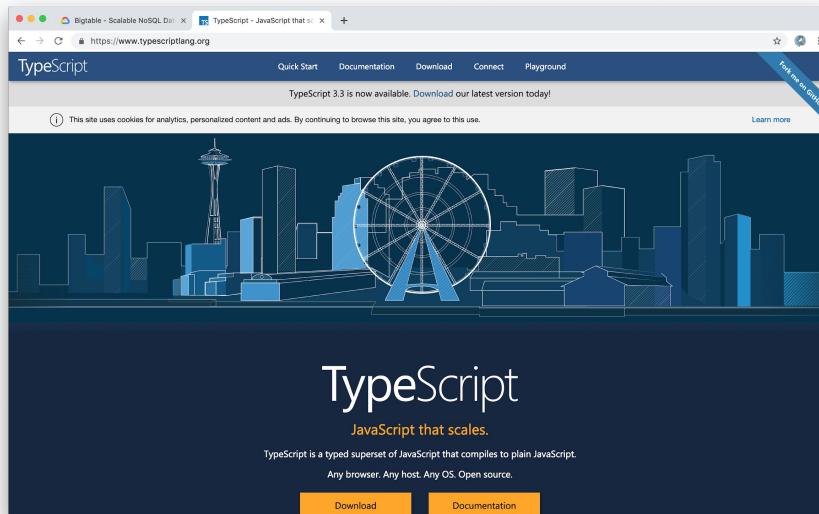


# How to scale

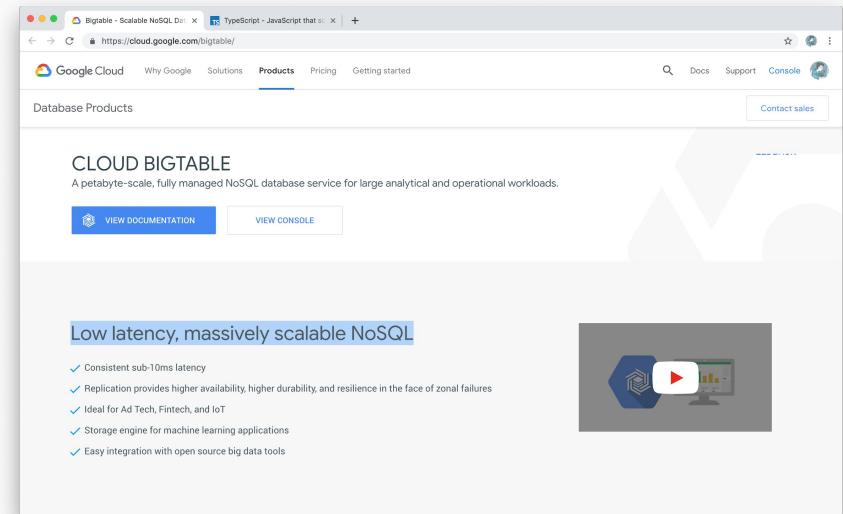
Presenter: Sam Zhou

# Overloaded meaning of scale

Scale for large codebase



Scale for large workload



# Overloaded meaning of scale

Scale for codebase:

Get the Bug-O right.   $(n!)$

Scale for workload:

Get the Big-O right.  $O(n)$

<https://overreacted.io/the-bug-o-notation/>

# Scaling your codebase: the Bug-O notation

Not formally defined like Big-O

A sketchy measure of the complexity of code

We usually want  $\mathcal{O}(1)$  or  $\mathcal{O}(\lg n)$  so things can be under control

Optimize Bug-O: better abstractions and tools

Optimize for change

# Bad

“I hope our IDs won’t collide.”

“Is my UI synced with data?”

Direct DOM manipulation

Can be easily messed up by other components

↖(n), n = # of components in the app

```
29     function nestedListElement(parent) {
30
31         var ul = document.createElement("ul");
32         var li = document.createElement("li");
33
34         li.appendChild(document.createTextNode("Placeholder Text"));
35         li.setAttribute("id", "ToDoList");
36         li.setAttribute("class", "sortable-element");
37         li.setAttribute("contenteditable", true);
38
39
40         var span = document.createElement("SPAN");
41         var txt = document.createTextNode("\u2713");
42         span.className = "close";
43         span.setAttribute("contenteditable", false);
44         span.appendChild(txt);
45         li.appendChild(span);
46
47
48         var span2 = document.createElement("span");
49         txt2 = document.createTextNode("\u2195 \u00a0 \u00a0");
50         span2.className = "handle";
51         span2.setAttribute("contenteditable", false);
52         span2.appendChild(txt2);
53         li.insertBefore(span2, li.firstChild);
54
55         var span3 = document.createElement("span3");
56         txt3 = document.createTextNode("\u2719");
57         span3.className = "hider";
58         span3.setAttribute("contenteditable", false);
59         span3.setAttribute("data-toggle", "collapse");
60         span3.appendChild(txt3);
61         li.appendChild(span3);
62
63         ul.appendChild(li);
64         ul.setAttribute("class", "sortable");
65         ul.setAttribute("contenteditable", false);
66         parent.appendChild(ul);
67     }
```

# Good

Use React Component to control complexity

Modular Code

Problem: What if we need to refactor?

✳ (C\*n)

n = # of component that uses this

C = some not-very-small constant

```
67 render() {
68   return (
69     <form className={styles.NewTaskWrap} onSubmit={this.handleSave}>
70       <input value={this.state.name} onChange={this.handleTaskNameChange} type="text" className={styles.NewTaskComponent} pl
71       <div className={styles.NewTaskActive}>
72
73         <div className={styles.NewTaskClass}>
74           <input id="changeClassCheckbox" type="checkbox" ref={this.openClassChange} />
75           <label htmlFor="changeClassCheckbox" style={{ backgroundColor: this.props.tagColorPicker[this.state.tag] }} ref={th
76             <ul>
77               {Object.keys(this.props.tagColorPicker).map(cTitle => <NewTaskClassPicker key={cTitle} classColor={this.props.tag
78             </ul>
79           </div>
80
81           <div className={styles.NewTaskDate}>
82             <label htmlFor="changeDateCheckbox">📅 </label>
83             <input id="changeDateCheckbox" type="checkbox" ref={this.openDateChange} />
84             <div className={styles.NewTaskDatePicker}>
85               <Calendar
86                 onChange={this.handleDateChange}
87                 value={new Date()}>
88
89               </>
90             </div>
91           </div>
92
93           </div>
94         </form>
95       );//<input type="date" value={this.state.date} onChange={this.handleDateChange} min={new Date(new Date().getTime() - new Da
96     }
97   }
98 }
```

# Less Good

React Ref is harder to reason

Avoid it unless absolutely need it

```
15  constructor(props) {  
16    super(props);  
17    this.state = this.initialState();  
18    this.changeClass = React.createRef();  
19    this.addTask = React.createRef();  
20    this.openClassChange = React.createRef();  
21    this.openDateChange = React.createRef();  
22 }
```

# Better

Runtime type-checking components

Self-documenting

Still need good test coverage to ensure there is no prop-type errors

✿ (c\*n),

n = # of component that uses this

c = a small constant

```
35 renderCourses() {
36   if (this.props.query !== "") {
37     return this.props.allCourses.slice(0,100).map((course) => (
38       //create a new class "button" that will set the selected class to this class when it is clicked.
39       <Course key={course._id} info={course} query={this.props.query}/>
40     ));
41   }
42   else {
43     return <div />;
44   }
45 }

46 showDropdown = () => {
47   this.setState({showDropdown: true})
48 }

49 hideDropdown = () => {
50   this.setState({showDropdown: false})
51 }

52 render() {
53   return (
54     <div className="search-bar text-left" id="searchbar" >
55       <input className="search-text" id="search" onChange={this.props.queryFunc} placeholder="Search for classes (e.g. CS 211"
56         <ul id="output" style={this.state.showDropdown ? {} : { display: 'none' }}>
57           {this.renderCourses()}
58         </ul>
59       </div>
60     );
61   }
62 }

63 );
64 }
65 }

66 // SearchBar requires the user's query from the parent component, a function
67 // to call when the query changes so the parent can update its copy of the query,
68 // and a list of all courses that satisfy the query.
69 SearchBar.propTypes = {
70   allCourses: PropTypes.array.isRequired,
71   loading: PropTypes.bool, // optional
72   query: PropTypes.string.isRequired,
73   queryFunc: PropTypes.func.isRequired
74 };
75 }
```

# Even Better

Type-annotate the components

Type-checkers ensure you use the component correctly at compile-time

💡(1)

Your prop types can never go wrong once you pass the type checker

```
17
18 type DefaultProps = {
19   +className?: string;
20   children?: Node;
21   +newSubTaskDisabled?: boolean;
22   +onFocus?: (event: SyntheticFocusEvent<HTMLElement>) => void;
23   +onBlur?: (event: SyntheticFocusEvent<HTMLElement>) => void;
24   +refFunction?: (HTMLElement | null) => void; // used to get the DO
25 };
26 type Props = {
27   ...Task; // The task given to the editor at this point.
28   ...DefaultProps; // Props with default values.
29   // actions
30   +editMainTask: (partialMainTask: PartialMainTask, doSave: boolean)
31   +editSubTask: (subtaskId: string, partialSubTask: PartialSubTask,
32   +addSubTask: (subTask: SubTask) => void;
33   +removeTask: () => void;
34   +removeSubTask: (subtaskId: string) => void;
35   +onSave: () => void;
36   // subscribed from redux store.
37   +getTag: (id: string) => Tag;
38 };
39
40 type State = {
41   +mainTaskNameCache: string | null;
42   +oneSubTaskNameCache: [string, string] | null;
43   +doesShowTagEditor: boolean;
44   +doesShowDateEditor: boolean;
45   +needToSwitchFocus: number | null; // number: expected array lengt
46 };
47
48 /**
49 * The component of an standalone task editor.
50 * It is designed to be wrapped inside another component to extend i
51 * editor itself does not remember the state of editing a task, a wr
52 * You can read the docs for props above.
53 */
54 class TaskEditor extends React.Component<Props, State> {
55   state: State = {
56     mainTaskNameCache: null,
57     oneSubTaskNameCache: null,
58   };
59
60   /**
61    * The task editor used to edit task inline, activated on
62    */
63   export default function InlineTaskEditor({ task, className
64   const [disabled, setDisabled] = React.useState(true);
65
66   const { id } = task;
67   // To un-mount the editor when finished editing.
68   const onFocus = () => setDisabled(false);
69   const onBlur = () => setDisabled(true);
70   const taskEditorProps = {
71     ...task,
72     editMainTask: (partialMainTask: PartialMainTask, onSave: boolean) => {
73       editMainTask(id, partialMainTask);
74       if (onSave) {
75         onBlur();
76       }
77     },
78     editSubTask: (subtaskId: string, partialSubTask: PartialSubTask) => {
79       editSubTask(subtaskId, partialSubTask);
80       if (onSave) {
81         onBlur();
82       }
83     },
84     addSubTask: (subTask: SubTask) => addSubTask(id, subTask);
85     removeTask: () => removeTask(task);
86     removeSubTask: (subtaskId: string) => removeSubTask(subtaskId);
87     onSave: onBlur,
88     className,
89     newSubTaskDisabled: disabled || !task.inFocus,
90     onFocus,
91     onBlur,
92   };
93
94   return <TaskEditor {...taskEditorProps} />;
95 }
```

# Scaling your workload

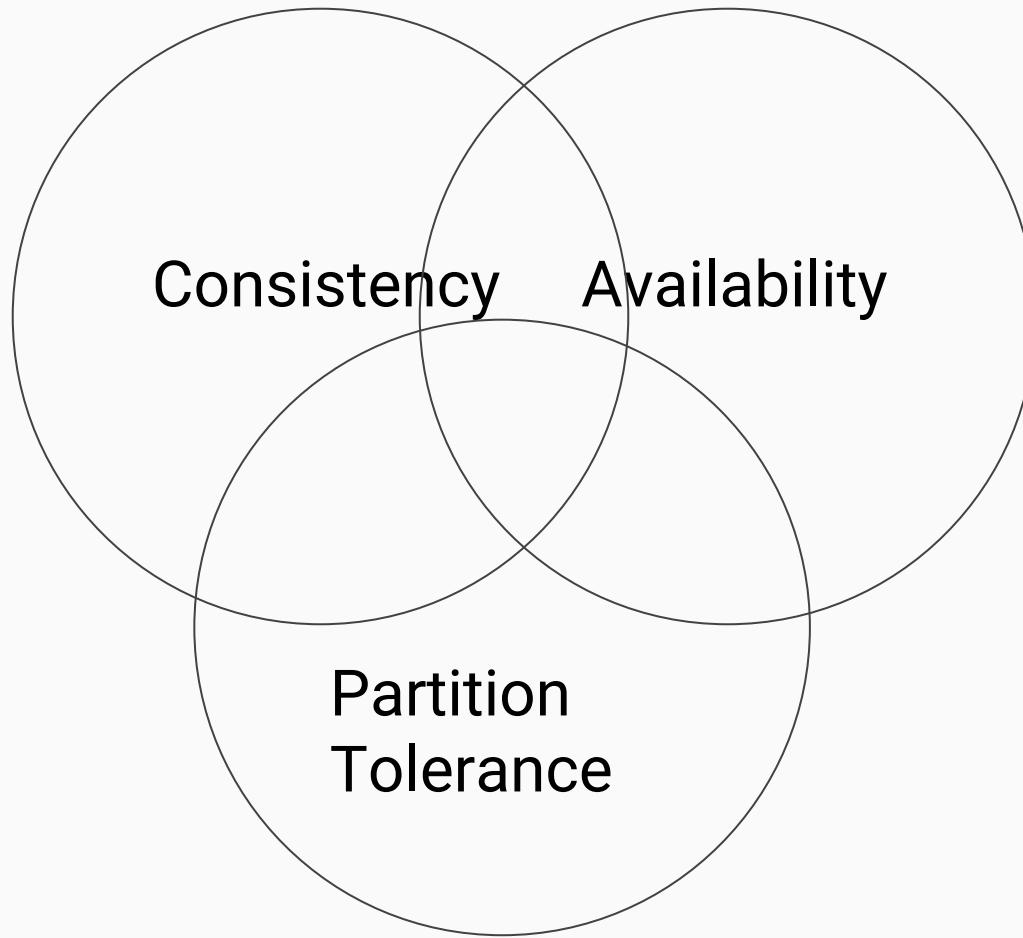
Loosely coupled & independently scalable

Relational vs NoSQL Database, CAP Theorem

On the cloud: managed solutions

### lb-ingress.yaml X

```
1  apiVersion: extensions/v1beta1
2  kind: Ingress
3  metadata:
4    name: lb-ingress
5  annotations:
6    kubernetes.io/ingress.global-static-ip-name: "web-static-ip"
7  spec:
8    rules:
9      - http:
10        paths:
11          - path: /apis/*
12            backend:
13              serviceName: backend-workload
14              servicePort: 8080
15            - backend:
16              serviceName: frontend-workload
17              servicePort: 8080
18
```



Firebase helps mobile app teams succeed.

Build apps fast, without managing infrastructure

Backed by Google, trusted by top apps

One platform, with products that work better together

GET STARTED WATCH THE VIDEO

Firebase gives you functionality like analytics, databases, messaging and crash reporting so you can move quickly and focus on your users.

Firebase is built on Google infrastructure and scales automatically, even for the largest apps.

Firebase products work great individually but share data and insights, so they work even better together.

## Functions

Build apps faster with a serverless architecture

Accelerate your development with an event-driven, serverless compute experience. Scale on demand and pay only for the resources you consume.

Start free >

Sign in to your account >

Explore Azure Functions: Pricing details Documentation Updates Community Serverless computing

Take advantage of serverless compute with Functions

Easily build the apps you need using simple, serverless functions that scale to meet demand. Use the programming language of your choice, and don't worry about servers or infrastructure.

## Google Cloud Functions

Event-driven serverless compute platform

VIEW CONSOLE VIEW DOCUMENTATION

- Simplest way to run your code in the cloud
- Automatically scales, highly available and fault tolerant
- No servers to provision, manage, patch or update
- Pay only while your code runs
- Connects and extends cloud services

## AWS Lambda

Run code without thinking about servers. Pay only for the compute time you consume.

Get started with AWS Lambda

TECH TALK Best Practices for Serverless Queue Processing - February 19

Learn the best practices of serverless queue processing, using Amazon SQS as an event source for AWS Lambda.

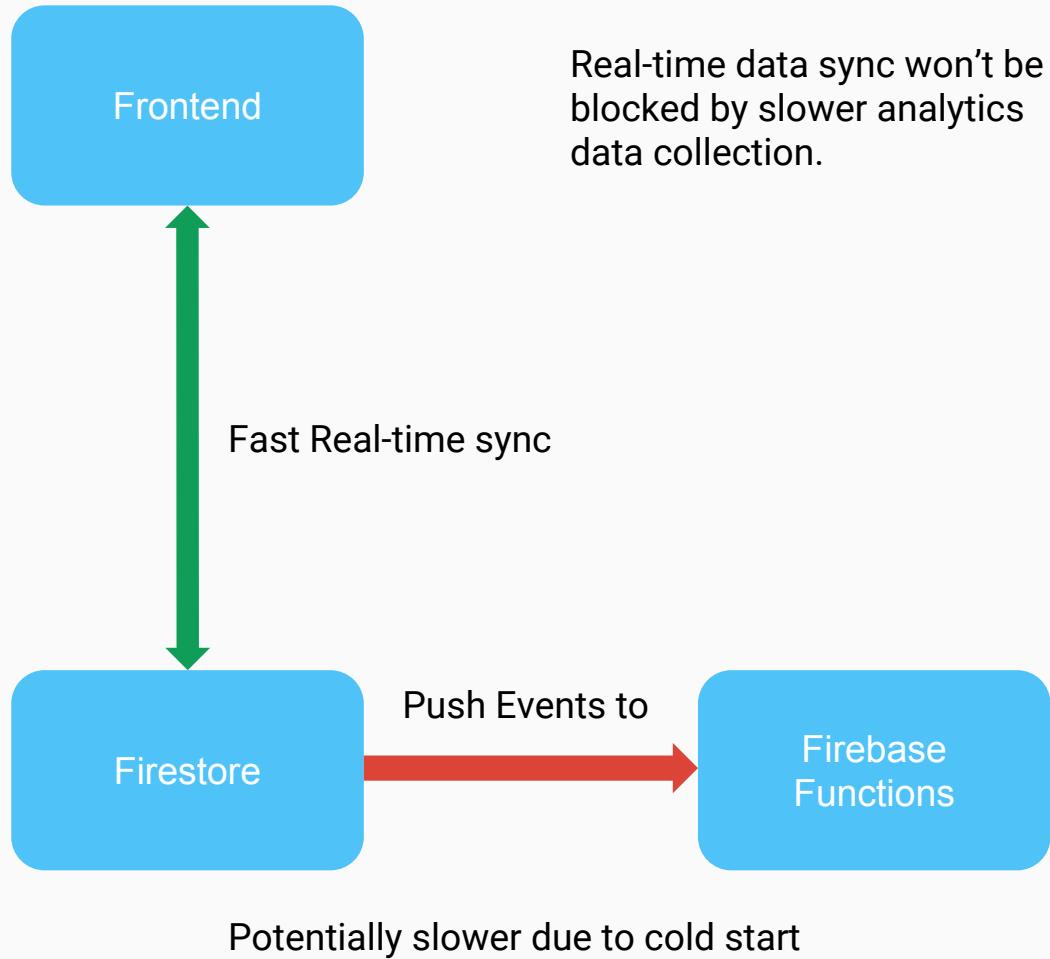
Register now >

AWS Lambda lets you run code without provisioning or managing servers. You pay only for the compute time you consume - there is no charge when your code is not running.

With Lambda, you can run code for virtually any type of application or backend service - all with zero administration. Just upload your code and Lambda takes care of everything required to run and scale your code with high availability. You can set up your code to automatically trigger from other AWS services or call it directly from any web or mobile app.

# Case Study

Samwise's Real-time Interactions & Analytics



However ... Scaling is easy!

kelseyhightower/nocode: The best way to write secure and reliable applications. Write nothing; deploy nowhere.

[Pull requests](#) [Issues](#) [Marketplace](#) [Explore](#)

**kelseyhightower / nocode**

[Code](#) [Issues 2,187](#) [Pull requests 354](#) [Projects 0](#) [Wiki](#) [Insights](#)

[Watch](#) 261 [Unstar](#) 26,487 [Fork](#) 2,265

The best way to write secure and reliable applications. Write nothing; deploy nowhere.

3 commits 1 branch 1 release 1 contributor Apache-2.0

Branch: master [New pull request](#) [Create new file](#) [Upload files](#) [Find file](#) [Clone or download](#)

[kelseyhightower](#) add Docker support 7 Latest commit ed6c73f on Feb 6, 2018

<a href="#">CONTRIBUTING.md</a>	add no code	a year ago
<a href="#">Dockerfile</a>	add Docker support	a year ago
<a href="#">LICENSE</a>	add no code	a year ago
<a href="#">README.md</a>	add windows support	a year ago

[README.md](#)

## No Code

No code is the best way to write secure and reliable applications. Write nothing; deploy nowhere.

## Getting Started

Start by not writing any code.

<https://github.com/kelseyhightower/nocode>