How We Migrated Pinterest Profiles to React

Imad Elyafi imad@pinterest.com

2017

Agenda

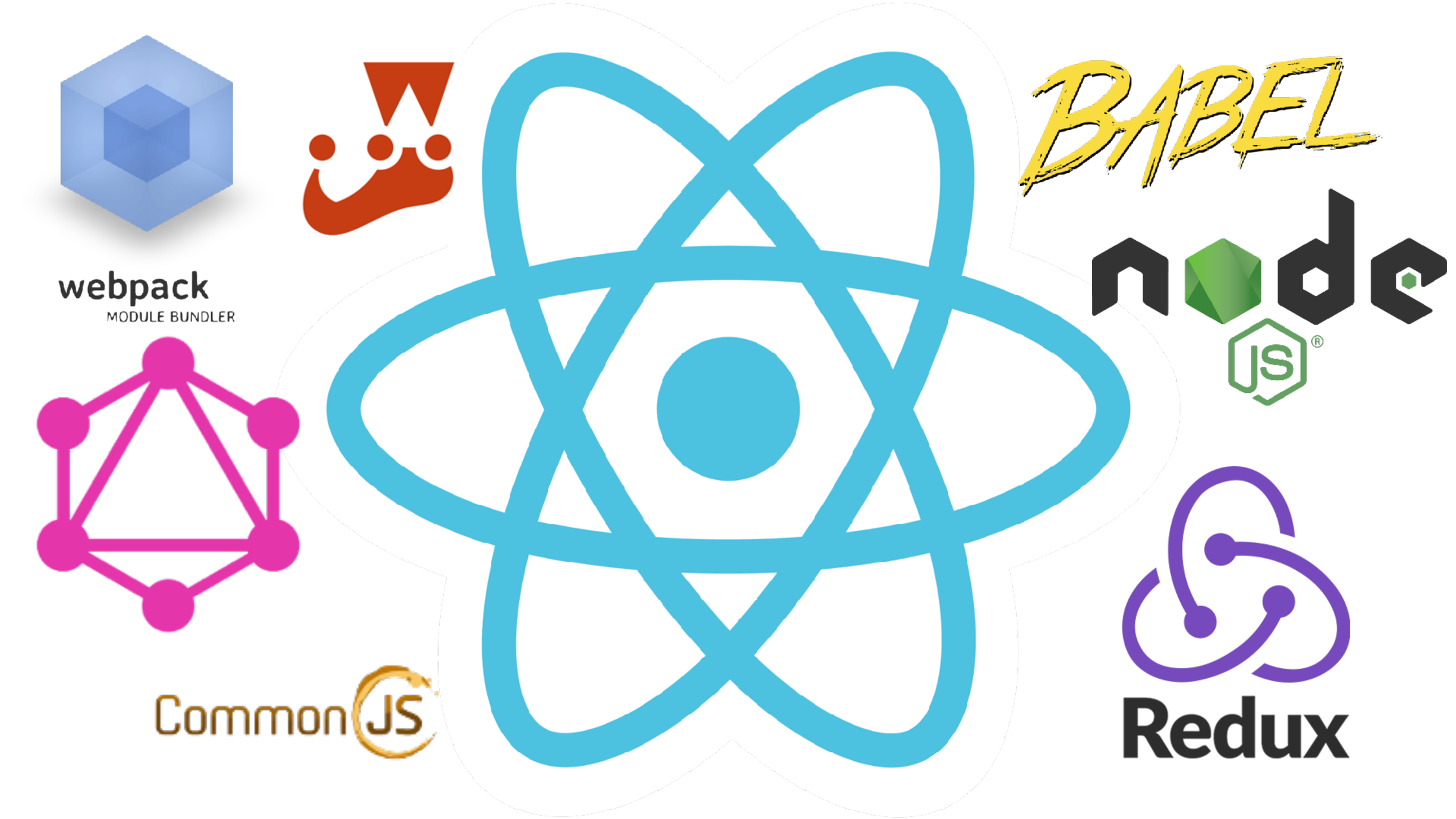
1. Why React ?
2. Road to React
3. UI experiments
4. Where are we at now

Why React ?

1 Large Developer Community
2 Design Patterns
3 Isomorphic Rendering
4 Performance
5 Developer Experience

BACKBONE.JS

python™


webpack
MODULE BUNDLER
BABEL
node
JS
CommonJS
Redux

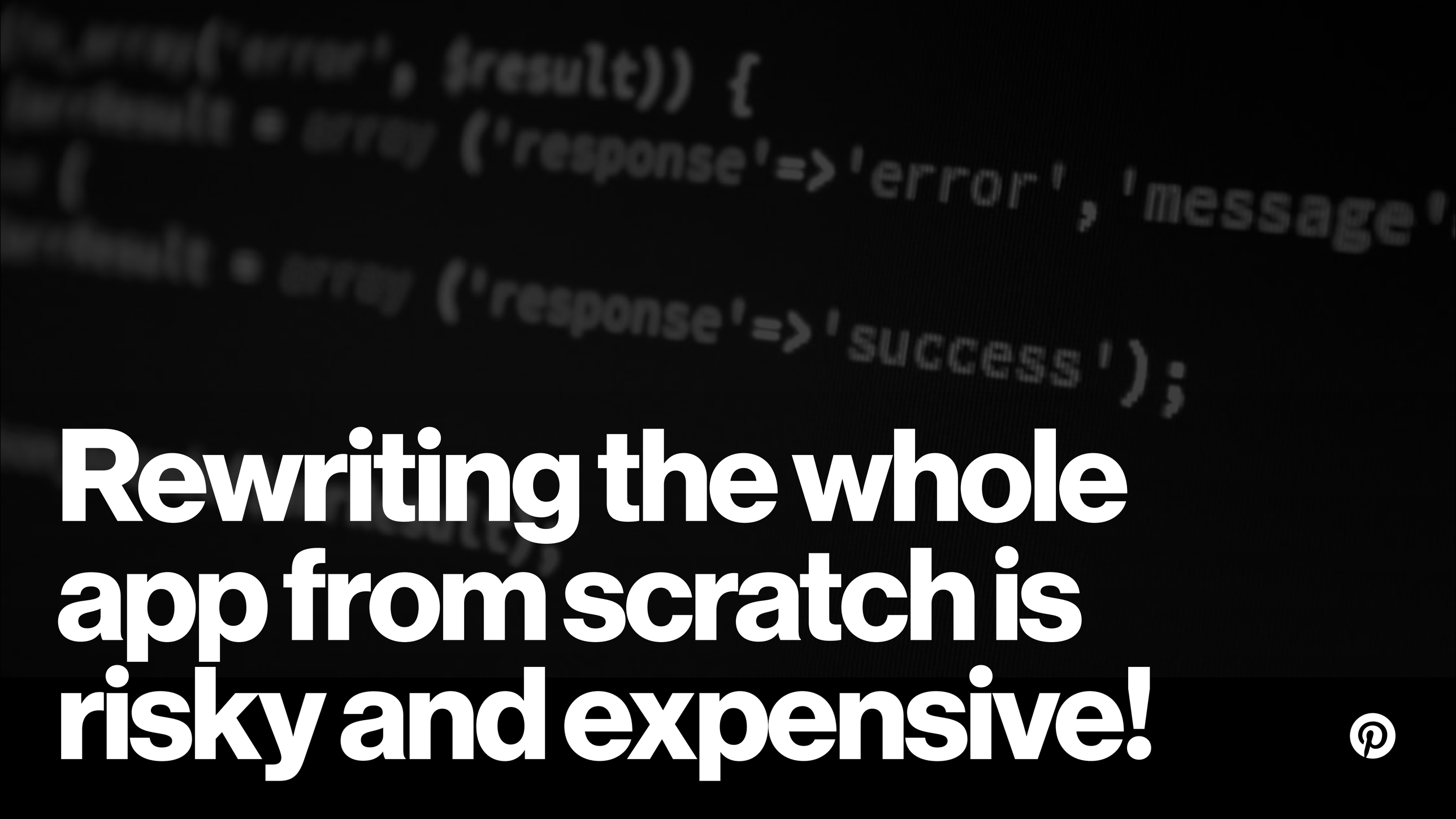
175M

Number of users on Pinterest (More than the number of residents in Russia)

150k

Requests per second


No code freeze.

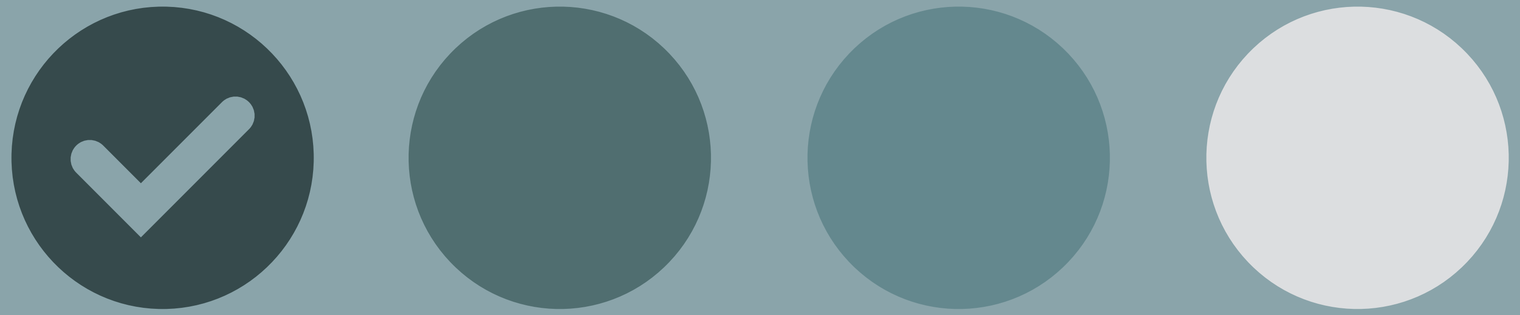

Rewriting the whole app from scratch is risky and expensive!

How can we avoid this?

Hybrid App

How can you make a Hybrid App

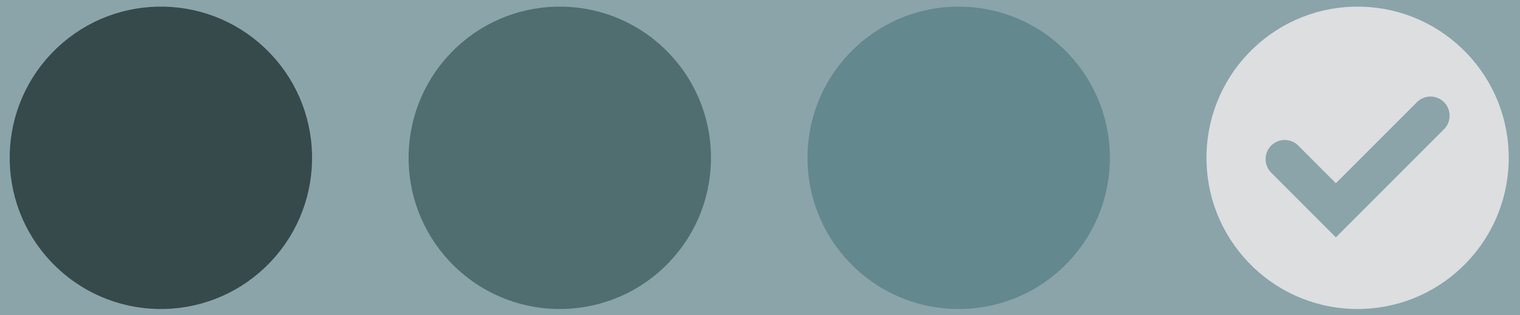
Javascript rendering on the server-side



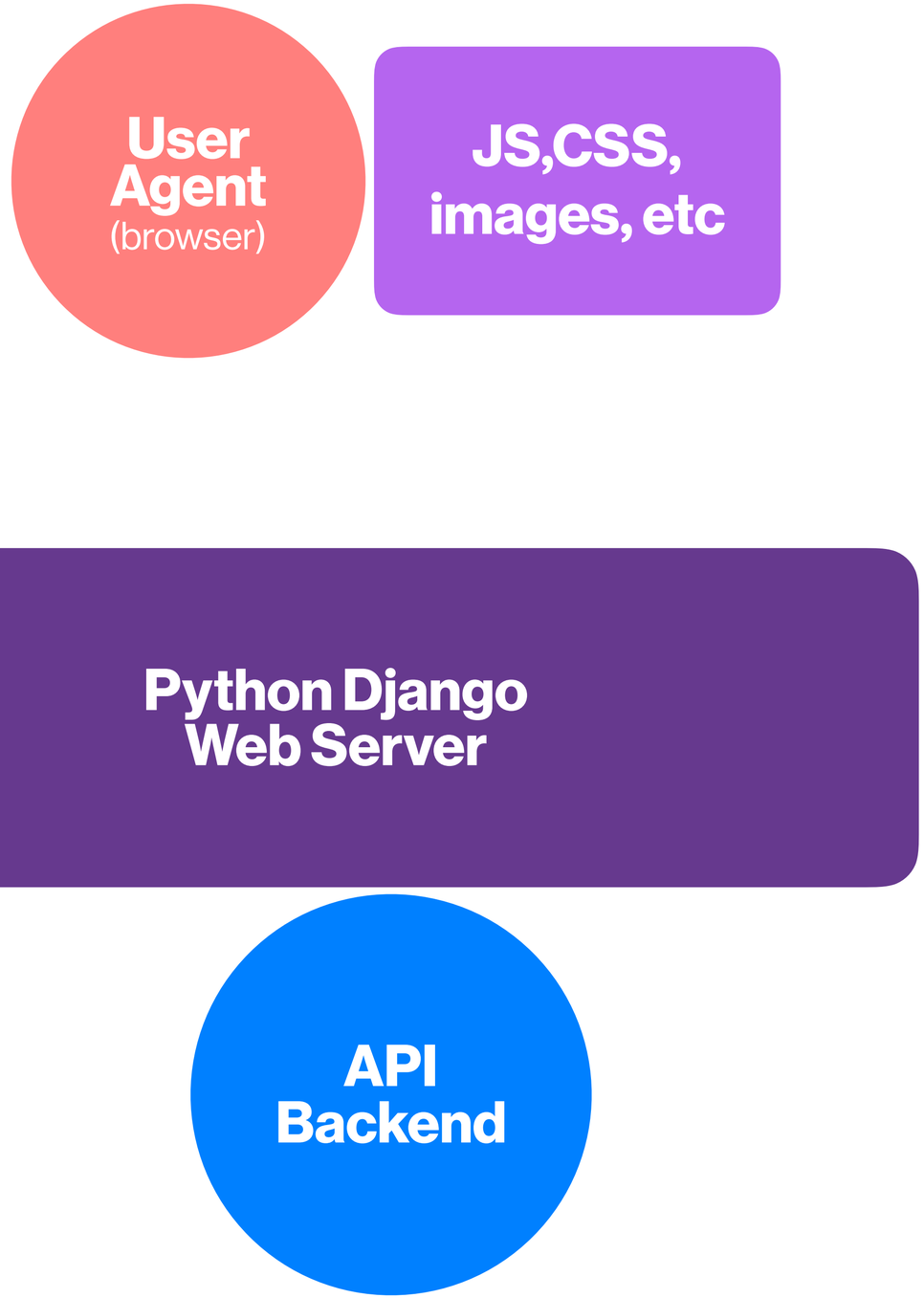
Render React components inside your old framework

Adapters for old data sources

Make sure that the new UI performs well

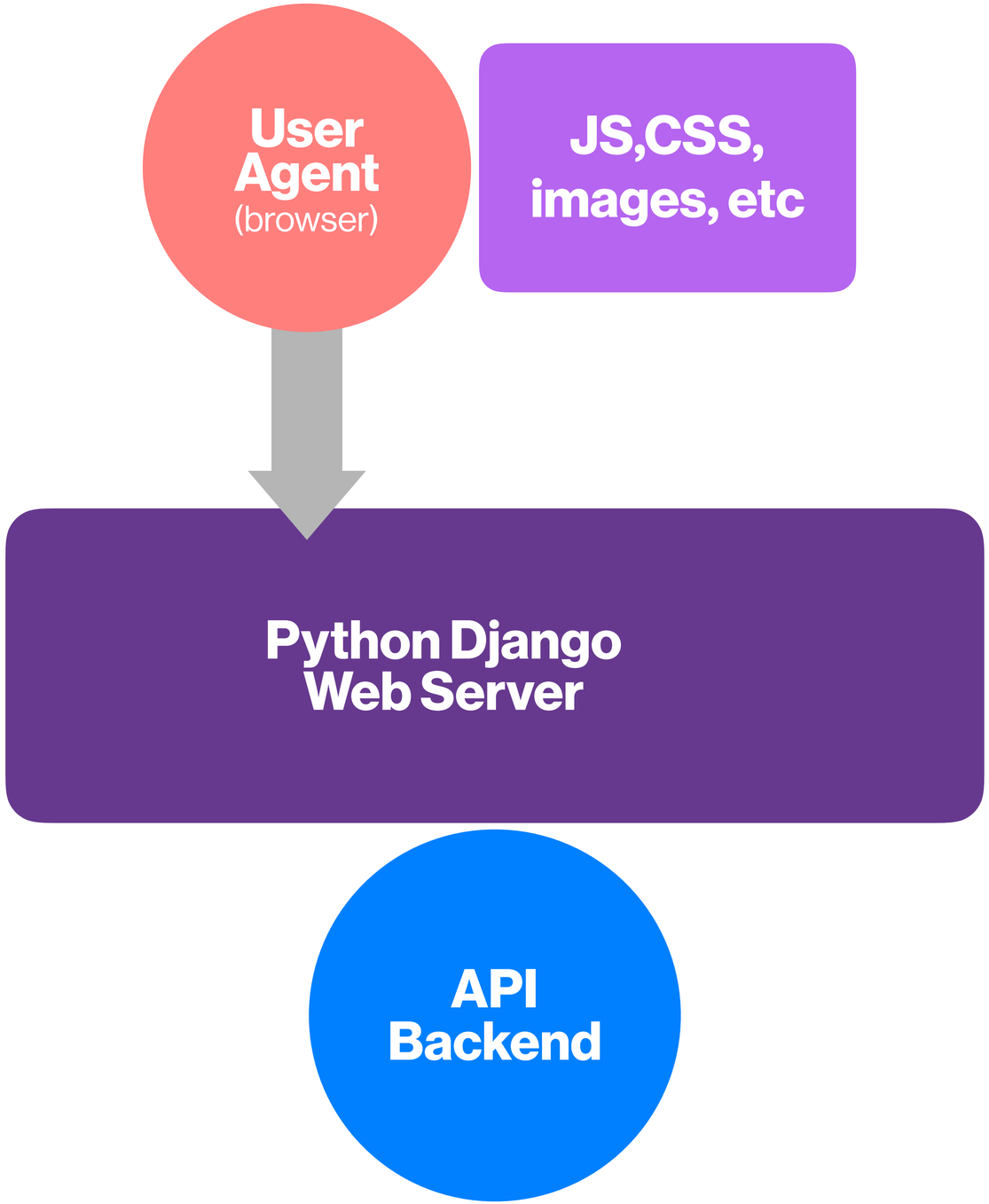


Web App Architecture



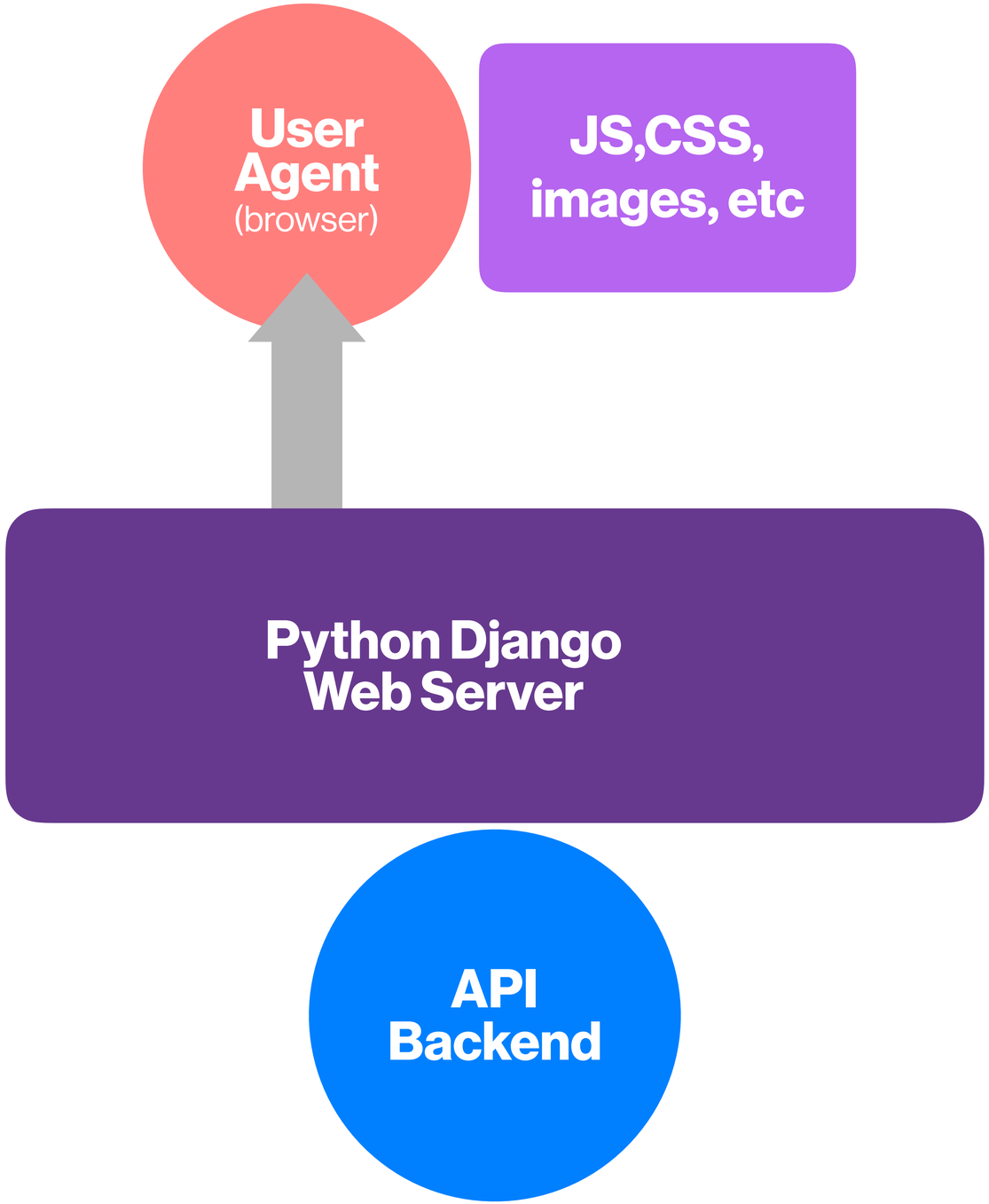
Web App Architecture

User agent requests
https://www.pinterest.com/



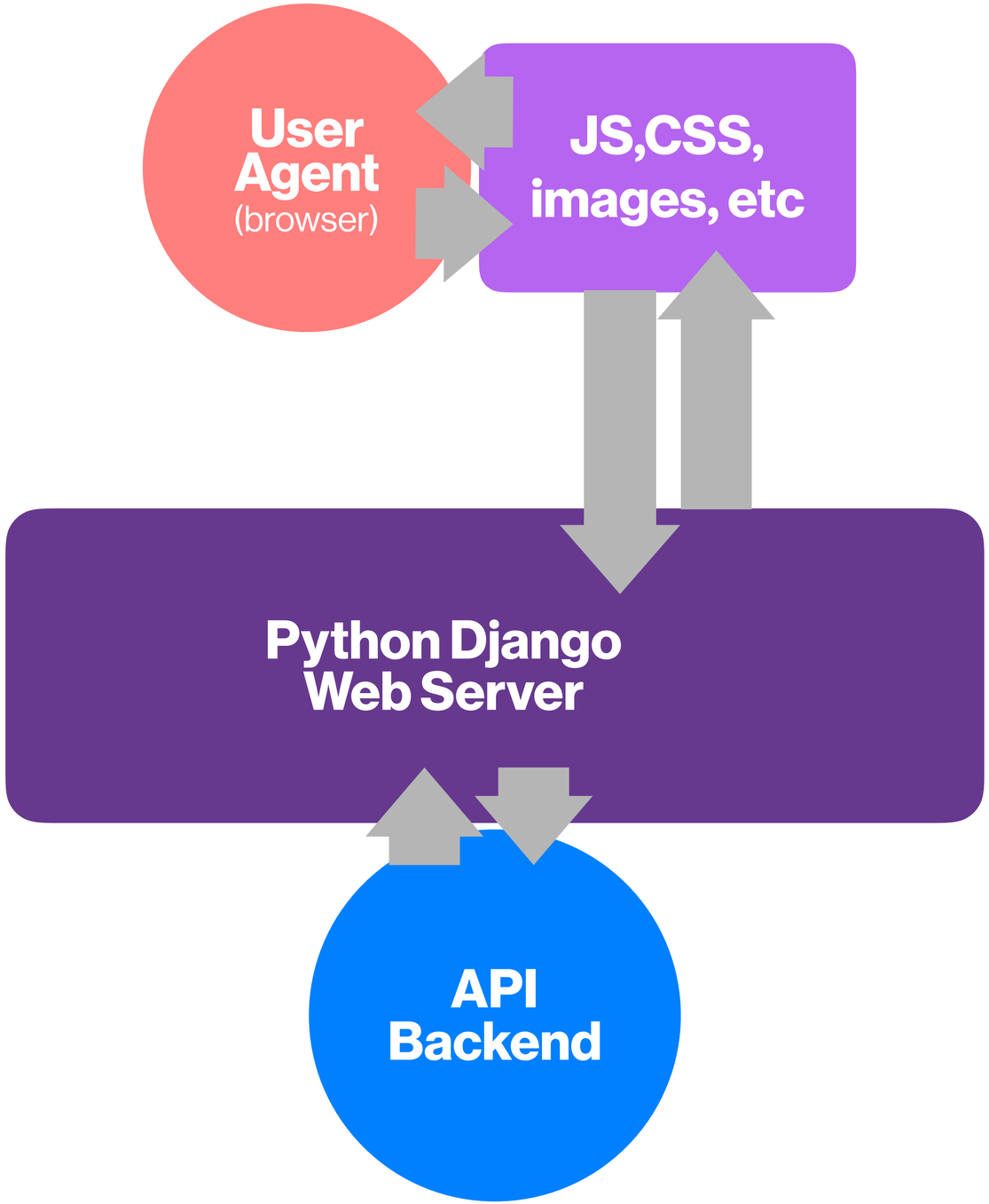
Web App Architecture

Server renders HTML with Jinja engine, and sends as a response to the client, which spawn additional requests for server and CDN.



Web App Architecture

Subsequent user agent events are handled by client javascript.
Javascript libraries Backbone & Nunjucks power client HTML re-renders.
Any requests to the server are mainly just to fetch additional data.

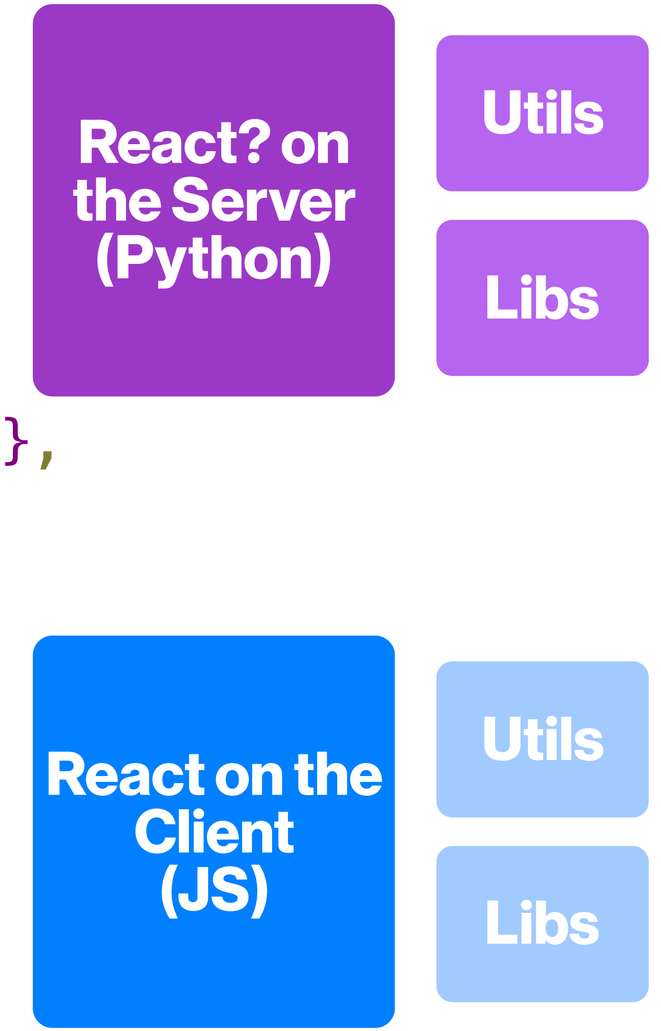


```
{% set greeting = data.get("greeting", "hello") %}
{% set name = data.get("name", "guest") %}
{% set pinData = data.get("pins") %}
```

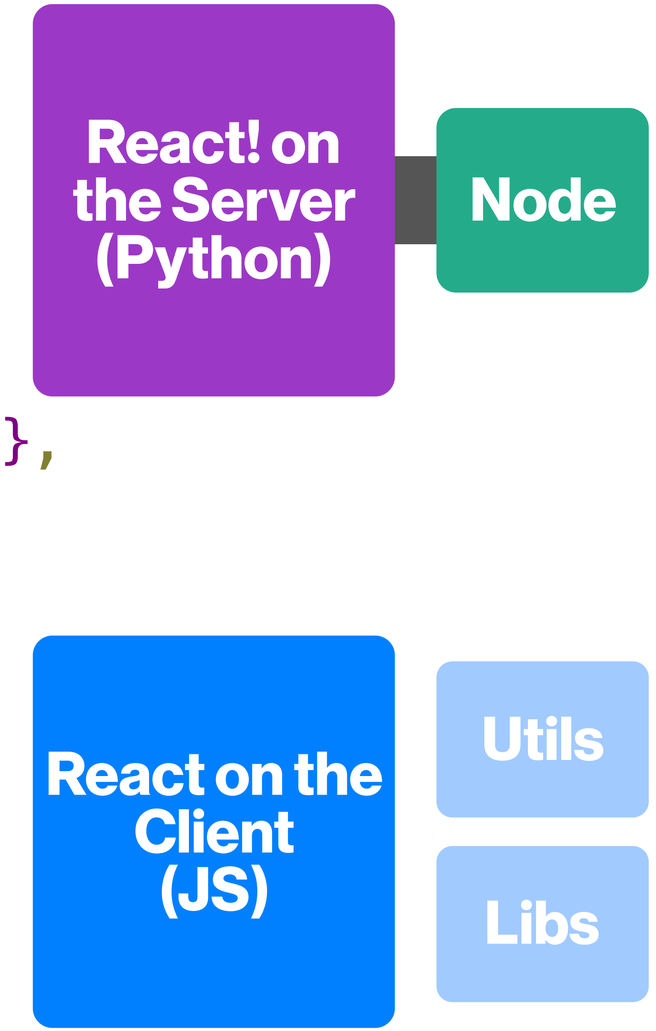
```
{% set greeting = data.get("greeting", "hello") %}
{% set name = data.get("name", "guest") %}
{% set pinData = data.get("pins") %}
{{ greeting }}, {{ name }}, welcome to Pinterest!
```

```
{% set greeting = data.get("greeting", "hello") %}
{% set name = data.get("name", "guest") %}
{% set pinData = data.get("pins") %}
{{ greeting }}, {{ name }}, welcome to Pinterest!
{{ module('HomeFeed', options={'pins': pinData,
                               'viewType': 'main'}) }}
```

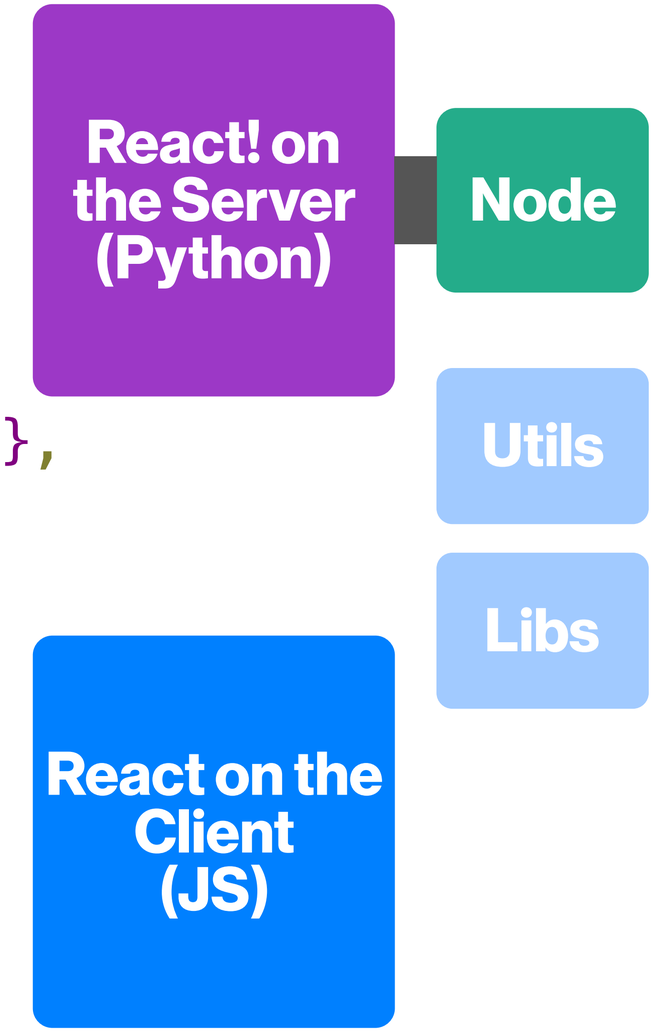
```
render() {
    return (
        <div>
            <h1>{this.props.greeting}, {this.props.name},
            welcome to Pinterest!</h1>
            <HomeFeed pins={this.props.pinData}
                      viewType='main' />
        </div>
    );
}
```



```
render() {
    return (
        <div>
            <h1>{this.props.greeting}, {this.props.name},
            welcome to Pinterest!</h1>
            <HomeFeed pins={this.props.pinData}
                      viewType='main' />
        </div>
    );
}
```

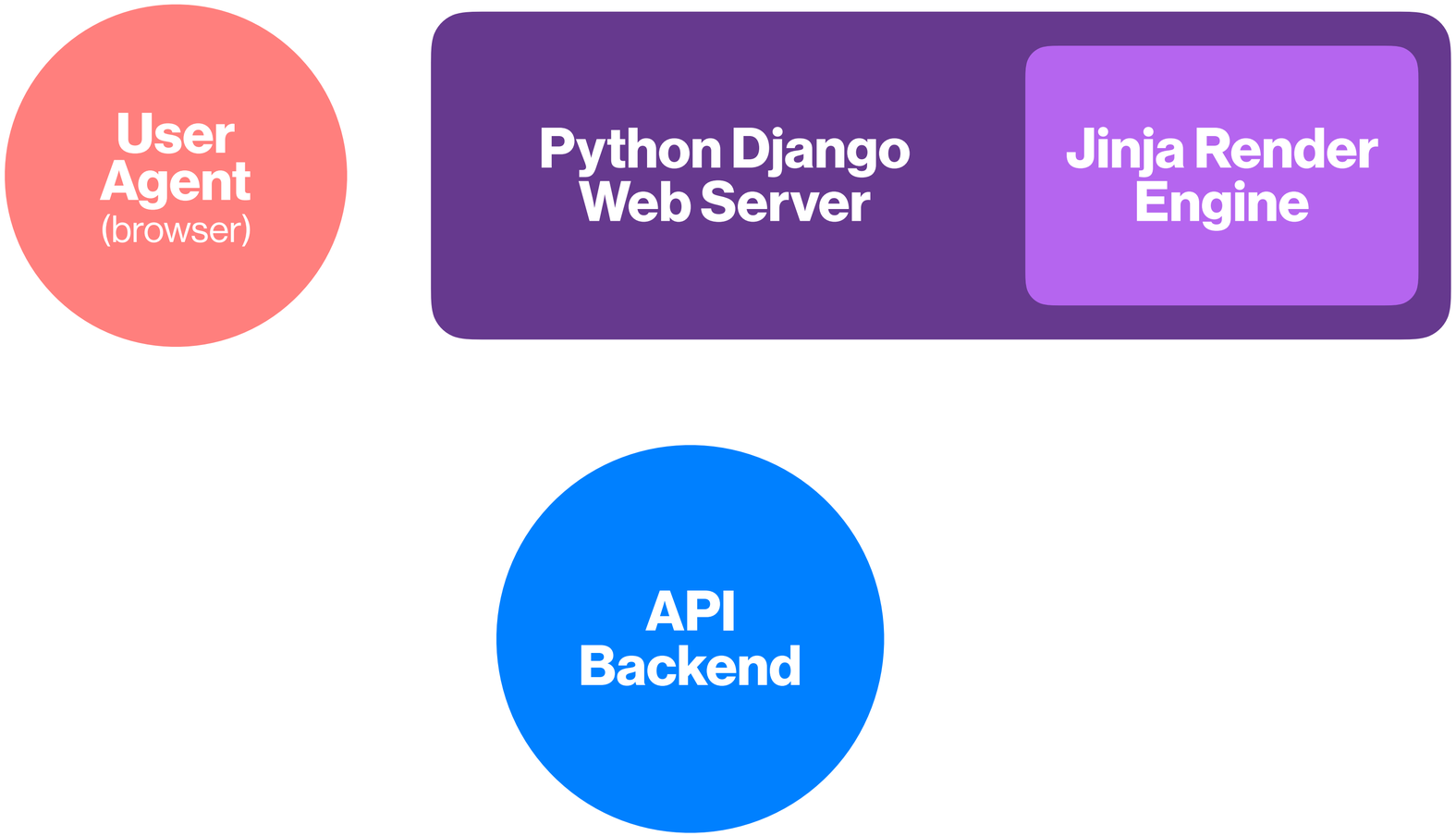


```
render() {
    return (
        <div>
            <h1>{this.props.greeting}, {this.props.name},
            welcome to Pinterest!</h1>
            <HomeFeed pins={this.props.pinData}
                      viewType='main' />
        </div>
    );
}
```

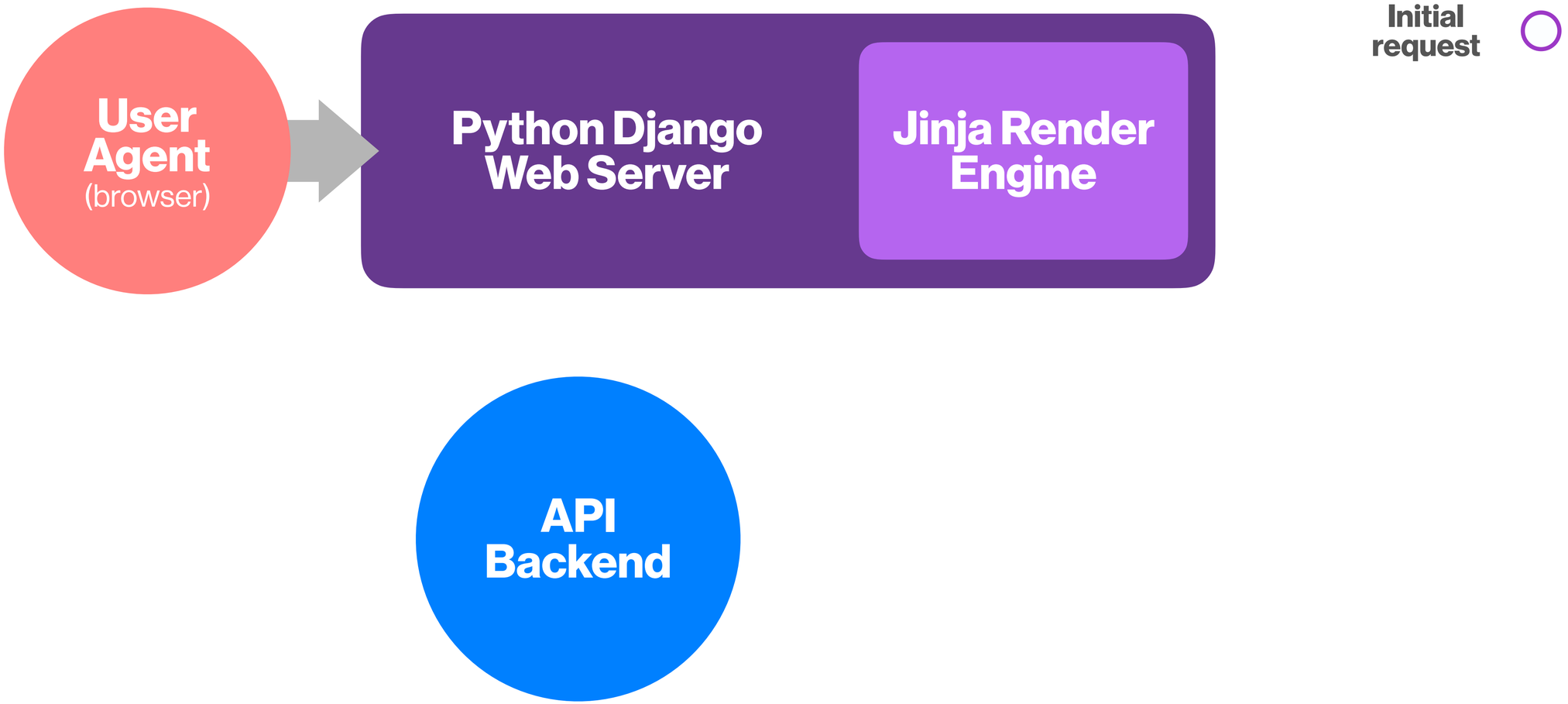



Jinja (Python)
&
Nunjucks (Client)
Nunjucks (Node)
&
Nunjucks (Client)
React (Node)
&
React (Client)

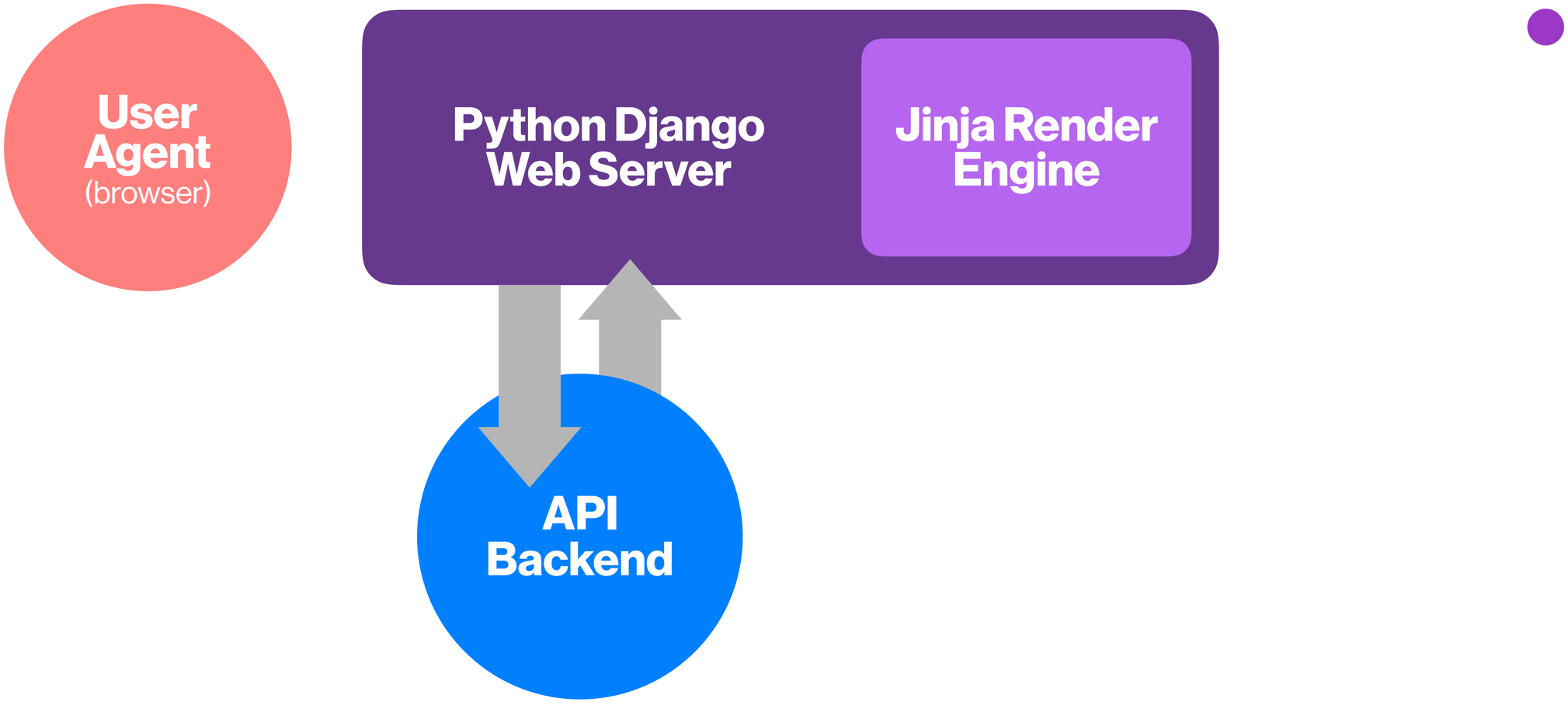
Python Jinja Renderer (before)



Python Jinja Renderer (before)



Python Jinja Renderer (before)

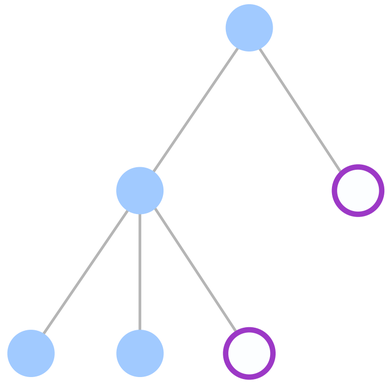


Python Jinja Renderer (before)

User Agent (browser)

Python Django Web Server

Jinja Render Engine



API Backend

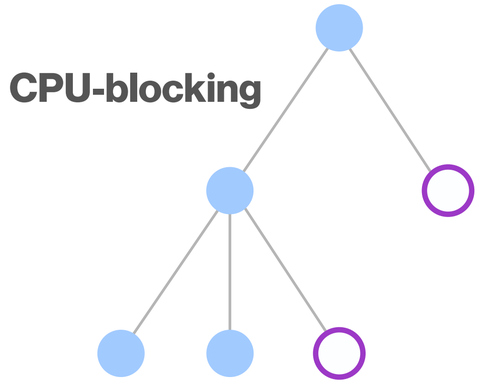
Jinja

Python Jinja Renderer (before)

User Agent (browser)

Python Django Web Server

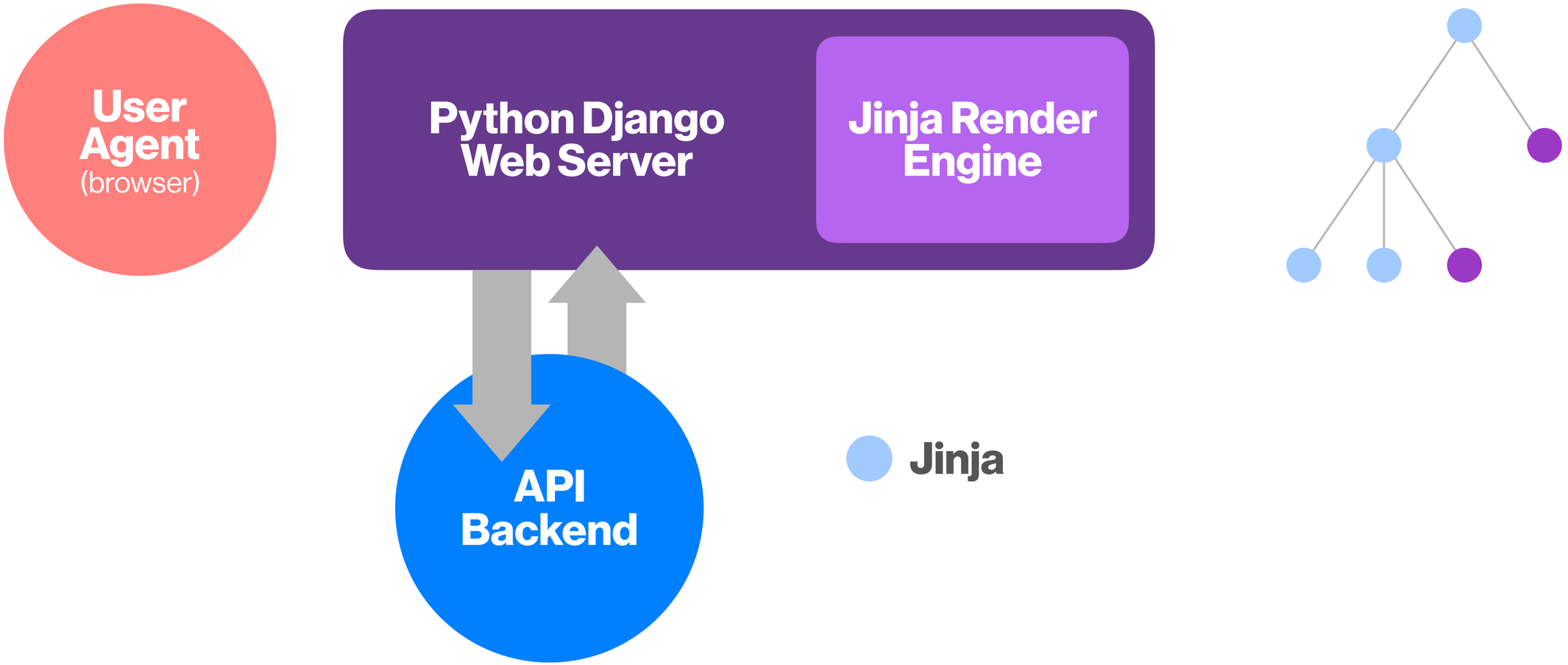
Jinja Render Engine



API Backend

Jinja

Python Jinja Renderer (before)

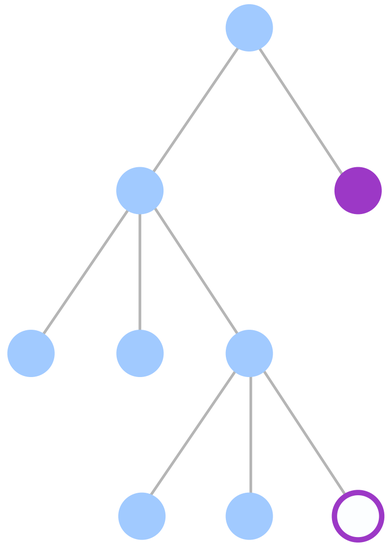


Python Jinja Renderer (before)

User Agent (browser)

Python Django Web Server

Jinja Render Engine



API Backend

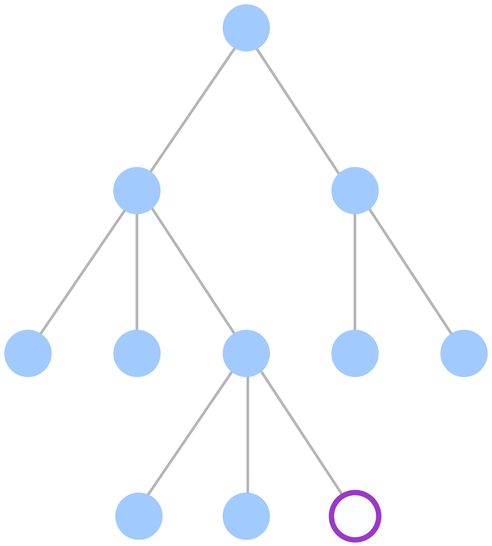
Jinja

Python Jinja Renderer (before)

User Agent (browser)

Python Django Web Server

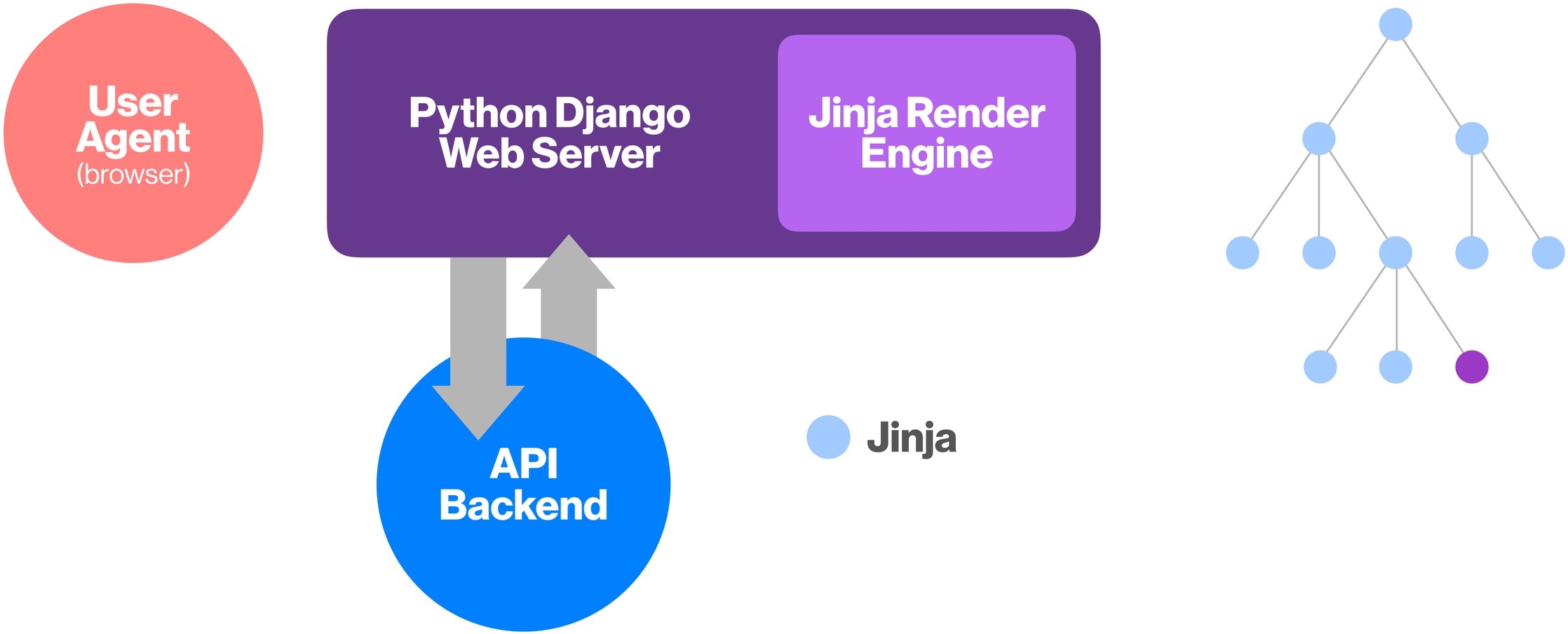
Jinja Render Engine



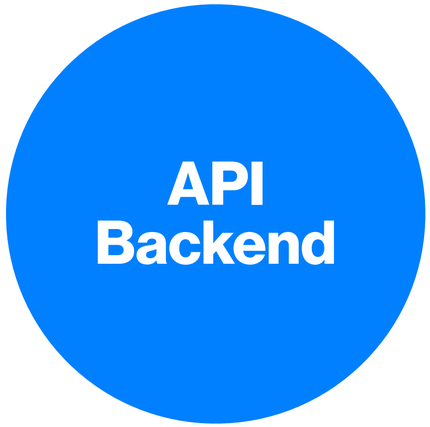
API Backend

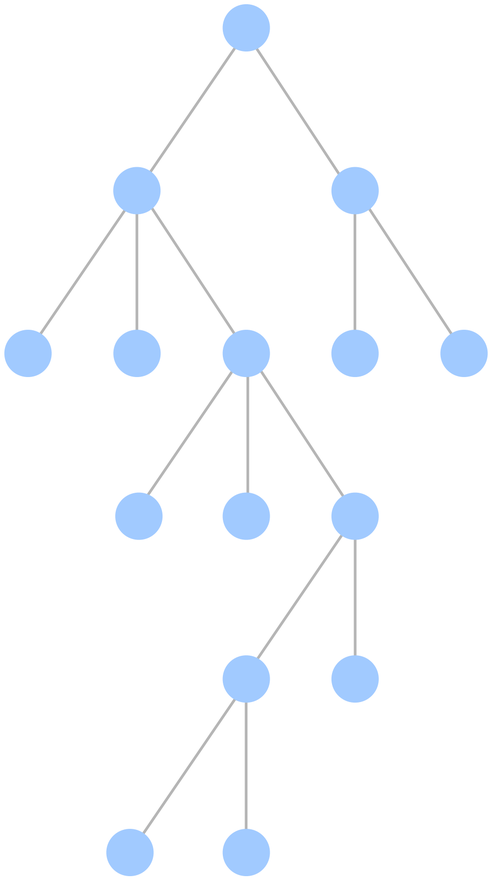
Jinja

Python Jinja Renderer (before)

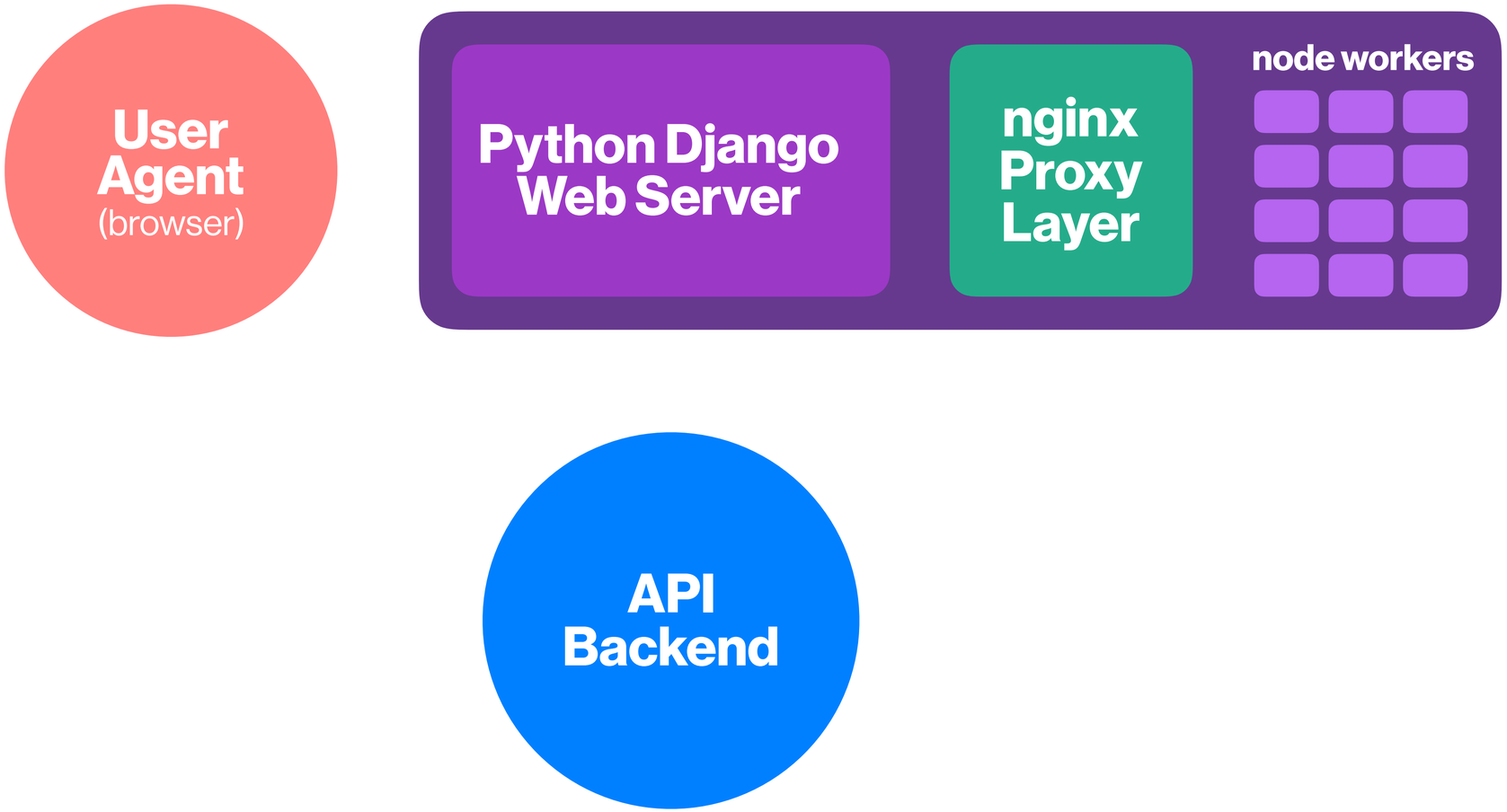


Python Jinja Renderer (before)

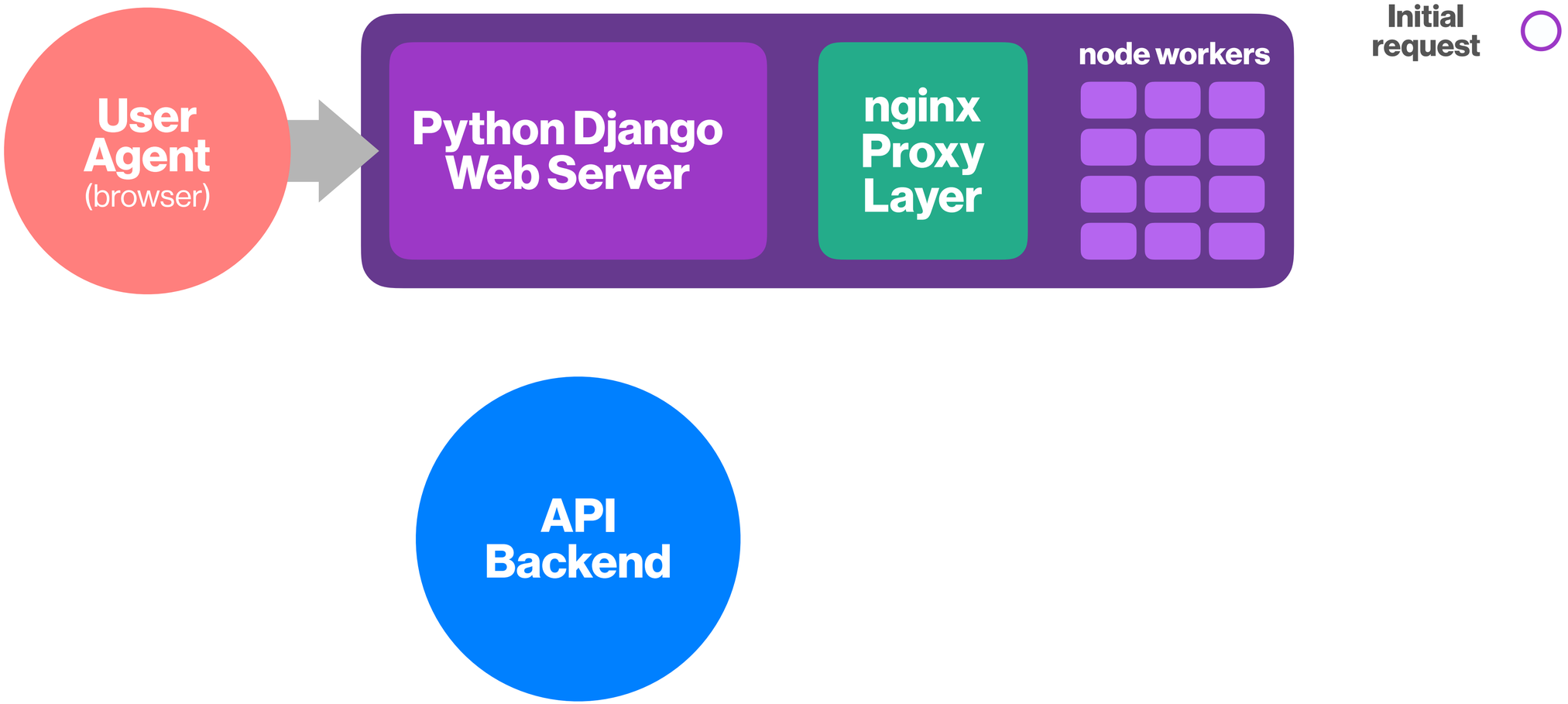




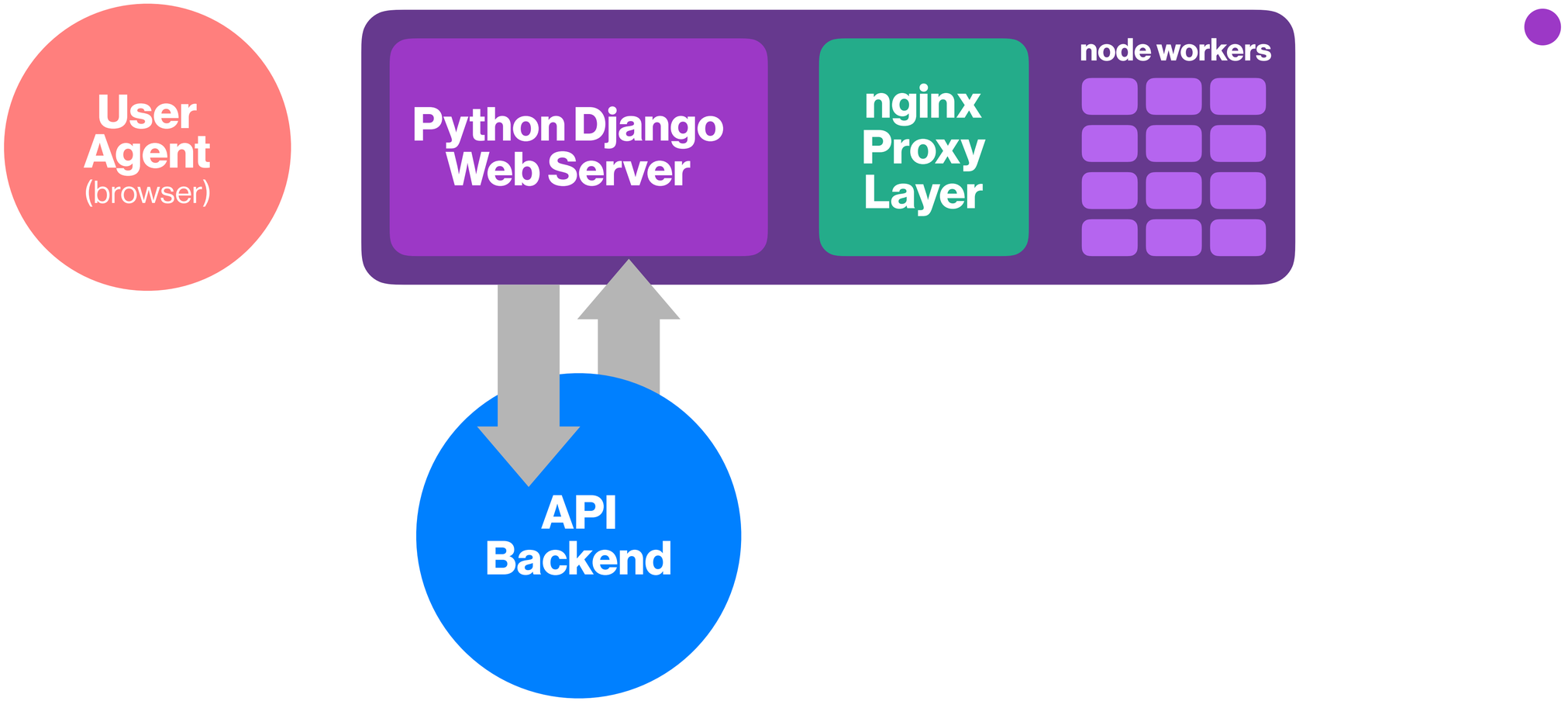
NodeJS Template Renderer (after)



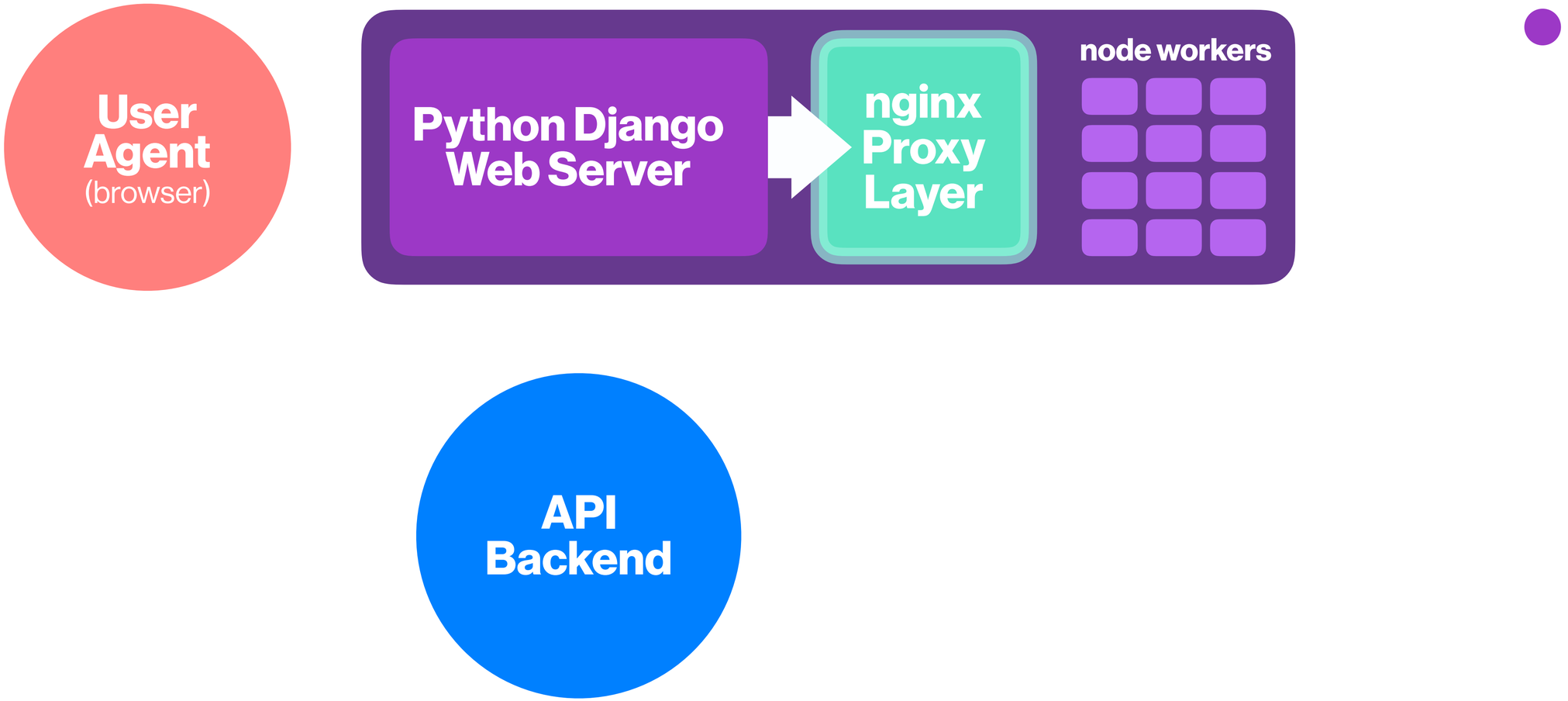
NodeJS Template Renderer (after)



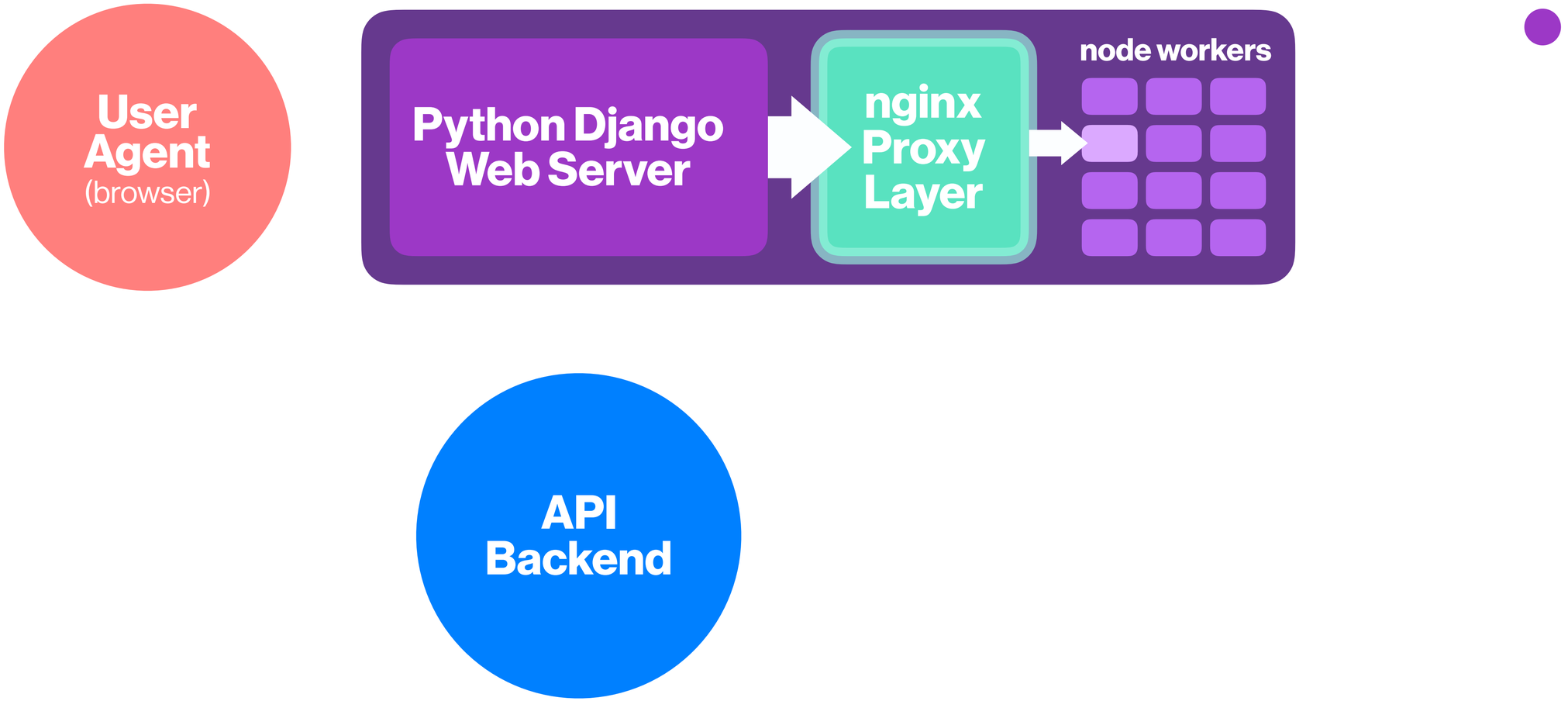
NodeJS Template Renderer (after)



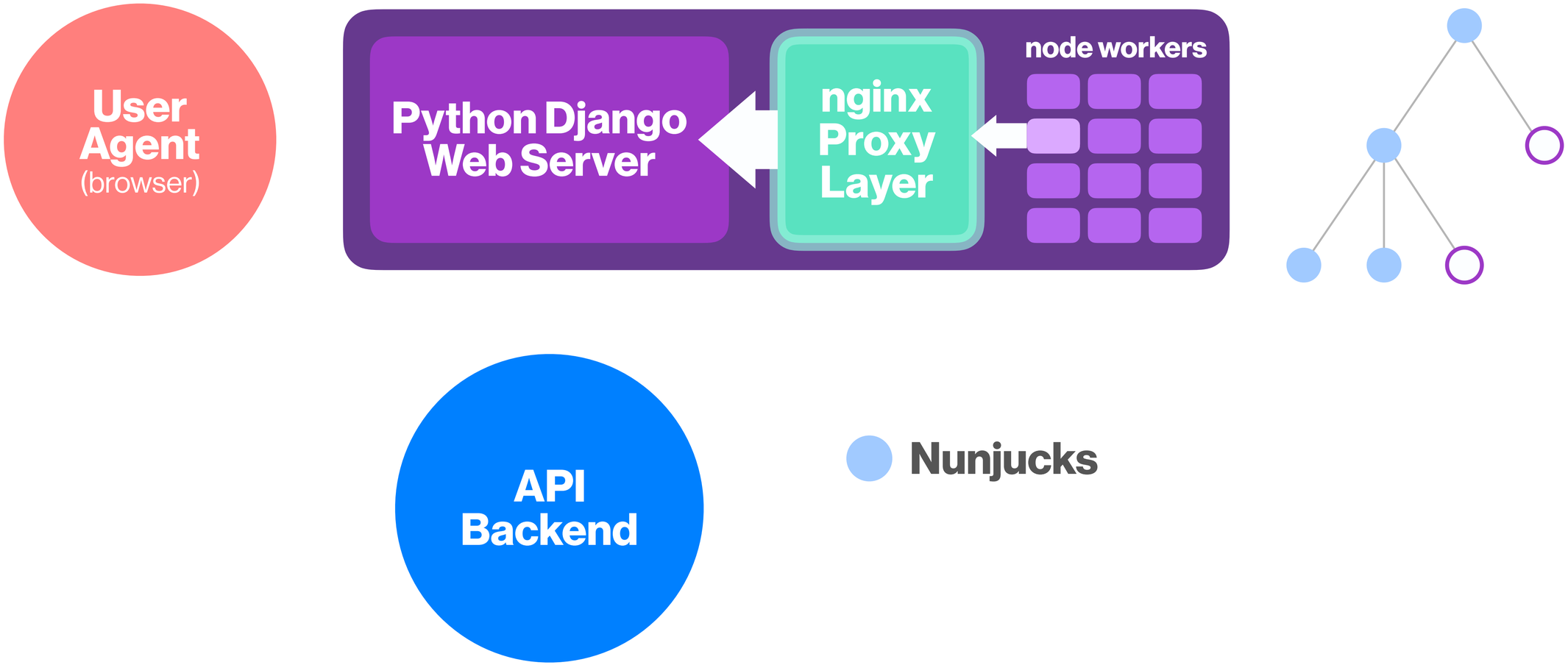
NodeJS Template Renderer (after)



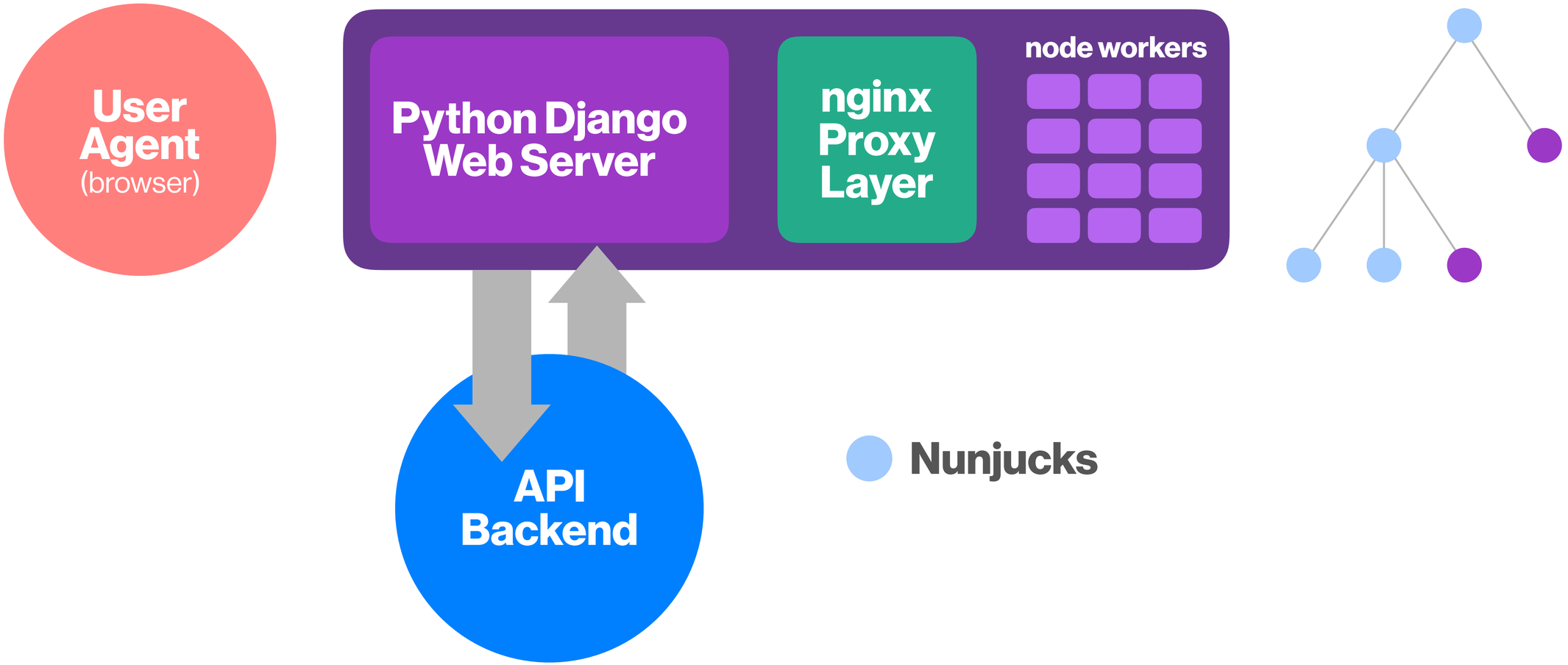
NodeJS Template Renderer (after)



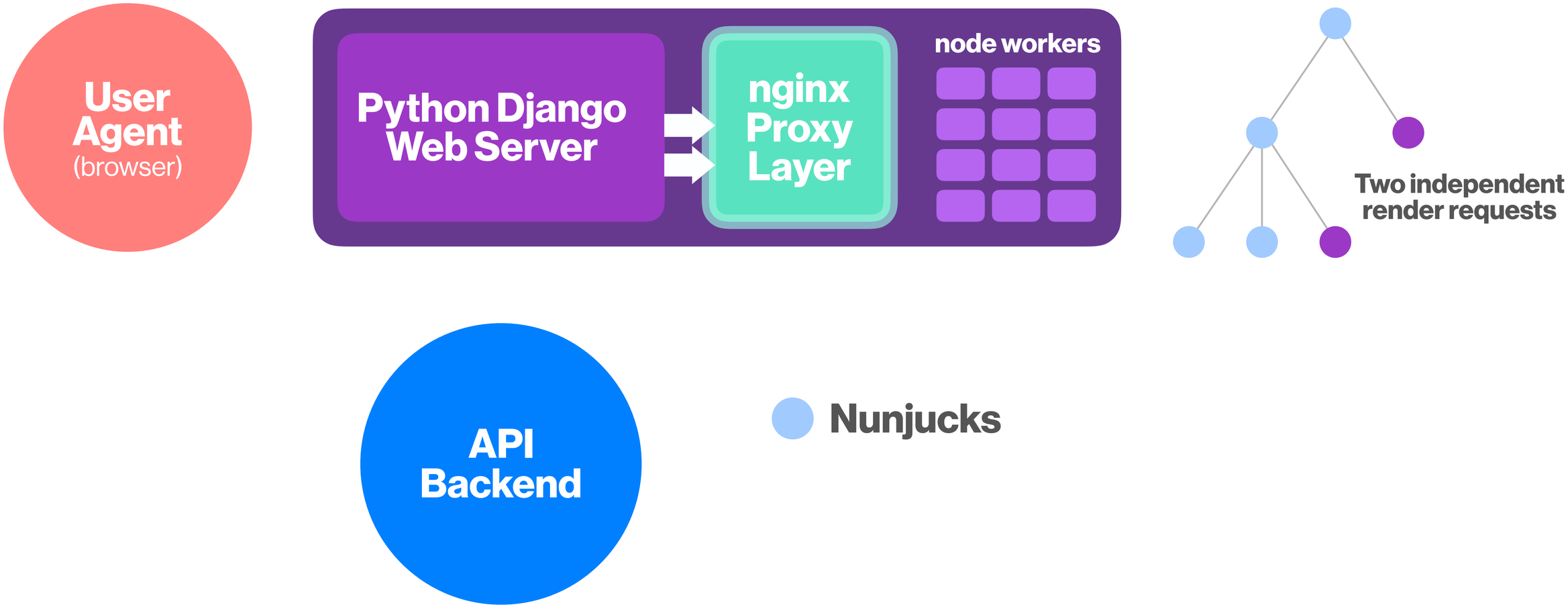
NodeJS Template Renderer (after)



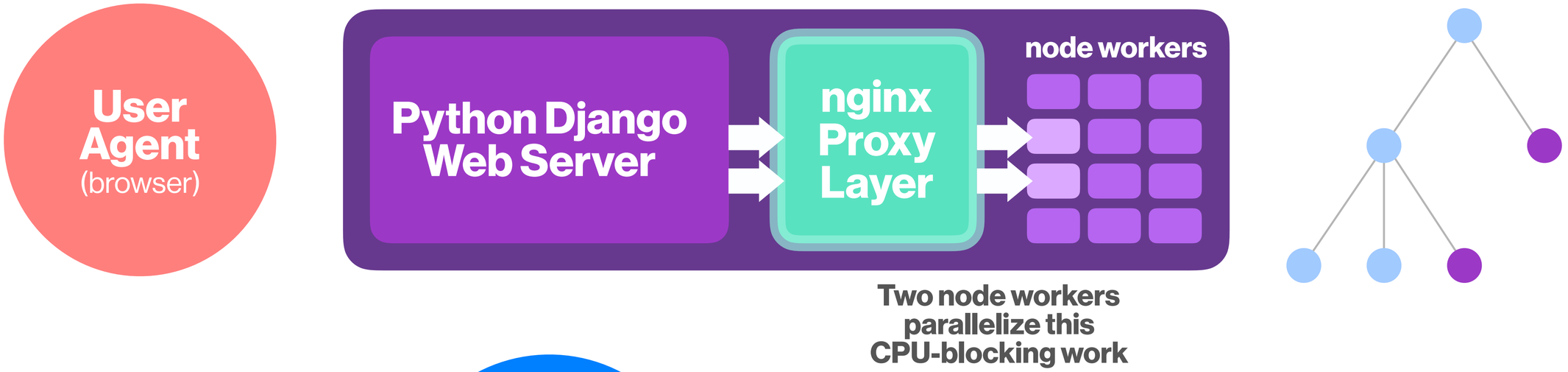
NodeJS Template Renderer (after)



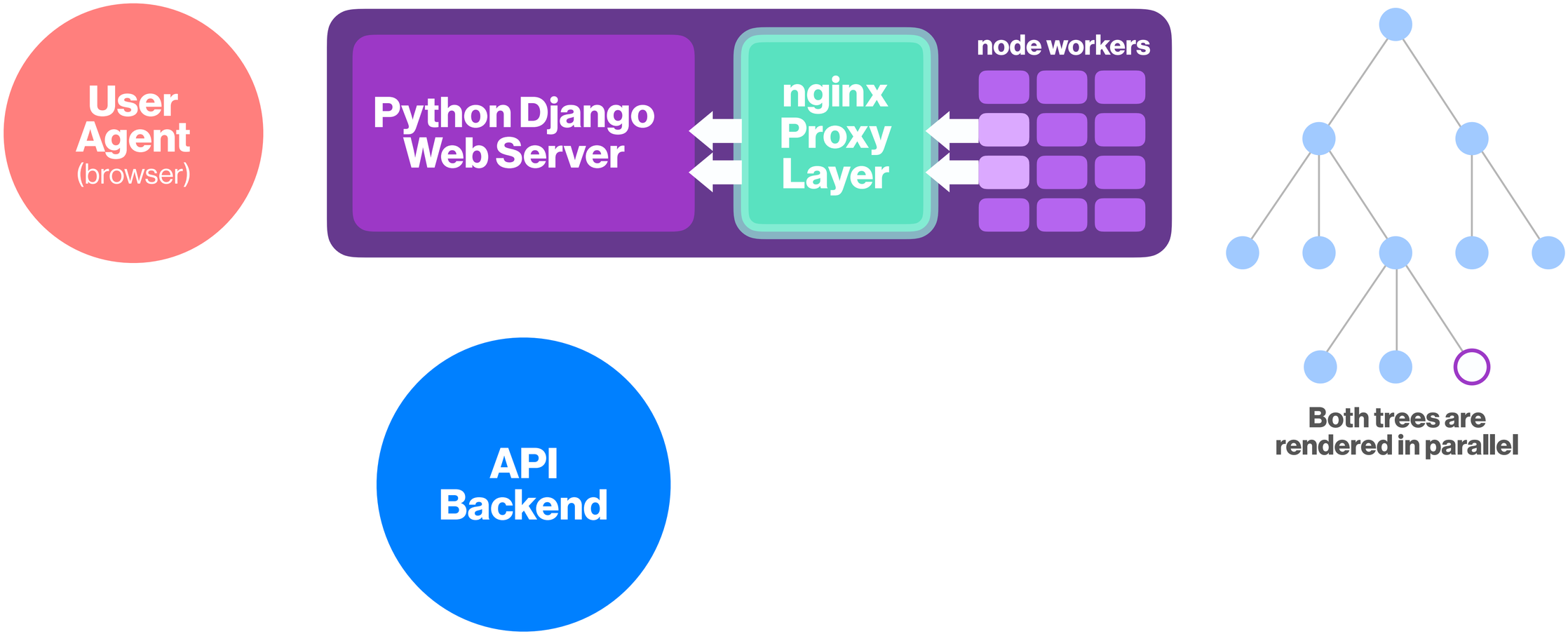
NodeJS Template Renderer (after)



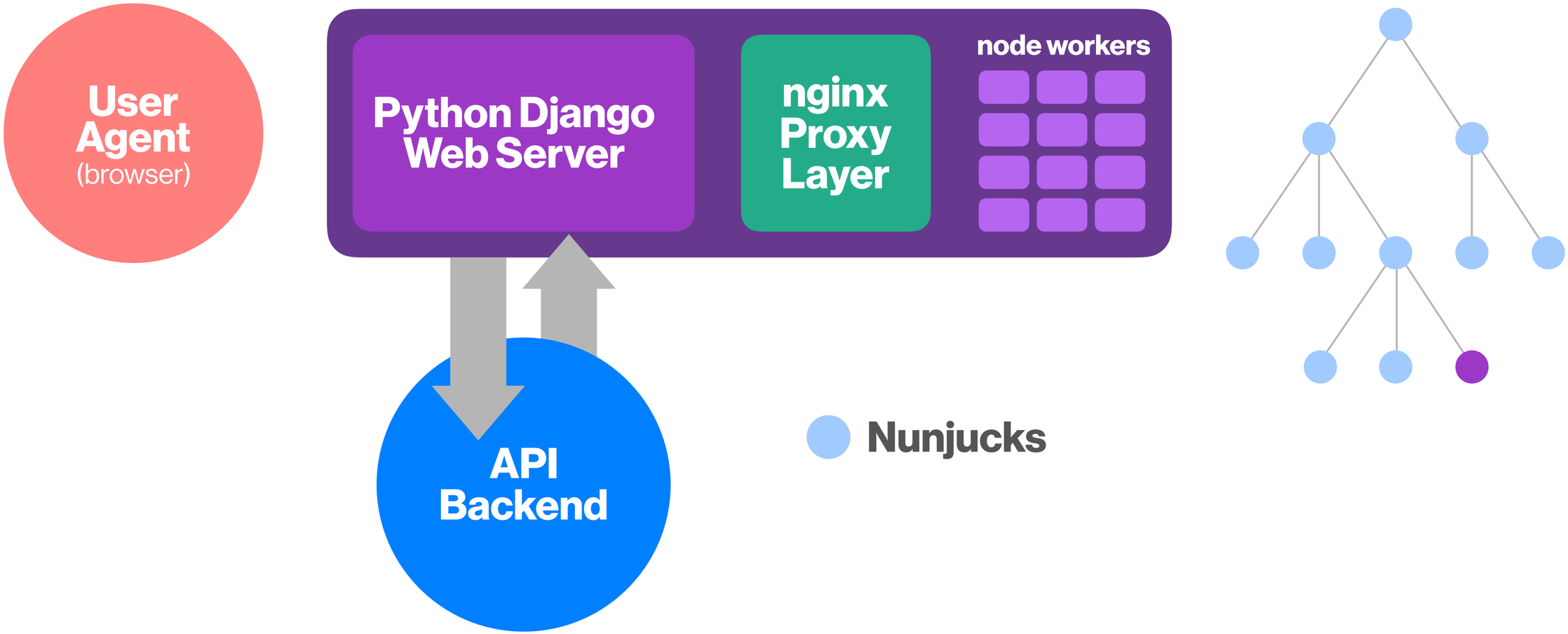
NodeJS Template Renderer (after)



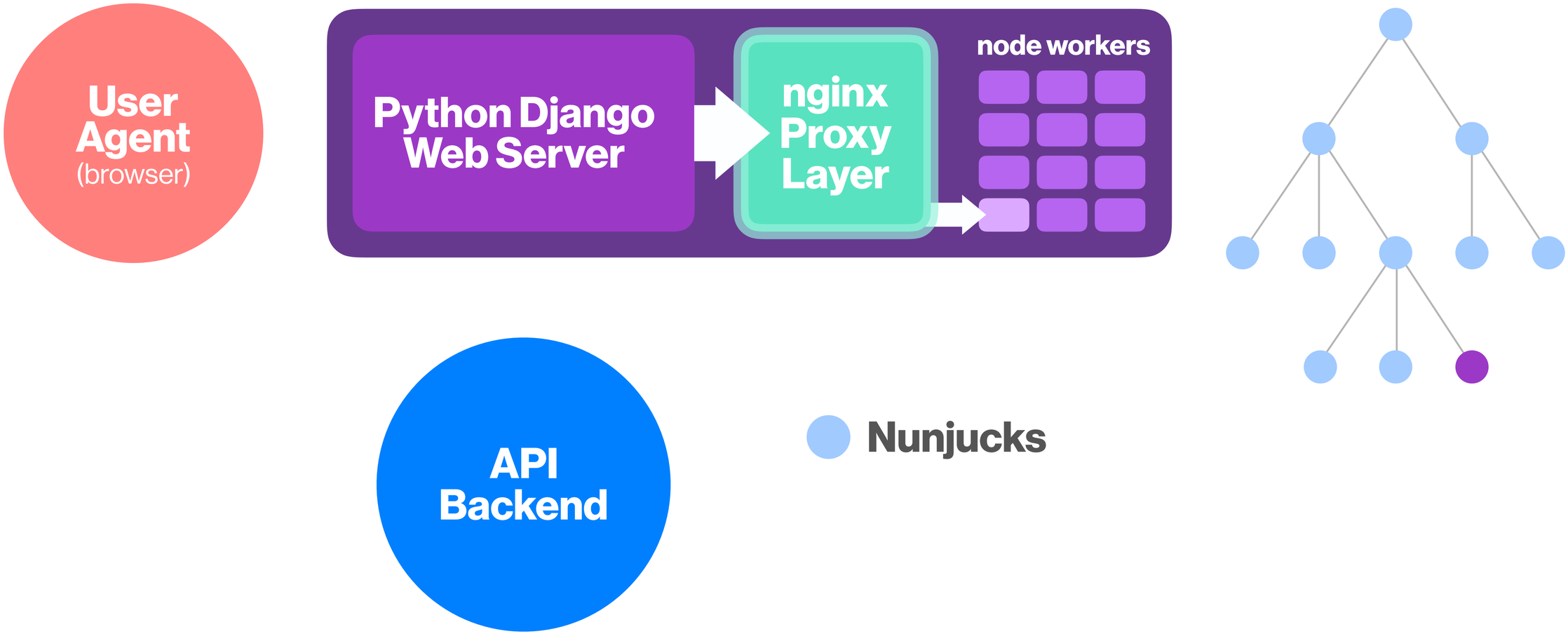
NodeJS Template Renderer (after)



NodeJS Template Renderer (after)

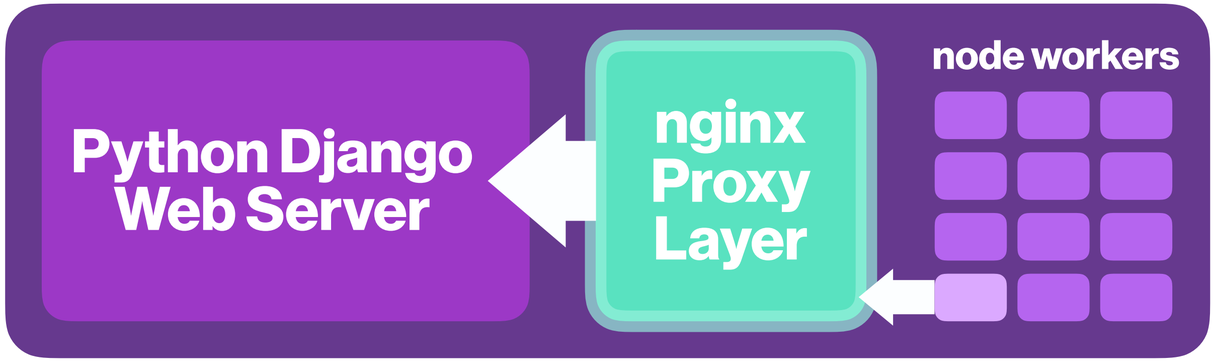


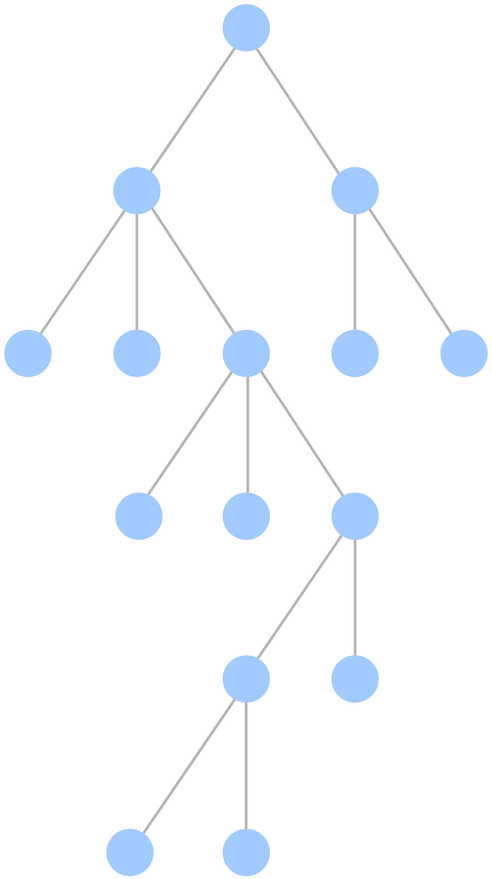
NodeJS Template Renderer (after)



NodeJS Template Renderer (after)

User Agent (browser)

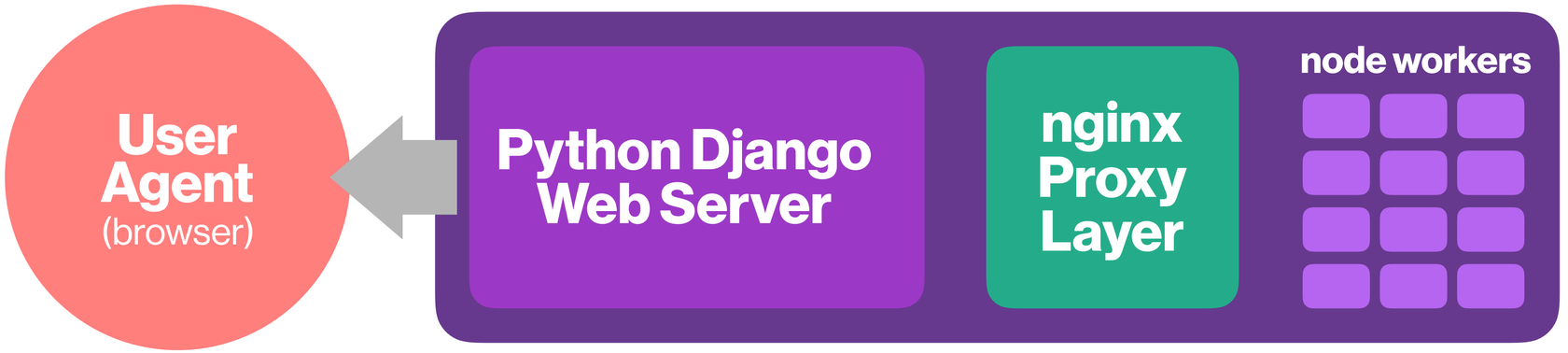


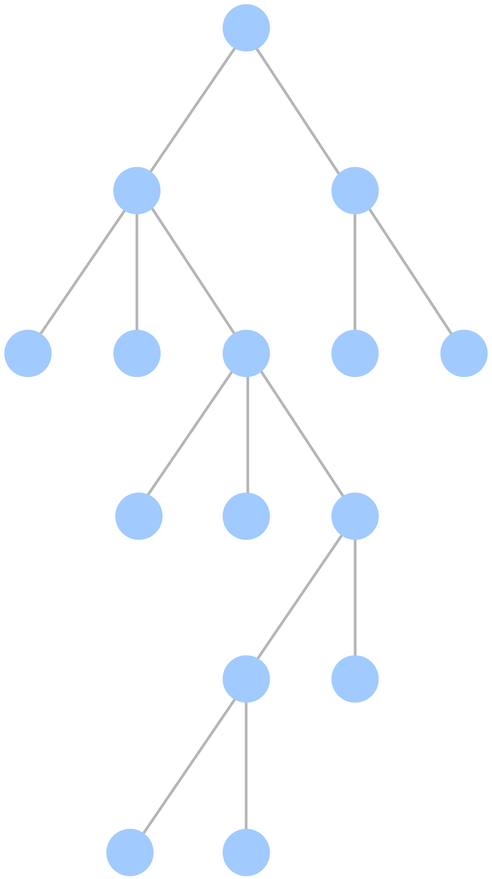


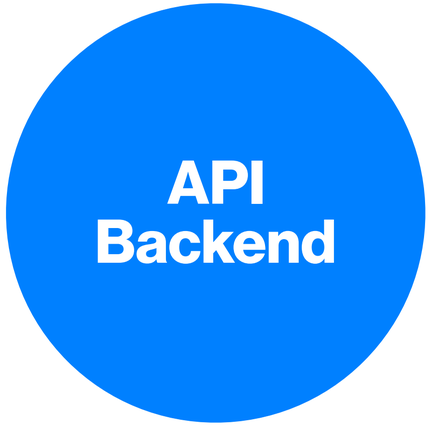
API Backend

Nunjucks

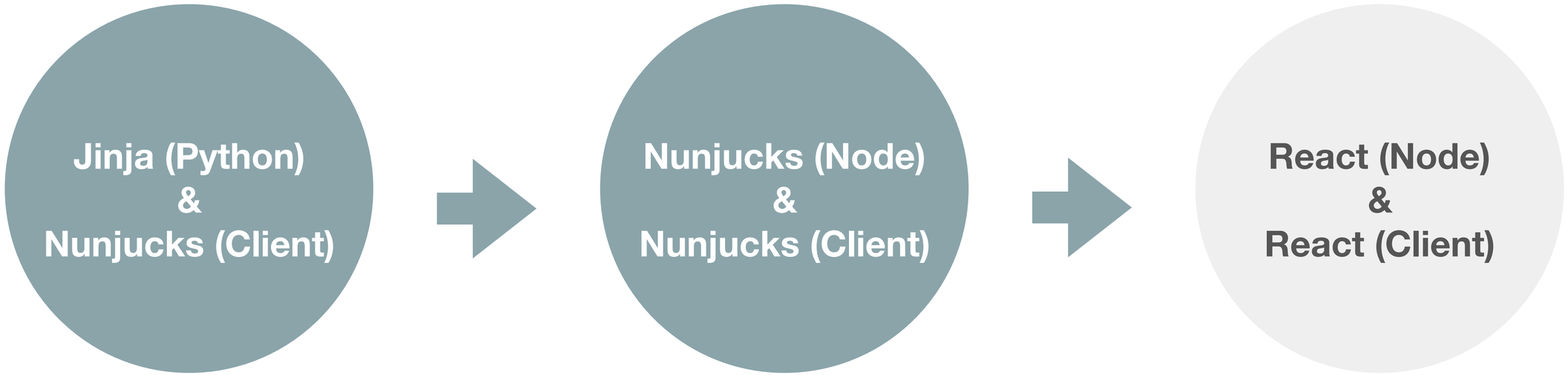
NodeJS Template Renderer (after)







Nunjucks


Jinja (Python)
&
Nunjucks (Client)
Nunjucks (Node)
&
Nunjucks (Client)
React (Node)
&
React (Client)

Denzel-React Bridge

```
// Denzel module in Nunjucks
{{ module(DenzelPin, {pinId: '123'}) }}

// React component in Nunjucks
{{ component(ReactPin, {pinId: '123'}) }}
```

Denzel-React Bridge

```
class DenzelReactBridge extends Module {
 onAddedToDom() {
  try {
   const { name, props } = this.options;
   // Mount React element to DOM
   render(getReactComponent(name, props), (this.el);
  } catch (error) {
   this.handleError(error);
  }
 }

 onBeforeEmpty() {
  // If empty() is called, we have to unmount before this.$el.empty()
  // Unmount React tree to trigger componentWillUnmount() lifecycle methods
  unmountComponentAtNode(this.el);
 }

 onRemovedFromDom() {
  // If empty() was called, React tree is already unmounted and unmounting
  // again does nothing. If it was not called, React tree is unmounted here.
  unmountComponentAtNode(this.el);
 }
}
```

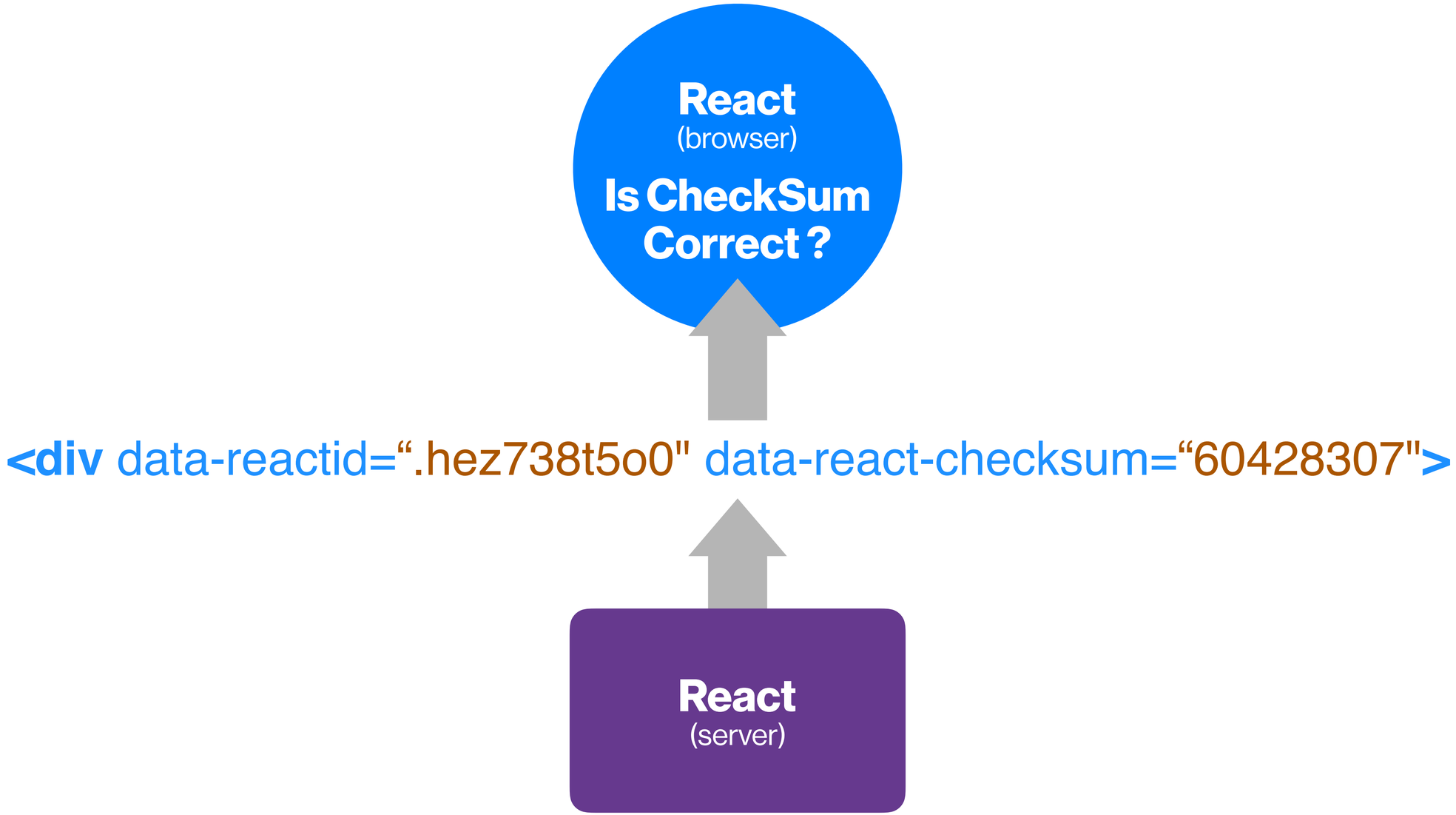
Denzel-React Bridge

```
class DenzelReactBridge extends Module {
 onAddedToDom() {
  try {
   const { name, props } = this.options;
   // Mount React element to DOM
   render(getReactComponent(name, props), (this.el);
  } catch (error) {
   this.handleError(error);
  }
 }

 onBeforeEmpty() {
  // If empty() is called, we have to unmount before this.$el.empty()
  // Unmount React tree to trigger componentWillUnmount() lifecycle methods
  unmountComponentAtNode(this.el);
 }

 onRemovedFromDom() {
  // If empty() was called, React tree is already unmounted and unmounting
  // again does nothing. If it was not called, React tree is unmounted here.
  unmountComponentAtNode(this.el);
 }
}
```

Denzel-React Bridge

```
class DenzelReactBridge extends Module {
 onAddedToDom() {
  try {
   const { name, props } = this.options;
   // Mount React element to DOM
   render(getReactComponent(name, props), (this.el);
  } catch (error) {
   this.handleError(error);
  }
 }

 onBeforeEmpty() {
  // If empty() is called, we have to unmount before this.$el.empty()
  // Unmount React tree to trigger componentWillUnmount() lifecycle methods
  unmountComponentAtNode(this.el);
 }

 onRemovedFromDom() {
  // If empty() was called, React tree is already unmounted and unmounting
  // again does nothing. If it was not called, React tree is unmounted here.
  unmountComponentAtNode(this.el);
 }
}
```

Reusing Server-Side HTML



Higher-Order Components as Adapters

```
const HOC = (InnerComponent) =>
 class extends Component {
  render() {
   return (
    <InnerComponent {...this.props} />
   );
  }
};
```

Adapters for old resources

```
import ResourceFactory from 'ResourceFactory';
// HOC withResource
export default withResource = ({ name, options }) => (Subject) => {
 return class extends Component {

  state = { data: null };

  componentDidMount() {
   const resource = ResourceFactory.create(name, options);

   resource.callGet().then((resourceResponse) => {
    this.setState({ data: resourceResponse.data });
   });
  }

  render() {
   return (
    <Subject data={this.state.data} {...this.props} />
   );
  }
 };
};
```

```
import withResource from 'withResource';
// Username component uses withResource HOC
class Username extends Component {
 render() {
  return (
   <div>{'Username:' + this.props.data.name}</div>
  );
 }
};

export default withResource({
 name: 'PinResource',
 options: props => {
  return { pin_id: props.pinId };
 },
})(Username);
```

Adapters for old resources

```
import ResourceFactory from 'ResourceFactory';

export default withResource = ({ name, options }) => (Subject) => {
 return class extends Component {

  state = { data: null };

  componentDidMount() {
   const resource = ResourceFactory.create(name, options);

   resource.callGet().then((resourceResponse) => {
    this.setState({ data: resourceResponse.data });
   });
  }

  render() {
   return (
    <Subject data={this.state.data} {...this.props} />
   );
  }
 };
};
```

```
import withResource from 'withResource';

class UsernameComponent extends Component {
 render() {
  return (
   <div>{'Username:' + this.props.data.name}</div>
  );
 }
};

export default withResource({
 name: 'PinResource',
 options: props => {
  return { pin_id: props.pinId };
 },
})(UsernameComponent);
```

Adapters for old resources

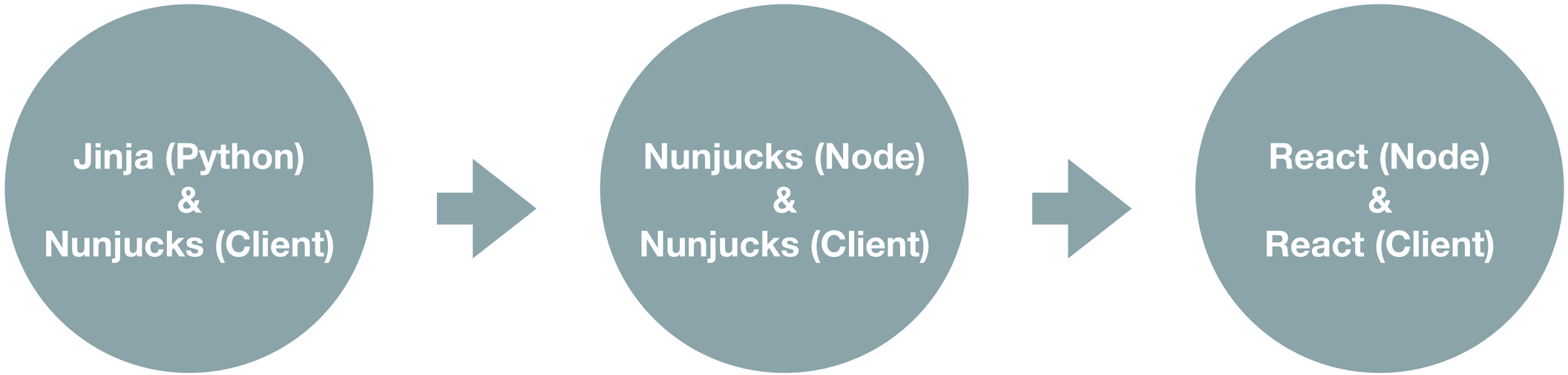
```
import ResourceFactory from 'ResourceFactory';

export default withResource = ({ name, options }) => (Subject) => {
 return class extends Component {

  state = { data: null };

  componentDidMount() {
   const resource = ResourceFactory.create(name, options);

   resource.callGet().then((resourceResponse) => {
    this.setState({ data: resourceResponse.data });
   });
  }

  render() {
   return (
    <Subject data={this.state.data} {...this.props} />
   );
  }
 };
};
```

```
import withResource from 'withResource';

class UsernameComponent extends Component {
 render() {
  return (
   <div>{'Username:' + this.props.data.name}</div>
  );
 }
};

export default withResource({
 name: 'PinResource',
 options: props => {
  return { pin_id: props.pinId };
 },
})(UsernameComponent);
```

Adapters for old resources

```
import ResourceFactory from 'ResourceFactory';

export default withResource = ({ name, options }) => (Subject) => {
 return class extends Component {

  state = { data: null };

  componentDidMount() {
   const resource = ResourceFactory.create(name, options);

   resource.callGet().then((resourceResponse) => {
    this.setState({ data: resourceResponse.data });
   });
  }

  render() {
   return (
    <Subject data={this.state.data} {...this.props} />
   );
  }
 };
};
```

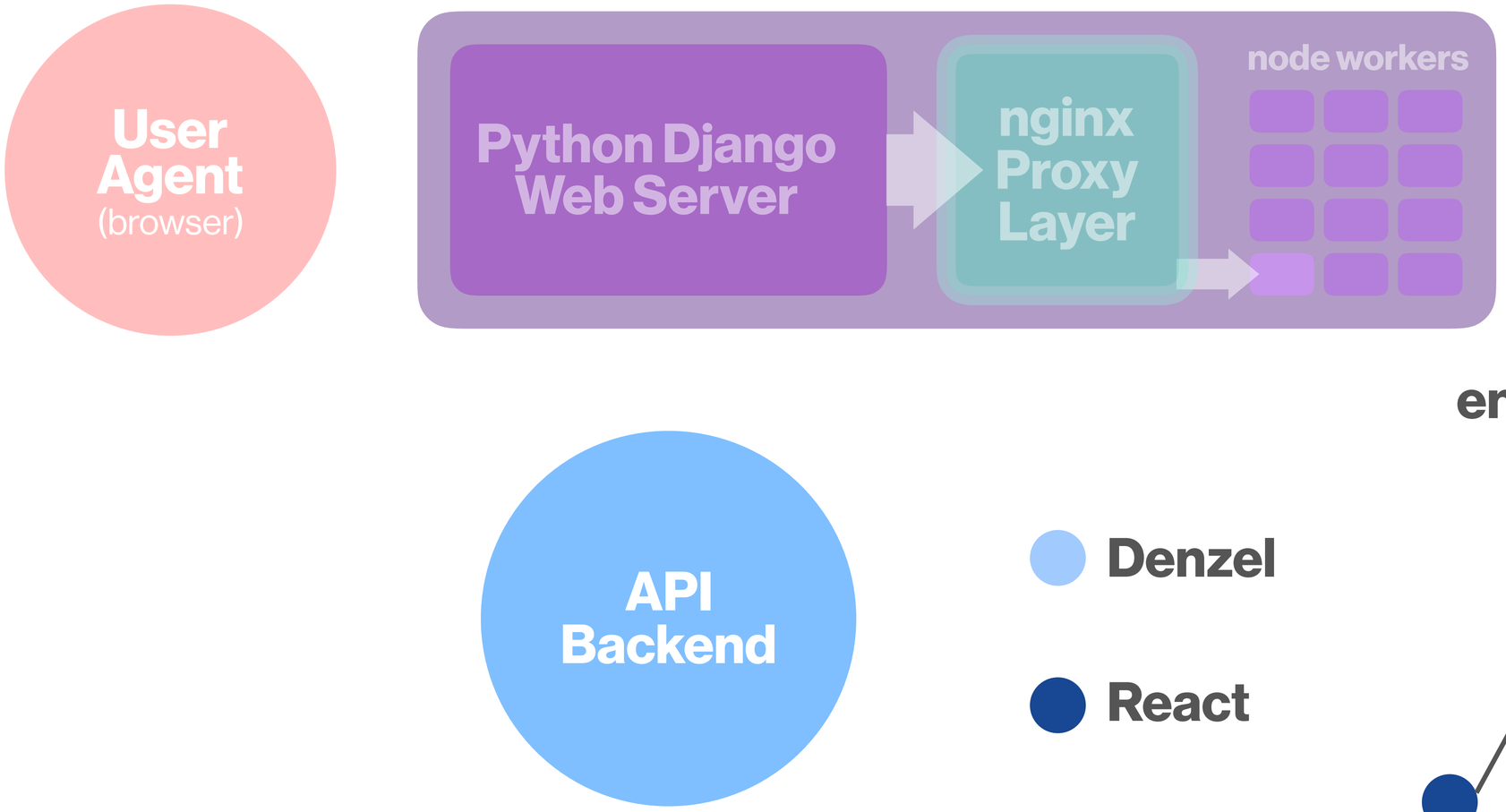
```
import withResource from 'withResource';

class UsernameComponent extends Component {
 render() {
  return (
   <div>{'Username:' + this.props.data.name}</div>
  );
 }
};

export default withResource({
 name: 'PinResource',
 options: props => {
  return { pin_id: props.pinId };
 },
})(UsernameComponent);
```

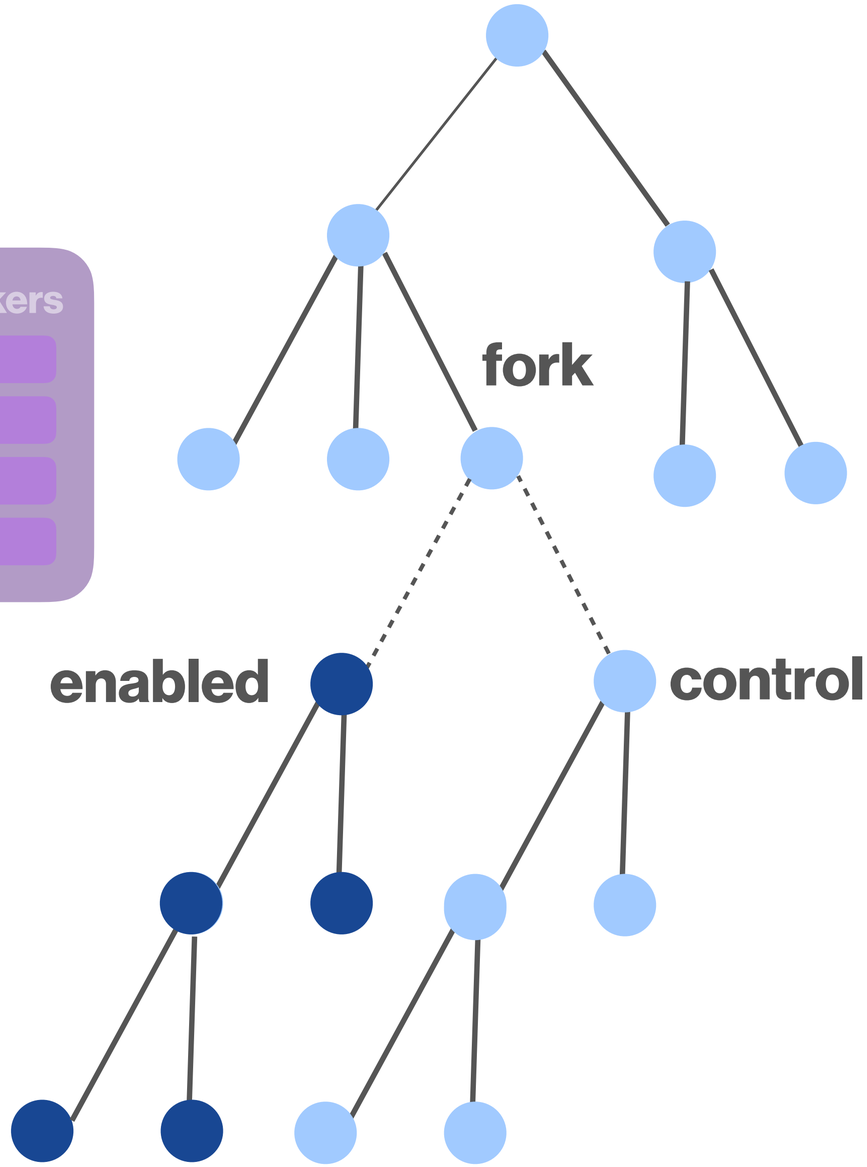

Jinja (Python)
&
Nunjucks (Client)
Nunjucks (Node)
&
Nunjucks (Client)
React (Node)
&
React (Client)

A/B testing Framework

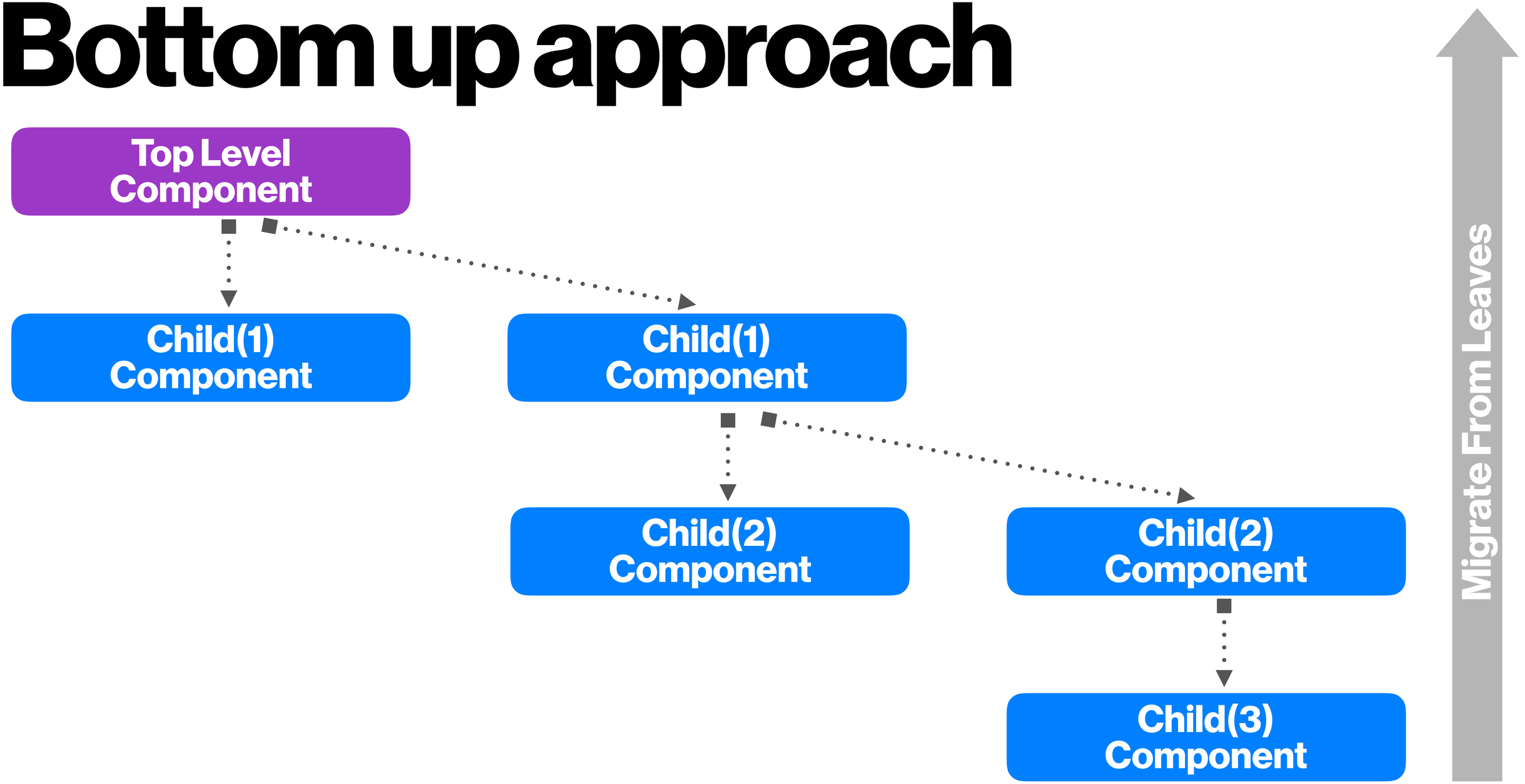
```
{% if context.inExperiment('react') %}
 {{ component(ReactPin, {pinId: '123'}) }}
{% else %}
 {{ module(DenzelPin, pinId='123') }}
{% endif %}
```

Experiment setup

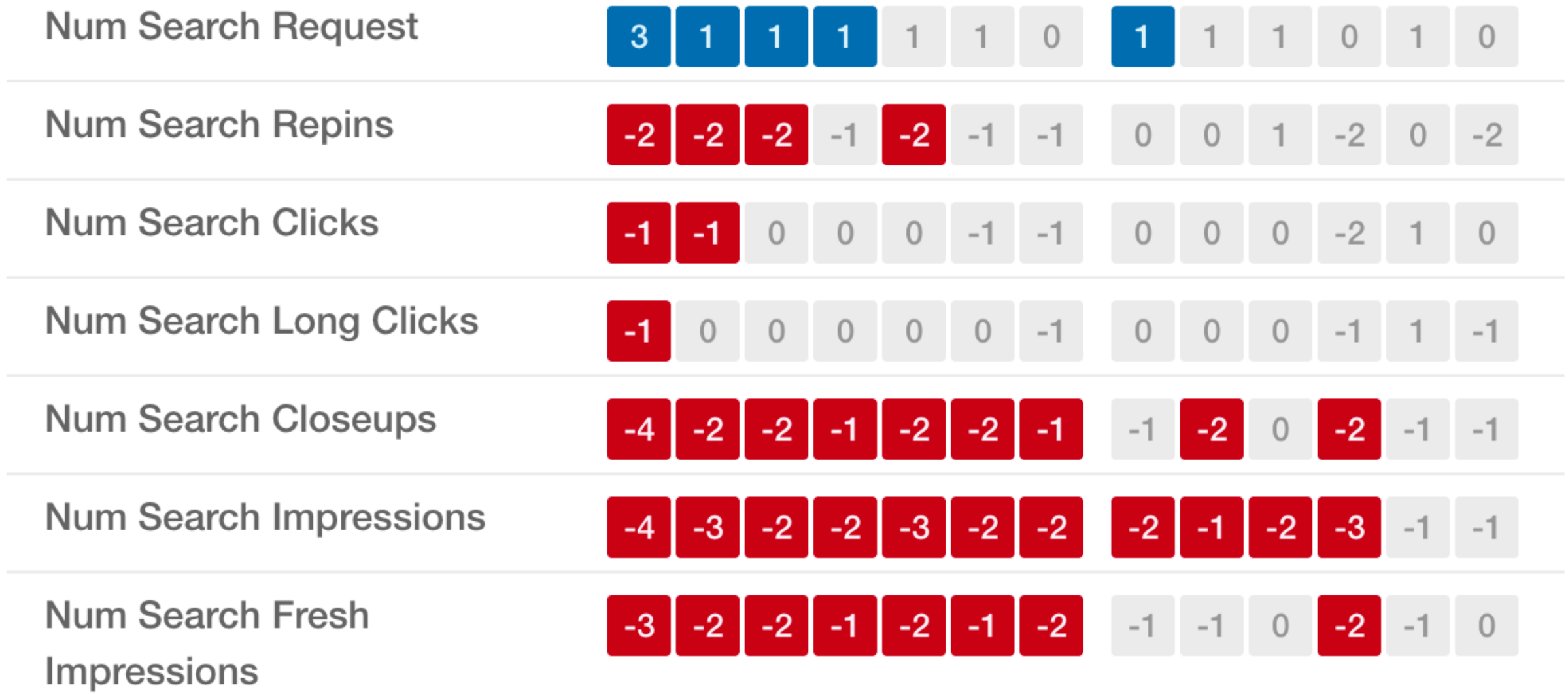




Bottom up approach



Experiment Dashboard

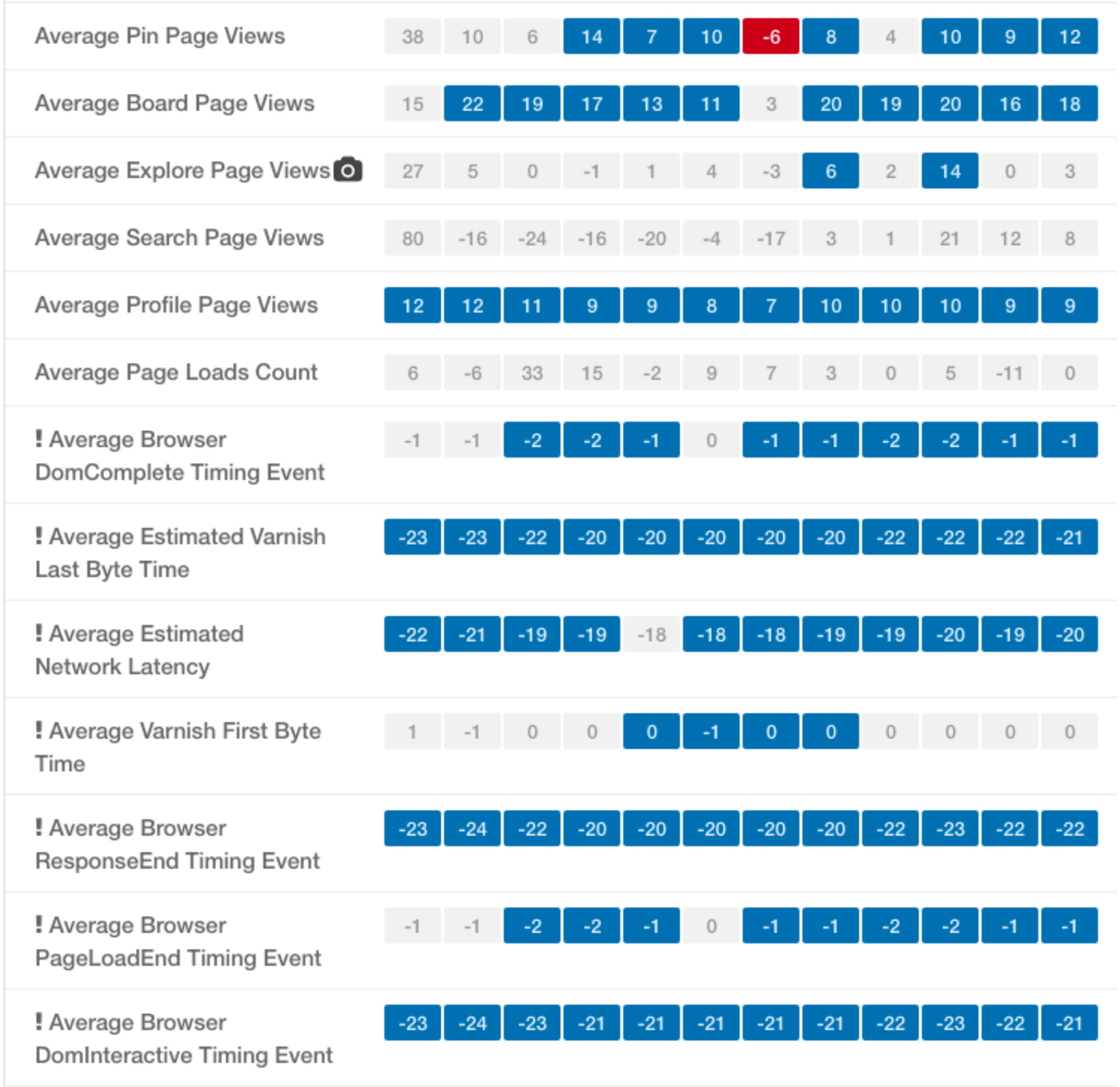


20%

domInteractive
responseEnd
boardPageViews

10%

pinPageViews
profilePageViews



Average Pin Page Views	38	10	6	14	7	10	-6	8	4	10	9	12
Average Board Page Views	15	22	19	17	13	11	3	20	19	20	16	18
Average Explore Page Views	27	5	0	-1	1	4	-3	6	2	14	0	3
Average Search Page Views	80	-16	-24	-16	-20	-4	-17	3	1	21	12	8
Average Profile Page Views	12	12	11	9	9	8	7	10	10	10	9	9
Average Page Loads Count	6	-6	33	15	-2	9	7	3	0	5	-11	0
! Average Browser DomComplete Timing Event	-1	-1	-2	-2	-1	0	-1	-1	-2	-2	-1	-1
! Average Estimated Varnish Last Byte Time	-23	-23	-22	-20	-20	-20	-20	-20	-22	-22	-22	-21
! Average Estimated Network Latency	-22	-21	-19	-19	-18	-18	-18	-19	-19	-20	-19	-20
! Average Varnish First Byte Time	1	-1	0	0	0	-1	0	0	0	0	0	0
! Average Browser ResponseEnd Timing Event	-23	-24	-22	-20	-20	-20	-20	-20	-22	-23	-22	-22
! Average Browser PageLoadEnd Timing Event	-1	-1	-2	-2	-1	0	-1	-1	-2	-2	-1	-1
! Average Browser DomInteractive Timing Event	-23	-24	-23	-21	-21	-21	-21	-21	-22	-23	-22	-21

Developer Experience

1. No Duplicating Nunjucks and Jinja libraries
2. No need to learn Nunjucks
3. Single language on Client and Server
4. Large developer community

imad@pinterest.com
github.com/eelyafi

Questions?