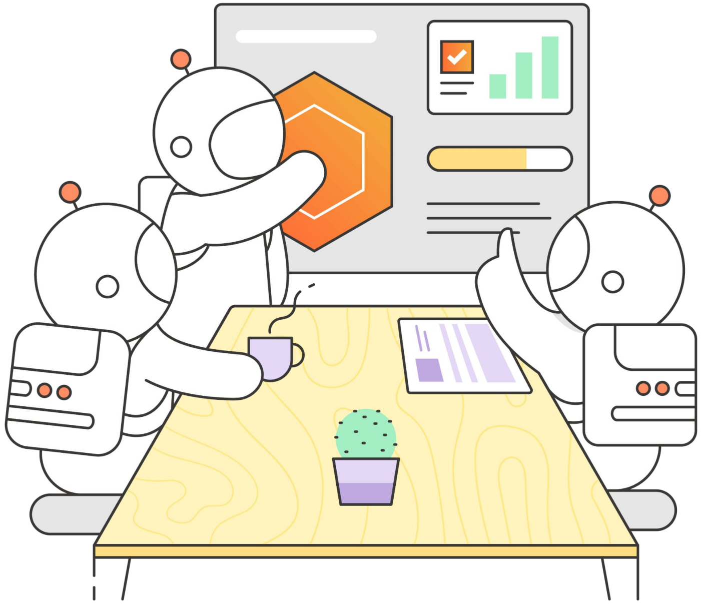
POSTMAN

Testing OpenAPI definitions for better and consistent APIS



Christos Gkoros

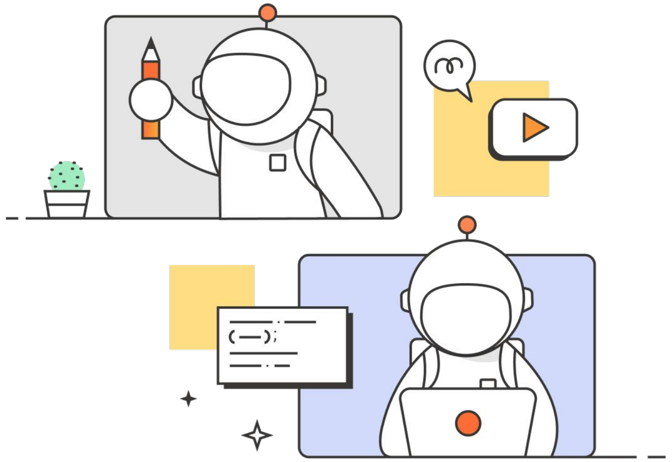
API Architect - Platform engineering

POSTMAN

Build APIs

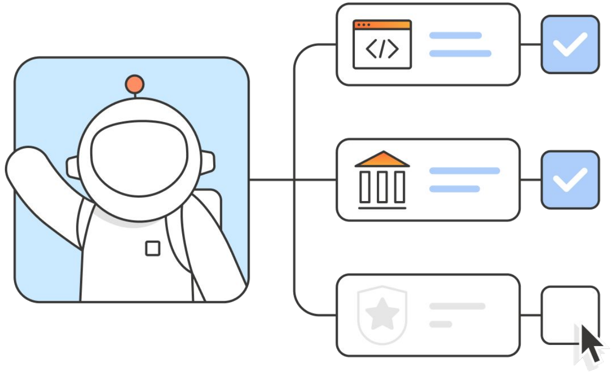
Use APIs

Collaborate

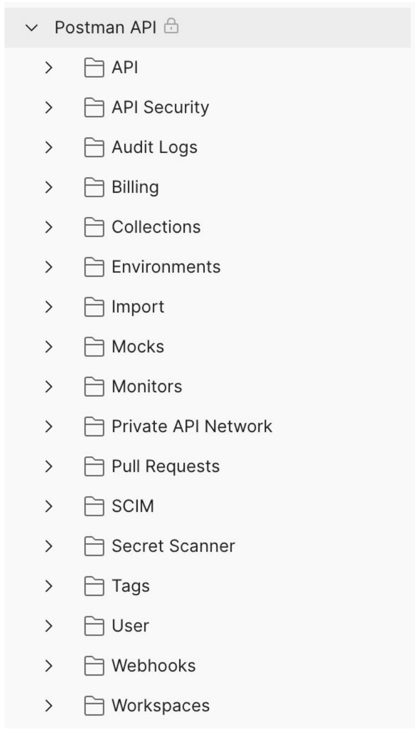


Inside Postman

1. API Design
2. Scale
3. Platform Engineering
4. OpenAPI
5. Spectral
6. Test examples

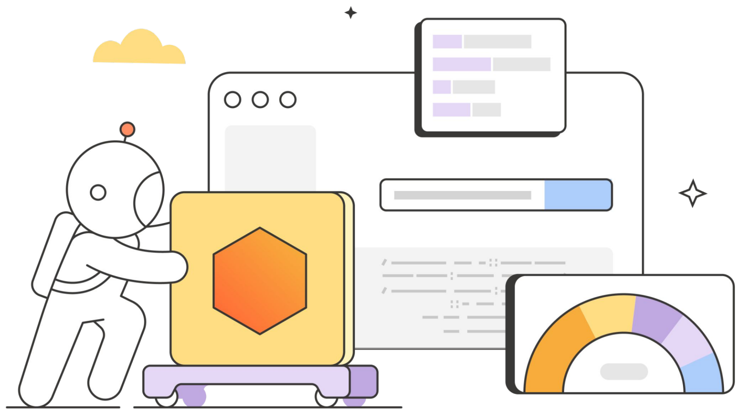


The Postman API



Postman API

- API
- API Security
- Audit Logs
- Billing
- Collections
- Environments
- Import
- Mocks
- Monitors
- Private API Network
- Pull Requests
- SCIM
- Secret Scanner
- Tags
- User
- Webhooks
- Workspaces



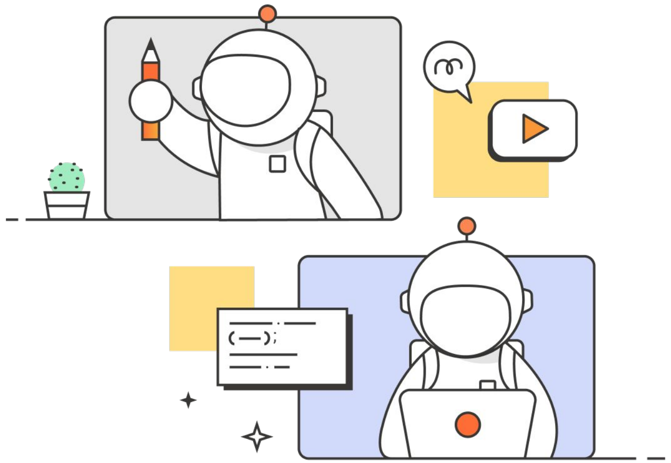
Why integrate with the Postman API

- **Enhance Postman's capabilities**

  Providing users with the tools and resources to optimize.

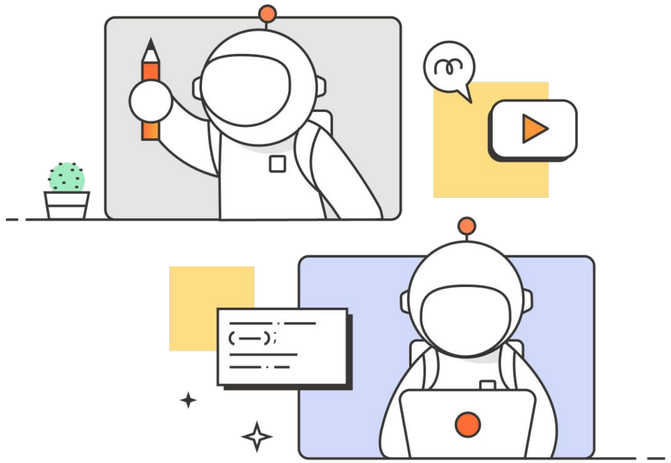
- **Embed Postman within other workflows**

  Like established processes enhances productivity and streamlines development.



As an API Architect I study

- **Why** our users need the API
- **What** functionalities they seek beyond the core product
- **How** are they trying to do that



Common Use cases

- **Automation**

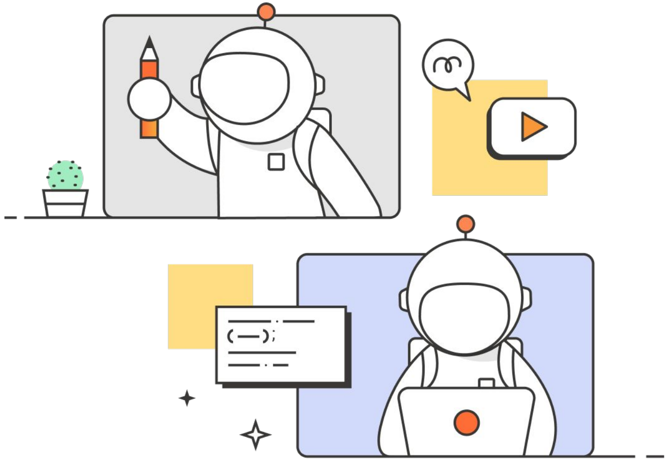
  Scaffolding of Postman resources

- **Auditing**

  Ensuring the Postman resources are as they should be

- **Tool integration**

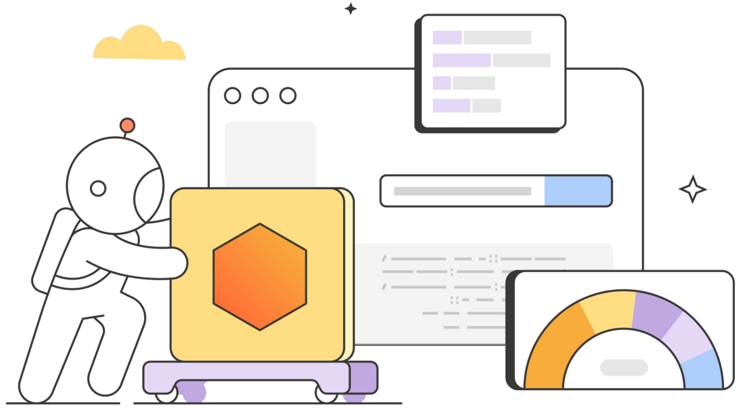
  Developer Portals
  CI/CD
  Test-planning, Test analysis, Test reporting ...



My goals for the Postman API

1. Simple and easy to use
2. Effective at its job
3. Increase its adoption and usage

API Design



Elements of a good design

- **Descriptive names**

  Names that are descriptive names and aligned with the API's goals

- **Rich Functionality**

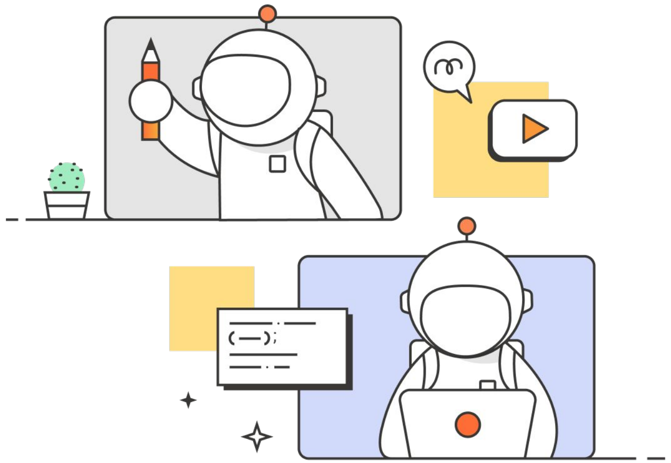
  User cans do easily what they need, like filtering with certain attributes.

- **Flexible formats**

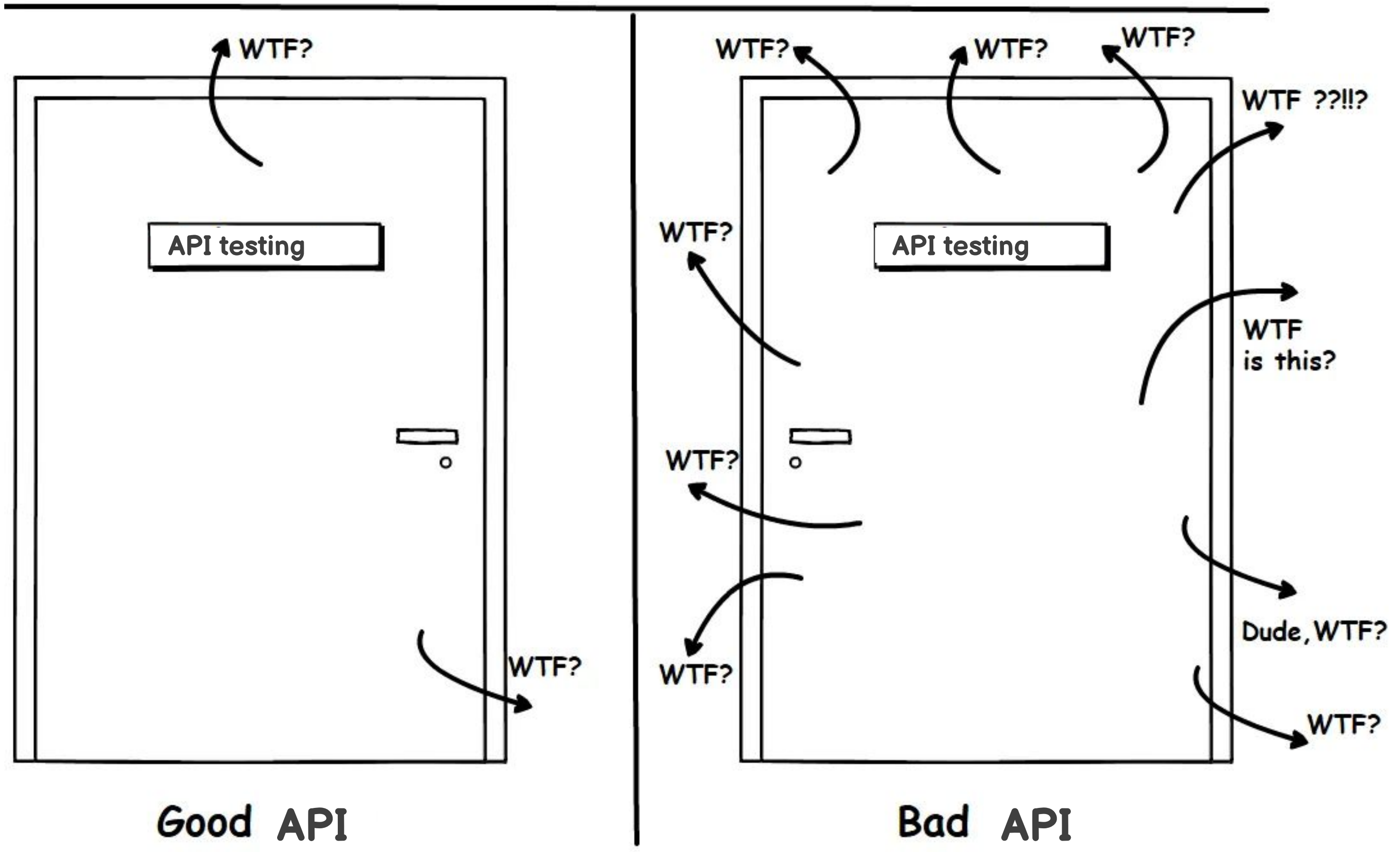
  The format is flexible and it adapts well on changes over time

- **Clear error messages**

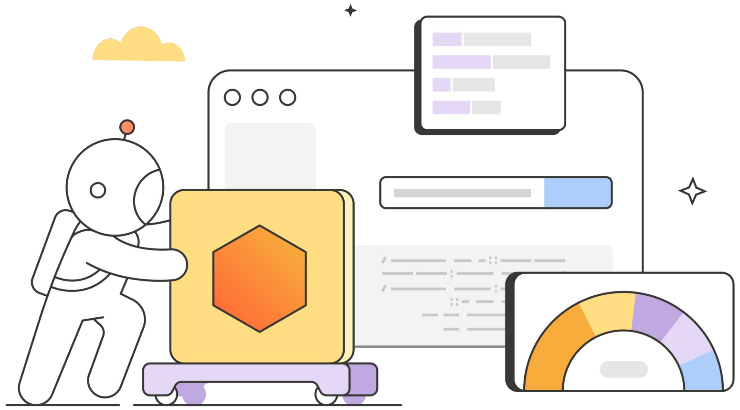
  Good experience especially during troubleshooting

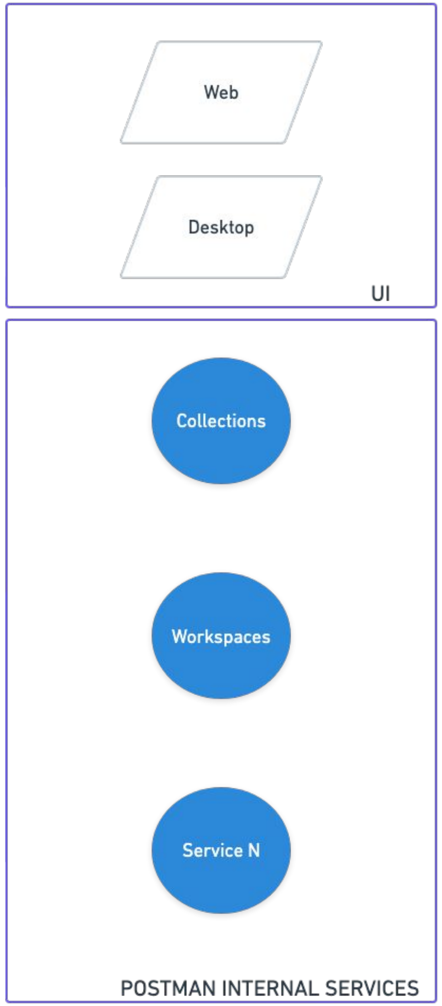


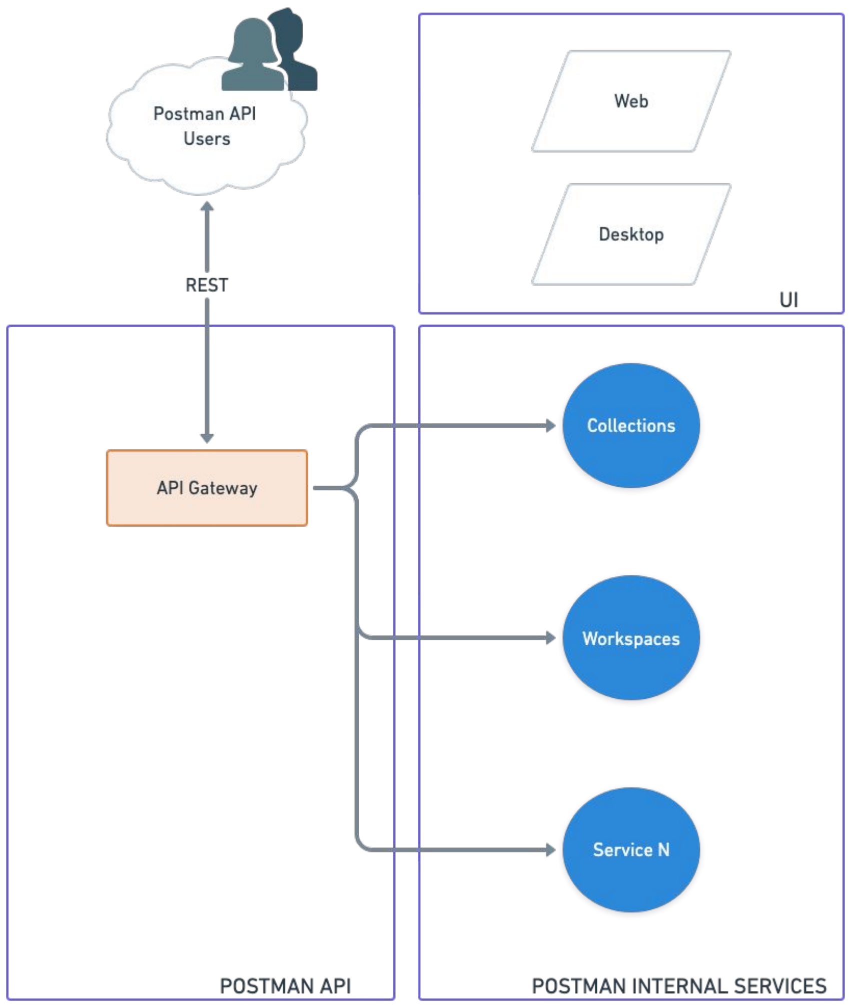
API quality measurement: WTFs per minute



Scale

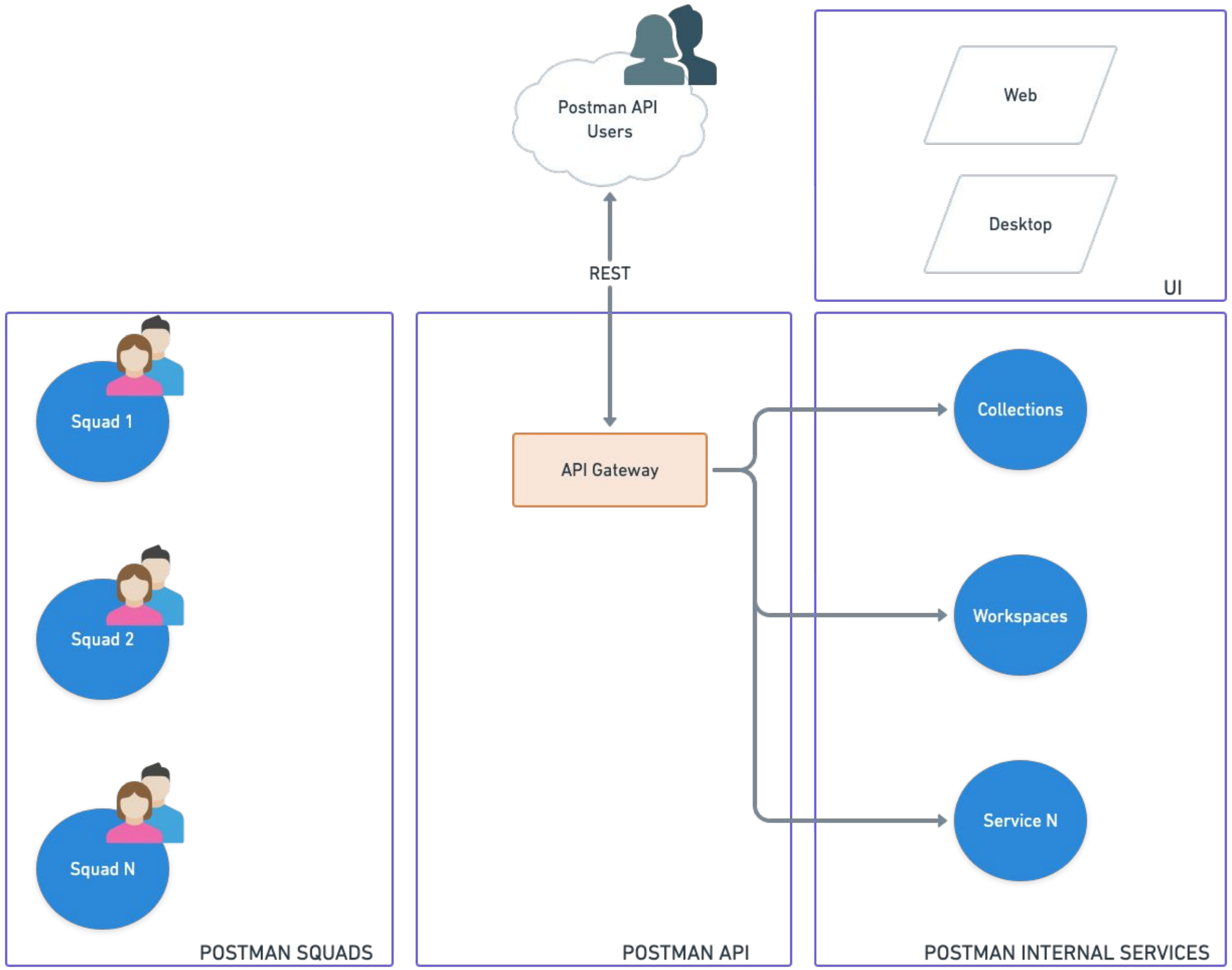



Web
Desktop
UI
Collections
Workspaces
Service N
POSTMAN INTERNAL SERVICES


Postman API Users
REST
Web
Desktop
UI
API Gateway
Collections
Workspaces
Service N
POSTMAN API
POSTMAN INTERNAL SERVICES

API Gateway brings common

1. Authentication
2. Security
3. Throttling
4. Routing
5. ....

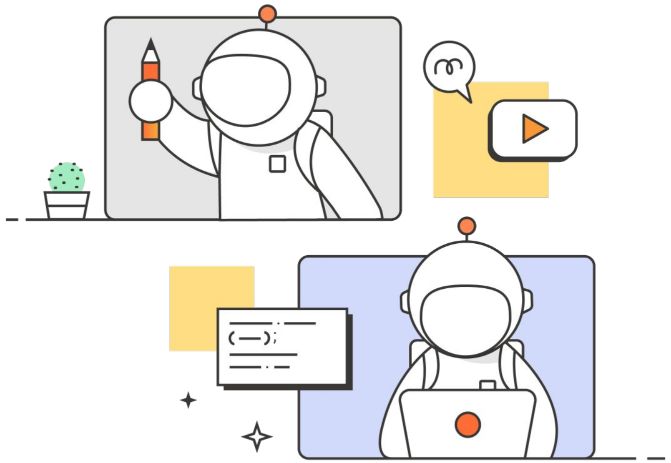

Postman API Users
Web
Desktop
UI
REST
Squad 1
Squad 2
Squad N
POSTMAN SQUADS
API Gateway
POSTMAN API
Collections
Workspaces
Service N
POSTMAN INTERNAL SERVICES

The Postman API needs to

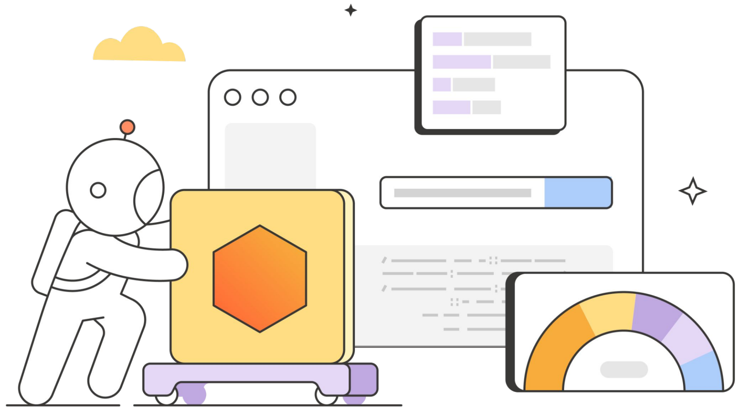
1. Have a common Look and Feel
2. Be a unified product rather a mix of different endpoints

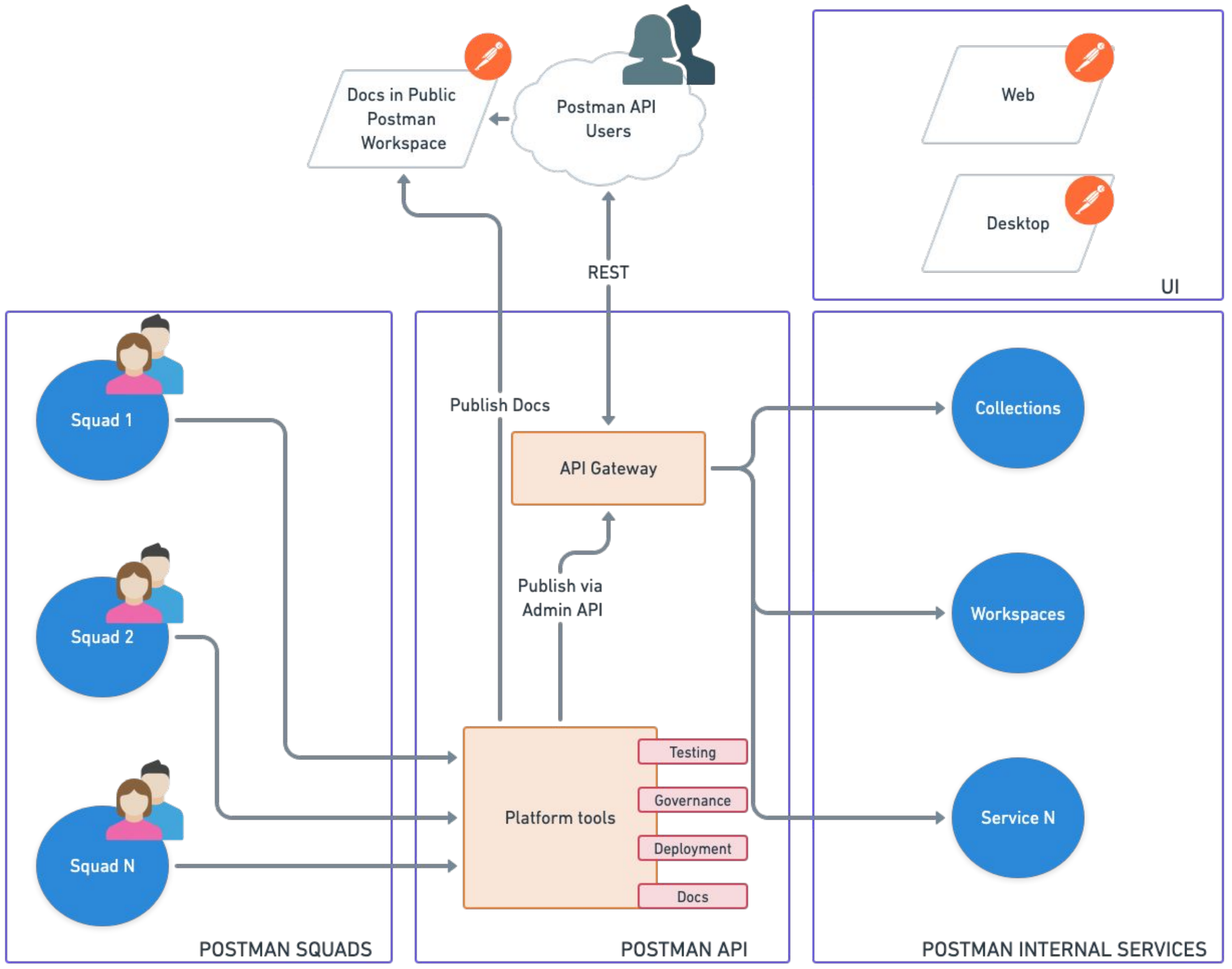
Recap - What do we need?

- Good API Design
- Consistency at scale
- Autonomy
- Fast delivery

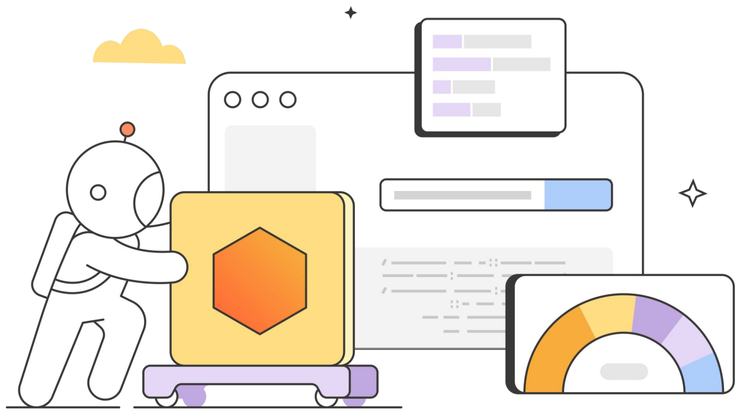


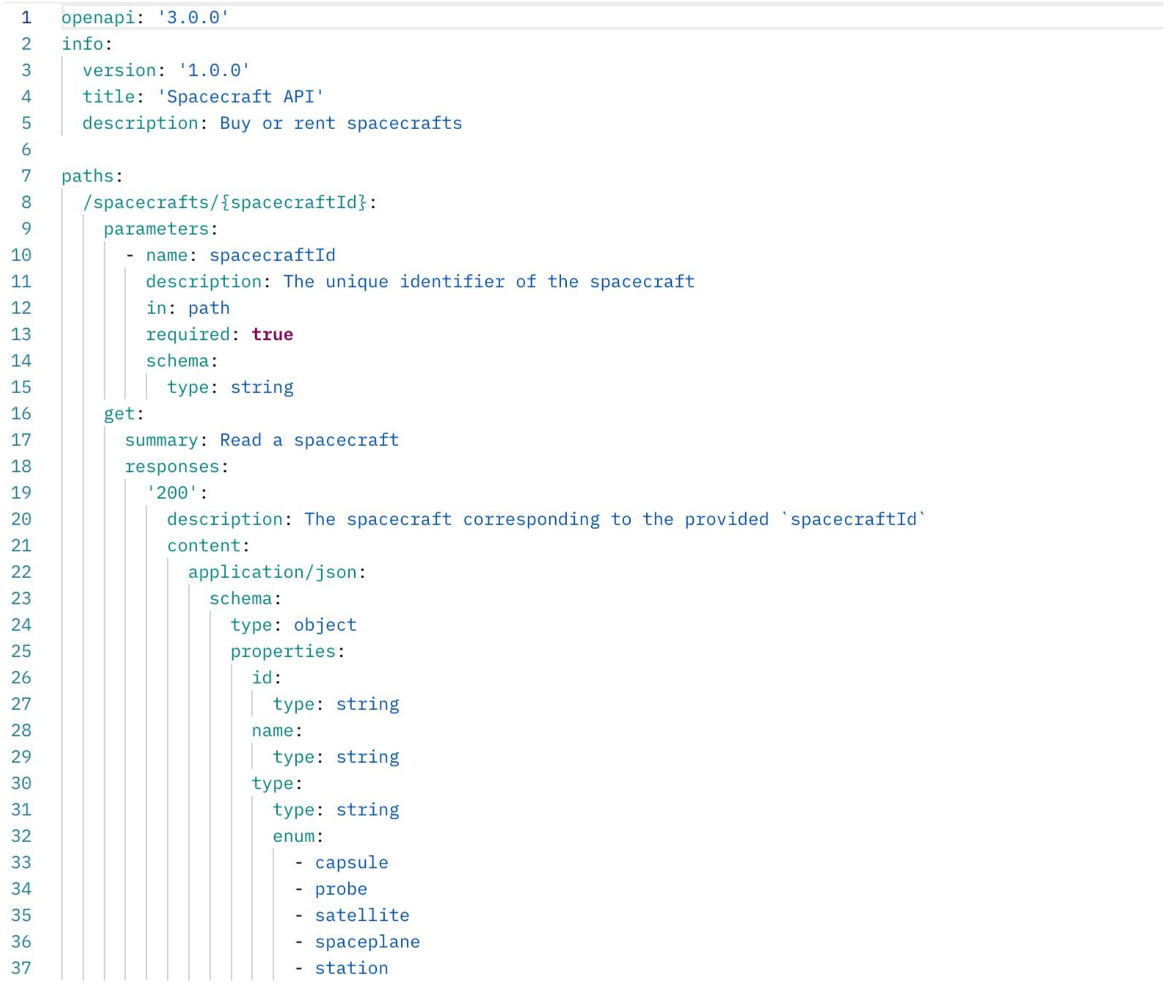
Platform Engineering




Docs in Public Postman Workspace
Postman API Users
Web
Desktop
UI
REST
Publish Docs
API Gateway
Publish via Admin API
Squad 1
Squad 2
Squad N
Platform tools
Testing
Governance
Deployment
Docs
Collections
Workspaces
Service N
POSTMAN SQUADS
POSTMAN API
POSTMAN INTERNAL SERVICES

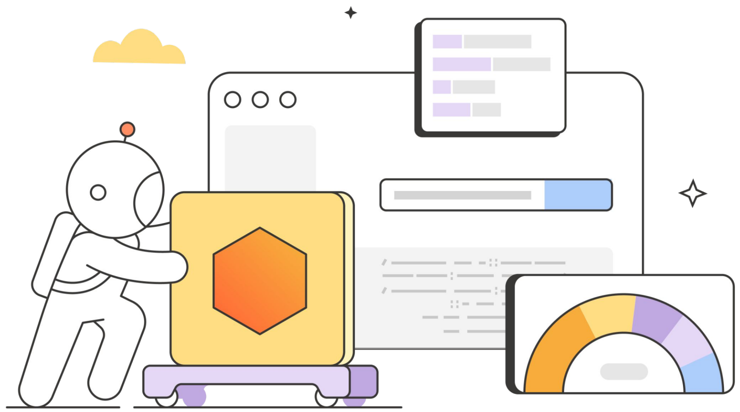
OPEN API
INITIATIVE

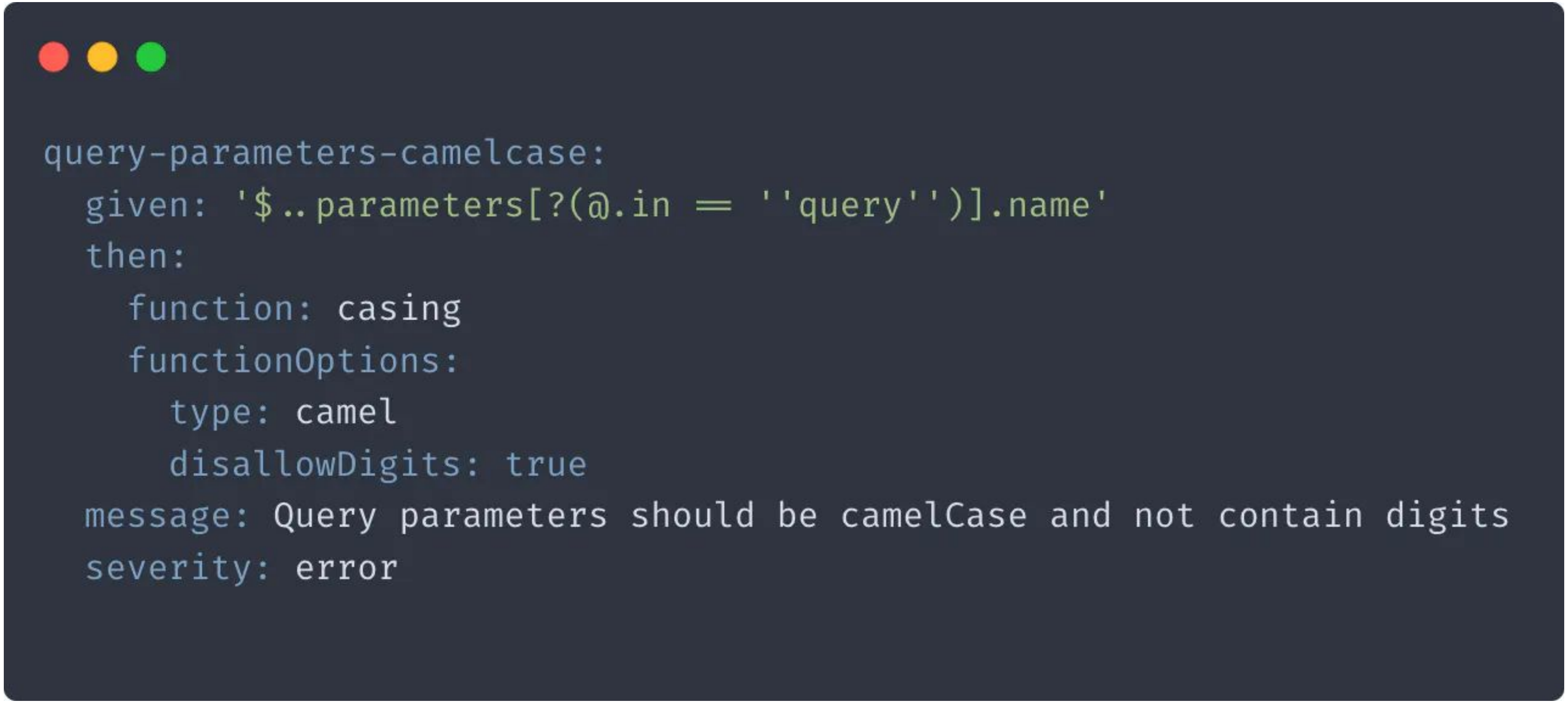




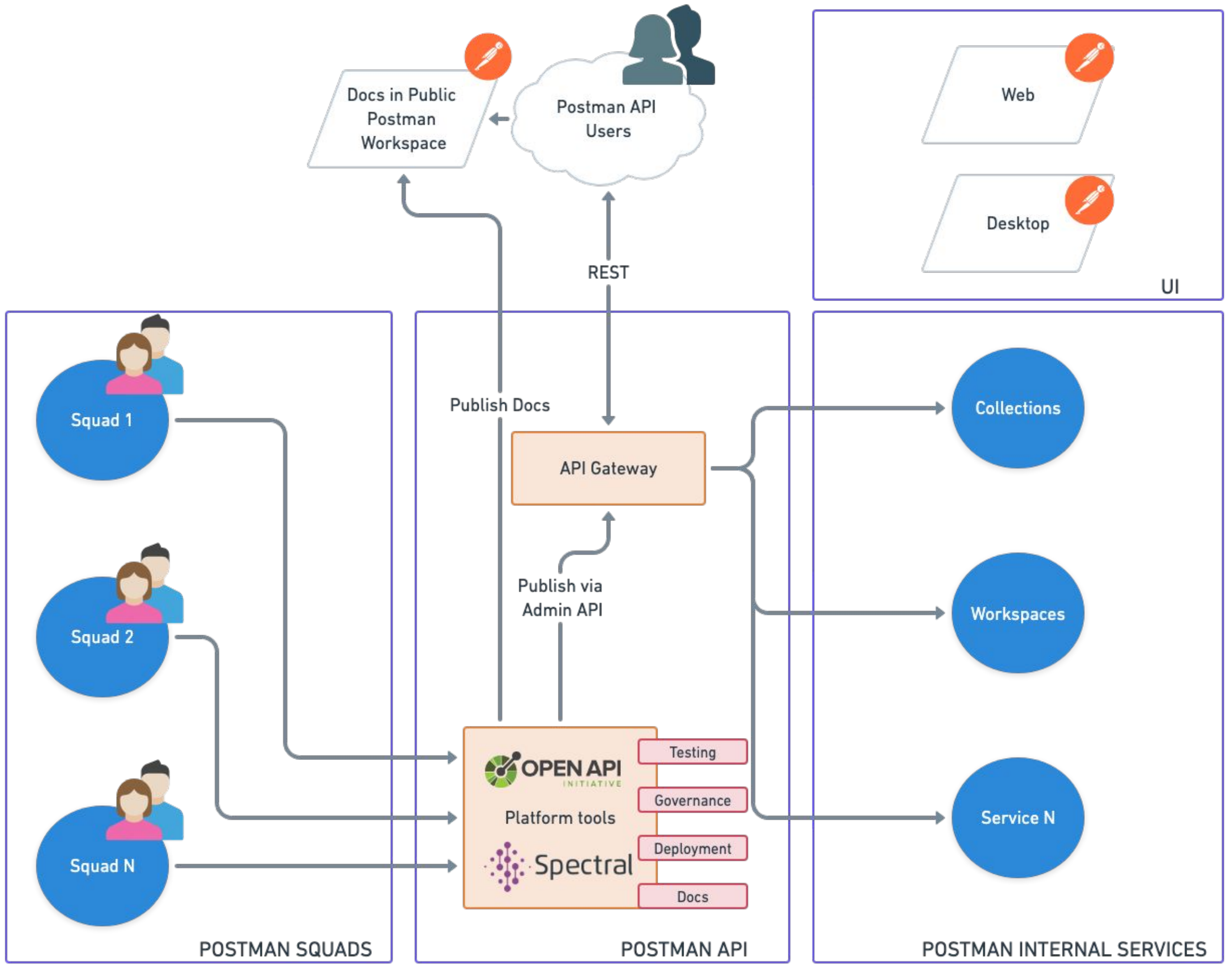
```yaml
openapi: '3.0.0'
info:
  version: '1.0.0'
  title: 'Spacecraft API'
  description: Buy or rent spacecrafts

paths:
  /spacecrafts/{spacecraftId}:
    parameters:
      - name: spacecraftId
        description: The unique identifier of the spacecraft
        in: path
        required: true
        schema:
          type: string
    get:
      summary: Read a spacecraft
      responses:
        '200':
          description: The spacecraft corresponding to the provided `spacecraftId`
          content:
            application/json:
              schema:
                type: object
                properties:
                  id:
                    type: string
                  name:
                    type: string
                  type:
                    type: string
                    enum:
                      - capsule
                      - probe
                      - satellite
                      - spaceplane
                      - station
```

Spectral

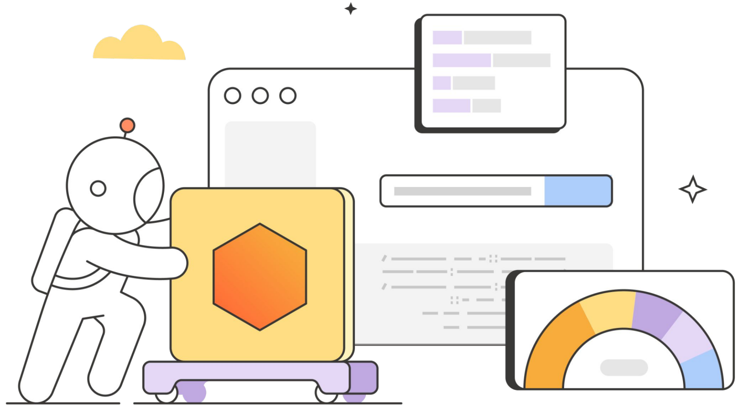




```yaml
query-parameters-camelcase:
  given: '$..parameters[?(@.in == ''query'')].name'
  then:
    function: casing
    functionOptions:
      type: camel
      disallowDigits: true
  message: Query parameters should be camelCase and not contain digits
  severity: error
```


Docs in Public Postman Workspace
Postman API Users
Web
Desktop
UI
REST
Publish Docs
Squad 1
Squad 2
Squad N
API Gateway
Publish via Admin API
Collections
Workspaces
Service N
OPEN API INITIATIVE
Platform tools
Spectral
Testing
Governance
Deployment
Docs
POSTMAN SQUADS
POSTMAN API
POSTMAN INTERNAL SERVICES

Test examples



Contact information

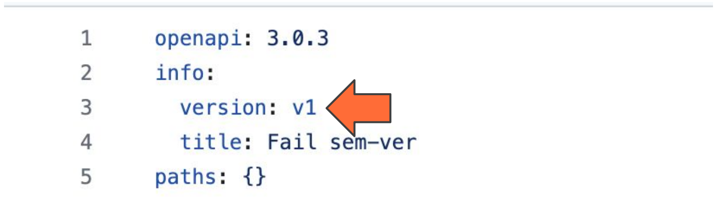
✗

```
openapi: 3.0.3
info:
  version: 1.0.0
  title: Fail Contact info
  description: A Spacecraft API
```

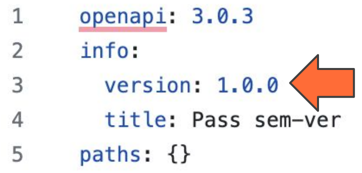
✓

```
openapi: 3.0.3
info:
  version: 1.0.0
  title: Pass Contact info
  description: A Spacecraft API
  contact:
   name: Spacecraft API team
   email: spacecraft-api@example.com
   x-slack-channel-id: CV1UH7H27
```

Semantic Versioning



```yaml
openapi: 3.0.3
info:
  version: v1
  title: Fail sem-ver
paths: {}
```



```yaml
openapi: 3.0.3
info:
  version: 1.0.0
  title: Pass sem-ver
paths: {}
```

Resources are plural nouns

✗

```yaml
openapi: 3.0.3
info:
  version: 1.0.0
  title: Fail Path plural

paths:
  /spacecraft/{id}:
    get:
      responses:
        '200':
          description: OK
```

✗

```yaml
openapi: 3.0.3
info:
  version: 1.0.0
  title: Fail Path Plural

paths:
  /spacecraft/{id}/launch:
    post:
      responses:
        '200':
          description: OK
```

✓

```yaml
openapi: 3.0.3
info:
  version: 1.0.0
  title: Pass Path plural

paths:
  /spacecrafts/{id}:
    get:
      responses:
        '200':
          description: OK
```

No resource nesting

✗

```yaml
openapi: 3.0.3
info:
  version: 1.0.0
  title: Fail Resource Nesting

paths:
  /account/transfer:
    get:
      responses:
        '200':
          description: OK
```

✓

```yaml
openapi: 3.0.3
info:
  version: 1.0.0
  title: Pass Resource Nesting

paths:
  /account-transfers:
    get:
      responses:
        '200':
          description: OK
```

✓

```yaml
openapi: 3.0.3
info:
  version: 1.0.0
  title: Pass Resource Nesting

paths:
  /accounts/{accountId}/transfers:
    get:
      responses:
        '200':
          description: OK
```

Camel case parameters

✗

```
openapi: 3.0.3
info:
  version: 1.0.0
  title: Fail Query Parameter

paths:
 /spacecrafts:
   get:
    parameters:
      - in: query
        name: spacecraft-id
        schema:
         type: string
    responses:
      '200':
        description: OK
```

✓

```
openapi: 3.0.3
info:
  version: 1.0.0
  title: Pass Query Parameter

paths:
 /spacecrafts:
   get:
    parameters:
      - in: query
        name: spacecraftId
        schema:
         type: string
    responses:
      '200':
        description: OK
```

Date Format

```yaml
openapi: 3.0.3
info:
  version: 1.0.0
  title: Fail Date Format

paths:
  /spacecrafts:
   post:
    requestBody:
     content:
       application/json:
        schema:
         properties:
           createdAt:
            type: number
    responses:
      '200':
        description: OK
```

```yaml
openapi: 3.0.3
info:
  version: 1.0.0
  title: Pass Date Format

paths:
  /resources:
   post:
    requestBody:
     content:
       application/json:
        schema:
         properties:
              createdAt:
                  type: string
                  format: date
                  example: '2023-06-16T06:43:34-07:00'
    responses:
      '200':
        description: OK
```

Collection must support pagination

✗

```yaml
openapi: 3.0.3
info:
  version: 1.0.0
  title: Pass Paginated

paths:
  /spacecrafts:
   get:
    responses:
      '200':
        description: OK
        content:
          application/json:
            schema:
             ...
```

✓

```yaml
openapi: 3.0.3
info:
  version: 1.0.0
  title: Pass Paginated

paths:
  /spacecrafts:
   get:
    responses:
      '200':
        description: OK
        content:
          application/json:
            schema:
              properties:
                offset:
                  type: number
                limit:
                  type: number
```

✓

```yaml
openapi: 3.0.3
info:
  version: 1.0.0
  title: Pass Paginated

paths:
  /spacecrafts:
   get:
    responses:
      '200':
        description: OK
        content:
          application/json:
            schema:
              properties:
                nextCursor:
                  type: string
```

Collections must support sorting

✗

```
openapi: 3.0.3
info:
  version: 1.0.0
  title: Sorting
paths:
  /resources:
   get:
    responses:
      '200':
        description: OK
```

✓

```
openapi: 3.0.3
info:
  version: 1.0.0
  title: Sorting
paths:
  /resources:
   get:
    parameters:
      - in: query
        name: sort
        schema:
          type: string
    responses:
      '200':
        description: OK
```

Parameters should have examples

✗

```yaml
openapi: 3.0.3
info:
  version: 1.0.0
  title: Parameter examples

paths:
  /resources:
   get:
    parameters:
      - in: query
        name: cursor
        schema:
          type: string
    responses:
      '200':
        description: OK
```

✓

```yaml
openapi: 3.0.3
info:
  version: 1.0.0
  title: Parameter examples

paths:
  /resources:
   get:
    parameters:
      - in: query
        name: cursor
        schema:
          type: string
        example: adfds23423ASDFasdfwerwq
    responses:
      '200':
        description: OK
```

No version in paths

✗

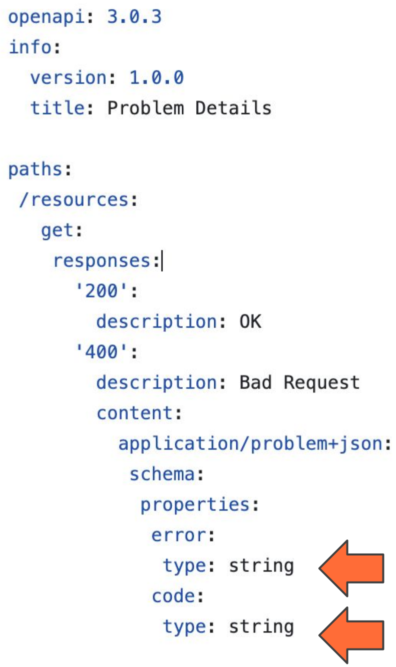
```
openapi: 3.0.3
info:
  version: 1.0.0
  title: No Version in Path

paths:
  /v1/spacecrafts:
   get:
    responses:
      '200':
        description: OK
```

✓

```
openapi: 3.0.3
info:
  version: 1.0.0
  title: No Version in Path

paths:
  /spacecrafts:
   get:
    responses:
      '200':
        description: OK
```

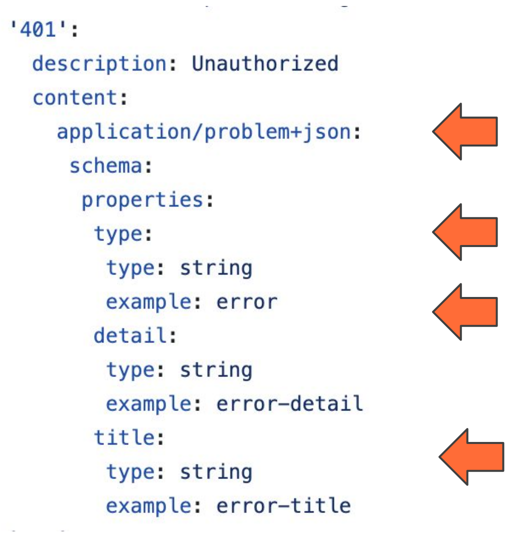
Error format - Problem Details

✗



```yaml
openapi: 3.0.3
info:
  version: 1.0.0
  title: Problem Details

paths:
 /resources:
   get:
    responses:
      '200':
        description: OK
      '400':
        description: Bad Request
        content:
          application/problem+json:
           schema:
            properties:
             error:
              type: string
             code:
              type: string
```

✓



```yaml
'401':
  description: Unauthorized
  content:
    application/problem+json:
     schema:
      properties:
       type:
        type: string
        example: error
       detail:
        type: string
        example: error-detail
       title:
        type: string
        example: error-title
```

Challenges

- **Testing the tests**

  Since the tests are configuration files we need actually writing tests for them again

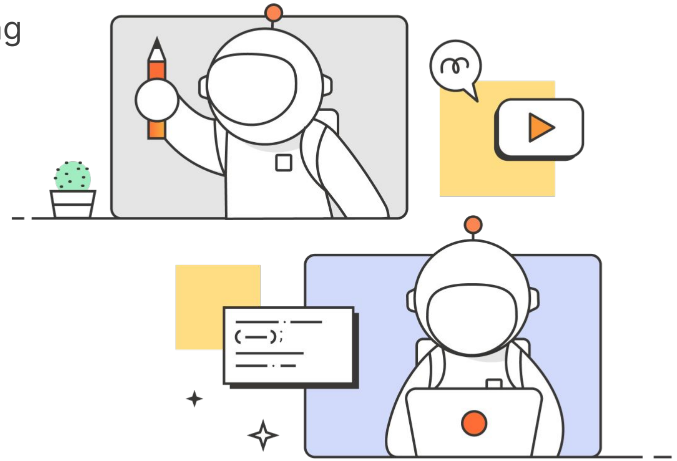
- **Some things are hard to test even with Spectral**

  For example descriptive names, but AI could help with that

- **Re-evaluating tests**

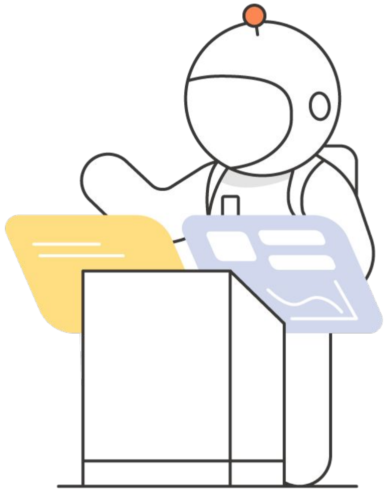
  As our API Design Guidelines involve we have to constantly adapt and update our tests

- **Design dept and breaking changes**
  Some of the old endpoints are not compliant but we cannot change them as this wall cause errors in existing users.



Actions that you can do

1. Figure out the API Design style you need
2. Create Spectral rules to codify it
3. Find a point in the critical path in the delivery life cycle that compliance testing can be performed
4. Enforce the use of OpenAPI
5. Implement your compliance testing
6. Come meet me at the Postman booth for further discussion.



Thank you