CYCLR

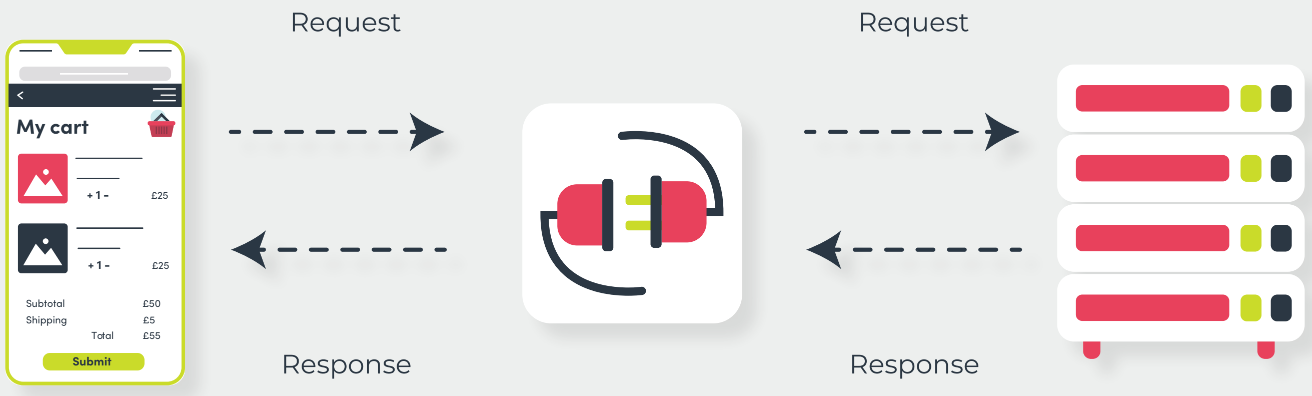
Building Integration First APIs to help with your Digital Transformation Strategy

Contents

What are APIs?

An API (application programming interface) is a software intermediary that allows applications to understand and communicate with each other.

For instance, they send requests and responses that allow you to retrieve information from a data source. Whether you work with them on a day-to-day basis or are just learning the ropes they are a vital part of automation and integration.



Why are APIs important?

APIs are an important aspect of the modern computing world, they have become the yarn that makes the digital world knit together as it were.

"The API ecosystem is global and growing. Postman users signed in from 234 different countries and made up 855 million API requests, up 56% from the previous year." **Postman, July 2021**

API data is easily and speedily exchanged which allows for modern innovation and daily conveniences. These conveniences include moving money between bank accounts, sharing video moments on social media, checking this week's weather or managing your at home appliances.

APIs provide businesses with the ability to adapt and develop products and services in the face of severe disruption. In the past two years more and more organisations have focused development efforts around APIs.

"49% of respondents said that more than half of the organisation's development effort is spent on APIs." **Postman, July 2021**

APIs are now a critical aspect of a business' digital transformation strategy and a vital element in consumers' experience.

APIs are connecting us and the world through data.

REST API

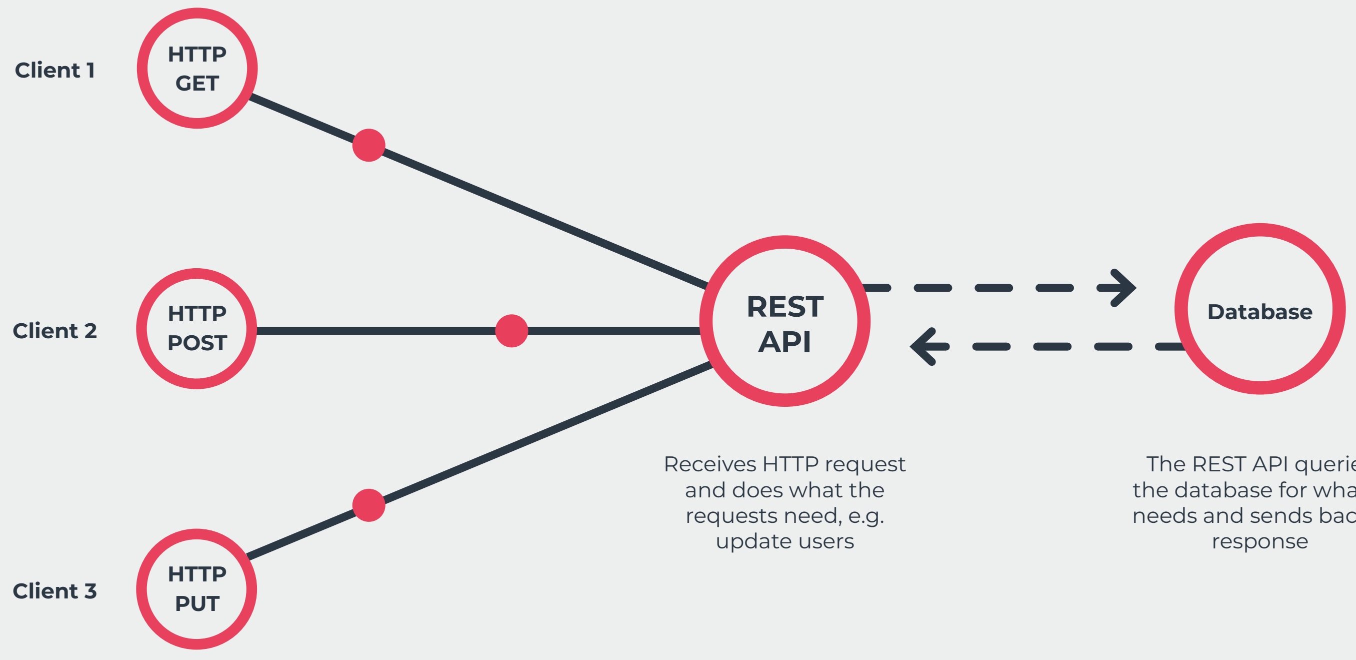
A REST or REpresentation State Transfer is a software architectural style that developers apply to web APIs. It can be used to communicate over the internet to make integrations simple and scalable.

REST's architectural style permits a requesting system to get robust access and predefine representations of web-based resources. It does so by deploying a predefined set of stateless protocols and standard operations.

Key Terms	
Client	The person using the API and making requests in order to retrieve data or change an element within an application.
Resource	Any piece of information or data that the API provides to the client.
Server	Used by the app which received the client request and contains the resource.

HTTP Request	
GET	Retrieves a resource
POST	Creates a new resource
PUT	Updates an existing resource
PATCH	Updates part of an existing resource
DELETE	Removes a resource

For example, a REST API works in a similar way to requesting or searching for something on Google. In response you get a list of results back from the service to which you submitted the request.


Client 1
HTTP GET
Client 2
HTTP POST
Client 3
HTTP PUT
REST API
Database
Receives HTTP request and does what the requests need, e.g. update users
The REST API queries the database for what it needs and sends back a response

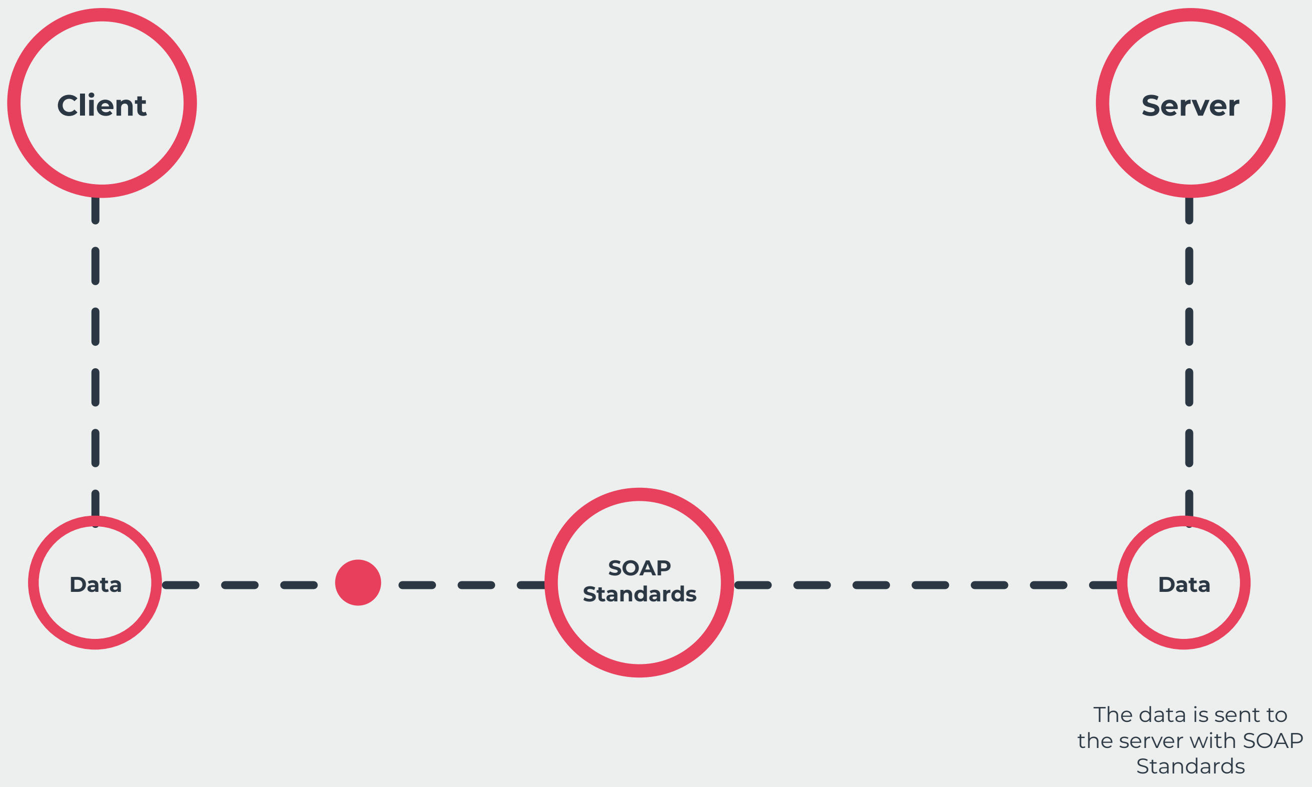
SOAP API

SOAP or Simple Object Access Protocol is a message specification for exchanging information between systems and applications.

This type of API enables processes that are using different operating systems via HTTP, and is designed to send messages to create, recover, update and delete records. Because of this a SOAP API has often been likened to a postal service, providing a reliable and trusted way to send and receive messages.

For instance, a SOAP API would be used within financial services, e-commerce payments and telecommunications. These tend to be within an enterprise environment, due to the use of formal contacts on a large scale and affect a big proportion of customers.

Due to it's secure nature connecting systems via a SOAP API can help identify customer consumption to generate billing information.



"Cyclr's knowledge of a vast number of public APIs that we use is of massive value to us as a company and they often go above and beyond to help with any questions that arise upon implementation of them." **Product Manager, SME**

GraphQL

GraphQL is a query language that lets developers construct requests that pull data from multiple data sources in a single API call.

A GraphQL API prioritises giving clients exactly what they request from a GraphQL server and no more.

GraphQL Primary Functions	
Query	To read the data
Mutation	To write the data
Subscription	To observe events and automatically send data

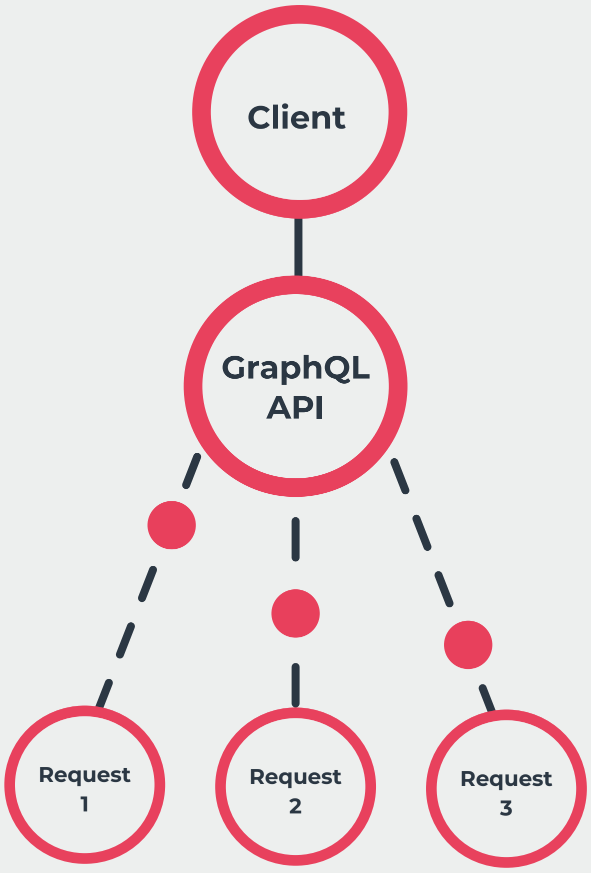
Several leading organisations use GraphQL such as Facebook, Pinterest, GitHub and PayPal as it is ideal for use with apps for devices such as, mobile phones or IoT devices, where speed and bandwidth is essential.

GraphQL is also used where data is nested and needs to be fetched in a single call as our diagram example shows. A real life example would be a social networking platform. They would use a GraphQL methodology where posts are required to be fetched, as well as comments, likes and shares.

An example from **Postman** demonstrates what GraphQL is capable of doing in comparison to REST APIs.

"If you wanted to use the Spotify REST API to get a list of all of the titles of all of the tracks of all of the albums by a particular artist, you would need to make three queries. You would need one query to find the artist ID, another to find all of the albums, and another to find all of the tracks on each album.

If you were using the Spotify GraphQL API, you could access all of that data with a single query."

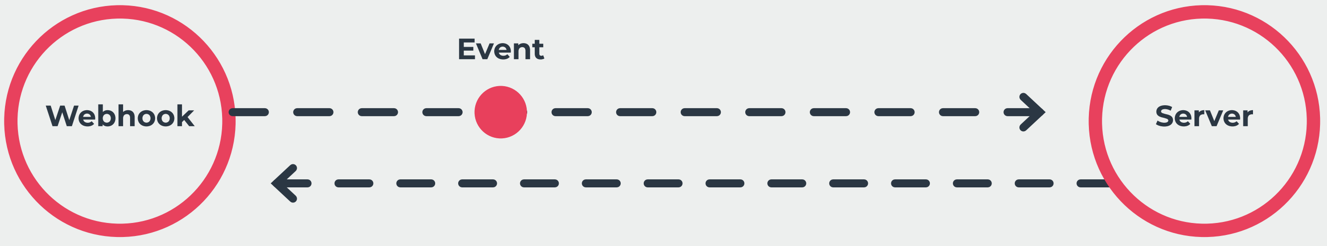


Webhook

A webhook has an event-driven architecture and allows you to send real time data from one application to another when a given event occurs.

For instance, a workflow can be triggered whenever a specific event happens in your internal system. This starts the integration process across as many platforms and processes as you may need.

The method of sending data is an event-driven push action, whereas typical APIs will request data via a GET request on a user's request or timed interval.



A webhook is triggered by an action or event

Server sends back a response to the event

For example; a webhook can be triggered when a user has initiated a chat or filled out a form on your website. The submitted data can be added and tracked in a CRM or marketing system using the webhooks packet data.

Implementing a webhook simply requires setting up a single POST request on the sending end, which sends a packet of data, a form fill for example, to a webhook URL. The webhook receiver can then process the packet of data in a range of ways.

APIs are the backbone of the web, and mobile applications.

A user friendly and powerful iPaaS giving you the ability to create and consume custom connectors. As well as a wide breadth of transport protocols supported, REST, SOAP, GraphQL amongst others." Technical Consultant, SME

API Design

	+	−
REST	• Easy to integrate with due to small learning curve • Use of HTTP so there are no expensive tools required to interact with the web service • Scalable, efficient and fast due to smaller message formats and no extensive processing	• Lack of standardisation • May require multiple calls to fetch larger sets of data • Inability to maintain state, it lies on the client which can make the application heavy • Requires additional security measures to be applied for secure usage
SOAP	• A dependable way to define and operate APIs across an enterprise at scale, and is the backbone applications and integrations upon which said enterprise depends • Standardised • Pre-built extensibility • Built-in error handling	• Slower to evolve and iterate on APIs • Longer to onboard new developers who aren't familiar with SOAP's older methods • Uses XML for all messages • Larger messaging size which impacts bandwidth
GraphQL	• Efficient so there is no under or over fetching • Ability to retrieve many resources in a single request saving time and bandwidth • Allows for precise querying when working with large APIs that return lots of data • Good for complex systems • Fast process	• Large number of complex relationships between data points can make managing large schemas a challenge • Difficult to specify the amount of requests in one day for rate-limiting • Complicated to implement simplified caching
Webhook	• Event-driven architecture is efficient • Events are posted in real time • Applications are automatically updated • Widely preferred by developers	• Initially difficult to set up • Less control over the data flow & have to accept the total amount of data given when an event occurs • No responses mean that it can't verify that a data packet was received by the targeted service • Works best with an 'Events' API endpoint to automate webhook management

API Security

In the digital world registering for an account on a website requires sharing some personal information, such as a username and password. They become a user's identifying credentials.

Then when the user is logging in with a username and password it is just one example of a technical process called **authentication.**

API keys are an authentication technique made up of unique strings which allow an API to identify which user is making a request.

API Key Locations	
Header	Done in lieu of a username and password
URL	e.g. http://example.com?api_key=my_secret_api_key
Request Body	Bury the key somewhere in the request body next to the data

Authorisation Characters	
The User	A person who wants to connect two websites they use
The Client	The website that will be granted access to the user's data
The Server	The website that has the user's data

Wherever the key is added they all have the same effect of letting the server authenticate the client.

OAuth 2

OAuth 2 authentication comes in several different specifications. One warning would be that many APIs follow these specifications loosely to fit their use cases, which then requires additional steps to authenticate.

OAuth 2 simply requires the user to submit information to an authentication server. If the information is correct the server will return an **Access Token** which will authorise the user as they access their intended server.

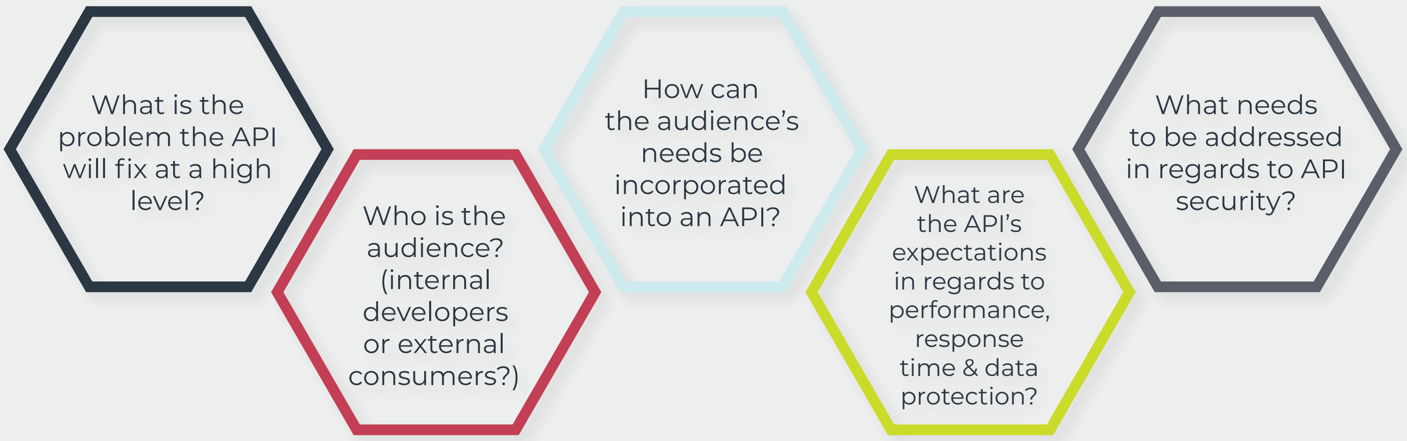
Three implementation examples include:

OAuth 2 Types	
Authorisation Code	To authorise, the user is redirected to an authentication page on the authentication server. The user will be required to login and approve the connection to receive their access token.
Client Credentials	Similar to Authorisation Code but does not require the user to login and approve the connection.
Password Credentials	Client will receive the access token from the authentication server by using their username and password.

API Preparation

When researching how to develop an API, many resources start with the planning and preparation of the API.

Initially, writing down questions you should be asking, such as what do you want to achieve with the API and what business capabilities the API exposes to consumers? These questions can help determine your API's requirements.



What is the problem the API will fix at a high level?

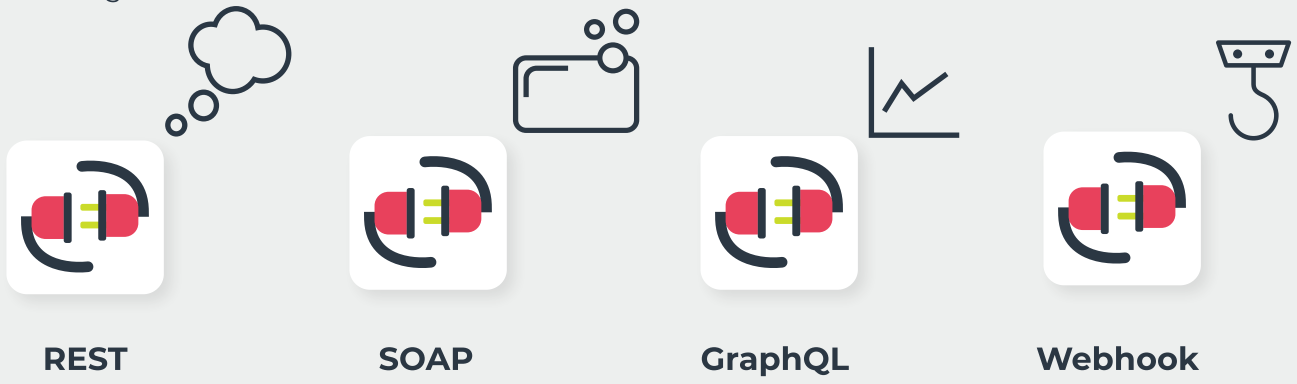
Who is the audience? (internal developers or external consumers?)

How can the audience's needs be incorporated into an API?

What are the API's expectations in regards to performance, response time & data protection?

What needs to be addressed in regards to API security?

Next you'll need to choose an API architectural style (mentioned on pages 4 & 5), which could be any of the following:



Then the design phase of the API to determine how it will look, as well as processes to make the API easy to understand for users.

Therefore, creating a useful API name and description is beneficial to users.

"Users of public APIs are usually developers, so it is important to build APIs that are simple and intuitive for them." AWS

API Development

It's time to actually create the API. You'll find there are plenty of API development tools available to help build your APIs such as outsystems, and PortSwigger.

Firstly, fill out your API's user-friendly name and description, describing the purpose of the API. Then implement important security policies as the API will be dealing with consumer data.

Secondly specify the data models that describe the API request and response messages.

Then create endpoints based on the user's needs GET, POST, PUT, DELETE.

GET	Return a list of all users in the database
POST	Create a new user in the database
PUT	Update an existing user in the database
DELETE	Delete an existing user whose unique username is supplied

These are important requirements that need to be met for integration, as well as including HTTP methods, that correspond to read, create, update, and delete operations respectively. They are especially key if you want to scale the API.

As well as including the correct data formats (either JSON, XML or Form), these are essential for building APIs for integration. As this is the type of data used by the API.

- For GET, this is the accept header.
- For POST, this is the request data.

"When we began using Cyclr there were definitely some API connectors we needed that were missing, but the team are super-responsive and built or extended existing API connectors to meet our needs." **Co-Founder, SME**

Building APIs for Integration

The thing with an API is that how you build it should have a lot to do with how other applications are going to interface with it. The feature sets that you build-out should be heavily influenced by making your API **'play nicely'** with the applications with which it is going to be connected. To meet user requirements and, moreover, to be integration friendly.

Updated Time

By making an "Updated Time" available on your records you can help lighten your API's request load, making it possible for users to retrieve only updated records from the server instead of pulling large record sets to process on their side.

Webhook Provisioning

You can dramatically improve your user experience by including the ability to set and delete webhooks using your API. This prevents the user from having to copy and paste webhook URLs in management platforms, automating it via a few API calls to an Event's endpoint.

Bulk Data Endpoints

When dealing with large arrays of data, adding endpoints for managing bulk data via GETs or POSTs will reduce the number of API calls to your servers. You can also look into Streaming APIs for large data requests.

Data Retrieval Endpoints

When retrieving data, it can be useful to offer metadata endpoints that allow the API Request to decide the "width" of the data that is returned. This could be by letting the Request state the fields to be returned for each object. This is great for endpoints working with custom fields.

Interactive Documentation

As mentioned above, OpenAPI provides a great standardisation for API documentation but you can also use this to provide interactive API docs. Developers can use these to test endpoints with their authentication credentials, aiding them through development.

Strong Additional Data

If your system is working extensively with external systems, having available fields for storing key data (CRM record ID for example) will reduce the number of API calls you need to access and modify records as you don't need to perform an initial record search.

API Design

Rate Limiting

A way to manage and protect backend access through an API. It puts a limit on the number of API calls a user can make within a time period.

Paging

Breaks down requested data into multiple API responses giving a fail-safe method to ensure the API is controlled and allows for receiving and sending records in batches.

Authentication

Authentication is a must for any API being used to transfer private data and is necessary for any multi-tenant system. A popular authentication type often chosen is OAuth. As well as set-up error messages/responses for common errors such as 'Not Found', 'Unauthorised', etc.

Documentation

Documenting your API helps with versioning, making your API readable and making amendments and additions. Self-generating systems such as OpenAPI make API documentation automated and can be added to Swagger tools to provide interactive API documentation.

Multi-Tenant

A public cloud, which is a single instance of software that runs on a server that is accessible to multiple users, is beneficial for centralised management.

Structured Endpoints

Offering endpoints is useful when retrieving data, allowing the API request to decide the "width" of the data. Providing an "add or update" endpoint helps perform the appropriate action.

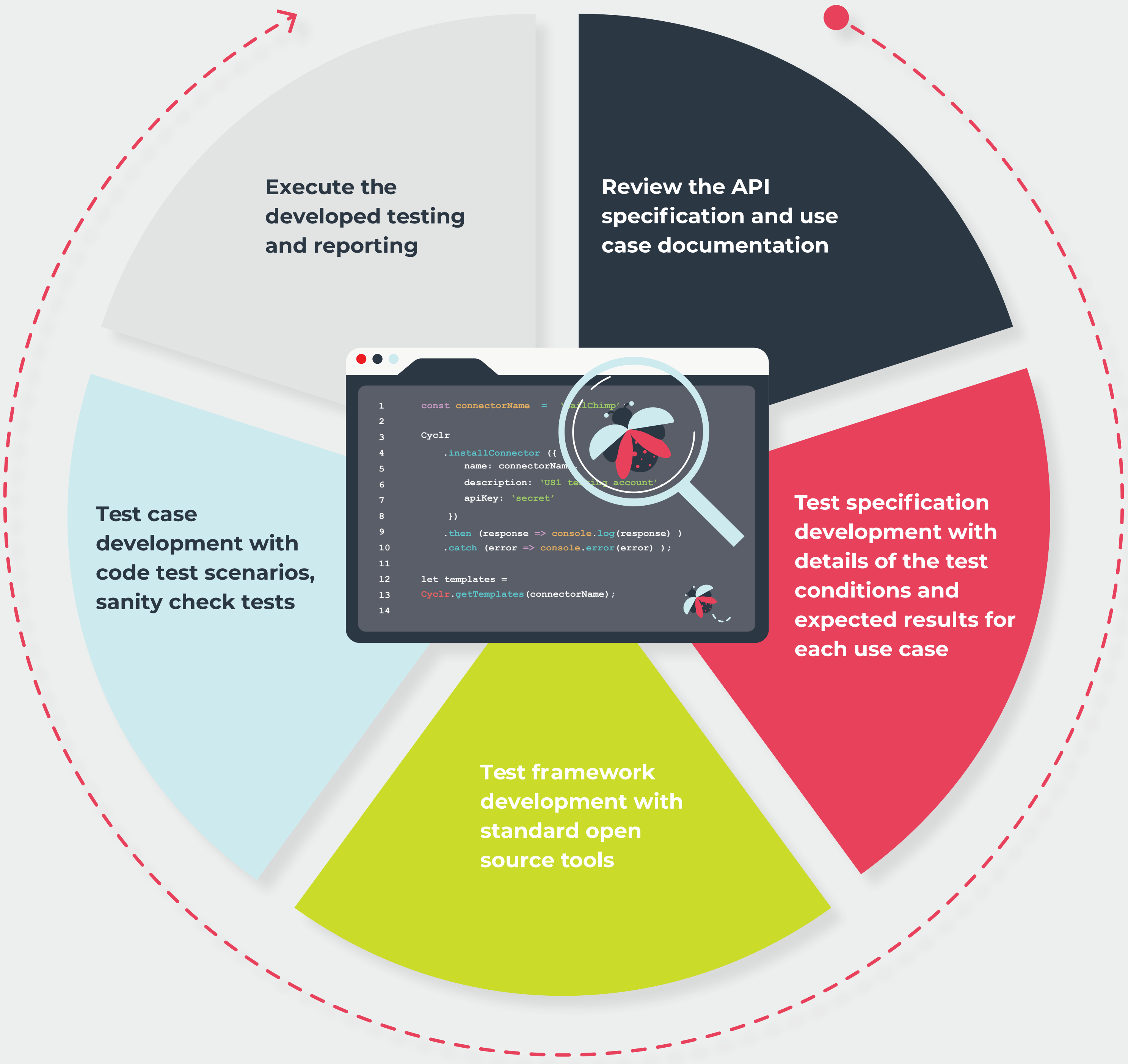
Early adoption of an iPaaS platform can therefore help you build your API for purpose, contextual to user requirements and the requirements of other platforms.

"Super quick API builds, just ask them and the API connector appears in days, sometimes in just hours." **Administrator, SME**

Testing your API

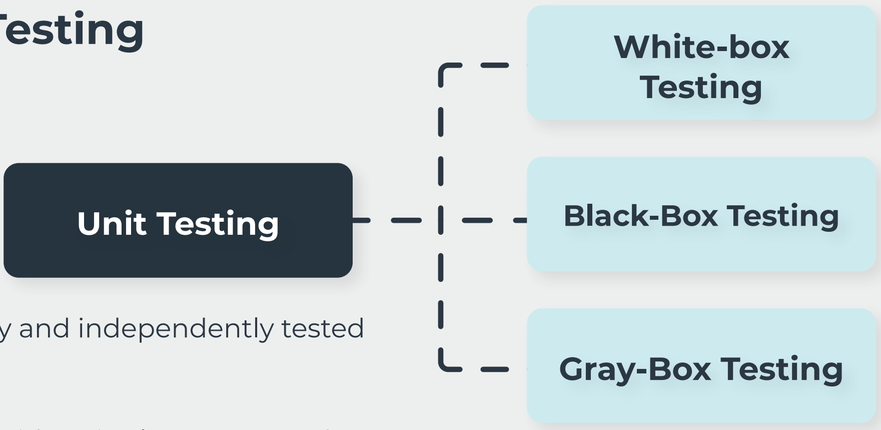
It is important to test the functionality of your API under different conditions in a testing environment.

"API testing consists of making requests to single or sometimes multiple API endpoints and validating the response for performance, security, functional correctness or status check". Codersera, December 2019


Review the API specification and use case documentation
Test specification development with details of the test conditions and expected results for each use case
Test framework development with standard open source tools
Test case development with code test scenarios, sanity check tests
Execute the developed testing and reporting

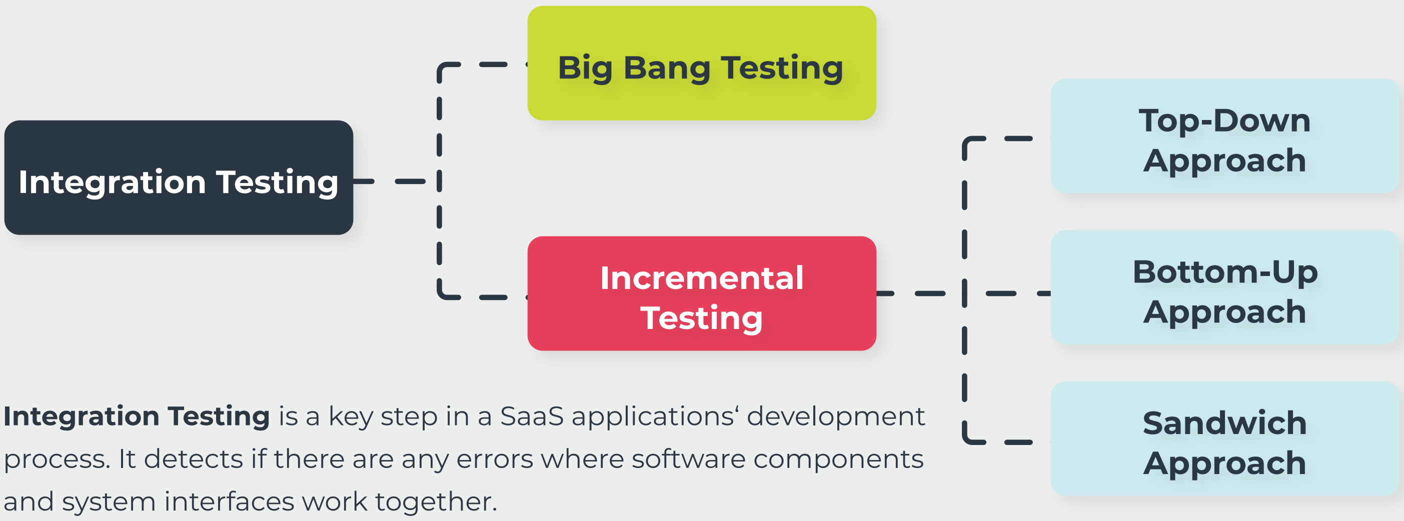
```
const connectorName  =  'MailChimp'

Cyclr
     .installConnector ({
         name: connectorName,
         description: 'US1 testing account',
         apiKey: 'secret'
      })
     .then (response => console.log(response) )
     .catch (error => console.error(error) );

let templates =
Cyclr.getTemplates(connectorName);

```

Different Types of API Testing



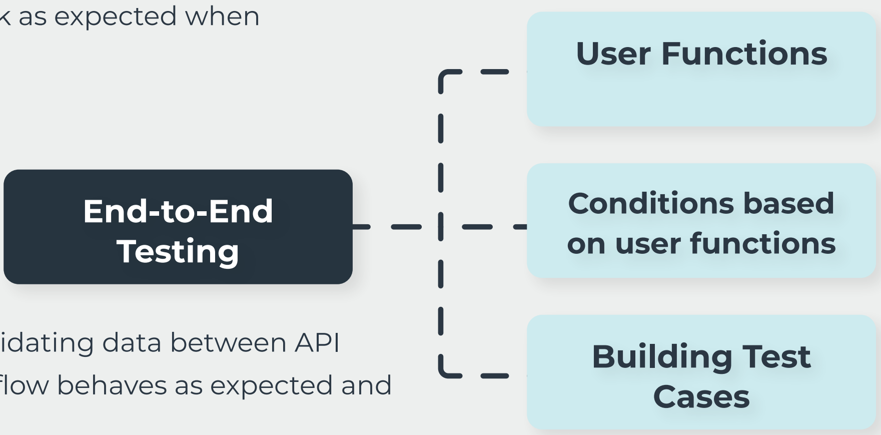
Unit Testing is a software development process in which the smallest testable parts of an application, called units, are individually and independently tested for operation.

For instance testing a single endpoint with a single request and response.



Integration Testing is a key step in a SaaS applications' development process. It detects if there are any errors where software components and system interfaces work together.

In other words testing two or more separate functions, or component groups to ensure they work as expected when integrated.



End-to-End Testing is a process that tests the entire software product from beginning to end.

The testing is designed to help with validating data between API connections. Ensuring the application flow behaves as expected and all integrated pieces work together.

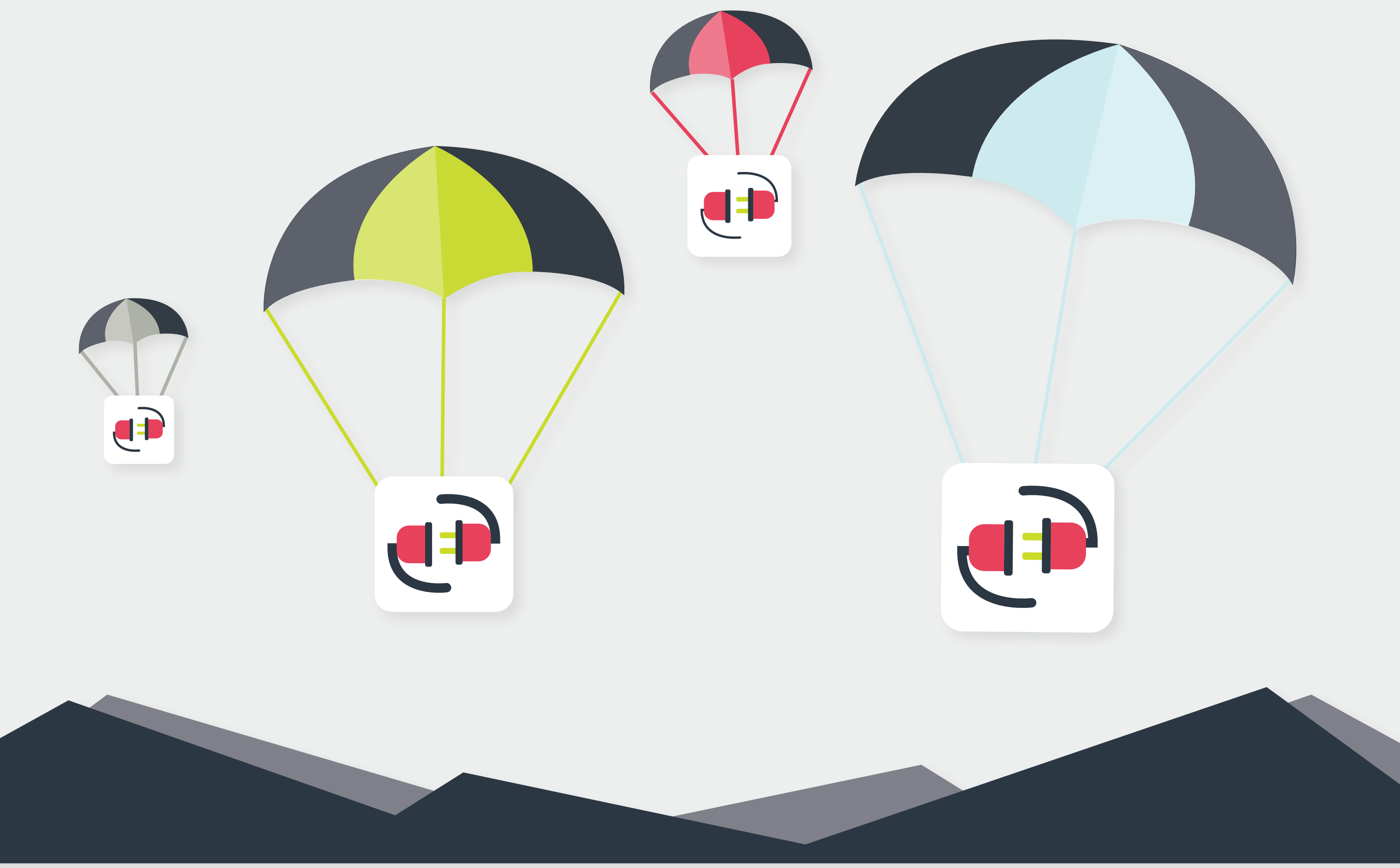
API Testing is beneficial as you can gain early evaluation, and feedback on the overall API's core functions.

There are various API testing tools. Popular options include Postman and JMeter, both of which can be used for functional API testing, an exploratory-type of API testing.

API Delivery

When you are ready to deploy your API, it is important to share the API description and update the URL. Then share or embed the API documentation with both internal users and external clients.

Finally, publish the API to a developer portal to encourage adoption.



Monitoring your API

Keeping tabs on your API is useful to understand how it is being used, developed and improved. As well as identify any issues arise that need to be addressed and iterated upon.

"It is great that you have the ability to use your own API as a connector enabling a single point of integration with multiple sources."

Co-founder & Platform Architect, SME

API Must Haves from Cyclr Developers

What do our developers say about developing APIs? Here are some of their API building best practices.

API Checklist

- [✓] Always include the real request/response information in the API documentation
- [✓] Make sure the credentials or access to a test or live system are available
- [✓] Be sure to keep documentation up to date and accessible as well as a live service, like Sandbox
- [✓] Create a consistent structure of endpoints and data structure
- [✓] Provide clear, concise and interactive documentation with relevant walkthroughs and tutorials
- [✓] Include a standard submission of defined data
- [✓] If you are building an API for internal processes take a step back to see how it can evolve, how would an outsider view this service.
- [✓] Be sure to communicate best practices of interacting with your API, such as rate limits and paging
- [✓] Ensure that the API can filter or mine records
- [✓] Run tests on your API before launching
- [✓] Build an API that is easy to read, use and is complete

Getting ahead in the API Economy

The API economy refers to business models and practices built around the use of APIs in the modern digital economy.

In other words, an API economy presents services and data through APIs to generate value for an organisation, in a controlled way.

It starts with developing a strategy and adopting an API-first approach, as well as making your API accessible. Making APIs accessible allows organisations to give partners access to core business capabilities.

This can result in a quicker time to market and a reduction in costs.

Hi Jerry, I'm Clyde the Chatbot! How can I help you today? Choose an option below.
Info about products I'm not yet using
Help with products I'm already using
Something else
Username
Password
Login
3:30:30
331
34
05
Payment Methods
Pay
API
Check In
Tuesday
15°

Adopting an API-first approach means making APIs that are consistent and reusable. Then establishing a contract for how the API is supposed to behave. This has a direct impact on the way the API is designed, as APIs are increasingly designed to be consumer-centric.

Tailoring their design to customers use cases ensures good customer experience. Then using that documentation and publication to later onboard similar customers faster and generate more revenue.

APIs are helping to shift from on-premises to cloud and microservice-based applications. Creating API microservices is a way of monetising your APIs. It separates portions of an application into smaller, self-contained services.

Making things easier for developers to manage, as well as giving businesses greater agility in maintaining, upgrading and scaling their products and services.

"APIs enable companies to more easily build products and services that would otherwise take too long to build," Kong co-founder and CTO Marco Palladino, VentureBeat, May 2021

The increased API usage has propelled organisations to not only continue their digital transformation efforts but to accelerate them.

"APIs are central to enterprises' needs and digital transformation efforts." The State of API Economy 2021 Report, Google

APIs and digital transformation therefore go hand in hand.

"We have the ability to include our own API as a connector that unlocks integration capability with a number of third-party sources."

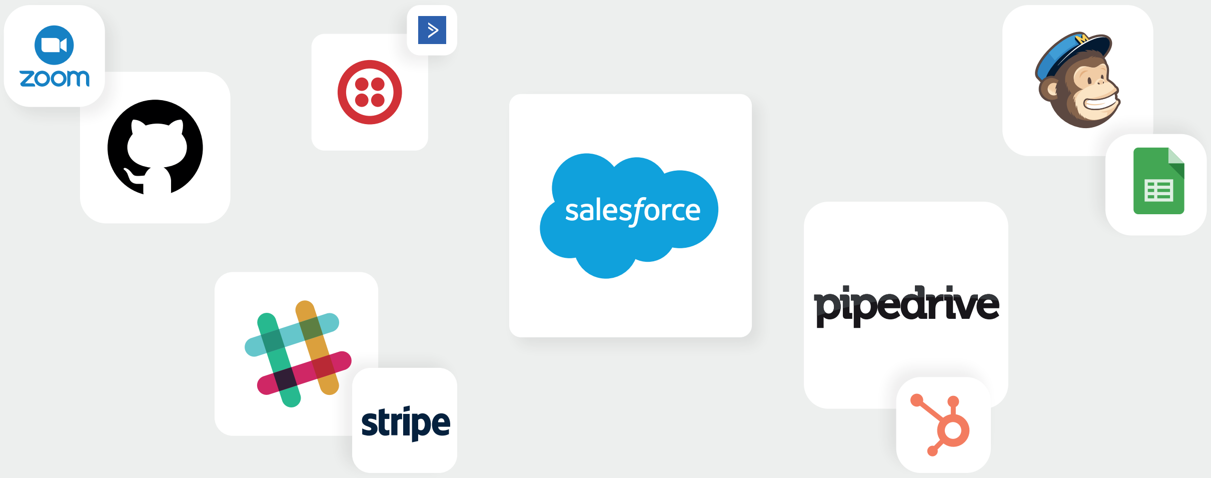
Principal Architect & CTO, SME

How can using Cyclr help with your API economy?

Cyclr is a low-code platform where you can **build integrations** between third party applications and your platform.

Once a Connector is created, Cyclr transforms this humble file into a fully working, drag and drop API.

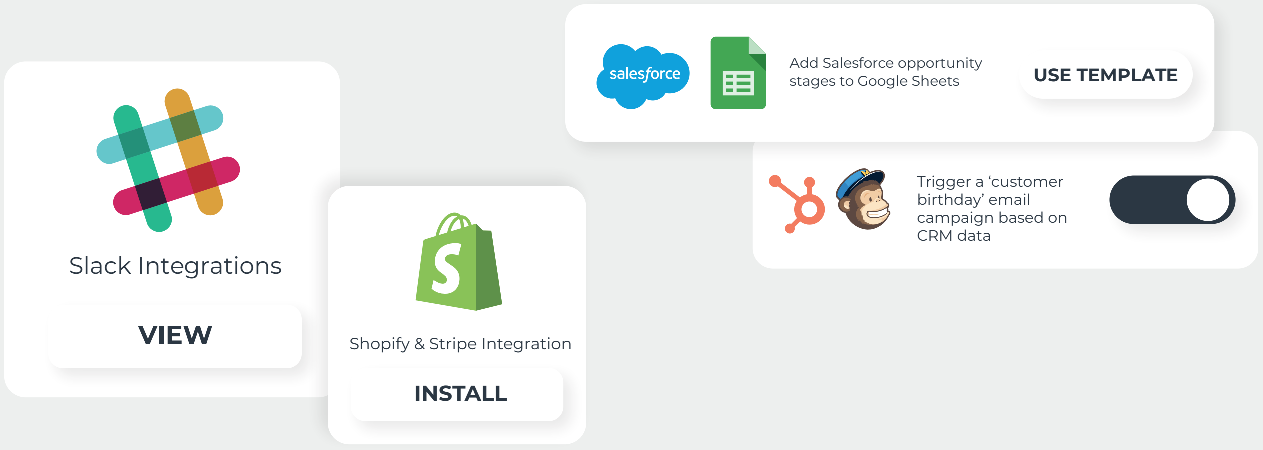
Cyclr then automatically handles all of the usual work with APIs such as authentication, paging, errors, etc.



As well as building **API connectors** Cyclr has an extensive array of connectors in its **connector library**. Thus giving you the easiest way of connecting your SaaS to a growing ecosystem and build integrations between platforms and data sources.

The platform can help you get ahead in the API economy by creating new revenue opportunities from your API.

Using Cyclr, your integrations can be a value-add feature, which can push users up through your pricing plans.


salesforce
Add Salesforce opportunity stages to Google Sheets
USE TEMPLATE
Trigger a 'customer birthday' email campaign based on CRM data
Slack Integrations
VIEW
Shopify & Stripe Integration
INSTALL

Thereby building reliable integration solutions for your customers and increasing your revenue.

Cyclr is an iPaas built from the ground up for embedding, simple to use, with flexible deployment.

An easy to use integration platform with low-code tools empowering anyone in your SaaS to respond to user integration requests without adding to your developer backlog.

A developer toolkit that can be used by commercial teams.

So, starting your integration journey is closer and easier than you think.

Momentum Leader

Users Most Likely to Recommend

Users Love Us

Easiest To Do Business With

High Performer

Capterra

crozdesk

cyclr.com