



Introduction to Firebase

Sam Zhou & Jessica Hong



What is Firebase?

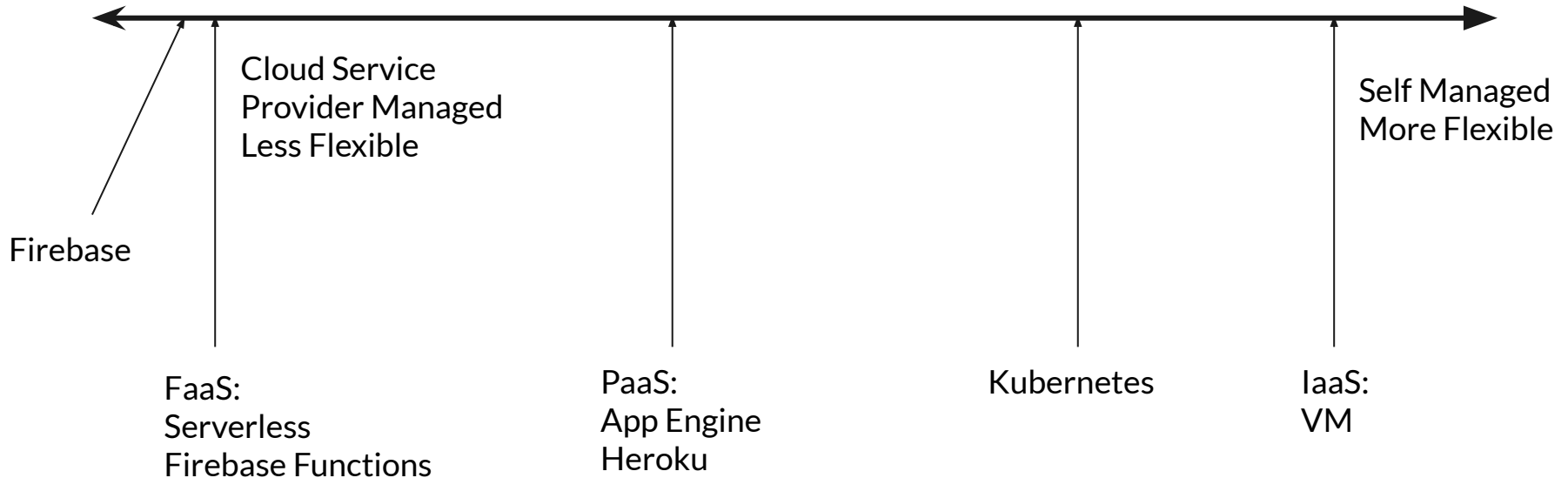


Serverless Computing

- No worrying about underlying infrastructure and hardware choices
- Scale Automatically
- Save \$\$\$











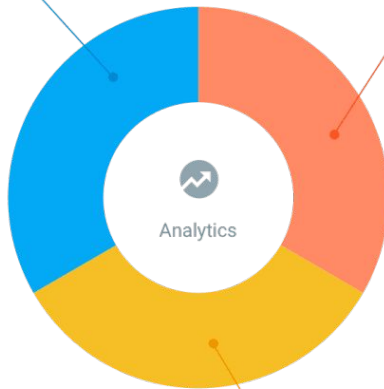
Various Options In BaaS Spectrum








Firebase Features

DEVELOP

-  Realtime Database
-  Authentication
-  Cloud Messaging
-  Storage
-  Hosting
-  Remote Config
-  Test Lab
-  Crash Reporting



GROW

-  Notifications
-  App Indexing
-  Dynamic Links
-  Invites
-  AdWords

EARN



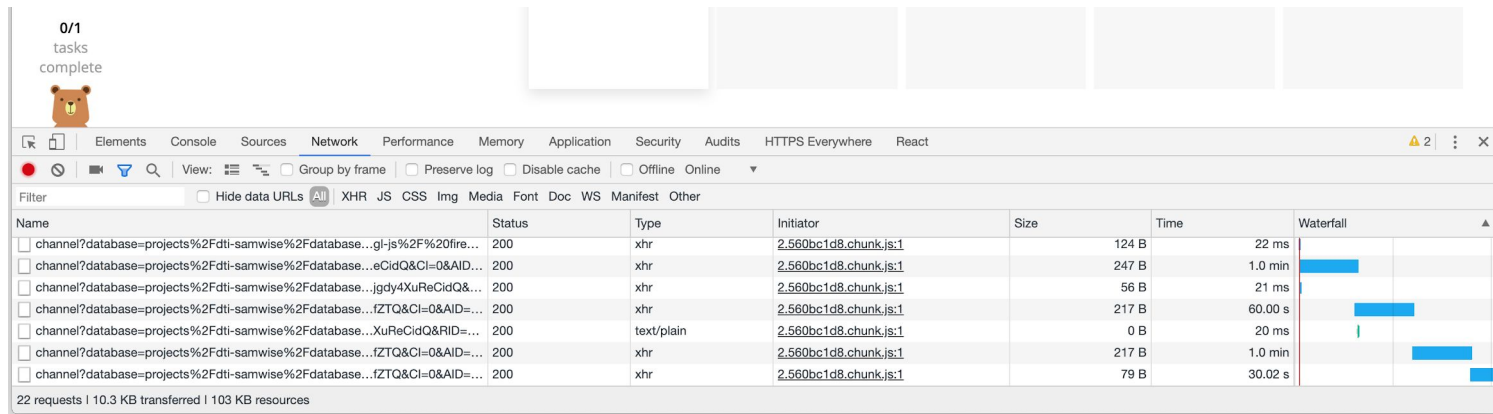


Why Samwise Switched

Demo Time!

Real Time Capabilities

- Real time capabilities out of box
 - Long pulling



Traditional Approach: ID Provisioning

```
function addTask(task) { latency!  
  fetch(`/task/add`, ...) →  
    .then(taskId => {  
      // ...  
      // ...  
    });  
}  
  
app.post(`/task/add`, (req, resp) => {  
  db.add(req.body)  
    .then(id => resp.send(id));  
}); latency!  
  
Without taskId from server, cannot call editTask  
  
function editTask(taskId, change) {  
  fetch(`/task/edit/${taskId}`, ...) →  
  // ...  
}
```

Without taskId from server, cannot call editTask

```
function editTask(taskId, change) {  
  fetch(`/task/edit/${taskId}`, ...) →  
  // ...  
}
```




Realtime Updates: Bad Approach

```
setInterval (() => {  
  fetch('/tasks').then(resp => { ... });  
}, 100);
```

Very Inefficient!



Fronted DB Operations

```
133 export const removeTask = (task: Task, noUndo?: 'no-undo'): void => {
134   const { subTasks } = store.getState();
135   const deletedSubTasks = task.children
136     .map((id) => {
137       const subTask = subTasks.get(id);
138       return subTask == null ? error('corrupted!') : subTask;
139     })
140     .toArray();
141   const batch = db().batch();
142   batch.delete(tasksCollection().doc(task.id));
143   task.children.forEach(id => batch.delete(subTasksCollection().doc(id)));
144   batch.commit().then(() => {
145     if (noUndo !== 'no-undo') {
146       const { children, ...rest } = task;
147       const fullTask = { ...rest, children: deletedSubTasks };
148       emitUndoRemoveTaskToast(fullTask);
149     }
150   });
151 };
```

```
95 const unmountTasksListener = listenTasksChange(ownerEmail, (snapshot) => {
96   const created: Task[] = [];
97   const edited: Task[] = [];
98   const deleted: string[] = [];
99   snapshot.docChanges().forEach((change) => {
100     const { doc } = change;
101     const { id } = doc;
102     if (change.type === 'removed') {
103       deleted.push(id);
104     } else {
105       const data = doc.data();
106       if (data === undefined) {
107         return;
108       }
109       const { owner, date: timestamp, children, ...rest } = data as FirestoreTask;
110       const task: Task = {
111         id,
112         date: timestamp instanceof Date ? timestamp : timestamp.toDate(),
113         children: Set(children),
114         ...rest,
115       };
116       if (change.type === 'added') {
117         created.push(task);
118       } else {
119         edited.push(task);
120       }
121     }
122   });
123   store.dispatch(patchTasks(created, edited, deleted));
124   firstTasksFetched = true;
125   reportFirstFetchedIfAllFetched();
126 });
---
```



Frontend DB Operations

Firestore Migration #101

[Edit](#)

mt-xing merged 30 commits into `master` from `firebase-migration` on Feb 17



Conversation 1



Commits 30



Checks 0



Files changed 68

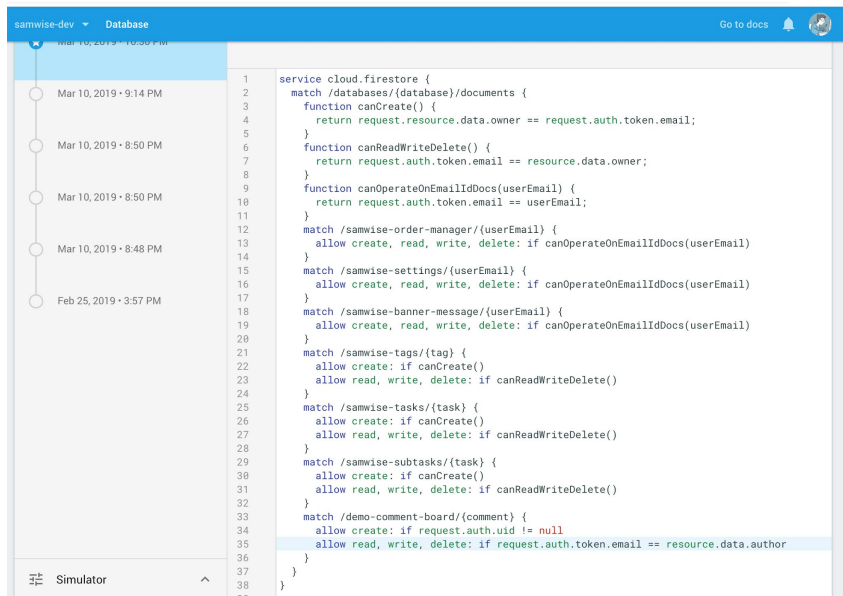
+1,120 -2,055 



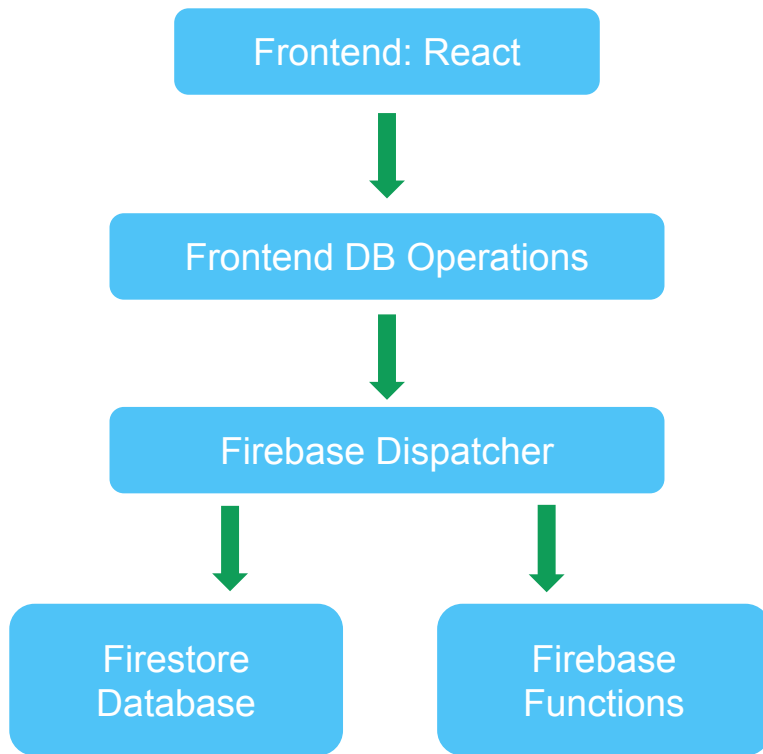
What about Security?

How We Use Firebase

- Authentication
- Security - Firestore Configs
- Firestore
- Firebase Functions



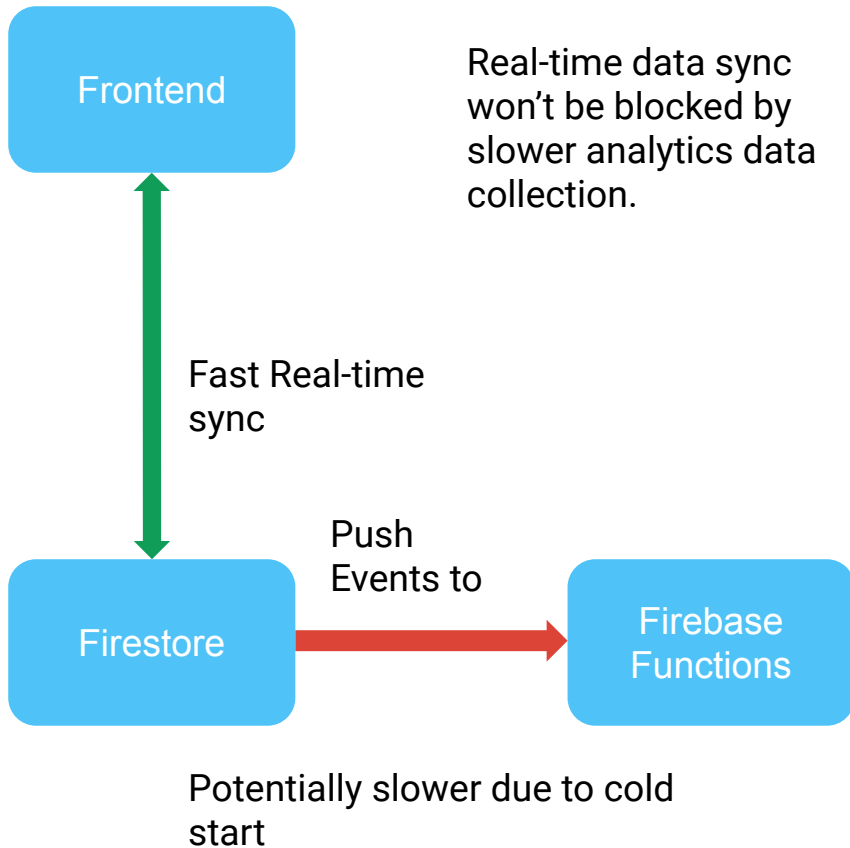
```
1 service cloud.firestore {
2   match /databases/{database}/documents {
3     function canCreate() {
4       return request.resource.data.owner == request.auth.token.email;
5     }
6     function canReadWriteDelete() {
7       return request.auth.token.email == resource.data.owner;
8     }
9     function canOperateOnEmailIdDocs(userEmail) {
10      return request.auth.token.email == userEmail;
11    }
12    match /samwise-order-manager/{userEmail} {
13      allow create, read, write, delete: if canOperateOnEmailIdDocs(userEmail)
14    }
15    match /samwise-settings/{userEmail} {
16      allow create, read, write, delete: if canOperateOnEmailIdDocs(userEmail)
17    }
18    match /samwise-banner-message/{userEmail} {
19      allow create, read, write, delete: if canOperateOnEmailIdDocs(userEmail)
20    }
21    match /samwise-tags/{tag} {
22      allow create: if canCreate()
23      allow read, write, delete: if canReadWriteDelete()
24    }
25    match /samwise-tasks/{task} {
26      allow create: if canCreate()
27      allow read, write, delete: if canReadWriteDelete()
28    }
29    match /samwise-subtasks/{task} {
30      allow create: if canCreate()
31      allow read, write, delete: if canReadWriteDelete()
32    }
33    match /demo-comment-board/{comment} {
34      allow create: if request.auth.uid != null
35      allow read, write, delete: if request.auth.token.email == resource.data.author
36    }
37  }
38 }
```





Problem and Solutions:

Cold Starting





Q&A

<http://tinyurl.com/dev-sesh-15>