

```

1 describe('API chaining in Cypress', () => {
2   it('should make API requests and chain them together', () => {
3     cy.request(
4       {
5         method: 'GET',
6         url: 'https://jsonplaceholder.typicode.com/posts'
7       })
8     .then((response) => {
9       expect(response.status).to.eq(200)
10      //extract the post id and store in variable.
11      const postId = response.body[0].id
12      //use return for request chaining.
13      //this is the first then statement.
14      return postId
15    })
16    //one request can have multiple then blocks.
17    //use the postId in another response as query parameter.
18    //then can also have any damn thing inside it,
19    //including request.
20    .then((postId) => {
21      cy.request({
22        method: 'GET',
23        url: `https://jsonplaceholder.typicode.com/comments?postId=${postId}`
24      })
25      .then((response) => {
26        expect(response.status).to.eq(200)
27        expect(response.body).to.have.length(5)
28      })
29    })
30  })
31
32
33 })

```

codesnap.dev

Another way for chaining using nested thens –

```
1  /*
2  POST https://gorest.co.in/public/v2/users
3  PUT https://gorest.co.in/public/v2/users/${userId}
4  DELETE https://gorest.co.in/public/v2/users/${userId}
5  */
6
7  //another way to chain request.
8  //here directly in then keep adding new request rather than return.
9  //also see one then can contain hundreds of other thens.
10 describe('GoRest User API Chaining', () => {
11
12     const auth_token='Bearer c35e10e748c6f113775527bcef204e9929b4c9f4b995a8ee253eec46aed57b06'
13
14     it('should create, update, and delete a user', () => {
15         // Create a new user
16         cy.request({
17             method:'POST',
18             url: 'https://gorest.co.in/public/v2/users',
19             body: {
20                 name: 'John Kenedy',
21                 gender: 'male',
22                 email: Math.random().toString(5).substring(2)+"@gmail.com",
23                 status: 'active'
24             },
25             headers:{
26                 Authorization: auth_token
27             }
28         })
```

codesnap.dev

Continue from line 28 –
Add from line 29 after the closure of request.

```
1 .then((response) => {
2     expect(response.status).to.equal(201)
3     const userId = response.body.id
4     // Update the user's name
5     cy.request(
6         {
7             method: 'PUT',
8             url: `https://gorest.co.in/public/v2/users/${userId}`,
9             body: {
10                 name: 'Scott'
11             },
12             headers:{
13                 Authorization: auth_token
14             }
15         })
16     .then((response) => {
17         expect(response.status).to.equal(200)
18         expect(response.body.name).to.equal('Scott')
19         // Delete the user
20         cy.request(
21             {
22                 method:'DELETE',
23                 url: `https://gorest.co.in/public/v2/users/${userId}`,
24                 headers:{
25                     Authorization: auth_token
26                 }
27             })
28         .then((response) => {
29             expect(response.status).to.equal(204)
30         })
31     })
32 })
33 })
34 })
35
```

codesnap.dev

Trying to paste all in one and seeing –

```

1  /*
2  POST https://gorest.co.in/public/v2/users
3  PUT https://gorest.co.in/public/v2/users/${userId}
4  DELETE https://gorest.co.in/public/v2/users/${userId}
5  */
6
7  //another way to chain request.
8  //here directly in then keep adding new request rather than return.
9  //also see one then can contain hundreds of other thens.
10 describe('GoRest User API Chaining', () => {
11
12     const auth_token=
13     'Bearer c35e10e748c6f113775527bcef204e9929b4c9f4b995a8ee253eec46aed57b06'
14
15     it('should create, update, and delete a user', () => {
16         // Create a new user
17         cy.request({
18             method: 'POST',
19             url: 'https://gorest.co.in/public/v2/users',
20             body: {
21                 name: 'John Kenedy',
22                 gender: 'male',
23                 email: Math.random().toString(5).substring(2)+"@gmail.com",
24                 status: 'active'
25             },
26             headers:{
27                 Authorization: auth_token
28             }
29         })
30         .then((response) => {
31             expect(response.status).to.equal(201)
32             const userId = response.body.id
33             // Update the user's name
34             cy.request(
35                 {
36                     method: 'PUT',
37                     url: `https://gorest.co.in/public/v2/users/${userId}`,
38                     body: {
39                         name: 'Scott'
40                     },
41                     headers:{
42                         Authorization: auth_token
43                     }
44                 })
45             .then((response) => {
46                 expect(response.status).to.equal(200)
47                 expect(response.body.name).to.equal('Scott')
48                 // Delete the user
49                 cy.request(
50                     {
51                         method: 'DELETE',
52                         url: `https://gorest.co.in/public/v2/users/${userId}`,
53                         headers:{
54                             Authorization: auth_token
55                         }
56                     })
57                 .then((response) => {
58                     expect(response.status).to.equal(204)
59                 })
60             })
61         })
62     })
63 })

```

Below is also time pass to see how it looks if we paste all—

```

1  /*
2  POST https://gorest.co.in/public/v2/users
3  PUT https://gorest.co.in/public/v2/users/${userId}
4  DELETE https://gorest.co.in/public/v2/users/${userId}
5  */
6
7  //another way to chain request.
8  //here directly in then keep adding new request rather than return.
9  //also see one then can contain hundreds of other thens.
10 describe('GoRest User API Chaining', () => {
11
12     const auth_token=
13     'Bearer c35e10e748c6f113775527bcef204e9929b4c9f4b995a8ee253eec46aed57b06'
14
15     it('should create, update, and delete a user', () => {
16         // Create a new user
17         cy.request({
18             method: 'POST',
19             url: 'https://gorest.co.in/public/v2/users',
20             body: {
21                 name: 'John Kenedy',
22                 gender: 'male',
23                 email: Math.random().toString(5).substring(2)+"@gmail.com",
24                 status: 'active'
25             },
26             headers: {
27                 Authorization: auth_token
28             }
29         })
30         .then((response) => {
31             expect(response.status).to.equal(201)
32             const userId = response.body.id
33             // Update the user's name
34             cy.request(
35                 {
36                     method: 'PUT',
37                     url: `https://gorest.co.in/public/v2/users/${userId}`,
38                     body: {
39                         name: 'Scott'
40                     },
41                     headers: {
42                         Authorization: auth_token
43                     }
44                 })
45             .then((response) => {
46                 expect(response.status).to.equal(200)
47                 expect(response.body.name).to.equal('Scott')
48                 // Delete the user
49                 cy.request(
50                     {
51                         method: 'DELETE',
52                         url: `https://gorest.co.in/public/v2/users/${userId}`,
53                         headers: {
54                             Authorization: auth_token
55                         }
56                     })
57                 .then((response) => {
58                     expect(response.status).to.equal(204)
59                 })
60             })

```