

Using Vuex Actions to Make API Calls

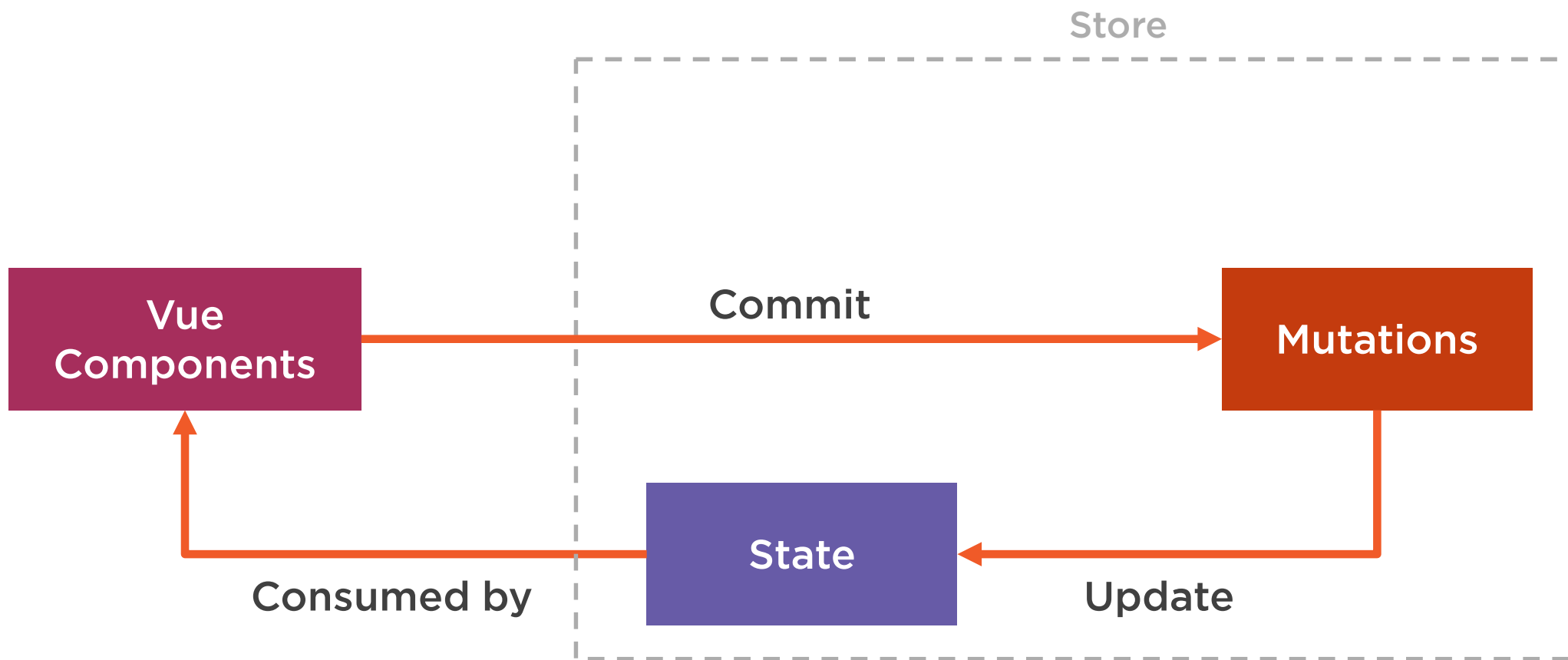


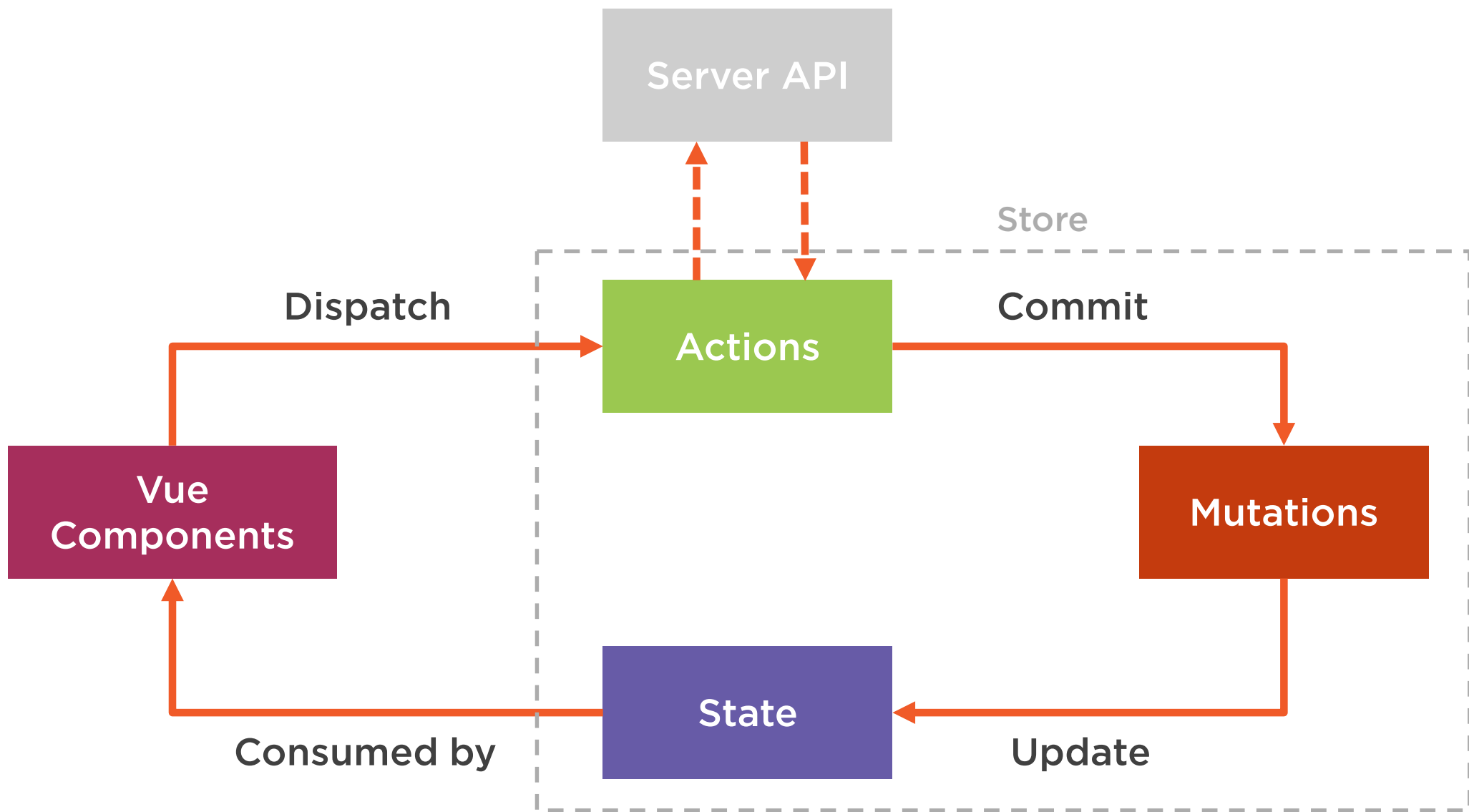
Jim Cooper

SOFTWARE ENGINEER

@jimthecoop jcoop.io







Mutations vs. Actions

Mutations

Synchronous

Can Directly Mutate State

Actions

Asynchronous

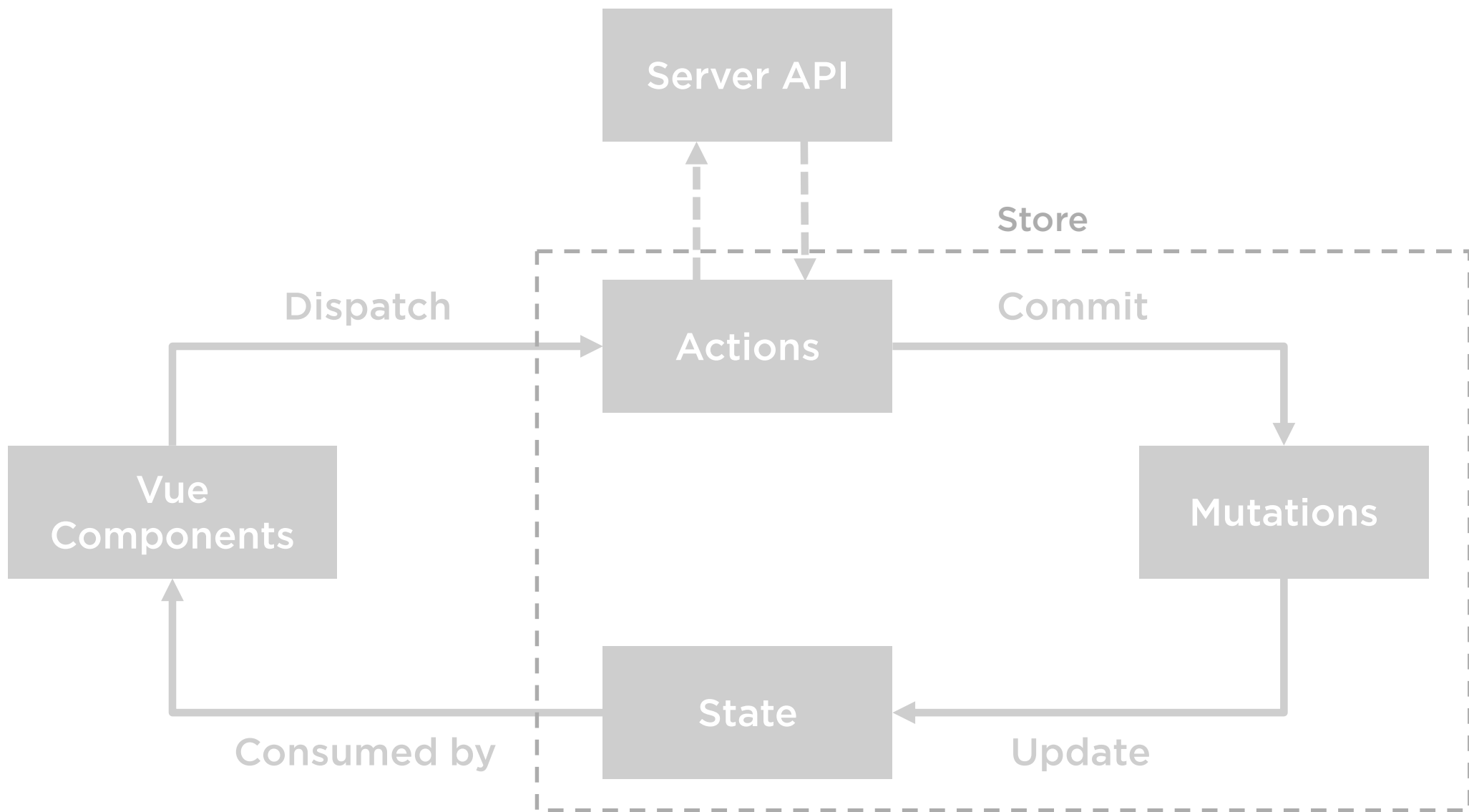
Commits Mutations



But Why?

Answer: Trackability

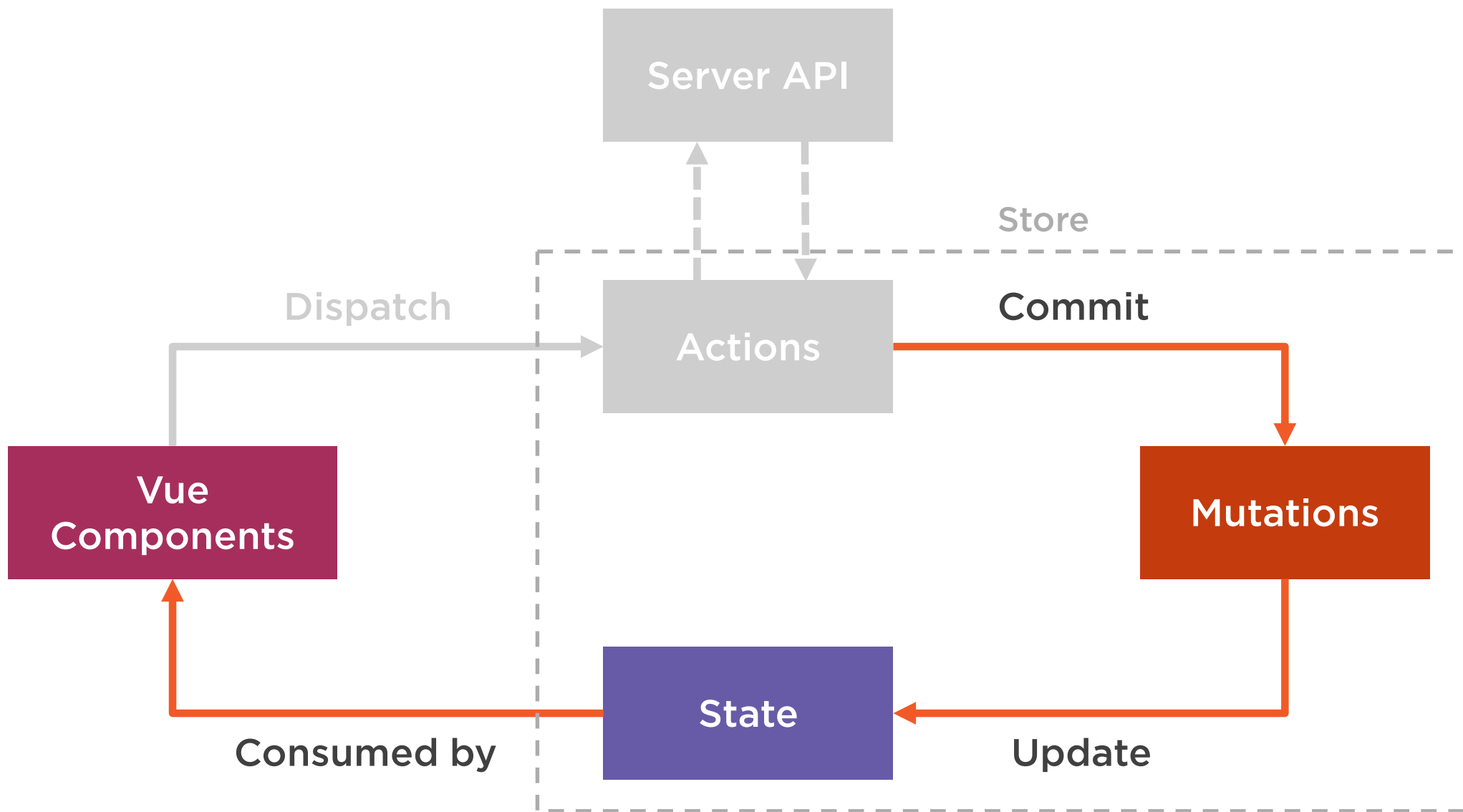






Let's review





Mutations vs. Actions

Mutations

Synchronous

Can Directly Mutate State

Actions

Asynchronous

Commits Mutations



Proxying an API Server

```
module.exports = {  
  devServer: {  
    proxy: {  
      '/api' : {  
        target: 'http://localhost:8081'  
      }  
    }  
  }  
}
```



Retrieving Data With Actions

```
actions: {  
  fetchProducts({ commit }) {  
    axios.get('/api/products')  
      .then((result) => commit('setProducts', result.data));  
  },  
}
```



Dispatching Actions

```
created() {  
  this.$store.dispatch('fetchProducts');  
},
```



Posting Data With Actions

```
registerUser({ commit }, user) {  
  return axios.post('/api/register', user)  
    .then((result) => commit('setUser', result.data));  
},
```

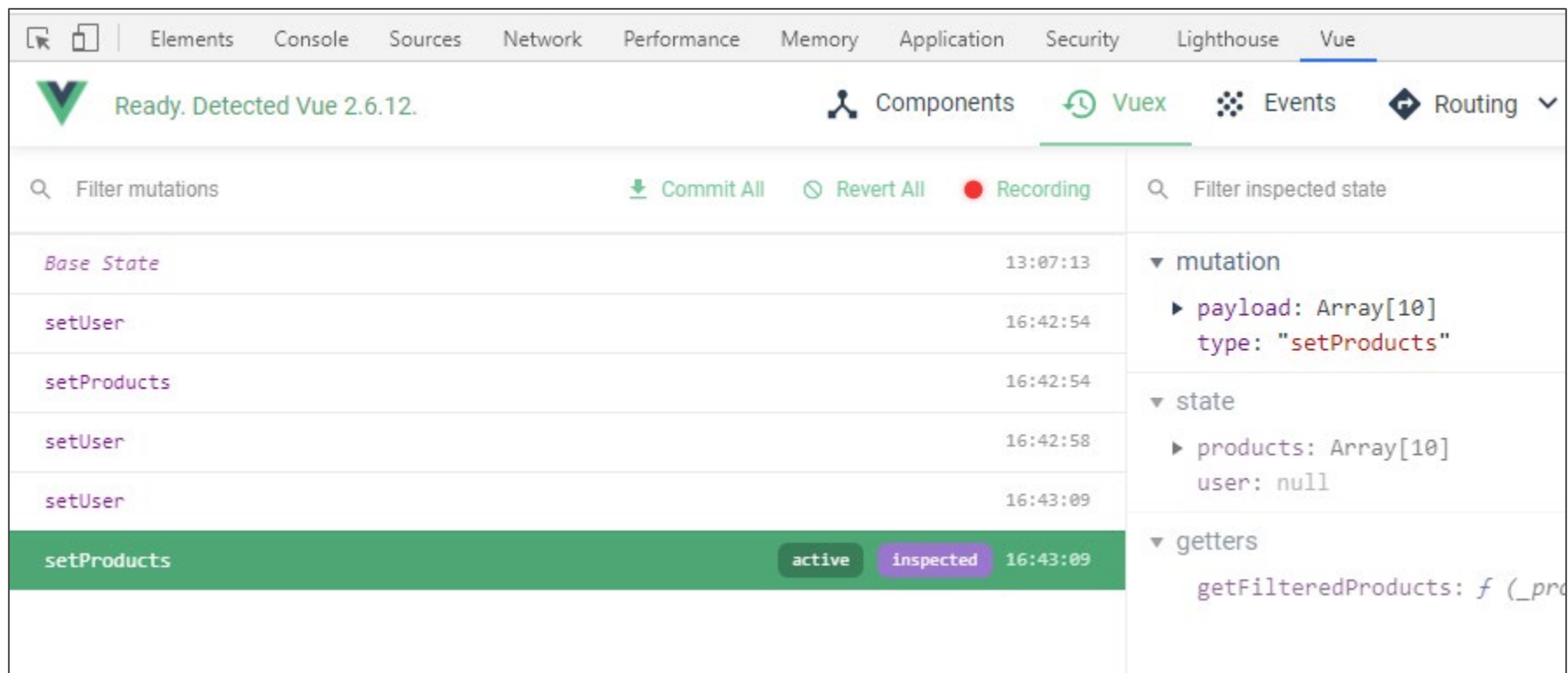


Posting Data to an API

```
this.$store.dispatch('signIn', userLogin)  
  .then(() => this.$router.push('/products'))  
  .catch(() => { this.signInError = true; });
```



Using Vue.js Dev Tools



The screenshot displays the Vue.js DevTools interface, specifically the Vuex tab. The top navigation bar includes tabs for Elements, Console, Sources, Network, Performance, Memory, Application, Security, Lighthouse, and Vue. The Vuex tab is active, showing a status bar with "Ready. Detected Vue 2.6.12." and icons for Components, Vuex, Events, and Routing. Below the status bar, there are controls for "Filter mutations", "Commit All", "Revert All", and "Recording". The main panel lists several mutations:

Mutation	Time
Base State	13:07:13
setUser	16:42:54
setProducts	16:42:54
setUser	16:42:58
setUser	16:43:09
setProducts	16:43:09

The "setProducts" mutation at 16:43:09 is highlighted in green and labeled "active" and "inspected". To the right of the mutation list, the "Filter inspected state" search bar is visible. The inspected state is expanded, showing the following structure:

- mutation
 - payload: Array[10]
 - type: "setProducts"
- state
 - products: Array[10]
 - user: null
- getters
 - getFilteredProducts: f (_pro





Next we'll explore how
to organize our store
into multiple modules!

