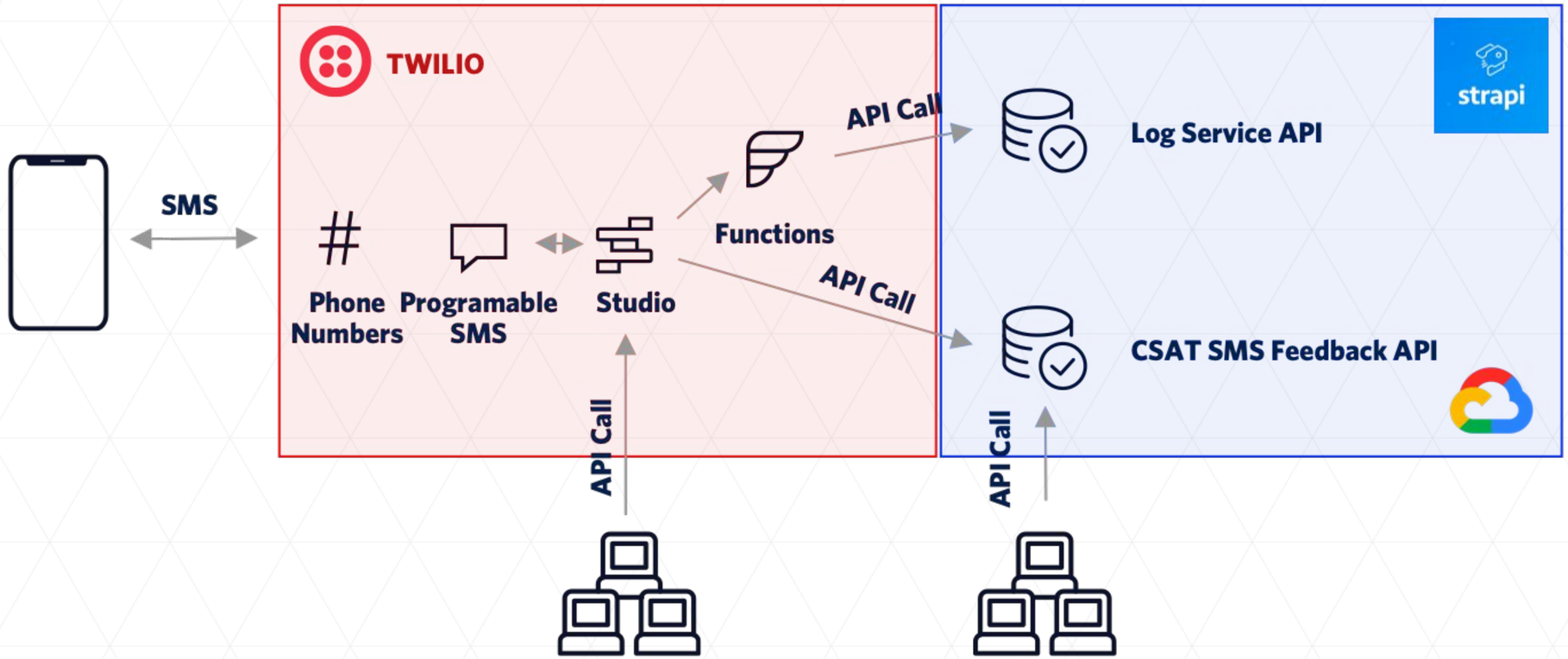
For example: Jest (node, React, Vue) or Cucumber (Java, Ruby)

HANDS-ON EXAMPLE

TWILIO SIGNAL

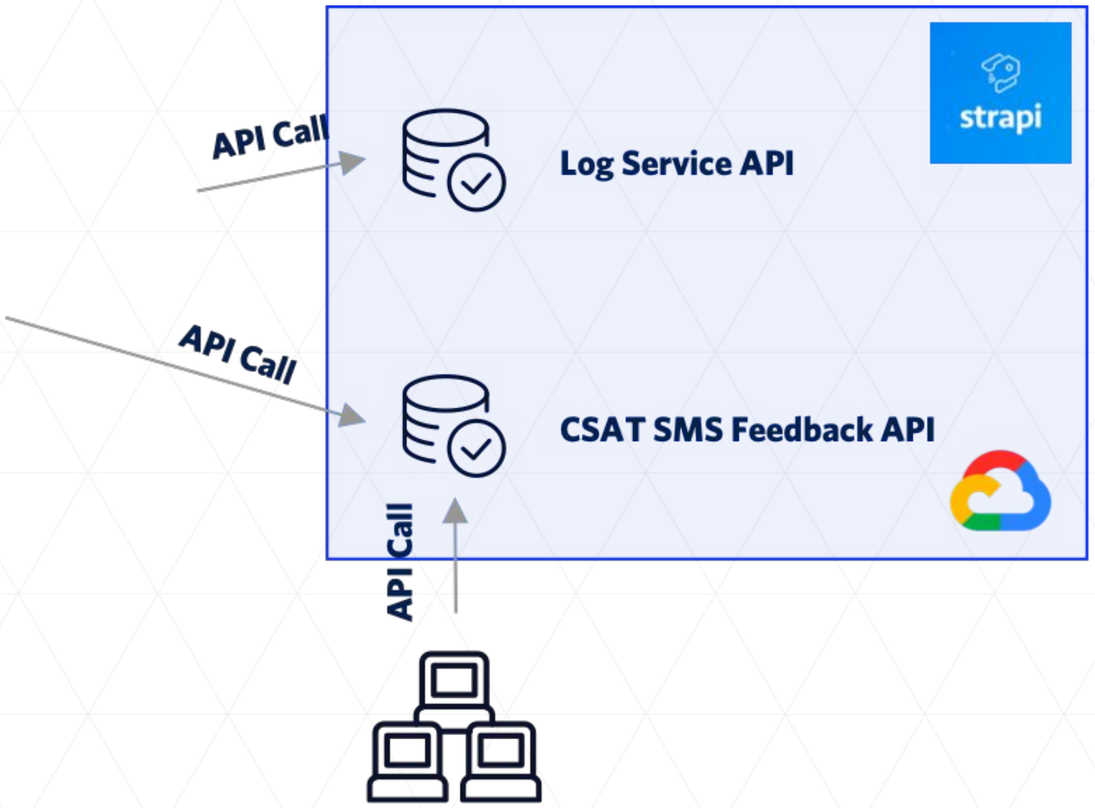
DESIGN A TWILIO PROJECT WITH AUTOMATION IN MIND

EXAMPLE: Customer Satisfaction (CSAT) SMS survey service



TWILIO SIGNAL

STRAPI DATABASE INTERFACE AND API



1. Implement Strapi dev Service from Strapi Quick Start (local machine or cloud dev machine) - REST API

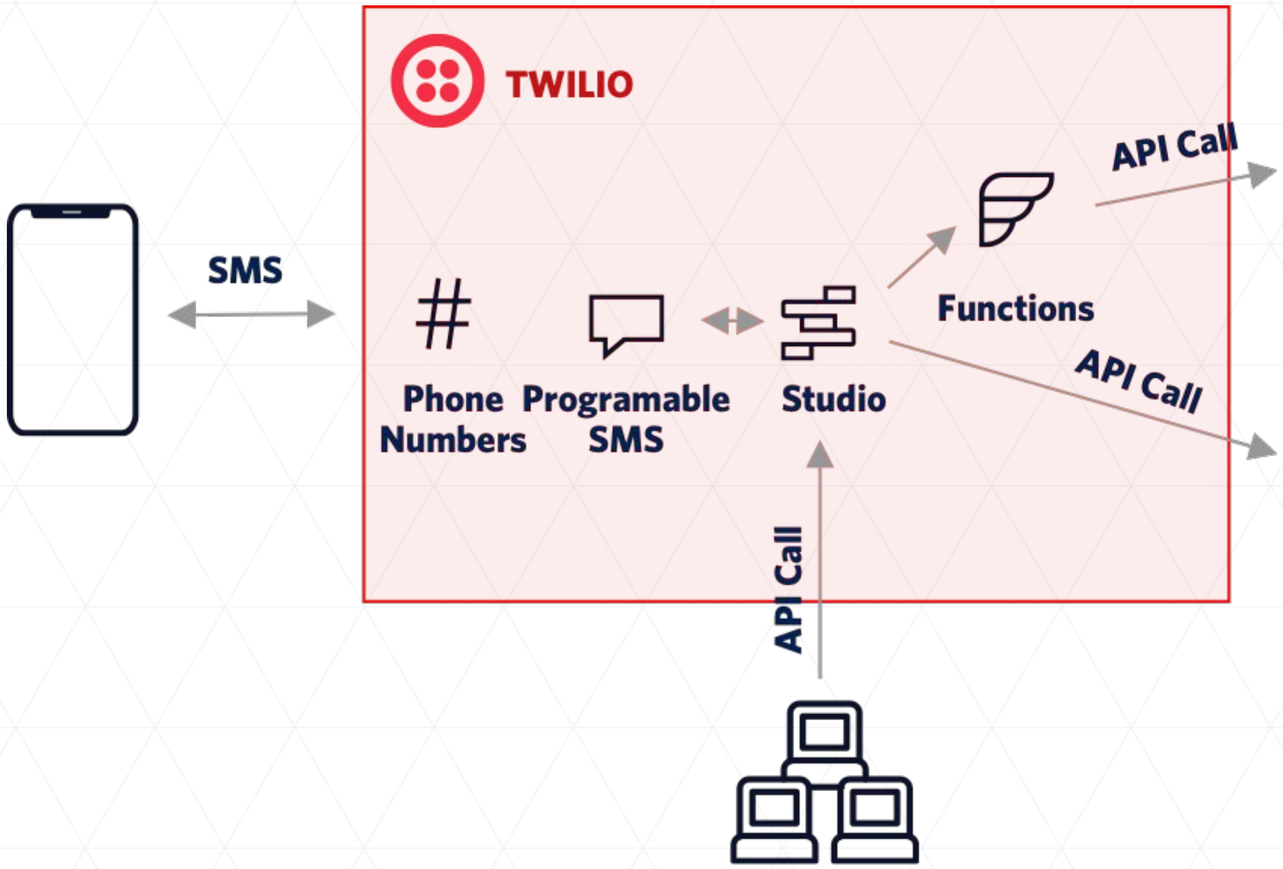
2. Create 2 Collection Types (2 APIS)

csat-form:	log:
name	date_time
surname	entry
phone_number	
service_score	
recommendation_score	
feedback	
job_id	

3. Create User and access token for Collections API

4. Test API (e.g. Postman collections)

TWILIO SMS INTERFACE FRONT-END



Code as if DB API existed with agreed contract (e.g. from Strapi CSAT-form / Log)

1. Create Functions to POST data into Log DB
2. Create the SMS survey with Studio following the BDD requirements
3. Add the Functions to the Studio flow
4. Buy a number (create a Regulatory bundle if needed)
5. Use a Mock Server to test your flow HAPPY PATHS & UNHAPPY PATHS

When all tests pass, you're ready to program automation 😎

TWILIO SIGNAL

TEST AUTOMATION: TWILIO COMMUNICATIONS INTERFACE

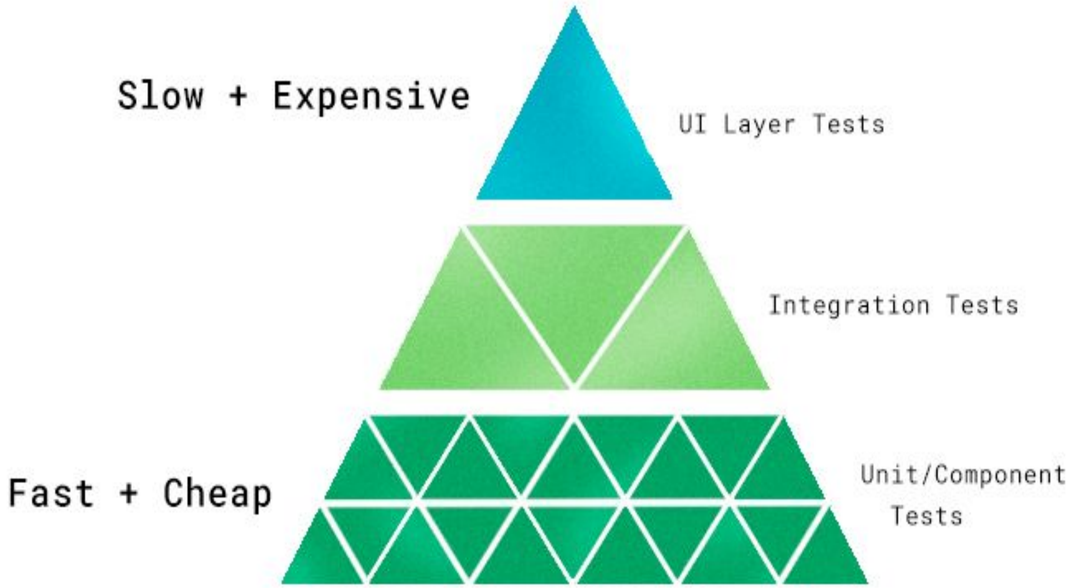
Functions + Twilio Apps : Test Driven Development (TDD)

Studio → **Behavior Driven Development**

Define all user flows and their corresponding tests
Given-When-Then framework (e.g. Cypress)

Twilio powers Communications User Interfaces across the public switch telecom network (PSTN)

⚠️ **Testing may incur in platform costs**



TWILIO SIGNAL

DEPLOYMENT AUTOMATION: USE TWILIO APIS & CLI

Prioritise order of service deployment by dependency:

- ➔ Start by deploying services with no dependencies
- ➔ Finish by deploying services that use previously deployed dependencies
- ➔ Identify static resources and code deployments
- ➔ The first deployment creates all resources, static and dynamic
- ➔ Consecutive deployments usually only update dynamic resources (software features)

Note Static resources could be any of the following:
Phone numbers, fixed IP addresses, DNS, SSL certs

STEPS

1. Deploy all external services to Twilio — gather the service URLs and authentication requirements
2. Deploy the server less functions and assets in Twilio — using the Twilio CLI serverless plugin
3. Fetch the Studio Flow from Dev environment — using the Twilio Studio API
4. Update Flow with the Staging services

THANK YOU

TWILIO SIGNAL

CHECK FONTS

FUTURA PT COND MEDIUM

Futura PT Book

FUTURA PT COND MEDIUM

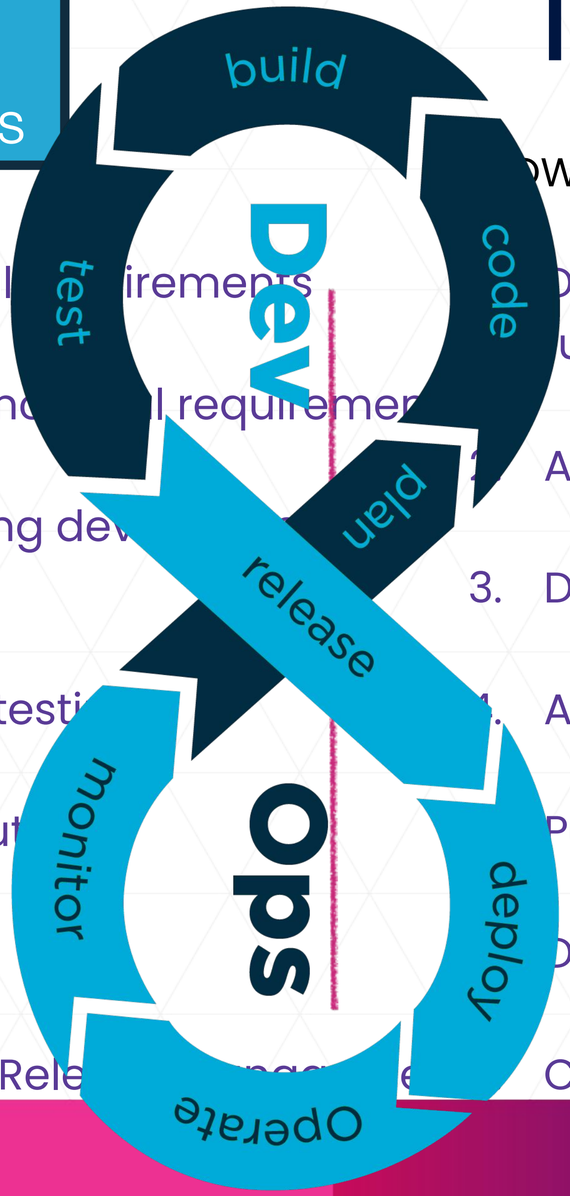
Futura PT Book

If the text on the left-hand column looks different, you will need to download
the correct fonts from our brand assets at (provide link)

TWILIO SIGNAL

LIFECYCLE OF A TWILIO PROJECT

DEVOPS AUTOMATION = + SPEED - MISTAKES



INITIAL SOLUTION / MVP

1. Business problem/request: functional irements
2. Functional technical design + non fun l requiremer
3. Set-up and distribution of work among de teams
4. Individual service development and testi
5. Integration and E2E testing of full solut
6. Deployment in production
7. Production checks and monitoring - Rele

W TO REFINE / MODIFY / SOLVE BUGS

1. Describe the change in Business terms or Non unctional terms
2. Assess of technical impact
3. Distribute work
4. Add changes to local code and test
5. Push and test changes in Integrated environment
6. Deploy in production
7. Check and monitor production